Министерство образования Белгородской области
Областное государственное автономное
профессиональное образовательное учреждение
«Белгородский индустриальный колледж»

Рассмотрено
предметно-цикловой комиссией
Протокол заседания № ____
От «____»_____ 2022
Председатель цикловой комиссии
_____/ Третьяк И.Ю.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

МДК 09.01 «Проектирование и разработка веб-приложений»

ПМ.09 «Проектирование, разработка и оптимизация веб-приложений»

по специальности

09.02.07 Информационные системы и программирование

Разработчик:

Солдатенко М.Н.
преподаватель ОГАПОУ
«Белгородский индустриальный
колледж»

Белгород 2022

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

МДК 09.01 Проектирование и разработка веб-приложений является специальным, формирующим базовые умения для получения выпускником профессиональных умений.

Методические рекомендации по выполнению лабораторных работ для студентов специальности 09.02.07 «Информационные системы и программирование» разработаны в соответствии с рабочей программой профессионального модуля **Проектирование, разработка и оптимизация веб-приложений**, соответствуют требованиям Федерального государственного образовательного стандарта по специальностям среднего профессионального образования.

Целью методических рекомендаций по выполнению лабораторных работ является организация и управление работой студентов на лабораторных занятиях при изучении данного междисциплинарного курса.

Методические рекомендации по выполнению лабораторных работ содержат тематический план и общие положения и требования к оформлению отчетов. Методические указания к каждой лабораторной работе включают в себя следующие элементы: название темы, цель занятия, ход работы, теоретическую часть, практическую часть (указания по выполнению) и контрольные вопросы.

Методические рекомендации содержат лабораторные работы, которые обеспечивают формирование базовых умений и навыков разработки веб-приложений с использованием современных стандартов; разработки веб-приложений в соответствии с требованиями заказчика, знаний норм и правил выбора стилистических решений; современных методик разработки веб-приложений.

В лабораторных работах, приведенных в данных методических рекомендациях, содержатся как задания с подробными указаниями к выполнению, так и задания без алгоритма работы.

Методические рекомендации предназначены для студентов очной формы обучения специальности 09.02.07 «Информационные системы и программирование». По учебному плану по МДК 09.01 Проектирование и разработка веб-приложений на лабораторные работы студентов отводится 78 часов.

Методические рекомендации направлены на повышение мотивации обучающихся к изучению междисциплинарного курса Проектирование и разработка интерфейсов пользователя, развитие гибкого логического и пространственного мышления обучающихся , развитие профессиональных компетенций учащейся молодежи.

Тематический план

№	Название лабораторной работы	Количество часов
1.	Составление анкеты и сбор материала для конкретной задачи веб-приложения. Описание задачи на языке UML	2
2.	Моделирование процесса разработки информационного ресурса CASE- средствами	2
3.	Изучение ГОСТ 19.201-78 в программе Консультант Плюс и составление технического задания на разработку веб-проекта	2
4.	Построение сетевого графика разработки веб-проекта в MS Project	2
5.	Создание серверных сценариев с использованием операторов PHP	2
6.	Программирование базовых конструкций на PHP, обработка строк.	2
7.	Программирование ветвлений и циклов на PHP.	2
8.	Программирование массивов на PHP.	2
9.	Работа с датой и временем.	2
10.	Разработка серверных сценариев для работы с функциями.	2
11.	Разработка серверных сценариев для работы с классами и объектами.	2
12.	Обработка данных, введенных в форму	2
13.	Реализация загрузки файлов. Организация файлового ввода-вывода	2
14.	Программирование опросов и счѐтчиков посещения web страницы	2
15.	Создание базы данных MySQL с помощью утилиты phpMyAdmin	2
16.	Извлечение информации из базы данных на PHP	2
17.	Организация поддержки базы данных в PHP	2
18.	Отслеживание сеансов (session)	2
19.	Создание проекта «Регистрация»	2
20.	Создание проекта «Форум»	2
21.	Создание проекта «Чат»	2

22.	Создание новостной ленты	2
23.	Создание проекта «Интернет магазин»	2
24.	Разработка блога на PHP с администрированием статей (архитектура MVC).	2
25.	Составление схем XML-документов	2
26.	Отображение XML-документов различными способами	2
27.	Разработка Web-приложения с помощью XML	2
28.	Разработка меню web-страницы на HTML5+CSS3+PHP	2
29.	Использование JavaScript для доступа и управления HTML DOM объектов	2
30.	Применение технологии AJAX	2
31.	Многоуровневое меню, многоуровневые списки в AJAX	2
32.	Реализация поиска и быстрого поиска в AJAX	2
33.	Использование библиотеки jQuery. Создание выпадающего списка	2
34.	Создание фотогалереи на JQuery	2
35.	Программирование сложных структур с использованием JSON	2
36.	Использование фреймворка для создания сайта	2
37.	Создание landing page с использованием фреймворка Bootstrap	2
38.	Создание сайта на CMS	2
39.	Администрирование сайта	1
40.	Публикация сайта на бесплатном хостинге	1
	Всего:	**78**

Лабораторная работа № 1

Составление анкеты и сбор материала для конкретной задачи веб-приложения. Описание задачи на языке UML

Цель: рассмотреть правила составления анкеты и сбора материала для конкретной задачи веб-приложения, а также описания задачи на языке UML

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

С помощью простого моделирования и строгого подхода к программированию можно сделать процесс разработки веб-приложения гораздо более гладким и гарантировать, что созданную web-систему будет просто поддерживать в дальнейшем.

UML (Unified Modeling Language) - унифицированный язык моделирования - это язык, с помощью которого описываются системы. Следовательно, этот язык может замечательно помочь вам описать и отобразить вашу будущую систему.

Стадия планирования.

С самого начала требуется составить план, чтобы гарантировать принятие правильного решения. Когда вы работаете в группе, состоящей из нескольких человек, ясное понимание плана движения вперед и четкое распределение работы становятся просто неоценимыми. Постарайтесь основательно разобраться в следующих аспектах системы на стадии планирования:

- Кто будет пользователями системы, и каковы будут роли этих пользователей
- Требования приложения
- Взаиморасположение страниц и порядок перемещения между ними
- Инструменты и технологии, которые будут использоваться на сайте

Знайте своих пользователей

Важно знать, какие типы пользователей будут работать с вашей системой. И не только потому, что для анализа системы вам придется беседовать с представителями этих пользователей (анкеты, письма, личные беседы и т.д.), но и потому, что часто вам придется моделировать систему так, чтобы в ней работали пользователи с различными ролями и привилегиями. Классифицируя пользователей и изучая их требования, вы узнаете такие сведения, которые скажут вам, как надо строить базу данных, какие ограничения обеспечить, как сгруппировать страницы, как организовать обучение и помощь, какая специфическая информация должна быть на сайте.

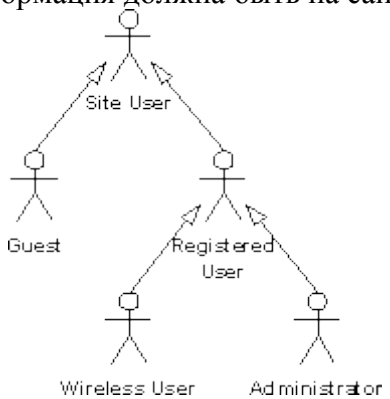


Рис. 1: Иерархия исполнитель-роль

На представленном выше рисунке изображены несколько групп пользователей (называемые на языке UML "исполнителями" (actors), которые будут работать с web-

сайтом. В данном случае самый общий тип пользователя ("Пользователь сайта" - Site user) расположен вверху диаграммы. Сплошная стрелка обозначает отношение обобщения, то есть у нас имеются две более специфические категории пользователей: посетители-гости и зарегистрированные пользователи. Характеристики, которые будут общими у обеих групп пользователей, будут приписаны исполнителю "пользователь сайта", а привилегии, специфичные для посетителей-гостей и зарегистрированных пользователей, будут приписаны более мелкой соответствующей роли. Помимо прочего пользователи делятся на посетителей, подключившихся к сайту по беспроводной связи, и администраторов. Обе эти группы – дальнейшее подразделение группы зарегистрированных пользователей, в системе у них будут различные функции.

Определите требования

Уточните, что именно вы собираетесь построить, прежде чем начнете работу. Часто нерентабельно писать подробную спецификацию требований к сайту, но вы обязаны хотя бы приблизительно набросать пару диаграмм и написать пару строк о том, какие функции будет выполнять сайт. Вот именно здесь и приходят на помощь блочные диаграммы. Рассматривайте блоки-действия как пакет функций - как нечто, что обычно соответствует странице сайта, приложению или действию, осуществляемому на сайте (например: проверка пароля посетителя, изменение настроек пользователя или удаление устаревших пользователей). Следующий рисунок представляет собой краткую диаграмму блоков-действий, с помощью которой планируется web-сайт. На данной диаграмме не показаны все действия, предусмотренные в нашем приложении - обычно для описания всей функциональности будущего сайта создается несколько диаграмм.

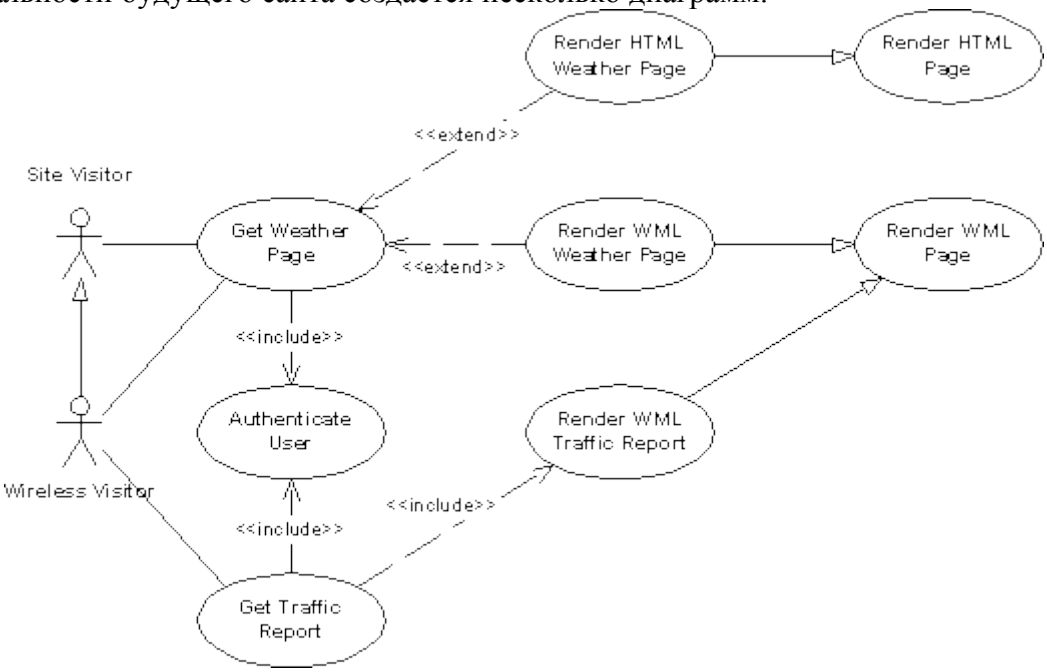


Рис.2: Диаграмма блоков-действий

Даже с помощью такой простой диаграммы как эта, можно запросто описать огромный объем информации. Например, отношение включения ("include"), показывает, что два определенных действия включают в себя одну и туже функцию аутентификации пользователя. Отношение расширения ("extend") указывает, что страница с информацией о погоде может выдаваться как в HTML так и в WML-формате. Отношение обобщения ("generalization") показывает, что для выдачи конкретных страниц будет использовать более общая процедура, под названием "выдача HTML-страницы" или "выдача WML-страницы" с целью обеспечить единство внешнего вида и поведения всех страниц на web-сайте.

Также диаграмма показывает, что посетители, подключенные к сайту по беспроводной связи, будут иметь доступ к тем его разделам, которые не будут видны для

других посетителей. В данной диаграмме только эта группа пользователей может посмотреть отчеты о дорожной обстановке. Мы посчитали, что отчеты о дорожной обстановке понадобятся лишь людям, находящимся в пути, и потому нам нет нужды прилагать усилия по генерации страниц в каких-то иных форматах, кроме формата WML. Следовательно, действие "Выдать отчет о дорожной обстановке" не нужно расширять (extend) вариантами WML или HTML страниц, оно может напрямую включить (include) действие "Создать WML-страницу".

Как правило, все эти диаграммы и блоки-действия сопровождаются какими-то краткими описаниями. Есть несколько статей, где обсуждается то, как создаются диаграммы с блоками-действиями и как пишутся к ним объяснения. Если вкратце, то вы обычно описываете, что должно произойти внутри блока-действия, кто этим действие будет пользоваться, как оно будет вызываться, как останавливаться, и какие варианты результатов действия могут получиться.

Спланируйте страницы

Во время работы над блоками-действиями у вас сформируются некоторые представления по поводу того, каких и сколько страниц понадобится создавать для сайта. Возможно, еще до начала работы у вас будут хорошие идеи о некоторых из страниц, однако с помощью блочных диаграмм вы сможете взглянуть на проблему с другой точки зрения. Нужно ли посетителю сайта столько много страниц? Не слишком ли много функций возложено на эту страницу? Трудно ли перемещаться по сайту? Скажем, трудно ли попасть с первой страницы на одну из часто выполняемых функций? Найдите ответы на все эти вопросы в блочных диаграммах, т.е. до того, как вы начнете делать наброски своих страниц и создавать прототип сайта.

После того, как вырисуется в общих чертах диаграмма блоков-действий, можно приступать к грубой прикидке структуры сайта. Для этого можно воспользоваться языком UML, например, использовать классовые диаграммы.

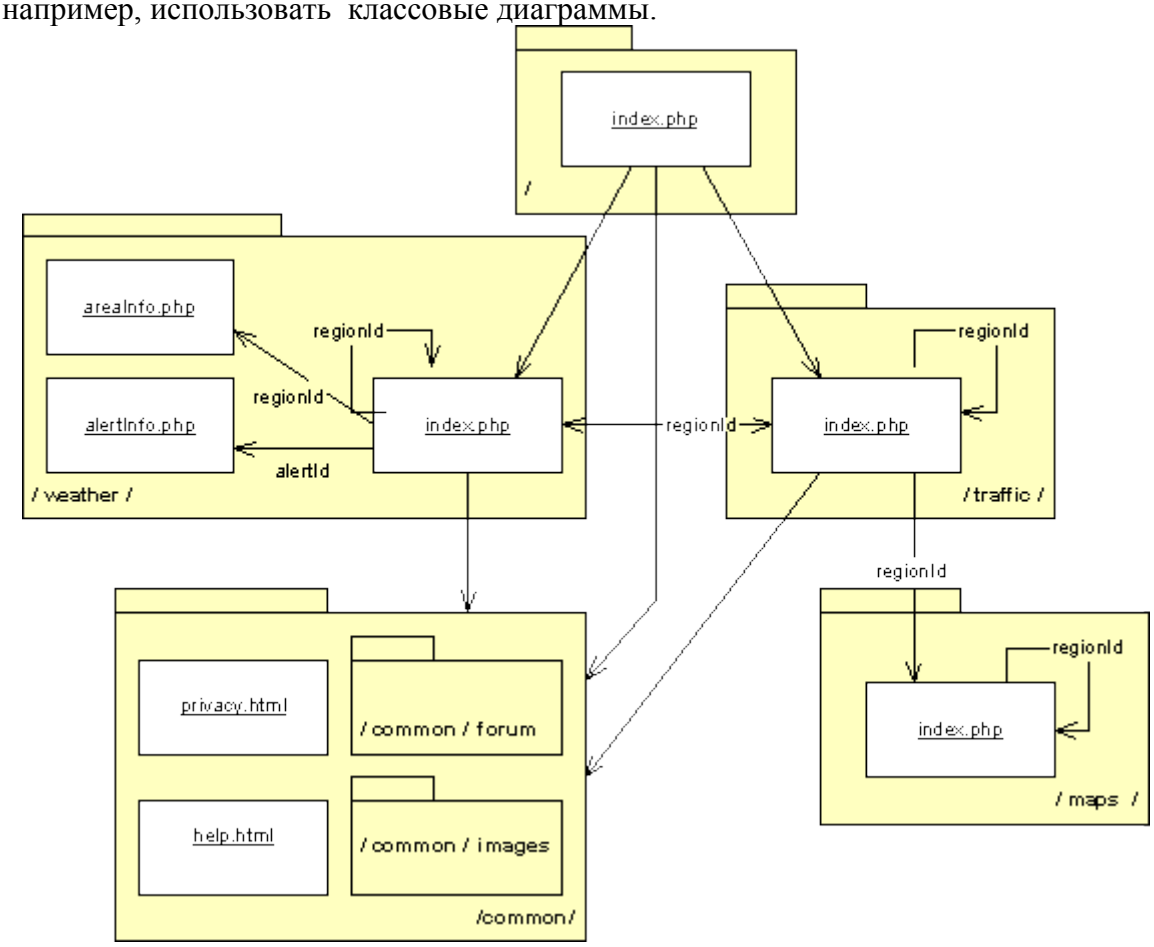


Рис. 3: Диаграмма страниц

На данной диаграмме все функции сайта были распределены на несколько областей:

- / - корень сайта
- /common/ - общие функции, изображения, скрипты, таблицы стилей и так далее
- /maps/ - изображения карт и направлений
- /traffic/ - отчеты о дорожной обстановке
- /weather/ - отчеты о погоде

Данный рисунок также показывает, какие параметры передаются между страницами. Переменная regionId - самая главная, так как она обозначает район, в котором заинтересован пользователь (будь то код округа, города или провинции). С помощью переменной regionId вы передаете информацию о регионе между страницами, таким образом пользователь сможет перейти от отчета о погоде к отчету о дорожной обстановке в том же самом регионе. Что касается каталога common, то здесь вы видите, что область состоит из целого пакета (package), а не из отдельных файлов. Это просто прием укрощения хаоса в диаграмме, так как пакеты будут взаимно использовать большинство, если не все, файлы, расположенные в каталоге /common/.

Данная диаграмма очень полезна для планирования сайта, так как она помогает избежать путаницы. Кроме того она служит вам шаблоном, которым вы будете пользоваться при создании других web-сайтов с такой же структурой.

Выберите инструменты

Для маленьких сайтов выбрать инструменты и технологии достаточно просто. В особенности в тех случаях, когда важную роль играет стоимость проекта, возможны лишь несколько комбинаций - Apache, MySQL или PostgreSQL, PHP или Perl или JSP/сервлеты.. На следующем рисунке показана примерная компонентная диаграмма, с помощью которой описывается архитектура сайта. Данная простая диаграмма подойдет для описания многих web-сайтов, работающих сейчас в Сети. Следовательно, ее не придется каждый раз воссоздавать заново, так как в ней нет какой-либо интересной, специфической или уникальной информации.

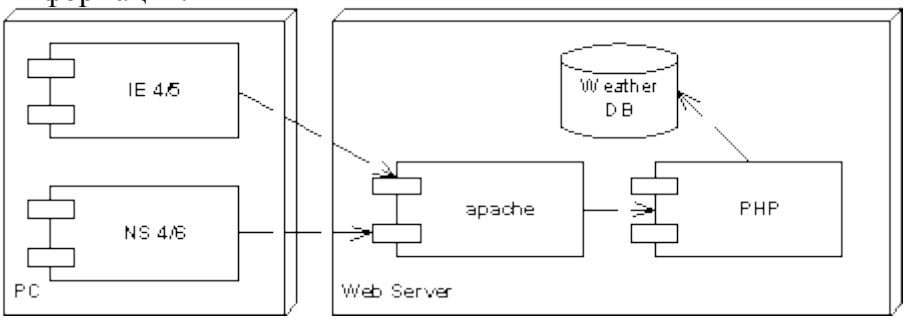


Рис. 4: Диаграмма архитектуры

Этап дизайна

Этап дизайна должен накладываться на этап анализа - уже в то время, когда вы узнаете все больше и больше подробностей о будущей системе, вы начинаете делать наброски того, как она будет построена. Невозможно сначала на 100% проанализировать систему, а затем прямо перейти к ее дизайну. Часто требования к системе развиваются, а иногда и ваша идея построения системы может послужить толчком к развитию требований к ней (и наоборот). Несколько минут потраченные на моделирование системы с помощью диаграмм в конце окупят с лихвой.

Дизайн на будущее

Большинство разработчиков проводит большую часть времени за отлаживанием и переделкой кода чем, за его написанием. В особенности это справедливо, когда вам приходится работать над несколькими web-сайтами сразу. Хорошие дизайнерские находки можно переносить с одного сайта на другой в части структуры, организации и построения кода. Однако если вы наворотите код без предварительного планирования, преимущества его использования в перспективе сойдут на нет. Очень эффективным

способом визуального изображения сайта является построение нескольких диаграмм классов. Приведенные рисунки показывают несколько ключевых отношений, используемых в диаграммах классов.

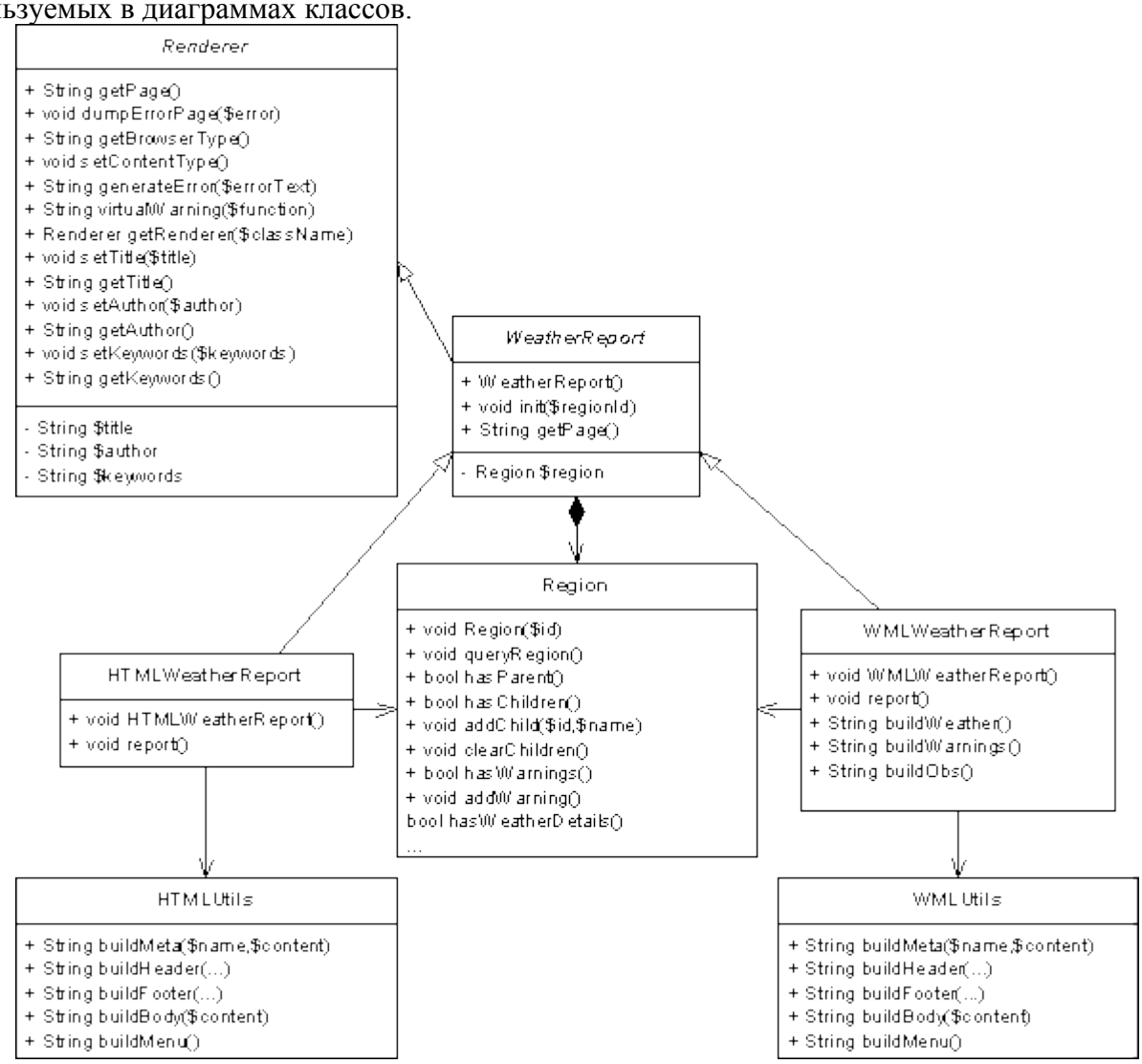


Рис. 5: Диаграмма классов

Диаграммы, подобные приведенной выше, можно построить очень быстро. Эта диаграмма, например, изображает сайт.

- Класс Renderer является абстрактным классом - его имя выделено курсивом. Это значит, что он не будет использоваться напрямую, лишь его потомки-подклассы будут инициализированы (например Region()). Все страницы, выводящие информацию на экран, должны быть подклассами этого класса, чтобы обеспечить задачу вывода информации для любого типа броузера.
- Класс WeatherReport отвечает за создание и владение всех объектов Region. На схеме это отношение изображено с помощью черного ромба, обозначающего сильное отношение группировки (aggregation). Это значит, что один объект владеет другими объектами и создает их.
- Знак "+" перед названиями методов обозначает, что эти методы имеют тип public, и они могут быть вызваны из других объектов или функций. Знак "-" обозначает, что переменная или метод имеют тип private, и они видимы только функциям, принадлежащим данному объекту. В PHP все методы и переменны имеют тип public, но мы должны всегда рассматривать эти переменные как private-переменные, и не должны к ним обращаться напрямую.

- Класс HTMLWeatherReport зависит от класса HTMLUtils. Отношение зависимости (dependency) обозначает, что один класс инициализирует и/или вызывает функции другого класса.
- В каждом классе диаграммы классов должны быть указаны все методы (и переменные. Если таковые имеются), их видимость (тип public, private или protected), тип значения, возвращаемого каждой функцией, аргументы-параметры этих функций, и также типы данных переменных. Функции указываются первыми, а за ними в отдельном блоке перечисляются переменные.

Если система, которую вы строите, не является объектно-ориентированной, диаграммы классов все равно могут помочь вам создать модель приложения. Классы вместо объектов просто будут обозначать различные включаемые файлы и функции. В этом случае на вашей диаграмме будут отсутствовать отношения наследования, слияния (composition), группировки (aggregation) и прочие отношения из ООП.

Моделирование работы системы

Иногда необходимо показать, как куски все составные части системы будут работать вместе в процессе. Диаграмма классов показывает все отношения между классами, но не показывает порядок, в котором делаются вызовы и то, как результат выполнения одной функции определяет, какая функция будет вызываться следующей. Для того, чтобы показать более динамические аспекты системы, язык UML предлагает ряд диаграмм различного типа. Диаграммы сценариев (Scenario diagrams) особенно полезны, если вы разрабатываете модель веб-сайта. Диаграммы сценариев делятся на два вида: диаграммы взаимодействия (collaboration diagrams) и диаграммы последовательностей (sequence diagrams). Как правило, не приходится моделировать все действия, происходящие в системе. Обычно, диаграммы сценариев служат для отображения самых сложных частей системы, либо для общего схематического изображения работы кода. Например, с помощью этой диаграммы можно показать, как код проверки пользователя вставляется в определенную страницу, либо показать, как набор утилит служит в различных страницах для создания единого интерфейса сайта. Два типа диаграмм сценариев приведены ниже на рисунках.

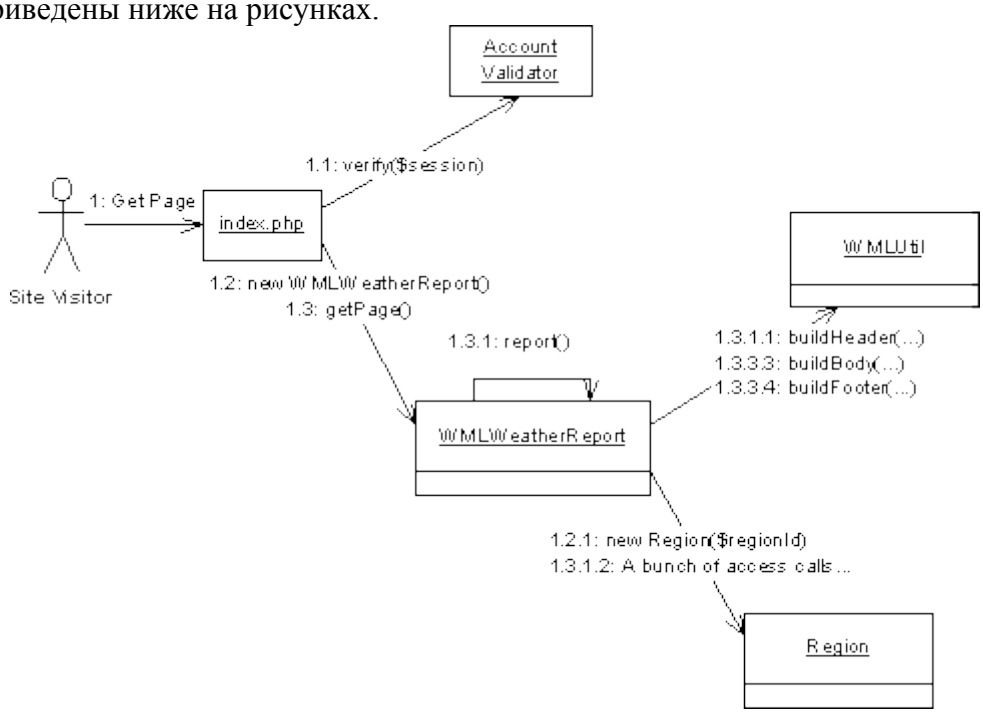


Рис. 6. Диаграмма взаимодействия.

Данная диаграмма взаимодействия показывает дизайн того, как на веб-сайте будет создаваться отчет о погоде. Вы сами можете проследить выполнение методов от "1" до "1.3.3.4". в некоторых командах принято нумеровать диаграммы как 1, 2, 3, 4, но вообще-то лучше показывать глубину вызовов используя следующую нотацию: 1, 1.1, 1.2, 2, 2.1 и

так далее. эта схема нумерации показывает систему передачи вызовов более наглядно. Например, диаграмма показывает, что метод report() вызывает несколько функций в объектах WMLUtil и Region. Функция buildHeader(...) в WMLUtil вызывается до того, как мы начали создавать отчет с помощью других функций. Наконец, мы вызываем метод buildFooter(...) в модуле WMLUtil до того, как возвращаемся к методу report() и из него - к методу getPage(). В диаграммы взаимодействия можно добавлять более подробную информацию, например, тип возвращаемых данных, ограничения на данные и прочие условия.

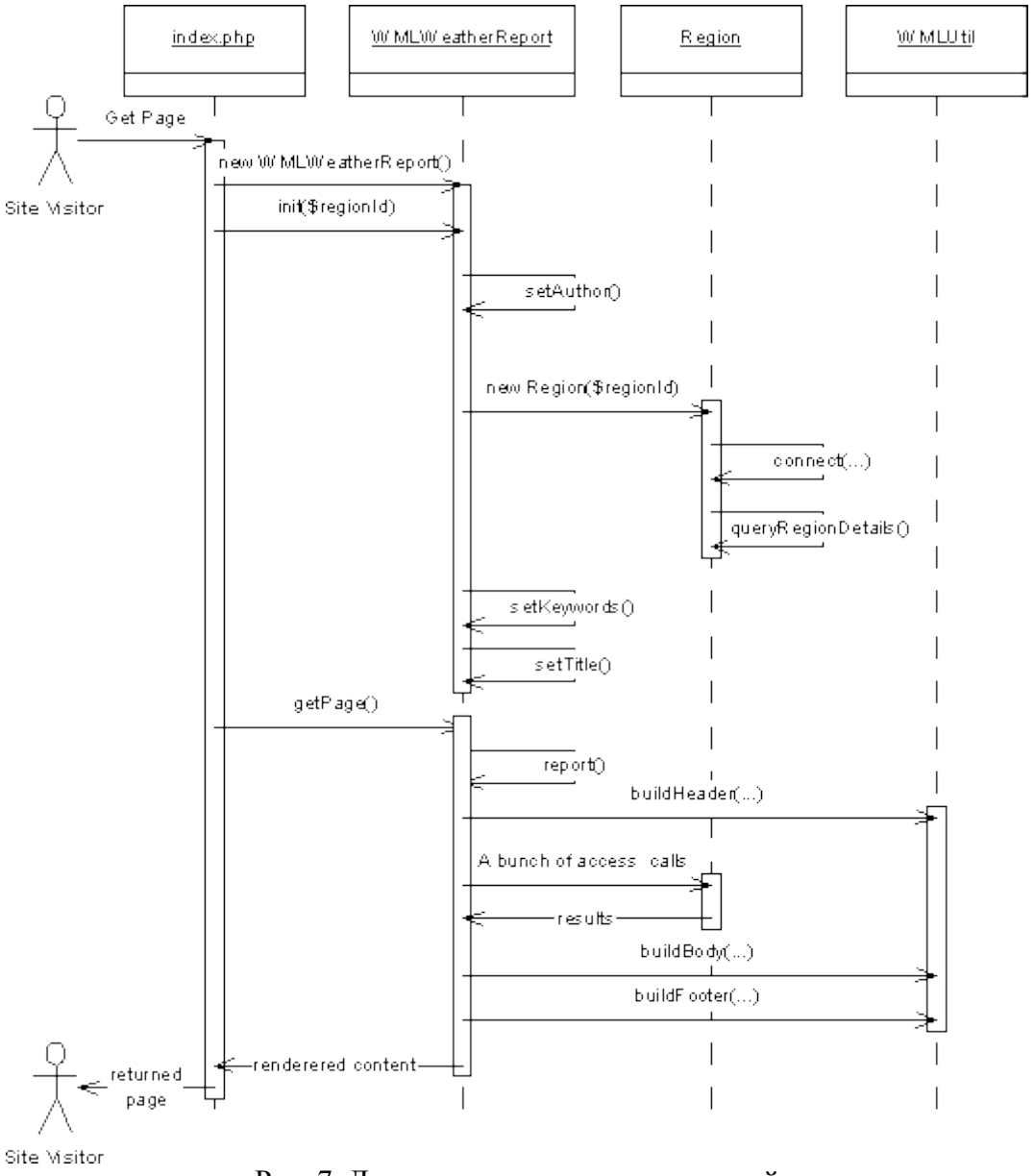


Рис. 7. Диаграмма последовательностей

Диаграмма последовательностей очень похожа на диаграмму взаимодействий по тому, какую информацию они показывают. Вообще-то, многие программы UML способны создавать диаграммы последовательностей из диаграмм взаимодействий и обратно. Главное отличие диаграммы последовательностей в том, что на этой диаграмме легче проследить последовательность действий. Кроме того, на этой диаграмме можно указать подробную информацию о времени существования того или иного объекта или о его поведении (например, время ожидания, взаимодействие параллельных нитей процесса, момент создания и уничтожения объектов).

Когда возникает вопрос, какую диаграмму строить, используйте критерии, которые помогут выбрать наиболее подходящий вариант:

- Используйте диаграмму последовательностей, если вы хотите показать детали кода, относящиеся к временным и межпроцессорным взаимодействиям
- Используйте диаграмму взаимодействий, если вы пытаетесь показать цепь взаимодействий между объектами
- Используйте диаграмму последовательностей, если вы хотите показать одновременные взаимодействия между несколькими объектами
- Используйте диаграмму взаимодействий, если вы хотите показать большое количество взаимодействий или передачи сообщений между малым количеством объектов.

План развертывания приложения

Диаграммы развертывания системы показывают архитектуру и организацию файлов. Что касается организации файлов, можно работать с инструментами моделирования классов или использовать диаграмму компонентов, но в зависимости от сложности сайта вы можете сами решать, нужна она или нет. Следующий ниже пример намеренно использует больше машин (узлов), чем наш простой реальный пример, который вполне уместился бы на одной машине, ну разве что в будущем его понадобилось бы промасштабировать для обслуживания более массовой аудитории.

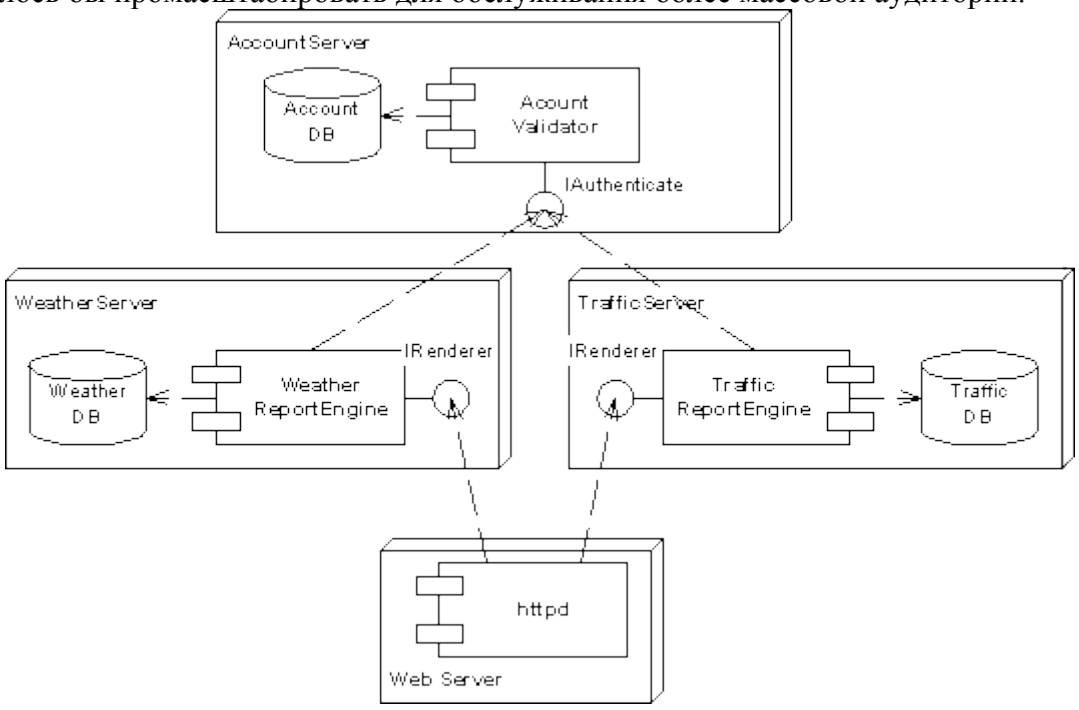


Рис.8: Диаграмма компонентов

UML обеспечивает поддержку всех этапов проектирования сайта и предоставляет для этих целей ряд графических средств - диаграмм.

На этапе создания концептуальной модели для описания бизнес-деятельности используются модели бизнес-прецедентов и диаграммы видов деятельности, для описания бизнес-объектов - модели бизнес-объектов и диаграммы последовательностей.

На этапе создания логической модели сайта описание требований к системе задается в виде модели и описания системных прецедентов, а предварительное проектирование осуществляется с использованием диаграмм классов, диаграмм последовательностей и диаграмм состояний.

На этапе создания физической модели детальное проектирование выполняется с использованием диаграмм классов, диаграмм компонентов, диаграмм развертывания.

Ниже приводятся определения и описывается назначение перечисленных диаграмм и моделей применительно к задачам проектирования ИС.

- диаграмма классов - статическая структурная диаграмма, описывающая структуру системы, она демонстрирует классы системы, их атрибуты, методы и зависимости между классами;
- диаграмма компонентов - статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонент могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т.п.;
- диаграмма развертывания - служит для моделирования работающих узлов (аппаратных средств) и артефактов, развернутых на них;
- диаграмма объектов - демонстрирует полный или частичный снимок моделируемой системы в заданный момент времени. На диаграмме объектов отображаются экземпляры классов (объекты) системы с указанием текущих значений их атрибутов и связей между объектами;
- диаграмма деятельности - диаграмма, на которой показано разложение некоторой деятельности на ее составные части. Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений;
- диаграмма состояний - диаграмма, на которой представлен конечный автомат с простыми состояниями, переходами и композитными состояниями;
- диаграмма вариантов использования - диаграмма, на которой отражены отношения, существующие между актерами и вариантами использования;
- диаграмма последовательности - диаграмма, на которой изображено упорядоченное во времени взаимодействие объектов. В частности, на ней изображаются участвующие во взаимодействии объекты и последовательность сообщений, которыми они обмениваются;
- диаграмма сотрудничества - этот тип диаграмм позволяет описать взаимодействия объектов, абстрагируясь от последовательности передачи сообщений. На этом типе диаграмм в компактном виде отражаются все принимаемые и передаваемые сообщения конкретного объекта и типы этих сообщений.

Язык UML представляет собой общецелевой язык визуального моделирования, который разработан для спецификации, визуализации, проектирования и документирования компонентов программного обеспечения, бизнес-процессов и других систем. Язык UML является достаточно строгим и мощным средством моделирования, которое может быть эффективно использовано для построения концептуальных, логических и графических моделей сложных систем различного целевого назначения. Этот язык вобрал в себя наилучшие качества и опыт методов программной инженерии, которые с успехом использовались на протяжении последних лет при моделировании больших и сложных систем.

Преимущества UML:

- UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных ОО-языках;
- UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;
- диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом;
- UML расширяет и позволяет вводить собственные текстовые и графические стереотипы, что способствует его применению не только в сфере программной инженерии;
- UML получил широкое распространение и динамично развивается.

Практическая часть

Задание. Опираясь на рассмотренные выше примеры, **с**оставить анкету и собрать материал для конкретной задачи веб-приложения, а также описать задачу веб-приложения на языке UML

Содержание отчета

1. Цель.
2. Виды диаграмм UML, используемы при проектировании сайта.
3. Выводы

Контрольные вопросы

1. Назначение UML.
2. Преимущества UML
3. Перечислите виды диаграмм языка UML.
4. Перечислите этапы моделирования процесса разработки информационного ресурса CASE- средствами.
5. Охарактеризуйте этапы моделирования процесса разработки информационного ресурса.

Лабораторная работа № 2

Моделирование процесса разработки информационного ресурса CASE- средствами

Цель: реализовать моделирование процесса разработки информационного ресурса CASE- средствами

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

 Теоретический материал:

Планирование сайта с помощью UML

Проектирование сайта с помощью диаграмм UML

Исходя из опыта взаимодействия с Заказчиком, мы вывели две основные успешные стратегии разработки сайтов. Каждая стратегия подходит для своего конкретного случая.

Надо ли проектировать сайт?

1. В первом случае Заказчик обращается к вам примерно с таким запросом: «Посмотрите на сайт, например, *http://www.planerka.info*. Делаем все точно так же, только меняем дизайн».

Мы делаем копию сайта, собираем админку для сайта, чтобы было удобно управлять данными сайта.

На следующем этапе Заказчик обращается к нам с запросом дооснастить сайт набором дополнительных модулей – другая платежная система для сайта, свой модуль отзывов или, например, личный кабинет для сайта.

2. Во втором случае у Заказчика есть своя оригинальная идея или концепция сайта. Обычно она оформлена в виде бумажных набросков. Иногда дело даже доходит до готового дизайна в макетах PSD.

Это уже значительный прогресс. Действительно, лучше один раз показать, чем десять раз рассказать.

И в первом и во втором случае проектирование сайта обязательно!

Это в разы облегчит взаимодействие между Заказчиком и Исполнителем. Более того, разработка сложного оригинального программного продукта в принципе невозможна без проектной документации. Другое дело, что Заказчик не всегда посвящен во все подробности разработки проектной документации к сайту. Заказчику для эксплуатации сайта, часть проектных документов просто не нужна, а необходима только Разработчику для ведения работ. **Надо отметить, что в момент передачи готового проекта Заказчику, Разработчик сайта передает проектную документацию к сайту в полном объеме, как часть проекта.**

Правилом хорошего тона считается, когда ссылки на проектную документацию сайта доступны из администраторской зоны сайта.

Какие инструменты проектирования использовать в разработке сайта? Диаграммы UML!

Важно чтобы Заказчик и Разработчик говорили на одном языке. Поэтому проектная документация должна быть выполнена с использованием универсального, широко распространенного стандарта. Таким стандартом в проектировании, в том числе сайтов, стал **UML – Unaited Modelling Language.**

В состав языка проектирования UML входит широкий набор видов диаграмм:

диаграммы классов,

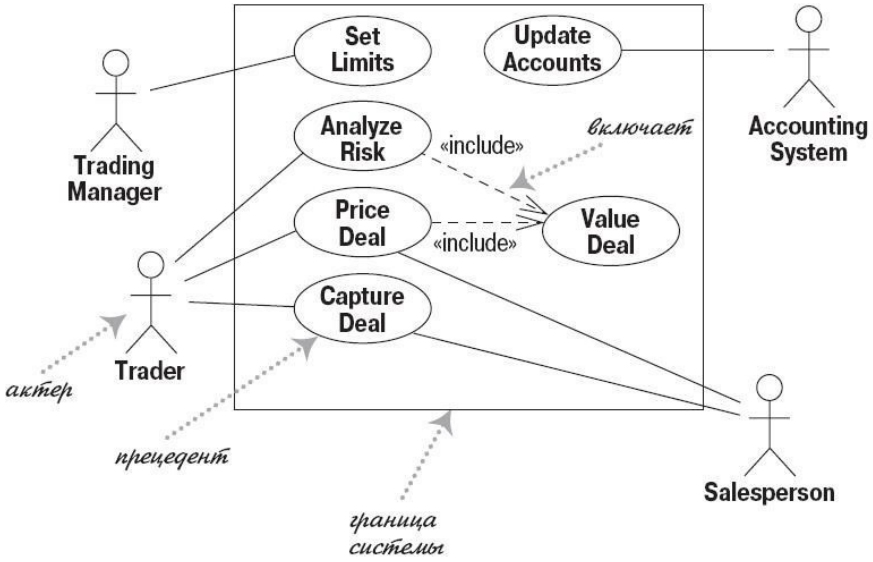
диаграммы последовательностей,

диаграмма объектов,

диаграмма компонентов,
диаграмма пакетов,
диаграмма развёртывания,
диаграмма вариантов использования (диаграмма прецедентов),
диаграмма автомата,
диаграмма деятельности,
диаграмма коммуникаций,
диаграмма времени (временнАя диаграмма).

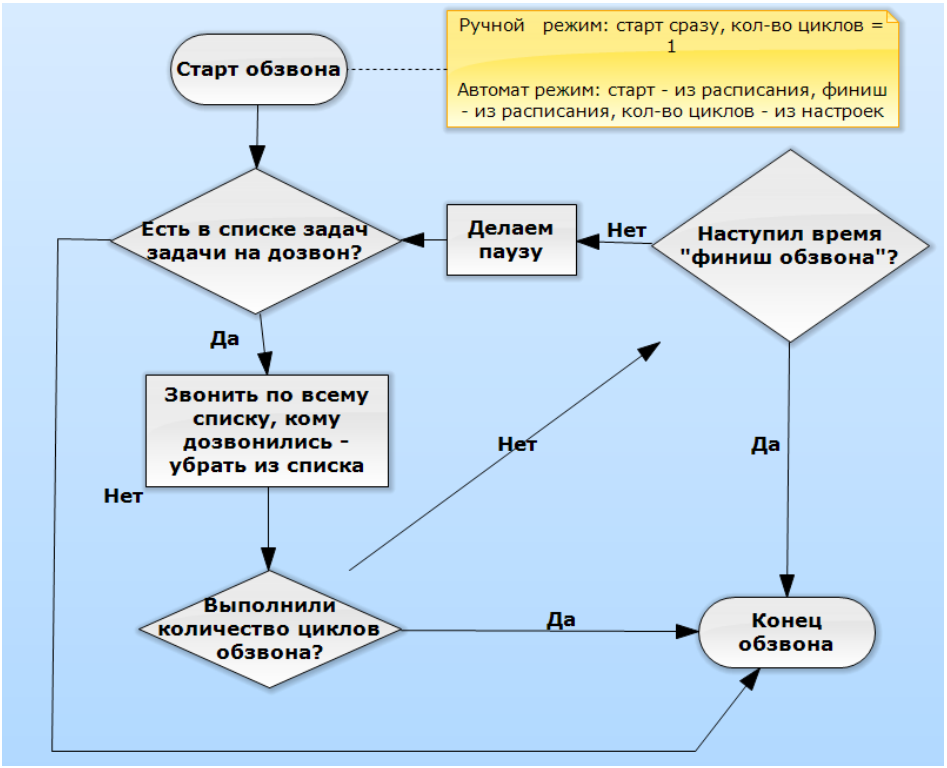
На практике, для проектного документирования сайта достаточно использовать всего несколько из них.

На самом начальном этапе, когда надо прояснить идею веб-приложения и понять видение клиента, лучше всего подойдет диаграмма вариантов использования (диаграмма прецедентов). Немного модифицированный вид этих диаграмм используется в приложении Microsoft Solutions Foundation, предназначенный для организации полного цикла работ по разработке программного обеспечения.

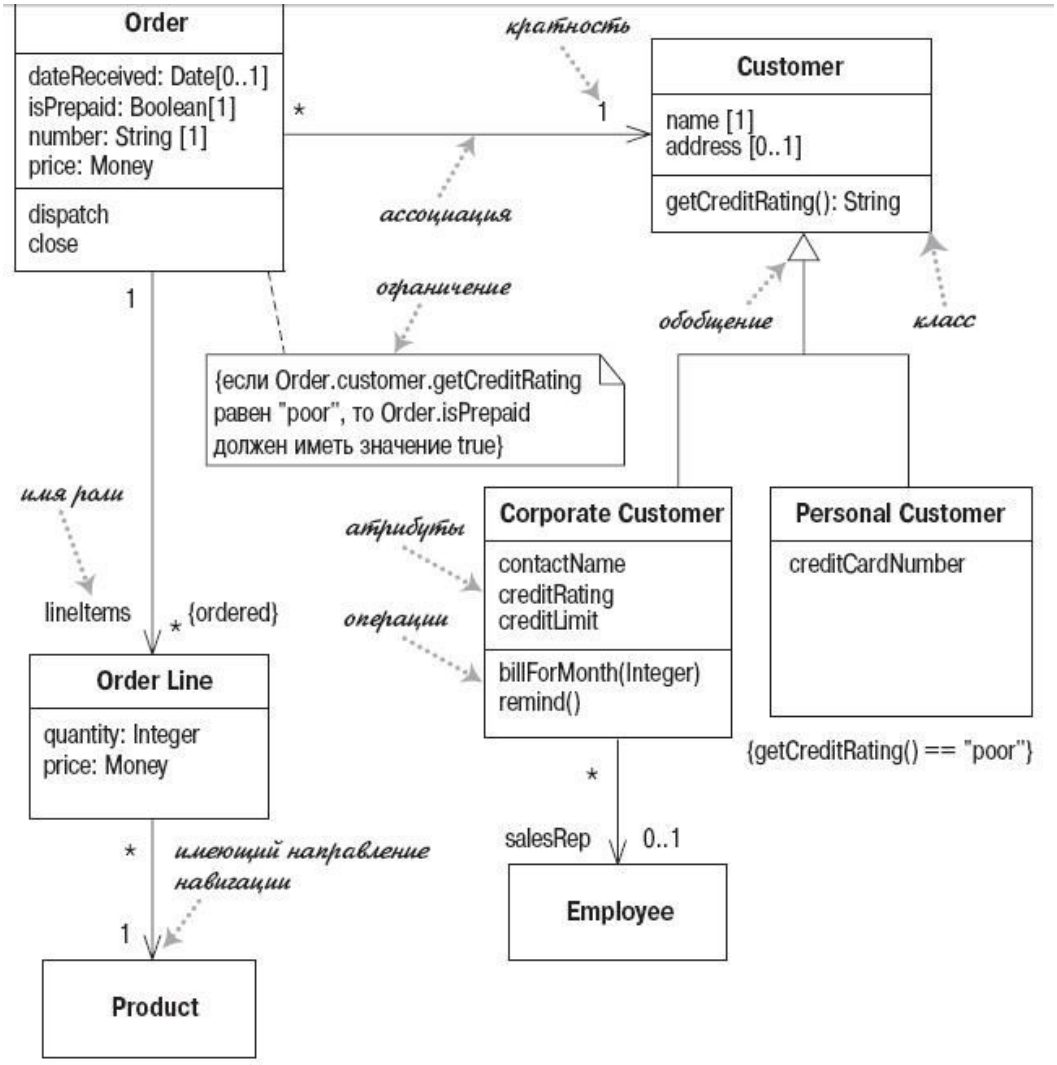


Диаграммы ВИС обычно сопровождаются текстовым словесным описанием.

Еще один очень удобный вид диаграмм UML – диаграмма деятельности. Мы рекомендуем использовать более простой, понятный и знакомый нам еще по школьным урокам информатики язык блок-схем.



Наконец, когда от прояснения и документирования концепции сайта вы переходите к проектированию структуры баз данных, как разработчику вам может пригодиться диаграмма классов:



Основное предназначение диаграммы классов UML - описание сущности в проектной системе. Также диаграммы классов UML показывают отношения между сущностями и реализацию родительских классов. После составления диаграммы классов становится понятным, как реализовывать связь между таблицами данных и кодом проекта в каждом конкретном случае.

Диаграмма последовательности UML полезна для того, чтобы описать последовательность переключения инициативы между несколькими участниками технологического процесса.

Здесь мы описали основные диаграммы UML, обычно используемые при разработке сайта.

А вот когда работы над вашим сайтом закончены и проект принят, пришло время размещать ваш сайт на рабочем (или как говорят «боевом») сервере. Чтобы помнить, где что лежит, на каких аппаратных ресурсах располагается ваша веб-система – понадобится UML диаграмма развертывания.

Даже беглого взгляда на картинки-иллюстрации, того как используются диаграммы, достаточно чтобы увидеть однозначную ценность диаграмм UML для организации работ по разработке сайта.

Итак, диаграммы UML используются на всех этапах разработки сайта: формулирования концепции, разработки структуры базы данных, описании бизнес-процессов и развертывания сайта на рабочем хостинге.

Разработчику знание диаграмм UML обязательно и для чтения и в первую очередь для составления проектной документации.

Практическая часть

Задание. Выполнить моделирование процесса разработки информационного ресурса в соответствии с вариантом в программе StarUML или средствами онлайн-сервиса draw.io

Содержание отчета

4. Цель.
5. Этапы моделирования процесса разработки информационного ресурса CASE-средствами.
6. Выводы

Контрольные вопросы

6. Перечислите известные CASE- средства для моделирования информационных ресурсов.
7. Назначение UML.
8. Перечислите виды диаграмм языка UML.
9. Перечислите этапы моделирования процесса разработки информационного ресурса CASE- средствами.
10. Охарактеризуйте этапы моделирования процесса разработки информационного ресурса.

Лабораторная работа № 3
Изучение ГОСТ 19.201-78 в программе Консультант Плюс и составление технического задания на разработку веб-проекта

Цель: изучить ГОСТ 19.201-78 в программе Консультант Плюс и составить техническое задание на разработку веб-проекта

Ход работы:

7. Изучить теоретический материал.
8. Проработать примеры.
9. Выполнить задания в соответствии с указаниями.
10. Предъявить преподавателю результаты работы.
11. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
12. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Техническое задание — это документ, в котором зафиксированы требования к сайту. Чем четче и подробнее расписаны эти требования, тем лучше все участники процесса понимают, каким он должен быть. А значит, растет шанс того, что все останутся довольны результатом.

Главная цель технического задания: убедиться, что клиент и исполнитель правильно поняли друг друга.

Пользы от технического задания много. Для каждой стороны она своя.

Польза для клиента:

- **Понять, за что он платит деньги, и каким будет сайт.** Можно сразу увидеть структуру, понять, что и как будет работать. Прикинуть, все ли устраивает. Если нет — без проблем поменять еще до начала разработки.
- **Увидеть компетентность исполнителя.** Если техзадание понятное и четкое — доверие к разработчику повышается. Если там написана каша — возможно, стоит бежать и не оглядываться.
- **Застраховаться от недобросовестности исполнителя.** Когда сайт готов, его можно проверить по техническому заданию. Есть несоответствия? Разработчик обязан их исправить. Если вы сотрудничаете официально и заключали договор — можно даже принудить через суд.
- **Упростить замену исполнителей.** Если клиент и разработчик повздорили и разбежались, создание сайта может сильно затянуться. Когда есть подробное техзадание, его можно передать новой команде — она втянется в работу в разы быстрее.
- **Узнать стоимость разработки сложного продукта.** Оценить точные сроки и стоимость разработки сложного веб-сервиса сходу нельзя. Сначала нужно понять, как будет работать сервис, и какие в нем будут функции. Для этого и нужно подготовить техзадание.

Польза для исполнителя:

- **Понять, что хочет заказчик.** Клиенту задают десятки вопросов, показывают примеры, предлагают решения. Затем записывают все в единый документ и согласовывают. Если все окей — ура, вы поняли правильно.
- **Застраховаться от внезапных хотелок клиента.** Иногда попадаются заказчики, которые хотят поменять задачу на полпути. Если вы согласовали и подписали ТЗ, вам не страшно подобное. В случае чего, даже суд будет на вашей стороне.
- **Показать свою компетентность.** Классно подготовленное техзадание покажет клиенту экспертность разработчиков. Если компания сомневалась, доверять ли вам разработку сайта, сомнения с большой вероятностью развеются.
- **Заработать деньги.** Некоторые студии и разработчики предлагают составление ТЗ как отдельную услугу.

- **Облегчить и ускорить процесс разработки**. В хорошем ТЗ указаны структура сайта, необходимые функции и элементы на каждой странице. Когда все требования уже перед глазами — остается только задизайнить и написать код.

Техзадание составляет исполнитель. Хорошее ТЗ всегда составляет исполнитель: проект-менеджер или разработчик. Очевидно, что веб-разработчик понимает в создании сайтов больше, чем владелец кафе или стоматологической клиники. Поэтому описывать проект придется ему.

Это не значит, что клиент исчезает и появляется в самом конце, чтобы написать: «Збс, одобряю». Он тоже должен участвовать в процессе:

- Познакомить исполнителя с компанией, продуктами и целевой аудиторией.
- Объяснить, зачем ему сайт.
- Рассказать, что он хочет, поделиться идеями.
- Показать примеры хороших с его точки зрения сайтов.
- Ответить на любые другие вопросы исполнителя.

Конечно, заказчик может набросать свой вариант ТЗ. Возможно, это ускорит процесс создания конечного техзадания. А возможно, получится мусор, который втихаря выкинут на помойку.

Пишите однозначно и точно

Этот совет вытекает из главной цели техзадания — «Убедиться, что клиент и исполнитель правильно поняли друг друга».

В техническом задании не должно быть качественных прилагательных: красивый, надежный, современный. Их нельзя однозначно понять. У каждого свои понятия красоты и современности.

Проверяйте, нет ли в тексте неоднозначностей. Если есть — перепишите. Ваши формулировки должны быть четкими и точными:

- Сайт должен загружаться быстро → Любая страница сайта должна иметь больше 80 баллов в Google PageSpeed Insights.
- Большие нагрузки → 50 тысяч посетителей одновременно.
- На главной странице выводится список статей → На главной странице выводится список последних 6 опубликованных статей.
- Минималистичный удобный интерфейс подписки → Поле «Оставьте e-mail» и кнопка «Подписаться» → *нарисованный эскиз*.

Укажите общую информацию

Все члены команды должны правильно понимать, чем занимается компания и кто ее целевая аудитория. Чтобы никто не запутался, это лучше прописать в самом начале техзадания.

А еще стоит указать цель сайта и описать его функционал в двух словах — чтобы не получить интернет-магазин вместо блога.

Поясните сложные термины

Первое правило техзадания — оно должно быть понятно всем, для кого предназначено. Если вы собираетесь использовать термины, которые может не понять ваша клиентка — владелица магазина детских игрушек — обязательно поясните их. Понятным языком, а не копипастой из «Википедии».

Напишите понятные пояснения в инфостиле

Опишите инструменты и требования к хостингу

Представьте, что вы 2 месяца делали крутой сайт. Каждый этап согласовывали с клиентом — он в восторге. И вот пришло время сдавать работу. Вы показываете админку, а клиент кричит: «Это что такое? Модэкс?! Я думал, вы сделаете на «Вордпрессе»!»

Чтобы таких проблем не было, опишите используемые инструменты, движки и библиотеки. Заодно укажите требования к хостингу. Мало ли, вы сделаете на PHP — а у клиента сервер на .NET.

Перечислите требования к работе сайта

Сайт должен работать во всех браузерах актуальных версий и на всех типах устройств. Да, это очевидно для любого разработчика и любого заказчика. Но лучше написать, чтобы защитить клиента от недобросовестно выполненной работы.

Клиент защищен от сайта без мобильной версии

Сюда же напишите требования к скорости загрузки сайта, устойчивости к нагрузкам, защите от хакерских атак и подобным вещам.

Укажите структуру сайта

До начала отрисовки дизайна и верстки вам нужно согласовать с клиентом структуру сайта.

Пообщайтесь с заказчиком, выясните, что ему надо. Соберите разработчиков, сеошников, маркетологов, главреда — и решите, какие страницы нужны на сайте. Подумайте, как они будут связаны между собой, с какой на какую можно перейти.

Можно показать структуру списком, можно нарисовать блок-схему. Как вам удобнее.

Структура сайта-визитки в XMind

Это один из важнейших этапов работы над сайтом. Структура — это фундамент. Если она неудачная — сайт получится кривой.

Объясните, что будет на каждой странице

Клиент должен понять, зачем нужна каждая страница и какие элементы на ней будут. Есть два способа это показать.

Прототип — более наглядный и однозначный способ. Исполнитель рисует эскизы каждой страницы и прилагает их к техзаданию. Клиент видит, как будет выглядеть интерфейс его будущего сайта и говорит, что ему нравится, а что стоит изменить.

Перечисление элементов — ленивая альтернатива прототипу. Просто напишите, какие блоки должны быть на странице, и что они делают.

По такому описанию можно представить, как будет выглядеть блог

Распишите сценарии использования сайта

Если вы делаете какой-то нестандартный интерфейс, просто показать структуру и эскизы страниц недостаточно. Важно, чтобы вся команда исполнителей и клиент поняли, как посетители будут пользоваться сайтом. Для этого отлично подходят сценарии. Схема сценария очень простая:

- Действие пользователя.
- Ответное действие сайта.
- …
- Результат.

Конечно, если вы делаете стандартную визитку или лендинг, писать сценарии не нужно. Но если на сайте будут какие-то интерактивные сервисы — очень желательно.

Определите, кто отвечает за контент

Одни разработчики делают сайт сразу с контентом. Другие ставят рыбу. Третьи могут написать тексты, но за дополнительную плату. Договоритесь об этом на берегу и зафиксируйте в техзадании, какой контент вы должны подготовить.

Распишите, кто за какой контент отвечает

Придумать объективные критерии оценки качества текстов довольно сложно. Лучше не пишите ничего, чем «Качественный, интересный и продающий контент, полезный для целевой аудитории». Это мусор, он никому не нужен.

Указать, что весь контент должен быть уникальный, — это полезно. Еще одна защита клиента от недобросовестных исполнителей.

Опишите дизайн (если сможете)

Как и в случае с текстом, объективные критерии оценки дизайна придумать сложно. Если вы с клиентом договорились о цветовой гамме — напишите ее. Если у него есть брендбук, в котором прописаны шрифты, — укажите и их.

Требования к контенту и дизайну в техзадании, которое составлял я для своих клиентов

Вместо вывода: структура техзадания

Для разных задач структура ТЗ будет своя. Глупо делать одинаковые технические задания для новой социальной сети и лендинга по оптовой продаже моркови. Но в целом вам нужны такие разделы:

1. Информация о компании и целевой аудитории, цели и задачи сайта.
2. Глоссарий терминов, которые могут быть непонятны клиенту.
3. Технические требования к верстке и работе сайта.
4. Описание используемых технологий и список требований к хостингу.
5. Подробная структура сайта.
6. Прототипы страниц или описания элементов, которые должны на них быть.
7. Сценарии использования нестандартного интерфейса (опционально).
8. Список контента, который делает разработчик.
9. Требования к дизайну (опционально).

Практическая часть

1. Внимательно изучите **ГОСТ 19.201-78 Единая система программной документации (ЕСПД).** Техническое задание. Требования к содержанию и оформлению (с Изменением N 1)
   (http://docs.cntd.ru/document/1200007648)
   ГОСТ 19.201-78 - Настоящий стандарт устанавливает порядок построения и оформления технического задания на разработку программы или программного изделия для вычислительных машин, комплексов и систем независимо от их назначения и области применения.
2. Разработайте техническое задание на создание веб-проекта в соответствии с вариантом, основываясь на ГОСТе и нижеприведенной структуре.

   2Структура технического задания:
   1. Термины, используемые в техническом задании
   2. Общие положения
   2.1. Название сайта
   2.2. Наименование предприятий (объединений) разработчика и заказчика (пользователя) сайта и их реквизиты
   2.3. Перечень документов, на основании которых создается сайт
   2.4. Состав и содержание работ по созданию системы
   2.5. Порядок оформления и предъявления заказчику результатов работ по созданию сайта
   3. Назначение и цели создания сайта
   3.1. Цели создания сайта

3.2. Задачи, решаемые при помощи сайта
4. Требования к сайту и программному обеспечению
4.1. Требования к программному обеспечению сайта
4.2. Общие требования к оформлению и верстке страниц
4.3. Требования к численности и квалификации персонала обслуживающего сайт
4.4. Требования к системе администрирования
5. Структура сайта
6. Языковые версии сайта
7. Группы пользователей
8. Дизайн сайта
9. Навигация по сайту
9.1. Основное навигационное меню
9.2. Дополнительная навигация по сайту
10. Описание страниц сайта
10.1. Описание статических страниц
10.2. Описание динамических страниц
11. Функционал сайта
12. Контент и наполнение сайта
12.1. Формат предоставления материалов для сайта
13. Дополнительная информация
14. Порядок контроля и приемки работ
15. Реквизиты и подписи сторон

Содержание отчета

7. Цель.
8. Техническое задание оформленное в соответствии с вариантом.
9. Выводы

Контрольные вопросы

11. Перечислите типы связей, которыми могут быть связаны задачи проекта.
12. Если некоторая задача может быть начата только через некоторое время после завершения другой (например, из технологических соображений), то как этот факт отображается в модели проекта?
13. Если некоторая задача должна начинаться несколько позже, чем закончится другая задача проекта, то как этот факт отразить в модели проекта?
14. Если длительность проекта такова, что диаграмма Ганта не помещается в пределах экрана, то что и как надо сделать, чтобы вся диаграмма Ганта была видна одновременно?
15. Перечислите характеристики ресурса.
16. Чем определяется стоимость задачи проекта?
17. Чем определяется стоимость проекта в целом?

Лабораторная работа № 4

Построение сетевого графика разработки веб-проекта в MS Project

Цель: изучить возможности программного продукта Microsoft Project, предназначенного для управления проектами

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

1. Основные сведения о программе

Для поддержки процесса управления проектом на различных этапах существует большое количество программных комплексов, целью применения которых является повышение эффективности реализации проекта, т. е. выполнение как всего проекта в целом, так и его отдельных этапов в заданные сроки и в рамках утвержденных денежных ресурсов. Основными задачами менеджера проекта является составление сетевого графика (расписания) проекта, распределение ресурсов между задачами проекта и слежение за ходом реализации проекта.

Внедрение компьютерных технологий в практику реализации проекта может оказать существенную помощь в эффективной реализации проектов.

Анализ существующих программных комплексов поддержки деятельности по управлению проектами показал, что для малых проектов, в которых удельный вес этапа реализации значителен, предпочтительным является пакет программ Microsoft Project (MS Project). Он позволяет эффективно выполнить структуризацию проекта путем разделения его на этапы и подзадачи, выявить критические задачи (задачи, длительность которых существенно влияет на длительность реализации всего проекта), получить сетевой график проекта, осуществить назначение ресурсов задачам проекта, контролировать загрузку ресурсов.

Microsoft Project позволяет эффективно управлять проектом на различных этапах его реализации. Он дает возможность выполнить структуризацию проекта путем разделения его на этапы, задачи и подзадачи, выявить критические задачи (задачи, длительность которых существенно влияет на длительность реализации всего проекта), получить сетевой график и календарный план проекта, осуществить назначение ресурсов задачам проекта, эффективно контролировать загрузку ресурсов. Пакет поддерживает все необходимые типы связей между задачами: FS (Finish-Start), SS (Start-Start), FF (Finish-Finish).

Поддерживая современные информационные технологии, пакет MS Project позволяет импортировать данные из файлов, созданных в среде других приложений, например MS Excel и MS Access. Неоспоримым достоинством пакета является наличие встроенного языка программирования Visual Basic For Application, что обеспечивает возможность разработки программных компонент, обеспечивающих решение специфических задач.

2. Построение модели

Методика использования пакета Microsoft Project для управления инновационным проектом на этапе подготовки к реализации, целью которой является получение сетевого графика и календарного плана проекта, может быть представлена в виде последовательности следующих шагов:

- создание календаря проекта (т. е. учет нерабочих и праздничных дней);

- составление списка задач, которые надо выполнить для успешной реализации проекта;
- определение связей между задачами;
- выявление задач, длительность реализации которых существенно влияет на длительность реализации всего проекта, и, возможно, изменение порядка выполнения задач проекта;
- формирование списка доступных для реализации проекта ресурсов;
- распределение ресурсов (назначение ресурсов конкретным задачам проекта).

2.1. Начало работы над проектом

Цель работы: умение использовать пакет Microsoft Project для создания календаря проекта и составления списка задач проекта.

Календарь

Одной из задач, решаемых на этапе подготовки к реализации проекта, является получение сетевого графика проекта. При построении сетевого графика MS Project использует календарь — таблицу, в которой отражены рабочие и нерабочие (выходные и праздничные) дни. Различают стандартный календарь и календари пользователя. Стандартный календарь предполагает, что рабочими днями являются все дни недели, за исключением субботы и воскресенья, и рабочий день длится 8 часов. Календарь пользователя учитывает реальные рабочие дни. Например, очевидно, что в России реальный календарь в январе и мае существенно отличается от стандартного.

При работе с проектом MS Project позволяет, помимо базового календаря, использовать несколько календарей пользователя. Можно, например, создать календарь для всего проекта в целом (основной календарь) и календарь для отдельного ресурса, который будет учитывать особенности его использования (например, недоступность ресурса в определенные дни или месяцы).

Задачи проекта

Задача – это некоторая деятельность, которую надо выполнить, чтобы завершить часть проекта. Работа над проектом начинается с составления списка задач проекта.

Большие, сложные задачи, как правило, могут быть естественным образом представлены в виде набора боле простых, более конкретных задач. Поэтому при составлении списка сначала записывают общую (обобщенную) задачу, затем – задачи, из которых эта общая задача состоит (подчиненные задачи).

При составлении списка задач проекта обычно используют метод, который часто называют «сверху — вниз». Суть метода заключается в том, что сначала составляют список главных (обобщенных) задач, затем этот список уточняется посредством добавления уточняющих задач.

Каждая задача проекта характеризуется длительностью, которая измеряется в минутах, часах, днях и неделях. Длительность обобщенной задачи определяется длительностью ее подчиненных задач. Длительность обобщенной задачи вычисляет MS Project.

Длительность подчиненной задачи нижнего уровня, т. е. задачи? у которой нет подчиненных задач, определяется временем необходимым для ее выполнения одной единицей ресурса. Например, один рабочий копает траншею в 5 метров 8 часов. Если для реализации проекта необходимо выкопать траншею длинной в 10 метров, то длительность задачи «Траншея» равна 16 часам. Длительность подчиненной задачи задает менеджер проекта на основе нормативной документации.

Контрольные точки проекта

В каждом проекте могут быть выделены этапы, после выполнения которых процесс реализации проекта переходит на новый качественный уровень. Например, после этапа согласования требований с заказчиком начинается этап подготовки технической документации. Можно считать, что последней задачей этапа согласования требований с заказчиком является задача утверждения требований. Подобные задачи, как правило,

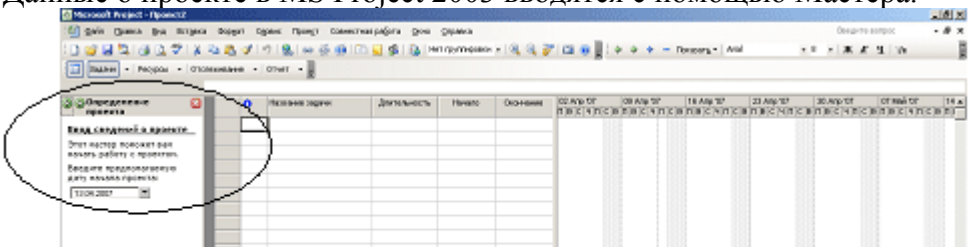
должны выполняться в конкретный, заранее установленный день, поэтому их называют контрольными точками. Длительность задач – контрольных точек – принимают равной нулю.

Задание 1:

1. Запустите Microsoft Project.
2. Создайте календарь проекта.
3. Установите предполагаемую дату начала реализации проекта, задайте название проекта.
4. Используя таблицу ввода списка задач, введите названия задач проекта. Для подчиненных задач нижнего уровня задайте длительность. (***Замечание*:** На первом этапе работы над проектом, когда задачи проекта представлены в виде простого списка, длительность обобщенной задачи равна длительности самой длительной подчиненной задачи. Позже, когда будут установлены связи между задачами, длительность обобщенной задачи будет вычислена как сумма длительностей подчиненных задач.)
5. Если выполнение какой-либо задачи должно начаться в определенный день, то введите дату начала выполнения этой задачи.
6. Сохраните проект в своем рабочем каталоге.

Пример выполнения

Данные о проекте в MS Project 2003 вводятся с помощью Мастера.



Определение опорных дат проекта

Опорными данными для каждого проекта являются даты начала и окончания проекта. Одна из них (любая) вводится разработчиком; другая в дальнейшем вычисляется автоматически. ***Установить начальную дату проекта: 19 апреля.***

Определение общих рабочих часов для проекта (Создание календаря)

Project содержит несколько шаблонов календаря, которые можно использовать как основу для календаря проекта.

Используя подсказки Мастера, установить:

Рабочие дни: с понедельника по субботу.

Время работы: с 8.30 до 17.00; в субботу — с 8.30 до 15.00.

Задать праздничные и выходные дни.

Проанализировать изменения графика рабочего времени.

Дополнительные календари (они связаны с ресурсами проекта) пока не определять.

Ввод задач проекта

Перечень задач может быть введен непосредственно в программе MS Project или перенесен из других программ (Копировать/Вставить).

Ввести следующие задачи (проект строительства дома):

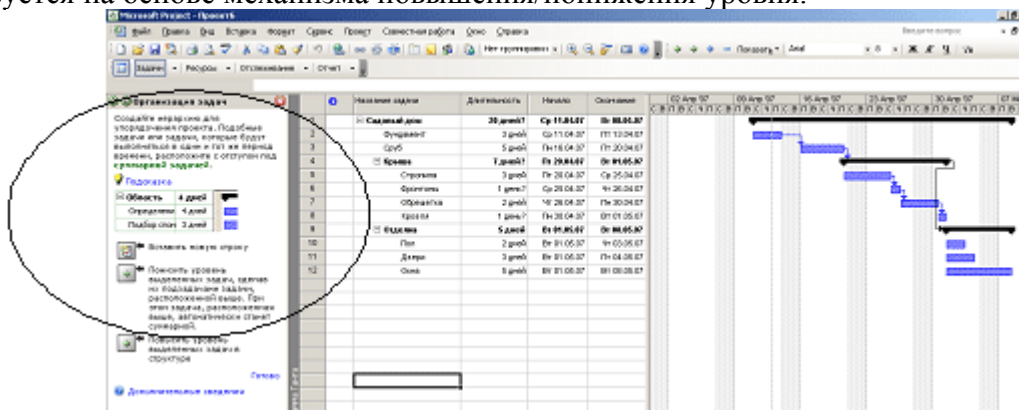
- Садовый дом
  - Фундамент
  - Сруб
  - Крыша
    - Стропила
    - Фронтоны
    - Обрешетка
    - Кровля
  - Отделка
    - Пол

- Двери
- Окна

Эти задачи в MS Project вводятся сначала простым списком, без указания их подчиненности. Указать длительность каждой задачи (произвольно).

Организация этапов задач (Задание иерархии задач)

Задачи проекта, как правило, не бывают равноправными. Список задач представляет собой некоторую иерархическую структуру. В MS Project такая структура реализуется на основе механизма повышения/понижения уровня.

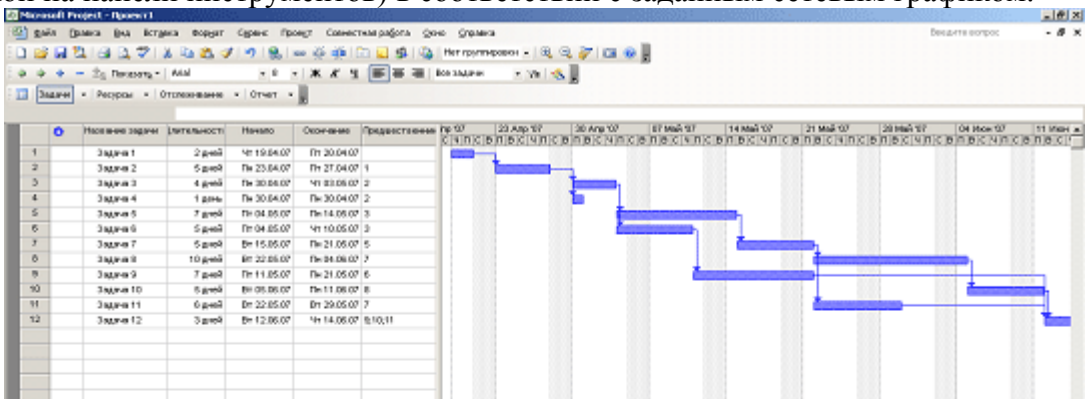


Планирование задач (Задание связей между задачами)

Задачи реального проекта связаны между собой во времени. Какие-то задачи могут выполняться одновременно (связь SS «начало–начало», например, ***Отделка дверей*** и ***Отделка окон***); другие не могут быть начаты, пока не закончатся некоторые предыдущие задачи (связь FS «конец-начало», например, ***Закладка фундамента*** и ***Создание сруба***); третьи должны заканчиваться одновременно (связь FF «конец-конец»). Поэтому после ввода списка задач и создания иерархии необходимо установить связи между задачами.

Связь SS устанавливается кнопкой ***«Связать задачи»*** на Панели инструментов. (Рядом находится кнопка «Разорвать связи»). Применить это к задачам ***Закладка фундамента*** и ***Создание сруба.***

Задание для самостоятельной работы: Открыть новый файл «Учебный проект». Ввести список задач, предварительно скопировав его из файла MS Excel. Установить дату начала проекта 19 апреля 2007 года. Определить рабочее время проекта (без изменений). Перейти к ссылке «Планирование задач» и ввести связи типа «окончание–начало» (кнопкой на панели инструментов) в соответствии с заданным сетевым графиком.



Проверить список предшественников.

2.2. Корректировка списка задач и формирование структуры проекта

Цель работы: умение вносить изменения в список задач проекта, умение формировать структуру проекта путем повышения и понижения уровня задач проекта.

Корректировка списка задач проекта

Во время работы над проектом довольно часто возникает необходимость внести изменения в список задач проекта: добавить новую задачу (в том числе не только в конец списка), удалить ошибочно введенную, изменить порядок следования задач.

Добавление задачи

Чтобы добавить в список новую задачу, надо выделить задачу (щелкнуть левой кнопкой мышки в поле задачи), перед которой нужно поместить новую задачу, и из меню **Правка** выбрать команду **Вставить**. В результате этих действий в список задач будет добавлена пустая строка, в которую можно ввести новую задачу.

Удаление задачи

Чтобы удалить задачу, надо выделить эту задачу (щелкнуть левой кнопкой мышки в поле задачи) и из меню **Правка** выбрать команду **Удалить**.

Перемещение задачи

Чтобы переместить задачу (или группу следующих одна за другой задач) в другое место списка, надо выделить нужную задачу (задачи), и из меню **Правка** выбрать команду ***Вырезать***. Затем выделить задачу, перед которой надо поместить выделенные на предыдущем шаге задачи, и из меню **Правка** выбрать команду ***Вставить***.

Формирование структуры проекта

Представление задач в проекте в виде простого списка недостаточно наглядно. Простой список не отражает структуру проекта, связи между задачами, не позволяет видеть главные и подчиненные задачи. Гораздо удобнее задачи проекта представить в виде иерархического списка, в котором задачи разделены по уровням.

Обычно во время работы над проектом сначала формулируется цель проекта (главная задача), затем она разбивается на фазы (крупные задачи), фазы разбиваются на задачи, задачи — на подзадачи более низкого уровня и т. д. до тех пор, пока не будут определены все необходимые для завершения проекта задачи. Таким образом, проект можно рассматривать как совокупность обобщенных и подчиненных задач. Обобщенная задача – своего рода заголовок, она суммирует стоимость и длительность задач нижнего уровня. Подчиненная задача — это часть обобщенной задачи.

Перевести задачу с одного уровня на другой, сделать ее подчиненной или главной (если задача уже является подчиненной) можно в режиме просмотра диаграммы Ганта, для перехода в который надо из меню **Вид** выбрать команду **Диаграмма Ганта** или в режиме просмотра списка задач. В этих режимах на панели инструментов появляются кнопки, позволяющие формировать структуру проекта.

В режиме просмотра диаграммы Ганта, рядом со списком задач выводится диаграмма Ганта, на которой подчиненные задачи изображаются горизонтальными столбиками, обобщенные — скобками. При этом, если подчиненные задачи начинаются одновременно, то длительность обобщенной задачи полагается равной длительности наиболее длительной подчиненной задачи.

Чтобы понизить уровень задачи, сделать ее подчиненной, надо выделить эту задачу и щелкнуть на кнопке со стрелкой вправо.

Чтобы повысить уровень задачи, сделать ее обобщенной, для задач за ней следующих, надо выделить эту задачу и щелкнуть на кнопке со стрелкой влево.

Следует обратить внимание, что при копировании или перемещении обобщенной задачи все подчиненные задачи также копируются или перемещаются. Если надо переместить только обобщенную задачу, то сначала надо перевести подчиненные задачи на уровень обобщенной. Проект можно просматривать с различной степенью детализации.

Щелкните на кнопке со значком минус, чтобы скрыть подчиненные задачи текущей выделенной, задачи. Если задача является обобщенной и подчиненные задачи скрыты, то нажмите на кнопку со значком плюс, чтобы просмотреть подчиненные задачи. Чтобы увидеть все скрытые задачи проекта, нажмите на кнопку с двумя плюсами.

***Задание 2*:**

1. Откройте файл «Учебный проект».
2. Добавьте в начало списка задач название вашего проекта и сделайте все введенные ранее задачи подчиненными этой задаче.

3. Сформируйте структуру своего проекта: определите главные и подчиненные задачи.

4. Просмотрите задачи проекта. Внесите в него необходимые изменения (например, добавьте несколько новых, уточняющих задач).

5. Научитесь просматривать проект с различной степенью детализации.

6. Сохраните измененный проект.

2.3. Назначение связей между задачами

Цель работы: знание типов связей между задачами, понятий «время опережения» и «время задержки»; умение связывать задачи проекта связями различного типа.

Связи между задачами

Задачи реального проекта связаны между собой во времени. Например, некоторые задачи могут выполняться одновременно, другие не могут быть начаты до тех пор, пока не завершится некоторая предыдущая задача. Поэтому после того, как составлен список задач и задачи распределены по уровням (определены обобщенные задачи и подзадачи), необходимо установить связи между задачами.

Различают три типа связей между задачами проекта:

- конец – начало (finish-to-start, FS);
- начало – начало (start-to-start, SS);
- конец – конец (finish-to-finish, FF).

Чтобы назначить связь типа «конец–начало» между следующими в списке друг за другом задачами, надо выделить эти задачи и щелкнуть на командной кнопке **Связать задачи.**

Время задержки (опережения) выполнения задачи

Иногда между завершением одной задачи и началом другой должно пройти некоторое время, или задача-приемник должна начинаться немного раньше, чем завершится задача-родитель. Например, пол, после покрытия лаком должен сохнуть 48 часов. Следовательно, задача «оборудование» может начаться только через 48 часов после завершения задачи «окраска».

Другой пример. Если монтируется линия электропередачи из 30 столбов, то совсем не обязательно ждать, пока будут установлены все столбы, чтобы приступить к монтажу проводов. Задачу монтажа можно начать после того, как будут установлены, например, 5 столбов. Подобные ситуации моделируются заданием времени отставания для задачи-приемника. Для задания времени отставания (опережения) используют форму. Время отставания задается вводом положительного числа в поле формы, опережения – отрицательного.

Временной масштаб диаграммы Ганта

Диаграмму Ганта удобно использовать для наблюдения связей между задачами проекта. В левой части окна отображается список задач проекта, в правой — диаграмма Ганта. При этом критические задачи, длительность которых оказывает влияние на длительность всего проекта, отображаются красным цветом.

Если проект длительный, то диаграмма Ганта не помещается в одном окне. Это не всегда удобно. Microsoft Project позволяет настроить временной масштаб диаграммы таким образом, чтобы в окне были видны сразу все задачи проекта.

В верхней части диаграммы Ганта выводится временная ось. Масштаб оси устанавливается в диалоговом окне, которое появляется при выборе из меню **Формат**.

***Задание 3*:**

1. Откройте файл «Учебный проект».

2. Установите связи между задачами проекта таким образом, чтобы модель проекта соответствовала реальному проекту.

3. Найдите в проекте задачи, которые должны выполняться с задержкой относительно других задач, и задайте для них время задержки.

4. Найдите в проекте задачи, которые могут выполняться одновременно с другими задачами, но с некоторым опережением начала их выполнения. (Задайте для этих задач время опережения.)

5. Установите такой временной масштаб диаграммы Ганта, чтобы вся диаграмма была видна в одном экране.

6. Сохраните файл проекта.

2.4. Ресурсы проекта

Цель работы: знание понятия «ресурс», характеристик ресурса как составной части проекта, умение составлять список ресурсов проекта.

Цена ресурса и стоимость задачи

Для выполнения задач проекта необходимы ресурсы: люди, оборудование, механизмы. За использование ресурса для реализации задачи надо платить. Стоимость использования ресурса определяется ценой ресурса и временем его использования. Цена ресурса измеряется в денежных единицах, отнесенных к единице времени (день, час, минута). Например, цена ресурса «грузовик» может быть 500 руб./ч. Это означает, что если для выполнения некоторой задачи необходимо использовать ресурс «грузовик» в течение двух часов, то стоимость этого ресурса равна 1000 руб.

Суммарная стоимость ресурсов, использованных для реализации задачи, определяет стоимость этой задачи. Из стоимости ресурсов, использованных для реализации задач проекта, складывается стоимость проекта в целом.

Ресурсы проекта

Работая над проектом, нужно иметь список доступных ресурсов. Наиболее легко составить и просмотреть список ресурсов проекта можно в режиме «Список ресурсов».

Задание 4:

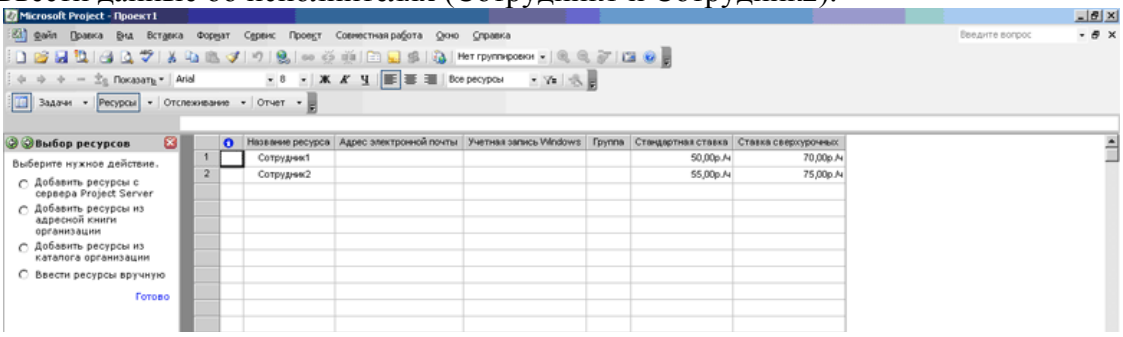
1. Откройте файл «Учебный проект».

2. Составьте список ресурсов, необходимых для реализации вашего проекта. Для каждого ресурса задайте цену, цену при сверхурочном использовании, величину фиксированной платы за ресурс, количество ресурса, доступное в рамках реализации проекта и другие характеристики.

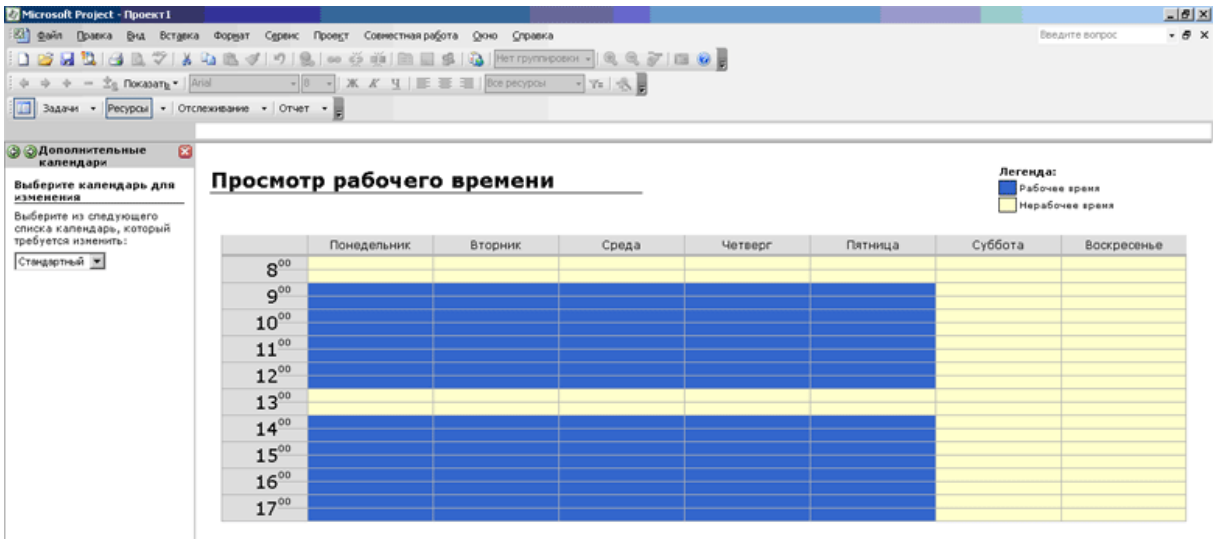
3. Сохраните измененный проект.

Пример выполнения

Ввести данные об исполнителях (Сотрудник1 и Сотрудник2).



Каждому сотруднику определить рабочее время (пока без изменений).



2.5. Назначение ресурсов задачам проекта

Цель работы: умение назначать ресурсы задачам проекта.

Стоимость и длительность реализации задачи

После того, как сформирована структура проекта (составлен список задач, определены главные и подчиненные задачи, назначены связи между задачами) и составлен список ресурсов проекта, можно приступить к назначению ресурсов задачам проекта. В результате назначения ресурсов задачам будет определена стоимость и длительность задач проекта.

Стоимость задачи определяется стоимостью ресурсов, необходимых для ее реализации. Чем больше ресурсов требует задача, тем она дороже, тем больший вклад вносит она в стоимость реализации проекта в целом.

Длительность реализации задачи зависит от количества единиц ресурса, выделенных для ее реализации. Здесь надо вспомнить, что на этапе составления списка задач длительность каждой задачи задавалась в предположении, что она реализуется одной единицей ресурса. После назначения ресурсов длительность выполнения задачи определяется уже количеством единиц ресурсов, выделенных для ее решения.

Назначение ресурсов

Время реализации проекта зависит от длительности реализации задач, которая в свою очередь зависит от количества ресурсов, выделенных для их реализации. Таким образом, чем больше ресурса будет выделено задаче, тем быстрее она будет выполнена. Следует обратить внимание, что некоторые задачи, которые могут выполняться одновременно, могут требовать одинаковых ресурсов. При этом возникает проблема распределения ресурсов между задачами: как распределить ресурсы таким образом, чтобы время реализации всего проекта было минимальным? Очевидно, что в первую очередь ресурсы надо выделять задачам, длительность которых определяет длительность реализации всего проекта (такие задачи называются критическими и на диаграмме Ганта изображаются красными прямоугольниками).

Чтобы назначить ресурс задаче, надо выделить, например в режиме просмотра диаграммы Ганта, задачу, которой назначается ресурс, и щелкнуть на кнопке «Ресурс».

Следует обратить внимание, что при назначении задачи одной единицы ресурса, значения количества работы и длительности задачи совпадают. Если задаче назначить несколько единиц ресурса, то длительность задачи сокращается пропорционально количеству назначенных единиц, при этом величина количества работы не изменяется.

Одной задаче можно назначить несколько разных ресурсов. Однако следует обратить внимание, что на длительность задачи влияет только первый из назначенных ресурсов. Остальные только изменяют стоимость задачи.

После назначения задачам ресурсов, необходимо убедиться, что ресурсы назначены правильно, например, что некоторый ресурс не назначен нескольким задачам, которые

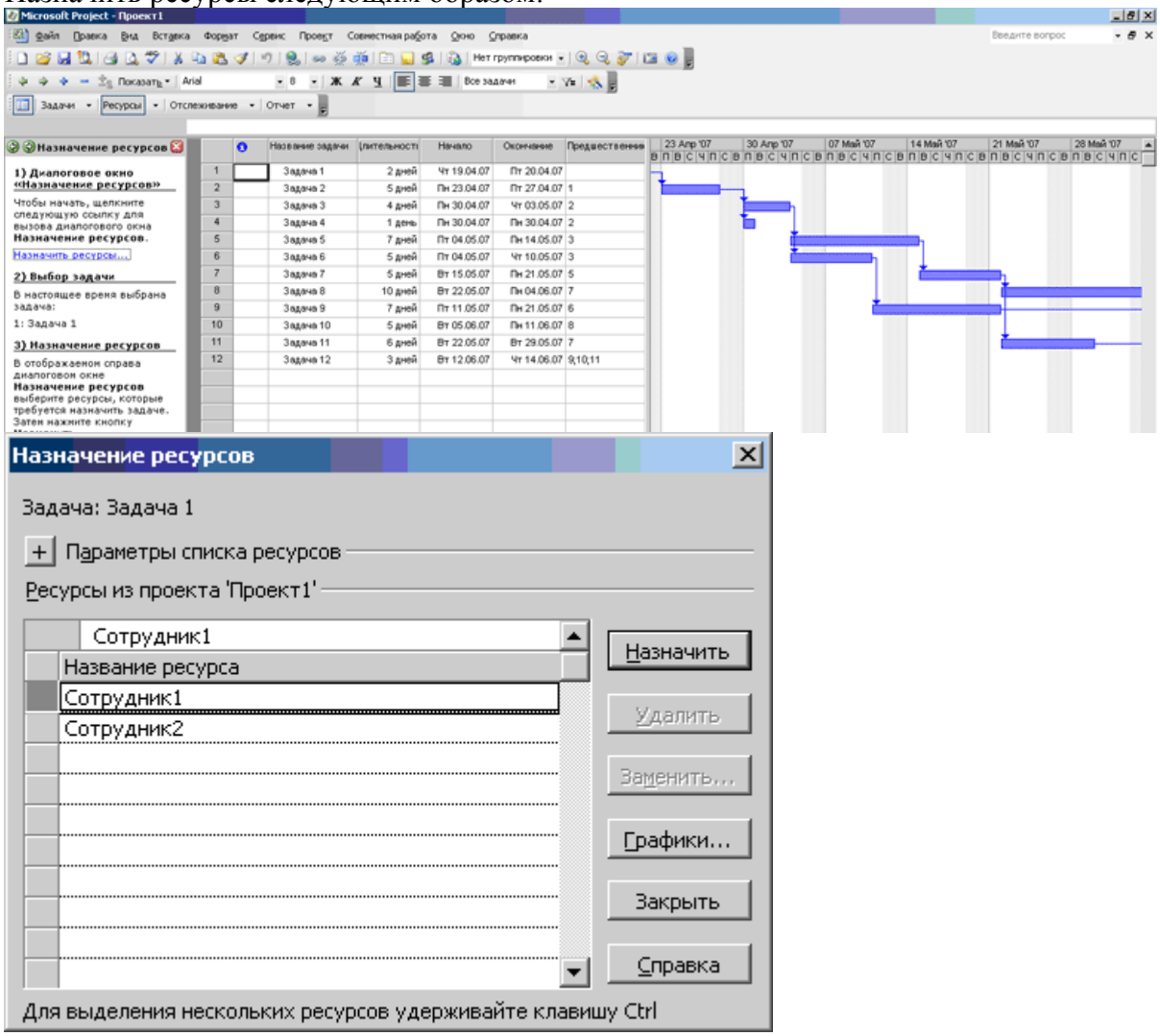
выполняются одновременно, или какой-либо задаче не назначено больше единиц ресурса, чем доступно для реализации проекта.

***Задание 4*:**

1. Откройте файл проекта, над которым вы работаете.
2. Назначьте задачам проекта ресурсы. Используя режим просмотра списка ресурсов, убедитесь, что ресурсы задачам назначены верно.
3. Сохраните измененный проект.

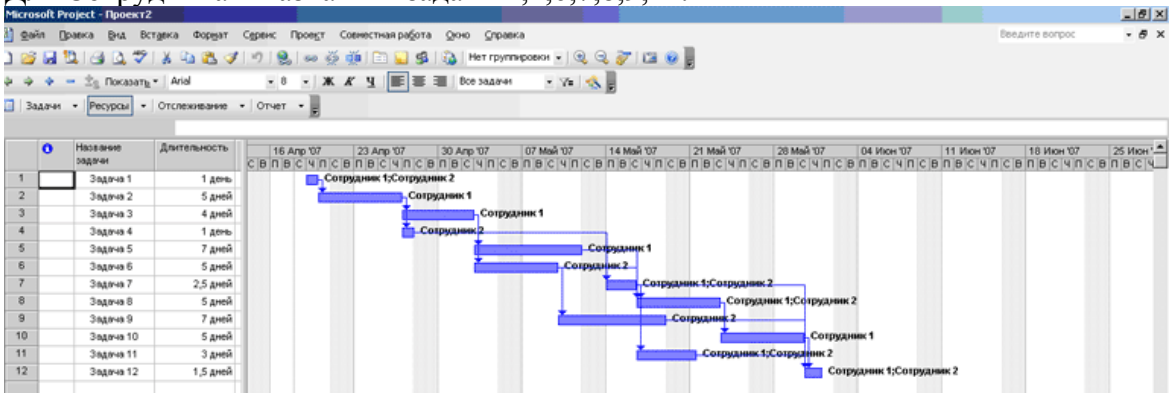
Пример выполнения

Назначить ресурсы следующим образом:



Для Сотрудника 1 выделить задачи 1,2,3,5,7,8,10,11 (удерживая клавишу Ctrl) и нажать кнопку «Назначить».

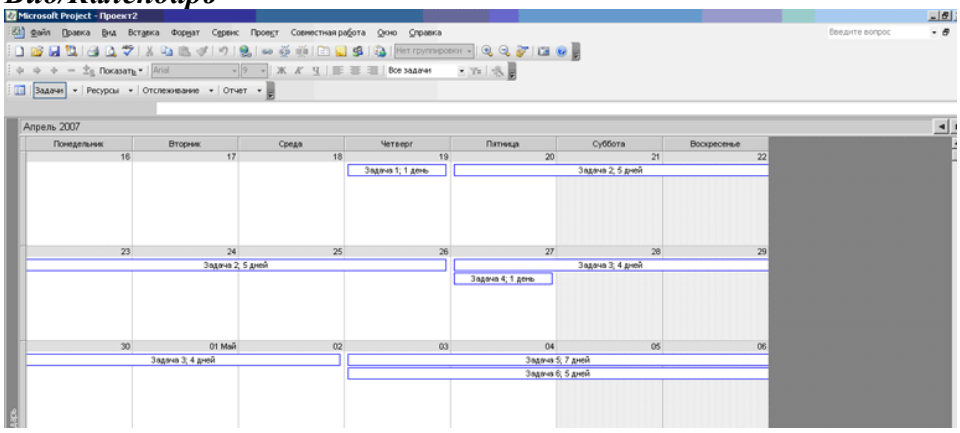
Для Сотрудника 2 назначить задачи 1,4,6,7,8,9,11.



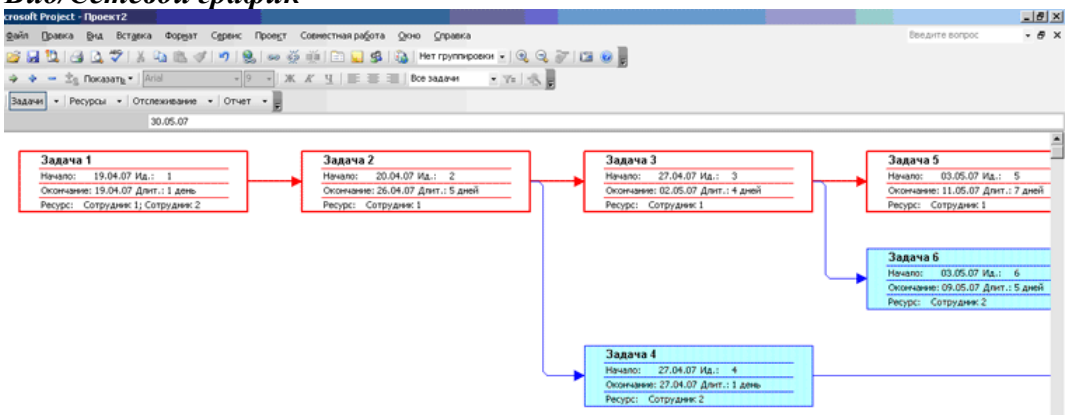
2.6. Варианты представления проекта (меню «Вид»)

Графические варианты представления проекта очень разнообразны. Примеры представлены далее.

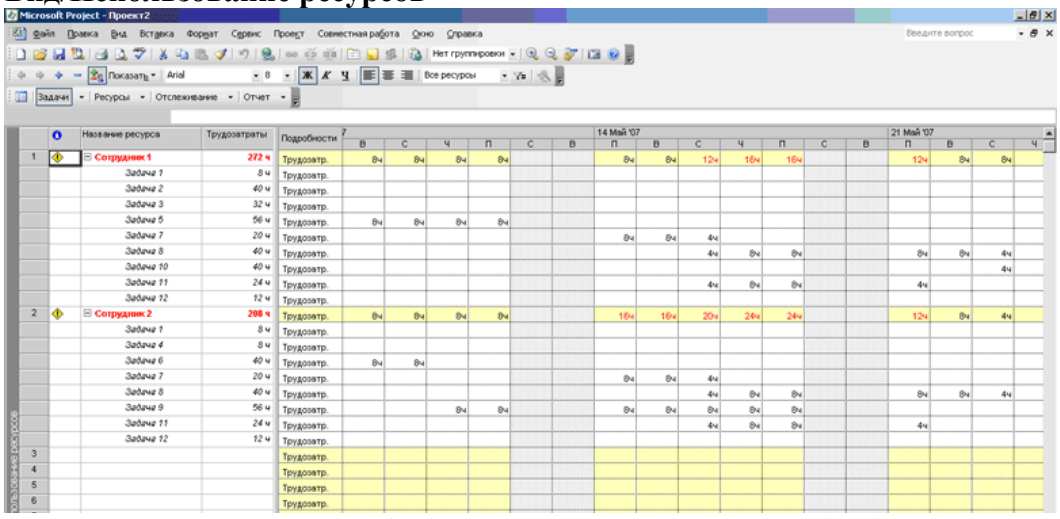
Вид/Календарь



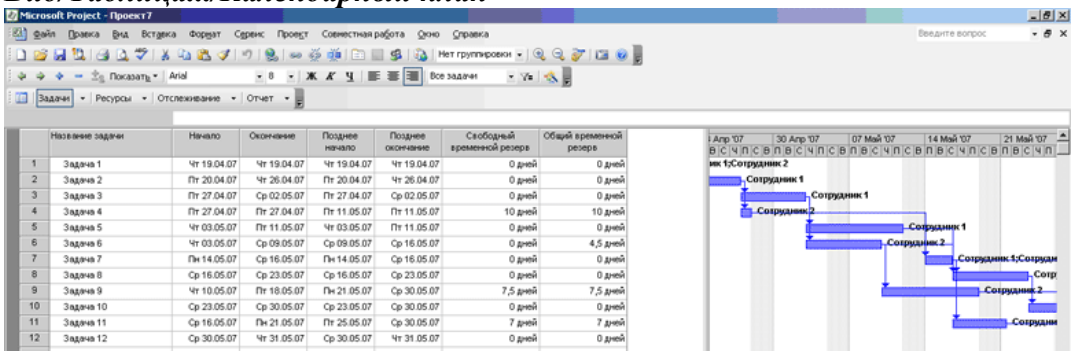
Вид/Сетевой график



Вид/Использование ресурсов

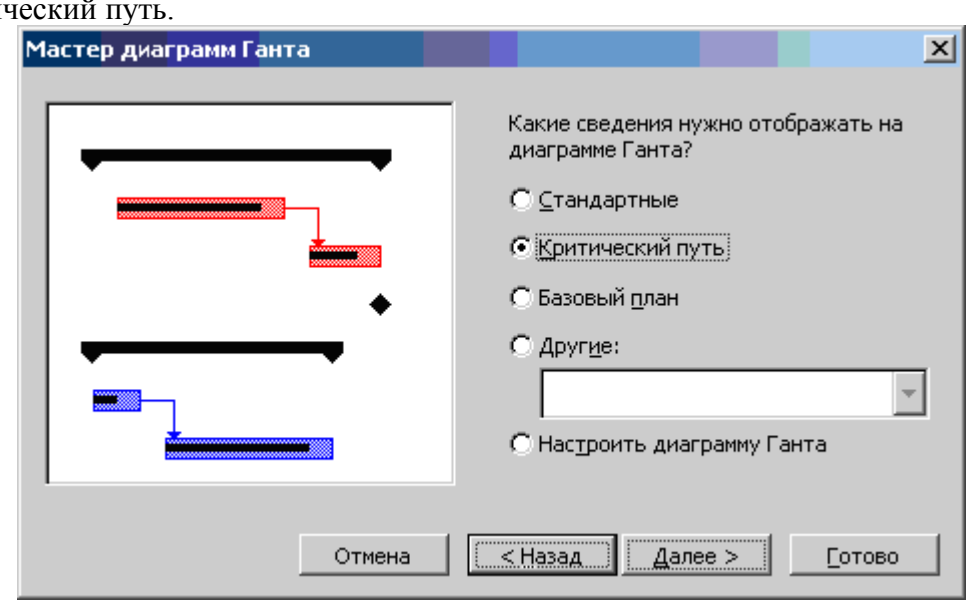


Вид/Таблица.../Календарный план

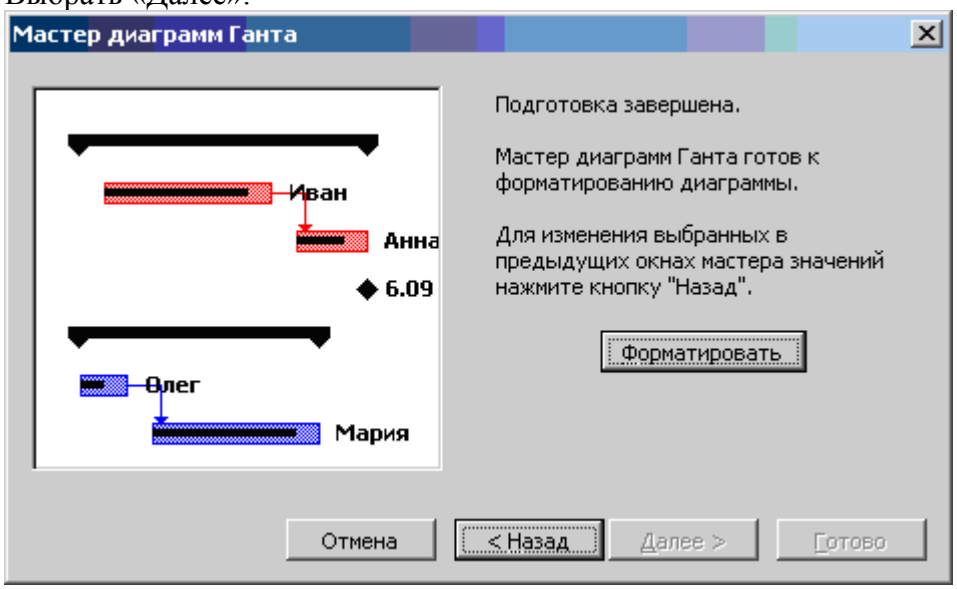


Просмотр критического пути

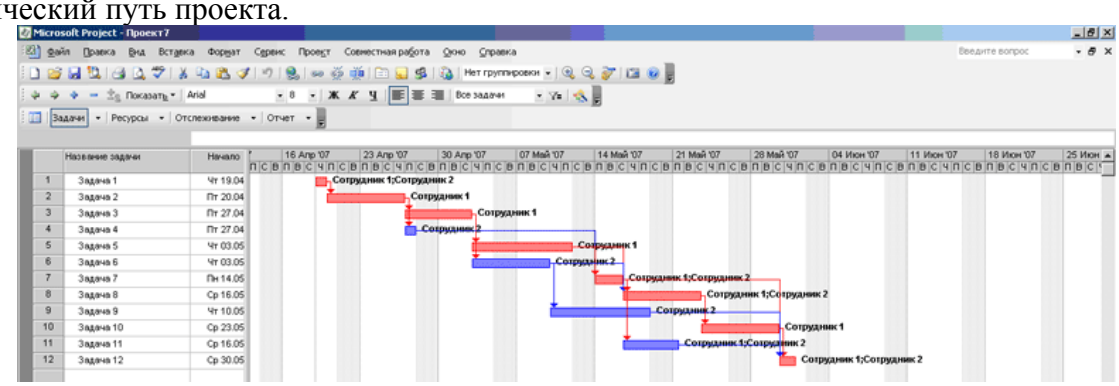
Для вывода на экран критического пути необходимо открыть меню ***Формат/Мастер диаграмм Ганта,*** и в окне Мастера диаграмм Ганта отметить точкой критический путь.



Выбрать «Далее».

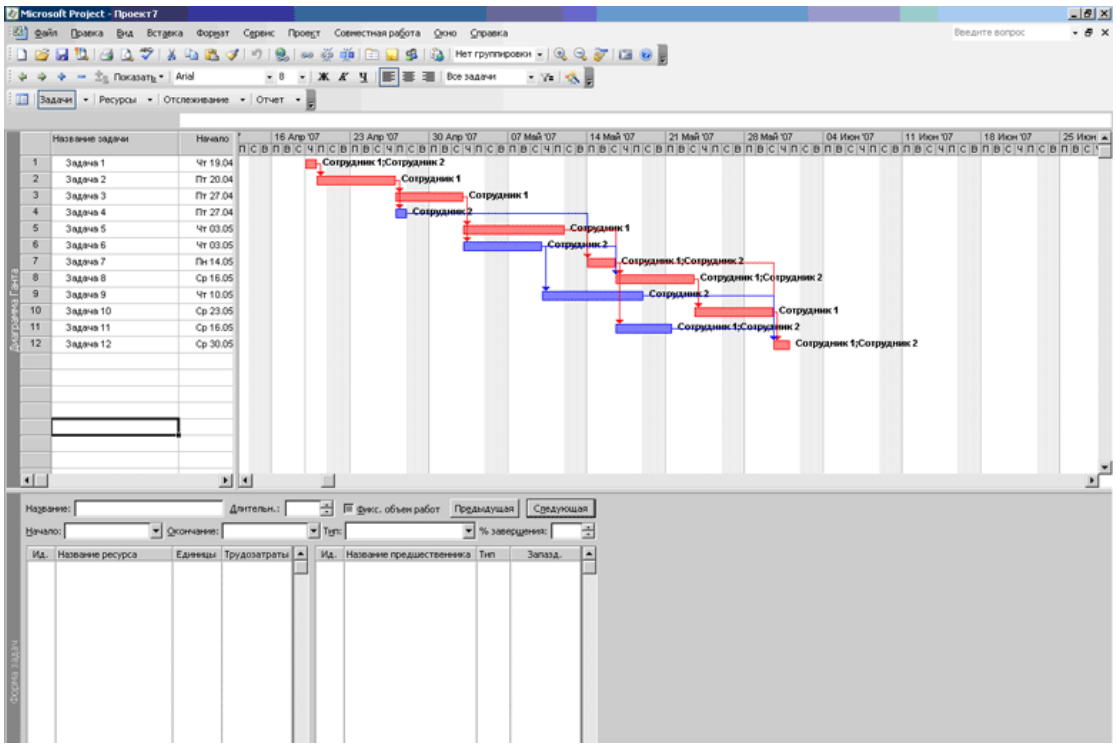


Выбрать «Форматировать» и «Выход». В результате в окне будет отражен критический путь проекта.

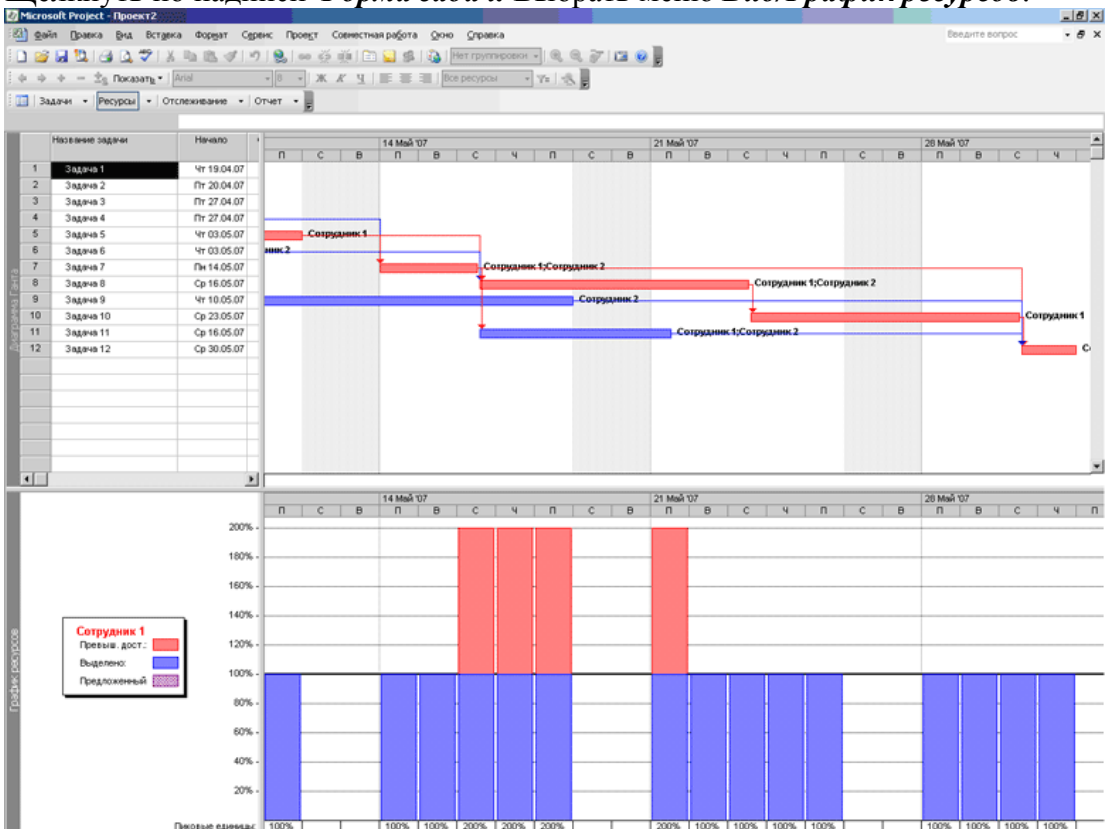


Просмотр графика загрузки ресурсов

Для просмотра графика загрузки ресурсов выбрать меню ***Окно/Разделить.***



Щелкнуть по надписи ***Форма задач.*** Выбрать меню ***Вид/График ресурсов***.



Результатом будет график загрузки ресурсов.

2.7. Устранение перегрузки ресурсов

В процессе выполнения предыдущего задания было выяснено, что и сотрудник 1, и сотрудник 2 работают с перегрузкой на временном отрезке 14 мая — 21 мая.

Далее начинается работа по устранению перегрузок ресурсов. Для этого в MS Project есть несколько путей, в частности:

- перенос (сдвиг) времени выполнения отдельных задач (только не лежащих на критическом пути);
- прерывание выполнения отдельных задач;

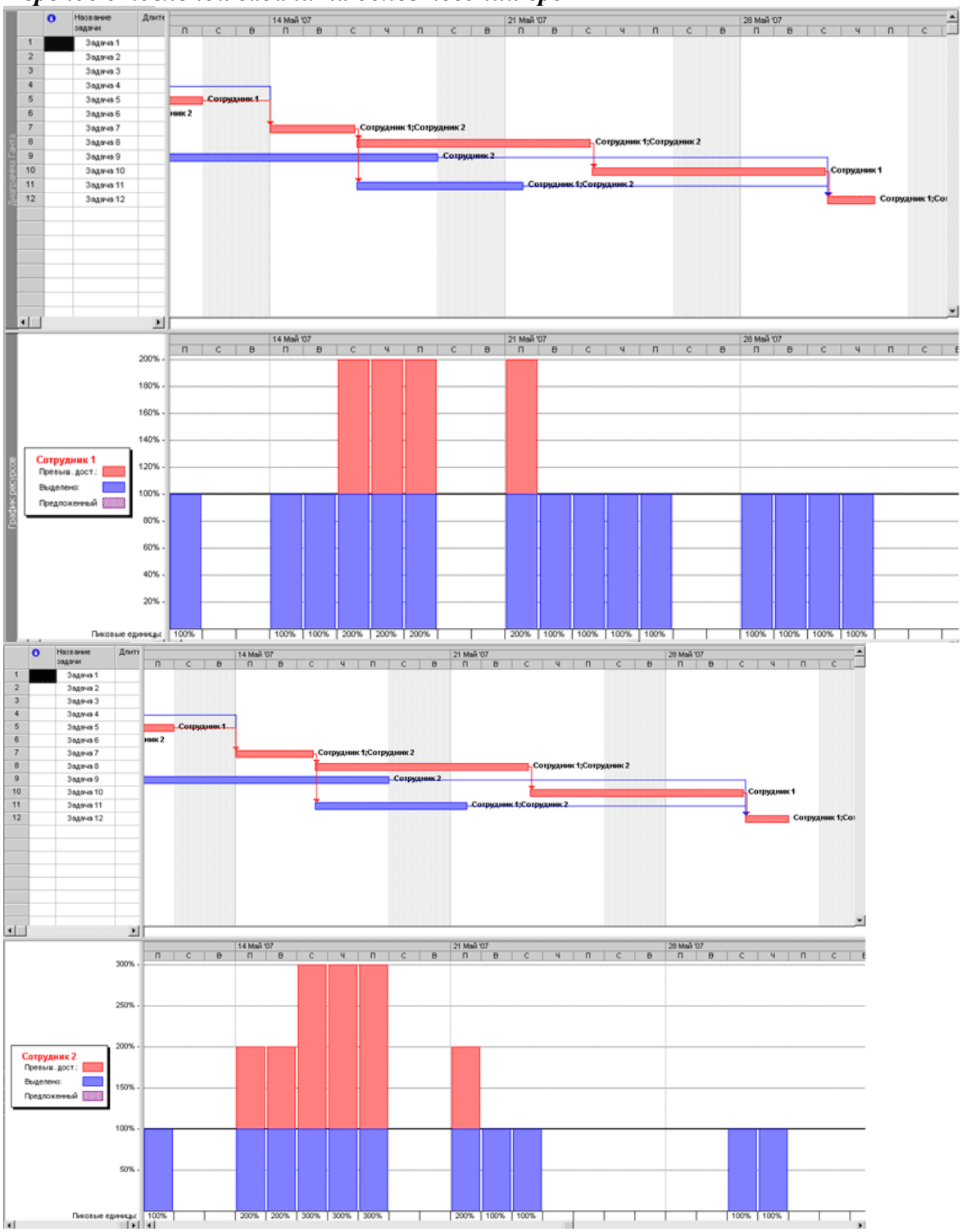
- добавление сверхурочных работ (что не устраняет перегрузки, но по-другому оплачивается);
- назначение работы в выходные дни;
- добавление в список ресурсов новых сотрудников и др.

Все эти методы требуют предварительного анализа ситуации. И не всегда приводят к отличному результату.

Поэтому необходимо оставить один файл в качестве образца, а все изменения проекта производить на его копиях.

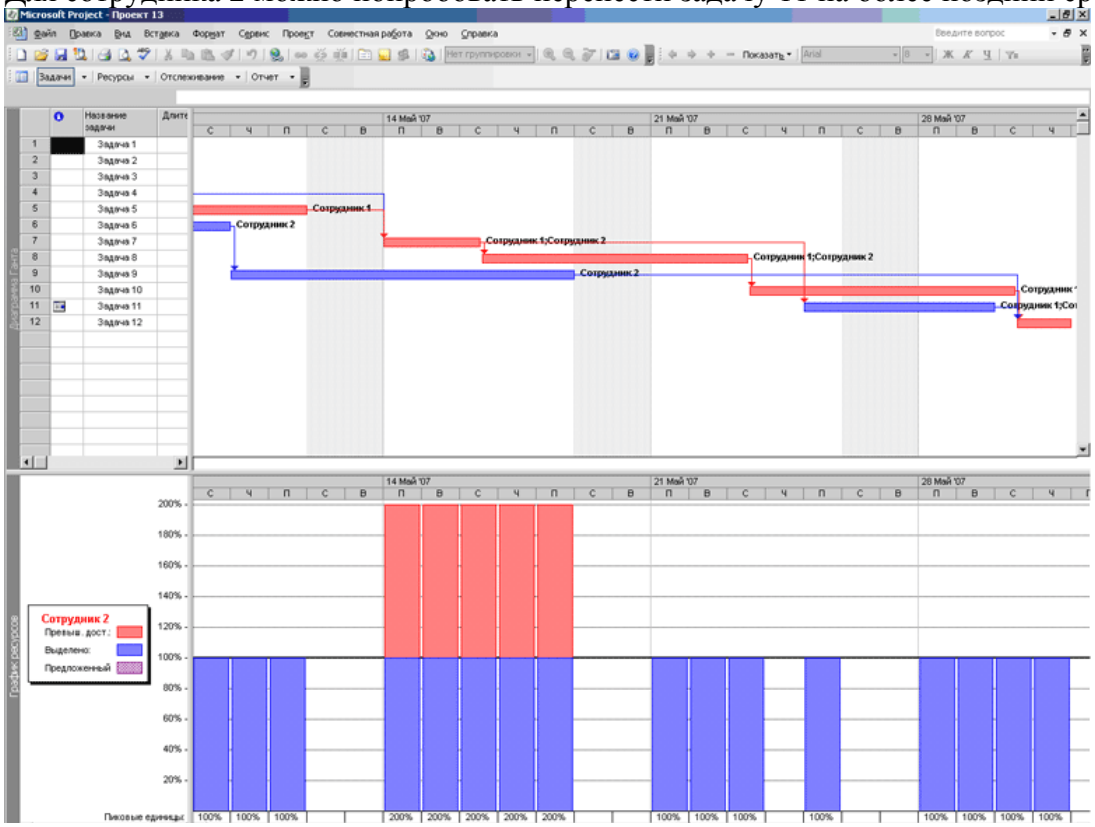
Рассмотрим конкретные примеры.

Перенос отдельной задачи на более поздний срок

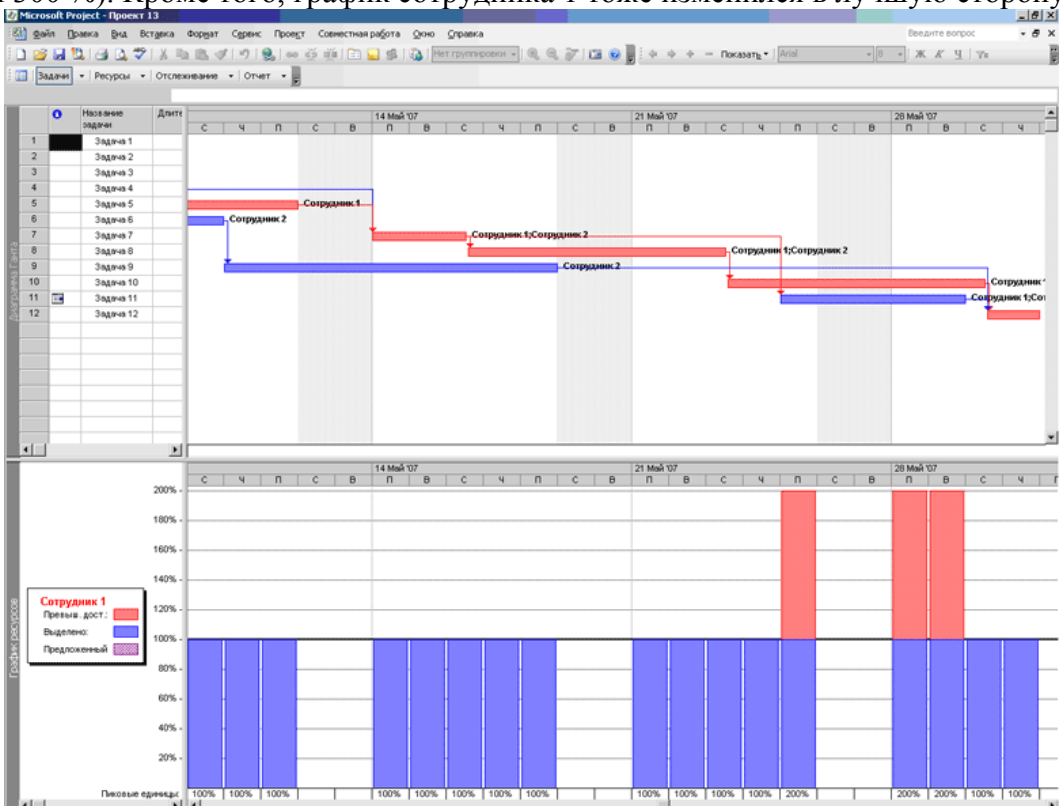


Из анализа графика сотрудника 1 ясно, что выполняемые им задачи передвигать не имеет смысла, так как этот сотрудник всегда занят на 100 %.

Для сотрудника 2 можно попробовать перенести задачу 11 на более поздний срок.

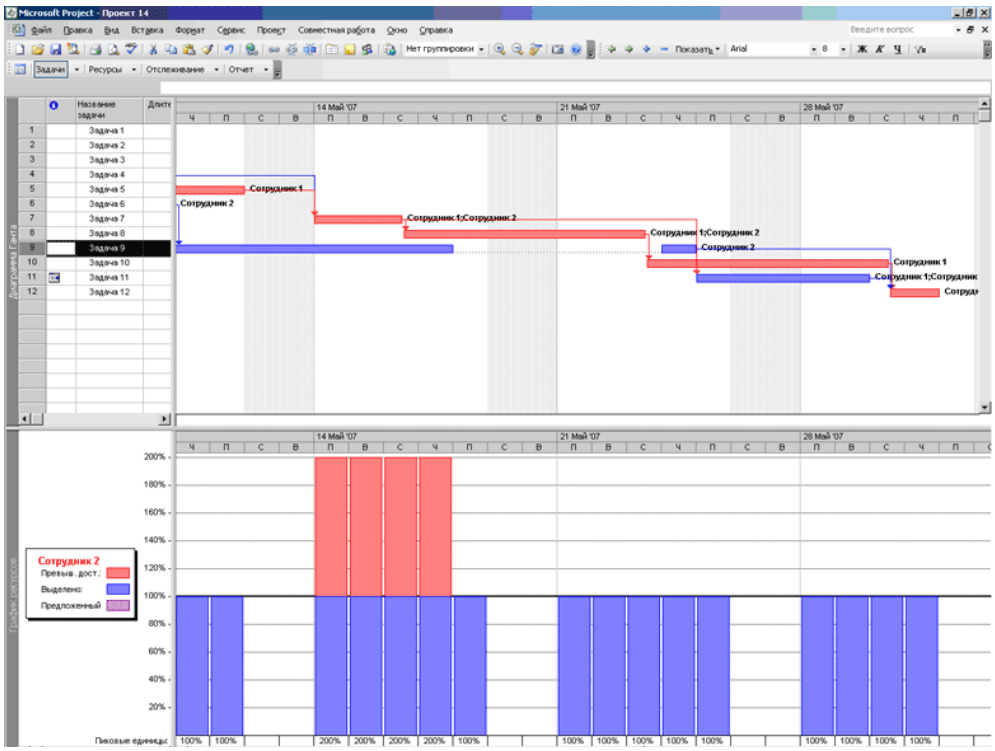


Часть перегрузки устранена. Осталось 5 дней с перегрузкой не более 200 % (ранее было и 300 %). Кроме того, график сотрудника 1 тоже изменился в лучшую сторону.



Прерывание выполнения задачи

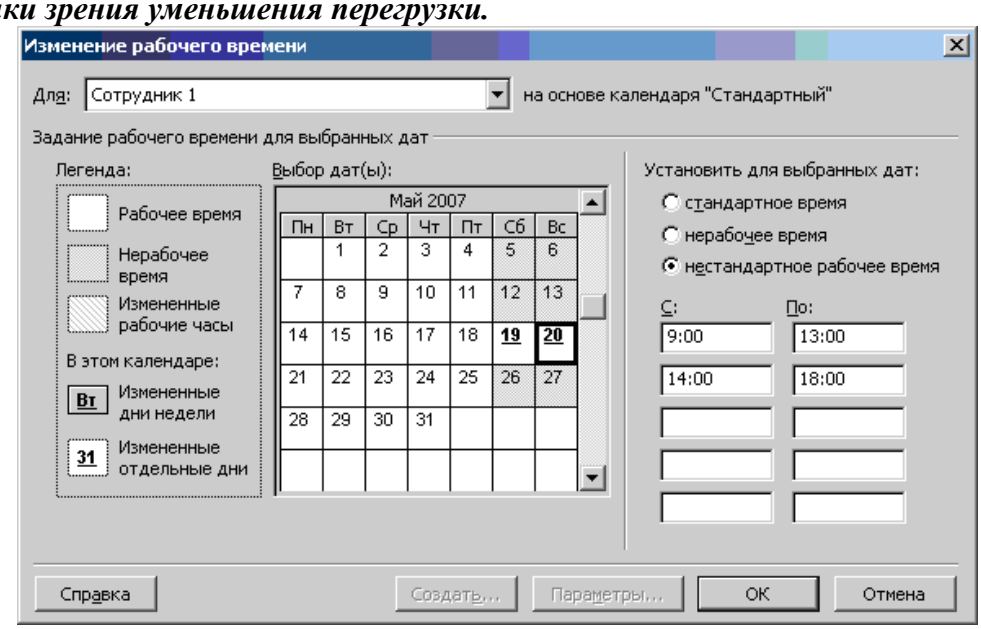
По графику сотрудника 2 видно, что 24 мая он не выполняет никаких работ. Поэтому задачу 9 можно прервать (есть кнопка на панели инструментов) и перенести на этот день.



Таким образом, в результате переноса и прерывания двух задач перегрузка сотрудников уменьшена примерно в 2 раза.

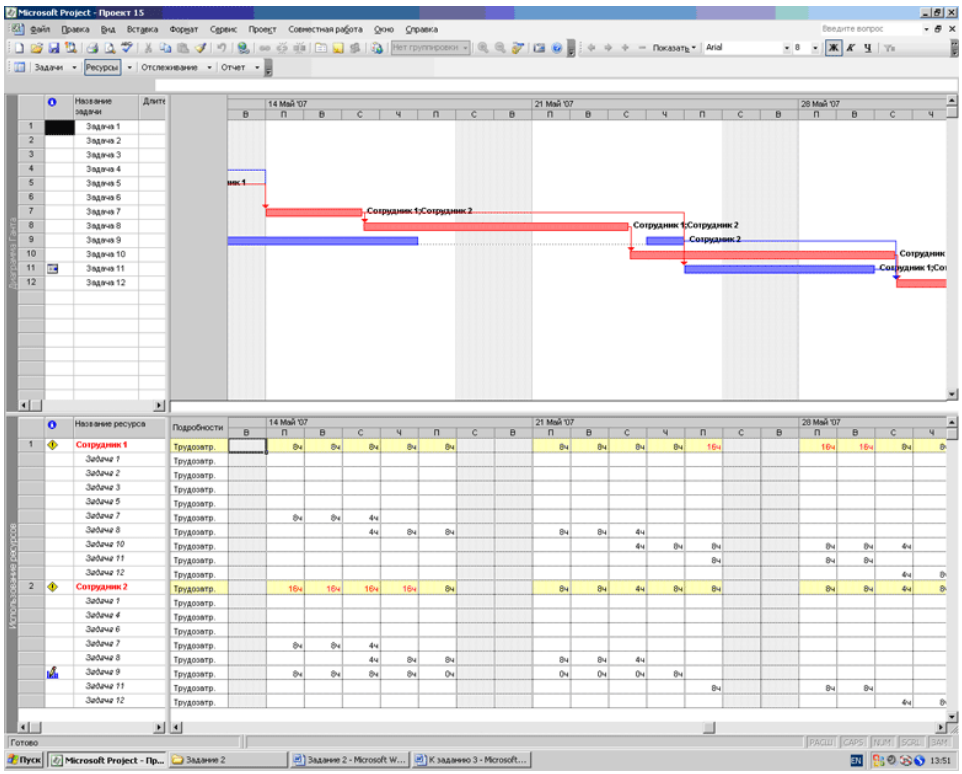
Использование выходных дней

Для этого варианта необходимо для конкретного сотрудника перейти в режим «Изменение рабочего времени» и установить там нестандартное рабочее время в выходные. ***Определите, какие выходные наилучшим образом для этого подходят с точки зрения уменьшения перегрузки.***

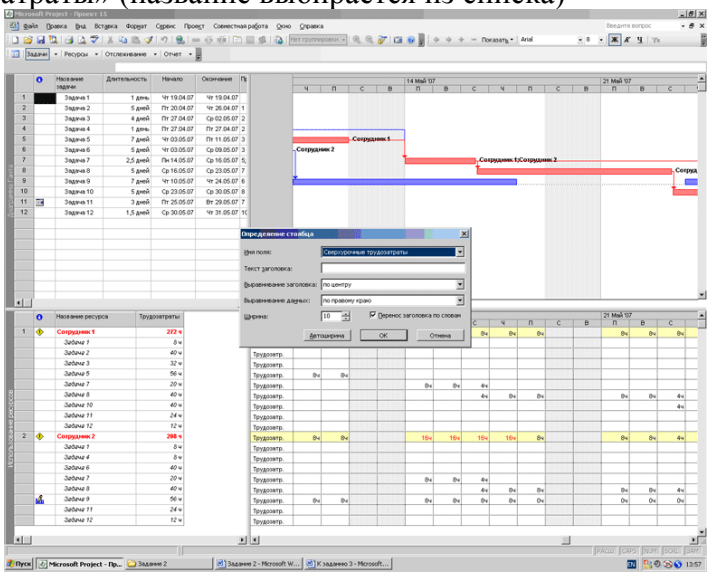


Назначение сверхурочных часов

Выбираем меню ***Вид/Использование ресурсов.***



Перегрузка отмечена красным цветом. Вставить новый столбец «Сверхурочные трудозатраты» (название выбирается из списка)



В таблице, которая после этого появляется, необходимо установить сверхурочные часы (трудозатраты) для конкретных задач.

Добавление нового сотрудника

Самостоятельно добавить в список ресурсов сотрудника 3 и назначить ему часть работ.

Все изменения графика перегрузки оформить в виде отчета.

Практическая часть

Задание на самостоятельную работу

Требуется составить проект сетевого графика разработки веб-проекта в MS Project и провести его оптимизацию.

В индивидуальное задание входят перечень и продолжительность работ, численность исполнителей. В разработке веб-проекта принимают участие дизайнеры, верстальщики и веб-программисты. Продолжительность работ задается как

оптимистическая (минимальная), ожидаемая (средняя) и пессимистическая. Необходимо спроектировать ход работы, устранить возможные перегрузки.

Создать отчет, в котором:

1. Вывести на печать (до оптимизации):
   - сетевой график;
   - диаграмму Ганта;
   - таблицу: календарный план;
   - таблицу: суммарные данные;
   - список ресурсов;
   - графики загрузки ресурсов;
2. Вывести на печать (после оптимизации):
   - диаграмму Ганта;
   - таблицу: суммарные данные;
   - графики загрузки ресурсов;
   - отчет: сводка по проекту.
3. Выводы по проекту.

Содержание отчета

1. Цель.
2. Техническое задание оформленное в соответствии с вариантом по образцу .
3. Выводы

Контрольные вопросы

1. Перечислите типы связей, которыми могут быть связаны задачи проекта.
2. Если некоторая задача может быть начата только через некоторое время после завершения другой (например, из технологических соображений), то как этот факт отображается в модели проекта?
3. Если некоторая задача должна начинаться несколько позже, чем закончится другая задача проекта, то как этот факт отразить в модели проекта?
4. Если длительность проекта такова, что диаграмма Ганта не помещается в пределах экрана, то что и как надо сделать, чтобы вся диаграмма Ганта была видна одновременно?
5. Перечислите характеристики ресурса.
6. Чем определяется стоимость задачи проекта?
7. Чем определяется стоимость проекта в целом?

Лабораторная работа № 5

Создание серверных сценариев с использованием операторов PHP

Цель: отработать навыки создания серверных сценариев с использованием операторов PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Веб-приложение — это приложение, работающее по принципу «клиент-сервер», в котором обмен информацией происходит по протоколу HTTP.

Денвер. Установка программного продукта.

Сервер на базе Денвер предназначен для создания собственных доменов на локальном компьютере, а также позволяет создавать сайты и тестировать их работоспособность, не имея доступа в Интернет.

Денвер — набор дистрибутивов (Apache, PHP, MySQL, Perl и т.д.) и оболочка для разработки сайтов на «домашней» (локальной) Windows-машине без выхода в Интернет.

Базовый (основной) пакет Денвера включает:

- Инсталлятор (поддерживается также инсталляция на flash-накопитель).
- Apache, SSL, SSI, mod_rewrite, mod_php.
- PHP7 с поддержкой GD, MySQL, sqLite.
- MySQL7 с поддержкой транзакций.
- Система управления виртуальными хостами, основанная на шаблонах. Чтобы создать новый хост, вам нужно лишь добавить директорию в каталог /home, править конфигурационные файлы не требуется. По умолчанию уже поддерживаются схемы именования директорий многих популярных хостеров; новые можно без труда добавить.
- Система управления запуском и завершением всех компонентов Денвера.
- phpMyAdmin — система управления MySQL через Web-интерфейс.
- Эмулятор sendmail и SMTP-сервера (отладочная «заглушка» на localhost:25, складывающая приходящие письма в /tmp в формате .eml); поддерживается работа совместно с PHP, Perl, Parser и т.д.

Отличительной особенностью Денвера является его полная автономность. Она заключается в следующем.

Денвер устанавливается в один-единственный каталог и вне его ничего не изменяет. Он не пишет файлы в Windows-директорию и не прописывается в Реестре. При желании вы можете даже поставить себе сразу два Денвера, и они не будут конфликтовать.

Никакие «сервисы» NT/2000 не «прописываются». Если вы запустили Денвер, то он работает. Если завершили — то перестает работать, не оставляя после себя следов.

Системе не нужен деинсталлятор — достаточно просто удалить каталог.

Все конфигурирование и настройка под конкретную машину происходит автоматически.

Эти же правила распространяются и на пакеты расширений.

В целях упрощения работы компонентов комплекса и улучшения совместимости с реальным Unix-хостером при старте создается специальный виртуальный диск, присоединенный к основной директории.

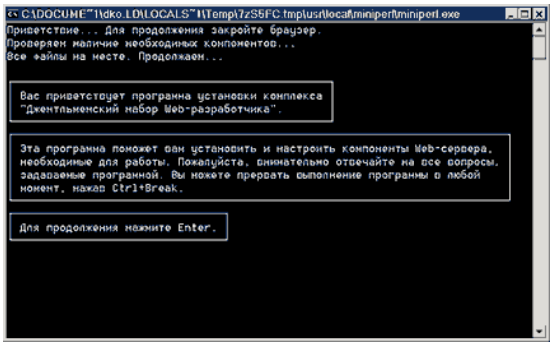
Виртуальный диск — это просто синоним для некоторой папки на реальном, или

физическом, диске. Подключается он при помощи команды subst, о чем заботятся скрипты Денвера. Вы можете работать с виртуальным диском, как с обычным. При этом все операции в действительности будут производиться с указанной директорией. Механизм работы виртуальных дисков встроен в ОС и не ведет к каким-либо издержкам и замедлениям.

За счет применения виртуального диска Денвер «изнутри» похож на маленький Unix: у него есть своя директория /home, /usr, /tmp... Различные компоненты и серверы расположены так, как это принято в Unix. Например, в /home располагаются виртуальные хосты, а в /usr — программные компоненты.

Установка Денвера

Для установки программного продукта Денвер (Денвер) необходимо скачать дистрибутив с сайта http://denwer.ru и запустить установочный файл. Во время установки необходимо выбрать следующие опции:

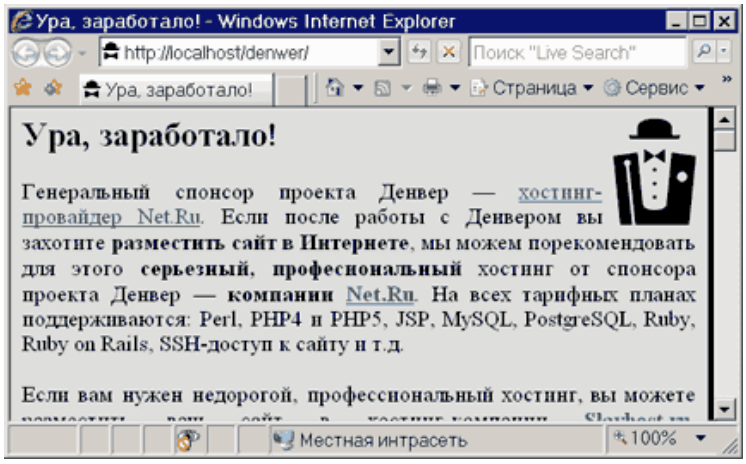


- Запустите скачанный инсталлятор Денвера.
- установку осуществить в папку C:\WebServer, а не в привычную папку C:\Program Files; по умолчанию используется C:\WebServers, вам нужно лишь нажать Enter, чтобы согласиться с этим выбором. В указанном каталоге будут расположены абсолютно все компоненты системы, и вне его никакие файлы в дальнейшем не создаются (исключая ярлыки на Рабочем столе).
- Далее вам предложат ввести имя виртуального диска, который будет связан с только что указанной директорией. Рекомендуем вам согласиться со значением по умолчанию (Z:). Важно, что диска с этим именем еще не должно содержаться в системе — чаще всего так и происходит с диском Z:. После этого начнется копирование файлов дистрибутива.
- Вам будет задан вопрос, как именно вы собираетесь запускать и останавливать комплекс. У вас есть две альтернативы:
- Создавать виртуальный диск при загрузке машины (естественно, инсталлятор позаботится, чтобы это происходило автоматически), а при остановке серверов его (диск) не отключать.
- Создавать виртуальный диск только по явной команде старта комплекса (при щелчке по ярлыку запуска на Рабочем столе). И, соответственно, отключать диск от системы — при остановке серверов. Этот вариант является более предпочтительным.
- Согласитесь создать ярлыки на рабочем столе для работы с Денвером.

Первый запуск Денвера

После установки Денвера его следует запустить вручную и проверить работоспособность.

Щелкните по созданному инсталлятором ярлыку Start Denwer на Рабочем столе, а затем, дождавшись, когда все консольные окна исчезнут, открывайте браузер и набирайте в нем адрес: http://localhost/denwer/. Выходить из Интернета при этом не обязательно.



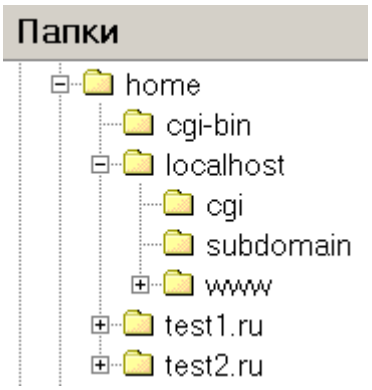
Прочитайте информацию из этого файла и обязательно протестируйте Денвер на работоспособность сервера.

Работа с виртуальными хостами

Вниманию пользователей Windows NT, 2000 или XP (и старше). Прежде, чем продолжить, убедитесь, что у вас запущена служба «DNS-клиент». Это можно сделать, открыв Панель управления — Администрирование — Службы. В противном случае виртуальные хосты работать не будут.

Если вы занимаетесь разработкой Web-сайтов, вам наверняка хотелось бы обслуживать одним сервером сразу несколько хостов. Иными словами, введя в браузере путь http://localhost, вы попадете на один сайт, а, напечатав http://test1.ru, — совсем на другой (но тоже на локальной машине).

Добавить новый виртуальный хост в Денвере чрезвычайно просто. Пусть это будет test1.ru. Вам нужно сделать следующее:



Откройте папку C:\WebServer;

Создать в папке /home директорию с именем, совпадающим с именем виртуального хоста (в нашем случае test1.ru). Причем имя директории содержит точку. Эта директория будет хранить директории документов доменов третьего уровня для test1.ru.

Например,

имя abc.test1.ru связывается сервером с директорией /home/test1.ru/abc/,

имя abc.def.test1.ru — с /home/test1.ru/abc.def/.

поддиректория www соответствует адресам www.test1.ru и просто test1.ru.

На рисунке показано, как может выглядеть директория /home. Обязательно создайте папку www в директории виртуального хоста, ведь именно в ней будут храниться его страницы и скрипты!

Перезапустить сервер, воспользовавшись, например, ярлыком Restart Denwer на Рабочем столе.

Это все, что нужно сделать для нормальной работы сервера.

Для остановки работы сервера необходимо нажать по ярлыку Stop Denwer.

Зачем вообще нужен локальный сервер?

В последнее десятилетие во всем мире наблюдается настоящий бум среди Web-разработчиков (по преимуществу это программисты). Они устанавливают у себя на Windows-машине сервер Apache с различными дополнениями к нему: PHP, Perl, MySQL и т.д. — преимущественно в целях более удобной отладки сайтов.

Многие (преимущественно дизайнеры) могут спросить: зачем вообще нужен локальный Web-сервер, когда страницы можно открывать и так — прямо с диска? Если это обычные (статические) HTML-страницы, то да, сервер не нужен. Однако даже для такой мелочи, как SSI (Server-side Includes — директивы в страницах, позволяющие

вставлять на нужное место содержимое других файлов), уже необходим сервер. Не говоря уж о скриптах — они без сервера просто не запустятся.

Обычно все эти проблемы решают при помощи FTP-клиентов: закачивают исправленные страницы и скрипты на «настоящий» сервер в Интернете, смотрят, что получилось, затем лезут в редактор, исправляют, снова закачивают и т.д. до бесконечности. Главный недостаток такого подхода очевиден: необходимо все время быть подключенным к Интернету. Также очень желательно иметь хорошую связь, потому что в противном случае работа будет продвигаться крайне медленно.

Введение в язык PHP

PHP - язык создания сценариев. PHP - это аббревиатура от слов Personal Home Page. PHP из набора инструментов превратился в полноценный язык программирования, а его название было изменено как рекурсивное образование PHP HyperText Preprocessor (препроцессор гипертекста PHP).

PHP - это серверный язык создания сценариев. Конструкции PHP, вставленные в HTML-текст, выполняются сервером при каждом посещении страницы. Результат их обработки вместе с обычным HTML-текстом передается браузеру.

В настоящее время основной версией PHP является седьмая.

Существуют два основных конкурента PHP: Active Server Pages (ASP) компании Microsoft и ColdFusion компании Allaire. По сравнению с ними PHP обладает рядом преимуществ, в числе которых:

- Высокая производительность. PHP-программы работают быстрее, чем ASP.
- Функциональность. Разработку PHP-программы можно отделить от собственно разработки Web-страницы, что упростит жизнь и программисту, и дизайнеру.
- Цена. PHP абсолютно бесплатен.
- Простота в использовании. Имеющие опыт программирования на распространенных языках найдут синтаксис PHP хорошо знакомым.
- Переносимость. Один и тот же PHP-код можно использовать как в среде NT, так и на платформах UNIX.

Общие правила построения PHP-программы

PHP-программы состоят из простого текста, поэтому набирать их можно в любом текстовом редакторе. Популярные HTML-редакторы имеют встроенную поддержку для редактирования PHP-программ.

Расширение файлов PHP-программ по умолчанию в PHP4 - .php. На основании этого расширения сервер распознает файл как PHP-программу и запускает интерпретатор.

PHP-программа должна быть отделена от обычного HTML-текста. Существует четыре стиля обрамления PHP-кода:

Стиль	Открывающий тег	Закрывающий тег
Сокращенный	<?	?>
XML (стандартный)	<?php	?>
ASP	<%	%>
SCRIPT (программный)	<SCRIPT LANGUAGE="php">	</SCRIPT>

При помощи небольших кусков php-кода можно эффективно создавать большие и сложные html-документы. Это одно из существенных преимуществ PHP.

Программа на языке PHP – это всегда не более чем html-страница, содержащая специальную PHP-разметку. PHP-код обрабатывается отдельной программой на сервере, и результаты включаются в веб-страницу до отправки ее пользователю.

Основы языка PHP

Комментарии в PHP-программе могут быть трех стилей:

```
/* Многострочный комментарий
в стиле классического Си */
```

// Однострочный комментарий в стиле C++

Однострочный комментарий в стиле Perl

Константы

Константы определяются в PHP-программе с помощью функции define( ). Например:

define("PI", 3.1415927);

После определения константа не может быть изменена. В имени константы обычно используются только заглавные буквы.

PHP имеет ряд предопределенных констант. Например:

FILE содержит имя файла, который в данный момент читает PHP;

LINE содержит номер строки этого файла.

Переменные

Имя любой переменной в PHP начинается со знака $. Имена переменных чувствительны к регистру символов.

Тип переменной не требуется задавать специально. Конкретный тип переменной устанавливается и меняется в ходе выполнения программы.

PHP поддерживает восемь типов данных: логический (принимает значения true или false); целое число; вещественное число с плавающей точкой; строка; объект; массив; ресурс (специальный тип); null (специальный тип).

Тип переменной можно проверить с помощью функции gettype().

Самостоятельно создайте в папке C:\WebServers\home\myserver.ru\www новый сценарий primer3.php и выполните код, приведенный в примере 3.

Пример 3. Использование переменных.

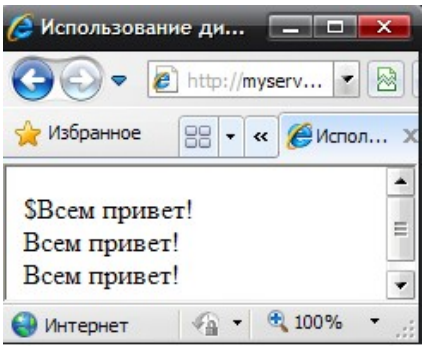
```
</head>
<body>
<?php
$name = "Вася Пупкин";
echo "Привет, $name <br>";
echo "Тип переменной - ".gettype($name)." <br>";
?>
</body>
</html>
```

Иногда для упрощения логики программы удобнее использовать переменные имена переменных. PHP предоставляет такую возможность в виде динамических переменных. Динамической называют переменную, имя которой хранится в ней самой.

Самостоятельно создайте в папке C:\WebServers\home\myserver.ru\www новый сценарий primer4.php и выполните код, приведенный в примере 4.

Пример 4. Использование динамической переменной

```
<html>
<head>
 <title>Использование динамической
переменной</title>
</head>
<body>
<?php
$t = "Всем";              // переменной t присваиваем
значение "Всем"
$$t = "привет!";          // Переменной "Всем"
присваиваем значение "привет!"
echo "$$t ".$$t."<br>";
echo "$t ${$t}<br>";
echo "$t $Всем";
```



```
?>
</body>
</html>
```

Операторы

Операторы PHP напоминают общеизвестные операторы языка Си.

Унарные операторы

-	Изменение знака на противоположный
!	Дополнение. Используется для реверсирования значения логических переменных
++	Увеличение значения переменной. Может применяться и как префикс, и как суффикс
--	Уменьшение значения переменной. Может применяться и как префикс, и как суффикс

Арифметические операторы

-	Вычитание
+	Сложение
*	Умножение
/	Деление
%	Остаток от деления

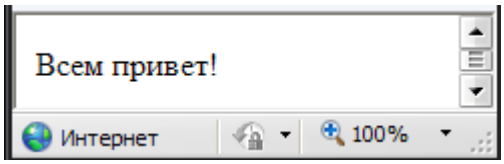
Оператор конкатенации

Оператор конкатенации "." присоединяет правую строку к левой.

Самостоятельно создайте в папке C:\WebServers\home\myserver.ru\www новый сценарий primer5.php и выполните код, приведенный в примере 5.

Пример 5. Конкатенация строк

```
<html>
<head>
 <title>Конкатенация строк</title>
</head>
<body>
<?php
$a = "Всем";
$b = $a." привет!";
echo $b;
?>
</body>
</html>
```



Оператор конкатенации обрабатывает операнды любых типов как строки. Результат его выполнения всегда является строкой.

Операторы присваивания

=	Присваивание
+=	Сложение ($n += 777; аналогично $n = $n + 777;)
-=	Вычитание ($n -= 777; аналогично $n = $n - 777;)
*=	Умножение
/=	Деление
%=	Остаток от деления
.=	Конкатенация ($n .= "777"; аналогично $n = $n."777";)

Битовые операторы

Битовые операторы позволяют изменять отдельные биты целых чисел.

&	И
|	ИЛИ
^	Исключающее ИЛИ

~	Инверсия
>>	Сдвиг вправо
<<	Сдвиг влево

Операторы сравнения

>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
==	Равно
===	Идентично
!=	Не равно
!==	Не идентично

Логические операторы

Логические операторы отличаются от битовых тем, что работают не с числами, а с логическими значениями: TRUE и FALSE.

and	И
or	ИЛИ
xor	Исключающее ИЛИ
!	Инверсия
>>	Сдвиг вправо
<<	Сдвиг влево
&&	И
||	ИЛИ

Логические операторы "И" и "ИЛИ" имеют два формата. Это не синонимы. Дело в том, что оператор or имеет приоритет ниже, чем ||, а and - ниже, чем &&. Таким образом, при построении сложных условных выражений можно обойтись без скобок. Однако, в таком способе указания порядка проще и запутаться.

echo(); - функция отвечающая за вывод значений переменных, во многом, аналогична функции **print().**

Практическая часть

Задание1. Установка веб-сервера

Ход работы.

Для изучения дисциплины и полноценной работы необходимо установить веб-сервер Apache (http://apache.org/) и интерпретатор языка программирования PHP (http://php.net/). Установка этого ПО достаточно сложное дело для начинающих пользователей (как это делается и для чего это необходимо можно прочитать здесь http://web.diwaxx.ru/web-server-doma.php), поэтому мы будем использовать готовую сборку Denwer (http://www.denwer.ru/). Инструкция по установке находится по адресу http://www.denwer.ru/base.html. Устанавливать Denwer необходимо на flash-накопитель, чтобы была возможность программировать как на занятиях, так и в домашних условиях. Устанавливайте Denwer в каталог первого уровня flash-накопителя, например f:\ WebServers. Под конец установки будет задан вопрос, как именно Вы собираетесь запускать и останавливать комплекс. Есть два альтернативных варианта:

1. Создавать виртуальный диск при загрузке машины (при этом, инсталлятор обеспечит, чтобы это происходило автоматически), а при остановке серверов виртуальный диск не отключать.

2. Создавать виртуальный диск только по явной команде старта комплекса. И, соответственно, отключать диск от системы — при остановке серверов.

Первый режим наиболее удобен, если комплекс устанавливается на жесткий диск компьютера, а не на flash-накопитель, поэтому необходимо выбрать второй вариант. На вопрос создавать ярлыки для запуска и остановки Denwer на рабочем столе отвечайте

отрицательно. Для запуска комплекса будем использовать файлы Run.exe, Stop.exe и Restart.exe в каталоге X:\WebServers\denwer\, где X:\WebServers\ - диск и папка, в которую установлен комплекс.

При запуске комплекса брандмауэр операционной системы может заблокировать запуск веб-сервера Apache. В этом случае брандмауэр потребует подтверждения Ваших намерений. Щелкните на Don't Block Anymore (не блокировать в дальнейшем).

Возможно, что на том компьютере, где Вы захотите запустить Denwer с Вашего flash-накопителя, Denwer не сможет создать виртуальный диск с той буквой, которую Вы задали при установке комплекса, т.к. в системе уже будет такой диск. В этом случае необходимо поменять букву виртуального диска, изменив в файле конфигурации X:\WebServers\denwer\CONFIGURATION.txt строку:

subst_drive = Z:

где необходимо поменять букву Z на любую другую, для которой в системе нет диска. Ни в коем случае не меняйте ничего больше в файле конфигурации, это может привести к нарушению работы комплекса.

Кроме Denwer Вам также потребуется текстовый редактор (не путать с текстовым процессором, подобным MS Word), желательно с подсветкой синтаксиса языков программирования и разметки, и современный веб-браузер с инструментами отладки. В качестве текстового редактора можно использовать Notepad++ (http://notepad-plus-plus.org/), а в качестве веб-браузера — Firefox с расширением Firebug. Их также необходимо установить на flash-накопитель.

Для проверки работы Denwer запустите файл X:\WebServers\denwer\Run.exe и наберите в браузере http://localhost. Если по этому адресу откроется служебная страница Denwer, значит все работает исправно, иначе идем на страницу http://www.denwer.ru/base.html и пытаемся разобраться с настройками сети и прокси-сервером. Для остановки Denwer выполните файл X:\WebServers\denwer\Stop.exe.

Задание 2. Создание собственного домена на локальном компьютере

После установки программы все необходимые для ее работоспособности файлы располагаются в папке C:\WebServers. Благодаря этому папку можно переносить с компьютера на компьютер, не нарушая работоспособности сервера.

Для создания собственного домена в рамках локального компьютера достаточно добавить новую папку с именем домена в каталог C:\WebServers\home. Например, самостоятельно создайте новый виртуальный хост myserver.ru (см. рисунок 1.1).

Файлы, которые должны находиться на этом сервере необходимо поместить в папку C:\WebServers\home\myserver.ru\www. При создании собственного домена следует учитывать нежелательность использования в имени домена букв кириллицы и специальных символов.

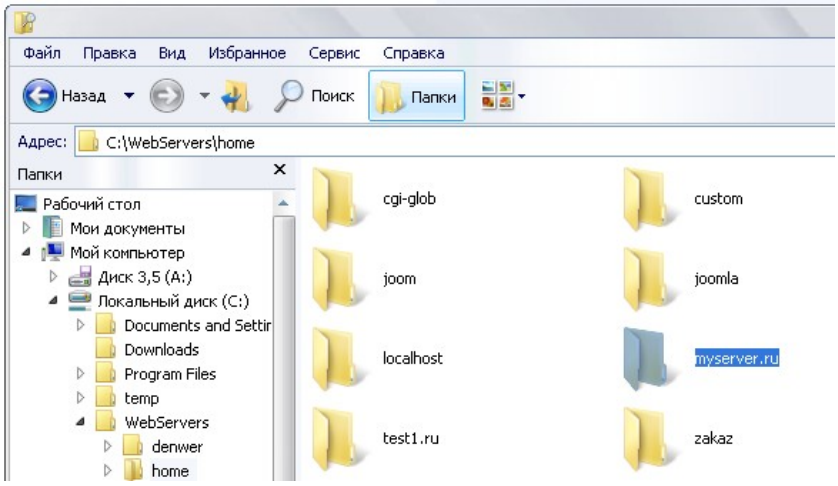


Рисунок 1.1 – Расположение доменов на Денвере

Для проверки работоспособности нового домена следует:

- перегрузить Денвер (сделать это можно с помощью ярлыка  Restart servers на рабочем столе или с помощью запускного файла C:\WebServers\Denwer\ Restart.exe);
- открыть в браузере страницу с адресом нового домена (например, http://myserver.ru). При этом в случае успешного запуска сервера на экране отобразится список файлов или содержимое файла с именем index. На большинстве сайтов при указании пути к домену или к папке отображается файл index.html или index.php или содержимое каталога, в случае если индексный файл отсутствует.
- Вариант когда новый поддомен не отображается в браузере:
- выполнить команду ping на новый домен в командной строке (ping myserver.ru). Если ответ получен, значит, Денвер с новыми настройками успешно запущен. Если ответа нет, то возможно:
    - а) Денвер не был перезапущен с момента добавления нового домена;
    - б) имя папки отличается от имени пингуемого домена;
    - в) внутри домена не создана папка www.
- если команда ping проходит, то возможно в браузере вы пользуетесь прокси-сервером для доступа в интернет. Например, в браузере Internet Explorer необходимо добавить ваш домен в список исключений. Для этого необходимо зайти в Свойства обозревателя - Подключения - Настройка сети - Дополнительно… в поле Исключения указать адрес вашего нового домена.
- еще раз проверьте, что имя домена содержит только латинские буквы и цифры, а так же не содержит пробелов.

После установки сервера и создания собственного домена можно приступать к созданию сайта.

Итак, у вас должен быть создан свой собственный домен под именем myserver.ru в нем должна быть обязательно создана папка www.

В папке www вашего домена (каталога) создайте файл index.html (именно с этого файла всегда начинается загрузка содержимого домена, если конечно, не указано что-то другое явно).

Откройте файл index.html с помощью блокнота. Запишите в него следующие строки:

Пример 1.

```
<html>
<head><title>Заголовок окна</title></head>
<body>Осваиваем web-технологии</body>
</html>
```

Сохраните файл. Перезапустите ссылку в поле адреса браузера (http://myserver.ru).
Проверьте в браузере результат.

Задание 3. Откройте текстовый редактор и наберите следующий код:

Листинг № 1.

```
<html>
<head><title>Вставка кода PHP</title></head>
<body>
<h1>Пример страницы с PHP кодом</h1>
<?
print(“<h2> PHP-фрагмент </h2>”);
?>
</body>
</html>
```

Или

Листинг 2

```
<?
```

```
print("<html><head><title>Вставка кода PHP</title></head><body>");
print("<h1>Пример страницы с PHP кодом</h1>");
print("<h2> PHP-фрагмент </h2>");
print("</body></html>");
?>
```

После этого выполните следующие действия:

1. Сохраните данный текстовый файл в каталоге **C:\WebServers\home\localhost\www\xp** (каталог **xp** необходимо создать) под именем **php_start1.php** (обратите внимание, расширение у файла .php).
2. Запустите web-сервер **Apache** и в строке адреса браузера наберите **http://localhost/xp**. Вы должны увидеть внутри виртуального каталога **xp** свой файл **php_start.php**.
3. Щелкните на нем и, если вы правильно набрали приведенный код, у вас должна загрузиться страница (рис 1.).

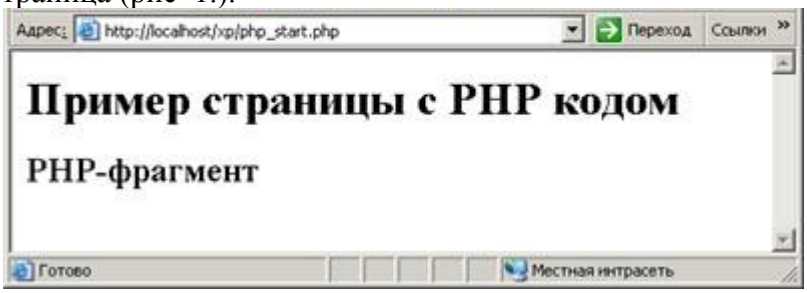


Рисунок 1.

4. Посмотрите html-код данной страницы (Контекстное меню → Просмотр HTML кода).

Задание 3. Далее представлены дополнительные упражнения для закрепления навыков создания простейших программ на языке программирования PHP. Выполните предложенные далее задания.

1. Выполните примеры из теоретической части.
2. Переменной $a необходимо присвоить значение 10, переменной $b присвоить значение 20. Выведите значения переменных на экран.
3. Затем переменной $c присвойте значение суммы этих переменных (переменной $a и переменной $b). Выведите значение переменной $c на экран.
4. Далее увеличьте значение переменной $c в три раза и выведите полученный результат на экран.
5. Разделите переменную $c на разность переменных $b и $a, выведите результат на экран.
6. Введите новые переменные $p и $b. Присвойте переменной $p значение «Программа», а переменной $b значение «работает».
7. Затем сложите переменные, содержащие эти слова («Программа» и «работает»), при этом слова должны быть разделены пробелом (‘ ‘). Результат необходимо присвоить переменной $result.
8. Далее с помощью оператора «.=» необходимо к строке «Программа работает» добавит слово «хорошо». Результат необходимо присвоить переменной $result.
8. Есть две переменные: $q = 5 и $w =7. Создайте скрипт, в результате выполнения которого эти две переменные «обмениваются» значениями переменная $q получает значение 7, переменная $w получает значение 5, при этом не создавая новых переменных (вариант $q = 7 и $w = 5 не рассматривается).

Содержание отчета

1. Цель
2. Этапы установки веб-сервера

3. Основы синтаксиса PHP
4. Программные листинги
5. Выводы

Контрольные вопросы

1. Назовите возможные элементы формы (изучите вопрос в Интернет).
2. В чем разница между методами формы GET и POST?
3. В какие массивы PHP передаются значения, посылаемые формами?
4. В чем заключается альтернативный синтаксис операторов PHP?
5. Как в код HTML вставляется значение, вычисляемое PHP?

Лабораторная работа № 6

Программирование базовых конструкций на PHP, обработка строк

Цель: освоить методы работы со строками с использованием строковых функций и регулярных выражений PHP

Задачи:

- научиться выделять требуемые части строк и использовать их для дальнейшей обработки;
- освоить методы замены выделенных фрагментов строк с использованием базовых строковых функций;
- создать собственную библиотеку по работе со строками;
- освоить методы преобразования нескольких строк в одну и обратно. Использовать функции преобразования строк для записи в гостевую книгу многострочных фрагментов;
- освоить методы разбиения строк на фрагменты и соединения фрагментов в строки.

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Строка – это набор символов. В PHP символ – это то же самое, что и байт, это значит, что возможно ровно 256 различных символов.

Строки в PHP можно определить тремя способами: одинарными кавычками, двойными кавычками и heredoc синтаксисом (многострочная строка).

```
<?php
print 'это простая строка';
print 'Также вы можете вставлять в строки символ новой строки таким образом, поскольку это нормально';
```

Обратите внимание: в строке присутствует одинарная кавычка. Для того, чтобы php понял, что это не конец строки, а часть ее, нам нужно ему об этом сообщить. Обратный слэш выполняет функцию экранирования.

```
print 'Это не вставит: \n новую строку';
```

Так же как и в одинарных, текст взятый в двойные кавычки это строка. Если строка определяется в двойных кавычках, либо при помощи heredoc, переменные внутри нее обрабатываются.

Если интерпретатор встречает знак доллара $, он захватывает так много символов, сколько возможно, чтобы сформировать правильное имя переменной. Если вы хотите точно определить конец имени, заключайте имя переменной в фигурные скобки.

```
<?php
$beer = 'Heineken';
print "$beer's taste is great"; // работает, "'" это неверный символ для имени переменной
print "He drank some $beers"; // не работает, 's' это верный символ для имени переменной
print "He drank some ${beer}s"; // работает print "He drank some {$beer}s"; // работает
?>
```

Heredoc-текст ведет себя так же, как и строка в двойных кавычках, при этом их не имея. Это означает, что вам нет необходимости экранировать кавычки

В heredoc.

```
<?php
print <<<HEREDOC1
Меня зовут "$name". Я печатаю very$fast.
Теперь я вывожу very{$fast}.
Это должно вывести заглавную букву 'A': \x41
HEREDOC1;
?>
```

В PHP есть два оператора для работы со строками. Первый — оператор конкатенации ('.'), который возвращает объединение левого и правого аргумента в одну строку. Второй — оператор присвоения вместе с конкатенацией.

```
<?php
$a = "Hello, ";
$b = $a . "world!"; // $b содержит строку "Hello, world!"
$a = "Hello, "; $a .= "world!";
 // $a содержит строку "Hello, world!"
?>
```

Все функции обработки строк можно условно разделить на 3 функциональные группы:

- поиск и выделение искомого фрагмента, или подстроки;
- поиск фрагментов с заменой их на новые фрагменты;
- функции, связанные с разделением и соединением строк.

Поиск вхождения решается с использованием функций **strpos(), strrpos, strstr(), strchr(). Stristr ().**

Поиск некоторой подстроки в заданной строке символов с использованием функции strpos(). Синтаксис strpos():

```
strpos (<исходная строка>,<строка для поиска > [,с какого символа искать])
```

Третий параметр является необязательным. Если он не указан, то поиск проводится с первого символа искомой строки. Функция возвращает номер символа, с которого поисковая строка входит в исходную строку или возвращает логическое FALSE, если вхождение не найдено. Кроме логического FALSE эта функция может возвращать и другие значения, которые приводятся к FALSE (например, 0 или ""). Поэтому для того, чтобы проверить, найдена ли искомая строка, рекомендуют использовать оператор эквивалентности «===». Исходная строка рассматривается как массив символов, поэтому по правилам нумерации массивов символы нумеруются с 0 (нуля). Это может привести к некорректности, если поисковый фрагмент расположен в самом начале исходной строки, т. к. номер возвращаемой позиции будет равен 0. А 0 в свою очередь интерпретируется как FALSE. Для того чтобы предотвратить это, необходимо обработать условие по эквивалентному оператору:

```
$ s =”Регулярные выражения являются наиболее мощным инструментом поиска и замены одних символов на другие.”;
$ fs =”Регулярные”;
$pos = strpos($s,$ fs);
if ($ pos !== false) echo "Искомая строка встречена в позиции номер $pos ";
else echo "Искомая строка не найдена";
```

Если мы применим простое сравнение !=, то в нашем случае получим ложный результат. Если вместо поисковой строки мы поставим числовой код, то интерпретатор будет считать, что мы ищем символ, код которого задан в качестве поиска. Функция strpos () находит первое вхождение подстроки. Противоположная функция strrpos (исходная строка, символ для поиска) позволяет найти последнюю позицию искомого символа. Однако для подстроки данная функция не работает, только для одиночного символа, который может быть задан как кодом, так и символом. Однако чаще возникает задача

не просто поиска вхождения подстроки, а выделения фрагмента, начиная с заданной подстроки.

Эту задачу решает функция **strstr** (исходная строка, строка для поиска).

Она находит первое появление искомой строки и возвращает подстроку, начиная с этой искомой строки до конца исходной строки. Если строка для поиска не найдена, то функция вернет FALSE. Функция чувствительна к регистру. Вместо strstr() можно использовать абсолютно идентичную ей функцию strchr(). Для реализации регистронезависимого поиска подстроки существует соответствующий аналог этой функции — функция stristr (исходная строка, искомая строка).

Выделение подстроки осуществляет функция strstr, ее формат strstr (исходная строка, строка для поиска) Она находит первое появление искомой строки и возвращает подстроку, начиная с этой искомой строки до конца исходной строки. Если строка для поиска не найдена, то функция вернет FALSE. Выделение подстроки по заданному номеру конечного символа и длине фрагмента выполняет функция:

substr (исходная строка, позиция начального символа [, длина])

Вычислить длину строки можно с помощью функции **strlen** (строка).

Практическая часть

Задание 1

Написать скрипт, который в переданной строке выделяет словосочетание «прикладная информатика». Если словосочетание найдено в начале символьной строки, то возвращается сообщение: «Найдено в начале текста», если найдено в конце текста, то выводится сообщение, что «Найдено в конце текста», если найдено в середине, то выводится соответствующее сообщение. В противном случае необходимо вывести сообщение «Не найдено». Поиск оформить в виде отдельной функции, которой передается символьная строка для анализа и искомая строка, в нашем случае «прикладная информатика». Разработать 3 варианта использования функции:

- задавать анализируемую строку в виде символьной константы;
- передавать строку как параметр при вызове скрипта;
- вводить строку в виде поля textarea в диалоговую форму и передавать ее на сервер. При анализе на нахождение поискового выражения в конце исходной строки использовать функцию strlen (), которая вычисляет длину строки.

Задание 2

Модифицировать задание 1. Разработать диалоговую форму, в которой задать 2 поля — одно текстовое длиной 40 символов, второе типа textarea. В первом поле задавать искомый фрагмент, во втором — текст, в котором надо проводить поиск. Результатом анализа должен быть текст, определяющий место нахождения заданной строки в тексте.

Задание 3

Написать скрипт, который ищет заданный фрагмент текста в текстовом файле. В результате обработки сформировать сообщение, в котором сообщить, в какой строке файла найден заданный фрагмент текста. Если фрагмент не найден, то выдать соответствующее сообщение. Модифицировать задание таким образом, чтобы в исходной форме передавался не только искомый текст в виде поля, но и имя файла, в котором надо найти указанный фрагмент.

Задание 4

Очень часто при обработке строк требуется узнать, заканчивается ли введенная строка символом перевода строки. Если строка при вводе содержит данный символ, то не требуется ее дополнять символом «перевод строки», например при записи в файл. Если мы этого не сделаем, то получим в файле пустую строку, что нежелательно. Для решения данной задачи нам потребуется функция ord () — она возвращает ASCII код символа. Код символа «перевод строки» равен 10. Теперь надо научиться выделять последний символ строки. Это можно сделать, используя две функции:

Strlen ($ str) – определяет длину строки в символах;

Substr ($ str,<начальный символ>,<конечный символ>) — выделяет подстроку из строки $ str с начального по конечный символ. Объединим теперь наше решение в следующий скрипт:

```
<? $str = $_POST['str']; // переданная строка $l=strlen($str); $s=substr($str,$l-1,$l);
if (ord ($ s) == 10) { echo "переданная строка \$ str – содержит в конце символ перевод строки.\ n "; } ?>
```

Для того, чтобы определить, нет ли в начале данного символа, надо выбрать первый символ и сравнить с кодом.

```
if (ord(substr($str,0,1)) == 10) { echo "первый символ \$str - перевод строки.\n"; }
```

Для просмотра и замены символов по всей строке применяется функция **strtr** (string str, string from, string to) — замена указанных символов по всей строке. Каждый встретившийся символ из части from заменяется на соответствующий символ из части to.

Подготовить форму ввода, которая содержит управляющий элемент textarea. Вводимую строку проверить на наличие символов перевода строки в любом месте и удалить все символы. Перед записью в файл добавить в конец строки символ перевода и занести в таком виде запись в файл. Проверить работу скрипта построчным выводом файла (см. предыдущую работу). Для этого можно использовать следующую функцию:

```
<?
Function print_file($f)
{
$file_array = file($f); while (list($line_num, $line) = each($file_array)) print "<b>Line $line_num: </b> ". htmlspecialchars($line)."<br>\n";
}
?>
```

Задание 5

Часто при занесении информации в базу данных или в файл в ней могут присутствовать специальные символы, которые при обработке могут интерпретироваться как управляющие символы. При записи информации необходимо обеспечить сохранение всех этих символов, но без исполнения соответствующих им команд. При предъявлении пользователям сохраненной информации необходимо восстановить ее в прежнем виде, т. е. со всеми специальными символами.

Для решения поставленной задачи необходимо использовать две функции PHP по обработке строк, которые имеют противоположное функциональное назначение:

- функция addslashes — экранирует спецсимволы в строке;
- функция stripslashes — убирает экранирующие слэши.

Однако обработка с использованием этих функций может зависеть от опции настройки сервера.

Если get _ magic _ quotes _ gpc ON, то экранирование всех спецсимволов происходит автоматически при вводе данных через протоколы GET и POST.

Если же в конфигурационном файле состояние

```
get _ magic _ quotes _ gpc OF
```

то экранирование необходимо делать перед записью данных в файл или в БД. Изменить состояние данной настройки в процессе работы скрипта невозможно, однако для того, чтобы избежать двойного экранирования, можно проанализировать существующие настройки. Для этого служит функция с одноименным названием get _ magic _ quotes _ gpc () — она возвращает значение 1, если опция автоматического экранирования спецсимволов включена и 0 в противном случае. С учетом всего вышесказанного, разработайте набор из 3 файлов:

- HTML-форма для ввода произвольной символьной строки, которая может иметь простой вид:

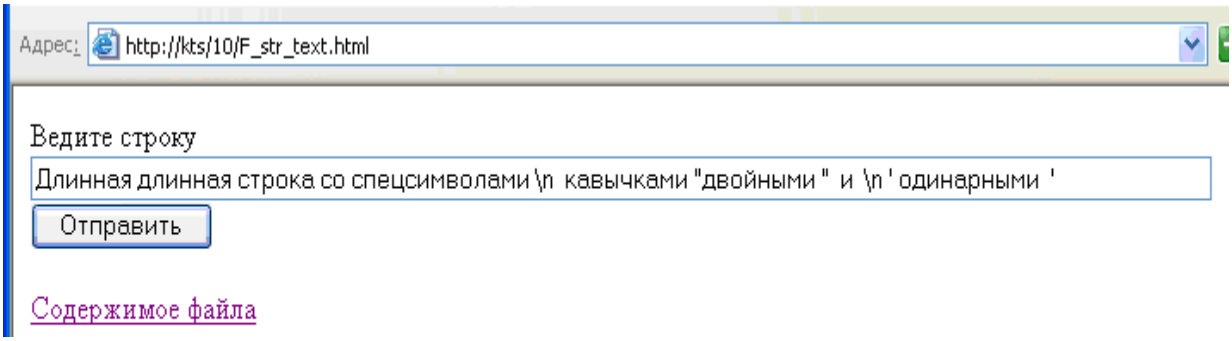


Рис. 1. Внешний вид формы с типовым текстовым полем ввода

- Обработчик, который анализирует системные настройки и экранирует спецсимволы, если это требуется, а затем записывает их в текстовый файл. Запись проводить в конец файла, обязательно дополнить конец записи спецсимволами «\ n» перед занесением в файл. Текстовый файл открывать в режиме дозаписи a +. После записи в файл управление передать исходной форме ввода информации.
- Третий файл — это программа построчного вывода заполняемого файла. В форме сделать на нее ссылку. Если вы оформили в предыдущем задании функцию построчного вывода, то вам надо только передать в эту программу имя файла, который надо вывести на печать. Однако для того, чтобы решалась поставленная задача, желательно в программе построчного вывода файла убрать внесенные слэши. Эту обработку мы добавим в функцию:

```
Function print_file($f)
{
$file_array = file($f); while (list($line_num, $line) = each($file_array))
{
$line=Stripslashes($line); // убрать экранирующие слэши. print "<b>Line $line_num: </b> ".
htmlspecialchars($line)."<br>"; }
}
```

Содержимое файла будет:

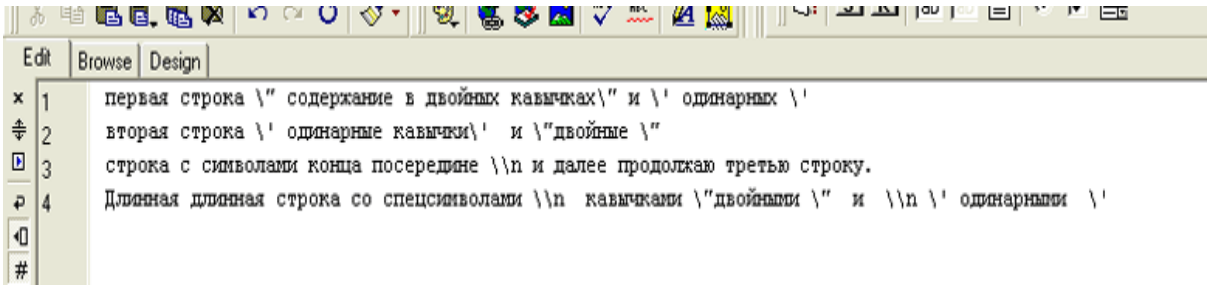


Рис. 2. Содержимое текстового файла с занесенными экранирующими слэшами

А при выводе мы должны получить следующий текст:

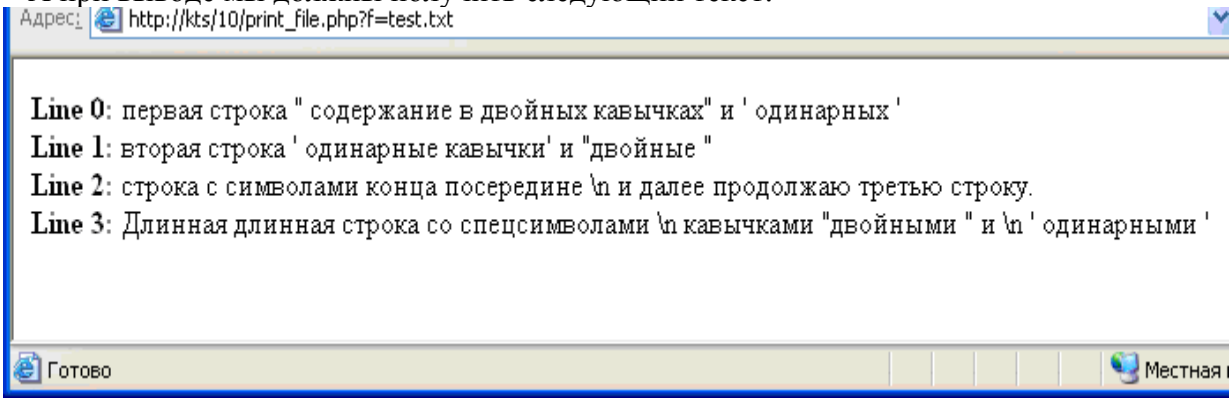


Рис. 3. Вывод содержимого файла с обратным преобразованием — удалением вставленных экранирующих слэшей

Проверить работоспособность созданных элементов для разных строк с различными спецсимволами.

Разработать новую форму, в которой простое текстовое поле ввода заменить на textarea. Скопируйте первую форму и измените там тип поля, все остальное оставьте без изменений.

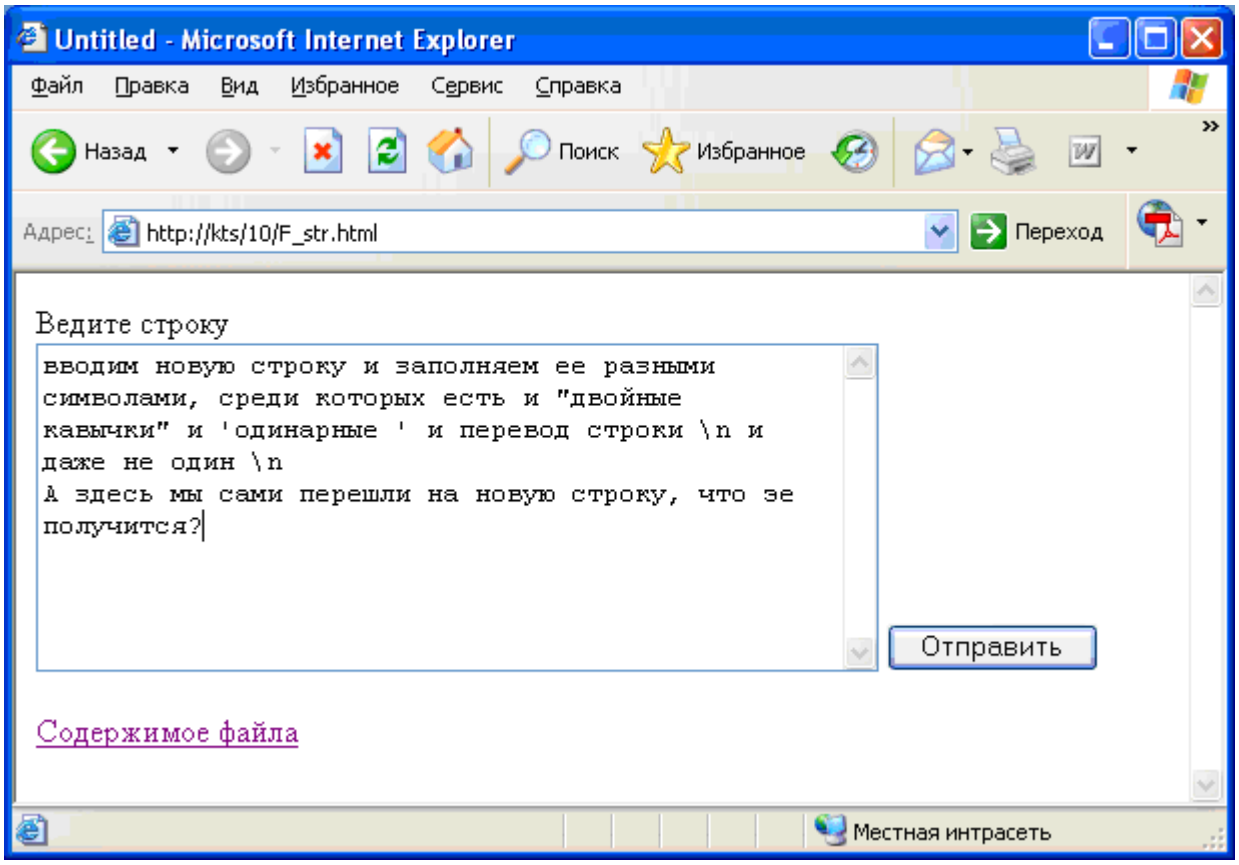


Рис. 4. Форма ввода с полем типа textarea

Исследуйте поведение данного типа передаваемых данных, возможно ли здесь экранирование спецсимволов? Можем ли мы весь введенный текст вытянуть в одну строку?

Поиск фрагментов с заменой их на новые фрагменты

Для замены символов или строк используется функция strtr (). При этом длины строк искомой и заменяемой должны быть одинаковые, в противном случае замены происходят по минимальной длине. Если вы хотите заменить все символы ‘\ n ’ на пробелы, то менять можно только на 2 пробела, при замене на один пробел останется символ n.

Однако у нас существует целый набор символов, которые необходимо искать и заменять. Конечно, для этого можно использовать функцию strtr (). Эта функция предполагает возможность замены сразу группы символов на группу символов. Можно условно представить формат этой функции следующим образом:

get _ magic _ quotes _ gpc OF

Strtr ($ str, from, to) — и каждый символ из группы from имеет эквивалент в группе to, и на него заменяется при поиске.

В этом случае искомые символы и символы замены объединяются в массив, в котором ключами являются искомые символы, а значениями — символы или строки замены.

Например, поставим задачу нахождения всех явных и неявных символов конца строк и замену их на символы * или ** в соответствии с длиной последовательности.

Символы конца текста и конца строки невидимы, их тяжело обнаружить, для того чтобы их найти, напишите простую тестирующую программу, которая выводит коды всех введенных символов. Для этого нам надо воспользоваться функцией ord ($ s), которая

возвращает код символа, который является ее аргументом. Символьную строку можно просматривать посимвольно, для этого применим следующий синтаксис:

$ s =$ str {10} — здесь $ s — символ под номером 10.

Нумерация символов в строке начинается с 0 (нуля). Поэтому фактически это 11 символ строки. Вместо константы можно подставить переменную. Имея функцию, которая определяет длину строки — srtlen ($ str), можно приступить к посимвольному выводу кодов символов, для того чтобы выявить невидимые символы перехода на новую строку.

Задать эти символы мы сможем только через интерфейсную форму с элементом textarea. Остальные элементы вводят всегда только одну строку. Воспользуйтесь формой с элементом ввода типа textarea, которую вы создали в предыдущем задании. Скопируйте ее и сохраните под новым именем.

Текст обработчика должен выводить посимвольно коды введенной строки. Для того чтобы вам было удобнее анализировать, резонно выводить заодно и сам символ и делать это в виде таблицы, пригодной для визуального анализа. Оформить лучше всего данное задание в виде функции:

```
35
36  function str_kod($str)
37  {// вывод кодов введенной строки
38  print '<table border=1><tr><td>Номер пп</td><td>символ</td><td>Код</td></tr>';
39  $len=strlen($str);
40  for ($i=0;$i<$len;$i++)
41  { $s=$str{$i};
42  print '<tr><td>'.$i.'</td><td>'.$s.'</td><td>'.ord($s).'</td></tr>';
43  }
44  print '</table>';
45  }
46  ?>
```

Рис. 5. Функция вывода кодов символов переданной строки

Если теперь ввести некоторую последовательность строк в исходную форму:

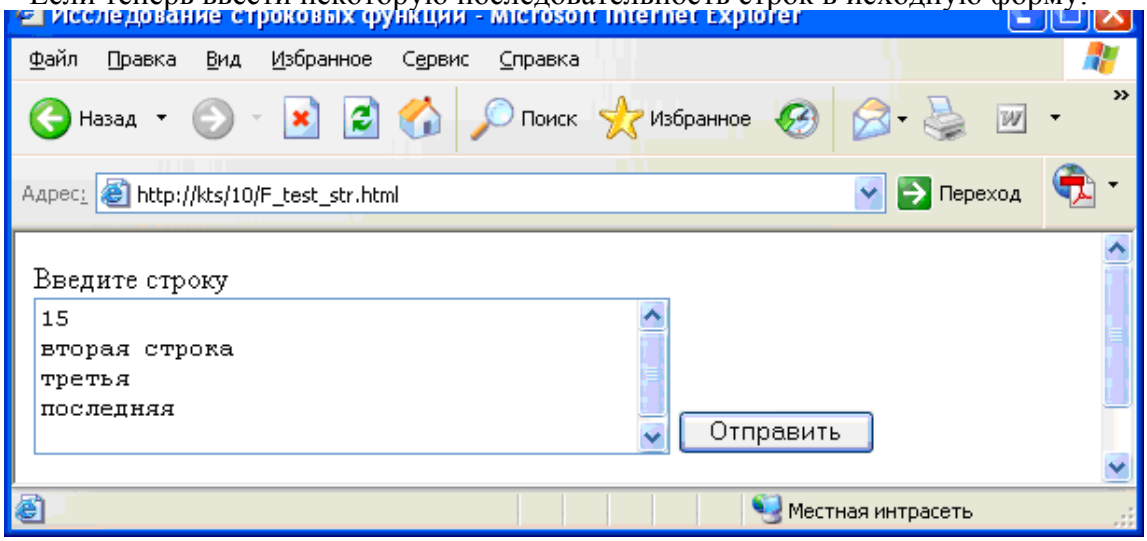
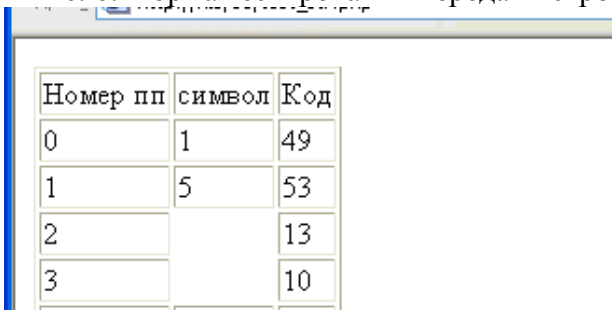


Рис. 6. Форма тестирования передачи строк



Номер пп	символ	Код
0	1	49
1	5	53
2		13
3		10

Рис. 7. Результат работы функции вывода кодов

По результирующей таблице (см. рис. 7) можно обнаружить, что невидимыми кодами перехода на новую строку является совокупность двух символов с кодами 13 и 10 соответственно. При занесении записи в файл, если мы не уберем или не заменим на другие эти невидимые символы перехода на новую строку, операция записи информации занесет в файл столько строк, сколько переходов на новую строку в нем было, в нашем случае — 4 строки. Однако очень часто требуется из всех введенных строк сформировать только одну и ее записать в файл. Это можно сделать, применив функции замены.

В PHP существует 2 функции замены — strtr (), которая позволяет заменять символы на символы в исходной строке, и функция str _ replace (), которая позволяет заменять совокупности символов на совокупности символов.

Функция strtr () позволяет заменить символы на пустую строку.

```
Function str_p($str)
{
$sk=chr(13); $sn=chr(10);
$ s = array ($ sk =>",$ sn =>"); // заменяем на 2 подряд идущие // одинарные кавычки
$str=strtr($str,$s); Return $str;
}
```

Если мы теперь снова запустим функцию печати кодов, то увидим, что действительно все неявные переходы на новую строку исчезли. Однако в некоторых случаях нам важно сохранить первоначальную разметку текста, которую нам передал пользователь и все-таки записать всю информацию только в одну запись. В этом случае резонно разработать совокупность из двух функций: одну — для прямого преобразования и замены символов перехода на новую строку на символ ‘*’, а вторую — наоборот замены двух звездочек ‘**’ на тег конца строки в HTML-формате — ‘< br >’.

При выполнении первой задачи мы также можем использовать функцию strtr (), но т. к. у нас несколько исходных искомых символов для замены, то нам требуется задать их в виде массива. Это альтернативный синтаксис функции, он предполагает наличие всего 2 параметров у функции: первый — исходная строка, а второй — массив, где ключами случат искомые символы, а значениями — символы замены.

В этом случае код функции может выглядеть следующим образом:

```
46  }
47  //
48  // функция замены переходов на новую строку.
49  Function str_n($str)
50  {
51  $sk=chr(13);
52  $sn=chr(10);
53  $s=array($sk=>'*',$sn=>'*');
54  $str=strtr($str,$s);
55  Return $str;
56  }
```

Рис. 8. Функция замены перехода на новую строку символами * Обратная функция заменяет ‘**' на ‘< br >'

Задание 5

Создать библиотеку дополнительных функций обработки строк, в которую войдут:

- функция вывода кодов символов переданной строки;
- функция замены символов перехода на новую строку на пустую строку;
- функция замены символов перехода на новую строку на символ * ;
- функция замены ‘**’ на ‘< br >’.

Сохранить все разработанные функции в файле str. inc. Отладить работу всех функций, вводя строки через интерфейсную форму с полем textarea. Обработчик формы должен содержать операцию подключения библиотечного файла INCLUDE (‘str.inc’);

Задание 6

Воспользуйтесь формой с элементом ввода типа textarea, которую вы создали в предыдущем задании. Обеспечьте запись вводимой информации в текстовый файл. Дополните предварительную обработку введенной информации перед записью ее в файл заменой символов неявного конца строк символами ‘*’ и символов явного конца строки, обозначаемых как ‘\ n ’ двумя звездочками ‘**’.

Преобразованную строку запишите в файл. Программа просмотра файла остается прежней. Работа созданного комплекса приведена на следующих рисунках.

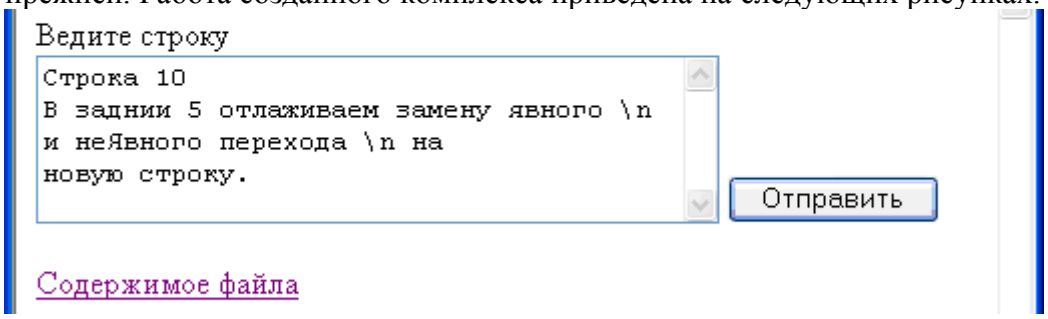


Рис. 9. Исходная информация, представленная несколькими строками

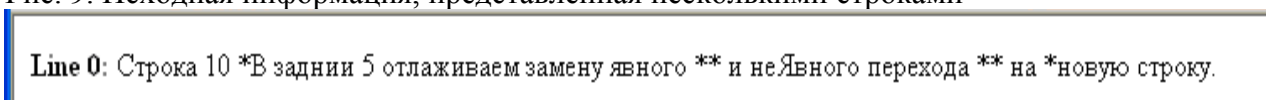


Рис. 10. Полученная подобная запись в текстовом файле

Задание 7

Теперь мы можем расширить наше задание с гостевой книгой и использовать форму не с текстовым однострочным полем для ввода сообщения, а форму с полем типа textarea, но перед занесением сообщения в файл преобразовать его с использованием рассмотренного ранее алгоритма замены символов конца строки.

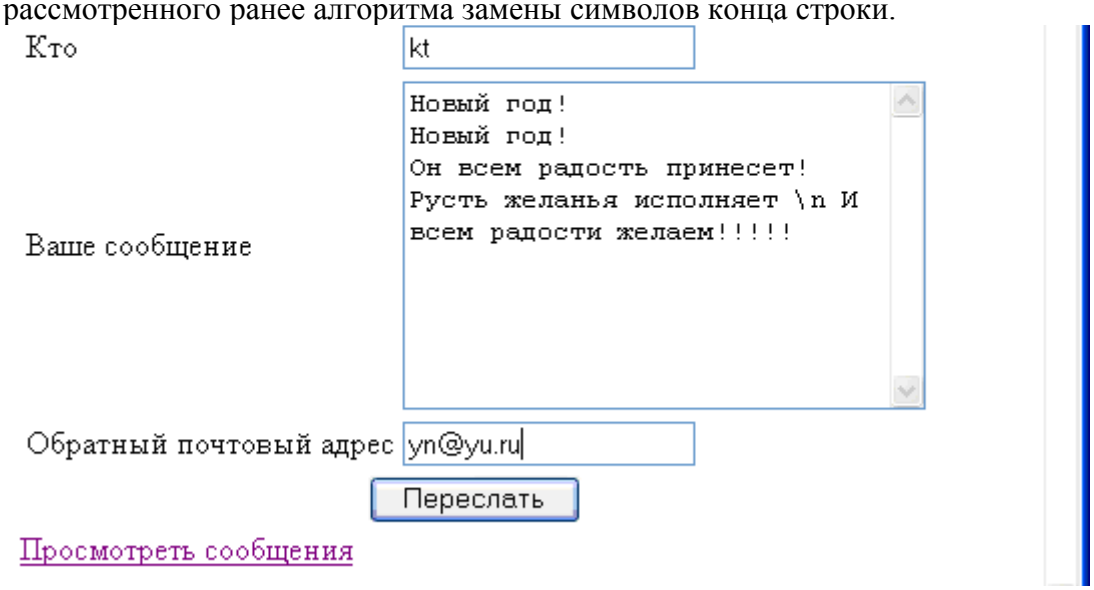


Рис. 11. Измененная форма для ввода сообщений в гостевую книгу

Если наш пользователь оставил такое сообщение, то при просмотре оно будет выглядеть следующим образом:

Кто	Сообщение	Обратная связь	Когда
kts	Очень важное сообщение	kts@ibi.metrocom.spb.ru	1.12.2207
nkt	Приветствую всех, рада видеть и читать, интересно.	ntr@mail.ru	2.12.2007
ivv	Не порядок, все надо делать иначе	ivv@yandex.ru	2.12.2007
lam	Все хорошо, скоро новый год	lam@yandex.ru	2.12.2007
uyt	последнее китайское предуреждение	rte@list.ru	3.12.2007
Gur	Изменился почтовый адрес	rte@list.ru	3.12.2007
klkl	Хорошо бы вывесить фото с последнего похода	rte@list.ru	4.12.2007
kkk	последнее сообщение скоро новый год???	uy@yandex.ru	14.12.2007
ttt	Новый год! *Поздравляем	kk@yu.ru	14.12.2007
kt	Новый год! *Новый год! *Он всем радость принесет! *Русть желанья исполняет ** И всем радости желаем!!!!!	yn@yu.ru	14.12.2007

Рис. 12. Просмотр сообщений при замене символов перевода строки на **

Вместо символов ‘**’ мы можем поставить пробелы, и тогда наша программа просмотра сообщений гостевой книги будет работать без изменений.

Однако если предположить, что наш гость поставил символы перехода на новую строку с некоторым умыслом, ему было удобно именно таким образом отформатировать сообщение, то мы можем восстановить начальное форматирование, использовав обратную замену. Мы теперь перед выводом можем заменить все символы звездочка на символы < br > — перехода на новую строку в формате HTML.

Воспользуемся уже разработанной ранее функцией обратной замены ‘**’ на ‘< br >’, тогда можем получить следующий результат в той же самой программе просмотра содержимого гостевой книги:

Кто	Сообщение	Обратная связь	Когда
kts	Очень важное сообщение	kts@ibi.metrocom.spb.ru	1.12.2207
nkt	Приветствую всех, рада видеть и читать, интересно.	ntr@mail.ru	2.12.2007
ivv	Не порядок, все надо делать иначе	ivv@yandex.ru	2.12.2007
lam	Все хорошо, скоро новый год	lam@yandex.ru	2.12.2007
uyt	последнее китайское предуреждение	rte@list.ru	3.12.2007
Gur	Изменился почтовый адрес	rte@list.ru	3.12.2007
klkl	Хорошо бы вывесить фото с последнего похода	rte@list.ru	4.12.2007
kkk	последнее сообщение скоро новый год???	uy@yandex.ru	14.12.2007

Рис. 13. Просмотр гостевой книги при возможности многострочной записи

Задание 8

Часто мы сталкиваемся с необходимостью обработки некоторой стереотипной информации. В этом случае мы так же можем применить функцию str_replace(). Эта функция может работать не только со строкой, но и с многомерным массивом.

str_replace(искомое значение, значение для замены, объект).

Функция str_replace() ищет в рассматриваемом объекте значение и заменяет его значением, предназначенным для замены. Почему мы говорим здесь не про строки для поиска и замены и исходную строку, а про значения и объект, в котором происходит замена? Дело в том, что начиная с PHP 4.0.5, любой аргумент этой функции может быть массивом. Если объект, в котором производится поиск и замена, является массивом, то эти действия выполняются для каждого элемента массива и в результате возвращается новый массив.

Если искомое значение и значение для замены — массивы, то берется по одному значению из каждого массива и производится их поиск и замена в объекте. Если значений для замены меньше, чем значений для поиска, то в качестве новых значений используется пустая строка.

Если значения для поиска — массив, а значение для замены — строка, то эта строка будет использована для замены всех найденных значений.

Функция str_replace() чувствительна к регистру, но существует ее регистронезависимый аналог — функция str_ireplace(). Однако надо помнить, что функция str_ireplace() не работает с символами кириллицы.

Еще один пример использования функции str_replace() — обработка шаблонов. Шаблоны — это эффективный метод формирования некоторых конкретных строк из заданного шаблона. Например, у нас есть шаблон описания выступления на некоторой конференции. В этот шаблон включаются следующие данные:

- название выступления (доклада);
- пол;
- ФИО участника.

Во время регистрации участники должны передать нам эти данные, а мы должны сформулировать ответные подтверждающие письма, которые направим тем, кто зарегистрировался и передал нам эти данные. Мы введем 3 шаблона — это условные строки, которые будем заменять на реальные данные.

- <! приветствие > — шаблон обращения
- <! участник >
- <! доклад >

Зададим текст общего сообщения, в который надо будет подставить конкретные данные:

```
$ ss =”<!приветствие> <!участник> приглашаем Вас принять участие в нашей конференции ‘Методы управления качеством образования в ВУЗе', проходящей в период с 12 по 15 марта 2008 года и выступить с докладом ‘<!доклад>’”;
```

Пусть данные о конкретном участнике находятся в соответствующих переменных:

```
$ fio – фамилия участника, $ sex – пол, $ doclad – название доклада.
Тогда код формирования конечного сообщения может быть представлен следующим образом:
If($sex=="м")$ss=str_replace("<!приветствие>","Уважаемый господин ",$ss);
else $ss=str_replace("<!приветствие>","Уважаемая госпожа ",$ss);
$ss=str_replace("<!приветствие>","Уважаемый господин ",$ss);
$ss=str_replace("<! участник >",$fio,$ss);
$ss=str_replace("<! доклад >",$doclad,$ss);
print $ss;
```

Формулировка задачи

Подготовьте форму для регистрации участника конференции. В этой форме должны быть заданы:

- название конференции: Информационные технологии, Информационные технологии в управлении;
- название секции: Прикладная информатика, Информационный менеджмент, Корпоративные ИС, Банковские ИС;
- вид участия: докладчик на секции, докладчик на пленарном заседании, гость;
- ФИО участника;
- пол участника.

После регистрации каждый участник должен получить информационное сообщение с приглашением на заданную конференцию и предложением выступить с соответствующим докладом на пленарном заседании или на соответствующей секции, или в качестве гостя. При регистрации предусмотреть возможность не указывать секцию

и название доклада, тогда этот участник интерпретируется как гость, если же доклад указан, а секция нет, то считается, что он предлагает доклад на пленарном заседании.

Формирование собственно сообщения оформить как функцию.

Разделение и соединение строк

Разделение строк на элементы может быть выполнено с использованием функций:

```
explode(разделитель,исходная строка [,максимальное число элементов])
split (шаблон, исходная строка [, максимальное число элементов])
preg_split (шаблон, исходная строка [, максимальное число элементов [,флаги]]).
```

Две последние используют регулярные выражения и будут рассмотрены в следующем практикуме.

Функция explode() делит исходную строку на подстроки, каждая из которых отделена от соседней с помощью указанного разделителя, и возвращает массив полученных строк. Если задан дополнительный параметр «максимальное число элементов», то число элементов в массиве будет не больше этого параметра, в последний элемент записывается весь остаток строки. Если в качестве разделителя указана пустая строка «""», то функция explode() вернет FALSE. Если символа разделителя в исходной строке нет, то возвращается исходная строка без изменений.

Допустим, у нас есть строка слов, отделенных друг от друга запятыми. Мы хотим написать скрипт, который эту строку преобразует в элемент «выпадающий список» для формы. Тогда мы можем следующим образом использовать функцию explode ():

```
$ spisok =”значение1,значение2,значение3,значение4”;
Echo "<Form action=str_5_1.php method=POST>";
$names = explode(",",$spisok);
$s = "<select name=elem>";
// создаем выпадающий список
foreach ($names as $k => $name) { $s.= "<option value=$k>$name";
}
$s.= "</select>";
echo $s;
echo “<input type=submit value=Передать> </form>”;
```

Мы получим следующую форму:

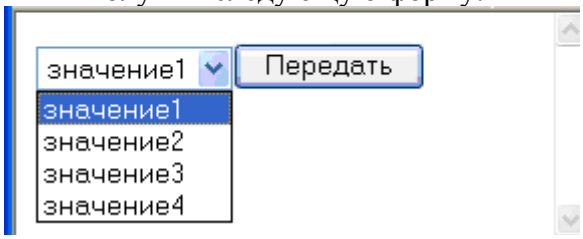


Рис.14. Пример создания выпадающего списка из строковой переменной

Кроме *разделения строки на части* иногда, наоборот, возникает необходимость *объединения нескольких строк в одно целое*. Функция, предлагаемая для этого языком PHP, называется *implode()* :

```
implode (массив строк, объединяющий элемент)
```

Эта функция объединяет элементы массива с помощью переданного ей объединяющего элемента (например, запятой). В отличие от функции *explode()*, порядок аргументов в функции *implode()* не имеет значения.

Допустим, мы храним имя, фамилию и отчество человека по отдельности, а выводить их на странице нужно вместе. Чтобы *соединить их в одну строку*, можно использовать функцию *implode()*:

```
<?php
$data =
array("Иванов","Иван","Иванович");
$str = implode($data," ");
echo $str;
?>
```

Задание 9

Создать 3 файла: f 1. inc, f 2. inc, f 3 inc.

Каждый файл должен содержать описание 4 строковых переменных:

$ s — список названий товаров;

$ sek — список названий способов доставки;

$ city — список городов;

$ mon — список способов оплаты.

Задать разное содержание всех переменных в файлах f 1—f 3.

Написать скрипт, формирующий форму заказа товара, выбор товара, способ доставки, город и способ оплаты формировать в виде выпадающих списков.

Обеспечить работоспособность формы с разными файлами. Файлы f 1—f 3 подгружать командой INCLUDE.

Задания для самостоятельной работы

Задание 1

У Вас имеется диалоговая форма заказа товаров (например, рис. 15.)

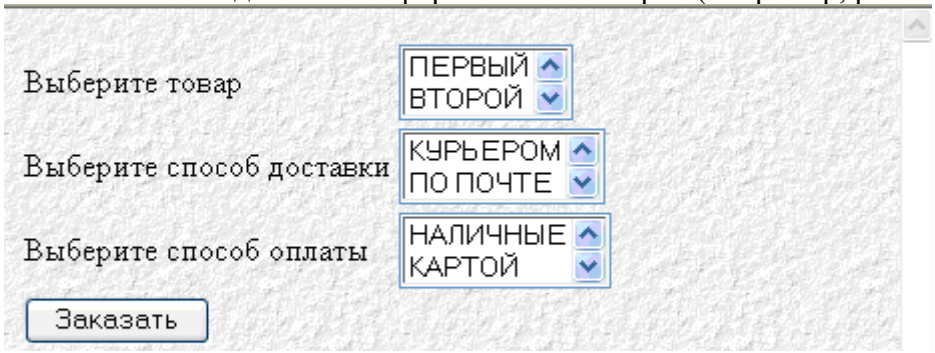


Рис.15. Диалоговая форма заказа товара

Напишите скрипт, который выводил бы форму на выбранном языке: русском или английском. Примените механизм шаблонов.

Создайте 2 массива, которые будут содержать список всех шаблонов, используемых в форме, и их перевод: один на русский язык, а второй массив — на английский. Например:

```
$ ar = array (<! text 1=>”Выберите товар”,<! text 1.1>=>”Первый” …);
```

Язык задавать в виде параметра запуска скрипта. Массивы с переводом строк на конкретный язык пригружать командой INCLUDE. Для замены шаблонов на текст на заданном языке применить функцию str _ replase ().

Задание 2

Написать скрипт, который извлекал бы из HTML-файла все рисунки и записывал бы их в отдельный текстовый файл в следующем формате: название рисунка, в качестве которого берется подрисуночная надпись, потом знак разделителя, например % и потом имя файла с рисунком.

Информация о каждом рисунке хранится в отдельной строке файла.

Информация дописывается в конец текстового файла.

Содержание отчета

1. Цель
2. Основные функции для обработки строк на PHP

3. Программные листинги
4. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 7

Программирование ветвлений и циклов на PHP

Цель: отработать навыки программирования ветвлений и циклов на PHP.

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Условные операторы

В языке PHP два условных оператора: if и ?.

Применение **условного оператора** аналогично использованию в других языках программирования. Синтаксис полностью идентичен языку Си.

Существует три типа оператора if. Первый тип - базовый условный оператор. Например:

```
if ($a > $b)
  {
  echo "A больше Б";
  }
```

Альтернативный синтаксис оператора if:

```
if ($a > $b):
  echo "A больше Б";
endif;
```

Второй тип - условный оператор if-else. Например:

```
if ($a > $b)
  {
  echo "A больше Б";
  }
else
  {
  echo "A не больше Б";
  }
```

Альтернативный синтаксис оператора if-else:

```
if ($a > $b):
  echo "A больше Б";
else:
  echo "A не больше Б";
endif;
```

Третий тип - условный оператор if-elseif. Например:

```
if ($a > $b)
  {
  echo "A больше Б";
  }
elseif ($a == $b)
  {
  echo "A равно Б";
  }
else
  {
```

```
  echo "А меньше Б";
  }
```
Альтернативный синтаксис оператора if-elseif:
```
if ($a > $b):
  echo "А больше Б";
elseif ($a == $b):
  echo "А равно Б";
else:
  echo "А меньше Б";
endif;
```
Условный оператор ? возвращает одно из двух значений, разделенных двоеточием. Использование оператора ? может сделать более компактным текст программы. Например:
```
$text = ($a == $b) ? "А равно Б" : "А не равно Б";
echo $text;
```

Самостоятельно решите следующую задачу:
Используя оператор условия определите, является ли значение переменной суммы двух чисел четным или нет.
Сохраните результат в файле primer6.php.

Оператор выбора
Оператор выбора switch оценивает одно выражение и в зависимости от его значения выполняет один из нескольких блоков программы. Выражение в операторе switch чаще всего бывает простой переменной. Например:
```
switch ( $a )
  {
  case 1:
          echo "А равно 1";
          break;
  case 2:
          echo "А равно 2";
          break;
  case 3:
          echo "А равно 3";
          break;
  default:
          echo "А не равно ни 1, ни 2, ни 3";
  }
```
Альтернативный синтаксис оператора switch:
```
switch ( $a ):
  case 1:
          echo "А равно 1";
          break;
  case 2:
          echo "А равно 2";
          break;
  case 3:
          echo "А равно 3";
          break;
  default:
          echo "А не равно ни 1, ни 2, ни 3";
```

endswitch;

В языке PHP существует функция **isset(),** которая очень активно применяется вместе с условным оператором. Ее назначение проверять наличие переменной. Например, мы создаем страницу, которая проверяет текущую дату с днем рождения зарегистрированного пользователя (которое хранится в переменной $data_r) и в случае совпадения, поздравляет его с праздником. Но выполнять операцию сравнения можно лишь в случае, если в данную страницу передана переменная $data_r.

Циклические конструкции

Смысл циклических конструкции в языке PHP такой же, как и в других языках программирования. Синтаксис полностью идентичен языку Си.

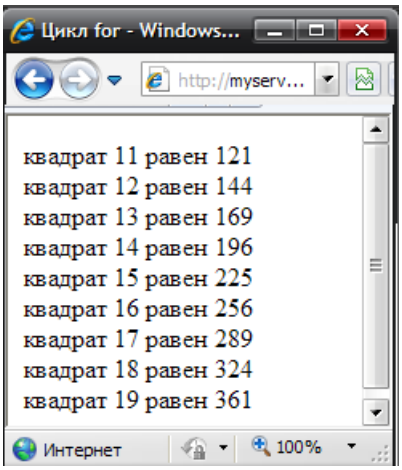
Цикл с параметром

Цикл с параметром for относится к наиболее старому и заслуженному виду цикла.

Самостоятельно создайте в папке C:\WebServers\home\myserver.ru\www новый сценарий primer7.php и выполните код, приведенный в примере 7.

Пример 7

```
<html>
<head>
 <title>Цикл for</title>
</head>
<body>
<?php
for ($a = 11; $a <= 19; $a++)
  {
   echo "квадрат $a равен ".($a*$a)."<br>";
  }
?>
</body>
</html>
```



В результате вы должны получить:

Альтернативный синтаксис оператора for:

```
for ($a = 11; $a <= 19; $a++):
   echo "квадрат $a равен ".($a*$a)."<br>";
endfor;
```

Циклы с условием

В языке PHP существует два типа цикла с условием:

- while - цикл с предусловием;
- do .. while - цикл с постусловием.

Оператор while оценивает значение условия и, если оно истинно, выполняет действия в фигурных скобках (тело цикла). Как только значение условия станет ложным, выполнение цикла прекращается.

Пример 8.

```
<html>
<head>
 <title>Цикл while</title>
</head>
<body>
<?php
$a = 11;
while ( $a <= 19 )
```

```
  {
  echo "квадрат $a равен ".($a*$a)."<br>";
  $a++;
  }
?>
</body>
</html>
```

Замечание для продвинутых: сокращенная запись не сработает как ожидается:

```
while ( $a <= 19 )
  {
  echo "квадрат $a равен ".($a*($a++))."<br>"; // так не работает!!!
  }
```

Самостоятельно создайте в папке C:\WebServers\home\myserver.ru\www новый сценарий primer8.php и выполните код, приведенный в примере 8 с альтернативным синтаксисом оператора while:

```
while ( $a <= 19 ):
  echo "квадрат $a равен ".($a*$a)."<br>";
  $a++;
endwhile;
```

Цикл do .. while отличается от цикла while лишь тем, что истинность условия проверяется не до, а после выполнения тела цикла.

```
<html>
<head>
 <title>Цикл do .. while</title>
</head>
<body>
<?php
$a = 11;
do {
  echo "квадрат $a равен ".($a*$a)."<br>";
  $a++;
  }
while ( $a <= 19 );
?>
</body>
</html>
```

В организации цикла могут участвовать еще два оператора: break (выход из цикла) и continue (переход на следующий шаг).

Практическая часть

Задание 1. Выполните примеры из теоретической части

Задание 2. Для изучения условного оператора создайте тест, состоящий из пяти вопросов (файл **test.htm**). После выбора правильных ответов, данные передаются в новый файл **analyse_test.php**, где вычисляется количество правильных ответов и выводится соответствующее сообщение. Обратите внимание, при ответе пользователь мог специально или случайно пропустить вопрос, поэтому перед проверкой каждого ответа на правильность нужно проверить, а передана ли соответствующая переменная в php-файл.

Задание 3. Создайте файл data_r.php и наберите в нем следующий программный код (листинг5):

Листинг №.5

```
<html>
<head><title>Проверка условия в языке PHP</title></head>
<body>
<H1>проверка даты рождения</H1>
<form action="data_r.php" method="get">
<P>месяц рождения: <input name="m" size="5"  type="text"></P>
<P>день рождения: <input name="d" size="5"  type="text"></P>
<P><input name="b" type="submit" value="проверить"></P>
</form>
<?
if(isset($b)){
$day = date("j");
$month = date("n");
if ($m = = $month && $d = = $day){
print("<h2>С днем рождения</h2>");
}
}
?>
</body>
</html>
```

Задание 4. Создайте PHP-страницу, в которой пользователь вводит в форму количество строк и количество столбцов. В результате по введенным значениям строится таблица. Примерный вид экрана представлен на рисунке

Содержание отчета

1. Цель
2. Синтаксис оператора ветвления и циклических конструкций на PHP
3. Программные листинги
4. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 8

Программирование массивов на PHP

Цель: отработать навыки программирования массивов на PHP.

Ход работы:

7. Изучить теоретический материал.
8. Проработать примеры.
9. Выполнить задания в соответствии с указаниями.
10. Предъявить преподавателю результаты работы.
11. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
12. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Массив - это тип данных, с данными этого типа должны быть определены операции.

В PHP существуют различные методы инициализации массивов:

1. простое присвоение значений

```
<?
   $car[] = "passenger car";
   $car[] = "land-rover";
   echo($car[1]); // выводит "land-rover"
?>
```

2. явное указание индекса массива:

```
<?
   $car[0] = "passenger car";
   $car[1] = "land-rover";
   echo($car[1]); // выводит "land-rover"
?>
```

3. использование конструкции array():

```
<?
   $car = array("passenger car", "land-rover");
   echo($car[1]); // выводит "land-rover"
?>
```

4. явное указание индексов (в этом случае применяется оператор =>)

```
<?
   $car = array("passenger car", 5 => "land-rover", "station-
wagon","victoria");
   echo($car[0]); echo("<br>"); // выводит "passenger car"
   echo($car[5]); echo("<br>"); // выводит "land-rover"
   echo($car[6]); echo("<br>"); // выводит "station-wagon"
   echo($car[7]); // выводит "victoria"
?>
```

5. индексами массива могут быть и строки:

```
<?
   $car = array("pc" => "passenger car", "lr" => "land-rover");
   echo($car["lr"]); echo("<br>"); // выводит "land-rover"
   echo($car["pc"]);  // выводит "passenger car"
?>
```

Для обработки элементов массива используют:

цикл FOREACH

```
foreach (array as [$key =>] $value)
{
   statements;
}
```

Пример:

```
<?
   $car = array("passenger car", "land-rover", "station-wagon",
"victoria");
   foreach($car as $index => $val)
   {
```

```
            echo("$index -> $val <br>");
        }
    ?>
```

Как видно из синтаксиса, переменная $key необязательна и может быть опущена:

```
    <?
        echo(
            "available cars: <br> <ul>"
        );
        $car = array("passenger car", "land-rover", "station-
wagon","victoria");
        foreach($car as $val)
        {
            echo("<li>$val</li>\n");
        }
        echo("</ul>");
    ?>
```

Складывают массивы с помощью стандартного оператора "+". Эту операцию по отношению к массивам точнее назвать объединением. Если есть два массива, $a и $b, то результатом их сложения (объединения) будет массив $c, состоящий из элементов $a, к которым справа дописаны элементы массива $b. Если складываются массивы в языке PHP, то от перемены мест слагаемых сумма меняется.

```
<?
$a = array("и"=>"Информатика", "м"=>"Математика");
$b = array("и"=>"История","м"=>"Биология", "ф"=>"Физика");
$c = $a + $b;
$d = $b +$a;
print_r($c);
/* получим: Array([и]=>Информатика [м]=>Математика [ф]=>Физика) */
print_r($d);
/* получим: Array([и]=>История [м]=>Биология [ф]=>Физика) */
?>
```

Сравнивать массивы можно, проверяя их равенство или неравенство либо эквивалентность или неэквивалентность. Равенство массивов - это когда совпадают все пары ключ/значение элементов массивов. Эквивалентность - когда кроме равенства значений и ключей элементов требуется еще, чтобы элементы в обоих массивах были записаны в одном и том же порядке. Равенство значений в PHP обозначается символом "= =", а эквивалентность - символом "= = =" [18].

Подсчет количества элементов массива. Для реализации в PHP есть специальная функция. Функция **count**:

```
<?
$del_items = array("langs" => array("10"=>"Python",
"12"=>"Lisp"),"other"=>"Информатика");
echo count($del_items) . "<br>";
// выведет 2
echo count($del_items,COUNT_RECURSIVE);
// выведет 4
?>
```

Функция **in_array** позволяет установить, содержится ли в заданном массиве искомое значение. Если третий аргумент задан как true, то в массиве нужно найти элемент, совпадающий с искомым не только по значению, но и по типу. Еще одна функция array_search используется для поиска значения в массиве. В отличие от in_array в результате работы array_search возвращает значение ключа, если элемент найден, и ложь - в противном случае.

Функция **array_keys()** выбирает все ключи массива.

Для вывода элементов массива можно использовать стандартные операторы вывода Echo и Print, но они пригодны только для поэлементного вывода, т. е. корректны будут операции:

```
Echo $ a [6]; Print $ a [11];
```

Однако вывести весь массив этими операторами нельзя.

Для вывода всего массива с указанием значений ключей используется функция:

Print _ r ($ a); // здесь $ a — это идентификатор массива.

Результат работы данной функции приведен на рис. 1.

```
Array ( [0] => 1 [1] => 4 [2] => 7 [3] => 10 [4] => 13 [5] => 16 [6] => 19 [7] => 22
[8] => 25 [9] => 28 [10] => 31 [11] => 34 [12] => 37 [13] => 40 [14] => 43 [15] =>
46 [16] => 49 [17] => 52 [18] => 55 [19] => 58 )
```

Рис. 1. Вывод массива с использованием функции Print _ r

Здесь в квадратных скобках представлен индекс каждого элемента, далее идет указатель ссылка => на значение элемента.

Можно выполнять операцию присваивания с явным указанием индекса элемента:

```
$a[20]=111;
```

Изменить значение элемента массива очень просто, для этого надо просто выполнить оператор присваивания для данного элемента. Для точного указания элемента нам требуется задать его ключ (или индекс). Например, изменить значение элемента массива $ a с индексом 20 можно следующим оператором:

```
$ a [20]=345;
```

Изменить ключ элемента массива нельзя. Можно удалить элемент массива с использованием функции *unset* () и потом заново присвоить старое значение элементу с новым ключом.

```
Unset ($ a [20]); // удаляет элемент с индексом 20.
Unset ($ a); // удаляет массив $ a целиком.
```

Если для элемента массива *ключ* не задан, то в качестве *ключа* берется максимальный числовой *ключ*, увеличенный на единицу. Если указать *ключ*, которому уже было присвоено какое-то *значение*, то оно будет перезаписано. Начиная с PHP 4.3.0, если максимальный *ключ* — отрицательное число, то следующим *ключом массива* будет ноль (0).

Заметим, что когда используются пустые *квадратные скобки*, максимальный числовой *ключ* ищется среди *ключей*, существующих в *массиве* с момента последнего *переиндексирования*. *Переиндексировать массив* можно с помощью функции *array_values* (). При использовании переиндексации происходит выстраивание числовых индексов заново по порядку, начиная с 0 (нуля).

Для обхода массива удобно использовать специальный цикл FOREACH ()

Существуют две разновидности команды FOREACH, предназначенные для разных типов массивов: линейных и ассоциативных массивов. Для линейных применим следующий синтаксис

```
foreach (массив as $элемент) { блок }
```

Для ассоциативных массивов применим иной синтаксис:

```
foreach (массив as $ключ => $элемент) { блок }
```

Например, для вывода линейного ассоциативного массива можно написать следующий фрагмент кода:

```
$s = array(array ("Попов А.А." =>4,"Архипова Т.С." => 5,"Быков П.А." => 4);
foreach ($s as $i => $mark) {
print " Студент $ i получил оценку $ mark < BR >";
}
```

Результаты работы скрипта представлены на рис. 2.

```
Студент Попов А.А. получил оценку 4
Студент Архипова Т.С. получил оценку 5
Студент Быков П.А. получил оценку 4
```

Рис. 2. Использование цикла Foreach

Примеры решения задач

Задача 1. Дан массив с элементами 'Привет, ', 'мир' и '!'. Необходимо вывести на экран фразу 'Привет, мир!'.

Решение:

```
<?php
        //Выведем фразу 'Привет, мир!':
        $arr = ['Привет, ', 'мир', '!'];
        echo $arr[0].$arr[1].$arr[2];
?>
```

Разберем это решение.

Слово 'Привет, ' хранится под номером **0**, это значит, что для доступа к нему мы должны написать **$arr[0]**.

Для доступа к слову 'мир' мы должны написать **$arr[1]**, а **$arr[2]** содержит в себе '!'. Далее с помощью оператора **'точка'** мы сложим три наши строки ('Привет, ', 'мир' и '!') в одну строку таким образом: **$arr[0].$arr[1].$arr[2]**, и выведем на экран с помощью **echo**.

Задача 2. Решим немного другую задачу: дан массив с элементами 'Привет, ', 'мир' и '!'. Необходимо записать в переменную **$text** фразу 'Привет, мир!', а затем вывести на экран содержимое этой переменной.

Решение:

```
<?php
        $arr = ['Привет, ', 'мир', '!'];
        $text = $arr[0].$arr[1].$arr[2];
        /*
                В переменной $text теперь лежит строка 'Привет, мир!'.
                Выведем ее на экран:
        */
        echo $text;
?>
```

Задача 3. Дан массив ['Привет, ', 'мир', '!']. Необходимо записать в первый элемент (то есть элемент с номером ноль) этого массива слово 'Пока, ' (то есть вместо слова **'Привет, '** будет **'Пока, '**).

Решение:

```
<?php
        $arr = ['Привет, ', 'мир', '!'];
        $arr[0] = 'Пока, '; //перезапишем первый элемент массива
        var_dump($arr); //посмотрим на массив и убедимся в том, что он изменился
?>
```

Ассоциативные массивы

Задача 4. Создайте массив заработных плат **$arr**. Выведите на экран зарплату Пети и Коли.

```
<?php
        //Этот массив дан:
```

```
        $arr = ['Коля'=>'1000$', 'Вася'=>'500$', 'Петя'=>'200$'];
?>
```

Решение: чтобы вывести зарплату Коли следует вывести значение элемента массива **с ключом 'Коля'**. Сделаем это:

```
<?php
        $arr = ['Коля'=>'1000$', 'Вася'=>'500$', 'Петя'=>'200$'];
        echo $arr['Коля']; //выведет 1000$
?>
```

Способы создания массива

Задача 5. Создайте массив **$arr** с элементами *1, 2, 3, 4, 5* двумя различными способами.

Решение:

Первый способ создать массив - объявить его через **[]**:

```
<?php
        $arr = [1, 2, 3, 4, 5];
?>
```

Второй способ создания массива - это поступить таким образом:

```
<?php
        $arr[] = 1;
        $arr[] = 2;
        $arr[] = 3;
        $arr[] = 4;
        $arr[] = 5;
?>
```

Многомерный массив

Задача 6. Дан многомерный массив **$arr**:

```
$arr = [
        'ru'=>['голубой', 'красный', 'зеленый'],
        'en'=>['blue', 'red', 'green'],
];
```

Выведите с его помощью слово **'голубой'**.

Решение: так как массив многомерный (в нашем случае двухмерный), то нам придется написать несколько квадратных скобок подряд. Поясню это по шагам. Давайте сделаем так:

```
<?php
        var_dump($arr['ru']);
?>
```

В этом случае результатом будет массив **['голубой','красный', 'зеленый']**, который является частью нашего многомерного массива. Чтобы вывести слово **'голубой'**, на необходимо **дописать** еще одну квадратную скобку с ключом, соответствующим этому элементу (у него нет явного ключа - значит его ключ **0**):

```
<?php
        echo $arr['ru'][0]; //выведет 'голубой'
?>
```

Выведем теперь слово **'красный'**:

```
<?php
        echo $arr['ru'][1]; //выведет 'красный'
?>
```

Выведем **'red'**:

```
<?php
        echo $arr['en'][1]; //выведет 'red'
?>
```

Практическая часть.

1. Выполнить прмеры, рассмотренные выше.
2. Выполнить задания, приведенные далее с решениями:
3. Выполнить задания для самостоятельной работы по вариантам.

Работа с массивами

1. Создайте массив **$arr=['a', 'b', 'c']**. Выведите значение массива на экран с помощью функции **var_dump()**. Скрыть решение.

Решение:

```
<?php
        $arr = ['a', 'b', 'c'];
        var_dump($arr)
?>
```

2. С помощью массива **$arr** из предыдущего номера выведите на экран содержимое первого, второго и третьего элементов. Скрыть решение.

Решение:

```
<?php
        $arr = ['a', 'b', 'c'];
        echo $arr[0].'<br>';
        echo $arr[1].'<br>';
        echo $arr[2].'<br>';
?>
```

3.Создайте массив **$arr=['a', 'b', 'c', 'd']** и с его помощью выведите на экран строку **'a+b, c+d'**. Скрыть решение.

Решение:

```
<?php
        $arr = ['a', 'b', 'c', 'd'];
        echo $arr[0].'+'. $arr[1].', '.$arr[2].'+'.$arr[3];
?>
```

4. Создайте массив **$arr** с элементами **2, 5, 3, 9**. Умножьте первый элемент массива на второй, а третий элемент на четвертый. Результаты сложите, присвойте переменной **$result**. Выведите на экран значение этой переменной. Скрыть решение.

Решение:

```
<?php
        $arr = [2, 5, 3, 9];
        $result = $arr[0] * $arr[1] + $arr[2] * $arr[3];
        echo $result;
?>
```

5. Заполните массив **$arr** числами от 1 до 5. Не объявляйте массив, а просто заполните его присваиванием **$arr[] = новое значение**. Скрыть решение.

Решение:

```
<?php
        $arr[] = 1;
        $arr[] = 2;
        $arr[] = 3;
        $arr[] = 4;
        $arr[] = 5;

        var_dump($arr);
?>
```

Ассоциативные массивы

6.Создайте массив **$arr**. Выведите на экран элемент с ключом **'c'**.

```
$arr = ['a'=>1, 'b'=>2, 'c'=>3];
```

Скрыть решение.

Решение:

```php
<?php
	$arr = ['a'=>1, 'b'=>2, 'c'=>3];
	echo $arr['c'];
?>
```

7.Создайте массив **$arr**. Найдите сумму элементов этого массива.

```php
$arr = ['a'=>1, 'b'=>2, 'c'=>3];
```

Скрыть решение.

Решение:

```php
<?php
	$arr = ['a'=>1, 'b'=>2, 'c'=>3];
	echo $arr['a'] + $arr['b'] + $arr['c'];
?>
```

8.Создайте массив заработных плат **$arr**. Выведите на экран зарплату Пети и Коли.

```php
$arr = ['Коля'=>'1000$', 'Вася'=>'500$', 'Петя'=>'200$'];
```

Скрыть решение.

Решение:

```php
<?php
	$arr = ['Коля'=>'1000$', 'Вася'=>'500$', 'Петя'=>'200$'];
	echo $arr['Петя'].' '.$arr['Коля'];
?>
```

9.Создайте ассоциативный массив дней недели. Ключами в нем должны служить номера дней от начала недели (понедельник - должен иметь ключ 1, вторник - 2 и т.д.). Выведите на экран **текущий** день недели. Скрыть решение.

Решение: пусть текущий день - четверг, тогда:

```php
<?php
	$arr = [1=>'пн', 2=>'вт', 3=>'ср', 4=>'чт', 5=>'пт', 6=>'сб', 7=>'вс'];
	echo $arr[4];
?>
```

10.Пусть теперь номер дня недели хранится в переменной **$day**, например там лежит число 3. Выведите день недели, соответствующий значению переменной **$day**. Скрыть решение.

Решение: пусть текущий день - четверг, тогда:

```php
<?php
	$arr = [1=>'пн', 2=>'вт', 3=>'ср', 4=>'чт', 5=>'пт', 6=>'сб', 7=>'вс'];
	$day = 3;
	echo $arr[$day];
?>
```

Многомерные массивы

11.Создайте многомерный массив **$arr**. С его помощью выведите на экран слова *'joomla', 'drupal', 'зеленый', 'красный'*.

```php
$arr = [
	'cms'=>['joomla', 'wordpress', 'drupal'],
	'colors'=>['blue'=>'голубой', 'red'=>'красный', 'green'=>'зеленый']
];
```

Скрыть решение.

Решение:

```php
<?php
	$arr = [
		'cms'=>['joomla', 'wordpress', 'drupal'],
		'colors'=>['blue'=>'голубой', 'red'=>'красный', 'green'=>'зеленый']
	];
```

```
        echo $arr['cms'][0]; //joomla
        echo $arr['cms'][2] //drupal;
        echo $arr['colors']['green']; //зеленый
        echo $arr['colors']['red']; //красный
?>
```

12. Создайте двухмерный массив. Первые два ключа - это **'ru'** и **'en'**. Пусть первый ключ содержит элемент, являющийся массивом названий дней недели *по-русски*, а второй - *по-английски*. Выведите с помощью этого массива понедельник по-русски и среду по английски (пусть понедельник - это первый день). Скрыть решение.

Решение:

```
<?php
        $arr = [
                'ru'=>[1=>'пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'],
                'en'=>[1=>'mn', 'ts', 'wd', 'th', 'fr', 'st', 'sn'],
        ];
        echo $arr['ru'][1]; //пн
        echo $arr['en'][2]; //wd
?>
```

13. Пусть теперь в переменной **$lang** хранится язык (она принимает одно из значений или 'ru', или 'en' - либо то, либо то), а в переменной **$day** - номер дня. Выведите словом день недели, соответствующий переменным **$lang** и **$day**. То есть: если, к примеру, $lang = 'ru' и $day = 3 - то выведем 'среда'

Решение:

```
<?php
        $arr = [
                'ru'=>['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'],
                'en'=>['mn', 'ts', 'wd', 'th', 'fr', 'st', 'sn'],
        ];
        $lang = 'ru';
        $day = 3;

        echo $arr[$lang][$day]; //ср
?>
```

Задания для самостоятельной работы

Задание 1

Создать линейный массив из 20 целых чисел, значения которых вычисляются по формуле $ a []=$ i *3+ N, здесь N — это номер студента по журналу; $ i — индекс элемента массива. Вывести полученный массив на печать с использованием функции print _ r;

Задание 2

Создать линейный массив из 30 целых чисел с индексами, соответствующими четным числам, начиная с 2. Значения элементов массива вычисляются по формуле $ a [$ i]=$ i /2 + N; здесь N — это номер студента по журналу; $ i — индекс элемента массива. Вывести массив в виде таблицы. При выводе в ячейки таблицы используйте выравнивание содержимого по центру ячейки. Таблицу сделать с границей. Вывод проводить в цикле поэлементно. Помнить, что в операторах вывода Echo и Print нельзя использовать внутри строки вывода арифметические операции. Все вычисления и присваивания надо сделать до начала цикла.

Тег создания таблицы может быть вынесен за текст PHP-скрипта, как и тег завершения, потому что это чисто HTML-операторы.

Обход элементов массива можно делать с использованием специальных функций перемещения по указателю (курсору). Это следующие функции (см. табл. 1):

Таблица 1. Функции по работе с массивами с использованием курсора

Функция	**Назначение**
count ($ arr 1)	Число элементов на внешнем уровне массива
Next ($ arr 1)	Установка указателя на следующий элемент массива
prev($arr1)	Установка указателя на предыдущий элемент массива
current($arr1)	Установка указателя на текущий элемент массива
reset ($ arr)	Сброс указателя массива в исходное состояние
End ($ arr)	Установка указателя на последний элемент массива
Key ()	Получение значение ключа текущего элемента массива

Задание 3

Создать с использованием оператора Array массив, соответствующий расшифровке цветов радуги с помощью ключевой фразы «Каждый охотник желает знать, где сидит фазан». Значениями должны стать слова фразы, ключами должны стать соответствующие цвета.

Обойти массив с использованием функций управления курсором сначала в одну сторону, а потом наоборот, распечатать массив при обходе в виде таблицы, аналогично заданию 2. Вывести значение и ключ текущего элемента массива сразу после создания массива.

Использовать цикл While. При перемещении по массиву функция next () будет отличаться от нуля (будет истинна), пока вы находитесь внутри массива. При перемещении в обратную сторону функция prev () также будет отличной от нуля, пока вы не выходите за границы массива.

Распечатать ключевую фразу в одну строку.

При работе с массивами часто возникает необходимость добавить или удалить элементы массива либо с конца, либо сначала. Конечно, добавить элемент массива в конец можно, пройдя весь его целиком, но для подобных действий в PHP существуют дополнительные функции, которые приведены в табл. 2.

Задание 4

Создайте ассоциативный массив из 5 элементов, ключами в массиве являются фамилии людей, значениями — номера их телефонов. Напишите скрипт, который по фамилии, переданной в качестве параметра, выводит на экран номер телефона данного человека.

При передаче мы пользуемся методом GET, т. е. передаем параметры через строку вывода, отделяя их знаком вопроса ? от имени скрипта.

Задание 5

Создать массив $ sessia, заполнив его данными для 5 студентов. Количество сданных экзаменов должно колебаться от 1 до 5. Напишите скрипт, который позволяет получить по фамилии студента количество сданных им экзаменов и их список с оценками. Фамилию студента передавать в скрипт как параметр через протокол GET.

Задание 6

Расширить обработку в задании 6. В качестве параметров у вас могут передаваться либо фамилия студента, либо название дисциплины. Для первого случая обработка остается такая же, как в задании 6. При передаче в качестве параметра названия дисциплины требуется вывести список всех студентов, которые сдали данную дисциплину с указанием оценки, которую студент получил на экзамене.

Задание 7

Для каждой группы задан перечень экзаменов, вынесенных на сессию. Список задан в виде массива, ключом которого является номер группы, а значением — массив предметов, вынесенных для данной группы на сессию.

Получить список предметов, вынесенных на сессию для всех групп без повторений. В исходных данных задать не менее 3 групп и не менее 3 экзаменов для каждой группы.

Еще одна важная функция — это функция поиска в массиве array _ search ($ b,$ s), здесь $ b — искомый элемент, $ s — массив, в котором проводится поиск. При этом результатом является истина, если найден искомый фрагмент среди значений исходного массива, и ложь, в противном случае. Рассмотрим пример с линейным массивом:

```
<?
$a=array('php','c#','sql','.net');
$b='sql';
If (array_search ($b,$a)) echo " Есть $b";
else echo "Нет $b ";
?>
```

Однако функцию array _ search ($ b,$ a) можно применять и в том случае, если искомым элементом является также массив. Например, допустим, что у нас есть массив, в котором находятся результаты сессии. Ключами являются фамилии студентов, значениями являются массивы из пар предмет-оценка. Зададимся вопросом, сдал ли кто-либо заданные предметы на заданные оценки. Далее приведен скрипт, который решает подобную задачу:

```
<?
$s= array('Иванов'=> array (‘Математика'=>5,'Информатика'=>4),
'Петров'=>array('Информатика'=>4,'Экономика'=>4));
$b=array('Математика'=>5,'Информатика'=>4);
$c=array_search ($b,$s);
If (array_search ($b,$s)) echo " Найдено $c ";
else echo " Не найдено ";
?>
```

Задание 8

Задан массив «Сессия» для 4 студентов, каждый из которых сдавал не менее 3 экзаменов. Написать скрипт, который позволяет выполнить следующие действия:

- по заданной фамилии получить перечень дисциплин с указанием оценки, полученной на сессии;
- по заданному названию дисциплины получить список всех студентов, которые сдавали данный предмет.

Параметры передавать в скрипт по протоколу GET, используя адресную строку. Для различения ситуаций использовать функцию isset ($ f), которая позволяет определить, была ли задана переменная $ f.

Задание 9

Задано расписание занятий на текущую неделю для конкретной аудитории. Считаем, что в качестве аудитории выбрана аудитория на одну группу, поэтому на каждой паре в ней могут проходить занятия только для конкретной группы. Допустим, это расписание может быть представлено в виде сложного массива, например:

```
$r=array('понедельник'=>array(
1=>array('Группа'=>145,'Предмет'=>'Программирование','Преподаватель'=>
'Богословская Н.В.'),
2=>array('Группа'=>145, 'Предмет'=>'ИС', 'Преподаватель'=>'Соколов Н.Е.'),
3=>array('Группа'=>141, 'Предмет'=>'Банковское дело', 'Преподаватель'=>
'Белоглазова Г.Н.')),
'вторник'=>array(
1=>array('Группа'=>145, 'Предмет'=>'ВС', 'Преподаватель'=>'Гришин П.В.'),
2=>array('Группа'=>141, 'Предмет'=>'Бухучет', 'Преподаватель'=>'Бусуек Н.А.')
));
```

Здесь задано расписание на 2 дня: понедельник и вторник, причем в понедельник заняты 3 пары в данной аудитории, а во вторник только 2.

Для того чтобы получить ответ, кто ведет занятия во вторник на второй паре, необходимо обратиться к элементу массива следующим образом:

```
$r[‘ вторник '][2][‘ преподаватель ']
```

Расширить расписание, заполнив его на 3 пары среды.

Написать скрипт, который позволяет ответить на следующие вопросы:

- вывести расписание для конкретной группы (на все дни недели с указанием предмета и преподавателя);
- вывести расписание для конкретного преподавателя, так же на все дни недели с указанием группы и предмета.

Группу или фамилию преподавателя задавать как параметры при вызове скрипта.

Содержание отчета

1. Цель
2. Синтакис задания массивов на PHP
3. Функции для работы с массивами на PHP
4. Выводы

Контрольные вопросы

6. В чем отличие php-страницы от html-страницы?
7. Какие типы переменных поддерживает язык PHP?
8. Как передать переменную в php-страницу?
9. Какие параметры существуют у функции date()?
10. Для чего используется функция isset()?

Лабораторная работа № 9
Работа с датой и временем

Цель: отработать навыки обработки данных с помощью методов объекта даты на PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

date(); - функция возвращающая текущую дату в виде строки. Функция имеет большое количество параметров.

Параметры функции date():

G - час, 24-часовой формат без ведущих нулей; т.е. от «0" до «23"
i - минуты; т.е. от «00" до «59"
j - день (число) месяца без ведущих нулей; т.е. от «1" до «31"
m - месяц; т.е. от «01" до «12"
H - час, 24-часовой формат; т.е. от «00" до «23"
n - месяц без ведущих нулей; т.е. от «1" до «12"
s - секунды; т.е. от «00" до «59"
Y - год, 4 цифры; например, «1999"
y - год, 2 цифры; например, «99"
z - день года; т.е. от «0" до «365"
пример использования функции: date()
$today = date("j, n, Y"); переменная $today примет значение: 10, 3, 2001 (число, месяц, год)
$today = date("H:i:s"); переменная $today примет значение: 17:16:54 (часы, минуты, секунды)

Примеры решения задач

Задача 1. Выведите **23 сентября 2031 года, 12:58:59** в формате **timestamp**.
Решение: воспользуемся функцией mktime (сентябрь - 9-тый месяц):

```
<?php
        echo mktime(12, 58, 59, 9, 23, 2031);
        /*
                Напоминаю, что месяц и день идут в неправильном порядке:
                поэтому '9, 23,' а не '23, 9'.
        */
?>
```

Можно также воспользоваться функцией strtotime, если представить нужную дату в формате **2031-09-23 12:58:59**:

```
<?php
        echo strtotime('2031-09-23 12:58:59');
?>
```

Задача 2. Найдите разницу между **1 сентября 2010 года, 7:25:59** и текущим моментом времени в секундах.
Решение: текущий момент времени в формате **timestamp** получим с помощью функции time, а **timestamp** для заданной даты - с помощью функции mktime. Отнимем одно число от второго и получим искомую разницу:

```
<?php
```

```
        echo time() - mktime(7, 25, 59, 9, 1, 2010);
?>
```

Функция date

Задача 3. Выведите текущую дату-время в формате **'2025.12.31 12:59:59'**.

Решение: воспользуемся функцией date, передав ей управляющие команды в таком порядке: год (команда **Y**), потом точку как символ, потом месяц (команда **m**), опять точку, день (команда **d**), час (команда **H**), двоеточие, минуту (команда **i**), секунду (команда **s**). Получится такая строка: **'Y.m.d H:i:s'**. Подставим ее в функцию **date**:

```
<?php
        echo date('Y.m.d H:i:s');
?>
```

Задача 4. Выведите 1-го сентября текущего года в формате **'2018.09.01'**.

Решение: для начала с помощью функции mktime преобразуем 1-го сентября текущего года в формат **timestamp**. Мы это делаем для того, чтобы подставить найденное число вторым параметром в функцию date (а первым параметром для **date** мы укажем формат вывода).

Так как требуется текущий год, то последний параметр (год) для **mktime** мы не указываем - тогда возьмется текущий год:

```
<?php
        //Выведем timestamp 1-го сентября текущего года:
        echo mktime(0, 0, 0, 9, 1);
?>
```

Ну, а теперь подставим найденный **timestamp** в функцию **date**:

```
<?php
        echo date('Y.m.d', mktime(0, 0, 0, 9, 1));
?>
```

Функция date. Вывод дня недели словом

Задача 5. Узнайте, какой день недели (словом) был 1 сентября 2010 года.

Решение: решение аналогично предыдущей задаче, только формат вывода для функции **date** мы сделаем в таком виде: **'w'**. В этом случае **date** вернет нам число, соответствующее дню недели за заданную дату (0 - воскресенье, 1 - понедельник и так далее):

```
<?php
        //День недели числом за нужную дату:
        echo date('w', mktime(0, 0, 0, 9, 1, 2010));
?>
```

Кстати, если бы мы хотели узнать, какой день недели сегодня - мы бы просто не передавали второй параметр функции **date** (тогда бы взялся текущий момент времени и, соответственно, вывелась бы **'w'** за текущий день).

Продолжим решать нашу задачу: мы вывели *номер* дня недели, а по задаче его следует вывести *словом*. Для этого составим массив дней недели **$week** и с его помощью выведем то, что нам нужно. Вот этот массив:

```
<?php
        //Массив дней недели:
        $week = ['вс', 'пн', 'вт', 'ср', 'чт', 'пт', 'сб'];
        //Выведем с его помощью, к примеру, понедельник:
        echo $week[1];
        //А теперь вторник:
        echo $week[2];
?>
```

Совместим теперь то, что вернет нам **date**, с нашим массивом **$week**:

```
<?php
```

```
//День недели цифрой за нужную дату:
$day = date('w', mktime(0, 0, 0, 9, 1, 2010));
//Массив дней недели:
$week = ['вс', 'пн', 'вт', 'ср', 'чт', 'пт', 'сб'];
//День недели словом:
echo $week[$day];
?>
```

Задача 6. Дана дата в формате **'31-12-2025'**. С помощью функций mktime и explodeпереведите эту дату в формат **timestamp**.

Решение: разобьем строку '31-12-2025' функцией **explode** в массив **$arr**:

```
<?php
	$arr = explode('31-12-2025'); //получим ['31', '12', '2025']
?>
```

В элементе **$arr[0]** будет лежать день, в элементе **$arr[1]** - месяц, в элементе **$arr[2]**- год. Подставим эти данные в функцию **mktime** (напоминаю, что она принимает данные в формате '...месяц-день-год', не '...день-месяц-год')

```
<?php
	$arr = explode('31-12-2025');
	echo mktime(0, 0, 0, $arr[1], $arr[0], $arr[2]);
?>
```

Практическая часть

Задание 1. **Передача переменных**

1. Создайте новый html-файл (forma.html), содержащий следующий код (листинг 12):

Листинг №3.

```
<html>
<head>
<title>ввод значений в форму</title>
</head>
<body>
<form action="age.php" method="get">
<p>ваше имя: <input name="user_name" size="20" type="text"></p>
<p>год рождения: <input name="user_yare" size="20" type="text"></p>
<input name="b1" type="submit" value="отправить"><input name="b2" type="reset"
value="очистить">
</form>
</body>
</html>
```

2. Создайте файл **age.php** и наберите в нем код, представленный в листинге №4.

Листинг №4.

```
<html>
<head>
<title>вычисление возраста</title>
</head>
<body>
<p>Добро пожаловать <? echo ($user_name) ?></p>
<?
$yare = date("Y");
$user_age = $yare - $user_yare;
print ("<p>вам  $user_age лет</p>");
?>
</body>
</html>
```

Введем имя пользователя, например, **Dik**, а год его рождения **1973,** нажав кнопку «отправить», данные будут переданы файлу **age.php**, и строка адреса примет вид:
http://localhost/xp/age.php?user_name=Dik&user_yare=1973&b1=%EE%F2%EF%F0%E0%E2%E8%F2%FC
- все передаваемые данные располагаются за символом **“?”**
- все данные собраны в виде: имя переменной = значение переменной;
- переменная b1 имеет значение «отправить», текст, содержащий кириллицу. Для таких переменных браузер автоматически выполняет URL-кодирование.

Задание 2. Самостоятельно создайте форму, в которой вводится имя студента, и его год его рождения. Данные из формы должны передаваться в php-файл, который определяет, на каком курсе учится человек (предполагая, что студент поступил в колледж в 16 лет).

Timestamp: time и mktime

Задание 3. Выведите текущее время в формате **timestamp**. Показать решение.
Задание 4. Выведите 1 марта 2025 года в формате **timestamp**. Показать решение.
Задание 5. Выведите 31 декабря текущего года в формате **timestamp**. Скрипт должен работать независимо от года, в котором он запущен. Показать решение.
Задание 6. Найдите количество секунд, прошедших с **13:12:59 15-го марта 2000 года** до настоящего момента времени. Показать решение.
Задание 7. Найдите количество **целых часов**, прошедших с **7:23:48 текущего дня** до настоящего момента времени. Показать решение.

Функция date

Задание 8. Выведите на экран текущий год, месяц, день, час, минуту, секунду. Показать решение.
Задание9. Выведите текущую дату-время в форматах **'2025-12-31'**, **'31.12.2025'**, **'31.12.13'**, **'12:59:59'**. Показать решение.
Задание 10. С помощью функций mktime и date выведите 12 февраля 2025 года в формате **'12.02.2025'**. Показать решение.
Задание 11. Создайте массив дней недели **$week**. Выведите на экран название текущего дня недели с помощью массива **$week** и функции date. Узнайте какой день недели был **06.06.2006**, в ваш день рождения. Показать решение.
Задание 12. Создайте массив месяцев **$month**. Выведите на экран название текущего месяца с помощью массива **$month** и функции date. Показать решение.
Задание 13. Найдите количество дней в текущем месяце. Скрипт должен работать независимо от месяца, в котором он запущен. Показать решение.
Задание 14. Сделайте поле ввода, в которое пользователь вводит год (4 цифры), а скрипт определяет **високосный** ли год. Показать решение.
Задание 15. Сделайте форму, которая спрашивает дату в формате **'31.12.2025'**. С помощью функций mktime и explode переведите эту дату в формат **timestamp**. Узнайте день недели (словом) за введенную дату. Показать решение.
Задание 16. Сделайте форму, которая спрашивает дату в формате **'2025-12-31'**. С помощью функций mktime и explode переведите эту дату в формат **timestamp**. Узнайте месяц (словом) за введенную дату.

Сравнение дат

Задание 17. Сделайте форму, которая спрашивает две даты в формате **'2025-12-31'**. Первую дату запишите в переменную **$date1**, а вторую в **$date2**. Сравните, какая из введенных дат больше. Выведите ее на экран. Показать решение.

Прибавление и отнимание дат

Задание 18. В переменной **$date** лежит дата в формате **'2025-12-31'**. Прибавьте к этой дате 2 дня, 1 месяц и 3 дня, 1 год. Отнимите от этой даты 3 дня.

Задание 19. Узнайте сколько дней осталось до Нового Года. Скрипт должен работать в любом году.

Задание 20. Сделайте форму с одним полем ввода, в которое пользователь вводит год. Найдите все пятницы 13-е в этом году. Результат выведите в виде массива дат.
Задание 21. Узнайте какой день недели был 100 дней назад.

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 10

Разработка серверных сценариев для работы с функциями

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Создание любого серьезного динамического сайта всегда предполагает разбиение его функциональности на части, которые могут быть запрограммированы и отлажены отдельно. Реализация подобного подхода может быть выполнена с использованием двух технологий: процедурной и объектно-ориентированной.

При процедурном подходе выделяются некоторые однотипные последовательности действий, которые оформляются как отдельные функции или процедуры и могут быть многократно использованы в разных местах работы.

При объектно-ориентированном подходе выделяются некоторые объекты, поведение которых определяется их методами. Применение каждого метода и вызывает изменение поведения объекта.

Во многом эти технологии пересекаются, процедурно-модульный подход был исторически реализован раньше, с появлением языков программирования, которые поддерживают объектно-ориентированный подход, появилась и стала развиваться соответствующая технология. До версии PHP5 язык слабо поддерживал объектную технологию, но, начиная с последней версии, разработчик может использовать и ту и другую технологию разработки программного проекта. Однако и при применении объектно-ориентированного подхода требуется тоже разрабатывать методы для информационных объектов, которые оформляются практически так же, как функции.

Определение функции обычно состоит из трех частей:

имени функции;

круглых скобок, в которых перечисляются необязательные входные параметры, разделенные запятыми;

тела функции, заключенного в фигурные скобки.

Обобщенный синтаксис функций PHP выглядит так:

```
function имя_функции ([$параметр1. $параметр2,.... $параметр_ n ]) {
тело функции }
```

В отличие от переменных имена функций, как системных, так и пользовательских, являются регистро-независимыми.

Для того чтобы функция возвращала значение, в ней должен присутствовать один или несколько операторов RETURN, после которых пишется имя переменной или массива, который возвращается.

Рассмотрим простейший пример функции, которая вычисляет сумму 3 входных параметров.

```
Function Sum($v1,$v2,$v3)
{
$s=$v1+$v2+$v3;
Return $s;
}
```

Вызвать функцию можно из любого места текста скрипта после ее определения. Для этого надо задать конкретные значения аргументов функции. Например, мы можем вызвать ранее описанную функцию с использованием следующего фрагмента кода:

```
$v1=188;
$v2=-56;
$v3=432k;
Echo Sum($v1,$v2,$v3);
```

Обращение к функции можно применять прямо внутри оператора вывода, что несомненно сделает более читаемым весь код.

Практическая часть

Задание 1. Напишите функцию, которая возвращает признак «високосный» или «невисокосный» для года, который является параметром функции. Напоминаю, что если год

- делится без остатка на 400 или
- делится без остатка на 100 или
- делится без остатка на 4,

то он является високосным, в остальных случаях — нет.

Остаток от деления определяется оператором %.

Проверить работу функции на входных значениях: 2001,2005,1760,2007,2008.

Задание 2. Написать функцию вычисления НОД — наибольшего общего делителя двух чисел.

Применить алгоритм Эвклида, здесь a и b исходные числа:

1. Если a > b, то переходим к пункту 2.
2. Вычислим r — остаток от деления числа a на b, a = bq+r, 0 <= r < b.
3. Если r = 0, то b есть искомое число.
4. Если r<>0 (остаток не ноль), то заменим пару чисел (a,b) парой (b,r) и перейдем к шагу 2.

Применить функцию вычисления НОД для следующих пар чисел:

1. 188, 12
2. 253, 11
3. 866, 244
4. 333, 99

В общем случае функция может возвращать не только простые элементарные данные, но и массивы. Рассмотрим функцию, которая вычисляет для заданного числа все его делители и возвращает этот список делителей в виде целочисленного массива.

Прежде всего отметим, что тривиальные делители, т. е. 1. и само число мы исключим из списка делителей. Далее для формирования массива целых чисел, которые могут быть претендентами на роль делителя, применим функцию формирования массива целых чисел range ($ n 1,$ n 2). Эта функция позволяет сформировать массив целых чисел, начиная с числа $ n 1 и заканчивая числом $ n 2. Мы формируем массив претендентов, начиная с числа 2 и заканчивая самим числом. Почему мы не хотим сразу исключить и само число из списка претендентов? В этом случае нам будет сложнее выявлять ситуацию отсутствия делителей. Если же мы оставляем это число, то в любом случае функция формирования массива делителей нам вернет массив, в котором по крайней мере всегда будет присутствовать один элемент — само число. И далее мы можем эффективно использовать этот момент. Мы проверяем, если элемент возвращаемого массива с нулевым индексом равен самому числу, значит, у этого числа нет делителей.

Скрипт с описанием самой функции и ее вызовом приведен ниже. Исходное число мы передаем в адресной строке, поэтому при тестировании просто меняем параметр вызова и получаем соответствующие результаты. Для того чтобы не включать

само число в список его делителей, мы применили уже известную нам функцию array _ pop (), которая удаляет последний элемент массива.

```
<?
// Описание функции
function dels ($ n)
{
// создаем массив целых чисел от 2 до самого числа
$a= rang(2,$n);
foreach ($a as $d)
{ if ($n%$d==0)$ del []=$d;
}
return $de;
}
// Применение функции.
// Число $ n передаем в адресной строке как параметр
$ dd = dels ($ n);
if ($ dd [0]==$ n) echo "у числа $ n нет делителей";
else
{// удаляем последний элемент, который равен самому числу
array _ pop ($ dd);
echo "Делители числа $ n < br >";
foreach ($dd as $v) echo $v."<br>";
}
?>
```

Результат работы функции приведен на рис. 1.

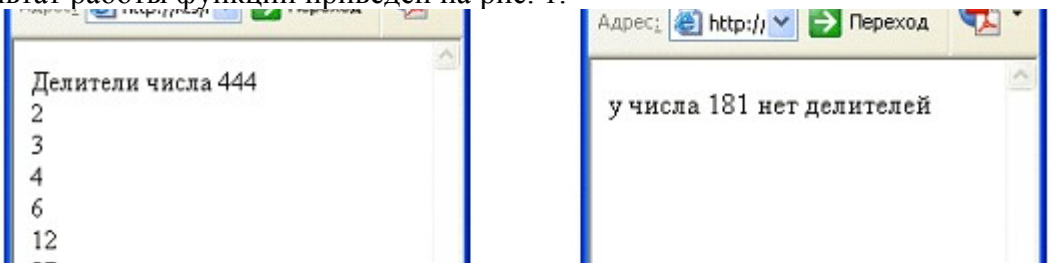


Рис. 1. Работа функции вычисления делителей числа

Задание 3. Написать функцию, аргументами которой являются 2 целых числа, а результатом работы является массив, содержащий набор общих делителей для исходных чисел, или сообщение о том, что таких общих делителей нет, если пересечение массивов делителей пусто.

При написании функции можно использовать функцию dels (), приведенную выше, или не использовать ее. Если вы будете использовать функцию dels (), то необходимо будет включить ее текст в ваш скрипт, при этом описание функции должно предшествовать ее использованию.

Исходные числа передавать как параметры в адресной строке. Проверить работу функции на следующих наборах чисел:

1. 1768,157
2. 337,1981
3. 888,444
4. 168,222

До сих пор мы рассматривали в качестве аргументов функции только простые числа. В общем случае аргументом может быть и строковая переменная, и массив. Рассмотрим пример. Дан массив, который соответствует недельному расписанию занятий. Требуется написать функцию, которая считает количество занятий у конкретного преподавателя в расписании.

```
<?
Function col_z($r,$prep)
{ $n=0;
foreach ($r as $day)
{
foreach ($day as $p)
if ($p[‘ Преподаватель ']==$prep) $n++;
}
Return $n;
}

$r=array(' понедельник '=>array(
1=>array(' Группа '=>145,' Предмет '=>' Программирование ',' Преподаватель '=>' Богословская Н. В.'),
2=>array('Группа'=>145, 'Предмет'=>'ИС', 'Преподаватель'=>'Соколов Н.Е.'),
3=>array('Группа'=>141, 'Предмет'=>'Банковское дело', 'Преподаватель'=>'Белоглазова Г.Н.')),
'вторник'=>array(
1=>array('Группа'=>145, 'Предмет'=>'ВС', 'Преподаватель'=>'Гришин П.В.'),
2=>array('Группа'=>141, 'Предмет'=>'Бухучет', 'Преподаватель'=>'Бусуек Н.А.')
));
$ prep ='Гришин П.В.';
Echo “У преподавателя $ prep ”. col _ z ($ r,$ prep). “ занятий на данной неделе”;
?>
```

Часто как сами функции, так и некоторые фрагменты текста удобно пригружать к тексту основного скрипта с использованием оператора INCLUDE (имя файла), в котором задается имя файла, содержащего загружаемый текст. Следует заметить, что текст функции и массив с расписанием необходимо при хранении в отдельных файлах заключить в стандартные теги зрз <?...?>.

Так мы можем из приведенного фрагмента текста убрать описание массива с расписанием занятий и сохранить его в отдельном файле, например с именем ras. inc, а описание функции так же сохранить в файле с именем lib. inc. Теперь текст нашего скрипта изменится существенно и будет выглядеть следующим образом:

```
<?
Include (‘lib.inc');
Include (‘ras.inc');
$ prep ='Гришин П.В.';
Echo “У преподавателя $ prep ”. col _ z ($ r,$ prep). “ занятий на данной неделе”;
?>
```

Задание 4. Разработать функцию, которая считает количество занятий на неделе для конкретной группы.

Объединить данную функцию и функцию для подсчета занятий у преподавателя в одном библиотечном файле с именем lib. inc. Подготовить 3 файла с разными расписаниями, содержащими расписания на всю неделю (5 дней), задать им имена ras 1. inc, ras 2. inc, ras 3. inc.

Разработать скрипт-обработчик, который в соответствии с передаваемыми параметрами подключает разные расписания и вычисляет количество занятий у преподавателя либо количество занятий у группы. Номер группы так же передавать как параметр в скрипт, а фамилию и инициалы преподавателя задавать в виде константы прямо в обработчике. Последнее связано с тем, что при передаче последовательности символов кириллицы при наличии пробела он заменяется в строке вызова

последовательностью кодов 20 % и это не позволяет введенную символьную последовательность сопоставить со значением, заданным в массиве.

Аргументы функции

У каждой функции может быть, как мы уже говорили, список аргументов. С помощью этих аргументов в функцию передается различная информация (например, значение числа, делители которого надо найти). Все аргументы функции играют роль входных параметров. Возвращаемое значение передается через оператор Return. Каждый аргумент представляет собой переменную, константу или идентификатор массива.

С помощью аргументов данные в функцию можно передавать тремя различными способами. Это передача аргументов по значению (используется по умолчанию), по ссылке и *задание значения аргументов по умолчанию*. Рассмотрим эти способы подробнее.

Когда аргумент передается в функцию по значению, изменение значения аргумента внутри функции не влияет на его значение вне функции. Чтобы позволить функции изменять ее аргументы, их нужно передавать по ссылке. Для этого в определении функции перед именем аргумента следует написать знак амперсанд «&». При этом знак косвенной адресации — амперсанд мы можем ставить не обязательно при описании функции. Мы можем его использовать при обращении к функции. И в этом случае все изменения начинают происходить в том объекте, который был параметром при вызове функции. Рассмотрим пример: создадим функцию, которая увеличивает каждый элемент числового массива на 1. Текст скрипта, соответствующий данной функции, приведен ниже:

```
<?
Function ar1($a){
$cc=count($a);
for ($i=0;$i < $cc; $i++)
$a[$i]=$a[$i]+1;
}
$a=array(1,2,3,4,5);
$b=array(3,4,5,6,7,8);
print " Исходные <br>";
print_r($a);
echo "<br>__________b___ <br>";
print_r($b);
ar1(&$a);
ar1($b);
print "<br> Конечные <br>";
print_r($a);
echo "<br>__________b___ <br>";
print_r($b);
?>
```

Мы применили разработанную функцию к массиву $a, задав косвенную адресацию и к массиву $b, применив обычную адресацию.

Результаты работы данного скрипта приведены на рис. 2.

Переменные в функциях имеют локальную область видимости. Это означает, что если даже локальная и внешняя переменные имеют одинаковые имена, то изменение локальной переменной никак не повлияет на внешнюю переменную.

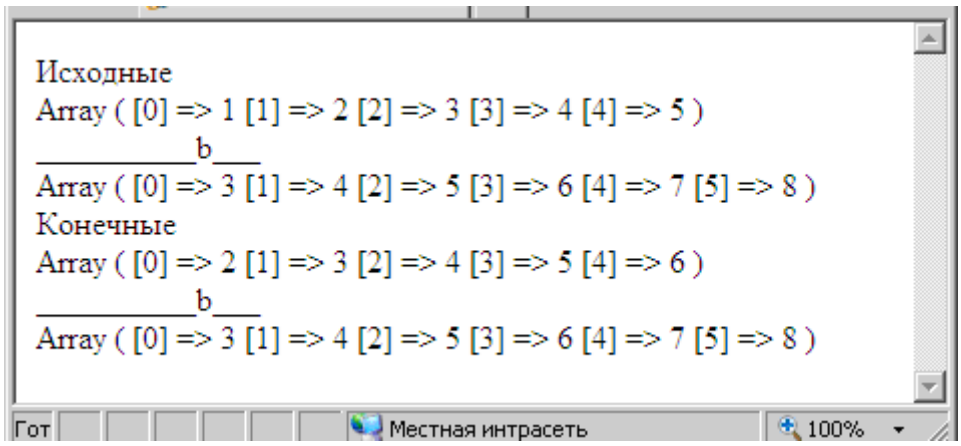


Рис. 2. Пример передачи параметров по ссылке и по значению

```
<?
function get_sum()
{
$var = 5; // локальная переменная
echo $var;
}
$var = 10; // глобальная переменная
get_sum(); // выводит 5 (локальная переменная)
echo("<br>$var"); // выводит 10 (глобальная переменная)
?>
```

Локальную переменную можно сделать глобальной, если перед ее именем указать ключевое слово global. Если внешняя переменная объявлена как global, то к ней возможен доступ из любой функции:

```
<?
function get_sum()
{
global $var;
$var = 5; // изменяем глобальную переменную
echo $var;
}
$var = 10;
echo("$var<br>"); // выводит 10
get_sum(); // выводит 5 (глобальная переменная изменена)
?>
```

Доступ к глобальным переменным можно получить также через ассоциативный массив $GLOBALS:

```
<?
function get_sum()
{
$GLOBALS["var"] = 20; // изменяем глобальную переменную $var
echo($GLOBALS["var"]);
}
$var = 10;
echo("$var<br>"); // выводит 10
get_sum(); // выводит 20 (глобальная переменная изменена)
?>
```

Массив $GLOBALS доступен в области видимости любой функции и содержит все глобальные переменные, которые используются в программе.

Временем жизни переменной называется интервал выполнения программы, в течение которого она существует. Поскольку локальные переменные имеют своей областью видимости функцию, то время жизни локальной переменной определяется временем выполнения функции, в которой она объявлена. Это означает, что в разных функциях совершенно независимо друг от друга могут использоваться переменные с одинаковыми именами. Локальная переменная при каждом вызове функции инициализируется заново, поэтому функция-счетчик в приведенном ниже примере всегда будет возвращать значение 1:

```
function counter () { $counter = 0 ; return ++ $counter; }
```

Для того чтобы локальная переменная сохраняла свое предыдущее значение при новых вызовах функции, ее можно объявить статической при помощи ключевого слова **static** :

```
function counter () { static $counter = 0; return ++ $counter; }
```

Временем жизни статических переменных является время выполнения сценария, т. е., если пользователь перезагружает страницу, что приводит к новому выполнению сценария, переменная **$counter** в этом случае инициализируется заново.

Задание 5. Считаем, что у нас есть список посетителей нашего сайта, каждый из посетителей имеет ФИО и некоторый условный nik, который представлен последовательностью латинских букв не более 6 без пробелов и других специальных знаков.

Разработать функцию, которая бы считала для каждого из наших посетителей количество его посещений нашего сайта в течение текущего сеанса и выдавала бы соответствующее приветствие, например:

Уважаемый Николай Константинович, Вы сегодня посетили наш сайт уже 5 раз. Мы рады Вас приветствовать.

Процедуру подсчета для каждого посетителя оформить в виде функции.

Список наших посетителей, с указанием их ФИО, nik и пола (для формирования грамотного обращения) сохранять в виде массива.

Подумать, где можно хранить количество посещений для каждого посетителя.

Задание 6. Разработать функцию формирования пригласительного письма на конференцию. В пригласительном письме учесть, является ли адресат гостем или участником конференции. Если адресат является участником, то он может быть докладчиком либо на пленарном заседании, либо на некоторой секции, номер которой указывается в качестве параметра вызова процедуры.

Список секций хранить в виде массива, ключами в массиве должны быть номера секций, а значениями — полные названия секций и аудитории, в которых они будут проводиться. Разумнее значения хранить тоже как массив из двух элементов.

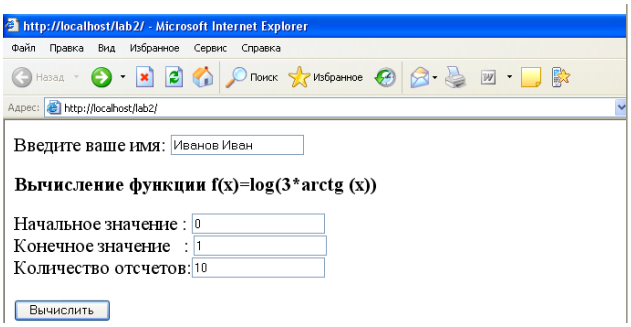
Фамилию, имя и отчество передавать в виде параметров. Для корректного обращения желательно передавать пол адресата.

Примерный текст приглашения должен быть следующим:

«Уважаемая Наталья Александровна!

Рады пригласить Вас участвовать в нашей конференции по проблемам качества образования, которая будет проходить 21—22 июня 2007 года по адресу Российская Федерация, Санкт-Петербург, Невский 58. Вы являетесь участником пленарного заседания, которое будет проходить в Дубовом зале в 10.00 21-го июня.»

Задание 7. Создать форму следующего содержания. Написать обработчик данных формы, который представляет результаты в виде следующей таблицы:



Задание выполнил: Иванов Иван

Таблица значений функции f(x)=log(3*arctg (x))

x	f(x)
0	-INF
0.1	-1.20729178113
0.2	-0.523933356729
0.3	-0.134251213424
0.4	0.13235994667
0.5	0.329981810013
0.6	0.483202700004
0.7	0.605505364795
0.8	0.705185837594
0.9	0.787750431167
1	0.857047813398

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. Как описать функцию?
2. Как использовать математическую функцию?
3. Как считать данные с текстового поля формы?
4. Как выполнить PHP-код?

Лабораторная работа № 11

Разработка серверных сценариев для работы с классами и объектами

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Определение класса

Класс - это шаблон кода, который используется для создания объектов. Класс определяется с помощью ключевого слова **class,** после которого указывается произвольное имя класса. В имени класса может использоваться любое сочетание букв и цифр, но они не должны начинаться с цифры. Код, связанный с классом должен быть заключен в фигурные скобки, которые указываются после имени. Определение класса описывает, какие элементы будут содержаться в каждом новом экземпляре этого класса.

С8интаксис определения класса на примере:

```
<?php
  class first {
    // Тело класса
  }
?>
```

Класс first из приведенного примера - уже полноправный класс, хотя пока и не слишком полезный. Но тем не менее мы сделали нечто очень важное. Мы определили тип, т.е. создали категорию данных, которые мы можем использовать в своих сценариях. Важность этого станет для вас очевидной по мере дальнейшего чтения главы.

Создание объекта

Так как класс - это шаблон для создания объектов, следовательно, **объект** - это данные, которые создаются и структурируются в соответствии с шаблоном, определенным в классе. Объект также называют экземпляром класса, тип которого определяется классом. Для создания нового экземпляра класса нам понадобится оператор **new**. Он используется совместно с именем класса следующим образом:

```
<?php
  // Ключевое слово new сообщает интерпретатору PHP о необходимости
  // создать новый экземпляр класса first
  $obj1 = new first();
  $obj2 = new first();
?>
```

После оператора new указывается имя класса, на основе которого будет создан объект. Оператор new создает экземпляр класса и возвращает ссылку на вновь созданный объект. Эта ссылка сохраняется в переменной соответствующего типа. В результате выполнения этого кода будет создано два объект типа first. Хотя функционально они идентичны (т.е. пусты) $obj1 и $obj2 - это два разных объекта одного типа, созданных с помощью одного класса.

Представьте, что класс - это форма для отливки, с помощью которой изготавливаются пластмассовые машинки. Объекты - это и есть машинки. Тип создаваемых объектов определяется формой отливки. Машинки выглядят одинаковыми во

всех отношениях, но все-таки это разные предметы. Другими словами, это разные экземпляры одного и того же типа.

Определение свойств

В классе можно определить переменные. Переменные, которые определены в классе называются свойствами (или полями данных). Они определяются с одним из ключевых слов protected, public или private, характеризующих управление доступом. Определим некоторые свойства с помощью ключевого слова public:

```
<?php
  class first {
    public $num = 0;
    public $str = 'some text';
  }
?>
```

Как видите, мы определили два свойства, присвоив каждому из них значение. Теперь любые объекты, которые мы будем создавать с помощью класса first, будут иметь два свойства с указанными значениями.

Примечание: **значения инициализирующие свойства должны быть литералами (константными значениями), инициализировать свойства в классе не обязательно (если значение не указано, по умолчанию это будет NULL).**

К свойствам объекта можно обращаться с помощь символов '->', указав объект и имя свойства. Поскольку свойства объектов были определены как public, мы можем считывать их значения, а также присваивать им новые значения, заменяя тем самым начальные значения, определенные в классе:

```
<?php
  class first {
    public $num = 0;
    public $str = 'some text';
  }
  $obj = new first();
  echo $obj->str;
  // присваиваем свойству объекта новое значение
  $obj->str = 'новая строка';
  echo "<br>$obj->str";
?>
```

Работа с методами

Методы - это обычные функции, которые определяются внутри класса, они позволяют объектам выполнять различные задачи. Объявление метода напоминает определение обычной функции, за исключением предваряемого одного из ключевых слов protected, public или private. Если в определении метода вы опустите ключевое слово, определяющее видимость, то метод будет объявлен неявно как public. К методам объекта можно обращаться с помощь символов '->', указав объект и имя метода. При вызове метода, так же как и при вызове функции нужно использовать круглые скобки.

```
<?php
  class first {
    public $str = 'some text';
    // определение метода
    function getstr() {
      echo $this->str;
    }
  }
  $obj = new first();
```

```
// вызов метода объекта
$obj->getstr();
?>
```

Мы добавили метод getstr() к классу first. Обратите внимание на то, что при определении метода мы не использовали ключевое слово, определяющее область видимости. Это означает, что метод getstr() относится к типу public и его можно вызвать за пределами класса.

В определении метода мы воспользовались специальной псевдопеременной **$this**. Она используется для обращения к методам или свойствам внутри класса и имеет следующий синтаксис:

```
$this->имя переменной или метода
```

Значением переменной $this является ссылка на текущий объект. Чтобы стало понятнее посмотрите на следующий пример:

```
class first {
  public $str = 'some text';
  // при определении метода в классе, переменная $this не имеет никакого значения
  function getstr() {
    echo $this->str;
  }
}
// создаем объект
$obj = new first();
// созданный нами объект имеет свойство и метод
// теперь в методе объекта переменная $this имеет
// ссылку на текущий объект, а именно на $obj
// т.е. если в методе заменить $this текущим экземпляром объекта
$this->str;
// это будет выглядеть как простое
// обращение к свойству текущего объекта
$obj->str;
```

Примечание: **переменной $this нельзя ничего присваивать. Помните, что переменная $this всегда ссылается на текущий объект.**

Специальный метод - конструктор

У класса может быть определен специальный метод - **конструктор**, который вызывается каждый раз при создании нового экземпляра класса (объекта) с целью инициализировать его, например, установить значения свойств. Конструктор, как и любой другой метод, может иметь параметры. Чтобы определить метод в качестве конструктора его необходимо назвать __construct(). Обратите внимание на то, что имя метода должно начинаться с двух символов подчеркивания. Посмотрим, как это работает:

```
<?php
 class first {
  // определяем два свойства
  public $num1 = 0;
  public $num2 = 0;
  // определяем конструктор класса
  function __construct($num1, $num2) {
   $this->num1 = $num1;
   $this->num2 = $num2;
  }
  // метод, который складывает два числа
  function summa() {
```

```
    return $this->num1 + $this->num2;
  }
 }
 // создаем объект и передаем два аргумента
 $obj = new first(15, 35);
 // вызываем метод и сразу выводим результат его работы
 echo $obj->summa();
?>
```

Метод __construct вызывается, когда создается объект с помощью оператора new. Указанные в скобках аргументы передаются конструктору. В методе конструктора используется псевдопеременная $this для присвоения значений соответствующим свойствам создаваемого объекта.

Примечание: если конструктор не имеет параметров и при создании новых экземпляров класса не передаются никакие аргументы, круглые скобки () после имени класса можно опустить:

```
$obj = new first;
```

Указание типа аргумента в методах

По умолчанию метод может принимать аргументы любого типа, но бывают случаи, когда необходимо сделать так, чтобы метод мог принимать в качестве аргумента только экземпляры определенного класса. Для указания типа принимаемого аргумента, просто поместите в определении метода перед именем параметра название класса:

```
<?php
 // определяем два пустых класса
 class cat {}
 class wrong {}
 class write {
  // метод, который принимает аргументы только типа cat
  function getobj(cat $getCat) {   // определяем параметр типа cat
   echo 'Получен объект типа cat';
  }
 }
 // создаем экземпляр типа write
 $kitty = new write();
 // работает: передали в качестве аргумента экземпляр типа cat
 $kitty->getobj( new cat() );
 // здесь будет ошибка: передали в качестве аргумента экземпляр типа wrong
 $kitty->getobj( new wrong() );
?>
```

Теперь в качестве аргумента методу getobj() можно передавать только экземпляры типа cat. Поскольку метод getobj() содержит уточнение типа класса, передача ему объекта типа wrong приведет к ошибке.

Указание типа нельзя использовать для определения параметров элементарных типов, таких как строки, числа и т.д. Для этой цели в теле метода следует использовать функции проверки типов, например is_string(). Также есть возможность определить, что передаваемый аргумент является массивом:

```
<?php
 class write {
  $my_arr;
  // аргументом для метода может быть только массив
  function setArray(array $some_arr) {
   this->my_arr = $some_arr;
  }
```

```
  }
?
```

И последнее о чем осталось сказать: если параметр метода определяется с указанием определенного класса, разрешается указать значение по умолчанию, на случай, если методу не было передано никакого объекта. В качестве значения по умолчанию может быть использовано только значение NULL:

```
function getobj(cat $getCat = null) {
  $this->someVar = $getCat;
 }
```

Если вместо NULL указать какое-либо другое значение по умолчанию, будет выдана ошибка.

Практическая часть

1. Выполнить примеры из теоретической части.
2. Выполнить задания по вариантам самостоятельно

Разработать классы для описанных ниже объектов. Включить в класс методы set (…), get (…), show (…). Определить другие методы.

1. **Student**: Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс. Создать массив объектов. Вывести:
   а) список студентов заданного факультета;
   б) списки студентов для каждого факультета и курса;
   в) список студентов, родившихся после заданного года.
2. **Abiturient**: Фамилия, Имя, Отчество, Адрес, Оценки. Создать массив объектов. Вывести:
   а) список абитуриентов, имеющих неудовлетворительные оценки;
   б) список абитуриентов, сумма баллов у которых не меньше заданной;
   в) выбрать N абитуриентов, имеющих самую высокую сумму баллов, и список абитуриентов, имеющих полупроходной балл.
3. **Aeroflot**: Пункт назначения, Номер рейса, Тип самолета, Время вылета, Дни недели. Создать массив объектов. Вывести:
   а) список рейсов для заданного пункта назначения;
   б) список рейсов для заданного дня недели;
   в) список рейсов для заданного дня недели, время вылета для которых больше заданного.
4. **Book**: Автор, Название, Издательство, Год, Количество страниц. Создать массив объектов. Вывести:
   а) список книг заданного автора;
   б) список книг, выпущенных заданным издательством;
   в) список книг, выпущенных после заданного года.

5. **Worker**: Фамилия и инициалы, Должность, Год поступления на работу, Зарплата. Создать массив объектов. Вывести:

а) список работников, стаж работы которых на данном предприятии превышает заданное число лет;

б) список работников, зарплата которых больше заданной;

в) список работников, занимающих заданную должность.

6. **Train**: Пункт назначения, Номер поезда, Время отправления, Число общих мест, Купейных, Плацкартных. Создать массив объектов. Вывести:

а) список поездов, следующих до заданного пункта назначения;

б) список поездов, следующих до заданного пункта назначения и отправляющихся после заданного часа;

в) список поездов, отправляющихся до заданного пункта назначения и имеющих общие места.

7. **Product**: Наименование, Производитель, Цена, Срок хранения, Количество. Создать массив объектов. Вывести:

а) список товаров для заданного наименования;

б) список товаров для заданного наименования, цена которых не превышает указанной;

в) список товаров, срок хранения которых больше заданного.

8. **Patient**: Фамилия, Имя, Отчество, Адрес, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести:

а) список пациентов, имеющих данный диагноз;

б) список пациентов, номер медицинской карты которых находится в заданном интервале.

9. **Bus**: Фамилия и инициалы водителя, Номер автобуса, Номер маршрута, Марка, Год начала эксплуатации, Пробег. Создать массив объектов. Вывести:

а) список автобусов для заданного номера маршрута;

б) список автобусов, которые эксплуатируются больше 10 лет;

в) список автобусов, пробег у которых больше 10 000 км.

10. **Customer**: Фамилия, Имя, Отчество, Адрес, Телефон, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести:

а) список покупателей в алфавитном порядке;

б) список покупателей, номер кредитной карточки которых находится в заданном интервале.

в) список квартир, имеющих площадь, превосходящую заданную.

14. **Phone**: Фамилия, Имя, Отчество, Адрес, Номер, Время внутригородских разговоров, Время междугородних разговоров. Создать массив объектов. Вывести:

а) сведения об абонентах, время внутригородских разговоров которых превышает заданное;

б) сведения об абонентах, воспользовавшихся междугородней связью;

в) сведения об абонентах, выведенные в алфавитном порядке.

15. **Person**: Фамилия, Имя, Отчество, Адрес, Пол, Образование, Год рождения. Создать массив объектов. Вывести:

а) список граждан, возраст которых превышает заданный;

б) список граждан с высшим образованием;

в) список граждан мужского пола.

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 12

Обработка данных, введенных в форму

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Работа с формами

Для передачи данных от пользователя Web-страницы на сервер используются HTML-формы. Для работы с формами в PHP предусмотрен ряд специальных средств.

Изучим особенности языка PHP по передачи на сервер данных введенных пользователем специфику передачи данных методами **POST** и **GET**; динамически формировать странички с использованием данных форм.

Форма в HTML –документе реализуется тегом-контейнером FORM, в котором задаются все управляющие элементы – поля ввода, кнопки и т.д. Если управляющие элементы указаны вне тега FORM, то они не создают форму, а используются для построения пользовательского интерфейса на веб-странице.

В теге FORM обязательно необходимо задать два атрибута – это **method** и **action**. Атрибут **method** устанавливает каким образом данные будут отправлены на сервер. Существуют два основных метода : **GET** и **POST**. Метод **POST** является более мощным и гибким, и поэтому его мы будем использовать в решении практических примеров.

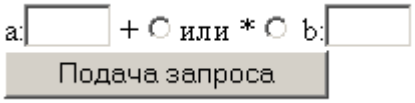
Другой атрибут тега **FORM** – это атрибут **action**. С его помощью устанавливают URL программы, которая будет обрабатывать форму. Этот атрибут связывает веб-страницу с программой, которая считывает с нее данные и выдает результаты в виде другой страницы. В URL может быть записана абсолютная ссылка (начинается на **http://**) и содержит полное доменное имя обрабатывающей программы или относительная ссылка (это означает, что программа находится в том же каталоге что и исходная веб-страница).

Необязательный параметр **enctype** указывает тип содержимого формы, используемый для определения формата кодирования при ее пересылке. В HTML определены два возможных значения атрибута **enctype: application/x-www-form-urlencoded** (используется по умолчанию) и **multipart/form-data.**

Практическая часть

3. Рассмотрим пример.

Самостоятельно создайте в папке **C:\WebServers\home\myserver.ru\www** новую html-страницу **primer1.html**, который будет содержать форму с двумя полями для ввода чисел и переключатель, который будет определять, какое действие надо выполнить с числами (сложить или перемножить):



Пример 11. (файл primer11.html)

```
<html>
<head>
<title>Заполнение простой формы</title>
</head>
<body>
```

```
<form method="post" action="primer11.php">
   a:<input type="text" name="a" size="3">
   +<input type="radio" name="diya" value="plus">
или  *<input type="radio" name="diya" value="umnozh">
   b:<input type="text" name="b" size="3"><br>
   <input type="submit">
</form>
</body>
</html>
```

Сохраните файл на своем сервере, откройте браузер, введите в адресной строке **http://myserver.ru/primer11.html** и проверьте его работоспособность.

В этом примере страница **primer11.html** содержит форму, атрибут **action** которой присвоено значение **primer11.php**. Как только пользователь нажмет на кнопку отправки формы, значения всех полей собираются в один пакет и передаются в программу с названием **primer11.php,** которая по умолчанию должна быть расположена в том же каталоге, что и исходная страница **primer11.html.**

Создадим сценарий для извлечения и обработки данных.

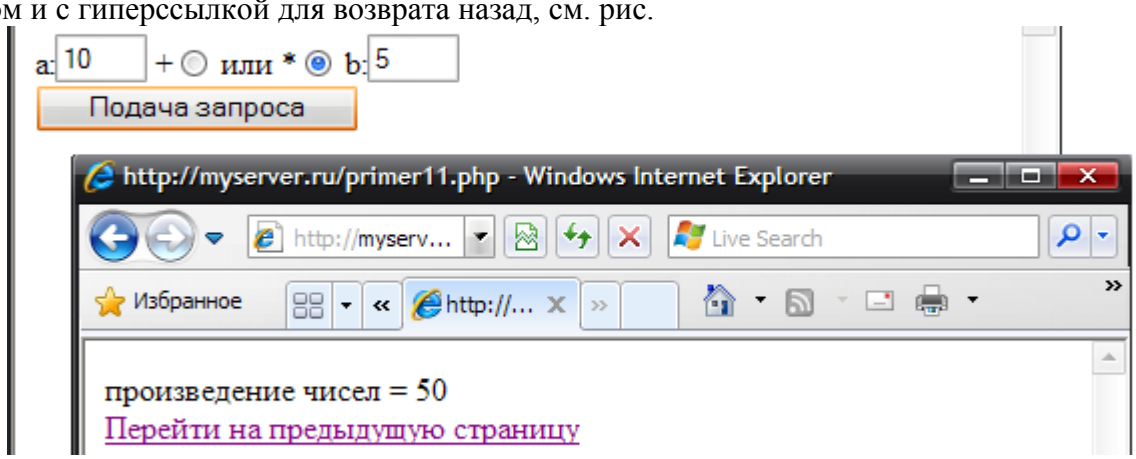
Создайте php-файл **primer11.php**, который будет обрабатывать информацию введенную форме. В зависимости от выбранного положения переключателя выполнить соответствующее действие. Добавим также ссылку для возврата на предыдущую страницу.

Файл primer11.php:

```
<?
 if ($_POST['diya']=='plus')
   {
      $c=$_POST['a']+$_POST['b'];
      echo "сумма чисел = $c";
   } else {
      $c=$_POST['a']*$_POST['b'];
      echo "произведение чисел = $c";
   }
echo "<br><a href='primer11.html'> Перейти на предыдущую страницу</a>";
?>
```

Сохраните файл в текущую папку **C:\WebServers\home\myserver.ru\www.**

Для того, чтобы проверить работоспособность данного запроса, запустите html-файл (http://myserver.ru/primer11.html) и выберите действие над переменными **a** и **b**, нажмите кнопку **Подача запроса**. В результате у вас загрузится сценарий **primer11.php** с ответом и с гиперссылкой для возврата назад, см. рис.



Программа **primer11.php** ожидает данные со страницы **primer11.html.** когда пользователь отправляет html-форму php-программа выдает результат, подобный на рис.

В общем случае тип элемента, помещенного на HTML-форму, не играет никакой

роли. Интерпретатор PHP просто обрабатывает имя каждого элемента и его значение. К тому моменту когда эти данные попадают на сервер уже не имеет значения какого типа эти данные были использованы. PHP автоматически создает переменную, соответствующую каждому элементу формы. Значением этой переменной становится значение элемента.

Код программы **primer11.php** иллюстрирует сказанное. Форма, вызывающая эту программу, должна содержать элементы под именами a и b. В листинге для того, чтобы получить данные из формы использовался вызов следующего вида **$_POST['a']** и **$_POST['b'].**

Когда пользователь передает форму в программу, интерпретатор PHP для каждого элемента формы исходной html-страницы создает переменную с таким же именем.

Поскольку на странице **primer11.html** присутствуют элементы a и b, то любая PHP-программа, к которой обратится страница **primer11.html**, автоматически получит доступ к переменной с именем **$a** и **$b**. эти переменные будут содержать значения, которые пользователь введет в полях формы, перед тем, как нажмет кнопку отправки формы.

Параметры текстовых полей обрабатываются легко. Интерпретатор создает для каждого параметра переменную с соответствующим именем.

Извлечем данные из других элементов формы.

Программа на PHP способна считать данные из любого элемента HTML формы. Во всех случаях атрибут **Name** объекта html-формы становится на PHP именем переменной. В общем случае этой переменной присваивается значение атрибута **value** объекта формы.

При обработки элемента с многозначным выбором для доступа ко всем выбранным значениям нужно к имени элемента добавить пару квадратных скобок (например, если списку присвоить имя **Item**, то для обращения из сценария к значению этого списка необходимо будет писать **Item[]**).

Например, добавьте в веб-страницу **primer11.html** в конец формы (перед кнопкой **Submit**) описание списка.

```
Что вы будете пить сегодня?<br>
<select name="Item[]" size=5 multiple>
<option>Чай
<option>Кофе
<option>Молоко
<option>Ветчина
<option>Сыр
</select>
<br>
```

Сохраните и обновите страницу в браузере.

При обработке списка с множественным выбором параметры передаются так:

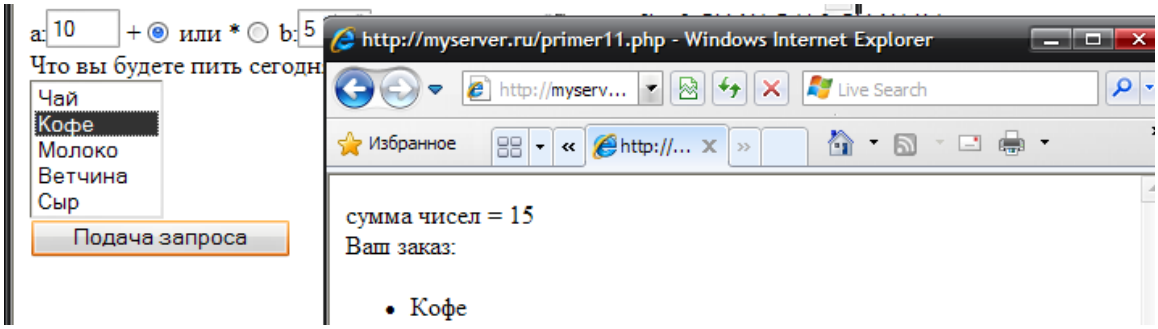
Item=value1&Item=value2….

При стандартном подходе переменная **$Item** будет содержать только последнее значение **valueN**. Однако разработчики PHP позаботились об этом: множественный список можно представить в виде массива, а обрабатывать его элементы можно с помощью цикла **foreach**. Нам даже не нужно знать количество переданных элементов списка. Предварительно нужно лишь сообщить PHP, что будем передавать массив (квадратные скобки – это признак массива).

Для того, что обработать список значений добавьте в сценарий следующий код **primer11.php**:

```
echo "Ваш заказ: <p><ul>";
foreach ( $_POST[Item] as $value )   echo "<li>$value";
echo "</ul>";
```

В результате вы получите:



Рассмотрим как от пользователя принять большой текстовый фрагмент.

Для этого самостоятельно в веб-странице **primer11.html** добавьте в конец формы (перед кнопкой **Submit**) описание текстового поля.

<TEXTAREA COLS="25" ROWS="5" NAME="comment">Ваш комментарий...</TEXTAREA>

Сохраните и обновите страницу в браузере.

Для того, что обработать текстовое поле добавьте в сценарий следующий код **primer11.php**:

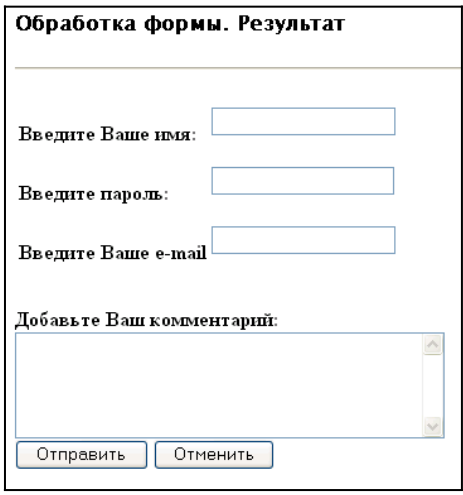
```
## Модель скрипта, принимающего текст от пользователя.
echo "<br>Комментарий: $_POST[comment]";
```

Можно обрабатывать формы, не заботясь о фактических именах полей. Для этого можно использовать (в зависимости от метода передачи) ассоциативный массив **$HTTP_GET_VARS** или **$HTTP_POST_VARS**. Эти массивы содержат пары имя/значение для каждого элемента переданной формы.

Самостоятельно создайте новый файл **primer12.php**:

```
<html>
<head>
<title>Обработка произвольного ввода независимо от метода передачи</title>
</head>
<body>
<?php
$params = ( $REQUEST_METHOD == "GET" )
  ? $HTTP_GET_VARS : $HTTP_POST_VARS;
 foreach ( $params as $key=>$value )
   echo "$key == $value<br>";
?>
</body>
</html>
```

2. Создайте каталог php2 и в нем файл forma.html. Создайте форму по следующему образцу:



1. Создайте файл обработчик формы, который должен выполнять следующие действия:

- Проверку, все ли поля заполнены:

```
If ($_GET[nm]==' ' or ($_GET[pass]==' ' or ($_GET[email]==' ')
{echo "<font color='red'>Вы ввели не все данные</font>";
Exit; }
```

- Выводить приветствие с указанием имени;
- Формировать и отправлять письмо:

```
$komu="xxx.ru";
$tema="Вопрос от ". $_GET['nm']." ".$_GET['email'];
$text_p=$_GET['quest'];
mail($komu, $tema, $text_p);
echo "Ваш вопрос отправлен администратору";
```

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. Какие типы переменных поддерживает язык PHP?
2. В чем отличие php-страницы и html-страницы?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции data()?
5. Что возвращает web-сервер при запросе php-страницы?

Лабораторная работа № 13

Реализация загрузки файлов. Организация файлового ввода-вывода

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

7. Изучить теоретический материал.
8. Проработать примеры.
9. Выполнить задания в соответствии с указаниями.
10. Предъявить преподавателю результаты работы.
11. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
12. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Передача файла на сервер.

PHP позволяет передавать на сервер файлы. HTML-форма, предназначенная для передачи файла, должна содержать аргумент **enctype="multipart/form-data"**.

Кроме того в форме перед полем для копирования файла должно находиться **скрытое поле** с именем **max_file_size**. В это скрытое поле должен быть записан максимальный размер передаваемого файла (обычно не больше 2 Мбайт).

Само поле для передачи файла - обычный элемент **INPUT** с аргументом **type="file"**.

Например в исходной html-странице primer.html, добавьте атрибут **enctype="multipart/form-data"** в описание формы :

```
<form method="post" action="primer11.php" enctype="multipart/form-data">
```

И в конец формы перед кнопкой Submit добавим два элемента:

```
<br>
Загрузить файл на сервер<br>
<input type="hidden" name="max_file_size" value="51200">
<input type="file" name="myfile"><br>
```

После того, как файл передан на сервер, он получает уникальное имя и сохраняется в каталоге для временных файлов. Полный путь к файлу записывается в глобальную переменную, имя которой совпадает с именем поля для передачи этого файла. Кроме этого PHP сохраняет еще некоторую дополнительную информацию о переданном файле в других глобальных переменных:

Переменная	Описание
$myfile	Путь к временному файлу
$myfile_name	Имя переданного файла
$myfile_size	Размер переданного файла
$myfile_type	Тип переданного файла в системе MIME

Для работы с файлами в php существует несколько функций.

`fopen` – функция для открытия файла.

```
$fp = fopen('filename, 'param');
```

filename и param это обязательные параметры. Первый отвечает за имя файла, который необходимо открыть, а второй определяет режим файла:

1. r – открытие файла только для чтения.
2. r+ - открытие файла одновременно на чтение и запись.
3. w – создание нового пустого файла. Если на момент вызова уже существует такой файл, то он уничтожается.
4. w+ - аналогичен r+, только если на момент вызова фай такой существует, его содержимое удаляется.

5. a – открывает существующий файл в режиме записи, при этом указатель сдвигается на последний байт файла (на конец файла).

6. a+ - открывает файл в режиме чтения и записи при этом указатель сдвигается на последний байт файла (на конец файла). Содержимое файла не удаляется.

7. Записывать данные в файл при помощи PHP можно при помощи функции fwrite(). Это функция принимает 2 обязательных параметра и 1 необязательный. В качестве обязательных параметров выступает дескриптор файла и режим файла:

$test = fwrite($fp, $mytext);

По завершению работы с файлом, его нужно закрыть, используя функцию fclose($fp).

Для того, чтобы обработать переданный файл от пользователя введите в сценарий **primer11.php** следующий обработчик:

```
if ( ! isset( $myfile ) )
  {
   echo "путь: $myfile<br>";
   echo "имя: $myfile_name<br>";
   echo "размер: $myfile_size<br>";
   echo "тип: $myfile_type<br>";
  }
```

Доступ к загруженным файлам осуществляется по имени (в примере myfile). При этом не используется описанные выше методы **$_POST, $_GET, $_REQUEST**. Для обработки файлов предназначен специальный массив **$_FILES**. Данный массив является двухмерным, где первым индексом является имя поля для загрузки файла. Второй элемент массива принимает фиксированный набор значений, все его возможные значения представлены в таблице:

Переменная	**Описание**
$_FILES [myfile] [‘name’]	Имя переданного файла
$_FILES [myfile] [‘tmp_name’]	Имя временного файла
$_FILES [myfile] [‘size’]	Размер переданного файла
$_FILES [myfile] [‘type’]	Тип переданного файла в системе MIME

После успешной загрузки содержимое файла сохраняется в каталоге для временных файлов, а имя этого временного файла помещается в элемент массива **S_FILES [myfile] [‘tmp_name’].**

Еще один пример структуры html-файла с формой для загрузки локального файла:

```
<form enctype="multipart/form-data" action="obrabotka.php" method="post">
 <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
 Отправить этот файл: <input name="userfile" type="file" />
 <input type="submit" value="Send File" />
</form>
```

Пример фрагмент файла obrabotka.php

```php
<?php
$uploaddir = '/var/www/uploads/';  // имя папки в которую необходимо будет загрузить файл
$uploadfile = $uploaddir . basename($_FILES['userfile']['name']);//имя файла вместе с путем файла

print "<pre>";
if (move_uploaded_file($_FILES['userfile']['tmp_name'], $uploadfile)) {
    print "Файл успешно загружен на сервер.  Дополнительная отладочная информация:\n";
    print_r($_FILES); // распечатка ассоциативного массива $_FILES
} else {
    print "Файл не загружен. Возможно проблеммы с программистом или сервером!  Дополнительная отладочная информация:\n
    print_r($_FILES);
}
print "</pre>";

?>
```

Практическая часть

1. Выполнить примеры из теоретической части

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 14

Программирование опросов и счетчиков посещения web страницы

Цель: приобретение навыков использования функций обработки файлов на PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

fopen(имя файла, режим работы) – возвращает число – дескриптор открытого файла, по которому можно обращаться к открытому файлу.

Режим работы имеет 2 составляющие:

- способ работы с информацией (текстовый (t) и бинарный(b));
- способ работы с файлом:

Способ	Описание
R	Файл открывается для чтения, указатель текущей позиции в начале файла. Если файла не существует, возникает ошибка
R+	Файл открывается для чтения и записи, указатель текущей позиции в начале файла. Если файла не существует, возникает ошибка
W	Создается пустой файл и открывается для записи, указатель текущей позиции в начале файла. Если файл существует, он перезаписывается
W+	Создается пустой файл и открывается для чтения и записи, указатель текущей позиции в начале файла. Если файл ∃, он перезаписывается
A	Файл открывается для записи, указатель текущей позиции в конце файла. Если файла не существует, он создается
A+	Файл открывается для записи и чтения, указатель текущей позиции в конце файла. Если файла не существует, он создается

flock (дескриптор файла, режим блокировки) – блокирует файл для использования других пользователей.

Режимы блокировки:

2- устанавливает блокировку;

3- снятие блокировки;

fgets(дескриптор файла) – считывает данные из файла.

ftruncate(дескриптор файла, размер) – обрезает данные из файла до заданного размера (указывается в байтах). Возвращает TRUE (при успешном выполнении) или FALSE.

fputs(дескриптор файла, данные) – осуществляет запись данных в файл.

fclose(дескриптор файла) – закрытие файла.

die(текст сообщения об ошибке) – выводит текст, переданный в качестве параметра, и осуществляет выход из программы.

Одновременное использование двух функций:

fopen() or die() – если результат выполнения первой функции FALSE, то в этом случае выполняется вторая функция.

Практическая часть

Количество посещений любой страницы хранится в текстовом файле с именем counter.txt.

1. Ввести код программы-счетчика посещений counter.php

```
<?php
$f=fopen(“counter.txt”, “a+t”) or die(“Невозможно открыть файл”);
flock(  $f, 2);
$s = fgets($f);
$s+=1;     // $s=$s+1;
ftruncate ($f, 0);
fputs ($f, $s);
flock ($f,3);
fclose($f);
echo $s;
?>
```

2. Открыть код страницы forma.html первой практической работы.
3. Добавить код для подключения счетчика в нижней части левой панели:

```
<?php
echo “Количество посещений – “; require_once(“counter.php”);
?>
```

Require_once(имя файла) – подключает модуль, имя которого указано в параметре. В качестве модуля используют программы PHP или HTML.

4. Заменить расширение файла: forma.php

Вариация 2 Создание счетчика.

Код файла count.php:

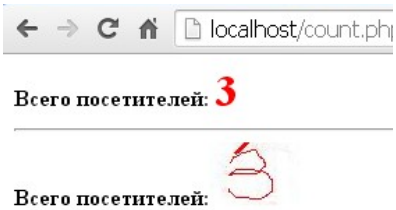
```
<?
$text=0;
$new_text=0;
 // считываем из файла предыдущее значение счетчика
$abc = fopen("counter.txt", "r");
$text= fread($abc,filesize("counter.txt"));
fclose($abc);
 // записываем в файл значение счетчика увеличенное на единицу
$abc = fopen ("counter.txt", "w");
$new_text= (integer) $text + 1;
fwrite ($abc, $new_text);
fclose($abc);
 // вывод текстового сообщения о количестве посетителей
echo "<b>Всего посетителей: <font size=6 color=#FF0000>" . $new_text . "</font></b>";
 $myvar=$new_text;
for ($i=0; $i<10; $i++) {
 // находим и заменяем цифры соответствующими изображениями из каталога img
$myvar = str_replace ("$i","<img src=img/$i.JPG>", $myvar);
    }
 // вывод графического счетчика
echo "<hr><b>Всего посетителей:</b>$myvar";
?>
```

Данный счетчик реализован в виде отдельного php-файла, который можно поместить в любое место вашей страницы используя функцию include () следующим образом:

```
<? include ("count.php"); ?>
```

Значение счетчика хранятся в файле counter. Изображения цифр (создать самостоятельно) для счетчика хранятся в каталоге img - эти файлы являются обычными jpg-файлами, и могут быть заменены на любые другие графические изображения в этом формате.

Результат работы скрипта:



Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 15

Создание базы данных MySQL с помощью утилиты phpMyAdmin

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

СУБД — это приложение, запущенное на некотором компьютере. Также как веб-сервер это приложение имеет архитектуру "Клиент-Сервер". То есть серверная часть этого приложения находится в постоянном ожидании запросов, при получении запроса выполняет этот запрос и отсылает клиенту результат.

Веб-приложение должно иметь возможность соединяться с СУБД, посылать запросы, получать результаты и затем эти результаты обрабатывать.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

В состав денвера входит СУБД под названием phpmyadmin. Чтобы начать работать с ней, нужно запустить денвер и в адресной строке набрать http://localhost/Tools/phpMyAdmin/. Изучите работу с phpmyadmin.

Для работы с базой данных используйте расширение MySQLi. MySQLi является улучшенной версией старого драйвера PHP MySQL, предлагающего различные улучшения.

В файле для регистрации сразу после строчки с началом сессии подключитесь к базе данных. Это делается при помощи функции mysqli_connect(). Она создает соединение с MySQL сервером.

```
$link = mysqli_connect(
'localhost',                /*Хост, к которому мы подключаемся  */
'root',                     /*Имя пользователя */
'',                 /* Используемый пароль */
'');                     /* Базаданных для запросов по умолчанию  */
```

Создайте проверку на подключения к базе данных.

```
if (!$link) {
printf("Невозможно подключиться к базе данных. Код ошибки: %s\n",
mysqli_connect_error());
exit;
}
```

Используя функцию **mysqli_query($link, «sql запрос»)**, можно отправлять различные запросы в базу данных. Для этого используется язык запросов SQL. С синтаксисом языка можно ознакомиться на википедии. Создайте базу данных, если она не существует.

```
mysqli_query($link, 'CREATE DATABASE IF NOT EXISTS my_db');
```

Теперь нужно подключиться к нужной базе. Это можно сделать функцией mysqli_select_db().

```
mysqli_select_db($link,'my_db');
```

mysqli_close

Эта функция разрывает соединение с сервером MySQL, и возвращает true при успешном выполнении операции и false в противном случае. Функция принимает в качестве аргумента дескриптор соединения с базой данных, возвращаемый функцией mysql_connect.

```php
// устанавливаем соединение с базой данных
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
 // Выводим предупреждение
 echo ("<P>В настоящий момент сервер базы данных не доступен, поэтому
 корректное отображение страницы невозможно.</P>");
 // Завершаем работу в случае неудачи
 exit();
}
if(mysql_close($dbcnx)) // разрываем соединение
{
 echo("Соединение с базой данных прекращено");
}
else
{
 echo("Не удалось завершить соединение");
}
```

mysql_select_db

функция mysql_select_db выбирает базу данных для дальнейшей работы, и все последующие SQL-запросы применяются к выбранной базе данных. Функция принимает в качестве аргументов название выбираемой базы данных database_name и дескриптор соединения resource. Функция возвращает true при успешном выполнении операции и false в противном случае. К примеру:

```php
<?php
// Код соединения с базой данных
if (!@mysql_select_db($dbname, $dbcnx))
{
 echo( "<P>В настоящий момент база данных не доступна, поэтому
 корректное отображение страницы невозможно.</P>" );
 exit();
}
?>
```

Имеет смысл помещать функции для соединения и выбора базы данных в тот же файл (config.php), где объявлены переменные с именами сервера, пользователя и паролем:

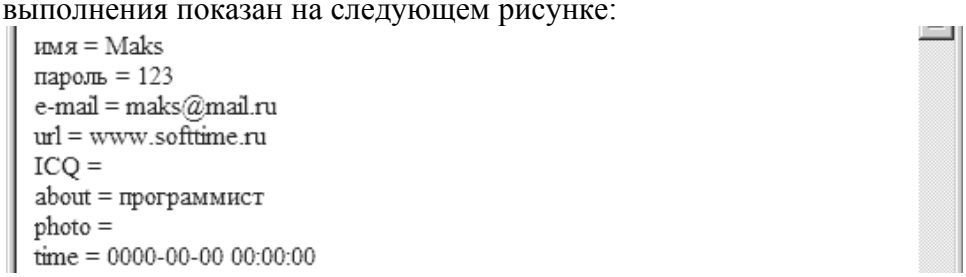
```php
<?php
$dblocation = "localhost";
$dbname = "softtime";
$dbuser = "root";
$dbpasswd = "123456";
$dbcnx = @mysql_connect($dblocation,$dbuser,$dbpasswd);
if (!$dbcnx)
{
 echo( "<P>В настоящий момент сервер базы данных не доступен, поэтому
 корректное отображение страницы невозможно.</P>" );
 exit();
}
if (!@mysql_select_db($dbname, $dbcnx))
{
 echo( "<P>В настоящий момент база данных не доступна, поэтому
 корректное отображение страницы невозможно.</P>" );
 exit();
}
?>
```

mysql_query

Эта функция применяется для отправки серверу SQL-запросов. Функция возвращает дескриптор запроса в случае успеха и false в случае неудачного выполнения запроса. В данном примере показан код, с помощью которого извлекается одна строка из таблицы **authors** базы данных **forum.**

```php
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
 $author = mysql_fetch_array($ath);
 echo "<br>имя = ".$author['name']."<br>";
 echo "пароль = ".$author['passw']."<br>";
 echo "e-mail = ".$author['email']."<br>";
 echo "url = ".$author['url']."<br>";
 echo "ICQ = ".$author['icq']."<br>";
 echo "about = ".$author['about']."<br>";
 echo "photo = ".$author['photo']."<br>";
 echo "time = ".$author['time'];
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b></p>";
 exit();
}
?>
```

Результат выполнения показан на следующем рисунке:



mysql_fetch_array

Эта функция возвращает значения полей в виде ассоциативного массива, в качестве аргумента принимает дескриптор запроса возвращаемый функцией mysql_query. Вот как с помощью этой функции можно вывести все строки таблицы authors:

```
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
 // Определяем таблицу и заголовок
 echo "<table border=1>";
 echo "<tr><td>имя</td><td>пароль</td><td>e-mail</td><td>url</td></tr>";
 // Так как запрос возвращает несколько строк, применяем цикл
 while($author = mysql_fetch_array($ath))
 {
 echo "<tr><td>".$author['name']." </td><td>".$author['passw']."
   </td><td>".$author['email']." </td><td>".
 $author['url']." </td></tr>";
 }
 echo "</table>";
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b><p>";
 exit();
}
?>
```

Результат показан на следующем рисунке:

имя	пароль	e-mail	url
Maks	123	maks@mail.ru	www.softtime.ru
Igor	123	igor@mail.ru	http://www.softtime.ru
Sergey	123	sergey@mail.ru	http://www.softtime.ru

mysql_result

С помощью этой функции можно получить доступ к отдельному полю записи. Допустим, нам нужно вывести имя автора, который первым найдется в базе данных. Сделать это можно следующим образом:

```
<?php
include "config.php";
$ath = mysql_query("select name from authors;");
if($ath)
{
 echo mysql_result($ath,0,'name');
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b><p>";
 exit();
}
?>
```

mysql_fetch_object

Эта функция возвращает поля записи данных в виде объекта. Ниже приведен пример, в котором с помощью этой функции из таблицы authors выводятся имя, URL и e-mail авторов.

```php
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
 while($row = mysql_fetch_object($ath))
 {
 echo "<p>name: ".$row->name."</p>";
 echo "<p>url: ".$row->url."</p>";
 echo "<p>email: ".$row->email."</p>";
 }
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b><p>";
 exit();
}
?>
```

mysql_fetch_row

В отличие от функции mysql_fetch_object, эта функция возвращает не объект, а массив, в котором содержатся значения полей:

```php
<?php
include "config.php";
$ath = mysql_query("select * from authors;");
if($ath)
{
 while($row = mysql_fetch_row($ath))
 {
 echo "<p>name: ".$row[1]."</p>";
 echo "<p>url: ".$row[4]."</p>";
 echo "<p>email: ".$row[3]."</p>";
 }
}
else
{
 echo "<p><b>Error: ".mysql_error()."</b><p>";
 exit();
}
?>
```

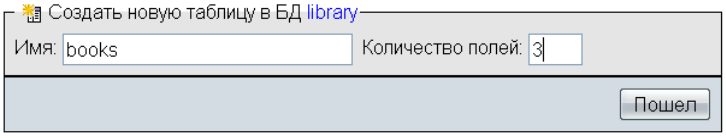
Указания по выполнению

Задание 1.

Выполните примеры из теоретической части. Для их реализации в ***phpMyAdmin*** ***(http://localhost/phpmyadmin)*** создайте базу данных **forum** с таблицей **authors**, поля таблицы определить из примера использования функции mysql_query

Задание 2.

1. Открыть в браузере phpMyAdmin http://localhost/phpmyadmin.
2. Создайте базу данных library451 или library452. В поле «Новая база данных» укажите имя и нажмите кнопку «Создать».
3. Сделайте базу данных library451 активной. Для этого в левом фрейме выберите базу library451. Сейчас в базе нет таблиц.
4. Создайте в базе данных таблицу books, состоящую из трех полей.



5. Укажите типы полей: id – int ключевое, autoincriment, name и title – varchar на 255 символов.

6. Используя команду верхнего меню PhpMyAdmin «Вставить» добавьте название 3-4 книг и авторов.

7. Используя команду «Обзор» просмотрите таблицу.

8. База данных создана, наполнена приступим к программированию.

9. Создайте на своем сайте папку books, а в нее поместите следующие файлы

index.php

```
<html>
<head>
  <title>Управление книгами</title>
</head>

<body>
  <h1>Список книг</h1><a href="new.html">Добавить книгу</a>
  <table border="1">
    <tr>
      <td>#</td>
      <td>Автор</td>
      <td>Название</td>
      <td>Редактить</td>
      <td>Уничтожить</td>
    </tr><?php
    mysql_connect("localhost", "root", "") or die("Невозможно подключиться к серверу");
    mysql_select_db("library") or die("А нет такой таблицы!");
    $rows = mysql_query("SELECT id, author, title FROM books");
    while ($stroka = mysql_fetch_array($rows)) {
      echo "<tr>";
      echo "<td>" . $stroka['id'] . "</td>";
      echo "<td>" . $stroka['author'] . "</td>";
      echo "<td>" . $stroka['title'] . "</td>";
      echo "<td><a href='edit.php?id=" . $stroka['id'] . "'>Редактировать</a></td>";
      echo "<td><a href='delete.php?id=" . $stroka['id'] . "'>Удалить</a></td>";
      echo "</tr>";
    }
    ?>
  </table>
</body>
</html>
```

Просмотрите файл, обратите внимание на ссылки сформированные автоматически.

10. Создайте след. файлы:

new.html

```
<html>
<head>
  <title>Добавление новой книги</title>
</head>

<body>
  <form action="save_new.php" method="get">
    Автор<input name="author"><br>
    Название<input name="title"><br>
    <input type="submit" name="" value="Сохранить">
  </form>
</body>
</html>
```

save_new.php

```php
<html>
<head>
  <title></title>
</head>
<body>
  <?php
  mysql_connect("localhost","root","") or die("Невозможно подключиться к серверу");
  mysql_select_db("library") or die ("А нет такой таблицы!");
  $zapros="INSERT INTO books SET author='".$_GET['author']."', title='".$_GET['title']."'";
  mysql_query($zapros);
  if (mysql_affected_rows()>0)
  {
    echo "Все замечательно сохранено.";
  }
  else
  {
    echo "Ошибочка вышла.";
  }
  ?><a href="index.php">Вернуться к списку книг</a>
</body>
</html>
```

11. Проверьте как работает создание новой книги.
12. Посмотрите через PhpMyAdmin как изменилась таблица books.
13. Создайте следующие файлы:

edit.php

```php
<html>
<head>
  <title>Редактирование книги</title>
</head>
<body>
  <?
  mysql_connect("localhost","root","") or die("Невозможно подключиться к серверу");
  mysql_select_db("library") or die ("А нет такой таблицы!");
  $rows=mysql_query("SELECT author, title FROM books WHERE id=".$_GET['id']);
  while ($stroka=mysql_fetch_array($rows))
  {
    $id=$_GET['id'];
    $author=$stroka['author'];
    $title=$stroka['title'];
  }
  ?>
  <form action="save_edit.php" method="get">
    Автор <input name="author" value="&lt;?echo $author;?&gt;"><br>
    Название <input name="title" value="&lt;?echo $title;?&gt;"><br>
    <input type="hidden" name="id" value="&lt;?echo $id;?&gt;"><br>
    <input type="submit" value="Сохранить">
  </form>
</body>
</html>
```

save_edit.php

```
<body>
  <?php
  mysql_connect("localhost","root","") or die("Невозможно подключиться к серверу");
  mysql_select_db("library") or die ("А нет такой таблицы!");
  $zapros="UPDATE books SET author='".$_GET['author']."', title='".$_GET['title']."' WHERE id=".$_GET['id'];
  mysql_query($zapros);
  if (mysql_affected_rows()>0)
  {
    echo "Все замечательно изменино.";
  }
  else
  {
    echo "Ошибочка вышла.";
  }
  ?><a href="index.php">Вернуться к списку книг</a>
</body>
```

14. Проверьте как редактируются книги
15. Создайте файл для удаления страниц

delete.php

```
<?php
mysql_connect("localhost","root","") or die("Невозможно подключиться к серверу");
mysql_select_db("library") or die ("А нет такой таблицы!");
$zapros="DELETE FROM books WHERE id=".$_GET['id'];
mysql_query($zapros);
header('Location: index.php');
?>
```

16. Проверьте удаление книг. Обратите внимание на то, как работает функция header().
17. Для авторизации доступа создайте (скопируйте и измените) файлы .htaccess и .htpasswd. Поместите их в ту директорию, которая должна быть защищена паролем (логин: admin, пароль: admin). Что бы добавить в файл свои логин и пароль можно воспользоваться утилитой htpasswd.
18. Дополните в библиотечку возможностью загружать книги в электронном формате и Preview с обложками книг.

Содержание отчета

1. Цель
2. Последовательность работы с базой данных в MySQL
3. Функции взаимодействия с MySQL: **mysql_connect, mysql_select_db, mysql_query, mysql_fetch_array, mysql_result, mysql_result, mysql_fetch_object, mysql_fetch_row, mysql_close.**
4. Ход работы с выдержками из сценариев.
5. Выводы

Контрольные вопросы

1. Что такое PHP?
2. Что такое MySQL?
3. Язык SQL, основные конструкции.
4. Что необходимо для работы с базой данных?
6. Функции взаимодействия с MySQL: **mysql_connect, mysql_select_db, mysql_query, mysql_fetch_array, mysql_result, mysql_result, mysql_fetch_object, mysql_fetch_row, mysql_close.**

Лабораторная работа № 16

Извлечение информации из базы данных на PHP

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Для работы с базой данных используйте расширение MySQLi. MySQLi является улучшенной версией старого драйвера PHP MySQL, предлагающего различные улучшения.

В файле для регистрации сразу после строчки с началом сессии подключитесь к базе данных. Это делается при помощи функции mysqli_connect(). Она создает соединение с MySQL сервером.

```
$link = mysqli_connect(
            'localhost',   /* Хост, к которому мы подключаемся */
            'root',        /* Имя пользователя */
            '',   /* Используемый пароль */
            '');      /* База данных для запросов по умолчанию */
```

Создайте проверку на подключения к базе данных.

```
if (!$link) {
   printf("Невозможно подключиться к базе данных. Код ошибки: %s\
       n", mysqli_connect_error());
   exit;
}
```

Используя функцию **mysqli_query($link, «sql запрос»)**, можно отправлять различные запросы в базу данных. Для этого используется язык запросов SQL. С синтаксисом языка можно ознакомиться на википедии. Создайте базу данных, если она не существует.

```
mysqli_query($link, 'CREATE DATABASE IF NOT EXISTS my_db');
```

Теперь нужно подключиться к нужной базе. Это можно сделать функцией mysqli_select_db().

```
mysqli_select_db($link,'my_db');
```

Практическая часть

Задание.

- Создать БД согласно варианту задания. Заполнить таблицу данными
- Создать HTML-документ с формой, служащей для ввода исходных данных и передачи их PHP-скрипту
- Создать PHP-скрипт для получения из БД информации согласно указанным в задании запросам.
- Организовать в скрипте вывод результата в Web-браузер пользователя.
- Предусмотреть операторы диагностики и обработки исключительных ситуаций

Варианты индивидуальных заданий.

1.*Базы данных:* Автозаправочные станции = {Название АЗС, объем бензина (л), город }.

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

Найти суммарный объем бензина на бензоколонках города, название которого ввел пользователь;

Найти бензоколонки, в названии которых присутствует строка, введенная пользователем;

Найти бензоколонки, в названии которых количество бензина больше введенного пользователем значения.

2. *Базы данных:* Детские сады = {Название детского сада, количество детей, количество работников, средняя зарплата }.

Необходимо реализовать в PHP – сценариях следующие запросы к таблице из этой базе данных:

· Найти детские сады, в которых средняя зарплата меньше введенного пользователем значения

· Найти детские сады, в названии которых присутствует строка, введенная пользователем;

· Найти количество работников и количество детей в детском саду, название которого ввел пользователь.

3. *Базы данных:* Программные Продукты (ПП) ={Название ПП, стоимость , Назначение фирмы-разработки }.

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

· Найти самый дорогой ПП фирмы, название которого ввел пользователь;

· Найти все программные продукты фирмы, название которой ввел пользователь;

· Найти программные продукты, в названии фирмы-разработчика которых присутствует строка, введенная пользователем.

4. *Базы данных:* Продуктовые магазины (ПМ) ={Название ПМ, город, Год, Объем продаж (тыс. руб)}.

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

Найти магазины, у которых объем продаж в 2009 г. превысил указанное пользователем значение;

·Найти магазины города, название города которого ввел пользователь;

найти суммарный объем продаж в магазинах , в названии которых присутствует строка, введенная пользователем.

5. *Базы данных:* Компьютеры = ={Номер компьютера, Объем свободного места на жестких дисках, Номера дисплейного класса}.

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

найти компьютеры, у которых свободное место на жестких дисках превышает введенное пользователем значение

найти дисплейные классы, в которых количество компьютеров больше или равно введенному пользователем числу;

найти компьютеры дисплейного класса, номер которого задал пользователь.

6. *Базы данных:* Книги = {Название книги, Типография, Количество страниц, Год издания }

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

найти названия книг, напечатанных в том году, который ввел пользователь;

·найти типографии, которые выпустили n и более различных книг, где n – это число, введенное пользователем;

·найти книги, имеющие количество страниц от A до B, где A и B – введенные пользователем числа.

7. *Базы данных:* Документы = {Наименование документа, количество страниц, Название кафедры}

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

найти общее количество документов кафедры, название которой ввел пользователь

найти документы, имеющие количество страниц от A до B, введенные пользователем числа.

Найти кафедры, на которых разработаны документы, в названии которых содержится строка, введенная пользователем.

8. *Базы данных:* Изделия = {Наименование изделия, Количество входящих в него деталей, Общая трудоемкость изготовления изделия (час), Стоимость}

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

Найти изделия, у которых количество деталей больше или равно введенному пользователем числу;

Найти названия и трудоемкость изготовления изделий, стоимость которых меньше введенного пользователем значения;

Найти информацию по изделиям, в названии которых присутствует строка, введенная пользователем.

9. *Базы данных:* Кабинеты = {Номер кабинета, Номер корпуса, Количество компьютеров, Количество парт}

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

Найти кабинет, в котором количество парт и количество компьютеров превышают введенные пользователем значения;

Найти количество компьютеров в корпусе, номер которого ввел пользователь;

Найти корпус, в котором суммарное количество компьютеров не меньше, чем значение, введенное пользователем.

10. *Базы данных:* Участники конференции ={ФИО, название, доклада, город, Название секции}

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

Найти участников, которые приехали из города, название которого ввел пользователь;

Найти секции с количеством докладчиков большим или равными числу, введенному пользователем;

Найти количество участников секции, для которой пользователь ввел название или его часть.

11. *Базы данных:* Автомобили ={Модель автомобиля, цена, название завода-изготовителя}

В PHP – сценариях необходимо реализовать следующие запросы к таблице из этой базы данных:

Найти автомобили, цена которых не превышает введенное пользователем значение;

Найти заводы-изготовители, количество выпускаемых моделей которых не меньше, чем число, введенное пользователем;

Найти информацию по автомобилям, в названии которых присутствует строка, введенная пользователем.

Содержание отчета

1. Цель
2. Последовательность работы с базой данных в MySQL
3. Функции взаимодействия с MySQL: Ход работы с выдержками из сценариев.
4. Выводы

Контрольные вопросы

1. Что такое MySQL?

2. Язык SQL, основные конструкции.
3. Что необходимо для работы с базой данных?
4. Функции взаимодействия с MySQL:

Лабораторная работа № 17

Организация поддержки базы данных в PHP

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP Приобретение навыков создания и управления базой данных с помощью программы phpMyAdmin.

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Структура базы данных TOVARS:

1. Таблица tovar, содержит учетные записи товаров

№	Название поля	Описание	Тип
1	id	Поле-счетчик	INT
2	name	Название товара	VARCHAR (20)
3	cost	Стоимость	INT
4	kol	Количество товара	INT
5	date	Дата реализации	DATE

Пример записей:

id	name	cost	kol	date
1	Хлеб столовый	24	100	25.03.10
2	Хлеб ржаной	20	50	27.03.10

Выбор данных:

SELECT column1,… FROM table WHERE definition ORDER BY col_name

Добавление данных:

INSERT INTO table VALUES (value1, …)

Удаление данных:

DELETE FROM table WHERE definition

Основные функции для работы с MySQL:

int mysql_connect(string hostname, string username, string password) - создать соединение с *MySQL*. Функция возвращает параметр типа *int*, который больше 0, если соединение прошло успешно, и равен 0 в противном случае.

hostname – имя хоста, на котором находится база данных.

Username – имя пользователя.

Password – пароль пользователя.

int mysql_select_db(string database_name, int link_identifier) - выбрать базу данных для работы. Функция возвращает значение true или false

Database_name – имя базы данных.

link_identifier – ID соединения, которое получено в функции *mysql_connect*. (параметр необязательный, если он не указывается, то используется *ID* от последнего вызова *mysql_connect*)

int mysql_query(string query, int link_identifier) - функция выполняет запрос к базе данных. Функция возвращает *ID* результата или 0, если произошла ошибка.

query – строка, содержащая запрос *link_identifier* – см. предыдущую функцию.

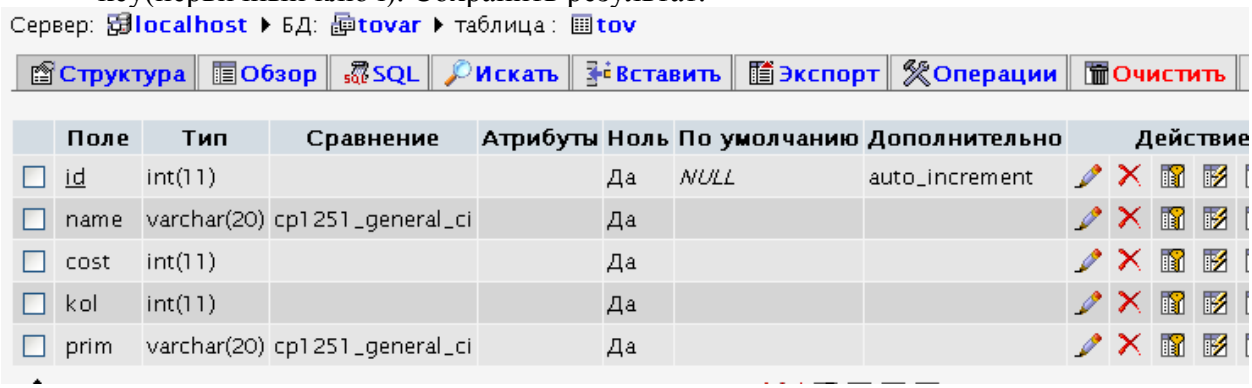
int mysql_result(int result, int i, column) - функция возвращает значение поля в столбце *column* и в строке *i*.

int mysql_close(int link_identifier) - функция закрывает соединение с *MySQL*. Функция возвращает значение *true* или *false*.

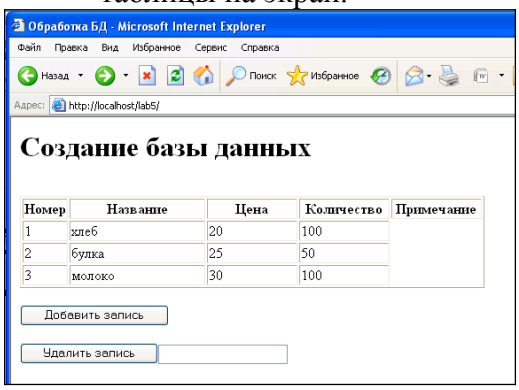
link_identifier – см. выше.

Порядок выполнения работы

1. Открыть программу phpMyAdmin: набрать в строке браузера localhost и выбрать пункт phpMyAdmin.
2. В разделе Привелегии добавить нового пользователя homeuser со всеми правами.
3. В поле Создать новую БД ввести имя базы TOVAR – Создать (Create Database)
4. Создать таблицу tov:
   a. В поле Имя ввести tov;
   b. В поле Поля – число 5;
   c. Кнопка Пошел.
5. Указать название полей таблицы и их тип. В поле Дополнительно указать auto_increment (автоматическое присвоение). Установить переключатель primary key(первичный ключ). Сохранить результат.



6. Ввести 4 записи для данных базы.
7. Создать папку lab4. В ней файл index.php, в котором описать код вывода данных таблицы на экран.



Пример кода:

```
<?php
echo "<h1>Создание базы данных</h1><br>";
echo "<table border=1><tr  align=center><td width=10%><b>Номер</td><td
width=30%><b>Название</td><td width=20%><b>Цена</td><td
width=20%><b>Количество</td><td width=20%><b>Примечание</td></tr>";

$sqlhost="localhost"; $sqluser="homeuser"; $sqlpass=""; $bd="TOVARS";
// соединение с базой данных
mysql_connect($sqlhost,$sqluser,$sqlpass) or die ("нет доступа!".mysql_error());
// выбирает базу для последующей работы
mysql_select_db($bd) or die ("нет соединения".mysql_error());
$zap="SELECT * FROM tovar ORDER BY id";
// выполнение SQL-запроса выбора данных из БД
```

```
$zap_res=mysql_query($zap);
while (list($id, $name, $cena, $kol, $prim)=mysql_fetch_row($zap_res))
{
 echo "<tr>
 <td>$id</td>  <td>$name</td>  <td> $cena</td>  <td>$kol</td>  <td>$prim</td> </tr>";
}
echo "</table>"; ?>
```

7. Разместить 2 кнопки Добавить запись и Удалить запись № и текстовое поле для указания № удаляемой записи.
8. Создать файл insert.php, в котором разместить форму для ввода данных в таблицу.

Пример кода:

```
<?php
if (isset($_REQUEST))
{
 foreach($_REQUEST as $key=>$val)
 {$key=$val;}
 }
$sqlhost="localhost"; $sqluser="homeuser"; $sqlpass=""; $bd="TOVARS";
mysql_connect($sqlhost,$sqluser,$sqlpass) or die ("нет доступа!".mysql_error());
mysql_select_db($bd) or die ("нет соединения".mysql_error());
$zap="INSERT INTO tovar( name, cost, kol, prim) VALUES ($name, $cena, $kol,
$prim)";
$zap_res=mysql_query($zap);
if (zap_res==true)
   echo "Запись успешно добавлена";   else  echo "Ошибка при записи данных";
?>
```

9. Создать файл delete.php, в котором описать код для удаления записи по заданному номеру.

Пример кода:

```
<?php
if (isset($_REQUEST))
{
 $num=$_REQUEST[num];
 }
$sqlhost="localhost"; $sqluser="homeuser"; $sqlpass=""; $bd="TOVARS";
mysql_connect($sqlhost,$sqluser,$sqlpass) or die ("нет доступа!".mysql_error());
mysql_select_db($bd) or die ("нет соединения".mysql_error());
$zap="DELETE FROM ‘tovar’ WHERE id = $num ";
$zap_res=mysql_query($zap);
if (zap_res==true)
   echo "Запись успешно удалена";
 else  echo "Ошибка при удалении данных";
?>
```

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?

Лабораторная работа № 18

Отслеживание сеансов (session)

Цель: освоение методов создания пользовательских функций и методов процедурного программирования на языке PHP

Ход работы:

1. Изучить теоретический материал.
2. Проработать примеры.
3. Выполнить задания в соответствии с указаниями.
4. Предъявить преподавателю результаты работы.
5. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
6. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Веб-сервер не поддерживает постоянного соединения с клиентом, и каждый запрос обрабатывается, как новый, безо всякой связи с предыдущими. То есть, нельзя ни отследить запросы от одного и того же посетителя, ни сохранить для него переменные между просмотрами отдельных страниц. Для решения этих двух задач и были изобретены сессии.

Собственно, сессии, если в двух словах - это механизм, позволяющий однозначно идентифицировать браузер и создающий для этого браузера файл на сервере, в котором хранятся переменные сеанса.

Работа с сессиями. Создание простой авторизации на PHP

При разработке интернет магазинов практически всегда требуется **авторизация**. Например, для входа в личный кабинет пользователя, или систему управления интернет-магазином.

Форма авторизации обычно состоит из двух текстовых полей (Логин и пароль) и кнопки Войти. Сейчас при создании в интернет-магазинах личного кабинета, набирает моду установка полей Телефон и Имя, или Имя и email, и другие комбинации. Но советуем так ни когда не делать, т.к. Имя можно написать по-разному, телефон тоже (84952322323 или 232-23-23).

PHP сессии

В этом занятии вы познакомитесь с правильной **работой с сессиями на PHP**.

Сессии нужны для сохранения определенной информации на стороне браузера. Практически тоже самое, что и переменные, отличается только тем, что переменные при переходе на другую страницу сайта, или при обновлении текущей страницы теряют свои значения, а данные записанные в сессии сохраняются, и доступны на любых страницах сайта.

Например, на странице http://ox2.ru/index.php мы записали в сессию 'session_test' значение '123'. На странице http://ox2.ru/shop.php мы можем прочитать сессию session_test, и получить значение 123.

Для работы с сессиями на php нужно на каждой странице где будет производиться работа с сессиями написать session_start(), в самом начале, до вывода любой информации на экран.

Для записи в сессии существует переменная $_SESSION.

Например, простой пример:

Файл index.php будет **запись в сессию**:

```php
<?php
session_start();
$_SESSION["test"] = "Сессия - тест";
?>
```

Файл session.php будет **чтение сессии**:

```php
<?php
session_start();
echo $_SESSION['test']; //На экране будет 'Сессия - тест'
?>
```

Практическая часть

1. Создайте файл registration.php, который будет содержать форму регистрации пользователя.

```
<form method="POST" action="registration.php">
<p>Имя пользователя: <input type="text" name="username"></p>
<p>Email: <input type="text" name="email"></p>
<p>Пароль: <input type="password" name="password"></p>
<p>Пароль(повторить): <input type="password"
name="password2"></p>
<p><input type="submit"></p>
</form>
```

В самом начале файла необходимо вызвать функцию session_start(), она создает сессию или продолжает текущую на основе текущего идентификатора сессии, который передается через запросы, такие как GET, POST или cookie. В большинстве случаев используют сессии на cookie, поэтому перед функцией start_session() не должно быть функций, возвращающих сообщение в браузер. Затем делаем проверку, аутентифицирован ли пользователь.

```
if ($_SESSION['username']) {
echo 'Вы уже зарегистрированы';
return 0;
}
```

Далее проверяем, существует ли POST запрос, если да, то проверяем совпадают ли пароли и заносим в переменную сессии данные из POST запроса.

```
if ($_POST) {
if ($_POST['password'] == $_POST['password2'])
{ $_SESSION['username'] = $_POST['username'];
echo 'Пользователь '.$_SESSION['username'].' зарегистрирован'; return 0;
} else {
echo 'Введенные пароли не совпадают';
return 0;
}
}
```

в index.php заменяем все POST на SESSION.

Теперь необходимо создать файл logout.php. Этот файл будет содержать функцию, которая разрушит все данные, зарегистрированные в сессии – session_destroy().

```
<?php
session_start();
session_destroy();
header('Location: index.php');
?>
```

Перед тем как отправить форму, ее нужно проверить. Проверка формы будет осуществляться посредством javascript. Для этого у формы определим событие onsubmit="return checkForm(this)". Функция checkForm будет вызываться перед отправкой данных. Проверять будем имя пользователя, заполнено он или нет и email.

Шаблон email`a будет [английские_буквы]@ [английские_буквы]. [английские_буквы]. Такую проверку можно сделать, используя регулярные выражения - это формальный язык поиска и осуществления манипуляций с подстроками в тексте, основанный на использовании. По сути это строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Более подробно о регулярных выражениях можно почитать на Википедии (http://ru.wikipedia.org/wiki/Регулярные_выражения).

Вообще регулярное выражение для проверки электронной почты довольно громоздкое.

```
^[-a-z0-9!#$%&'*+/=?^_`{|}~]+(?:\.[-a-z0-9!#$%&'*+/=?^_`{|}~]+)*@(?:[a-
z0-9]([-a-z0-9]{0,61}[a-z0-
```

```
9])?\.)*(?:aero|arpa|asia|biz|cat|com|coop|edu|gov|info|int|jobs|mil|mobi
|museum|name|net|org|pro|tel|travel|[a-z][a-z])$
```

Но оно слишком тяжело для понимания, поэтому воспользуемся простой формой [a-zA-Z]*@[a-zA-Z]*\.[a-zA-Z].

В начале функции определим все переменные, которые нам понадобятся, и массивы с ошибками.

```
var el, // Сам элемент
elName, // Имя элемента формы
value, // Значение
type; // Атрибут type для input-ов
reg = /[a-zA-Z]*@[a-zA-Z]*\.[a-zA-Z]/;
var errorList = [];
var errorText = {
: "Не заполнено поле 'Имя'",
1 : "Не заполнено поле 'E-mail'",
}
```

Далее проходимо по всем элементам формы и проверяем все теги input. Если поля для имени пустое или электронная почта не соответствует шаблону, записываем номера ошибок в массив.

```
for (var i = 0; i < form.elements.length; i++) { el =
form.elements[i];
elName = el.nodeName.toLowerCase();
value = el.value;
if (elName == "input") {
type = el.type.toLowerCase();
switch (type) {
case "text" :
if                              (el.name    ==    "name"    &&    value    == "")
errorList.push(1);
if                             (el.name   ==   "email"   &&   !value.match(reg))
errorList.push(2);
break;
default :
break;
}
}
}
```

Затем проверяем массив с ошибками. Если он пуст, то возвращаем true, иначе формируем текст для ошибки, выводм это на экран и возвращаем false.

```
if (!errorList.length) return true;
var errorMsg = "При заполнении формы допущены следующие ошибки:\n\n"; for (i
= 0; i < errorList.length; i++) {
errorMsg += errorText[errorList[i]] + "\n";
}
alert(errorMsg);
return false;
```

2. Теперь в файле registration.php после строки `if '($_POST) {'` поставим проверку файла, если он существует, то открываем его и перемещаем указатель в конец строки, если нет, то создаем его.

```
$fp = fopen('users.txt', 'a+');
if (!$fp) {
$fp = fopen('users.txt', 'w+');

}
```

В условии проверки паролей сформируем строку с данными, которые будут разделены знаком &, запишем ее в файл и закроем его.

```
$mytext ='username='.$_POST['username'].'&password='.$_POST['password'].'&email
='.$_POST['email']. "\r\n";
$test = fwrite($fp, $mytext);
fclose($fp);
```

Следующим шагом будет преобразование файла login.php. Теперь мы будем получать данные о пользователях не в самом сценарии, а из отдельного файла. Для этого в самом начале файла снова начнем сессию, подключим файл debug.php для отладки и делаем проверку, если существует пост запрос, то открываем файл user.txt, построчно считываем его содержимое и сравниваем с данными POST запроса. В случае успеха, добавляем пользователя в сессию и переходим на index.php.

```
<?php
session_start();
require_once('debug.php');
if ($_POST) {
$fp = fopen('users.txt', 'r');
while (!feof ($fp)) {
$buffer = fgets($fp);
preg_match('/username=([^&]*)&/', $buffer, $user);
preg_match('/password=([^&]*)&/', $buffer, $pass);
         if ($_POST['username'] == $user[1] && $_POST['password'] == $pass[1])
{
$_SESSION['username'] = $_POST['username']; header('Location:
index.php');
}
}
fclose ($fp);
} else { ?>
<html>
<head>
<title>Login page</title>
</head>
<body><h3>Войти</h3>
<form method="POST" action="">
<p>Имя пользователя: <input type="text" name="username"></p>
<p>Пароль: <input type="password" name="password"></p>
      <p><input type="submit"></p>
</form>
<?php
}
?>
</body>
</html>
```

В данном примере авторизации основную функцию будет выполнять класс AuthClass. Приведен код с пояснениями.

Создайте файл **session.php:**

```php
<?php
session_start(); //Запускаем сессии

/**
 * Класс для авторизации
 * @author дизайн студия ox2.ru
 */
class AuthClass {
    private $_login = "demo"; //Устанавливаем логин
    private $_password = "www.ox2.ru"; //Устанавливаем пароль

    /**
     * Проверяет, авторизован пользователь или нет
     * Возвращает true если авторизован, иначе false
     * @return boolean
     */
    public function isAuth() {
        if (isset($_SESSION["is_auth"])) { //Если сессия существует
            return $_SESSION["is_auth"]; //Возвращаем значение переменной сессии is_auth (хра
        }
        else return false; //Пользователь не авторизован, т.к. переменная is_auth не создана
    }

    /**
     * Авторизация пользователя
     * @param string $login
     * @param string $passwors
     */
    public function auth($login, $passwors) {
        if ($login == $this->_login && $passwors == $this->_password) { //Если логин и пароль
            $_SESSION["is_auth"] = true; //Делаем пользователя авторизованным
            $_SESSION["login"] = $login; //Записываем в сессию логин пользователя
            return true;
        }
        else { //Логин и пароль не подошел
            $_SESSION["is_auth"] = false;
            return false;
        }
    }

    /**
     * Метод возвращает логин авторизованного пользователя
     */
    public function getLogin() {
        if ($this->isAuth()) { //Если пользователь авторизован
            return $_SESSION["login"]; //Возвращаем логин, который записан в сессию
        }
    }


    public function out() {
        $_SESSION = array(); //Очищаем сессию
        session_destroy(); //Уничтожаем
    }
}

$auth = new AuthClass();

if (isset($_POST["login"]) && isset($_POST["password"])) { //Если логин и пароль были отправлены
    if (!$auth->auth($_POST["login"], $_POST["password"])) { //Если логин и пароль введен не правильно
        echo "<h2 style="color:red;">Логин и пароль введен не правильно!</h2>";
    }
}

if (isset($_GET["is_exit"])) { //Если нажата кнопка выхода
    if ($_GET["is_exit"] == 1) {
        $auth->out(); //Выходим
        header("Location: ?is_exit=0"); //Редирект после выхода
    }
}

?>
```

```php
<?php
if ($auth->isAuth()) { // Если пользователь авторизован, приветствуем:
    echo "Здравствуйте, " . $auth->getLogin() ;
    echo "<br/><br/><a href='?is_exit=1'>Выйти</a>"; //Показываем кнопку выхода
}
else { //Если не авторизован, показываем форму ввода логина и пароля
?>
<form method="post" action="session.php">
    Логин: <input type="text" name="login"
    value="<?php echo (isset($_POST["login"])) ? $_POST["login"] : null;
    // Заполняем поле по умолчанию ?>" />
    <br/>
    Пароль: <input type="password" name="password" value="" /><br/>
    <input type="submit" value="Войти" />
</form>
<?php
}
```

Это самый простой пример **создания авторизации на php**. Пароли и имена пользователей должны браться из базы данных, шифроваться и т.д.

Содержание отчета

1. Цель
2. Программные листинги
3. Выводы

Контрольные вопросы

1. В чем отличие php-страницы от html-страницы?
2. Какие типы переменных поддерживает язык PHP?
3. Как передать переменную в php-страницу?
4. Какие параметры существуют у функции date()?
5. Для чего используется функция isset()?

Лабораторная работа № 19
Создание проекта «Регистрация»

Цель: сформировать навыки и умения создания проектов регистрации пользователей.

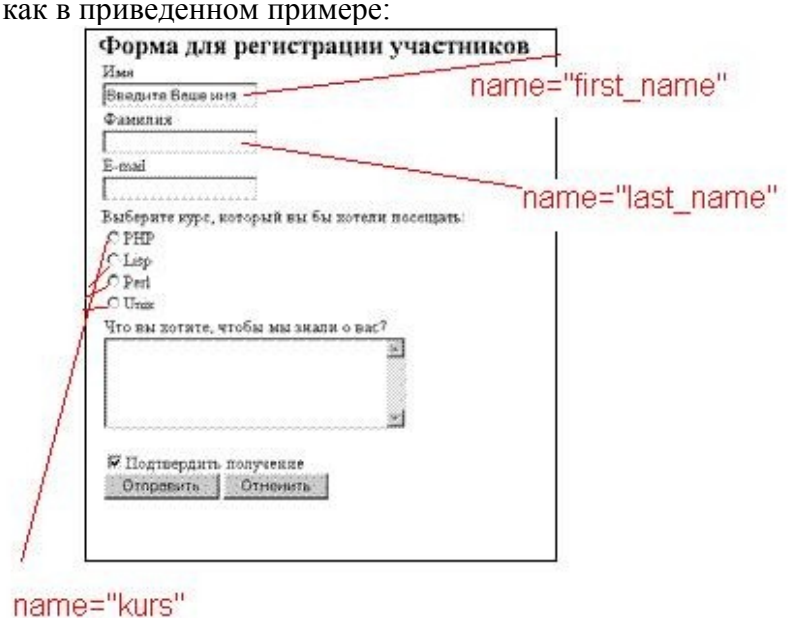
Ход работы:

1. Выполнить задания в соответствии с указаниями.
2. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
3. Подготовить ответы на контрольные вопросы (устно)

Указания по выполнению

Задание 1. Создать форму для регистрации пользователей на сайте. Реализовать отправку "универсального письма" всем зарегистрировавшимся.

1. Создайте форму для регистрации участников заочной школы программирования, как в приведенном примере:



В форме можно указывать метод передачи данных GET или POST.

2. Для того чтобы обработать созданную форму, необходимо в файле php написать:

```
<?php
$str = "Здравствуйте,
    ".$_REQUEST["first_name"]. "
    ".$_REQUEST["last_name"]."! <br>";
$str .="Вы выбрали для изучения курс по
    ".$_REQUEST["kurs"];
echo $str;
?>
```

Файл 1.php, обрабатывающий форму form.html

Разместите скрипт на сервере: *C:\AppServ\www*. Форма должна ссылаться именно на этот файл, поэтому в ее атрибуте action должно быть записано: **action="http://localhost/1.php"**).

Протестируйте работу скрипта. В результате, если в форму мы ввели имя «Вася», фамилию «Петров» и выбрали среди всех курсов курс по PHP, на экране браузера получим такое сообщение:

Здравствуйте, Вася Петров!

Вы выбрали для изучения курс по PHP

3. Измените начальный вариант формы регистрации таким образом, чтобы каждый регистрирующийся мог выбрать сколько угодно курсов для посещения.

Здесь все достаточно просто и понятно. Единственное, что можно отметить, – это способ передачи значений элемента checkbox. Когда мы пишем в имени элемента kurs[], это значит, что первый отмеченный элемент checkbox будет записан в первый элемент массива kurs, второй отмеченный checkbox – во второй элемент массива и т.д.

Скрипт, который все это будет разбирать и обрабатывать, называется 2.php (форма должна ссылаться именно на этот файл, что записано в ее атрибуте action: **action="http://localhost/2.php"**). По умолчанию используется для передачи метод GET, укажем POST. По полученным сведениям от зарегистрировавшегося человека, скрипт генерирует соответствующее сообщение. Если человек выбрал какие-то курсы, то ему выводится сообщение о времени их проведения и о лекторах, которые их читают. Если человек ничего не выбрал, то выводится сообщение о следующем собрании заочной школы программистов.

4. Создайте второй скрипт и разместите его на сервере: *C:\AppServ\www*

```
<?
// создадим массивы соответствий курс-время его
// проведения и курс-его лектор
$times = array("PHP"=>"14.30","Lisp"=>"12.00",
    "Perl"=>"15.00","Unix"=>"14.00");
$lectors = array("PHP"=>"Василий Васильевич",
"Lisp"=>"Иван Иванович", "Perl"=>"Петр Петрович", "Unix"=>"Семен
Семенович");
define("SIGN","С уважением, администрация");
    // определяем подпись письма как константу
define("MEETING_TIME","18.00");
    // задаем время собрания студентов
$date = "12 мая"; // задаем дату проведения лекций
    //начинаем составлять текст сообщения
$str = "Здравствуйте, уважаемый " . $_POST["first_name"]
    . " " . $_POST["last_name"]."!<br>";
$str .= "<br>Сообщаем Вам, что ";
$kurses = $_POST["kurs"];   // сохраним в этой переменной
                                    // список выбранных
курсов
if (!isset($kurses)) { // если не выбран ни один курс
    $event = "следующее собрание студентов";
    $str .= "$event состоится $date ". MEETING_TIME . "<br>";
} else { // если хотя бы один курс выбран
    $event = "выбранные Вами лекции состоятся $date <ul>";
        //функция count вычисляет число элементов в массиве
    for ($i=0;$i<count($kurses);$i++){
                            // для каждого выбранного курса
        $k = $kurses[$i]; // запоминаем название курса
        $lect = $lect . "<li>лекция по $k в $times[$k]";
                            // составляем сообщение
        $lect .= " (Ваш лектор, $lectors[$k])";
    }
    $event = $event . $lect . "</ul>";
    $str .= "$event";
}
$str .= "<br>". SIGN; // добавляем подпись
echo $str; // выводим сообщение на экран
?>
```

Протестируйте работу скрипта.

Задание 2. Реализовать систему регистрации на сайте.

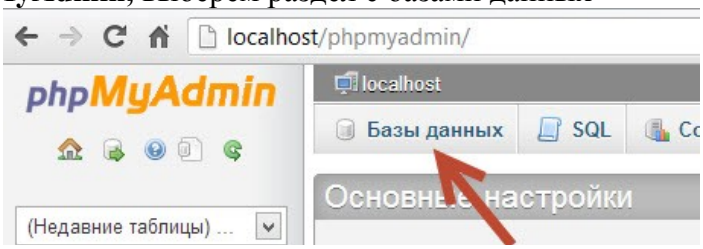
Разместите скрипты на сервере: *C:\AppServ\www*. Протестируйте работу.

Для реализации системы регистрации будем использовать язык серверного программирования **PHP** и систему управления базами данных **MySQL** для хранения информации о пользователях.

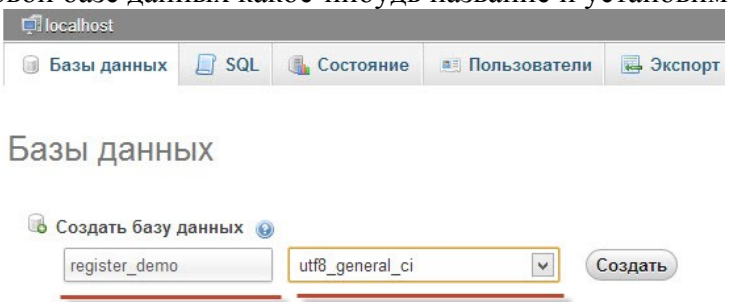
Что нам будет нужно:

- **Web-Server,** с установленным и настроенным PHP
- **MySQL**
- **phpMyAdmin** - удобное средство для работы с базой MySQL.
- **Notepad++**

1. Откроем **phpMyAdmin,** выберем раздел с базами данных



Дадим нашей новой базе данных какое-нибудь название и установим кодировку **UTF-**



Можно смело нажимать на кнопку **Создать**.

На виртуальных хостингах не разрешено создавать свою базу данных. Нужно использовать ту, которая включена в ваш тарифный план.

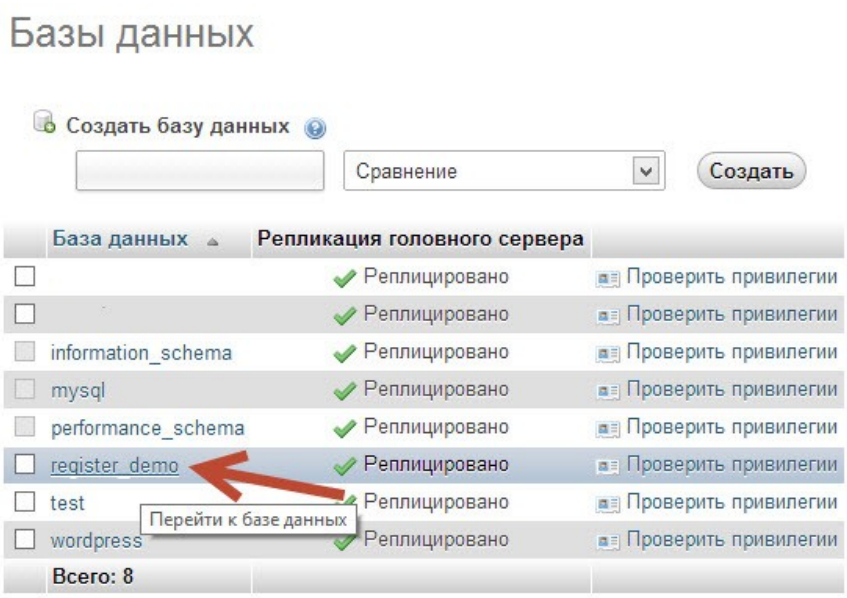
2. Следующим шагом идёт создание таблицы в базе данных, которая будет содержать информацию о пользователях. Будем использовать простую таблицу с несколькими полями, но никто не мешает вам добавить в неё новые поля для своих нужд. Итак, какие поля в таблице нам потребуются:

   - **UserID (Первичный ключ)** - Уникальный идентификатор пользователя
   - **Username** – имя пользователя
   - **Password** – пароль пользователя
   - **EmailAddress** – почта пользователя

Как видите, всего четыре поля, которые будут характеризовать наших зарегистрированных пользователей

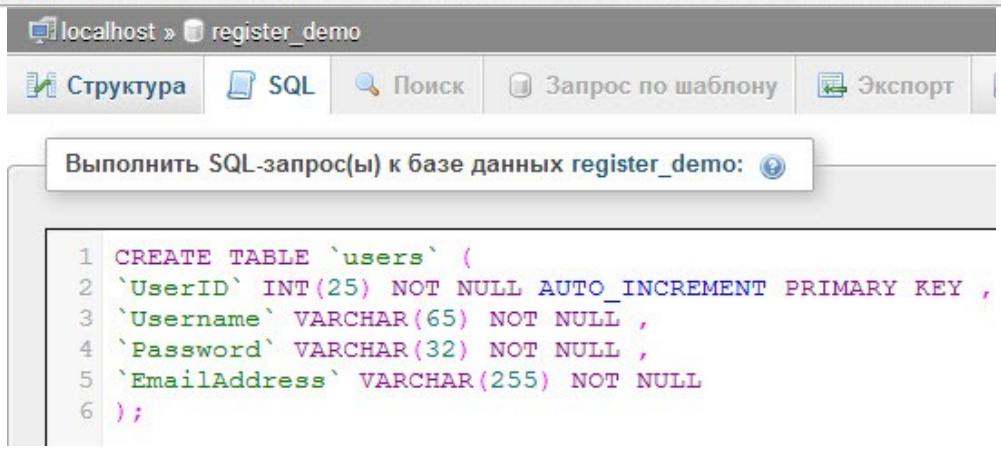
Первичный ключ (англ. primary key) – если говорить простым языком, то в базах данных это поле, которое уникальное для каждой записи и по-которому можно эту запись идентифицировать. В нашем случае это – **UserID**. С каждым новым пользователем, ему должен присваиваться свой номер, поэтому при создании этого поля мы будем использовать специальную команду MySQL –**AUTO_INCREMENT**. Она автоматически будет увеличивать на единицу значение поля **UserID** при каждой новой записи.

Выбираем нашу базу данных



и переходим на вкладку **SQL**. Для создания таблицы в базе данных мы будем использовать следующий SQL-запрос:

```
CREATE TABLE `users` (
`UserID` INT(25) NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`Username` VARCHAR(65) NOT NULL ,
`Password` VARCHAR(32) NOT NULL ,
`EmailAddress` VARCHAR(255) NOT NULL
);
```



3. С базой пока всё. Теперь откроем текстовый редактор. Первым файлом нашего проекта, который мы создадим, будет файл с конфигурационной информацией, который будет подключается к каждому другому файлу.

Такая техника экономит место и время, если конфигурация изменяется. Нам потребуется поменять её только в одном месте, а не перелопачивать все файлы проекта.

Назовём его **base.php**. Он будет содержать следующий код**:**

```
<?php
session_start();

$dbhost = "localhost"; // Адрес сервера MySQL. На локальном сервере этот параметр все
$dbname = "database"; // Имя базы данных
$dbuser = "username"; // Пользователь базы данных
$dbpass = "password"; // Пароль пользователя базы данных

mysql_connect($dbhost, $dbuser, $dbpass) or die("Ошибка MySQL: " . mysql_error());
mysql_select_db($dbname) or die("Ошибка MySQL: " . mysql_error());
?>
```

Здесь

- **session_start()** – функция, которая начинает новую сессию для каждого пользователя. Чуть позже мы будем хранить в ней информацию о пользователях, чтобы определять, кто уже авторизован.
- **mysql_connect** – функция, обеспечивающая подключение нашего скрипта к базе данных, используя информацию, которую мы вынесли в отдельные переменные выше.
- **mysql_select_db** – после успешного подключения нужно выбрать базу данных. Эта функция как раз это и производит.
- Если же хотя бы одна из функций не смогла отработать по какой-либо причине, то работа нашего скрипта прекращается и выводится ошибка. За это ответственна функция **die**. А за вывод самой ошибки отвечает функция **mysql_error()**.

4. Файл с настройками создан (вы же не забыли изменить значения переменных на свои?) и теперь подошла очередь создать главный файл – **index.php.** Самой первой строчкой в нём должно быть подключение нашего конфигурационного файла.

```php
<?php include "base.php"; ?>
```

Далее нам нужно создать базовую разметку **HTML**. У нас будет один главный контейнер, в который сложим заголовок и все поля для ввода данных. Ах ну да, ещё и кнопку. Пока все элементы будут некрасивыми, но это ОК. Позже преукрасим их с помощью **CSS**.

```html
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <title>Регистрация пользователей на PHP</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="main">

    </div>
</body>
</html>
```

5. Прежде чем выводить содержимое страницы, у нас есть пара вопросов, на которые нужно ответить:
    - Пользователь уже зарегистрирован?
- **ДА** – показываем ему страницу с опциями для зарегистрированных пользователей
- **НЕТ** – переходим к следующему вопросу
    - Пользователь зарегистрировался и отправил свои данные?
- **ДА** – Проверяем данные, регистрируем пользователя и производим авторизацию
- **НЕТ** – переходим к следующему вопросу
    - Если на оба вопроса ответ **НЕТ**, то выводим форму для авторизации.

Теперь давайте реализуем эти условия на языке **PHP**. Внутри нашего главного блока в **HTML**, напишем несколько конструкций **If**

```php
<?php
if(!empty($_SESSION['LoggedIn']) && !empty($_SESSION['Username']))
{
    // даём доступ пользователю к главной странице
}
elseif(!empty($_POST['username']) && !empty($_POST['password']))
{
    // позволим пользователю войти на сайт
}
else
{
    // выводим форму для авторизации
}
<?>
```

Когда пользователь входит на сайт, мы храним информацию о нём в его сессии. Чтобы получить к ней доступ, нужно обратиться к глобальному массиву PHP - **$_SESSION**. Также мы будем использовать функцию **empty()** для проверки, пусто ли в переменной или нет. Оператор **!** перед функцией говорит об её отрицании, т.е:

Если переменная $_SESSION['LoggedIn'] не пустая и переменная $_SESSION['Username'] тоже не пустая, то выполнить часть кода ниже.

Следующие строки кода имеют схожую механику, но мы используем уже глобальный массив**$_POST**. В нём хранится любая информация, которая была отправлена с формы при входе на сайт или регистрации. Эту форму мы создадим чуть позже.

Последняя строка выполниться, только когда все верхние проверки не сработали. Мы просто показываем пользователю форму для авторизации на сайте.

%2. Теперь, когда мы понимаем логику работы, давайте добавим в наши условия код для выполнения, если они соблюдены.

```
    <div id="main">
    <?php
    if(!empty($_SESSION['LoggedIn']) && !empty($_SESSION['Username']))
{
    // даём доступ пользователю к главной странице
?>
     <h1>Закрытый раздел!</h1>
     <p>Привет, <b><?=$_SESSION['Username']?></b>. Твоя почта - <b><?=$_SESSION['EmailAddress']?></b>.</p>
     <?php
}
elseif(!empty($_POST['username']) && !empty($_POST['password']))
{
    // позволим пользователю войти на сайт
    $username = mysql_real_escape_string($_POST['username']);
    $password = md5(mysql_real_escape_string($_POST['password']));

    $checklogin = mysql_query("SELECT * FROM users WHERE Username = '".$username."' AND Password = '".$password."'");

    if(mysql_num_rows($checklogin) == 1)
    {
        $row = mysql_fetch_array($checklogin);
        $email = $row['EmailAddress'];

        $_SESSION['Username'] = $username;
        $_SESSION['EmailAddress'] = $email;
        $_SESSION['LoggedIn'] = 1;

        echo "<h1>Успех!</h1>";
        echo "<p>Сейчас вы будете перенаправлены в закрытый раздел.</p>";
        echo "<meta http-equiv='refresh' content='2;index.php'>";
    }
    else
    {
        echo "<h1>Ошибка</h1>";
        echo "<p>Прости, но мы не нашли такого аккаунта. Можешь <a href=\"index.php\">попробовать ещё раз</a>.</p>";
    }
}
else
{
    // выводим форму для авторизации
    ?>
   <h1>Авторизация</h1>

   <p>Спасибо за то, что пришли! Войдите или <a href="register.php">зарегистрируйтесь</a>.</p>

    <form method="post" action="index.php" name="loginform" id="loginform">
    <fieldset>
        <label for="username">Логин:</label><input type="text" name="username" id="username"><br>
        <label for="password">Пароль:</label><input type="password" name="password" id="password"><br>
        <input type="submit" name="login" id="login" value="Войти">
    </fieldset>
    </form>

   <?php
} ?>
    </div>
```

Давайте по-порядку:

mysql_real_escape_string – очень полезная функция для очищения входных данных. Она экранирует все специальные символы, введённые в форму, тем самым предотвращая их попадание в базу данных.

md5 – скажу кратко, функция зашифровывает всё, что ей пришло. В нашем случае – пароль пользователя.

Далее.

```
$checklogin = mysql_query("SELECT * FROM users WHERE Username = '".$username."' AND Password = '".$password."'");

   if(mysql_num_rows($checklogin) == 1)
   {
      $row = mysql_fetch_array($checklogin);
      $email = $row['EmailAddress'];

      $_SESSION['Username'] = $username;
      $_SESSION['EmailAddress'] = $email;
      $_SESSION['LoggedIn'] = 1;
```

Это сердце нашего кода. Для начала мы запускаем проверку введённых пользователем данных в нашей БД. Пытаемся извлечь из таблицы **users** всё, что соответсвтует имени пользователя **$username** и паролю **$password**. И если нам возвращается положительный результат то мы уже знаем, что такой пользователь существует и можно его авторизовать. Далее нам нужно получить массив данных из запроса в БД. Это осуществляется с помощью функции **mysql_fetch_array.** Ну и далее используется переменная для хранения email'а пользователя, чтобы потом её использовать.

Далее устанавливаем сессию и передаём в неё логин пользователя и его почту.

После нужно сообщить пользователю об успешности авторизации и обновить страницу. Это достигается с помощью мета-тега HTML с атрибутом refresh

```
echo "<meta http-equiv='refresh' content='2;index.php'>";
```

%2. Давайте взглянем на нашу страницу в браузере:

%2. Авторизация работает, но как это проверить? Нужно создать страницу, которая будет регистрировать пользователей. Открываем текстовый редактор и создаём новый файл – **register.php**.

%2. Как вы уже догадались, первой его строчкой будет подключение нашего файла конфигурации.

```
<?php include "base.php"; ?>
```

%2. Затем скопируем всё содержимое файла **index.php** в файл **register.php** и удалим всё, что находится в нашем главном блоке с **id="main"**.

%2. В этом файле не будет ничего принципиального нового, почти тот же код PHP.

```php
<?php include "base.php"; ?>
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="utf-8">
    <title>Регистрация пользователей на PHP</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="main">
    <?php
    if(!empty($_POST['username']) && !empty($_POST['password']))
    {
    // позволим пользователю зарегистрироваться
    $username = mysql_real_escape_string($_POST['username']);
    $password = md5(mysql_real_escape_string($_POST['password']));
    $email = mysql_real_escape_string($_POST['email']);

     $checkusername = mysql_query("SELECT * FROM users WHERE Username = '".$username."'");

     if(mysql_num_rows($checkusername) == 1)
     {
        echo "<h1>Ошибка</h1>";
        echo "<p>Извините, такое имя пользователя уже используется. Вернитесь назад и попробуйте снова.</p>";
     }
   else
   {
     $registerquery = mysql_query("INSERT INTO users (Username, Password, EmailAddress) VALUES('".$username."', '".$password."', '".$email."')");
     if($registerquery)
     {
        echo "<h1>Успех!</h1>";
        echo "<p>Ваша учётная запись создана. <a href=\"index.php\">Авторизуйтесь</a>.</p>";
     }
     else
     {
          echo "<h1>Ошибка</h1>";
          echo "<p>Мы не смогли вас зарегистрировать. Вернитесь назад и попробуйте снова.</p>";
       }
     }
}
else
{
    ?>

   <h1>Регистрация</h1>

   <p>Пожалуйста заполните несколько полей ниже.</p>

    <form method="post" action="register.php" name="registerform" id="registerform">
    <fieldset>
        <label for="username">Логин:</label><input type="text" name="username" id="username"><br>
        <label for="password">Пароль:</label><input type="password" name="password" id="password"><br>
        <label for="email">Email:</label><input type="text" name="email" id="email"><br>
        <input type="submit" name="register" id="register" value="Зарегистрироваться">
    </fieldset>
    </form>

   <?php
} ?>
    </div>
</body>
</html>
```

Обращу внимание на строчку с SQL-запросом:

```php
$registerquery = mysql_query("INSERT INTO users (Username, Password, EmailAddress) VALUES('".$username."', '".$password."', '".$email."')");
```

Именно она добавляет введённую пользователем информацию в нашу таблицу в базе данных. Сначала идут поля в которые необходимо записать данные, а затем, с помощью **VALUES** мы записываем значения переменных. Важно понимать, в каком

порядке вы запишите первые поля, в которые нужно записать данные, в таком же логическом порядке должны быть перечислены значения переменных, откуда брать данные.

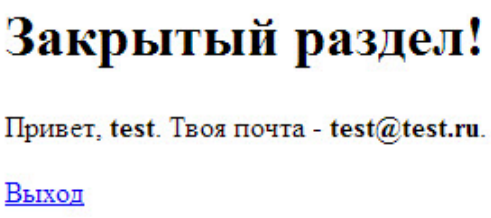
%2. Давайте попробуем. Зарегистрируем пользователя с логином test, а затем авторизуемся под ним.

%2. Есть один момент. Авторизовались мы успешно, но теперь я хочу выйти из своего аккаунта…А нельзя =) Откроем текстовый редактор и создадим новый файл под именем **logout.php**. В этом файле будет уничтожаться наша сессия и обновляться страница.

```php
<?php
include "base.php";
$_SESSION = array();
session_destroy();
?>
<meta http-equiv="refresh" content="0;index.php">
```

Осталось только добавить в файл index.php ссылку на этот файл.

```php
<h1>Закрытый раздел!</h1>
<p>Привет, <b><?=$_SESSION['Username']?></b>. Твоя почта - <b><?=$_SESSION['EmailAddress']?></b>.</p>
<p><a href="logout.php">Выход</a></p>
```

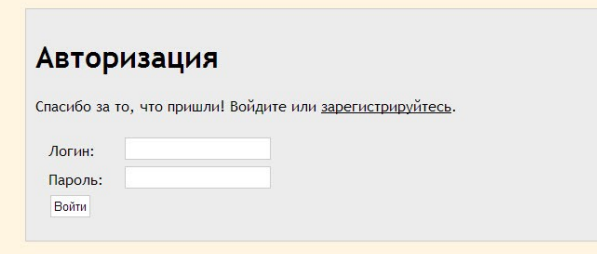


Вот теперь всё

%2. Мы почти закончили и теперь нам нужно придать красивый внешний вид нашей форме. Открываем текстовый редактор, создаём файл **style.css** и в нём напишем несколько простых стилей для используемых нами элементов:

```css
body {
    margin: 0;
    padding: 0;
    font-family: Trebuchet MS;
    background: #fff5e1;
}
a {
    color: #000;
}
a:hover, a:visited, a:active {
    text-decoration: none;
}
#main {
    width: 780px;
    margin: 250px auto 0 auto;
    padding: 10px;
    border: 1px solid #ccc;
    background: #ecece;
}
form fieldset {
    border: none;
}
form fieldset br {
    clear: left;
}
label {
    margin-top: 5px;
    display: block;
    width: 80px;
    padding: 0;
    float: left;
}
input {
    border: 1px solid #ccc;
    margin-bottom: 5px;
    padding: 3px;
    background: #fff;
}
input:hover {
    border: 1px solid #222;
    background: #eee;
}
```

Очень простая и лёгкая система. При желании можно нарастить ей сколько угодно функционала.



Содержание отчета

1. Цель
2. Основы PHP: переменные, константы, синтаксис.
3. Ход работы с выдержками из сценариев.
4. Выводы

Контрольные вопросы

1. Что такое сессии в PHP? Массив $_SESSION.
2. Особенности метода GET.
3. Особенности метода POST.
4. Какие управляющие последовательности вы использовали?

Лабораторная работа № 20
Создание проекта «Форум»

Цель: сформировать навыки и умения создания проектов регистрации пользователей.

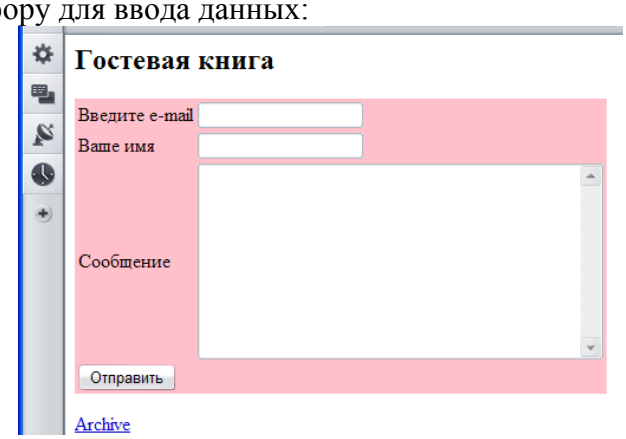
Ход работы:

1. Выполнить задания в соответствии с указаниями.
2. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
3. Подготовить ответы на контрольные вопросы (устно)

Указания по выполнению

Для создания гостевой книги понадобятся два файла. В первом (guest.php) необходимо разместить форму для ввода данных и сам скрипт гостевой книги, а во втором будут храниться результаты введенных данных в специальном формате. После закачки на сервер этих двух файлов на файл с результатами (guest.txt) нужно будет установить атрибуты, разрешающие запись в файл.

1. Создайте фору для ввода данных:



В этой форме три поля - адрес электронной почты (переменная email), имя посетителя (переменная name) и сообщение (переменная msg).

После того как посетитель введет данные и нажмет кнопку «Отправить», все эти переменные будут доступны скрипту, причем значения переменных будут соответствовать введенным данным.

Обработка данных

2. Определите имя файла, в котором будут записываться данные, и максимальное количество сообщений, которое может быть выведено на экран:

```
<?
$files = "guest.txt";
$qq = 50;
```

Дальше:

```
if (!$email) {$email = "нет";}
$msg = substr($msg,0,999);
$email = substr($email,0,39);
$name = substr($name,0,39);
```

Здесь выведенные данные обработаны таким образом, чтобы переменная адреса не была пустой. При этом каждая переменная обрезается, чтобы ограничить количество вводимых символов.

3. Проверьте, не являются ли сообщения или имя пустыми строками:

```
if ($msg != " " && $name != " ") {
```

4. Если сообщение или имя не указаны вообще, скрипт ничего никуда не запишет, а просто продолжит обработку дальше и выведет сообщения гостевой книги на экран. Но

если и имя, и сообщение введены, скрипт, прежде чем вывести данные на экран, должен сделать запись отформатированных данных в файл для сообщений:

```
$time = Date("h:i:M:d");
$soo = "\n<b>$time $name (<a href=\"mailto: $email\">
$email</a>) </b><br>$msg<hr>";
$fp = fopen($files, "a+");
$fw = fwrite($fp, $soo);
fclose($fp); }
```

Этот код сначала определяет и форматирует время ввода сообщения. Затем формирует строку для записи в файл.

5. Дальше вывод результатов записи

```
$lines = file($files);
$a = count($lines);
$u = $a - $qq;
for($i = $a; $i >= $u; $i--) {echo $lines[$i]; } /* вывод результатов на экран
?>
```

Этим кодом в массив считывается файл сообщений, и при помощи цикла выводится на экран его содержимое. Если количество сообщений превысило ограничение, то они просто не показываются. Новые сообщения всегда отображаются вверху, около формы для ввода, так как вывод идет снизу вверх по индексу массива.

6. Для доступа к архиву сообщений поставьте ссылку:

```
<a href="guest.txt">Archive</a>
```

Обратите внимание, что код не учитывает ввод посетителем HTML-тегов.

Данный скрипт может использоваться также в любом месте, где нужно узнать мнение посетителей. Также это – простейший форум.

Или

Первоначально нужно определить функционал гостевой книги, т.е. как она будет работать. Сообщения могут оставлять только зарегистрированные пользователи, а пользователи – гости будут видеть только сами комментарии и надпись, что нужно зарегистрироваться. Гостевая книга будет находиться в файле index.php. Все данные будут записываться в файл guestbook.txt.

Для создание гостевой книги необходима форма, которая будет добавлять сообщения, поэтому разместим ее в файле после поля для логина.

```
<form method="POST">
<p> Тема поста: <input type="text" name="theme"></p>
<p>Текст поста:<textarea rows="10" cols="45" name="text"></textarea></p>
<p><input type="submit"></p>
</form>
```

Затем запрос этой формы необходимо обработать. Поместим в начало файла после запуска сессии проверку, если существует POST запрос с параметрами theme, text и пользователь находится в сессии, то открываем или создаем файл guestbook.txt, записываем туда данные, закрываем файл и создаем переменную с сообщением "Комментарий был успешно добавлен".

```
if ($_POST['theme'] && $_POST['text'] && $_SESSION['username'])
      { $fp = fopen('guestbook.txt', 'a+');
      if (!$fp) {
            $fp = fopen('guestbook.txt', 'w+');}
      $mytext      =      'username='.$_SESSION['username'].'&text='.
$_POST['text'].'&theme='.$_POST[' theme']."&"."\r\n";
      $test = fwrite($fp,
      stripslashes($mytext)); fclose($fp);
      $msg = "Комментарий был успешно добавлен";}
```

Перед формой разместим условие, если `$msg` существует, вывести его на экран, и если пользователь не в сессии то вместо формы выводим «Чтобы оставлять комментарии вы должны быть зарегистрированы».

```
<?php
      if ($msg) {
      ?><p><?php echo $msg; ?></p><?php
      }
if ($_SESSION['username']) {
?>
<form method="POST">
<p> Тема поста: <input type="text" name="theme"></p>
<p>Текст поста:<textarea rows="10" cols="45" name="text"></textarea></p>
<p><input type="submit"></p>
</form>
<?php
} else {
?><p>Чтобы оставлять комментарии вы должны быть зарегистрированы</p>
<?php
}
?>
```

Далее следуют сами комментарии. Выводить их будем в таблице вида

имя пользователя	заголовок

текст сообщения

Для этого открываем файл guestbook.txt с параметром “r”, затем при помощи функций feof() и fgets() считываем построчно файл. foef() – проверяет, достигнут ли конец файла, fgets() – возвращает строку из файла. Затем при помощи регулярных выражений вытаскиваем имя пользователя, заголовок, текст сообщения и заносим данные в таблицу.

```
<table class="table">
<?php
      $fp = fopen('guestbook.txt', 'r');
      while (!feof ($fp)) {
      ?>
      <tr class="main">
      <?php
            $buffer = fgets($fp);
            preg_match('/username=([^&]*)&/', $buffer, $user);
            preg_match('/text=([^&]*)&/', $buffer, $text);
            preg_match('/theme=([^&]*)&/', $buffer, $theme);
      ?>
            <td>
                  <?php echo $user[1];?>
            </td>
            <td>
                  <?php echo $theme[1];?>
            </td>
      </tr>
      <tr class="text">
            <td colspan=2>
                  <?php echo $text[1]; ?>
            </td>
      </tr>
      <?php
      }
      fclose ($fp);
?>
</table>
```

Гостевая книга готова, но чтобы было удобнее работать, ее нужно оформить при помощи css. Объединим поля для регистрации и входа в один div с id=reg.

```
<div class="reg">
```

```
<a href="registration.php">Регистрация</0a>
<a href="login.php">Войти</a>
<p>Привет,
<?php
      if ($_SESSION['username']) {
            echo $_SESSION['username'];
?>
            <a href="logout.php">Выйти</a>
<?php
      } else {
            echo 'Гость';
      }
?>
</p>
</div>
```

Создадим по подключим файл guestbook.css.

Пример файла guestbook.css:

```
table {
      width: 300px;
}
.main{
      font-size: 120%;
      font-family: Verdana, Arial, Helvetica, sans-serif;
      color: #336;
      border: 10px solid #666;
}
.text{
    color: red;
    border: 1px solid #666;
    background: #eee;
    padding: 5px;
}
.reg{
    position: absolute;
    right: 10px;
    top: 10px;
    width: 225px;
    height: 180px;
    background: #f0f0f0;
}
```

Содержание отчета

1. Цель
2. Основы PHP: переменные, константы, синтаксис.
3. Ход работы с выдержками из сценариев.
4. Выводы

Контрольные вопросы

1. Что такое сессии в PHP? Массив $_SESSION.
2. Особенности метода GET.
3. Особенности метода POST.
4. Какие управляющие последовательности вы использовали?

Лабораторная работа № 21
Создание проекта «Чат»

Цель: сформировать навыки и умения создания проектов с использованием базы данных.

Ход работы:

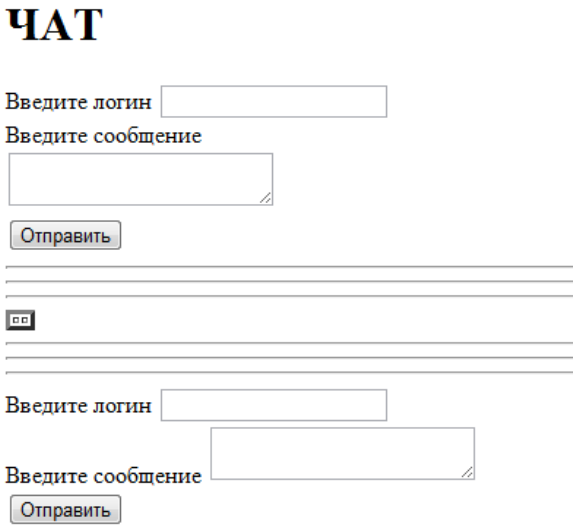
1. Выполнить задания в соответствии с указаниями.
2. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
3. Подготовить ответы на контрольные вопросы (устно)

Указания по выполнению

Для создания чата понадобится один файл *.php. Кроме того, вам дана база *chat*, папку с которой необходимо разместить по адресу C:\AppServ\MySQL\data.

Путь для размещения файлов вашего чата: *C:\AppServ\www\chat*

1. В файле **.php* создайте форму для ввода данных:



Создаем форму для имитации работы двух пользователей.

Для первого пользователя:

Первая строка

```
<form action="имя.php" method="post">
```

В этой форме три поля – логин (имя *login,* текстовое поле), поле для сообщения (имя *pass,* textarea), кнопка Отправить <*input type=submit*>.

После того как посетитель введет данные и нажмет кнопку «Отправить», все эти переменные будут доступны скрипту, причем значения переменных будут соответствовать введенным данным.

Обработка данных

2. Импортируем скрипт в html-документ для подключения к базе данных:

```
<?php
$db=mysql_connect("localhost","root");
mysql_select_db("chat");
$name=$_POST['login'];
$message=$_POST['pass'];
$mysql_query = mysql_query("INSERT INTO bd(name,msg)VALUES ('$name','$message')");
$data=mysql_query("SELECT * FROM bd");
?>
```

Если для доступа к базе данных нужен пароль, то есть код выдает ошибку подключения к базе данных, изменить строку *$db=mysql_connect("localhost", "root", "12345")*

3. Для вывода информации из базы данных добавляем код

```
<div id="first">
<table border="3">

<?php while($datas=mysql_fetch_array($data)){?>
<tr>
        <td><?=$datas['name']?></td><td><?=$datas['msg']?></td></tr>
<?php }?>

</table>
    </div>
<?php mysql_close($db)
    ?>
```

4. Для разделения полей пользователей добавьте горизонтальные полосы с помощью тега <hr>

5. Создадим форму для второго пользователя. В этой форме три поля – логин (имя *login1*, текстовое поле), поле для сообщения (имя *pass1*, textarea), кнопка Отправить <*input type=submit*>. Для кнопки код `<input type=submit onclick="redirect()">`

6. Для вывода информации добавляем код

```
<?php
$db=mysql_connect("localhost","root");
mysql_select_db("chat");
$name=$_POST['login1'];
$message=$_POST['pass1'];
$mysql_query = mysql_query("INSERT INTO bd(name,msg)VALUES ('$name','$message')");
?>
```

Запустите проект, проверьте его работоспособность

Содержание отчета

1. Цель
2. Ход работы с выдержками из сценариев.
3. Выводы

Контрольные вопросы

1. Массив $_REQUEST.
2. Особенности метода GET.
3. Особенности метода POST.
4. Переменные окружения.
5. Опишите алгоритм создания гостевой книги (форума).
6. Какие функции вы использовали при создании сценария?
7. Какие операторы используются для работы с файлами?
8. Какие управляющие последовательности вы использовали?

Лабораторная работа № 22
Создание новостной ленты

Цель: сформировать навыки и умения создания проектов с использованием базы данных.

Ход работы:

1. Выполнить задания в соответствии с указаниями.
2. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
3. Подготовить ответы на контрольные вопросы (устно)

Теоретическая часть

Файл RSS (он же «*RSS-канал*», «*RSS-поток*», «*RSS-лента*», «*Feed*», «*фид*») — это файл с перечнем публикаций сайта (или их анонсов), отсортированных по времени создания/обновления. Он не должен включать ссылки на страницы со служебной информацией или предназначенные для навигации, например, корзину интернет-магазина или страницы рубрик.

RSS используют читатели, узнавая о новом материале сайта в RSS-ридерах, и агрегаторы (например, Яндекс.Новости), чтобы постоянно не обходить весь сайт.

Символ RSS

```
<svg viewBox="0 0 10 10" xmlns="http://www.w3.org/2000/svg" height="80" width="80" style="overflow: hidden;"><circle r="1.5" cx="1.5" cy="8.5" fill="#FF993C"></circle><g fill="none" stroke="#FF993C" stroke-linecap="round" stroke-width="2"><circle r="5.8" cx="0" cy="10"></circle><circle r="9" cx="0" cy="10"></circle></g></svg>
```

Указывая ширину (width) и/или высоту (height) иконки, изображение будет пропорционально изменяться и подстраиваться под любой размер.

Пример RSS

http://shpargalkablog.ru/rss.php

Валидатор RSS

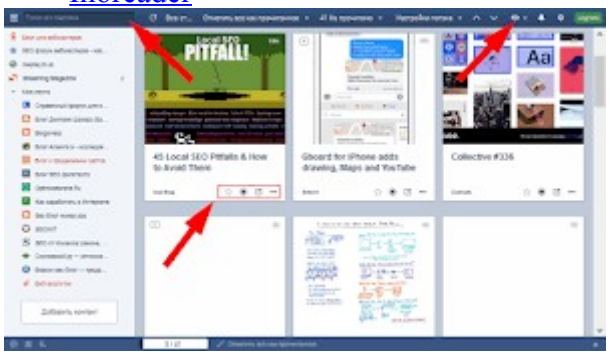
https://validator.w3.org/feed/

Что такое RSS-ридер

RSS-ридер (он же «*RSS-агрегатор*») — это программа, которая объединяет RSS-файлы нескольких сайтов в одну ленту. Ссылки на новые материалы всех интересующих сайтов собраны в одном месте. Не требуется переходить на каждый сайт в отдельности только для того, чтобы узнать есть ли новые публикации. Просмотренные ссылки помечаются как прочитанные. К непрочитанным можно вернуться в любое удобное время.

Пример RSS-ридера

Inoreader



Термин **RSS** имеет много разных определений - он может быть переведен как *Действительно Простая Синдикация,* также он может быть переведен и по-другому, однако суть данной технологии от этого не изменится.

RSS – это передача и обновление новостей в автоматическом режиме.

Подавляющая часть новостных сайтов, блогов и т.д., публикует анонсы статей, событий, новостей в **формате RSS.** Он обеспечивает регулярное автоматическое обновление, так что Ваши посетители увидят самые последние и актуальные новости.

Формат RSS не предназначен для чтения человеком. Он представляет собой **XML документ**, специально разработанный для чтения машинами. Я уже писал про то, Как сделать RSS на сайте. А в этой статье я покажу, как **читать RSS-ленту через PHP.**

Но для начала, как вообще читается **RSS-лента. Для чтения RSS** используются специальные программы, называемые **агрегаторами**. Многие из них похожи на почтовые программы, но вместо входящих писем они **отображают новости из различных** источников (со всех новостных лент в которых вы зарегистрированы или на которые вы подписаны). Причем, как и в почте, непрочитанные новости отображаются жирным шрифтом.

RSS агрегаторы сильно облегчают слежение за новостями из огромного количества источников, доставляя все новости в одно место. Но принимая во внимание тот факт, что сегодня все больше распространяются смартфоны, то существуют специализированные сайты –**Web-RSS агрегаторы. С их помощью можно следить за новостями** с любого устройства, на котором есть браузер.

Все современные браузеры также имеют встроенную возможность чтения **RSS-ленты,**однако она ограниченна.

В конечном счете, некоторые сайты собирают, агрегируют новости из различных источников на один сайт. Таким образом, осуществляется **“синдикация”.**

После того как **RSS сервис** создан пора его размещать на хостинге. Фактически **RSS-лента**– это ссылка подобного вида – http:/mysite.ru/rss.php, которая возвращает контент в формате **XML**.

Сегодня используются различные **версии RSS.** Так, например, **RSS 2.0** – это наиболее общеупотребительный формат. Он используется для новостных сайтов и блогов, а также для размещения подкастов.

Также существует новый формат, называемый **Atom,** который предлагает более стандартизированный подход в обновлении **XML контента.** Однако он крайне мало распространен за пределами блог-сообщества. Практически все движки блогов могут генерировать **Atom-ленту** на лету.

Указания по выполнению

Как создать rss.php на PHP и MySQL

Для базы данных **table**, содержащей такие столбцы:

```
+---------------+---------------------------+-----------------------------------+----------------------------+----------------------------
+-----------------------------------------------+
| id            | url                       | title                             | update                     | meta                       | description                                |
| smallint(5)   | varchar(255)              | varchar(255)                      | timestamp                  | set('noindex', 'feed', '') | varchar(255)                               |
| UNSIGNED      |                           |                                   | CURRENT_TIMESTAMP          |                            |                                            |
| AUTO_INCREMENT |                          |                                   | ON UPDATE CURRENT_TIMESTAMP |                           |                                            |
+---------------+---------------------------+-----------------------------------+----------------------------+----------------------------
+-----------------------------------------------+
| 1             | 2017/02/rss-php-mysql.html | Как сделать RSS канал | PHP примеры | 2017-02-14 09:07:30        | feed                       | Что такое RSS-канал, зачем
он нужен, пример файла, |
|               |                           |                                   |                            |                            | как сделать его на PHP и MySQL и если требуется    |
|               |                           |                                   |                            |                            | с помощью .htaccess перенаправить в Feedburner     |
+---------------+---------------------------+-----------------------------------+----------------------------+----------------------------
+-----------------------------------------------+
| 2             | 2017/02/example.html      | Название статьи                   | 2017-01-26 12:00:00        |                            | Описание статьи                            |
+---------------+---------------------------+-----------------------------------+----------------------------+----------------------------
+-----------------------------------------------+
```

Файл db.php

```
<?php
if (defined('dbOn')) {
   $mysqli = new mysqli('localhost', 'my_user', 'my_password', 'my_db'); // подключение к
серверу MySQL: тут указывается пароль к базе данных
   if ($mysqli->connect_error) {
      die('Connect Error (' . $mysqli->connect_errno . ') ' . $mysqli->connect_error);
```

```
    }
} else {
    exit();
}
```

Файл rss.php

```
<?php
if ($_SERVER['QUERY_STRING'] == '') { // убирать невостребованные страницы с параметрами, например, site.ru/rss.php?p=0, site.ru/rss.php?p=1 и т.д.
    // подключить файл с паролем от базы данных
    define('dbOn', '');
    require_once 'абсолютный_адрес/db.php';
    if (!$mysqli->set_charset("utf8")) {
        printf("Ошибка при загрузке набора символов utf8: %s\n", $mysqli->error);
        exit();
    } else {
        if ($result = $mysqli->query("SELECT url, title, update, description FROM table WHERE meta LIKE '%feed%' ORDER BY update DESC LIMIT 10;")) {  // ORDER BY осуществляет сортировку строк по дате обновления, условие WHERE meta LIKE '%feed%' — выбор только тех, в ячейках которых на пересечении со столбцом meta есть значение feed (если перед обновлением строки не убрать feed, то читатели увидят дубликат записи), а LIMIT 10 останавливает перебор и вывод после первых 10-и найденных
            header("Content-Type: application/rss+xml;");
            echo '<?xml version="1.0"?>
  <rss version="2.0">
    <channel>
      <title>Шпаргалка блоггера</title>  // своё название RSS-ленты
      <link>http://shpargalkablog.ru/</link>  // свой URL сайта, данные которого транслируются в RSS-ленте
      <description>Вебмастеру в помощь: SEO, аналитика, HTML, CSS, JS, PHP</description>  // своё описание RSS-ленты одним предложением
      <language>ru</language>';
            while ($row = $result->fetch_assoc()) {
                echo '
        <item>
          <title>'. $row['title'] .'</title>
          <link>http://shpargalkablog.ru/'. $row['url'] .'</link>
          <description>'. $row['description'] .'</description>
          <pubDate>'. date('r', strtotime($row['update'])) .'</pubDate>
          <guid>http://shpargalkablog.ru/'. $row['url'] .'</guid>
        </item>';
            }
            echo '
    </channel>
  </rss>';
        }

    }
    $mysqli->close();
    exit();
} else { // 404 ошибка
    http_response_code(404);
```

```
    include_once 'http://site.ru/404.php'; // подключить файл со своим оформлением 404 ошибки
    exit();
  }
```

Файл .htaccess

```
  # если нужен редирект с /rss.php на Feedburner (см. страница справки)
  RewriteEngine On # если запись отсутствует
  RewriteCond %{HTTP_USER_AGENT} !FeedBurner [NC]
  RewriteCond %{QUERY_STRING} ^$ [NC]
  RewriteRule ^rss.php$ http://feeds.feedburner.com/shpargalkablog [R=301,L]
```

HTML разметка страниц сайта

Не нужно непременно искать адрес RSS-ленты. В RSS-ридер достаточно добавить адрес любой страницы сайта, в исходном коде которой есть запись:

```
  <!DOCTYPE html>
  <html>
  <head>
  <link title="RSS | Шпаргалка блоггера" type="application/rss+xml" rel="alternate"
  href="/rss.php"/>
```

Для того, чтобы читать **RSS-ленту** нам необходимо найти ссылку на эту самую ленту. Для этого можно зайти на сайт http://news.yandex.ru/export.html. Там выбираете категорию, которая вам понравилась, и копируете ссылку, например такую – http://news.yandex.ru/gadgets.rss.

Далее необходимо прочитать содержимое файла **gadgets.rss,** для этого воспользуемся встроенной в PHP функцией file_get_contents. Код далее:

```
<?php
  $url = “http://news.yandex.ru/gadgets.rss”;
  $content = file_get_contents($url);
?>
```

Далее мы передаем содержимое переменной $content в конструктор классаSimpleXmlElement и получим объектное представление содержимого RSS-ленты в переменной $items.

```
<?php
  $items = new SimpleXmlElement($content);
  print “<ul>”;
  foreach($items -> channel -> item as $item) {
    print “<li><a href = ‘{$item->link}’ title = ‘$item->title’>” . <
    $item->title . ”</a> - ” . $item -> description . ”</li>”; <
  }
  print “</ul>”;
?>
```

Итоговая функция для получения содержимого RSS-ленты будет следующая:

```
<?php
function getFeeds($url) {
  $url = “http://news.yandex.ru/gadgets.rss”;
  $content = file_get_contents($url);
  $items = new SimpleXmlElement($content);
  print “<ul>”;
   foreach($items -> channel -> item as $item) {
    print “<li><a href = ‘{$item->link}’ title = ‘$item->title’>” .
    $item->title . ”</a> - ” . $item -> description . ”</li>”;
   }
  print “</ul>”;
```

```
  }
?>
```

По стандарту, **RSS-лента всегда** следует похожей базовой структуре: каждая лента включает в себя корневой тег **“channel”.** Затем каждый элемент в ленте представлен тегом **channel**, у которого есть свои теги:

- link - ссылка на новость
- title - отображаемый текст ссылки
- description - небольшой вводный текст новости

Таким образом, сегодня мы с Вами научились разбирать **RSS-ленту с помощью PHP.**

Содержание отчета

1. Цель
2. Ход работы с выдержками из сценариев.
3. Выводы

Контрольные вопросы

1. Массив $_REQUEST.
2. Особенности метода GET.
3. Особенности метода POST.
4. Переменные окружения.
5. Опишите алгоритм создания гостевой книги (форума).
6. Какие функции вы использовали при создании сценария?
7. Какие операторы используются для работы с файлами?
8. Какие управляющие последовательности вы использовали?

Лабораторная работа № 23

Создание проекта «Интернет магазин»

Цель: сформировать навыки использования возможностей PHP для создания Web-приложений.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

 Теоретический материал:

Интернет-магазины

Интернет-магазины являются воплощением электронной коммерции в ее классическом понимании (купля-продажа товаров и услуг в Интернете). Интернет-магазин представляет собой компанию, осуществляющую торговлю в Интернете с помощью Web-сайта.

Сайт Интернет-магазина содержит каталоги товаров с их описаниями, фотографиями и ценами. Специальная форма online-заказа позволяет клиентам выбрать, заказать и оплатить интересующие их товары, заранее рассчитать стоимость всего заказа с учетом доставки. Как правило, клиент имеет возможность отслеживать на сайте магазина то, в какой стадии находится исполнение его заказа. Часто Интернет-магазины размещают в специальных разделах или в описаниях конкретных товаров отзывы покупателей и другую полезную для клиентов информацию. Ассортимент товаров Интернет-магазина может колебаться от единиц до многих десятков тысяч наименований.

Прежде, чем заказать товар в Интернет-магазине, пользователь обычно должен зарегистрироваться. Далее, ознакомившись с ассортиментом и ценами и «сложив» интересующие его товары в виртуальную корзину, покупатель выбирает способ доставки и оплаты товара. После указания всех этих параметров покупатель получает итоговую стоимость заказа и, если она его устраивает, подтверждает заказ. После получения заказа администрация магазина связывается с покупателем посредством электронной почты или по телефону для подтверждения заказа и уточнения условий доставки (если в этом есть необходимость). В зависимости от выбранных условий, клиент оплачивает заказ при получении или совершает предоплату одним из традиционных способов или при помощи специализированных систем Интернет-платежей.

Виртуальный магазин (Интернет-магазин) – это реализованное путем создания Web-сервера в сети Интернет представительство для продажи товаров и услуг другим пользователям сети Интернет. Виртуальный магазин – это сообщество территориально разобщенных сотрудников магазина (продавцов, кассиров) и покупателей, которые могут общаться и обмениваться информацией через электронные средства связи при полном (или минимальном) отсутствии личного прямого контакта.

Виртуальный магазин имеет доменный адрес. Как любой Web-сервер, виртуальный магазин состоит из целого ряда гипертекстовых страниц, зачастую с мультимедийными элементами (мультимедиа – это компьютерная технология, позволяющая гибко управлять потоками разнородной информации, т. е. информации, представленной в виде текстов, графиков, видеоизображений, картинок, музыки и т. п.).

Функции, осуществляемые он-лайн магазином, совершенно банальны и сводятся к двум основным: во-первых, предоставить клиенту информацию о товаре (услуге); во-вторых - получить от клиента заказ на товар (услугу). Иногда (при использовании онлайновых платежных систем, которые далее будут рассмотрены более подробно) добавляется третья функция – получение оплаты; а при торговле информацией еще и четвертая – отправка оплаченного товара.

Сегодня под названием "он-лайн магазин" предлагается целый спектр решений различного масштаба и назначения. Можно предложить следующие градации:

- интернет-витрина;
- торговый автомат;
- автоматический магазин.

Интернет-витрина – это логичное и простое расширение банального Web-сайта. Просто на сайт выкладывается информация о товарах и с минимальной регулярностью обновляется. Таким образом, все, что может сделать посетитель, - это получить информацию (более или менее подробную и актуальную) о товарах и услугах. Практическая полезность такой витрины вполне очевидна, затраты на ее создание и администрирование могут быть довольно низкими, но это еще не торговля, а лишь разновидность рекламы. В этом решении отсутствует интеграция с бизнес-процессом торгующей фирмы. Для осуществления покупки потенциальный покупатель сначала должен посетить Интернет-витрину, а позже пройти обычный цикл покупки: звонок, визит или электронное письмо в компанию, оплата и т. д.

Название "торговый автомат" очень хорошо описывает следующий вид решений. Кроме тех функций, которые осуществляет Интернет-витрина, торговый автомат может принимать заказы, а затем в он-лайне, либо пакетном режиме, передавать их менеджеру. Дальнейшая обработка заказа производится по обычной для компании схеме. Так как в данном случае речь идет о реальном товаре, то становится необходимой синхронизация содержимого сайта с реальными ценам и на товар.

Проблема отслеживания наличия товара на складе решается следующим образом: в автомат вводится лимит, в пределах которого производятся продажи. Периодически лимит изменяется - по мере изменения остатков на складе (предположим, раз в сутки) - аналогично тому, как это происходит с автоматом, торгующим газетами, который продает партию газет, заложенную в него оператором, а потом требует перезарядки. Принципиальное отличие торгового автомата от Интернет-витрины состоит в том, что заказы на покупку и счета на оплату заказанного товара выписываются без участия человека. Таким образом, значительно вырастает возможность ошибки.

В отличие от Интернет-витрины, торговый автомат осуществляет реальную торговлю и соотношение затрат к результату выглядит наиболее предпочтительным для пилотных и тестовых проектов с небольшими потоками покупателей. Для использования торгового автомата уже необходима некоторая интеграция с бизнес-процессом, так как обновление информации на сайте должно происходить регулярно и достаточно быстро. В связи с этим возникает необходимость сопряжения базы данных предприятия с торговым автоматом.

Типовая структура Интернет-магазина

Программный комплекс управления Интернет-магазином или торговой частью системы - это программное обеспечение позволяющее разрабатывать и поддерживать торговую систему, работающую в онлайновом режиме.

Структура комплекса управления Интернет-магазином или торговой частью системы реализуется в виде трехзвенной архитектуры клиент/сервер

Процесс обработки данных происходит по схеме "клиент - сервер приложений - база данных". Поступивший запрос обрабатывается сервером приложений, который в свою очередь связывается с хранилищем данных и платежной системой, а при наличии подключения к бизнес процессу организации, производит обмен данными с соответствующими системами.

В общем случае минимум компонентов необходимых для функционирования Интернет-магазина включает в себя:

• **Web-сервер** - распределяет поступающие запросы, производит разграничение доступа;

• **Сервер приложений** - управляет работой всей системы, в частности бизнес-логикой Интернет-магазина ;

• **СУБД** - осуществляет хранение и обработку данных о товарах, клиентах, счетах и т.п.

К этому комплексу подключаются платежные системы, а в некоторых случаях и системы доставки. Для полной интеграции с бизнес процессами компании может быть организован шлюз для электронной передачи данных между Интернет-магазином и внутренней системой автоматизации документооборота.

СУБД phpmyadmin

Система управления базами данных (СУБД) —совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных .

В состав денвера входит СУБД под названием phpmyadmin. Чтобы начать работать с ней, нужно запустить денвер и в адресной строке набрать http://localhost/Tools/phpMyAdmin/

. Изучите работу с phpmyadmin. Для работы с базой данных используйте расширение MySQLi. MySQLi является улучшенной версией старого драйвера PHP MySQL, предлагающего различные улучшения. В файле для регистрации сразу после строчки с началом сессии подключитесь к базе данных. Это делается при помощи функции mysqli_connect(). Она создает соединение с MySQL сервером.

$link = mysqli_connect('localhost', /* Хост, к которому мы подключаемся */ 'root', /* Имя пользователя */

", /* Используемый пароль */ "); /* База данных для запросов по умолчанию *Создайте проверку на подключения к базе данных.

if (!$link) {

printf("Невозможно подключиться к базе данных. Код ошибки: %s\ n",mysqli_connect_error());

exit;

}

Используя функцию mysqli_query($link, «sql запрос») можно отправлять различные запросы в базу данных. Для этого используется язык запросов SQL.

С синтаксисом языка можно ознакомиться на википедии.

Создайте базу данных, если она не существует.

mysqli_query($link, 'CREATE DATABASE IF NOT EXISTS my_db');

Теперь нужно подключиться к нужной базе. Это можно сделать функцией mysqli_select_db().

mysqli_select_db($link,'my_db');

Удалите все функции для работ ы с файлами. После проверки паролей создайте таблицу с полями id, username, email, password с проверкой существования таблицы

$query = "CREATE TABLE IF NOT EXISTS users(`id` INT(11) NOT NULL AUTO_INCREMENT ,`username` VARCHAR(150) NOT NULL ,`email` VARCHAR(50) NOT NULL ,`password` VARCHAR(50) NOT NULL ,PRIMARY KEY (`id`));";

mysqli_query($link, $query);

Вставьте в базу данные, которые пришли из POSTзапроса.

mysqli_query($link, 'INSERT INTO users(`username`, `email`, `password`) VALUES ('".$_POST['username']."','".$_POST['email']."', '".$_POST['password']."')');

Если какой-то запрос не выполняется, всегда можно посмотреть в чем ошибка, используя функцию

mysqli_error($link). Она хранит последнюю ошибку.

В файле index.php после запуская сессии подключитесь к базе и попытайтесь создать базу данных как и в registration.php. Создаем таблицу с полями id,

id_username, title, text.

id_username хранит в себе id пользователя из таблицы users. После проверки POST запроса уберите все функции для работы с файлами и сделайте запрос в базу для получения id пользователя, присвойте его переменной $id. При вылнении запроса

SELECT, mysqli_query возвращает массив, который нужно обработать функцией

mysqli_fetch_assoc(), на выходе которой получается ассоциативный массив.

$result = mysqli_query($link, 'SELECT id FROM `users` WHERE username='".$_SESSION['username']."'");

while($row = mysqli_fetch_assoc($result)){ $id= $row['id'];}

Указания по выполнению

1. Создайте файл заказа. Код файла index.php:

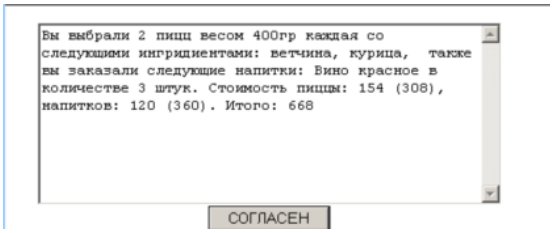
```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=Windows-1251">
  <title>Лабораторная работа № 31-33</title>
</head>
<body>
<font face=Verdana>
<center><font size=6><b>ФОРМА ЗАКАЗА</b></font></center><br>
<form method="post" action="check.php" name="form01">
  <table border=1 cellpadding="2" cellspacing="2" width=100%>
      <tr>
        <td><big><b>Пицца</b></big>     Кол-во:<input name="k01" value="1" size=1> шт.</td
        <td><big><b>Вес</b></big></td>
        <td><big><b>Напитки</b></big>     Кол-во:<input name="k02" value="1" size=1> шт.</
      </tr>
      <tr>
        <td><input name="opt01" type="checkbox">грибы<br>
        <input name="opt02" type="checkbox">помидоры<br>
        <input name="opt03" type="checkbox">сыр<br>
        <input name="opt04" type="checkbox">ветчина<br>
        <input name="opt05" type="checkbox">ананасы<br>
        <input name="opt06" type="checkbox">курица<br>
        <input name="opt07" type="checkbox">колбаса
        </td>
        <td><input name="one01" type="radio" value="z01">100гр<br>
        <input name="one01" type="radio" value="z02">200гр<br>
        <input name="one01" type="radio" value="z03">400гр<br>
        <input name="one01" type="radio" value="z04">600гр<br>
        <input name="one01" type="radio" value="z05">1000гр<br>
        </td>
        <td><input name="one02" type="radio" value="z01">Сок<br>
        <input name="one02" type="radio" value="z02">Пиво<br>
        <input name="one02" type="radio" value="z03">Вино белое<br>
        <input name="one02" type="radio" value="z04">Вино красное<br>
        <input name="one02" type="radio" value="z05">Кока-кола<br>
        </td>
      </tr>
<tr>
<td colspan=3 align=center><input name="ok" value="ЗАКАЗАТЬ" type="submit">
<input name="canc" value="Очистить" type="reset"></td></tr>
  </table>
</form>
</font>
</body>
</html>
```

Вид формы заказа:

2. Создайте файл, который выполняет функцию обработки заказа. Код файла check.php:

```php
<?
error_reporting(0);
    switch($one01)
    {
      case "z01":
      $ves="100гр";
$koof=1.1;
        break;
       case "z02":
      $ves="200гр";
$koof=1.2;
       break;
       case "z03":
      $ves="400гр";
$koof=1.4;
        break;
       case "z04":
      $ves="600гр";
$koof=1.6;
       break;
       case "z05":
      $ves="1000гр";
$koof=2;
       break;
//       default:
//          statements;
    }
    switch($one02)
    {
       case "z01":
      $vot="Сок";
      $vot_cena=40;
        break;
       case "z02":
      $vot="Пиво";
      $vot_cena=20;
       break;
       case "z03":
      $vot="Вино белое";
      $vot_cena=100;
        break;
       case "z04":
      $vot="Вино красное";
      $vot_cena=120;
       break;
       case "z05":
      $vot="Кока-кола";
      $vot_cena=10;
        break;
//       default:
//          statements;
    }
$vot_cena_all=$vot_cena*$k02;
$komp_cena=20;
if ($opt01==True) {
$komp=$komp . "грибы, ";
$komp_cena=$komp_cena+10;
}
if ($opt02==True) {
$komp=$komp . "помидоры, ";
$komp_cena=$komp_cena+15;
}
if ($opt03==True) {
$komp=$komp . "сыр, ";
$komp_cena=$komp_cena+20;
}
if ($opt04==True) {
$komp=$komp . "ветчина, ";
$komp_cena=$komp_cena+30;
}
if ($opt05==True) {
$komp=$komp . "ананасы, ";
$komp_cena=$komp_cena+25;
}
if ($opt06==True) {
$komp=$komp . "курица, ";
$komp_cena=$komp_cena+60;
}
if ($opt07==True) {
$komp=$komp . "колбаса, ";
$komp_cena=$komp_cena+20;
}
$komp_cena_ves=$komp_cena*$koof;
$komp_cena_all=$komp_cena_ves*$k01;
$cena_vsego=$komp_cena_all+$vot_cena_all;
?>
<form method="post" action="apply.php" name="form01">
<center><textarea cols="50" rows="10" name="z1">
<?
echo "Вы выбрали $k01 пицц весом $ves каждая со следующими ингридиентами:
 $komp также вы заказали следующие напитки: $vot в количестве $k02 штук. Стоимость пиццы: $komp_cena_ves ($komp_cena_all),
 напитков: $vot_cena ($vot_cena_all). Итого: $cena_vsego"; ?>
</textarea><br>
<input name="ok" value="СОГЛАСЕН" type="submit"></center></form>
```

Вид конечного заказа:



3. Создайте файла apply.php, выполняющего функцию сохранения заказа:

```
<?
$today = date("ymd_Gis");
$abc = fopen ($today, "a");
fwrite ($abc, $z1);
fclose($abc);
header("Location: index.php");
?>
```

Работа формы заказа начинается с главной страницы index.php на которой располагаются возможные варианты товаров, которые магазин предоставляет покупателю. После выбора необходимого товара (или его совокупности), управление передается файлу check.php, в котором происходит постобработка введенных данных - форматирование сообщения о заказанных товарах и предоставляет покупателю возможность просмотреть его заказа и отменить в случае неверного заполнения формы. После нажатия на кнопку «Согласен» заказ записывается в файл вида <год><месяц><день>_<текущее время> и происходит автоматический переход на главную страницу, где можно сделать новый заказ.

Содержание отчета

1. Цель
2. Типовая структура Интернет-магазина
3. Ход работы с выдержками из сценариев.
4. Выводы

Контрольные вопросы

1. Сколько файлов необходимо создать для простейшего интернет-магазина?
2. Опишите содержимое этих файлов, связь между ними.
3. Интернет-магазин, веб-витрина, особенности

Лабораторная работа № 24

Разработка блога на PHP с администрированием статей (архитектура MVC)

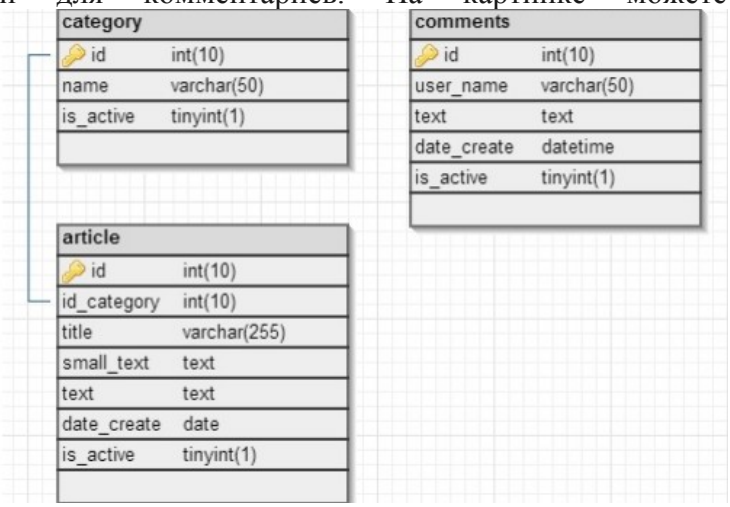
Цель: сформировать навыки использования возможностей PHP для создания Web-приложений.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Прежде чем приступить к разработке блога, нужно спроектировать базу данных, но еще раньше нужно определиться, что у нас вообще должно получиться. Давайте для начала создадим не большой, в плане функционала, блог, который будет иметь категории статей, статьи и будет возможность писать комментарии к статьям. Возможно, потом расширим возможности, но пока для того, чтобы лучше понять MVC нам и этого хватит. Итак, приступим к созданию бд, нам потребуется три таблицы – для категорий, для статей и для комментариев. На картинке можете посмотреть структуру таблиц:



А также создать бд можете, воспользовавшись sql-патчем:

```
CREATE TABLE `category` (
  `id` INT(10) NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(50),
  `is_active` TINYINT(1),
  PRIMARY KEY  (`id`)
);

CREATE TABLE `article` (
  `id` INT(10) NOT NULL AUTO_INCREMENT,
  `id_category` INT(10),
  `title` VARCHAR(255),
  `small_text` TEXT,
  `text` TEXT,
  `date_create` DATE,
  `is_active` TINYINT(1),
  PRIMARY KEY  (`id`)
);

CREATE TABLE `comments` (
  `id` INT(10) NOT NULL AUTO_INCREMENT,
  `user_name` VARCHAR(50),
  `text` TEXT,
  `date_create` DATETIME,
  `is_active` TINYINT(1),
  PRIMARY KEY  (`id`)
);
```

ALTER TABLE `article` ADD CONSTRAINT `article_fk1` FOREIGN KEY (`id_category`) RE

Проектируем структуру блога

- Создадим модели для таблиц

- Создадим контроллеры и экшены
- Создадим шаблон и вьюхи(отображения)

Разработка функционала

Теперь приступим к самой интересной части – разработке функционала, начнем с главной страницы – контроллер index, экшен index, тут мы ни чего делать не будем, а просто выведем текст, то есть просто будем подключать нужную вьюху:

$this->template->view('index');

Давайте создадим контроллер, который будет отвечать за вывод всех статей для определенный категории — Controller_Category. Мы будем передавать в него ID категории и делать по нему запрос:

```
// экшен
function index() {
    $idCategory = (isset($_GET['id'])) ? (int)$_GET['id'] : false;
    if($idCategory){
        $select = array(
            'where' => "is_active = 1 AND id_category = $idCategory", // условие
            'order' => 'date_create DESC' // сортируем
        );
        $model = new Model_Article($select); // создаем объект модели
        $articles = $model->getAllRows(); // получаем все строки
    }else{
        $articles = false;
    }

    $this->template->vars('articles', $articles);
    $this->template->view('index');
}
```

Во вьюхе для этого контроллера и экшена сделаем вывод заголовков, короткого описания и ссылки на полный текст статьи.

И наконец, в последнем контроллере(Controller_Article) реализуем получение конкретной статьи по ее ID:

```
function index() {
    $idArticle = (isset($_GET['id'])) ? (int)$_GET['id'] : false;
    if($idArticle){
        $select = array(
            'where' => "id = $idArticle" // условие
        );
        $model = new Model_Article($select); // создаем объект модели
        $article = $model->getOneRow(); // получаем статью
    }else{
        $article = false;
    }

    $this->template->vars('article', $article);
    $this->template->view('index');
}
```

Во вьюхе, которая подключается в этом контроллере выведем заголовок статьи и полное описание.

Содержание отчета

1. Цель
2. Типовая структура Интернет-магазина
3. Ход работы с выдержками из сценариев.
4. Выводы

Контрольные вопросы

1. Сколько файлов необходимо создать для простейшего интернет-магазина?
2. Опишите содержимое этих файлов, связь между ними.
3. Интернет-магазин, веб-витрина, особенности

Лабораторная работа №25

Составление схем XML-документов

Цель: *ознакомиться с XML, научиться разбирать структуру XML-документа; ознакомиться с XSL, с помощью XSL научиться выполнять преобразование XML-документа с одной структурой в XML-документ с другой структурой, в частности, в HTML; сформировать навыки сортировки, фильтрации и выборки данных из документа XML.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

XML (*Extensible Markup Language*) - это язык разметки, описывающий целый класс объектов данных, называемых XML- документами. Этот язык используется в качестве средства для описания грамматики других языков и контроля за правильностью составления документов. Т.е. сам по себе XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания.

Сам процесс создания XML документа очень прост и требует от нас лишь базовых знаний HTML и понимания тех задач, которые мы хотим выполнить, используя XML в качестве языка разметки. Таким образом, у разработчиков появляется уникальная возможность определять собственные команды, позволяющие им наиболее эффективно определять данные, содержащиеся в документе. Автор документа создает его структуру, строит необходимые связи между элементами, используя те команды, которые удовлетворяют его требованиям и добивается такого типа разметки, которое необходимо ему для выполнения операций просмотра, поиска, анализа документа.

Структура XML- документа

Любой XML-документ состоит из следующий частей:

- Необязательный пролог
- Тело документа
- Необязательный эпилог, следующий за деревом элементов

Стилевые таблицы XSL, Переменные и параметры

Переменные и **параметры** в XSLT очень похожи между собой и сильно отличаются от переменных в традиционных языках программирования. Классическое определение переменной, характерное для большинства языков программирования звучит следующим образом:

- Переменная - это имя, присвоенное ячейке памяти или ячейкам, которые могут содержать представление конкретного типа данных
- Значение переменной может изменяться в процессе выполнения программы.

С точки зрения логической модели преобразования, такие технические детали, как адрес переменной в памяти, нас не интересуют, хотя он несомненно существует. Гораздо более удобным является другое определение:

- Переменная - это объект определенного типа, с которым связанно **имя**

По этому имени мы можем обращаться к объекту, использовать его значение и т.д. Иными словами, в XSLT под переменной понимается не более чем ассоциация между значением и именем, и если мы скажем, что переменная **x** имеет значение **5**, это будет означать, что имя **"x"** связанно с объектом численного типа, значение которого равно **5**.

Теперь, когда с понятием переменной в XSLT кое-что прояснилось, отметим следующий факт: **переменные в XSLT не могут быть изменены!** Разработчикам, которые до сих пор не имели опыта логического программирования и использовали в своей практике традиционные языки программирования, будет нелегко с этим смириться.

То, что *переменные* не могут быть *изменены*, дискредитирует само название - *переменные*, ибо они уже более похожи на константы.

В XSLT объявление переменной есть создание ассоциации между объектом и именем.

Элемент xsl:variable

Синтаксис:

```
<xsl:variable name="имя" select="выражение">
  <!-- Содержимое: шаблон -->
</xsl:variable>
```

Элемент **xsl:variable** в XSLT используется для связи имени переменной, указанного атрибутом **name** со значением. Значение переменной может быть полученно вычислением выражения, заднного необязательным атрибутом **select** или выполнением шаблона в содержимом элемента. Использовать объявленную таким образом переменную можно, указывая перед именем символ $:

$имя

Переменные XSLT могут быть глобальными и локальными. Если элемент xsl:variable является элементом верхнего уровня, то переменная будет глобальной; если элемент xsl:variable объявлен внутри шаблонного правила - локальной.

Областью видимости глобальной переменной является все преобразование. То есть значение переменной, объявленной элементом верхнего уровня, может быть использовано в преобразовании где угодно. К такой переменной можно обращаться даже до ее объявления. Единственное ограничение: **переменная не должна объявляться через собственное значение**.

Локальную переменную можно использовать только после объявления и только в том же родительском элементе, которому принадлежит объявляющий элемент **xsl:variable**. Если существует одноименная глобальная переменноя, то локальная переменная в своей области видимости перекрывает ее. Имена локальных переменных могут совпадать, если их области видимости не пересекаются.

Использование переменных

Переменные могут содержать значения выражений, которые многократно используются в преобразовании. Это избавит процессор от необходимости пересчитывать выражение каждый раз по новому.

Переменной может присваиваться результат преобразования, что позволяет манипулировать уже сгенерированными частями документа.

Случай 1. Первый случай использования совершенно очевиден: если в преобразовании многократно используется выражение, требующее серьезных вычислений или просто громоздкое для записи, переменная может быть использованна для хранения единожды вычисленного результата. Например, если мы множество раз обращаемся ко множеству ссылок данного документа посредством вычисления выражения вида

//a

Гораздо удобнее и экономней с точки зрения вычислительных ресурсов объявить переменную вида

```
<xsl:variable name="links" select="//a" />
```

и использовать ее в преобразовании как $links. Например:

```
...количество ссылок в документе: <xsl:value-of select="count($links)" />
```

Другим примером к этому же случаю может служить следующая ситуация: выражение просто для вычисления, но очень громоздко для записи. Гораздо приятней один раз вычислить значение этого выражения и присвоить его какой нибудь переменной, а затем использовать вышеописонную нотацию для обращения к нему. Например:

Объявление

```
<xsl:variable name="image_path" select="http://www.site.com/chapter-files/images">
```

Использование

```
<img>
  <xsl:attribute name="src">
    <xsl:value-of select="concat($image_path, '/picture1_1.gif')" />
  </xsl:attribute
</img>
```

Случай 2. Второй типовый случай использования переменных также заметно облегчает создание выходного дерева, делая этот процесс гибким и легко конфигурируемым. Примером формирования HTML документа при помощи такого подхода может быть следующий шаблон:

```
<xsl:template match="/">
  <html>
    <xsl:copy-of select="$head" />
    <xsl:copy-of select="$body" />
  </html>
</xsl:template>
```

Достоинство такого подхода состоит в том, что переменные, содержащие фрагменты деревьев, как бы становятся модулями, из которых в итоге собирается документ.

Более практичным применением возможности переменных содержать фрагменты деревьев является условное присваивание переменной значения. Представим себе следующий алгоритм:

если условие1 то
  присвоить переменной1 значение1
иначе
  присвоить переменной1 значение2

В традиционном языке программирования с изменяемыми переменными (например C++ или Javascript) это выглядело бы так:

переменная1=условие1?значение1:значение2;

или в чуть более широком варианте:

if (условие1) переменная1=значение1
else
  переменная1=значение2

Однако если в XSLT написать что нибудь подобное

```
<xsl:choose>
  <xsl:when test="условие1">
    <xsl:variable name="переменная1" select="значение1" />
  </xsl:when>
  <xsl:otherwise>
    <xsl:variable name="переменная1" select="значение2" />
  </xsl:otherwise>
</xsl:choose>
```

то требуемого результата все равно не достигли бы по той причине, что переменная будет иметь в этом случае локальную область видимости, ограниченную элементом **xsl:when** или **xsl:otherwise** и их дочерними элементами. Правильный шаблон для решения этой задачи выглядит следующим образом:

```
<xsl:variable name="переменная1"
  <xsl:choose>
    <xsl:when test="условие1">
      <xsl:copy-of select="значение1" />
```

```
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy-of select="значение2" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:variable>
```

Конечно, это не то же самое - на самом деле мы получаем не само значение, а дерево, содержащее это значение, но для строковых и численных значений особой разницы нет: дерево будет вести себя так, как будто это число или строка (см. правила приведения типов в одном из предыдущих уроков).

Параметры

Параметры в XSLT практически полностью идентичны переменным. Они точно так же связывают с объектом имя, посредством которого можно к ним обращиться. Главным различием является то, что значение, данное параметру при инициализации, является всего лишь значением по умолчанию, которое может быть переопределенно при вызове.

Давайте немного изменим угол зрения и посмотрим на шаблоны не как на правила преобразования, а представим их себе как *функции*, каждая из которых преобразует входной узел и возвращает фрагмент дерева в качестве результата. С этой точки зрения параметры шаблонов являются ни чем иным, как *аргументами* этих функций.

Работа с параметрами обеспечивается двумя элементами XSLT:

xsl:param - служит для объявления параметра;

xsl:with-param - указывает значение параметра ри вызове шаблона.

xsl:param

Синтаксис этого элемента следующий:

```
<xsl:param name="имя" select="выражение">
  <!-- Содержимое: шаблон -->
</xsl:param>
```

Элемент **xsl:template**, задающий в преобразовании шаблонное правило, может включать несколько элементов **xsl:param**, которые и будут определять параметры этого шаблона. Кроме того, **xsl:param** может быть элементов верхнего уровня - в этом случае объявляемый параметр будет *глобальным*.

Атрибут **name** задает имя параметра. Имя параметра может имет расширенную форму, например "user:param", однако, чтобы не возиться с пространствами имен, на практике всегда используют простые имена: "i", "count" или что-то в этом роде.

Параметру может быть присвоенно значение по умолчанию.Это значение будет использованно с том случае, если параметра с таким именем шаблону при вызове переданно не было. Значение по умолчанию вычисляется следующим образом:

если в элементе xsl:param определен атрибут select, то значением по умолчанию будет результат вычисления выражения, указанного в этом атрибуте. Пример:

<xsl:param name="x" select="2 * 2" />

если атрибут select не определен, носам элемент xsl:param содержит дочерние узлы, то значением параметра по умолчанию будет фрагмент дерева, полученный в результате выполнения содержимого xsl:param. Пример:

```
<!--
  в результате будет получен фрагмент дерева
  состоящий из корневого узла и текстового узла "4"
-->
<xsl:param name="x">
  <xsl:value-of select="2 * 2" />
</xsl:param>
```

если атрибут select не определен и сам элемент xsl:param не содержит дочерних элементов, то значением по умолчанию будет пустая строка. Пример:

```
<xsl:param name="x" /> <!-- получим параметр x = " -->
```

Область видимости параметра определяется так же как область видимости переменной. Единственное, что нужно запомнить - это то, что элементы **xsl:param**, определяемые в шаблонах, должны всегда быть его **первыми дочерними элементами**. Абсолютно логично и то, что в одном шаблоне **не может быть объявленно 2 или более одноименных параметра**, так как их области видимости обязательно будут пересекаться.

xsl:with-param

Синтаксис:

```
<xsl:with-param name="имя" select="выражение">
  <!-- Содержимое: шаблон -->
</xsl:with-param>
```

Синтаксис **xsl:with-param** абсолютно идентичен синтаксису **xsl:param** и отличаются они только именем. Практически настолько же похоже и их действие, только элемент **xsl:with-param** связывает параметр со значением, которое при вызове шаблона будет использоваться **вместо** значения параметра по умолчанию. Таким образом, значение параметра, заданного в шаблоне, выбирается следующим образом:

- если в элементе, который вызывает этот шаблон, присутствует элемент xsl:with-param, передающий значение этого параметра, то в шаблоне будет использоваться переданное значение;
- в противном случае будет использоваться значение по умолчанию.

Указания по выполнению

1. Создайте файл XML.Откройте блокнот, введите текст и сохраните как index.xml.

```
<?xml version="1.0" encoding="windows-1251" ?>
<notepad>
    <note id="1" date="11/04/99" time="13:30">
    <subject>Важная деловая встреча</subject>
    <text>
      Надо встретиться с
      <person id="1625">Иваном Ивановичем</person>
      , предварительно позвонив ему по телефону
      <tel>123-12-12</tel>
    </text>
  </note>
    <note id="2" date="12/04/99" time="13:00">
    <subject>Позвонить домой</subject>
    <text>
      <tel>124-13-13</tel>
    </text>
  </note>
   <note id="3" date="13/04/99" time="5:00">
    <subject>Поехать с Максом на рыбалку</subject>
    <text>
      Напомнить Максу чтобы он сварил кашу и взял блёсна.
      <tel>124-10-13</tel>
    </text>
  </note>
  </notepad>
```

2. Откройте файл. В результате выполнения будет получено следующее:

```
<?xml version="1.0" encoding="windows-1251" ?>
- <notepad>
  - <note id="1" date="11/04/99" time="13:30">
      <subject>Важная деловая встреча</subject>
    - <text>
        Надо встретиться с
        <person id="1625">Иваном Ивановичем</person>
        , предварительно позвонив ему по телефону
        <tel>123-12-12</tel>
      </text>
    </note>
  - <note id="2" date="12/04/99" time="13:00">
      <subject>Позвонить домой</subject>
    - <text>
        <tel>124-13-13</tel>
      </text>
    </note>
  - <note id="3" date="13/04/99" time="5:00">
      <subject>Поехать с Максом на рыбалку</subject>
    - <text>
        Напомнить Максу чтобы он сварил кашу и взял блёсна.
        <tel>124-10-13</tel>
      </text>
    </note>
</notepad>
```

3. Создайте связь с таблицей стилей. Для этого в начале файла *index.xml* напишите строку

```
<?xml:stylesheet type="text/css" href="1.css"?>
```

4. Создайте файл содержащий стиль оформления *1.css:*

```
text
{
    font-family: Verdana, Arial, Helvetica, Sans-Serif;
    font-size: 12px;
    font-style: normal;
    color: White;
    background-color: #202036
}
subject
{
    font-family:Arial;
    font-size: 12px;
    font-style:normal;
    color:White;
    background-color:gray
}
tel
{
    color:Yellow
}
```

5. В результате выполнения вы должны получить следующее:

Важная деловая встреча Надо встретиться с Иваном Ивановичем , предварительно позвонив ему по телефону 123-12-12 Позвонить домой 124-13-13 Поехать с Максом на рыбалку Напомнить Максу чтобы он сварил кашу и взял блёсна. 124-10-13

Преобразование XML с помощью XSL

1. Создайте простой пример XML-файла (ex01.xml).

```
<?xml version="1.0" encoding="Windows-1251"?>
<?xml-stylesheet type="text/xsl" href="ex01-1.xsl"?>
<tutorial>
<title>"Заметки об xSL"</title>
<author>Леонов Игорь Васильевич</author>
</tutorial>
```

2. Создайте XSL-файл ex01.xsl. Текст файла приведен ниже.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<p><strong><xsl:value-of select="//title"/></strong></p>
<p><xsl:value-of select="//author"/></p>
</xsl:template>
</xsl:stylesheet>
```

3. Откроете файл ex01.xml в браузере Internet Explorer - задача решена, - на экране осталась только необходимая информация, все теги исчезли. Результат, который вы получите на экране браузера, приведен ниже.

"Заметки об xSL"

Леонов Игорь Васильевич

4. Рассмотрите следующий XML-файл ex02-1.xml. В этом файле информация хранится не в содержании элементов, а в виде значений атрибутов.

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<?xml-stylesheet type='text/xsl' href='ex02-1.xsl'?>
<tutorial>
<dog caption="Собака: " name="Шарик">
<dogInfo weight="18 кг" color="рыжий с черными подпалинами"/>
</dog>
</tutorial>
```

5. Файл ex02-1.xsl имеет вид

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<P><B><xsl:value-of select="//dog/@caption"/></B>
<xsl:value-of select="//dog/@name"/>
<xsl:value-of select="//dogInfo/@weight"/>
<xsl:value-of select="//dogInfo/@color"/></P>
</xsl:template>
</xsl:stylesheet>
```

Обратите внимание на синтаксис ссылки на атрибут элемента - //dog/@name. Имя элемента и имя атрибута разделены парой символов "/@". В остальном синтаксис тот же самый, что и для ссылки на содержание элемента.

6. Создайте следующий XML-файл - ex03.xml.

```
<?xml version="1.0" encoding="WINDOWS-1251"?>

<tutorial>
<enimals>
<dogs>
<dogsCaption>Собаки</dogsCaption>
<dogsCaptionName>Кличка</dogsCaptionName>
<dogsCaptionWeight caption="кг">Вес</dogsCaptionWeight>
<dogsCaptionColor>Цвет</dogsCaptionColor>
<dog>
<dogName>Шарик</dogName>
<dogWeight caption="кг">18</dogWeight>
<dogColor>рыжий с черными подпалинами</dogColor>
</dog>
<dog>
<dogName>Тузик</dogName>
<dogWeight caption="кг">10</dogWeight>
<dogColor>белый с черными пятнами</dogColor>
</dog>
<dog>
<dogName>Бобик</dogName>
<dogWeight caption="кг">2</dogWeight>
<dogColor>бело-серый</dogColor>
</dog>
<dog>
<dogName>Трезор</dogName>
<dogWeight caption="кг">25</dogWeight>
<dogColor>черный</dogColor>
</dog>
</dogs>
</enimals>
</tutorial>
```

7. Предположим, что это результат запроса к базе данных и выведем на экран соответствующую таблицу.

8. Первый шаг - это, как всегда, добавление шаблона преобразования. Модифицируйте файл, добавив в него ссылку на шаблон.

```
2    <?xml-stylesheet type='text/xsl' href='ex03_1.xsl'?>
```

9. В результате получится файл ex03-1.xml. В этот файл добавлен шаблон преобразования ex03-1.xsl:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<table border="1">
<tr bgcolor="#CCCCCC">
 <td align="center"><strong><xsl:value-of select="//dogsCaptionName"/></strong></td>
 <td align="center"><strong><xsl:value-of select="//dogsCaptionWeight"/></strong></td>
 <td align="center"><strong><xsl:value-of select="//dogsCaptionColor"/></strong></td>
</tr>
<xsl:for-each select="tutorial/animals/dogs/dog">
<tr bgcolor="#F5F5F5">
 <td><xsl:value-of select="dogName"/></td>
 <td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of select="dogWeight/@caption"/></td>
 <td><xsl:value-of select="dogColor"/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

 Первые две строки шаблона являются уже привычными. Следующие девять строк - это строка, содержащая заголовки столбцов таблицы. Конструкция для извлечения текста заголовков таблицы вам уже знакома. А вот девятая строка является новой:

```
<xsl:for-each select="tutorial/animals/dogs/dog">
```

Этот элемент шаблона позволяет выбрать и просмотреть все группы информации, полный путь к которым задается списком тегов "tutorial/animals/dogs/dog".

Правильный результат приведен ниже.

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами
Тузик	10 кг	белый с черными пятнами
Бобик	2 кг	бело-серый
Трезор	25 кг	черный

10. **Сортировка**

В предыдущих примерах порядок строк в таблице полностью соответствовал группам тегов в XML-файле. Этот порядок можно изменять. Добавьте в тег

```
<xsl:for-each select="tutorial/animals/dogs/dog">
```

атрибут order-by

```
<xsl:for-each select="tutorial/animals/dogs/dog" order-by="dogName">
```

Наша таблица примет вид (ex03-2.xml, ex03-2.xsl).

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Трезор	25 кг	черный
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами

11. Более интересные результаты можно получить, если отсортировать таблицу по столбцу "Вес". Вначале по аналогии с предыдущим примером - атрибут order-by="dogName" заменим на order-by="dogWeight".

```
<xsl:for-each select="tutorial/animals/dogs/dog" order-by="dogWeight">
```

Результат приведен ниже (ex03-3.xml, ex03-3.xsl).

Кличка	Вес	Цвет

Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами
Бобик	2 кг	бело-серый
Трезор	25 кг	черный

12. Таблица действительно отсортирована по столбцу "вес", но это не числовая, а строковая сортировка! Для того, чтобы браузер воспринял значения как числа, ему необходимо об этом сказать, - вместо order-by="dogWeight" необходимо написать order-by="number(dogWeight)".

```
<xsl:for-each select="tutorial/enimals/dogs/dog" order-by="number(dogWeight)">
```

Результат (ex03-4.xml, ex03-4.xsl):

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами
Трезор	25 кг	черный

13. Приведем теперь пример сортировки по нескольким столбцам. Различные элементы в атрибуте order-by должны разделяться символом ";" - order-by="number(dogWeight); dogName" (ex03-5.xml, ex03-5.xsl).

```
<xsl:for-each select="tutorial/enimals/dogs/dog" order-by="number(dogWeight); dogName">
```

Таблица приведена ниже.

Кличка	Вес	Цвет
Трезор	10 кг	черный
Тузик	10 кг	белый с черными пятнами
Бобик	18 кг	бело-серый
Шарик	18 кг	рыжий с черными подпалинами

14. В следующем примере строки сортируются по одному столбцу - по кличке собаки. Этот пример уже приводился выше, однако теперь мы используем новый синтаксис (ex03-6.xml, ex03-6.xsl).

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<table border="1">
<tr bgcolor="#CCCCCC">
 <td align="center"><strong><xsl:value-of select="//dogsCaptionName"/></strong></td>
 <td align="center"><strong><xsl:value-of select="//dogsCaptionWeight"/></strong></td>
 <td align="center"><strong><xsl:value-of select="//dogsCaptionColor"/></strong></td>
 </tr>
<xsl:for-each select="tutorial/enimals/dogs/dog">
 <xsl:sort order="ascending" select="dogName"/>
<tr bgcolor="#F5F5F5">
 <td><xsl:value-of select="dogName"/></td>
 <td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of select="dogWeight/@caption"/></td>
 <td><xsl:value-of select="dogColor"/></td>
 </tr>
 </xsl:for-each>
 </table>
 </xsl:template>
 </xsl:stylesheet>
```

Отметим разницу.

При использовании нового синтаксиса используется ссылка на другое пространство имен

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

Кроме того, мы убрали атрибут order-by в элементе xsl:for-each и добавили другой элемент

```
<xsl:sort order="ascending" select="dogName"/>
```

Таблица результатов приведена ниже.

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Трезор	25 кг	черный
Тузик	10 кг	белый с черными пятнами
Шарик	18 кг	рыжий с черными подпалинами

15. С использованием нового синтаксиса легко сменить сортировку по возрастанию на сортировку по убыванию (ex03-7.xml, ex03-7.xsl). Разница заключается в одной строке

```
<xsl:sort order="descending" select="dogName"/>
```

Измените значение атрибута order - ascending на descending.

Таблица результатов приведена ниже.

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами
Тузик	10 кг	белый с черными пятнами
Трезор	25 кг	черный
Бобик	2 кг	бело-серый

16. Покажем теперь сортировку по нескольким полям (ex03-8.xml, ex03-8.xsl). В этом примере у нас фигурируют две строки с элементом xsl:sort.

```
<xsl:sort order="ascending" select="number(dogWeight)" data-type="number"/>
<xsl:sort order="ascending" select="dogName"/>
```

Строки вначале сортируются по весу собаки, а затем по их кличкам в алфавитном порядке. Обратите внимание - для того, чтобы сортировка выполнялась в числовой последовательности, в элемент xsl:sort мы добавили атрибут data-type.

Содержание отчета

1. Цель
2. Ход работы
3. Структура XML-документа.
4. Основные инструкции XSL.
5. Выводы

Контрольные вопросы

1. Назначение XML.
2. Назначение XSL.
3. Что представляет собой шаблонное правило в XSL?
4. Инструкции XSL, используемые при выполнении лабораторной работы. Их назначение, синтаксис
1. Как осуществляется связывание данных XML с HTML?
2. Атрибут version.
3. Атрибут encoding.
4. Элемент xsl:value-of.
5. Правила записи комментария.
6. Синтаксис ссылки на атрибут элемента.

Лабораторная работа №26

Отображение XML-документов различными способами

Цель: *ознакомиться с XML, научиться разбирать структуру XML-документа; ознакомиться с XSL, с помощью XSL научиться выполнять преобразование XML-документа с одной структурой в XML-документ с другой структурой, в частности, в HTML; сформировать навыки сортировки, фильтрации и выборки данных из документа XML.*

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

1. Элемент XSL:IF - фильтр

Рассмотрим теперь способы фильтрации строк таблицы. Первый пример использует старый синтаксис. В нем условие фильтрации указывается непосредственно в атрибуте select (ex04-1.xml, ex04-1.xsl).

Ниже приведена строка, в которую мы внесли необходимые изменения.

```
<xsl:for-each select="tutorial/animals/dogs/dog[dogWeight$gt$10] " order-by="dogWeight">
```

И таблица результатов.

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами
Трезор	25 кг	черный

Вы видите, что в таблице остались только те собаки, чей вес превышает 10 кг, причем первым стоит Шарик, чей вес меньше.

2. Более гибкие возможности нам предоставляет новый синтаксис (ex04-2.xml, ex04-2.xsl). Обратите внимание - в новом синтаксисе атрибут order-by в элементе xsl:for-each не поддерживается, вместо него мы вставили два элемента xsl:sort.

```
<xsl:sort order="ascending" select="number(dogWeight)"/>
<xsl:sort order="ascending" select="dogName"/>
```

Кроме того, условие фильтра у нас вынесено в отдельный элемент xsl:if.

```
<xsl:if test="dogWeight&gt;10">
```

Не забывайте указывать конечный тег элемента xsl:if.

```
<xsl:if test="dogWeight&gt;10">
<tr bgcolor="#F5F5F5">
 <td><xsl:value-of select="dogName"/></td>
<td align="right"><xsl:value-of select="dogWeight"/>
 <xsl:value-of select="dogWeight/@caption"/></td>
 <td><xsl:value-of select="dogColor"/></td>
 </tr>
 </xsl:if>
```

В этом примере таблица результатов полностью аналогична предыдущей.

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами
Трезор	25 кг	черный

3. Полностью преимущества нового синтаксиса проявляются при использовании функций.

Рассмотрим следующий пример (ex04-3.xml, ex04-3.xsl). В этом примере используется функция position(), определяющая порядковый номер фрагмента в исходном XML-файле.

Соответствующий элемент xsl:if.

```
]<xsl:if test="position()&gt;2">
```

Результат.

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами
Тузик	10 кг	белый с черными пятнами

4. Продемонстрируем теперь использование более интересных функций - start-with(string,startSubstring) и contains(string,anySubstring). Функция start-with(string,startSubstring) проверяет, начинается ли строка string с подстроки startSubstring. Пример - ex04-4.xml, ex04-4.xsl).

Синтаксис элемента xsl:if.

```
<xsl:if test="starts-with($varDogName,$varStartWith)">
```

В этом элементе мы использовали переменные. Значения переменных были инициализированы ранее

```
<xsl:variable name="varStartWith"><xsl:value-of select="//letter"/></xsl:variable>
<xsl:for-each select="tutorial/animals/dogs/dog">
<xsl:variable name="varDogName"><xsl:value-of select="dogName"/></xsl:variable>
```

Ниже приводится полный код файла ex04-4.xsl:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<table border="1">
  <tr bgcolor="#CCCCCC">
    <td align="center"><strong><xsl:value-of select="//dogsCaptionName"/></strong></td>
    <td align="center"><strong><xsl:value-of select="//dogsCaptionWeight"/></strong></td>
    <td align="center"><strong><xsl:value-of select="//dogsCaptionColor"/></strong></td>
  </tr>
  <xsl:variable name="varStartWith"><xsl:value-of select="//letter"/></xsl:variable>
  <xsl:for-each select="tutorial/animals/dogs/dog">
  <xsl:variable name="varDogName"><xsl:value-of select="dogName"/></xsl:variable>
  <xsl:if test="starts-with($varDogName,$varStartWith)">
  <tr bgcolor="#F5F5F5">
    <td><xsl:value-of select="dogName"/></td>
    <td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of select="dogWeight/@caption"/></td>
    <td><xsl:value-of select="dogColor"/></td>
  </tr>
  </xsl:if>
  </xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

Переменная varStartWith представляет собой подстроку, с которой должны начинаться требуемые нам клички. Она не меняется, поэтому инициализируется перед циклом. Переменная varDogName содержит кличку собаки, она меняется на каждом шаге цикла и, соответственно, инициализируется в теле цикла.

Элемент letter XML-файла содержит букву "T": добавьте в файл XML

```
<letter>T</letter>
```

Результат.

Кличка	Вес	Цвет
Тузик	10 кг	белый с черными пятнами
Трезор	25 кг	черный

5. Функция contains(string,anySubstring) проверяет, содержит ли строка string подстроку anySubstring. Пример - ex04-5.xml, ex04-5.xsl.

Синтаксис элемента xsl:if.

```
<xsl:if test="contains($varDogName,$varStartWith)">
```

Этот пример полностью аналогичен предыдущему. Элемент letter XML-файла содержит букву "o". Добавьте в XML-файл `<letter>o</letter>`

Результат.

Кличка	Вес	Цвет
Бобик	2 кг	бело-серый
Трезор	25 кг	черный

6. Два элемента xsl:if, вложенные друг в друга, дают нам эффект оператора AND (ex04-6.xml, ex04-6.xsl).

Соответствующий фрагмент XSL-файла.

```
<xsl:if test="dogWeight&gt;10">
<xsl:if test="dogWeight&lt;20">
<tr bgcolor="#F5F5F5">
  <td><xsl:value-of select="dogName"/></td>
  <td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of select="dogWeight/@caption"/></td>
  <td><xsl:value-of select="dogColor"/></td>
</tr>
</xsl:if>
</xsl:if>
```

Результат.

Кличка	Вес	Цвет
Шарик	18 кг	рыжий с черными подпалинами

7. Можно добиться и эффекта оператора OR. Для этого нам нужно включить два цикла, в каждом из которых формируется своя выборка (ex04-7.xml, ex04-7.xsl).

Соответствующий фрагмент XSL-файла.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<table border="1">
  <tr bgcolor="#CCCCCC">
    <td align="center"><strong><xsl:value-of select="//dogsCaptionName"/></strong></td>
    <td align="center"><strong><xsl:value-of select="//dogsCaptionWeight"/></strong></td>
    <td align="center"><strong><xsl:value-of select="//dogsCaptionColor"/></strong></td>
  </tr>
  <xsl:for-each select="tutorial/animals/dogs/dog">
  <xsl:sort order="ascending" select="number(dogWeight)" data-type="number"/>
  <xsl:if test="dogWeight&lt;10">
  <tr bgcolor="#F5F5F5">
    <td><xsl:value-of select="dogName"/></td>
    <td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of select="dogWeight/@caption"/></td>
    <td><xsl:value-of select="dogColor"/></td>
  </tr>
  </xsl:if>
  </xsl:for-each>
  <xsl:for-each select="tutorial/animals/dogs/dog">
  <xsl:sort order="ascending" select="number(dogWeight)"/>
  <xsl:if test="dogWeight&gt;15">
  <tr bgcolor="#F5F5F5">
    <td><xsl:value-of select="dogName"/></td>
    <td align="right"><xsl:value-of select="dogWeight"/> <xsl:value-of select="dogWeight/@caption"/></td>
    <td><xsl:value-of select="dogColor"/></td>
  </tr>
  </xsl:if>
  </xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

Результат:

Кличка	Вес	Цвет
Бобик	2кг	бело-серый
Шарик	18кг	рыжий с черными подпалинами
Трезор	25кг	черный

Если сортировка не требуется, то можно вставить два элемента xsl:if в один элемент xsl:for-each.

Элемент XSL:IF - улучшение внешнего вида таблиц

8. Элемент xsl:if можно применять не только для фильтрации строк выборки. Очевидно, что он может быть полезен и во многих других областях. В этом параграфе мы разберем пример использования элемента xsl:if для улучшения внешнего вида таблицы. Заодно мы продемонстрируем реальное использование функции position(). Мы будем использовать эту функцию для того, чтобы чередовать цвет четных и нечетных строк таблицы (ex04-8.xml, ex04-8.xsl).

Фрагмент XSL-файла, который отвечает за требуемое чередование.

```
<xsl:if test="position() mod 2 = 0">
  <xsl:attribute name="bgcolor">#CCCCCC</xsl:attribute>
</xsl:if>
```

С элементом xsl:if и с функцией position() мы уже знакомы. Операция mod 2 дает нам остаток от деления на 2. А элемент xsl:attribute позволяет нам динамически подставлять в файл результатов различные атрибуты. Это очень мощный элемент, мы разберем еще одно применение этого элемента в следующем параграфе.

Результат:

Кличка	Вес	Цвет
Шарик	18кг	рыжий с черными подпалинами
Тузик	10кг	белый с черными пятнами
Бобик	2кг	бело-серый
Трезор	25кг	черный

Задание. Разработайте структуру XML документа, содержащего информацию о работниках организации.

1. Задайте статус работника, имя, страну проживания, организацию, заработную плату. Попробуйте задать структуру документа, используя собственные теги, при этом, можно опираться на образец:

```
<?xml version="1.0" encoding="windows-1251" ?>
<?xml:stylesheet type="text/xsl" href="lab.xsl" ?>
  <catalog>
    <cd>
      <title>Работник спецслужб</title>
      <artist>Kordun Maxim</artist>
      <country>Ukraine</country>
      <company>Sokol</company>
      <prise>19.45</prise>
    </cd>
    <cd>
      <title>Работник спецслужб</title>
      <artist>Sokotsky Igor</artist>
      <country>Ukraine</country>
      <company>Sokol</company>
      <prise>10.45</prise>
    </cd>
    <cd>
      <title>Работник спецслужб</title>
      <artist>Dayneko Alexsey</artist>
      <country>Ukraine</country>
      <company>Sokol</company>
      <prise>12.45</prise>
    </cd>
    </catalog>
```

2. Создайте файл *lab.xsl* , содержащий инструкции по отображению данных в документе XML.

```
<?xml version="1.0" encoding="windows-1251" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <html>
      <body>
        <table border="2" bgcolor="lightyellow" bordercolor="red">
          <tr>
            <th>Статус</th>
            <th>Имя</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

3. **Дополните файлы XSL и XML, чтобы получить следующий результат работы:**

Статус	Имя	Страна	Организация	Зарплата(грн./час)
Работник спецслужб	Kordun Maxim	Ukraine	Sokol	19.45
Работник спецслужб	Sokotsky Igor	Ukraine	Sokol	10.45
Работник спецслужб	Dayneko Alexsey	Ukraine	Sokol	12.45
Служащий	Ivanov Ivan	Ukraine	Sokol	8.45
Служащий	Petrov Alexsey	Ukraine	Sokol	7.45

Содержание отчета

1. Цель
2. Ход работы
3. Пример создания XML и XSL документов.
4. Выводы

Контрольные вопросы

1. Инструкции XSL, используемые при выполнении лабораторной работы. Их назначение, синтаксис
2. Как осуществляется связывание данных XML с HTML?
3. Синтаксис ссылки на атрибут элемента.
4. Элемент XSL:IF
5. Атрибут order-by.
6. Реализация сортировки.

Лабораторная работа №27

Разработка Web-приложения с помощью XML

Цель: изучение программирования поставщиков и клиентов Web-сервисов

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Задачи

Задачами лабораторной работы являются овладение навыками программирования Web-сервисов на различных платформах, построения WSDL-описаний сервисов в автоматизированном режиме, разработки приложений, использующих веб-сервисы в качестве поставщиков данных.

 Теоретический материал:

Веб-сервисы. Наиболее просто веб-сервис представить как интерфейс в глобальную сеть для некоторого абстрактного программного обеспечения. Этот интерфейс позволяет фактически абсолютно прозрачно выполнять какие-то функции, возложенные на это программное обеспечение на удаленном компьютере. Например, на удаленном компьютере находится база данных, например, аэрофлота, и веб-сервис по запросу может позволить получить данные обо всех изменениях в расписании полетов.

Иными словами, каждый Web-сервис – это удаленная функция, к которой можно обратиться через Web, передав некоторый набор входных параметров и получив в ответ выходные значения. То есть если Web-приложение предназначено для организации пользовательского интерфейса к системе через Web-браузер, то Web-сервис нужен для программного доступа со стороны других приложений с использованием Интернет-протоколов.

Структура веб-сервисов. Веб-сервисы базируются на применении открытых, утверждаемых консорциумом IT-сообщества стандартах и протоколах, ключевыми из которых являются следующие:

SOAP. Базовым протоколом, обеспечивающим взаимодействие в среде веб-сервисов, является протокол SOAP (Simple Object Access Protocol). Он позволяет приложениям взаимодействовать между собой через Internet, используя для этого XML-документы, называемые сообщениями SOAP.

UDDI. Спецификация UDDI (UDDI (Universal Description, Discovery and Integration – универсальное описание, расположение и интеграция)) описывает базирующийся на протоколе SOAP веб-сервис, в задачи которого входит определение местоположения и описание протокола взаимодействия любого веб-сервиса. По сути, это каталог доступных веб-сервисов.

WSDL. После того как нужный веб-сервис найден в одном из каталогов UDDI, нужна информация о том, как собственно обратиться к веб-сервису и какие конкретно условия или правила необходимо соблюсти, чтобы сделать это правильно. Для этого используется WSDL (Web Service Description Language – язык описания веб-сервиса), который предназначен, как и следует из его названия, для описания веб-сервисом своих возможностей, своего интерфейса и некоторых метаданных предназначенных для использования теми, кто будет использовать этот веб-сервис.

Веб-сервисы в PHP. Наиболее распространенными способами создания и использования веб-сервисов в PHP являются библиотека NuSOAP и PHP-расширение SOAP Extension. В лабораторной работе предлагается использовать первый из них.

NuSOAP – представляет из себя набор PHP-классов, позволяющих разработчикам создавать и использовать веб-сервисы на SOAP. NuSOAP может также генерировать WSDL-описание веб-сервиса и использовать его., поддерживает различные виды сервисов

(rpc/encoded и document/literal). В то же время необходимо учитывать, что поддержка SOAP и WSDL в NuSOAP реализована не полностью по сравнению с другими реализациями, например, .NET или Apache Axis.

Для использования NuSOAP достаточно скачать его (например, с официального сайта SourceForge), а потом разместить копию файла nusoap. php в дерево своей программы, чтобы можно было подключать ее в PHP-код.

Ключевым классом, который используется при создании веб-сервиса, является soap_server. Методы сервиса представляют собой обычные PHP-функции, которые с помощью метода register преобразуются в веб-методы веб-сервиса.

NuSOAP предоставляет несколько возможностей, полезных для отладки веб-сервиса. В NuSOAP при отладке можно просмотреть посланный запрос и ответ сервера. Класс soapclient содержит атрибуты request и response, которые позволяют отобразить соответственно запрос и ответ.

Кроме того, NuSOAP позволяет автоматически генерировать WSDL для сервиса, используя дополнительные аттрибуты и методы класса soap_server. Информация о сервисе указывается при помощи метода configureWSDL. Информация о каждом методе определяется указанием дополнительных параметров методу register.

Для создания клиента веб-сервиса используется класс soapclient, в качестве параметра которому передается URL-адрес сервиса. Вызов веб-метода выполняется с помощью метода call того же класса soapclient.

Практическая часть

Технология PHP и СУБД MySQL.

1. Программирование Web-сервиса

1. Создать папку, в которой будут размещены все файлы Web-приложения, и назвать ее.

2. Создать папку, в которой будут размещены файлы Web-сервиса, и назвать ее, к примеру, server.

3. Скопировать в папку server файл nusoap. php (размещен на диске D в папке Student) – библиотеку для работы с сообщениями SOAP.

4. В папке server создать новый файл и назвать его, к примеру, index. php. Открыть файл с помощью любого редактора.

5. Ввести открывающий тег сценария <?php

6. Подключить код сценария nusoap. php:

require_once('nusoap. php');

7. Установить соединение с базой данных MySQL University

7.1 Задать объект link соединения с базой данных

$link = @mysql_connect("localhost", "root") or die("Невозможно соединиться с сервером");

7.2 Выбрать базу данных University

$db=@mysql_select_db("university") or die("Нет такой базы данных");

7.3 Установить кириллическую кодировку (cp-1251) для корректной передачи и получения данных из базы данных. Для этого в сценарий PHP из предыдущего пункта ввести следующий код:

@mysql_query("SET SESSION character_set_results = cp1251;");
@mysql_query("SET SESSION Character_set_client = cp1251;");
@mysql_query("SET SESSION Character_set_results = cp1251;");
@mysql_query("SET SESSION Collation_connection = cp1251_general_ci;");
@mysql_query("SET SESSION Character_set_connection = cp1251;");

8. Создать экземпляр SOAP-сервера:

$server = new soap_server;

9. Зарегистрировать метод faculty (извлечения информации о факультетах)

9.1 Вызвать метод register объекта server

$server->register(
9.2 Задать название метода – faculty:
'faculty',
9.3 Указать выходные параметры метода:
array('return' => 'xsd:string'),
9.4 Задать используемое пространство имен:
'uri:facultyquery',
9.5 Задать заголовок SOAPAction
'uri:facultyquery/faculty',
9.6 Задать вид сервиса rpc/encoded
'rpc',
'encoded'
);
10. Определить метод faculty как функцию PHP.
10.1 Задать имя функции (faculty) и открывающую фигурную скобку:
function faculty() {
10.2 Создать переменную f_query для хранения текста запроса к базе данных для извлечения информации о специальностях факультета
$f_query="select * from `faculty`";
10.3 Выполнить запрос к базе данных:
$f=mysql_query($f_query);
10.4 Создать строковую переменную facOptions для хранения XML-представления информации о факультетах. Записать в нее открывающий тег корневого элемента faculties
10.5 Сформировать цикл while по всем строкам результирующего набора f:
while ($fac = mysql_fetch_array($f))
{
}
10.6 Сформировать XML-элемент faculty, соответствующий одной записи из таблицы Faculty. В элементе задать атрибуты ID и name, которые соответствуют одноименным полям в таблице базы данных. Для этого между открывающей и закрывающей фигурными скобками цикла ввести следующий код:
$facOptions = $facOptions."<faculty ID='".$fac['ID']."' name='".$fac['name']."'/>";
10.7 В качестве значения, возвращаемого функцией faculty, указать текстовую переменную facOptions, дополненную закрывающим тегом элемента faculties:
return $facOptions.'</faculties>';
}
11. Аналогичным образом (см. п. 9) зарегистрировать метод spec, формирующий XML-список специальностей.
12. Определить метод spec как функцию PHP (см. п. 10).
13. Зарегистрировать метод gr, формирующий XML-список групп и определить его как функцию PHP.
14. Зарегистрировать метод usp (успеваемость) с входным параметром name (номер группы). Для этого ввести следующий код:
$server->register(
'usp',
array('group' => 'xsd:string'),
array('return' => 'xsd:string'),
'uri:uspquery',
'uri:uspquery/usp',
'rpc',
'encoded'
);

15. Определить метод usp как функцию PHP:

```
function usp($group){
$usp_query="SELECT `fio`, `subject`, `ocenka`, `data`
FROM `uspev`, `student`
WHERE `uspev`.`student`= `student`.`zk` AND `gr`=".$group;
$usp=mysql_query($usp_query);
$usp_count = mysql_num_rows($usp);
$usOptions = '<uspev>';
while ( $u = mysql_fetch_array( $usp ) )
{
$usOptions = $usOptions."<usp fio='".$u['fio']."' subject='".$u['subject']."' ocenka='".
$u['ocenka']."' data='".$u['data']."'/>";
}
return $usOptions.'</uspev>';
}
```

16. Вызвать сервис с помощью HTTP-запроса

```
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ?
$HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
```

17. Ввести закрывающий тег сценария?>.

2. Использование Web-сервиса

1. В папку Lab4_PHP скопировать файл nuSOAP (из предыдущего пункта) и файлы Index. php и Browse. php из первой лабораторной работы.

2. Открыть в любом редакторе файл Index. php.

3. Ввести открывающий тег сценария <?php

4. Подключить код сценария nusoap. php:

```
require_once('nusoap. php');
```

5. Создать экземпляр SOAP-клиента.

5.1 Создать переменную client

5.2 Инициализировать конструктор класса soapclient и передать ему в качестве параметра URL SOAP-сервера (из предыдущего пункта):

```
$client = new soapclient('http://localhost/denwer/Lab4_PHP/server/index. php');
```

6. Вызвать SOAP-метод faculty:

```
$faculty = $client->call('faculty');
```

7. Аналогичным образом вызвать SOAP-методы spec и gr:

```
$spec = $client->call('spec');
$gr = $client->call('gr');
```

8. Записать результаты в экземпляры DOM-объектов

8.1 Создать экземпляр DOM-объекта и записать его в переменную fac_dom:

```
$fac_dom = new DOMDocument();
```

8.2 Задать кириллическую кодировку DOM-объекта:

```
$fac_dom -> encoding = "windows-1251";
```

8.3 Загрузить в DOM-объект XML-данные о факультетах, полученные в результате вызова SOAP-метода faculty:

```
$fac_dom->loadXml(iconv("windows-1251", "utf-8", $faculty));
```

8.4 Аналогичным образом создать DOM-объекты spec_dom (для XML-данных о специальностях) и gr_dom (для XML-данных о группах).

9. Ввести закрывающий тег сценария?>

10. В раскрывающийся список факультетов (HTML-элемент select с атрибутом name = "faculty") добавить данные из XML-списка факультетов. Для этого после строки

```
<option value="0">Выберите факультет</option>
```

10.1 Ввести открывающий тег сценария:

<?php
10.2 Построить цикл по элементам DOM-объекта fac_dom:
foreach ($fac_dom->documentElement->childNodes as $item) {
10.3 Создать строковую переменную value и занести в нее значение XML-атрибута ID:
$value = iconv("utf-8", "windows-1251", $item->getAttribute('ID'));
10.4 Создать строковую переменную text и занести в нее значение XML-атрибута name:
$text = iconv("utf-8", "windows-1251", $item->getAttribute('name'));
10.5 Вывести HTML-код элемента списка option со значением из переменной value и отображаемым текстом из переменной text.
echo "<option value='".$value."'>".$text."</option>";
}
10.6. Ввести закрывающий тег сценария?>
11. Аналогичным образом заполнить раскрывающиеся списки специальностей (из DOM-объекта spec_dom) и групп (из DOM-объекта gr_dom).
12. Открыть в любом редакторе файл Browse. php.
13. Подключить код сценария nusoap. php:
require_once('nusoap. php');
14. Создать строковую переменную gr и занести в нее значение параметра gr, переданного методом POST со страницы Index. php:
$gr = $_POST['gr'];
15. Создать экземпляр SOAP-клиента (см. п. 5) client.
16. Вызвать метод usp и в качестве входного параметра передать ему значение строковой переменной gr:
$result = $client->call('usp', array('group' => $gr));
17. Записать результаты в экземпляр DOM-объекта usp_dom:
$usp_dom = new DOMDocument();
$usp_dom -> encoding = "windows-1251";
$usp_dom->loadXml(iconv("windows-1251", "utf-8", $result));
18. Заполнить таблицу TABLE со стилем textborder данными из XML-набора записей об успеваемости студентов
18.1 Ввести открывающий тег сценария:
<?php
18.2 Построить цикл по элементам DOM-объекта usp_dom:
foreach ($usp_dom->documentElement->childNodes as $item)
{
18.3 Занести в переменные fio, subject, ocenka и data значения одноименных XML-атрибутов:
$fio= iconv("utf-8", "windows-1251", $item->getAttribute('fio'));
$subject = iconv("utf-8", "windows-1251", $item->getAttribute('subject'));
$ocenka = iconv("utf-8", "windows-1251", $item->getAttribute('ocenka'));
$data = iconv("utf-8", "windows-1251", $item->getAttribute('data'));
18.4 Ввести закрывающий тег сценария?>
18.5 Ввести открывающий тег элемента TR:
<TR>
18.6. Ввести открывающий тег первого столбца (шириной 30% от общей ширины таблицы). Задать стиль представления текста в столбце tabletext:
<TD width="30%"><SPAN class="tabletext">
18.7 Вывести значение переменной fio
<?php
echo $fio;

?>

18.8 Закрыть элементы SPAN и TD:

</SPAN></TD>

18.9 Аналогичным образом построить столбцы для представления данных из переменных subject, data и ocenka соответственно.

19. Протестировать полученное Web-приложение.

Содержание отчета

Отчет должен содержать:

– титульный лист, название и цель работы;

– листинг программного кода;

– скриншоты результатов работы Web-приложения с различными вариантами запросов;

– выводы по работе.

Контрольные вопросы

1. Что такое веб-сервис?
2. Из чего состоит веб-сервис?
3. Для чего используются SOAP-сообщения?
4. Что такое UDDI?
5. Как описать свойства и методы веб-сервиса?

Лабораторная работа №28

Разработка меню web-страницы на HTML5+CSS3+PHP

Цель: научиться разрабатывать меню средствами HTML5+CSS3+PHP

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Разместить на странице меню с набором гиперссылок – одна из важных задач в практике веб-дизайнера. Чтобы не утомлять внимание пользователя просмотром всего набора, ссылки обычно группируют по категориям, и в основное меню включают только категории. Только при выборе категории должны отобразиться соответствующие ссылки (или подкатегории).

Как правило, такие динамические меню принято создавать средствами языка Javascript, позволяющего совершать любой сложности манипуляции с элементами веб-страницы. Однако, существует и решение на CSS – довольно простое и красивое, хотя и не любую фантазию дизайнера выполняющее.

Практическая часть

Задание 1. Разработать навигационное меню на странице, типа Contact используя массив в качестве структуры меню.

Создайте ассоциативный массив $menu

Заполните массив соблюдая следующие условия:

Название ячейки является пунктом меню, например: Home, About, Contact...

Значение ячейки является именем файла, на который будет указывать ссылка, например: index.php, about.php, contact.html...

Задание 2. Используя цикл foreach отрисуйте вертикальное меню, структура которого описана в массиве $menu

Решение:

```php
<?php
$menu = array(
        "Home"=>"index.php",
        "Contact"=>"contact.php",
        "About"=>"about.php",
        "Project"=>"project.php",
        "Map"=>"map.php"
        );
?>
<ul style="list-style-type:none">
    <?php
    foreach ($menu as $link=>$href){
        echo "<li><a href=\"$href\">", $link, '</a></li>';
    }
    ?>
</ul>
```

Задание 3. Создайте следующее меню.

Для создания меню будем описывать стили внутри html-документа:

```
<html>
<head>
<style>
ul.menu1 {
list-style: none;
border-bottom: 1px #888899 solid;
padding-bottom: 10px
}
ul.menu1 li {
display: inline;
margin-right: 5px
}
ul.menu1 li a {
color: #888899;
text-decoration: none;
background: #f7f7f9;
border: 1px #bbbbcc solid;
border-bottom: none;
padding: 10px 14px
}
ul.menu1 li a:hover {
padding: 14px 14px 10px 14px
}
ul.menu1 li a.selected {
color: #555566;
background: #ffffff;
border: 1px #888899 solid;
border-bottom: 1px #ffffff solid;
padding: 14px 14px 10px 14px
}
</style>
<title>Создание меню 1 для сайта CSS методами</title>
</head>
<body>
<ul class="menu1">
<li><a href="#">Главная</a></li>
<li><a href="#">Пункт1 </a></li>
<li><a href="#" class="selected">Пункт2</a></li>
<li><a href="#">Пункт3</a></li>
</ul>
</html>
```

Назначив конкретные ссылки, сделайте меню рабочим. Измените цвет меню, поэкспериментируйте, применяя псевдо класс **selected**к разным пунктам списка меню.

В результате создайте следующее меню, например:

Задание 4. Создайте следующее меню.

```
<html>
<head>
<style>
#menu2 ul {
list-style: none;
font-family: Georgia, serif;
font-size: 18px;
font-style: italic;
line-height: 1.4em;
border:2px solid #000000;
border-left: 1px solid #000000;
float: left;
padding: 0;
margin: 12px 0 25px 24px
}
#menu2 ul li {
float: left
}
#menu2 ul li a {
display: block;
text-decoration: none;
background-color: #595959;
padding: 5px 10px 0 10px;
color: #fefefe;
width: 120px;
border-right: 1px solid #797979;
border-left: 1px solid #191919
}
#menu2 ul li a span {
display: block
}
#menu2 ul li a span.text-top {
border-bottom: 1px solid #595959
}
#menu2 ul li a:hover span.text-top {
border-bottom: 1px dashed #fefefe;
color: #ffddbb
}
#menu2 ul li a span.text-bottom {
visibility: hidden;
font-size: 11px;
text-align: right
}
#menu2 ul li a:hover span.text-bottom {
visibility: visible
}
</style>
<title>Создание красивого горизонтального CSS меню для сайта</title>
</head>
<body>
<div id="menu2">
<ul>
<li><a href="#">
<span class="text-top">Главная</span>
<span class="text-bottom">Я подпись</span>
</a></li>
<li><a href="#">
<span class="text-top">Пункт1</span>
<span class="text-bottom">Подпись</span>
</a></li>
<li><a href="#">
<span class="text-top">Пункт2</span>
<span class="text-bottom">Подпись</span>
</a></li>
<li><a href="#">
<span class="text-top">Пункт3</span>
<span class="text-bottom">Подпись</span>
</a></li>
</ul>
</div>
</html>
```

Назначив конкретные ссылки, сделайте меню рабочим. Измените цвет меню, сместите подпись влево и измените используемый для нее шрифт.

Задание 5. Создать развернутое горизонтальное меню с выпадающими списками.

1. Создайте в теле документа набор категорий в виде неупорядоченного списка (ul), каждый элемент которого содержит список ссылок:.

```
<ul id="MainMenu">
<li><a href="#" title="Популярные блюда">Популярные блюда</a>
<ul>
<li><a href="/Italian.htm">Итальянская кухня</a></li>
<li><a href="/Snack.htm">Закуски</a></li>
<li><a href="/Breakfast.htm">Закуски</a></li>
<li><a href="/Sweets.htm">Сласти</a></li>
<li><a href="/Fruits.htm">Фрукты</a></li>
</ul>
.....
</ul>
```

2. Названия ссылок и категорий придумайте самостоятельно. Важно, чтобы атрибут id внешнего списка имел значение MainMenu- далее ему будет назначен особый стиль по этому идентификатору).

С помощью стиля расположим пункты меню горизонтально за счёт обтекания. В таблице стилей добавьте следующее правило:

```
#MainMenu > li {
float: left;
list-style-type: none;
}
```

3. Пусть вложенные списки ссылок будут невидимыми (добавьте им свойство display:none;) и появляются только при наведении курсора на название соответствующей категории. Следующее правило с селектором псевдокласса hover имеет такой смысл: у списка (ul), вложенного в пункт списка (li), на который наведён указатель (:hover) и который вложен в элемент с id=#MainMenu, способ отображения следует сделать блочным (а не невидимым):

```
#MainMenu li:hover ul {
display:block;
}
```

4. Сохраните разрабатываемый документ, откройте его в браузере и убедитесь, что меню работает правильно.

Назначьте якорям любого уровня вложенности в меню (правило #MainMenu a) желаемый цвет (color), гарнитуру (font), а также уберите подчёркивание (text-decoration).

Назначьте элементам списка категорий (правило #MainMenu > li) фоновый цвет (background), внутренний отступ (padding) и рамку справа (border-right).

Назначьте элементам вложенного списка ссылок (правило #MainMenu li li) такой же фоновый цвет, как и в списке категорий, а также небольшой отступ и рамку снизу. Кроме того, уберите маркировку списка (list-style-type).

Уберите у списка ссылок (правило #MainMenu li ul) поля и отступы (marginи padding).

Небольшой проблемой является то, что некоторые элементы списка категорий увеличиваются в ширину при наведении на них указателя. Это является следствием того, что ширина элемента списка определяется шириной всего содержимого - включая вложенный список. Однако если выбросить вложенный список из нормального потока и позиционировать его абсолютно, то его ширина более не будет влиять на ширину родительского элемента. Поэтому укажите для списка ссылок абсолютное позиционирование, а для элементов списка категорий - относительное. Проверьте в браузере. Затем уточните положение вложенного списка относительно его контейнера путём задания значения свойств left и top (не опускайте выпадающий список слишком низко, иначе он станет пропадать при попытке выбрать в нём ссылку).

Добавьте последний штрих: пусть элементы обоих списков при наведении указателя немного изменяют цвет фона (правило #MainMenu li:hover).

Задание 6. Создайте выпадающее горизонтальное меню с полем для поиска с помощью CSS3 и HTML5.

HTML часть

Навигации состоит из маркированного списка (с классом .nav), который содержит внутри:

- Элементы списка, которые содержат обычные ссылки и не имеют никакого идентификатора или класса;
- #settings внутри которого изображения как ссылки;
- #search содержит поле для поиска и кнопку отправки текста;
- #options содержит ссылку и список (с классом .subnav), который работает как выпадающее меню.

```
<ul class="nav">
  <li id="settings">
    <a href="#"><img src="images/settings.png"></a>
  </li>
  <li>
    <a href="#">Приложения</a>
  </li>
  <li>
    <a href="#">Сервис</a>
  </li>
  <li id="search">
    <form action="" method="get">
      <input name="search_text" id="search_text" placeholder="Поиск..." type="text">
      <input name="search_button" id="search_button" type="button">
    </form>
  </li>
  <li id="options">
    <a href="#">Меню</a>
    <ul class="subnav">
      <li><a href="#">Настройки</a></li>
      <li><a href="#">Приложения</a></li>
      <li><a href="#">Сервис</a></li>
      <li><a href="#">Разное</a></li>
    </ul>
  </li>
</ul>
```

CSS часть

Базовые стили:

```
* {
    margin: 0;
    padding: 0;
}
```

Навигация и вид списка

```
.nav {
    background: #232323;
    height: 60px;
    display: inline-block;
}

.nav li {
    float: left;
    list-style-type: none;
    position: relative;
}
```

Здесь задали фон для меню, а также расположили горизонтально элементы списка.

Ссылки в меню

```
.nav li a {
    font-size: 16px;
    color: white;
    display: block;
    line-height: 60px;
    padding: 0 26px;
    text-decoration: none;
    border-left: 1px solid #2e2e2e;
    font-family: sans-serif, Arial;
    text-shadow: 0 0 1px rgba(255, 255, 255, 0.5);
}
.nav li a:hover {
    background-color: #2e2e2e;
}

#settings a {
    padding: 18px;
    height: 24px;
    font-size: 10px;
    line-height: 24px;
}
```

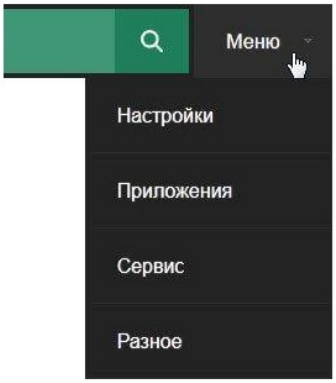
Ссылки имеют высоту блока, то есть весь блок будет ссылкой. При наведении на ссылку фон меняется на серый. Блок с идентификатором **#settings** содержит изображение, поэтому для него задали такие свойства.

Поле поиска

```css
#search {
    width: 357px;
    margin: 4px;
}
#search_text {
    width: 297px;
    padding: 15px 0 15px 20px;
    font-size: 16px;
    font-family: sans-serif, Arial;
    border: 0 none;
    height: 52px;
    margin-right: 0;
    color: white;
    outline: none;
    background: #1f7f5c;
    float: left;
    box-sizing: border-box;
    transition: all 0.15s;
}
::-webkit-input-placeholder { /* WebKit браузеры */
    color: white;
}
:-moz-placeholder { /* Mozilla Firefox 4 to 18 */
    color: white;
}
::-moz-placeholder { /* Mozilla Firefox 19+ */
    color: white;
}
:-ms-input-placeholder { /* Internet Explorer 10+ */
    color: white;
}
#search_text:focus {
    background: rgb(64, 151, 119);
}
#search_button {
    border: 0 none;
    background: #1f7f5c url("../images/search.png") center no-repeat;
    width: 60px;
    float: left;
    padding: 0;
    text-align: center;
    height: 52px;
    cursor: pointer;
}
```

Здесь идет оформление поля с поиском. Заданы разные свойства, при наведении, при фокусе и т.д.

Выпадающая часть



```css
#options a {
    border-left: 0 none;
}
#options>a {
    background-image: url("../images/triangle.png");
    background-position: 85% center;
    background-repeat: no-repeat;
    padding-right: 42px;
}
.subnav {
    visibility: hidden;
    position: absolute;
    top: 110%;
    right: 0;
    width: 200px;
    height: auto;
    opacity: 0;
    transition: all 0.1s;
    background: #232323;
}
.subnav li {
    float: none;
}
.subnav li a {
    border-bottom: 1px solid #2e2e2e;
}
#options:hover .subnav {
    visibility: visible;
    top: 100%;
    opacity: 1;
}
```

Блок с классом **.subnav** изначально спрятан, и появляется он только при наведении. Если есть какие-то вопросы, задавайте их в комментариях ниже.

Содержание отчета

1. Цель
2. Ход работы с выдержками из сценариев.
3. Выводы

Контрольные вопросы

1. Способы создания навигационного меню
2. Использование свойств CSS3

Лабораторная работа №29

Использование JavaScript для доступа и управления HTML DOM объектов

Цель: закрепить и расширить практические знания по программированию на языке javascript. Получить представление об практическом использовании объектной модели веб-документа (DOM) и использовании веб-форм

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

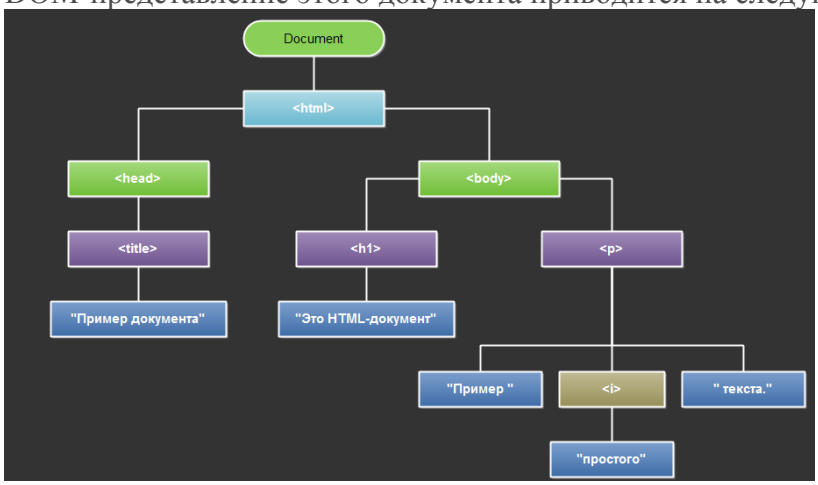
Теоретический материал:

Объектная модель документа (Document Object Model, DOM) - это фундаментальный прикладной программный интерфейс, обеспечивающий возможность работы с содержимым HTML и XML-документов. Прикладной программный интерфейс (API) модели DOM не особенно сложен, но в нем существует множество архитектурных особенностей, которые вы должны знать.

Прежде всего, следует понимать, что вложенные элементы HTML или XML-документов представлены в виде дерева объектов DOM. Древовидное представление HTML-документа содержит узлы, представляющие элементы или теги, такие как <body> и <p>, и узлы, представляющие строки текста. HTML-документ также может содержать узлы, представляющие HTML-комментарии. Рассмотрим следующий простой HTML-документ:

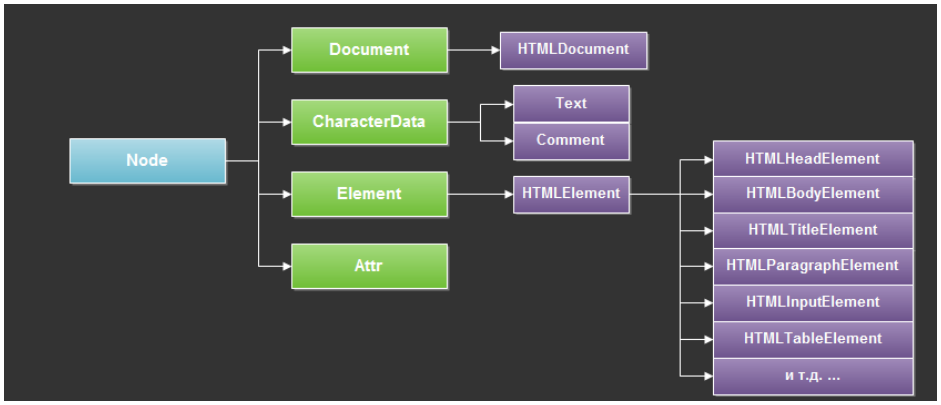
```
<html>
  <head>
    <title>Пример документа</title>
  </head>
  <body>
    <h1>Это HTML-документ</h1>
    <p>Пример <i>простого</i> текста.</p>
  </body>
</html>
```

DOM-представление этого документа приводится на следующей диаграмме:



Каждый прямоугольник на этой диаграмме является узлом документа, который представлен объектом Node. Обратите внимание, что на рисунке изображено три различных типа узлов. Корнем дерева является узел Document, который представляет документ целиком. Узлы, представляющие HTML-элементы, являются узлами типа Element, а узлы, представляющие текст, - узлами типа Text. Document, Element и Text - это подклассы класса Node. Document и Element являются двумя самыми важными классами в модели DOM.

Тип Node и его подтипы образуют иерархию типов, изображенную на диаграмме ниже. Обратите внимание на формальные отличия между обобщенными типами Document и Element, и типами HTMLDocument и HTMLElement. Тип Document представляет HTML и XML-документ, а класс Element представляет элемент этого документа. Подклассы HTMLDocument и HTMLElement представляют конкретно HTML-документ и его элементы:



На этой диаграмме следует также отметить наличие большого количества подтипов класса HTMLElement, представляющих конкретные типы HTML-элементов. Каждый из них определяет JavaScript-свойства, отражающие HTML-атрибуты конкретного элемента или группы элементов. Некоторые из этих специфических классов определяют дополнительные свойства или методы, которые не являются отражением синтаксиса языка разметки HTML.

Выбор элементов документа

Работа большинства клиентских программ на языке JavaScript так или иначе связана с манипулированием элементами документа. В ходе выполнения эти программы могут использовать глобальную переменную document, ссылающуюся на объект Document. Однако, чтобы выполнить какие-либо манипуляции с элементами документа, программа должна каким-то образом получить, или выбрать, объекты Element, ссылающиеся на эти элементы документа. Модель DOM определяет несколько способов выборки элементов. Выбрать элемент или элементы документа можно:

- по значению атрибута id;
- по значению атрибута name;
- по имени тега;
- по имени класса или классов CSS;
- по совпадению с определенным селектором CSS.

Все эти приемы выборки элементов описываются в следующих подразделах.

Выбор элементов по значению атрибута id

Все HTML-элементы имеют атрибуты id. Значение этого атрибута должно быть уникальным в пределах документа - никакие два элемента в одном и том же документе не должны иметь одинаковые значения атрибута id. Выбрать элемент по уникальному значению атрибута id можно с помощью метода getElementById() объекта Document:

```
var section1 = document.getElementById("section1");
```

Выбор элементов по значению атрибута name

Выбрать HTML-элементы, опираясь на значения их атрибутов name, можно с помощью метода getElementsByName() объекта Document:

```
var radiobuttons = document.getElementsByName("favorite_color");
```

Метод getElementsByName() определяется не классом Document, а классом HTMLDocument, поэтому он доступен только в HTML-документах и не доступен в XML-документах. Он возвращает объект NodeList, который ведет себя, как доступный только для чтения массив объектов Element.

Выбор элементов по типу

Метод getElementsByTagName() объекта Document позволяет выбрать все HTML или XML-элементы указанного типа (или по имени тега). Например, получить подобный массиву объект, доступный только для чтения, содержащий объекты Element всех элементов в документе, можно следующим образом:

```
var spans = document.getElementsByTagName("span");
```

Подобно методу getElementsByName(), getElementsByTagName() возвращает объект NodeList. Элементы документа включаются в массив NodeList в том же порядке, в каком они следуют в документе, т.е. первый элемент <p> в документе можно выбрать так:

```
var firstParagraph = document.getElementsByTagName("p")[0];
```

Кроме того, классом Element также определяет метод getElementsByTagName(). Он действует точно так же, как и версия метода в классе Document, но выбирает только элементы, являющиеся потомками для элемента, относительно которого вызывается метод. То есть отыскать все элементы внутри первого элемента <p> можно следующим образом:

```
var firstParagraph = document.getElementsByTagName("p")[0];
var firstParagraphSpans = firstParagraph.getElementsByTagName("span");
```

Выбор элементов по классу CSS

Значением HTML-атрибута class является список из нуля или более идентификаторов, разделенных пробелами. Он дает возможность определять множества связанных элементов документа: любые элементы, имеющие в атрибуте class один и тот же идентификатор, являются частью одного множества. Слово class зарезервировано в языке JavaScript, поэтому для хранения значения HTML-атрибута class в клиентском JavaScript используется свойство className.

Обычно атрибут class используется вместе с каскадными таблицами стилей CSS, с целью применить общий стиль отображения ко всем членам множества. Однако кроме этого, стандарт HTML5 определяет метод getElementsByClassName(), позволяющий выбирать множества элементов документа на основе идентификаторов в их атрибутах class.

Подобно методу getElementsByTagName(), метод getElementsByClassName() может вызываться и для HTML-документов, и для HTML-элементов, и возвращает «живой» объект NodeList, содержащий все потомки документа или элемента, соответствующие критерию поиска.

Метод getElementsByClassName() принимает единственный строковый аргумент, но в самой строке может быть указано несколько идентификаторов, разделенных пробелами. Соответствующими будут считаться все элементы, атрибуты class которых содержат все указанные идентификаторы. Порядок следования идентификаторов не имеет значения. Обратите внимание, что и в атрибуте class, и в аргументе метода getElementsByClassName() идентификаторы классов разделяются пробелами, а не запятыми.

Ниже приводится несколько примеров использования метода getElementsByClassName():

```
// Отыскать все элементы с классом "warning"
var warnings = document.getElementsByClassName("warning");

// Отыскать всех потомков элемента с идентификаторам "log"
// с классами "error" и "fatal"
var log = document.getElementById("log");
var fatal = log.getElementsByClassName("fatal error");
```

Выбор элементов с использованием селекторов CSS

Каскадные таблицы стилей CSS имеют очень мощные синтаксические конструкции, известные как селекторы, позволяющие описывать элементы или множества элементов документа. Наряду со стандартизацией селекторов CSS3, другой стандарт

консорциума W3C, известный как Selectors API, определяет методы JavaScript для получения элементов, соответствующих указанному селектору.

Ключевым в этом API является метод querySelectorAll() объекта Document. Он принимает единственный строковый аргумент с селектором CSS и возвращает объект NodeList, представляющий все элементы документа, соответствующие селектору.

Структура документа и навигация по документу

После выбора элемента документа иногда бывает необходимо отыскать структурно связанные части документа (родитель, братья, дочерний элемент). Объект Document можно представить как дерево объектов Node. Тип Node определяет свойства, позволяющие перемещаться по такому дереву. Существует еще один прикладной интерфейс навигации по документу, как дерева объектов Element.

Документы как деревья узлов

Объект Document, его объекты Element и объекты Text, представляющие текстовые фрагменты в документе - все они являются объектами Node. Класс Node определяет следующие важные свойства:

parentNode

Родительский узел данного узла или null для узлов, не имеющих родителя, таких как Document.

childNodes

Доступный для чтения объект, подобный массиву (NodeList), обеспечивающий представление дочерних узлов.

firstChild, lastChild

Первый и последний дочерние узлы или null, если данный узел не имеет дочерних узлов.

nextSibling, previousSibling

Следующий и предыдущий братские узлы. Братскими называются два узла, имеющие одного и того же родителя. Порядок их следования соответствует порядку следования в документе. Эти свойства связывают узлы в двусвязный список.

nodeType

Тип данного узла. Узлы типа Document имеют значение 9 в этом свойстве. Узлы типа Element - значение 1. Текстовые узлы типа Text - значение 3. Узлы типа Comments - значение 8 и узлы типа DocumentFragment - значение 11.

nodeValue

Текстовое содержимое узлов Text и Comment.

nodeName

Имя тега элемента Element, в котором все символы преобразованы в верхний регистр.

С помощью этих свойств класса Node можно сослаться на второй дочерний узел первого дочернего узла объекта Document, как показано ниже:

```
document.childNodes[0].childNodes[1] ==
  document.firstChild.firstChild.nextSibling
```

Задание

Написать скрипт, проверяющий код защиты от автоматического постинга и вырезающий ссылки из формы ввода комментария (на странице отзывов и комментариев)

CAPTCHA (от англ. «Completely Automated Public Turing test to tell Computers and Humans Apart», рус., сленг. - КАПЧА) — полностью автоматизированный публичный тест Тьюринга для различия компьютеров и людей) компьютерный тест, используемый для того, чтобы определить, кем является пользователь системы: человеком или компьютером. Основная идея теста: предложить пользователю такую задачу, которую может решить человек, но которую несоизмеримо сложно предоставить для решения компьютеру. В основном это задачи на распознавание символов. CAPTCHA чаще всего используется при необходимости предотвратить использование интернет-сервисов

ботами, в частности, для предотвращения автоматических отправок сообщений, регистрации, скачивания файлов, массовых рассылок и т. п.

Пояснение: В ходе выполнения задания требуется написать клиентсткую программу на javascript, которая генерирует арифметический пример, ответ на который должен дать пользователь. Другой вариант — генерация произвольной строки, которую должен воспроизвести пользователь. После того, как пользователь ввел ответ программа должна проверить его правильность. Смысл этого задания не столько в разработке эффективного теста Тьюринга, сколько в освоении javascript.

Указания к работе

Элементы управления

Элементы управления — это интерактивные объекты, позволяющие получить данные от пользователя. Их назначение и внешний вид идентичны элементам пользовательского интерфейса современных операционных систем с графическим интерфейсом (кнопки, поля ввода, чекбоксы и т.п.).

Элемент input

Тэг <input> представляет различные элементы, в зависимости от значения атрибута type (табл.1).

Таблица 1. Типы элементов управления (атрибут type)

Значение type	**Описание**
text	Однострочное поле ввода. Используйте атрибуты maxlength и size для определения максимальной длинны вводимого значения в символах и размера отображаемого поля ввода на экране (по умолчанию принимается 20 символов).
password	То же самое, что и атрибут text, но вводимое пользователем значение скрыто замещающими символами (звездочки, точки и т.п.).
checkbox	Флажок (маркер множественного выбора). Используется для отметки выбранных вариантов.
hidden	Скрытое поле. Не отображается браузером и не дает пользователю изменять присвоенные данному полю значение. Это можно сделать только программным путем (или изменением значения поля при передаче данных через адресную строку или в теле запроса).
image	Кнопка-картинка. Позволяет использовать графический рисунок в качестве кнопки. Все значения атрибута value игнорируются. Само описание картинки осуществляется через атрибут src и по синтаксису совпадает с тегом <img>.
radio	Радиокнопка. Позволяет вводить одно значение из нескольких альтернатив. Для создания набора альтернатив вам необходимо создать несколько полей ввода с атрибутом type="radio" с разными значениями атрибута value, но с одинаковыми значениями атрибута name. При выборе одного из полей ввода типа radio все остальные поля данного типа с тем же именем (атрибут name) автоматически станут невыбранными на экране.
button	Пользовательская кнопка. Должна быть запрограммирована на обработку нажатий. Атрибут value содержит текст надписи на кнопке.
submit	Кнопка отправки данных. При ее нажатии будет будет вызван обработчик, описанный в заголовке формы (form action="scriptname", подробнее о теге form - в лабораторной работе №8) и ему будут переданы значения всех элементов, описанных в теге form. Атрибут value содержит текст надписи на кнопке.
reset	Кнопка сброса. При нажатии ее все поля формы примут значения, заданные по

Значение type	Описание
	умолчанию.

Атрибуты элемента input

- type — определяет тип поля ввода. По умолчанию равно text.
- name — имя поля ввода. Используется как идентификатор переменной при передаче данных на сервер и для программного обращения к элементу из скрипта javascript.
- id — идентификатор элемента. Должен быть уникальным в пределах веб-документа.
- checked — означает, что checkbox или radio будет выбран.
- maxlength — определяет количество символов, которое пользователи могут ввести в поле ввода. При превышении количества допустимых символов броузер реагирует на попытку ввода нового символа звуковым сигналом и не дает его ввести.
- size — определяет визуальный размер поля ввода на экране в символах.
- src — URL, указывающий на картинку (используется совместно со значением type="image").
- value — значение по умолчанию или установленное значение.

Элемент textarea

Тэг <textarea> используется для того, чтобы позволить пользователю вводить более одной строки информации (многострочный текст). При передаче значения из textarea сохраняются все символы форматирования (табуляция, перевод строки, возврат каретки).

Атрибуты, используемые с тегом <textarea> задают его размеры (в символах и строках):

- rows — высота поля ввода в символах
- cols — ширина поля ввода в строках

Пример использования тега <textarea>:

<textarea rows=10 cols=50>Москва, Дмитровское шоссе, д.9Б, офис 448</textarea>

Элемент select

Элемент select отображает на странице список выбора, который может быть представлен следующими способами:

- select — выпадающий список.
- select single — развернутый список.
- select multiple — список с множественным выбором.

Примеры описания элемента select:

```
<select single name="group" size="3">
<option>зима</option>
<option>весна</option>
<option>лето</option>
<option>осень</option>
</select>
```

Объект document

Объект document это абстрактная структура данных, представляющая полное описание веб-страницы. Набор свойств и методов этого объекта позволяет управлять как поведением веб-страницы целиком, так и отдельных ее объектов (элементов управления, ссылок, текстовых блоков, изображений и т.д.). Доступ к свойствам и методам реализован через стандартные программные интерфейсы (рис. 1).

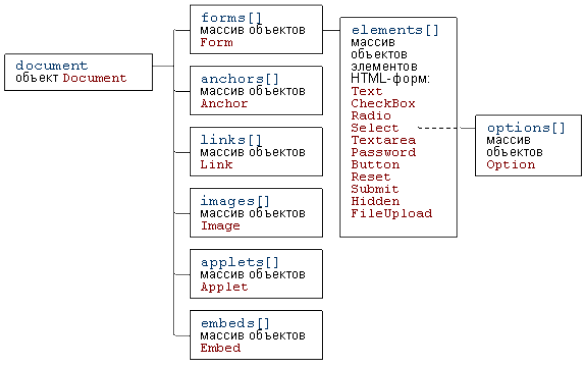


Рис. 1. Интерфейсы объекта document

Свойства объекта Document

Начнем со свойств, общих для всех браузеров. Большинство их доступны как для чтения, так и для изменения. Все значения свойств — строковые.

- title — текст заголовка документа (содержимое элемента title);
- fgColor и bgColor — цвет текста и цвет фона документа;
- linkColor, vLinkColor, aLinkColor — цвета непосещенных, посещенных и активных гиперссылок;
- lastModified (только для чтения) — дата изменения документа;
- referrer (только для чтения) — адрес источника перехода;
- URL, location — собственный адрес документа.

Более интересны и полезны для разработчика свойства-массивы объекта Document. Все они, естественно, имеют свойство length (количество элементов в массиве). Большинство свойств, специфичных для объектов, хранящихся в этих массивах, ассоциируются с атрибутами соответствующих элементов HTML (список неполный):

- объект Anchor (якорь) имеет единственное свойство name;
- объект Link (ссылка) имеет свойства href, target;
- объект Image (изображение) имеет свойства src, width, height.

К объектам документа, хранящимся в массивах images, controls и прочим, а также к элементам форм можно обращаться по имени (свойство name) или идентификатору (свойство id). Пусть, например, в документе имеется описание <**img src**="images/cat.jpg" **id**="cat_id" **name**="cat_name"> и оно является n-ым изображением, встречающимся в документе. К этому элементу img можно обратиться по крайней мере следующими способами:

1. Как к элементу массива images по индексу (индексация начинается с 0): window.document.images[n-1]

2. Как к элементу хэш-массива images по ключу (значение name как ключ массива): window.document.images["cat_name"]

3. Используя значение атрибута name как свойство объекта: window.document.cat_name

4. Используя значение атрибута id и свойство getElementById: window.document.getElementById("cat_id")

Методы объекта Document

- open() — открывает новый документ; при этом все его содержимое удаляется.
- close() — закрывает ранее открытый документ.
- write() — записывает в документ заданную в качестве аргумента строку.
- writeln() — аналогичен предыдущему, но выведенная в документ строка заканчивается символом перевода строки.

Методы write() и writeln() весьма полезны и часто используются для динамического формирования содержимого документа. Вот как, например, можно включить в документ дату его последнего изменения:

<script>document.write(document.lastModified);</script>

События

Для всех элементов документа имеется возможность отслеживать различные события (загрузка, перемещение мыши, мышеклики и проч.) и вызывать функции обработки таких событий. В таблице 2 приведено краткое описание событий, доступных для использования в программах на javascript:

Таблица 2. События веб-документа

Событие	**Описание**
OnLoad	Броузер заканчивает открытие документа HTML
OnUnload	Броузер выгружает документ HTML
OnClick	Пользователь щелкнул мышью по элементу
OnDblClick	Пользователь дважды щелкнул мышью по элементу
OnMouseDown	Пользователь нажимает кнопку мыши
OnMouseOver	Пользователь перемещает мышь поверх элемента
OnMouseMove	Пользователь перемещает мышь поверх элемента
OnMouseOut	Пользователь перемещает мышь, выходя из элемента
OnFocus	Элемент получает фокус ввода
OnBlur	Элемент теряет фокус ввода
OnKeyPress	Пользователь нажимает и отпускает клавишу
OnKeyDown	Пользователь нажимает клавишу над элементом
OnKeyUp	Пользователь отпускает клавишу над элементом
OnSubmit	Данные из формы переданы Web-серверу
OnReset	Форма очищена
OnSelect	Пользователь выбирает текст в текстовом поле
OnChange	Потеря фокуса ввода элементом после изменения его значения

Назначение обработчика события выполняется путем указания имени события в виде атрибута тега, например так:

say "Hello"

При работе с веб-страницами часто возникает необходимость выполнить сложную обработку текста. В javascript для этого имеется встроенный объект RegExp, который позволяет работать с регулярными выражениями.

Работа с объектном RegExp в javascript мало отличается от работы с любыми другими объектами, но сам синтаксис регулярных выражений требует понимания и практики. Хорошая статья по этой теме написана М.С.Выскорко, она приводится здесь в качестве руководства по регулярным выражениям в javascript.

Примеры скриптов

В листингах 1-5 приведены примеры простых скриптов, иллюстрирующими базовые возможности javascript при работе с объектами веб-документа. При выполнении заданий используйте предлагаемые примеры в качестве образцов.

Листинг 1. Ограничение количества символов

```
<html>
<head>
<title>Ограничение количества вводимых символов</title>
<script type="text/javascript">
var maxLen = 25;
function checkMaxinput(form) {
   if (form.message.value.length > maxLen)
      form.message.value = form.message.value.substring(0, maxLen);
   else
      form.remLen.value = maxLen —
      form.message.value.length;
      }
</script>
</head>
<body>
<form name=myform action="somehandler.cgi">
<h1>Ограничение количества вводимых символов<h1>
<textarea name=message cols=28 rows=4 onKeyDown="checkMaxinput(this.form)"
   onKeyUp="checkMaxinput(this.form)"></textarea>
<p>Осталось <input readonly type=text name=remLen size=3 value="25"> символов</p>
</form>
</body>
</html>
```

Листинг 2. Проверка ввода

```
<html>
<head>
<title>Проверка ввода
</title>
<SCRIPT type="text/javascript">
function checkIt(){
   var t0=document.getElementById('first').value;
   var t1=document.getElementById('second').value;
   if (t0 == "" || t0 == "Имя") {
      alert("Вы не указали свое имя!"); return false;
      }
   if (t1 == "") {
      alert("Вы не ввели необходимую информацию!");
      return false;
      }
return true;
</SCRIPT>
</head>
<body>
<form method='get' action='somescript.php'>
<input id="first" type="text" size=60px value='Имя'>
<br>
<textarea id="second" rows=4 cols=60></textarea>
<br>
<input type='submit' onClick="if (!checkIt()){return false;}" value="OK">
</form>
</body>
</html>
```

Листинг 3. Управление окнами (используется объект window)

```
<html>
<head>
<title>Открытие/закрытие нового окна</title>
</head>
<body>
<p><a name="demoOpen" onClick="mywindow =
  window.open('window.htm','mywin','height=120, width=300, left=100, top=30');">Открыть</a>
<a name="demoClose" onClick="mywindow.window.close();">Закрыть</a>
</body>
</html>
```

Листинг 4. Изменение оформления

```
<TITLE>Изменение цвета объекта по щелчку мыши</TITLE>
</head>
<BODY>
<p onClick="fgColor='#3CB094';bgColor='#FFFF00';">CLICK 4 REDRAW</p>
</BODY>
```

Листинг 5. Текущее время (использован встроенный объект Date)

```
<html>
<HEAD>
<TITLE>Часы, отображающие текущее время</TITLE>
<script type="text/javascript">
function fulltime() {
   var time=new Date();
   document.clock.full.value=time.toLocaleString(); // 1-ый вариант
   document.getElementById("jsclock").innerHTML=time.toLocaleString(); // 2-ой вариант
   setTimeout('fulltime()',500) }
</script>
</head>
<body>
<form name=clock>
<input type=text size=20 name=full><!-- 1-ый вариант -->
<span id="jsclock"></span><!-- 2-ой вариант -->
</form>
<script type="text/javascript"> fulltime(); </script>
</BODY>
</HTML>
```

Содержание отчета

1. Цель
2. Создание сценариев JavaScript.
3. Обращение к свойствам и методам объектов.
4. Иерархия объектов JavaScript.
5. События JavaScript.
6. Выводы

Контрольные вопросы

1. Иерархия объектов JavaScript.
2. Назначение объекта window.
3. Перечислите свойства и методы объекта window, которые использовали при выполнении лабораторной работы.
4. Назначение объекта document.
5. Перечислите свойства и методы объекта document, которые использовали при выполнении лабораторной работы
6. Для чего предназначено свойство href объекта location?
7. Для чего используется функция eval?
8. Опишите синтаксис конфигурации нового окна.
9. Перечислите возможные параметры нового окна.
10. В каких случаях используют после href знак #?
11. Какой метод используется для закрытия окна?
12. Что означает запись: self.*name="main_window"?*

Лабораторная работа №30
Применение технологии AJAX

Цель: получить представление о практическом использовании технологии AJAX при программировании веб-приложений.

Задачи: овладение навыками создания связанных элементов управления на веб-странице приложения, использования объекта XmlHttpRequest, использования асинхронных запросов к серверу.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

 Теоретический материал:

Традиционное веб-программирование. Классическая модель веб-приложения действует следующим образом: большинство действий пользователя отправляют обратно на сервер HTTP-запрос. Сервер производит необходимую обработку – получает данные, обрабатывает числа, взаимодействует с различными унаследованными системами и затем выдает HTML страницу клиенту. Все это время пользователю не только приходится ожидать ответов от сервера, но и для каждой пары «запрос/ответ» страница перерисовывается заново.

Таким образом, веб-сервер отвечает на каждый запрос, возвращая совершенно новую страницу HTML с обновленными данными, а весь процесс повторяется снова и снова.

Кроме того, в традиционном веб-программировании все сведения, связанные со страницей, и элементы управления на странице теряются при каждом цикле обработки. Например, если пользователь ввел сведения в текстовое поле, то эти сведения будут потеряны в ходе цикла обработки от веб-обозревателя или клиентского устройства к серверу.

AJAX. Ajax – принципиально новый подход к веб-программированию. Расшифровывается AJAX как Asynchronous JavaScript and XML (Асинхронный JavaScript и XML).

Веб-страница отправляет запросы с помощью функции JavaScript, обеспечивающей взаимодействие с сервером. Для веб-сервера ничего не изменилось – он по-прежнему отвечает на каждый запрос. Однако ответ сервера содержит только данные, необходимые для страницы без разметки и представления. Большая часть страницы остается неизменной, обновляются только части, которые должны быть изменены. JavaScript динамически обновляет веб-страницу без перерисовки.

Простыми словами можно сформулировать так: AJAX – это когда с помощью JavaScript можно запрашивать и получать данные с сервера, не перезагружая станицу сайта. Несмотря на то, что в названии технологии явным образом присутствует слово «XML», сервер может возвращать свой ответ не только в XML-формате, но и в более привычном текстовом виде.

В основе технологии лежит использование нестандартного объекта XMLHttpRequest(), необходимого для взаимодействия со скриптами на стороне сервера. Объект может как отправлять, так и получать информацию в различных форматах включая XML, HTML и даже текст.

Главное отличие в том, что web-страница, созданная с помощью AJAX, может взаимодействовать с web-сервером в «асинхронном» режиме, запрашивая и получая данные без необходимости обновления страницы в браузере. Обычная web-страница для обмена данными с сервером должна быть обновлена целиком. Например, отправка данных web-формы на сервер и отображение ответа происходят на разных web-страницах: первая

содержит данные формы, а вторая генерируется сервером в ответ на запрос браузера. Такая технология требует от браузера дополнительного сетевого трафика и времени, затрачиваемого на отображение страницы. К тому же web-сервер даже при минимальных отличиях между двумя web-страницами часто вынужден передавать большие объемы информации при загрузке каждой из них.

AJAX же позволяет обновлять лишь те фрагменты web-страницы, которые действительно изменялись, не перезагружая страницу целиком. При этом браузер обменивается данными со сценарием на web-сервере, передавая минимальную информацию – только то, что нужно.

AJAX позволяет делать запросы к серверу асинхронными. Термин «асинхронность» означает, что во время обработки запроса пользователь может продолжить работу с приложением.

Объект XMLHTTPRequest. Объект XMLHttpRequest дает возможность браузеру делать HTTP-запросы к серверу без перезагрузки страницы.

В разных браузерах этот объект создается разными способами. Так, к примеру, в браузерах Safari, Firefox, Mozilla и большинстве альтернативных браузеров этот объект называется XMLHttpRequest, в Internet Explorer – это объект ActiveX «Msxml2.XMLHTTP», в других браузерах Microsoft – это объект ActiveX «Microsoft. XMLHTTP».

Для того, чтобы послать HTTP запрос из JavaScript, необходим экземпляр класса, который поддерживает такую функцию. После инициализации экземпляра необходимо указать ему, какая функция будет обрабатывать ответ. Это делается путем присваивания свойству onreadystatechange экземпляра имени JavaScript- функции.

Отправка запросов с помощью объекта XMLHttpRequest выполняется с помощью трех основных методов:

– open(Method, Url, async) – инициализация подключения: Method – HTTP-метод (POST или GET); Url – URL сценария, работающего на веб-сервере (куда направляется этот запрос); async – вид подключения (true, если подключение асинхронное, false, если подключение синхронное).

– send(data) – параметры отправки запроса. Параметром метода send() могут быть любые данные, которые вы хотите послать на сервер. Данные должны быть сформированы в строку запроса: name=value&name1=value1&….. Обычно в качестве параметра этому методу передается значение null, т. е. без данных;

– onreadystatechange – функция обратного вызова, т. е. название той JavaScript-функции, которая должна быть вызвана при получении ответа от сервера.

Данные, полученные в результате запроса к серверному сценарию, могут быть представлены как в форме текста (свойство ResponseText), так и в виде XML-разметки (свойство ResponseXml).

Объекты XMLHTTP или XMLHttpRequest отвечающие за обращение к файлам на сервере имеют свойство .readyState. С помощью этого свойства проверяется статус запроса. Если значение переменной статуса равно 4, то ответ от сервера получен и его можно обрабатывать.

Полный список значений кодов readyState такой:

0 – uninitialized.

1 – loading.

2 – loaded.

3 – interactive.

4 – complete.

Также проверяется - статус HTTP-ответа. Важен успешный код ответа 200 OK, к неуспешным относятся, например, 404 (Не найдено) или 500 (Внутренняя ошибка сервера).

Практическая часть

Задание.

1. Исследовать и проверить работоспособность приведенного примера.
2. Уметь пояснять логику программы.
3. Создать оригинальную веб-страницу с использованием технологии AJAX.

Пример.

```
<html>
<body bgcolor=green>
<script language=JavaScript>
var req;
var d;
function processStateChange(){
if (req.readyState < 4) document.getElementById(d).innerHTML = "<font color=red>Подгрузка
<font color=red>AJAX";
if (req.readyState == 4){
contentDiv = document.getElementById(d);
alert ('Ответ получен');
if (req.status == 200){
response = req.responseText;
contentDiv.innerHTML = response;
alert ('Статус HTTP ответа 200');
} else {
contentDiv.innerHTML = "Error: Status "+req.status;
}
}
}
function load(URL, destination){
d = destination;
if (window.XMLHttpRequest)
{
alert ('используется объект XMLHttpRequest');
req = new XMLHttpRequest();
req.onreadystatechange = processStateChange;
req.open("GET", URL, true);
req.send(null);
}
else if (window.ActiveXObject)
{
req = new ActiveXObject("Microsoft.XMLHTTP");
alert ('используется объект ActiveXObject');
if (req) {
req.onreadystatechange = processStateChange;
req.open("GET", URL, true);
req.send();
}
}
}
</script>
<h2><center>ЭТО ПОСТОЯННАЯ ЧАСТЬ САЙТА<hr><input type="button"
onclick="load('news.txt', 'd')"
value="В Ы П О Л Н И Т Ь СИНХРОНИЗАЦИЮ AJAX"><hr><div id="d">область загрузки
AJAX</div>
</body>
</html>
```

Примечания:

1. Частое использование сообщений (alert ('…');) связано только с целью пояснения работы программы.
2. Файл news.txt располагается в нашем случае в одной папке с основным файлом и содержит текстовую информацию.

Содержание отчета

1. Цель
2. Назначение технологии AJAX.
3. Объект XMLHTTPRequest, назначение.
4. Методы объекта XMLHTTPRequest

5. Выводы

Контрольные вопросы

1. В чем преимущество асинхронных запросов к серверу?
2. Каков механизм их формирования и обработки?
3. Какой инструментарий формирования и обработки AJAX-запросов вы знаете?
4. Какие форматы данных используются при асинхронном обмене данных между клиентом и сервером?
5. Каким атакам подвержен асинхронный обмен данными между клиентом и сервером?
6. Какие события сопровождают асинхронный обмен между клиентом и сервером?

Лабораторная работа №31

Многоуровневое меню, многоуровневые списки в AJAX

Цель: закрепить практические навыки использования технологии AJAX при программировании веб-приложений.

Задачи: овладение навыками создания связанных элементов управления на веб-странице приложения, использования объекта XmlHttpRequest, использования асинхронных запросов к серверу.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Содержание отчета

1. Цель
2. Назначение технологии AJAX.
3. Объект XMLHTTPRequest, назначение.
4. Методы объекта XMLHTTPRequest
5. Выводы

Контрольные вопросы

1. В чем преимущество асинхронных запросов к серверу?
2. Каков механизм их формирования и обработки?
3. Какой инструментарий формирования и обработки AJAX-запросов вы знаете?
4. Какие форматы данных используются при асинхронном обмене данных между клиентом и сервером?
5. Каким атакам подвержен асинхронный обмен данными между клиентом и сервером?
6. Какие события сопровождают асинхронный обмен между клиентом и сервером?

Лабораторная работа №32

Реализация поиска и быстрого поиска в AJAX

Цель: закрепить практические навыки использования технологии AJAX при программировании веб-приложений.

Задачи: овладение навыками создания связанных элементов управления на веб-странице приложения, использования объекта XmlHttpRequest, использования асинхронных запросов к серверу.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Поле SELECT с поиском

Предположим у нас есть таблица, в которой порядка миллиона записей. Пользователю необходимо выбрать всего одну запись из таблицы (реализация отношения "один ко многим"). Выбор пользователя является всего лишь одним из этапов заполнения большой веб-формы.

Естественно, для того, чтобы пользователь мог выбрать нужную запись из миллиона, нужны какие-то средства поиска этой самой записи. Например, простой текстовый поиск по наименованию.

В традиционном веб-приложении для этой цели пришлось бы использовать отдельную страницу и сохранять остальные данные формы в сессии пользователя, либо разбивать процесс заполнения формы на несколько этапов. В AJAX-приложении дополнительная страница не нужна.

Выбор записи будет реализован с помощью двух элементов веб-формы. Первый элемент - это текстовое поле, где пользователь вводит ключевое слово. Оно отсылается на сервер, а тот возвращает только те строки из таблицы, которые удовлетворяют условию поиска. Ответ сервера (в виде списка) помещается в поле SELECT, в котором пользователь и сделает окончательный выбор. Таким образом, при отправке всей формы на сервер попадет выбранное в поле SELECT значение в виде ID записи из большой таблицы.

Практическая часть

В HTML выглядеть это может так:

```
<input type="text"
onkeyup="lookup(this. value, 'id_select',
'http://localhost/cgi-bin/xmlhttp. cgi')" />
<select id="id_select" name="id_select">
<option selected="selected" value=""></option>
</select>
```

На любое событие KeyUp (отжатие кнопки) в текстовом поле вызывается функция lookup ('текст', 'id-selecta', 'url')

```
function lookup(text, select_id, url) {
// Получаем объект XMLHTTPRequest
if(! this. http) {
this. http = get_http();
this. working = false;
}
// Запрос
if (! this. working && this. http) {
var http = this. http;
```

```
// Если в текстовом поле менее трёх
// символов - не делаем ничего
if (text. length <3) return;
// добавляем закодированный текст
// в URL запроса
url = url + "? text="+encodeURIComponent(text);
// создаём запрос
this. http. open("GET", url, true);
// прикрепляем к запросу функцию-обработчик
// событий
this. http. onreadystatechange = function() {
// 4 - данные готовы для обработки
if (http. readyState == 4) {
fill(select_id, http. responseText);
this. working = false;
}else{
// данные в процессе получения,
// можно повеселить пользователя
// сообщениями
// ЖДИТЕ ОТВЕТА
}
}
this. working = true;
this. http. send(null);
}
if(! this. http) {
alert('Ошибка при создании XMLHTTP объекта! ')
}
}
```

Как видно, в начале мы получаем XMLHTTP-объект с помощью функции get_http(). Затем поисковый текст кодируется в стиле URL и формируется GET-запрос к серверу. URL запроса в данном случае будет выглядеть приблизительно так: http://localhost/cgi-bin/xmlhttp. cgi? text=...

Скрипт на сервере, получив значение text, делает поиск в таблице и отсылает результат клиенту. В обработчике событий объекта XMLHTTP, когда данные от сервера получены и готовы к использованию, вызывается функция fill('select_id', 'data'), которая заполнит список SELECT полученными данными.

Функция get_http() - это кросс-браузерная реализация получения объекта XMLHTTP (в каждом браузере он получается по-своему). Её реализацию с комментариями вы можете легко найти в интернете, это, так сказать, пример из учебника.

```
function get_http() {
var xmlhttp;
/*@cc_on
@if (@_jscript_version >= 5)
try {
xmlhttp = new ActiveXObject("Msxml2. XMLHTTP");
} catch (e) {
try {
xmlhttp = new
ActiveXObject("Microsoft. XMLHTTP");
} catch (E) {
xmlhttp = false;
}
}
@else
xmlhttp = false;
@end @*/
if (! xmlhttp && typeof XMLHttpRequest! = 'undefined') {
try {
xmlhttp = new XMLHttpRequest();
} catch (e) {
xmlhttp = false;
}
}
return xmlhttp;
}
```

Функция fill() получает на вход значение параметра ID списка SELECT, который необходимо заполнить, и сами данные, полученные с сервера.

Для простоты предположим, что данные с сервера мы получаем в виде таблицы, поля которой Разделены символом табуляции 't', а строки - символом переноса строки 'n':

id1tname1n

id2tname2n

На основании содержимого этой таблицы мы будем заполнять поле SELECT элементами OPTION.

```
function fill (select_id, data) {
// поле SELECT в переменную в виде объекта
var select = document. getElementById(select_id);
// очищаем SELECT
select. options. length = 0;
// если данных нет - не делаем больше ничего
if(data. length == 0) return;
// в массиве arr - строки полученной таблицы
var arr = data. split('n');
// для каждой строки
for(var i in arr) {
// в массиве val - поля полученной таблицы
val = arr [i]. split('t');
// добавляем новый объект OPTION к нашему SELECT
select. options [select. options. length] =
new Option(val [1], val [0], false, false);
}
}
```

Готово. Теперь для любой веб-формы приложения мы можем реализовать подобный выбор значения из многомиллионного списка, который для пользователя будет выглядеть как считанные нажатия клавиш.

Нам нужно динамически создавать <script> в документе. Отведем для него специальный блок:

Стоит заметить, что нам понадобится передавать параметры php-скрипту, а функции URL-кодирования в JavaScript нет. Значит нужно создать нечто вроде таблицы кодирования (выводится элементарным php-скриптом и немного правится руками):

Теперь напишем функцию, которая будет динамически создавать <script>. Функция принимает три параметра: адрес php-файла, массив имен передающихся переменных и массив значений этих переменных. Каждое значение мы будем URL-кодировать.

```
function LoadScript(addr,query,str) {
// составление строки запроса
for(k = 0; k < str. length; k++)
{
str2 = "";
// URL-кодируем
for(j = 0; j < str [k]. length; j++) str2 += '%' + chr [str [k]. charAt(j)] ;
// добавляем к концу запроса "переменная=значение"
addr += query [k] +"="+str2;
}
* разбиваем на две строки.
*/
_ajax. innerHTML = "MSIE... <script></"+"script>";
/*
* даем JavaScript 10 мсек на осознание того, что _ajax изменен
* и назначаем атрибут src.
*/
setTimeout(
function()
{
scr = _ajax. getElementsByTagName("script") [0] ;
scr. language = "javascript";
if (scr. setAttribute) scr. setAttribute("src",addr);
else scr. src=addr;
}
,
10);
}
```

Теперь, чтобы выполнить запрос index. php? action=view&id=49, нужно вызвать функцию LoadScript() следующим образом:

```
LoadScript("index. php",Array("? action","&id"),Array("view","49"));
```

Для выполнения скрипта без параметров следует передать функции LoadScript() его адрес и два пустых массива.

Дя этого в документе, из которого запускаем LoadScript(), создаем блок

```
<div id="_hz"></div>
```

А в php-скрипте пишем

```
echo<<<a
_hz. innerHTML="работает!!!";
a;
```

Все скрипты, вызывающиеся как <script src="address"></script> кэшируются браузером. Чтобы этого избежать, каждый php-скрипт начинаем с четырех строк:

```
Header("Expires: Mon, 26 Jul 1997 05: 00: 00 GMT");
Header("Cache-Control: no-cache, must-revalidate");
Header("Pragma: no-cache");
Header("Last-Modified: ". gmdate("D, d M Y H: i: s"). "GMT");
```

Содержание отчета

1. Цель
2. Назначение технологии AJAX.
3. Объект XMLHTTPRequest, назначение.
4. Методы объекта XMLHTTPRequest
5. Выводы

Контрольные вопросы

1. В чем преимущество асинхронных запросов к серверу?
2. Каков механизм их формирования и обработки?
3. Какой инструментарий формирования и обработки AJAX-запросов вы знаете?
4. Какие форматы данных используются при асинхронном обмене данных между клиентом и сервером?
5. Каким атакам подвержен асинхронный обмен данными между клиентом и сервером?
6. Какие события сопровождают асинхронный обмен между клиентом и сервером?

Лабораторная работа №33

Использование библиотеки jQuery. Создание выпадающего списка

Цель: получить практические навыки использования библиотеку jQuery при программировании веб-приложений.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

jQuery — библиотека, которая позволяет делать код короче, а также позволяет внутри страницы настроить код, который бы срабатывал как триггер (предопределенный набор действий, который выполняется автоматически при наступлении связанного с ним события, если этот код описываем в области <head> … </head>).

JQuery библиотека содержит следующий функционал:

операции с HTML/DOM (манипулирование компонентами HTML/DOM)
операции с CSS-селекторами

HTML-обработчики событий Эффекты анимации

AJAX Utilities

JQuery упрощает работу с JavaScript, а также вызовы AJAX и DOM-манипуляции.

Есть много фреймворков JavaScript, но JQuery, является самым популярным и используемым за счет своей расширяемости.

Начало работы с библиотекой

Библиотеку jQuery можно скачать с сайта http://jquery.com, а можно вставить в документ, используя известные интернет-адреса:

По адресу `http://code.jquery.com/jquery-latest.js` – доступна всегдапоследняя версия.

С Google: https://developers.google.com/speed/libraries/devguide?hl=ru#jquery можно загрузить любую из не слишком старых версий. Синтаксис такой:

`src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"`, где `1.8.3` — версия, причём можно указать её приблизительно: `1.8` означает последнюю версию вида `1.8.*`, а `1` — последнюю версию вида `1.*`. Файл `jquery.min.js` обозначает сжатый код, а `jquery.js` — несжатый, для удобства отладки.

Либо с Microsoft CDN:

`src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.9.1.min.js"`

Пример подключения ниже (!!! но использовать один из вариантов подключения библиотеки):

```
<!DOCTYPE html>
<html>
<head>
<!--если выбираем библиотеку от Microsoft CDN -->
<script src="http://ajax.aspnetcdn.com/ajax/jQuery/jquery-
1.9.1.min.js"></script> <!--если выбираем библиотеку от Google -->
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></
script> <!--третий вариант -->
<script src="http://code.jquery.com/jquery-latest.js"></script>
</head>
<body>
</body>
</html>
```

jQuery синтаксис

Базовая команда для библиотеки вяглядит как: **$(*селектор*).*action*()**,

где $ - предписание использовать jQuery;

(*селектор*) - это "запрос или элементы поиска" в HTML элементах страницы;

action() - это действия, которые должны быть выполнены над найденными элементами (это те элементы, которые удовлетворяют условиям селектора).

Например:

$(this).hide() – скрывает текущий элемент (где **this** – это указатель на текущий элемент, позволяет делать код универсальным за счет того, что не надо писать здесь имя или id элемента, над которым будет производится действие hide()).

$("p").hide() – скрывает все <p> элементы на странице.

$(".test").hide() – скрывает все элементы на странице, которые ассоциированы с классом "test".

$("#test").hide() – скрывает все элементы на странице, у которых id="test".

Событие Ready у объекта страницы Document

Вы, заметите, что в большинстве примеров jQuery-методы находятся внутри события документа Ready():

$(document).ready(function(){

/ *jQuery-методы размещаем здесь...*

});

Это необходимо для предотвращения любых срабатываний JQuery-кода, прежде чем документ не закончит полную загрузку.

Это хорошая практика, чтобы дождаться, пока документ будет полностью загружен и готов до работы с ним. Это также позволяет вам сформировать свой JavaScript код в головной части, прежде чем тело документа.

jQuery селекторы

jQuery селекторы позволяют вам делать выборку (поиск) и манипулировать с элементами HTML. Селекторы по сути это набор условных обозначений и правил для выборки и манипулирования (в конце лабораторной в приложении дан большой список примеров селекторов).С jQuery селекторами вы можете найти элементы страницы, основанные на идентификаторе id, классах (class), типах (type), атрибутах (attribute), значениях атрибутов (value) и др. Также они базируются и на CSS Selectors, в дополнении вы можете создать свой селектор. Все типы селекторов в jQuery начинаются с указания $ и парных скобок:**$( )**

Например, в следующем коде при нажатии на кнопку выполняется поиск на странице всех элементов, обозначенных тегом <p> , и все эти элементы скрываются на странице (срабатывает метод hide( )):

```
<!DOCTYPE html>
      <html>
      <head>
      <script
src="http://code.jquery.com/jquery-latest.js">
</script>
      <script>
    $(document).ready(function () {
        $("button").click(function () {
            $("p").hide();
        });
    });
</script>
</head>
<body>
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
```

```
</html>
```

Обратите внимание на то, что селектор в $( )указывается в кавычках а не в < >.

Практическая часть

Задание 1: Выполнить пример, проверить работу метода hile ( ). Попробовать заменить библиотеку `jquery-latest.js` на другие две, описанные выше, сравнить результаты работы.

Пояснения к скрипту:

```
<script>
//в строке ниже в качестве селектора использован весь объект –документ,
//при этом будет срабатывать jquery-запрос, как только наступит событие полной готовности //страницы к работе с пользователем и запустится метод ready(),
/ при срабатывании которого будет создана следующая функция $(document).ready(function () {
//созданная функция в свою очередь будет jquery-запросом, который ищет все элементы типа //button – кнопка, и с их методами click связывает (ассоциирует) действие в виде функции,
$("button").click(function () {
/ которая выполнит jquery-запрос, который для всех найденных элементов внутри тега <p> выполнит метод hide(), т.е. скроет их со страницы
$("p").hide();
//далее закрываем внутреннюю функцию
});
//далее закрываем внешнюю функцию
});
</script>
```

В таком исполнении скрипт, помещенный в заголовок страницы внутрь метода объект document.ready( ) работает как триггер, т.е. автоматически срабатывает при наступлении определенного события на странице. В примере – это событие click( ) кнопки. А так как в методе hide( ) нет никакого описания другого кода, то он выполняет те действия, для которых он изначально создан, а именно скрывает объект.

Селектор #id

Селектор jQuery #id использует id атрибут в HTML-тегах, чтобы найти определенный элемент.

Id должен быть уникальным внутри всей страницы, если вы хотите найти с его помощью конкретный уникальный элемент. Чтобы найти элемент с помощью id, то перед названием искомого идентификатора ставится знак #, например:

$("#test")

В примере ниже, при нажатии на кнопку ищется элемент с идентификатором test и скрывается со страницы.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("#test").hide();
        });
        });
</script>
</head>
<body>
<h2>This is a heading 2</h2>
<p>This is a paragraph.</p>
<p id="test">This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

Задание 2: Выполнить пример, проанализировать отличие его от примера в задании 1.

Ниже приведен пример, в котором работают две кнопки: одна – скрывает элемент с id=”test”, вторая – отображает этот элемент (используется метод show( )). Обратите внимание, чтобы распараллелить код по двум кнопкам, для каждой из них тоже были определены id, по которым определяется какую функцию запускать.

```
<!DOCTYPE html>
<html>
<head>
<script          src="http://code.jquery.com/jquery-latest.js">
</script>
<script>
$(document).ready(function () {
$("#but1").click(function () {
$("#test").hide();
});
$("#but2").click(function () {
$("#test").show();
});
});
</script>
</head>
<body>
<h2>This is a heading 2</h2>
<p>This is a paragraph.</p>
<p id="test">This is another paragraph.</p>
<button id="but1">Click me</button>
<button id="but2">Click me for show</button>
</body>
</html>
```

Задание3: Выполнить пример, проанализировать отличие его от примера в задании 2.

.class селектор Селектор jQuery class находит элементы определенного класса.

Для поиска элементов определенного класса указывается перед названием точка, например:

$(".test")

В примере ниже при нажатии на кнопку элементы с классом="test" будут скрыты:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $(".test").hide();
        });
    });
</script>
</head>
<body>
<h2 class="test">This is a heading</h2>
<p class="test">This is a paragraph.</p>
<p>This is another paragraph.</p>
<button>Click me</button>
</body>
</html>
```

Задание 4: Выполнить пример, проанализировать отличие его от примера в задании 3.

Другие примеры jQuery селекторов (делать необязательно, просто при необходимости для ознакомления)

Синтаксис	Описание	Пример
$("*")	Selects all elements	Try it
$(this)	Selects the current HTML element	Try it
$("p.intro")	Selects all <p> elements with class="intro"	Try it

$("p:first")	Selects the first <p> element	Try it
$("ul li:first")	Selects the first element of the first	Try it
$("ul li:first-child")	Selects the first element of every	Try it
$("[href]")	Selects all elements with an href attribute	Try it
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"	Try it
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"	Try it
$(":button")	Selects all <button> elements and <input> elements of type="button"	Try it
$("tr:even")	Selects all even <tr> elements	Try it
$("tr:odd")	Selects all odd <tr> elements	Try it

Индивидуальное задание 1:

Вариант 1: в исходный файл Experiments.html добавить в начало две кнопки: одну-для скрытия элементов, другую – для отображения скрытых элементов. Настроить методы click( ) кнопок, так чтобы они то скрывали, то отображали нечетные элементы с классом `MsoNormal.`

Вариант 2: в исходный файл Experiments.html добавить в начало две кнопки: одну-для скрытия элементов, другую – для отображения скрытых элементов. Настроить методы click( ) кнопок, так чтобы они то скрывали, то отображали четные элементы типа `<tr>.`

Вариант 3: в исходный файл Experiments.html добавить в начало две кнопки: одну-для скрытия элементов, другую – для отображения скрытых элементов. Настроить методы click( ) кнопок, так чтобы они то скрывали, то отображали элементы с атрибутом `href.`

Вариант 4: в исходный файл Experiments.html добавить в начало две кнопки: одну-для скрытия элементов, другую – для отображения скрытых элементов. Настроить методы click( ) кнопок, так чтобы они то скрывали, то отображали элементы, у которых атрибут `align` равен значению `center.`

jQuery-методы событий

Ниже представлены общие с технологией DOM события:

События мыши	**События клавиатуры**	**События формы**	**События документа/окна**
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

jQuery-синтаксис для методой событий

В jQuery, большинство DOM-событий имеют эквивалентный jQuery-метод.

Чтобы назначить событие нажатия мышкой на все элементы <p> на странице, вы можете написать: $("p").click();

Следующий шаг – это определение того, что будет происходить, когда наступит указанное событие.

Вы должны определить функцию для события:

```
$("p").click(function(){
//описание действий в функции
});
```

Часто используемые методы JQuery

**$(document).ready( )**$(document).ready() метод позволяет вам выполнить функцию, когда документ полностью загружен.

click( )Эта функция выполняется, когда пользователь нажимает на HTML элемент.

В примере событие click( ) активируется на <p> элементах, скрывая текущий <p> элемент:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-latest.js"> </script>
<script>
    $(document).ready(function () {
        $("p").click(function () {
        $(this).hide();
        });
    });
</script>
</head>
<body>
<p>If you click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>
</body>
</html>
```

Задание 5: Выполнить пример, проверить работу.

Обратите внимание! `$(this).hide();//указатель this позволяет выполнять действие //с текущим активированным элементом`

dblclick( )Срабатывает, когда пользователь двойным щелчком нажимает на HTML-элемент:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("p").dblclick(function () {
            $(this).hide();
        });
    });
</script>
</head>
<body>
<p>If you double-click on me, I will disappear.</p>
<p>Click me away!</p>
<p>Click me too!</p>
</body>
</html>
```

Задание 6: Выполнить пример, проверить работу.

mouseenter() Выполняется, когда указатель мыши наводится на HTML-элемент:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("#p1").mouseenter(function () {
            alert("You entered p1!");
        });
    });
</script>
</head>
<body>
<p id="p1">Enter this paragraph.</p>
</body>
</html>
```

Задание 7: Выполнить пример, проверить работу.

blur() Выполняется, когда поле формы теряет фокус.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("input").focus(function () {
            $(this).css("background-color", "#cccccc");
        });
        $("input").blur(function () {
            $(this).css("background-color", "#ffffff");
        });
    });
</script>
</head>
<body>
Name: <input type="text" name="fullname"><br>
Email: <input type="text" name="email">
</body>
</html>
```

Задание 8: Выполнить пример, проверить работу. Проверить как работает focus() и blur(), а также разобраться с настройкой стилей через .css("свойство", "значение").

hide() и show() С jQuery вы можете скрывать и отражать HTML-элементы: Также возможно настроить время затухания и появления:

Синтаксис:

$(*selector*).hide(*speed*);
$(*selector*).show(*speed*);

или

$(*selector*).hide(*speed,callback*);
$(*selector*).show(*speed,callback*);

Необязательный параметр скорости определяет скорость скрытия / показа, и может принимать следующие значения: "slow", "fast" или в миллисекундах.

Следующий пример демонстрирует параметр скорости скрытия элемента:

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-latest.js">
</script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("p").hide(1000);
        });
    });
</script>
</head>
<body>
<button>Hide</button>
<p>This is a paragraph with little
content.</p> <p>This is another small
paragraph.</p> </body>
</html>
```

Задание 9: Выполнить пример, проверить работу.

toggle() С помощью метода можно переключать выполнение методов hide() и show(). Также можно настроить и скорость срабатывания.

Синтаксис: $(*selector*).toggle(*speed*);

```
<!DOCTYPE html>
<html>
<head>
```

```
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("p").toggle();
        });
    });
</script>
</head>
<body>
<button>Toggle</button>
<p>This is a paragraph with little content.</p>
<p>This is another small paragraph.</p>
</body>
</html>
```

Задание 10: Выполнить пример, проверить работу.

Индивидуальное задание 2:

Вариант 1: в исходный файл Experiments2.html добавить обработку события «наведение мышки на элемент». Если этот элемент относится к таблице (в теге <img>), то при наведении на него курсора, он должен скрываться.

Вариант 2: в исходный файл Experiments2.html добавить обработку события «двойной щелчок мышки по элементу». Для элемента с атрибутом `align="center"` нужно настроить исчезновение/появление элемента.

Вариант 3: в исходный файл Experiments2.html добавить обработку события «изменение размера окна». При срабатывании этого метода скрыть с документа все элемента со ссылками (href).

Вариант 4: в исходный файл Experiments2.html добавить элемент формы «поле ввода», настроить для него обработку keypress, а именно случайным образом менять цвет этого поля (свойство `background-color`).

Fading методы

С JQuery вы можете исключать и включать видимость элементов.

Набор методов: fadeIn(), fadeOut(), fadeToggle(), fadeTo()

fadeIn() fadeIn() используется, чтобы реализовать проявление в скрытых элементах (например, элемент скрытый). Можно также использовать настройку скорости срабатывания.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("#div1").fadeIn();
            $("#div2").fadeIn("slow");
            $("#div3").fadeIn(3000);
        });
    });
</script>
</head>
<body>
<p>Demonstrate fadeIn() with different
parameters.</p> <button>Click to fade in
boxes</button>
<br><br>
<div id="div1"
style="width:80px;height:80px;display:none;background-
color:red;"></div><br>
```

```
<div id="div2"
style="width:80px;height:80px;display:none;background-
color:green;"></div><br>
<div id="div3" style="width:80px;height:80px;display:none;background-
color:blue;"></div>
</body>
</html>
```

Задание 11: Выполнить пример, проверить работу. Разобраться с настройками элементов , обратите внимание на свойство `display:none` , которое при загрузке странице позволяет не отражать .

fadeOut() fadeOut() используется, чтобы реализовать скрытие элементов из зоны видимости.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("#div1").fadeOut();
            $("#div2").fadeOut("slow");
            $("#div3").fadeOut(3000);
        });
    });
</script>
</head>
<body>
<p>Demonstrate fadeOut() with different parameters.</p>
<button>Click to fade out boxes</button>br><br>
<div id="div1" style="width:80px;height:80px;background-
color:red;"></div><br> <div id="div2"
style="width:80px;height:80px;background-color:green;"></div><br> <div
id="div3" style="width:80px;height:80px;background-color:blue;"></div>
</body>
</html>
```

Задание 12: Выполнить пример, проверить работу. Обратите внимание на отсутствие свойства `display:none`, поэтому настройки элементов сначала отражаются.

Индивидуальное задание 3:

Вариант 1: создать файл Individual_3.html, добавить два рисунка, невидимые при загрузке страницы, и две кнопки. При нажатии на одну кнопку должно в цикле срабатывать проявление первой картинки и исчезновение второй картинки со скоростью 2000, а затем наоборот, при нажатии на вторую кнопку обе картинки должны появиться и прекратить мигание.

Вариант 2: создать файл Individual_3.html, добавить пять областей div с id “div1”, “div2”,…,“div5”, в которые поместить тексты, невидимые при загрузке страницы, и две кнопки. При нажатии на одну кнопку должно определяться случайное число от 1 до 5. В зависимости от этого числа должна проявиться соответствующая область div со скоростью 3000. При нажатии на вторую кнопку все области div снова скрываются.

Вариант 3: создать файл Individual_3.html, добавить 10 параграфов (<p>) с каким-либо текстом, невидимые при загрузке страницы, и кнопку. При нажатии на кнопку должно в цикле срабатывать проявление четных параграфов и исчезновение нечетных со скоростью 2000, а затем наоборот.

Вариант 4: создать файл Individual_3.html, добавить пять областей div 40x40 пикселей с id “div1”, “div2”,…,“div5”, раскрашенные в разные цвета, невидимые при загрузке страницы, и кнопку. При нажатии на кнопку должно в цикле срабатывать волнообразное проявление областей div слево-направо, а затем наоборот, исчезновение. Цикл работает до тех пор, пока не закрыта страница.

Sliding методы

Реализацию движения по принципу слайдов можно выполнять с помощью методов: slideDown(), slideUp(), slideToggle()

slideDown() Используется для скольжения элемента вниз.

Синтаксис: $(*selector*).slideDown(*speed*);

Необязательный параметр скорости определяет скорость скольжения, и может принимать следующие значения: "slow", "fast" или в миллисекундах.

```
<!DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("#flip").click(function () {
            $("#panel").slideDown("slow");
        });
    });</script>
<style type="text/css">
#panel,#flip
{
padding:5px;
text-align:center;
background-color:#e5eecc;
border:solid 1px #c3c3c3;
}
#panel
{
padding:50px;
display:none;
}
</style>
</head>
<body>
<div id="flip">Click to slide down panel</div>
<div id="panel">Hello world!</div>
</body>
</html>
```

Задание 13: Выполнить пример, проверить работу. Разобраться с элементом <style type="text/css">. Обратите внимание, что настройки делаются как общие для нескольких элментов, так и индивидуальные.

Метод animate() Animate() используется для создания пользовательской анимации.

Синтаксис: $(*selector*).animate({*params*},*speed,callback*);

Обязательный ***params*** определяется параметрами CSS-свойств для анимации. Необязательный параметр ***speed*** определяет длительность эффекта. Он может принимать следующие значения: "slow", "fast", или в миллисекундах.

В следующем примере показан простой вариант использования метода, он перемещает элемент влево, пока он не достиг значения свойства **left**=250px: <!

```
DOCTYPE html>
<html>
<head>
<script src="http://code.jquery.com/jquery-
latest.js"> </script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            $("div").animate({ left: '250px' });
        });
    });
</script>
</head>
```

```
<body>
<button>Start Animation</button>
<p>By default, all HTML elements have a static position, and cannot be moved.
To manipulate the position, remember to first set the CSS position property
of the element to relative, fixed, or absolute!</p>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;">
</div>
</body>
</html>
```

Задание 14: Выполнить пример, проверить работу. Разобраться с элементом `<div>`.

С прочими примерами анимации можно познакомиться на http://www.w3schools.com/jquery/jquery_animate.asp

Перебор результатов поиска

Результатом поиска является jQuery-объект. Он похож на массив: в нём есть нумерованные элементы и length (длина массива), но методы у него совсем другие. jQuery-объект также называют «jQuery-коллекцией», «элементами, обёрнутыми в jQuery» и пр. Используем jQuery, чтобы выбрать все элементы по селектору `li> a` и перебрать их:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://code.jquery.com/jquery.js"></script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            var links = $('li> a');
 /  перебор результатов
            for (var i = 0; i < links.length; i++) {
                alert(links[i].href);
            }
        });
    });
</script>
</head>
<body>
<ul>
  <li><a href="http://jquery.com">jQuery</a></li>
  <li><a href="http://jqueryui.com">jQuery UI</a></li>
  <li><a href="http://blog.jquery.com">jQuery
Blog</a></li> </ul>
<button>Start query</button>
</body>
</html>
```

Задание 15: Выполнить пример, проверить работу. Разобраться, как организована работа с переменной **links** и работа с ней в цикле.

Контекст поиска

А что, если контекст поиска содержит много элементов? Например, как будет работать запрос `$('a', 'li')`, если `<li>` в документе много?

Если в контексте много элементов, то поиск будет произведён в каждом из них. А затем результаты будут объединены.

Повторы элементов при этом отфильтровываются, то есть два раза один и тот же элемент в результате не появится.

Например, найдём `$('a', 'li')` в многоуровневом списке:

```
<!DOCTYPE HTML>
<html>
<body>
<script src="http://code.jquery.com/jquery.js"></script>
<ul>
  <li>
```

```
      <a href="http://jquery.com">jQuery</a>
      <ul>
        <li><a href="http://blog.jquery.com">jQuery
      Blog</a></li> </ul>
    </li>
    <li><a href="http://sizzlejs.com">Sizzle</a></li></ul>
  <script>
      var links = $('a', 'li');
      for (var i = 0; i < links.length; i++) {
          alert(i + ": " + links[i].href); // 3 ссылки по очереди
     }
  </script>
  </body>
  </html>
```

Задание 16: Выполнить пример, проверить работу. Разобраться, как организована работа jQuery, когда скрипт размещается в теле страницы (нет объявления функций и привязок к методам).

Метод each()

Для более удобного перебора у jQuery-коллекции есть метод each. Его синтаксис похож на forEach массива:

```
.each( function(index, item) )
```

Он выполняет для каждого элемента коллекции перед точкой функцию-аргумент, и передаёт ей номер `index` и очередной элемент `item`.

Используем его вместо `for`, чтобы перебрать коллекцию найденных ссылок:

```
$('li a').each(function(i, a) {
alert( i + ": " + a.href);
});
```

У **`.each`** есть важная возможность, которой нет в **`forEach`**: возврат **`false`** из функции прекращает перебор.

Например:

```
<!DOCTYPE HTML>
<html>
<body>
<script src="http://code.jquery.com/jquery.js"></script>
<a href="http://wikipedia.ru">Википедия</a>
<ul>
  <li><a href="http://jquery.com">jQuery</a></li>
  <li><a href="http://sizzlejs.com">Sizzle</a></li>
  <li><a href="http://blog.jquery.com">jQuery Blog</li>
</ul>
<script>
    var links = $('li a'); // найти все ссылки на странице внутри LI
    links.each(function (i, a) {
        alert(i + ': ' + a.innerHTML);
        if (i == 1) return false; // стоп на элементе коллекции с индексом 1
    });
</script>
</body>
</html>
```

Задание 17: Выполнить пример, проверить работу. Разобраться, как организована работа метода each( ).

Получение контента с помощью text(), html(), and val()

Три основных, но полных jQuery метода для манипулирования данными технологии DOM:

text() - Задает или возвращает текстовое содержимое выбранных элементов.

html() - Задает или возвращает содержание отдельных элементов (включая HTML markup).

val() - Задает или возвращает значение поля формы.

В следующем примере показано, как получить контент с помощью text() и html() методов:

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://code.jquery.com/jquery.js"></script>
<script>
    $(document).ready(function () {
        $("#btn1").click(function () {
            alert("Text: " + $("#test").text());
        });
        $("#btn2").click(function () {
            alert("HTML: " + $("#test").html());
        });
    });
</script>
</head>
<body>
<p id="test">This is some <b>bold</b> text in a
paragraph.</p> <button id="btn1">Show Text</button>
<button id="btn2">Show HTML</button>
</body>
</html>
```

Задание 18: Выполнить пример, проверить работу. Разобраться, как организована работа метода `text()` и `html()`.

В следующем примере показано, как получить значение поля ввода с помощью метода

VAL():

```
<!DOCTYPE html>
<html>
<head>
<script
src="http://code.jquery.com/jquery.js"></script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            alert("Value: " + $("#test").val());
        });
    });
</script>
</head>
<body>
<p>Name: <input type="text" id="test" value="Mickey Mouse"></p>
<button>Show Value</button>
</body>
</html>
```

Задание 19: Выполнить пример, проверить работу. Разобраться, как организована работа метода `val()`.

Получение значений атрибутов с помощью attr()

В следующем примере показано, как получить значение атрибута HREF из ссылки:

```
<!DOCTYPE html>
<html>
<script
src="http://code.jquery.com/jquery.js"></script>
<script>
    $(document).ready(function () {
        $("button").click(function () {
            alert($("#w3s").attr("href"));
        });
    });
</script>
</head>
```

```
<body>
<p><a href="http://www.w3schools.com"
id="w3s">W3Schools.com</a></p> <button>Show href
Value</button>
</body>
</html>
```

Задание 20: Выполнить пример, проверить работу. Разобраться, как организована работа метода `attr()`. Определите, значения каких еще атрибутов можно так получить.

Индивидуальное задание 4:

Вариант 1: создать файл Individual_4.html, добавить сверху кнопку, а затем пять параграфов с каким-либо текстом и между каждым параграфом по маленькому рисунку. При нажатии на кнопку среди всех найденных параграфов для 3-го и 5-го изменить положение – сместить его на 100 пт влево относительно текущего положения. При нажатии на любую из картинок должно происходить движение текущей вниз на 30 пт.

Вариант 2: создать файл Individual_4.html, добавить сверху кнопку, а затем 6 маленьких рисунков одинакового размера, один под другим. При нажатии на кнопку среди всех найденных рисунков для четных выполнить эффект слайд-шоу – заезд/скрытие рисунка под соответствующим ему верхним рисунком со скоростью «slow».

Вариант 3: создать файл Individual_4.html, добавить сверху кнопку, а затем 6 параграфов с каким-либо текстом, один под другим. При нажатии на кнопку среди всех найденных рисунков для нечетных выполнить извлечение текста и присвоение этого текста соответствующему нижнему/четному параграфу и сместить нечетные элементы на 40 пт влево.

Вариант 4: создать файл Individual_4.html, добавить сверху кнопку, а затем 6 маленьких рисунков одинакового размера, один под другим. При нажатии на кнопку среди всех найденных рисунков для нечетных выполнить эффект слайд-шоу – заезд/скрытие рисунка под соответствующим ему нижним рисунком со скоростью «slow».

Получение конкретного элемента

Даже если элемент найден только один, всё равно результатом будет jQuery-коллекция.

Для получения одного элемента из jQuery-коллекции есть несколько способов:

1. Метод get(индекс), работает так же, как прямой доступ:

```
alert( $('body').get(0) ); // BODY
```

Если элемента с таким номером нет — вызов `get` возвратит `undefined`.

2. Метод eq(индекс) возвращает коллекцию из одного элемента — с данным номером. Он отличается от метода get(индекс) и прямого обращения по индексу тем, что возвращает именно jQuery-коллекцию с одним элементом, а не сам элемент.

```
/    DOM-элемент для первой ссылки
$('a').get(0);
/    jQuery-объект из одного элемента: первой ссылки
$('a').eq(0);
```

Во втором случае вызов `eq` создаёт новую jQuery-коллекцию, добавляет в нее нулевой элемент и возвращает. Это удобно, если мы хотим дальше работать с этим элементом, используя методы jQuery. Если элемента с таким номером нет — `eq` возвратит пустую коллекцию.

Содержание отчета

1. Цель
2. Программные листинги (выдержки)
3. Выводы

Контрольные вопросы

1. Что такое jQuery
2. Преимущества jQuery
3. В чем отличие jQuery от JavaScript
4. Как начать работу с jQuery

5. Синтаксис jQuery

Лабораторная работа №34
Создание фотогалереи на JQuery

Цель: закрепить практические навыки использования библиотеку jQuery при программировании веб-приложений.

Ход работы:

1. Выполнить задания в соответствии с указаниями.
2. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
3. Подготовить ответы на контрольные вопросы (устно)

Задание 1. jQuery работа с выпадающим списком select (пример)

Очень часто приходится сталкиваться с выпадающим HTML списком <select>, и тогда возникают вопросы, как можно управлять им. Данная статья посвящена некоторым селекторам jQuery для работы со списком <select>. Работа с выпадающим списком с помощью jQuery также проста, как и работа с любым другим элементом.

Для начала создайте простенький выпадающий список, чтобы наглядно показать работу селекторов:

```
<select id="test_select" name="test_select">
   <option value=""></option>
   <option value="1">Первое поле имеет значение value=1</option>
   <option value="2">Второе поле имеет значение value=2</option>
   <option value="3">Третье поле имеет значение value=3</option>
   <option value="4">Четвертое поле имеет значение value=4</option>
   <option value="5">Пятое поле имеет значение value=5</option>
   <option value="6">Шестое поле имеет значение value=6</option>
</select>
```

Обратите внимание на то, что также был добавлен пустой элемент списка, это пригодится, чтобы проверить выбран хоть один элемент списка или нет, а также для сброса выбора.

1 — Получить значение выбранного элемента: $("#test_select option:selected").val();
сокращенно: $("#test_select :selected").val();
или: $("select#test_select").val();
2 — Получить текст того же выбранного элемента: $("#test_select :selected").html();
или: $("#test_select :selected").text();
3 — Сделать <select> недоступным: $("#test_select").attr("disabled",true);
4 — Разблокировать <select>: $("#test_select").attr("disabled",false);
5 — Удалить выбранный элемент: $("#test_select :selected").remove();
6 — Удалить первый элемент: $("#test_select :first").remove();
7 — Удалить последний элемент: $("#test_select :last").remove();
8 — Удалить элемент, у которого value=’2′:
$("#test_select option[value='2']").remove();
сокращенно: $("#test_select [value='2']").remove();
9 — Очистить весь список <select>: $("#test_select").empty();
или: $('option', $("#test_select")).remove();
10 — Перебрать все элементы списка <select>:
$('#test_select option').each(function(){
alert(this.text);
});
11 — Сделать выбранным последний элемент:
$("#test_select :last").attr("selected", "selected");
12 — Сделать выбранным второй элемент:
$("#test_select :nth-child(2)").attr("selected", "selected");
13 — Сделать выбранным элемент, содержащий текст ‘Второе поле имеет значение value=2’:

$("#test_select :contains('Второе поле имеет значение value=2')").attr("selected", "selected");

или: $("#test_select").find("option:contains('Второе поле имеет значение value=2')").attr("selected","selected");

или только первое вхождение: $("#test_select :contains('Второе поле имеет значение value=2')").first().attr("selected", "selected");

или: $("#test_select").find("option:contains('Второе поле имеет значение value=2')").first().attr("selected","selected");

14 — Сделать выбранным элемент, value которого = 2:

$("#test_select [value='2']").attr("selected", "selected");

15 — Добавить элемент в начало списка <select>:

$("#test_select").prepend($('<option value="0">Нулевое поле имеет значение value=0</option>'));

16 — Добавить элемент в конец списка <select>:

$("#test_select").append($('<option value="7">Седьмое поле имеет значение value=7</option>'));

17 — Добавить новый элемент после второго:

$("#test_select option:nth-child(2)").after($('<option value="22">twotwo</option>'));

18 — Количество элементов option в списке <select>: $("select[id=test_select] option").size();

19 — Проверяем, выбран ли элемент в списке <select>: if($("#test_select").val())

20 — Сделать все элементы в списке <select> не выбранными:

```
$('#test_select option:selected').each(function(){
this.selected=false;
});
```

21 — Добавить несколько элементов option в список <select> из массива:

```
//добавим эти элементы несколькими способами
var newOptions = {
   "key_1": "test 1",
   "key_2": "test 2"
};
//способ 1
$.each(newOptions, function(key, value) {
   $('select#test_select').append($(new Option(key, value)));
});
//способ 2
var new_options = '';
$.each(newOptions, function(key, value) {
   new_options += '' + value + '';
});
$('#test_select').html(new_options);
//способ 3
var select = $('#test_select');
if(select.prop) {
   var options = select.prop('options');
} else {
   var options = select.attr('options');
}
$.each(newOptions, function(val, text) {
   options[options.length] = new Option(text, val);
});
```

Задание 2.

1. Создайте файл add.html и проверьте его работоспособность.

```
3  <head>
4    <title></title>
5    <script type="text/javascript">
6    function add5(){
7
8      var cep = document.createElement('p');
9      var cebr = document.createElement('br');  // создаем тег пер
10     var ctn1 = document.createTextNode('Текстовый узел1');
11     var ctn2 = document.createTextNode(' Текстовый узел2');
12
13         document.body.appendChild(cep);
14         cep.appendChild(ctn1);
15         cep.appendChild(cebr);
16         cep.appendChild(ctn2);
17
18   }
19   </script>
20 </head>
21
22 <body>
23 <input type="button" value="Добавить линию" onclick="add5();"/>
24 <select id="ss">
25
26 </select>
27 </body>
```

2. Ответьте на вопросы:

- что делает функция ***createElement***()?
- что делает функция ***createTextNode***()?
- что делает функция ***appendChild***()?
- имеет ли значение порядок выполнения функций в строках 14-16?

3. Создайте по аналогии с предыдущим задание страничку с одним текстовым полем ввода, одним пустым выпадающим списком и одной кнопочкой. При нажати на кнопочку, текст, введенный в поле ввода, должен появлятся новой строкой в выпадающем списке (должен добавляться в этот выпадающий список).

4. Создайте документ в котором по нажатию на кнопку будет добавляться абзац между вторым и четвертым (см. код ниже). После выполнения задания ответьте на вопросы:

```
3  <head>
4    <title></title>
5    <script type="text/javascript">
6    function add5(){
7      var abzac = document.createElement('p');
8      abzac.setAttribute("id", "three");
9      text = document.createTextNode('Третий абзац');
10     abzac.appendChild(text);
11
12     temp=document.getElementById('four');
13     document.getElementById('main').insertBefore(abzac,temp);
14   }
15   </script>
16 </head>
17
18 <body id="main">
19 <input type="button" value="Добавить линию" onclick="add5();"/>
20 <p id="one">Первый абзац</p>
21 <p id="two">Второй абзац</p>
22 <p id="four">Четвертый абзац</p>
23 <p id="five">Пятый абзац</p>
24 </body>
```

- за что отвечает метод ***setAttribute***()?
- для чего предназначен метод ***insertBefore***()? Что указывается в качестве аргументов метода insertBefore()?

5. Создайте (в любом графическом редакторе) пять картинок с именами 1.jpg, 2.jpg … 5.jpgДобавьте в документ абзац и кнопку. По нажатии на кнопку должны появляться картинки: вначале первая, затем вторая, заем третья… пятая, а если после пятой еще раз нажать на кнопку, то надо вывесте сообщение: картинки закончились.

6. Добавьте в четвертое задание код, который будет клонировать один из абзацев и добавлять его в конец документа:

```
16      clon = document.getElementById('one').cloneNode(true);
17      document.body.appendChild(clon);
```

7. Добавьте код, который будет заменять один элементов документа на новый, созданный элемент:

```
16      clon = document.getElementById('one').cloneNode(true);
17      document.body.replaceChild(clon, document.getElementById('five'));
```

8. Добавьте код, который будет удалять четвертый абзац в документе и результат удаления (то что вырезали) будет хранить в переменной qwerty:

```
qwerty = document.getElementById('main').removeChild(document.getElementById('four'))
```

9. Задание: создайте формочку для формирования тестов: вверху одно большое поле ввода, а внизу две кнопки: «Добавить ответ» и «Сохранить все». Так вот: кнопка «Добавить ответ» должна добавлять в документ новый абзац, в котором будет указано: «Ответ №2: 

».

10. Доработать формочку таким образом что бы кнопочка «Удалить ответ» работала.

Содержание отчета

1. Цель
2. Программные листинги (выдержки)
3. Выводы

Контрольные вопросы

1. Что такое jQuery
2. Преимущества jQuery
3. Синтаксис jQuery

Лабораторная работа №35

Программирование сложных структур с использованием JSON

Цель: получить практические навыки программирование сложных структур с использованием JSON при разработке веб-приложений; научиться создавать и анализировать JSON, и как получить доступ к данным, заблокированным внутри него.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

JSON - текстовый формат данных, следующий за синтаксисом объекта JavaScript, который был популяризирован Douglas Crockford. Несмотря на то, что он очень похож на буквенный синтаксис объекта JavaScript, его можно использовать независимо от JavaScript и многие среды программирования имеют возможность читать (анализировать) и генерировать JSON.

JSON существует как строка - полезно, когда вы хотите передавать данные по сети. Он должен быть преобразован в собственный объект JavaScript, если вы хотите получить доступ к данным. Это не большая проблема. JavaScript предоставляет глобальный объект JSON, который имеет методы для преобразования между ними.

Примечание: Преобразование строки в родной объект называется синтаксическим (*parsing)*, при преобразовании родного объекта в строку, поэтому он может быть передан через сеть, называется stringification.

Объект JSON может быть сохранен в собственном файле, который в основном представляет собой текстовый файл с расширением .json и MIME typeapplication/json.

Структура JSON

Как описано выше, JSON представляет собой строку, формат которой очень похож на буквенный формат объекта JavaScript. Вы можете включать одни и те же базовые типы данных внутри JSON, как вы можете, в стандартный объект JavaScript - строки, числа, массивы, булевы и другие объектные литералы. Это позволяет построить иерархию данных, например:

```
{
  "squadName": "Super hero squad",
  "homeTown": "Metro City",
  "formed": 2016,
  "secretBase": "Super tower",
  "active": true,
  "members": [
    {
      "name": "Molecule Man",
      "age": 29,
      "secretIdentity": "Dan Jukes",
      "powers": [
        "Radiation resistance",
        "Turning tiny",
        "Radiation blast"
      ]
    },
    {
      "name": "Madame Uppercut",
      "age": 39,
      "secretIdentity": "Jane Wilson",
      "powers": [
        "Million tonne punch",
        "Damage resistance",
        "Superhuman reflexes"
      ]
    },
    {
      "name": "Eternal Flame",
      "age": 1000000,
      "secretIdentity": "Unknown",
      "powers": [
        "Immortality",
        "Heat Immunity",
        "Inferno",
        "Teleportation",
        "Interdimensional travel"
      ]
    }
  ]
}
```

Если мы загрузили этот объект в программу JavaScript, например, проанализировали переменную, называемую superHeroes, мы могли бы затем получить

доступ к данным внутри нее, используя ту же самую нотацию точки / скобки, которую мы рассмотрели в статье JavaScript object basics. Например:

```
superHeroes.homeTown
superHeroes['active']
```

Чтобы получить доступ к данным дальше по иерархии, вам просто нужно объединить требуемые имена свойств и индексы массивов. Например, чтобы получить доступ к третьей сверхспособности второго героя, указанного в списке участников, вы должны сделать следующее:

```
superHeroes['members'][1]['powers'][2]
```

1. Сначала у нас есть имя переменной - superHeroes.
2. Внутри мы хотим получить доступ к свойству members, поэтому мы используем ['members'].
3. members содержат массив, заполненный объектами. Мы хотим получить доступ ко второму объекту внутри массива, поэтому мы используем [1].
4. Внутри этого объекта мы хотим получить доступ к свойству powers, поэтому мы используем ['powers'].
5. Внутри свойства powers находится массив, содержащий сверхспособности выбранного героя. Нам нужен третий, поэтому мы используем [2].

Примечание. Мы сделали JSON, увиденное выше, доступным внутри переменной в нашем примере JSONTest.html (см. исходный код). Попробуйте загрузить это, а затем получить доступ к данным внутри переменной через консоль JavaScript вашего браузера.

Массивы как JSON

Выше мы упоминали, что текст JSON в основном похож на объект JavaScript, и это в основном правильное. Причина, по которой мы говорили «в основном прав», состоит в том, что массив также действителен JSON, например:

```
[
  {
    "name": "Molecule Man",
    "age": 29,
    "secretIdentity": "Dan Jukes",
    "powers": [
      "Radiation resistance",
      "Turning tiny",
      "Radiation blast"
    ]
  },
  {
    "name": "Madame Uppercut",
    "age": 39,
    "secretIdentity": "Jane Wilson",
    "powers": [
      "Million tonne punch",
      "Damage resistance",
      "Superhuman reflexes"
    ]
  }
]
```

Вышесказанное вполне справедливо JSON. Вам просто нужно получить доступ к элементам массива (в его анализируемой версии), начиная с индекса массива, например [0] ["powers"][0].

Другие примечания

- JSON - это чисто формат данных - он содержит только свойства, а не методы.
- JSON требует двойных кавычек, которые будут использоваться вокруг строк и имен свойств. Одиночные кавычки недействительны.
- Даже одна неуместная запятая или двоеточие может привести к сбою JSON-файла и не работать. Вы должны быть осторожны, чтобы проверить любые данные,

которые вы пытаетесь использовать (хотя сгенерированный компьютером JSON с меньшей вероятностью включает ошибки, если программа генератора работает правильно). Вы можете проверить JSON с помощью приложения, такого как JSONLint.

- JSON может фактически принимать форму любого типа данных, который действителен для включения внутри JSON, а не только массивов или объектов. Так, например, одна строка или номер будет действительным объектом JSON.
- В отличие от кода JavaScript, в котором свойства объекта могут быть не отсортированы, в JSON в качестве свойств могут использоваться только цитируемые строки.

Практическая часть

Итак, давайте рассмотрим пример, чтобы показать, как мы можем использовать некоторые данные JSON на веб-сайте.

Начало работы

Для начала создайте локальные копии наших файлов heroes.html и style.css. Последний содержит простой CSS для стилизации нашей страницы, в то время как первый содержит очень простой HTML-код тела:

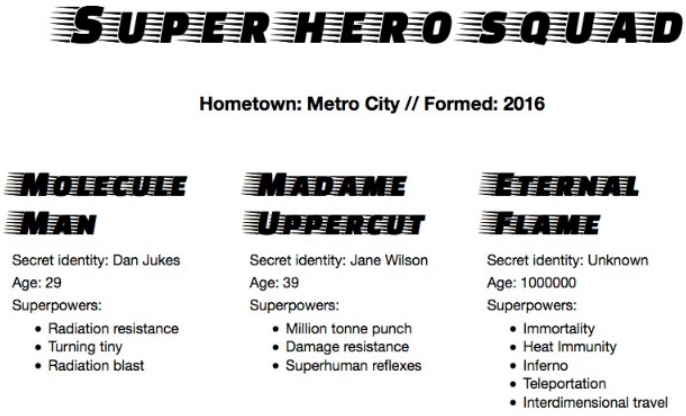
```
<header>
</header>

<section>
</section>
```

Плюс <script>, чтобы содержать код JavaScript, который мы будем писать в этом упражнении. На данный момент он содержит только две строки, которые захватывают ссылки на элементы <header> и <section> и сохраняют их в переменных:

```
var header = document.querySelector('header');
var section = document.querySelector('section');
```

Мы предоставили данные JSON на нашем GitHub, на https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json.

Мы собираемся загрузить его на нашу страницу и использовать некоторые изящные манипуляции DOM, чтобы отобразить их, например:



Получение JSON

Чтобы получить JSON, мы будем использовать API, называемый XMLHttpRequest (часто называемый **XHR**). Это очень полезный объект JavaScript, который позволяет нам делать сетевые запросы для извлечения ресурсов с сервера через JavaScript (например, изображения, текст, JSON, даже фрагменты HTML), что означает, что мы можем обновлять небольшие разделы контента без необходимости перезагрузки всей страницы. Это привело к более отзывчивым веб-страницам и звучит захватывающе, но, к сожалению, выходит за рамки этой статьи, чтобы учить его гораздо более подробно.

1. Начнем с того, что мы собираемся сохранить URL-адрес JSON, который мы хотим получить в переменной. Добавьте нижеследующий код JavaScript:

```
var requestURL = 'https://mdn.github.io/learning-area/javascript/oojs/json/superheroes.json';
```

2. Чтобы создать запрос, нам нужно создать новый экземпляр объекта запроса из конструктора XMLHttpRequest, используя ключевое слово new. Добавьте следующую ниже свою последнюю строку:

```
var request = new XMLHttpRequest();
```

3. Теперь нам нужно открыть новый запрос, используя метод open(). Добавьте следующую строку:

```
request.open('GET', requestURL);
```

Это занимает не менее двух параметров - есть другие доступные параметры. Нам нужен только два обязательных для этого простого примера:

- Метод HTTP, который следует использовать при выполнении сетевого запроса. В этом случае GET прекрасен, так как мы просто извлекаем некоторые простые данные.
- URL-адрес для запроса - это URL-адрес файла JSON, который мы сохранили ранее.

4. Затем добавьте следующие две строки: здесь мы устанавливаем responseType в JSON, так что XHR знает, что сервер будет возвращать JSON и что это должно быть преобразовано за кулисами в объект JavaScript. Затем мы отправляем запрос методом send():

```
request.responseType = 'json';
request.send();
```

5. Последний бит этого раздела предполагает ожидание ответа на возврат с сервера, а затем обращение с ним. Добавьте следующий код ниже вашего предыдущего кода:

```
request.onload = function() {
  var superHeroes = request.response;
  populateHeader(superHeroes);
  showHeroes(superHeroes);
}
```

Здесь мы сохраняем ответ на наш запрос (доступный в свойстве response) в переменной superHeroes; эта переменная теперь будет содержать объект JavaScript, основанный на JSON! Затем мы передаем этот объект двум вызовам функций - первый из них заполнит <header> правильными данными, а второй создаст информационную карту для каждого героя в команде и вставляет ее в <section>.

Мы завернули код в обработчике событий, который запускается, когда событие загрузки запускается в объекте запроса (см. onload) - это связано с тем, что событие загрузки запускается, когда ответ успешно возвращается; делая это таким образом, гарантирует, что request.response определенно будет доступен, когда мы приступим, чтобы попытаться что-то с этим сделать.

Заполнение заголовка

Теперь мы извлекли данные JSON и превратили его в объект JavaScript, давайте воспользуемся им, написав две функции, на которые мы ссылались выше. Прежде всего, добавьте следующее определение функции ниже предыдущего кода:

```
function populateHeader(jsonObj) {
  var myH1 = document.createElement('h1');
  myH1.textContent = jsonObj['squadName'];
  header.appendChild(myH1);

  var myPara = document.createElement('p');
  myPara.textContent = 'Hometown: ' + jsonObj['homeTown'] + ' // Formed: ' +
jsonObj['formed'];
  header.appendChild(myPara);
}
```

Мы назвали параметр jsonObj, чтобы напомнить себе, что этот объект JavaScript возник из JSON. Здесь мы сначала создаем элемент <h1> с createElement(), устанавливаем его textContent равным свойству squadName объекта, а затем добавляем его в заголовок с помощью appendChild(). Затем мы выполняем очень похожую операцию с абзацем: создаем его, устанавливаем его текстовое содержимое и добавляем его в заголовок. Единственное различие заключается в том, что его текст задан как конкатенированная строка, содержащая как homeTown, так и formed свойства объекта.

Создание информационных карт героя

Затем добавьте следующую функцию внизу кода, которая создает и отображает карты супергероев:

```
function showHeroes(jsonObj) {
  var heroes = jsonObj['members'];

  for (var i = 0; i < heroes.length; i++) {
    var myArticle = document.createElement('article');
    var myH2 = document.createElement('h2');
    var myPara1 = document.createElement('p');
    var myPara2 = document.createElement('p');
    var myPara3 = document.createElement('p');
    var myList = document.createElement('ul');

    myH2.textContent = heroes[i].name;
    myPara1.textContent = 'Secret identity: ' + heroes[i].secretIdentity;
    myPara2.textContent = 'Age: ' + heroes[i].age;
    myPara3.textContent = 'Superpowers:';

    var superPowers = heroes[i].powers;
    for (var j = 0; j < superPowers.length; j++) {
      var listItem = document.createElement('li');
      listItem.textContent = superPowers[j];
      myList.appendChild(listItem);
    }

    myArticle.appendChild(myH2);
    myArticle.appendChild(myPara1);
    myArticle.appendChild(myPara2);
    myArticle.appendChild(myPara3);
    myArticle.appendChild(myList);

    section.appendChild(myArticle);
  }
}
```

Для начала сохраним свойство members объекта JavaScript в новой переменной. Этот массив содержит несколько объектов, которые содержат информацию для каждого героя.

Затем мы используем for loop для циклического прохождения каждого объекта в массиве. Для каждого из них мы:

1. Создаем несколько новых элементов: <article>, <h2>, три <p> и <ul>.
2. Установливаем <h2>, чтобы содержать name текущего героя.
3. Заполняем три абзаца своей secretIdentity, age и строкой, в которой говорится: «Суперспособности:», чтобы ввести информацию в список.

4. Сохраняем свойство powers в другой новой переменной под названием superPowers - это содержит массив, в котором перечислены сверхспособности текущего героя.

5. Используем другой цикл for, чтобы прокрутить сверхспособности текущего героя - для каждого из них мы создаем элемент <li>, помещаем в него сверхспособности, а затем помещаем listItem внутри элемента <ul> (myList) с помощью appendChild().

6. Последнее, что мы делаем, это добавить <h2>, <p> и <ul> внутри <article>(myArticle), а затем добавляем <article> в <section>. Важное значение имеет порядок, в котором добавляются вещи, так как это порядок, который они будут отображать внутри HTML.

Примечание. Если вам не удается заставить этот пример работать, попробуйте обратиться к нашему исходному коду heroes-finished.html (см. также он работает в режиме live).

Примечание. Если у вас возникли проблемы после нотации точек / скобок, которые мы используем для доступа к объекту JavaScript, это может помочь открыть файл superheroes.json на другой вкладке или в текстовом редакторе и обратиться к ней, когда вы смотрите на наш JavaScript. Вы также можете обратиться к нашей статье JavaScript object basics для получения дополнительной информации о нотации точек и скобок.

Преобразование между объектами и текстом

Вышеприведенный пример был прост с точки зрения доступа к объекту JavaScript, потому что мы задали XHR-запрос для прямого преобразования ответа JSON в объект JavaScript, используя:

```
request.responseType = 'json';
```

Но иногда нам не так везет - иногда мы получаем сырую строку JSON и нам нужно будет преобразовать ее в объект самостоятельно. И когда мы хотим отправить объект JavaScript по сети, нам нужно будет преобразовать его в JSON (строку) перед отправкой. К счастью, эти две проблемы настолько распространены в веб-разработке, что встроенный объект JSON доступен в браузерах, который содержит следующие два метода:

- parse(): принимает строку JSON в качестве параметра и возвращает соответствующий объект JavaScript.
- stringify(): принимает объект как параметр и возвращает эквивалентную строковую JSON строку.

Вы можете увидеть первый в действии в нашем примере heroes-finished-json-parse.html(см. исходный код) - это делает то же самое, что и пример, который мы создали ранее, за исключением того, что мы установили XHR для возврата сырого JSON текста, затем используется parse(), чтобы преобразовать его в фактический объект JavaScript. Ключевой фрагмент кода находится здесь:

```
request.open('GET', requestURL);
request.responseType = 'text'; // now we're getting a string!
request.send();

request.onload = function() {
  var superHeroesText = request.response; // get the string from the response
  var superHeroes = JSON.parse(superHeroesText); // convert it to an object
  populateHeader(superHeroes);
  showHeroes(superHeroes);
}
```

Как вы могли догадаться, stringify() работает обратным образом. Попробуйте ввести следующие строки в консоль JavaScript браузера один за другим, чтобы увидеть его в действии:

```
var myJSON = { "name": "Chris", "age": "38" };
myJSON
var myString = JSON.stringify(myJSON);
myString
```

Здесь мы создаем объект JavaScript, затем проверяем, что он содержит, а затем преобразуем его в строку JSON, используя stringify() - сохраняя возвращаемое значение в новой переменной, а затем снова проверяем его.

Содержание отчета

1. Цель
2. Программные листинги (выдержки)
3. Выводы

Контрольные вопросы

1. Что такое JSON?
2. Что такое JSON-объект.
3. Понятие JSON-массива.
4. Структура JSON
5. В чем отличия JSON и XML
6. Какие фреймворки для работы с JSON вы знаете
7. Что лучше JSON или XML? Почему?

Лабораторная работа №36

Создание landing page с использованием фреймворка Bootstrap

Цель: получить практические навыки разработки веб-сайта с использованием фреймворка Bootstrap.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Фреймворк — это файл или несколько файлов с уже готовым кодом, которые подключаются к сайту в секции head, после чего становится возможным использование описанных в них свойств стилей и сценариев. По сути, фреймворки нужны, чтобы ускорить разработку и не создавать каждый раз «с нуля» однотипные элементы.

В чем же преимущество конкретно Bootstrap?

- этот фреймворк берет на себя кроссбраузерность и адаптивность;
- благодаря простому коду, шаблоны весят немного, отчего ваши лендинги будут загружаться достаточно быстро;
- наконец, он идеально подходит для новичков, так как его документация переведена на многие языки, в том числе и на русский.

Это главные достоинства данного фреймворка. Из недостатков можно выделить шаблонный дизайн элементов и присутствие в библиотеке большего количество кода, чем если бы сайт писался вами «с нуля». Однако обе эти проблемы решаются достаточно просто — во-первых, вы всегда можете персонализировать дизайн, если у вас есть хотя бы базовые знания верстки, а во-вторых, при сборке вы сами решаете, какие компоненты фреймворка загрузить в css-файл.

Фреймворк Bootstrap — это свободный набор инструментов для создания интерфейсов сайтов и веб-приложений. Его возможности ориентированы исключительно на фронтенд-разработку. Bootstrap — проект весьма популярный, о чём, например, говорит то, что он занимает (по состоянию на начало марта 2018-го года) второе место по количеству звёзд на GitHub.

Если вы хотите освоить Bootstrap, в частности, его самую свежую, четвёртую версию, значит, этот материал подготовлен специально для вас. Здесь, на небольшом сквозном примере, который реально освоить за полчаса, будут продемонстрированы основы Bootstrap, разобравшись с которыми вы вполне сможете сделать что-то своё, использовав этот фреймворк.

Предварительные требования

Эта материал ориентирован на начинающих веб-разработчиков, владеющих основами HTML, CSS и jQuery.

Загрузить и использовать Bootstrap можно несколькими способами. Для начала, можно воспользоваться npm. Тут понадобится такая команда: `npm install bootstrap`

Bootstrap можно подключить к странице с использованием сети доставки контента. Для этого надо добавить следующую ссылку в тег <head>:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootst
rap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/
dAiS6JXm" crossorigin="anonymous">
```

Свежую версию Bootstrap можно загрузить отсюда и использовать локально. Структура проекта должна выглядеть так, как показано на следующем рисунке.

Структура проекта

О возможностях Bootstrap 4

Первая стабильная версия Bootstrap 4 вышла в конце января сего года. Теперь Bootstrap включает в себя некоторые интересные возможности, которых не было в его предыдущей версии. А именно, если говорить об улучшениях и изменениях, можно отметить следующее:

- Bootstrap 4 переписан с использованием технологии flexbox, в то время как в Bootstrap 3 применялась технология float. Если вы не знакомы с flexbox — взгляните на этот материал.
- В Bootstrap 4, в CSS, применяются единицы измерения rem, в то время как раньше применялись единицы измерения px. Здесь можно узнать о том, чем они отличаются.
- Некоторые компоненты, такие, как панели (panels), были удалены. Тут можно найти подробности об изменениях внесённых в Bootstrap 4.
- На самом деле, в Bootstrap 4 много нового в сравнении с Bootstrap 3, если вам это нужно — можете самостоятельно ознакомиться с этими новшествами. Мы же приступаем к работе над нашим учебным проектом.

Система сеток Bootstrap

Система сеток Bootstrap (Bootstrap Grid System) предназначена для создания макетов страниц. Она упрощает разработку отзывчивых веб-сайтов. В новой версии Bootstrap не изменились имена классов (надо отметить, что класса .xs больше не существует).

Сетка разделена на 12 колонок, эта структура, настроенная так, как нужно разработчику, является основой макета страницы.

Для того чтобы использовать Bootstrap-сетку, нужно добавить класс .row к главному элементу страницы. При настройке размеров вложенных элементов используют следующие классы (вместо звёздочки в конце имени класса указывается число столбцов базовой 12-колоночной сетки, которое должен занимать конкретный элемент):

col-lg-* — класс, используемый для страниц, предназначенных для устройств с большим экраном вроде ноутбуков;

col-md-* — класс для страниц, рассчитанных на устройства с экраном среднего размера, таких, как планшеты;

col-sm-* — класс для страниц, которые рассчитаны маленькие экраны, например, такие, как у смартфонов.

Навигационная панель

Навигационные панели в Bootstrap создают с использованием класса .navbar. Фактически, это — обёртка, в которую помещают элементы, формирующие навигационную панель. Ниже показана панель, которую мы сейчас создадим. Она расположена в верхней части страницы и не исчезает при прокрутке.

Навигационная панель

Итак, для того, чтобы на странице появилась навигационная панель, добавим в index.html тег <nav> с классом .navbar, внутри которого, с использованием других классов, вроде .navbar-brand, .navbar-toggler и .nav-item, создают некоторые специальные элементы и структуру системы навигации по сайту. Класс .fixed-top позволяет зафиксировать навигационную панель в верхней части страницы. Вот разметка навигационной панели:

```
<nav class="navbar navbar-expand-lg fixed-top ">
   <a class="navbar-brand" href="#">Home</a>
   <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#na
vbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria
-label="Toggle navigation">
     <span class="navbar-toggler-icon"></span>
   </button>
<div class="collapse navbar-collapse " id="navbarSupportedContent">
     <ul class="navbar-nav mr-4">

       <li class="nav-item">
         <a class="nav-link" href="#">About</a>
       </li>
       <li class="nav-item">
         <a class="nav-link " href="#">Portfolio</a>
       </li>
       <li class="nav-item">
         <a class="nav-link " href="#">Team</a>
       </li>
       <li class="nav-item">
         <a class="nav-link " href="#">Post</a>
       </li>
       <li class="nav-item">
         <a class="nav-link " href="#">Contact</a>
       </li>
     </ul>

   </div>
</nav>
```

Теперь создадим файл main.css и подключим его к странице, поместив в тег <head> файла index.htmlследующее:

```
<link rel="stylesheet" type="text/css" href="css/main.css">
```

Это позволит настраивать стили элементов страницы, размещая в этом файле CSS-правила. Добавим в этот CSS-файл правила, задающие цветовое оформление навигационной панели:

```
.navbar{
 background:#F97300;
}
.nav-link , .navbar-brand{
 color: #f4f4f4;
 cursor: pointer;
}
.nav-link{
 margin-right: 1em !important;
}
.nav-link:hover{
 background: #f4f4f4;
 color: #f97300;
}
.navbar-collapse{
 justify-content: flex-end;
}
.navbar-toggler{
  background:#fff !important;
}
```

Новая сетка Bootstrap построена на основе flexbox, поэтому для выравнивания содержимого нужно пользоваться соответствующими свойствами. Например, для того,

чтобы поместить меню навигационной панели справа, нужно использовать свойство justify-content и установить его значение во flex-end:

```
.navbar-collapse{
 justify-content: flex-end;
}
```

Для настройки цвета фона навигационной панели можно воспользоваться классами .bg-light(светлый фон), .bg-dark (тёмный фон) и .bg-primary (основной цвет фона). Мы используем следующие настройки:

```
.bg-dark{
background-color:#343a40!important
}
.bg-primary{
background-color:#343a40!important
}
```

Шапка страницы

Для описания шапки страницы применяется тег <header>:

```
<header class="header">

</header>
```

Подготовим макет для шапки страницы. Мы хотим, чтобы она занимала всю высоту окна, поэтому тут нам пригодятся возможности jQuery. Создадим файл main.js и подключим его к index.html перед закрывающим тегом <body>:

```
<script type="text/javascript" src='js/main.js'></script>
```

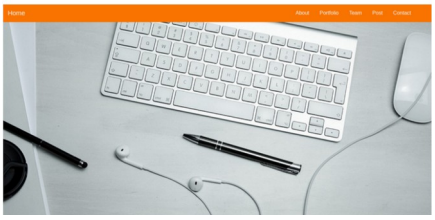
В файл main.js добавим следующее:

```
$(document).ready(function(){
 $('.header').height($(window).height());
})
```

Нелишним будет поместить в шапку страницы какую-нибудь приятную фоновую картинку. Сделаем это следующим образом:

```
/*header style*/
.header{
 background-image: url('../images/headerback.jpg');
 background-attachment: fixed;
 background-size: cover;
 background-position: center;
}
```

Вот что у нас в итоге получилось.



Шапка страницы с фоновым изображением

Пока шапка сайта выглядит пустовато, поэтому добавим в неё элемент , назначив ему класс .overlay, что приведёт к созданию блока, который расположен поверх фонового изображения шапки. Изменим тот участок файла index.html, где мы описывали шапку, следующим образом:

```
<header class="header">
  <div class="overlay"></div>
</header>
```

Затем, в main.css, добавим следующее:

```
.overlay{
 position: absolute;
 min-height: 100%;
 min-width: 100%;
 left: 0;
 top: 0;
 background: rgba(244, 244, 244, 0.79);
}
```

Теперь добавим в шапку описание проекта. Его мы поместим в новый элемент с классом .containter. Это — вспомогательный класс фреймворка Bootstrap, предназначенный для размещения содержимого с учётом нужд отзывчивого макета. Вот как изменится разметка на данном шаге:

```
<header class="header">
  <div class="overlay"></div>
   <div class="container">

   </div>

</header>
```

Теперь добавим сюда ещё один элемент , которому назначим класс .description:

```
<div class="description text-center">
   <h3><font color="#3AC1EF">
    Hello ,Welcome To My officail Website
    <p>
    cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
    proident, sunt in culpa qui officia deserunt mollit anim id est laborum.</p>
    <button class="btn btn-outline-secondary">See more</button>
   </font></h3>
  </div>
```

Этому тегу мы назначим ещё и класс .text-center, что позволит выровнять его содержимое по центру страницы. В конце описания сайта имеется кнопка. Поговорим о том, как её настроить.

Кнопки

В Bootstrap предусмотрено множество классов, предназначенных для кнопок. Посмотреть некоторые примеры оформления кнопок можно здесь. Мы, как видно в примере разметки из предыдущего раздела, добавили к элементу <button> классы .btn и .btn-outline-secondary.

Теперь настроим стили для класса .description:

```
.description{
    position: absolute;
    top: 30%;
    margin: auto;
    padding: 2em;
}
.description h1{
 color:#F97300 ;
}
.description p{
 color:#666;
 font-size: 20px;
 width: 50%;
 line-height: 1.5;
}
.description button{
 border:1px  solid #F97300;
 background:#F97300;
 color:#fff;
}
```

Вот как будет выглядеть шапка страницы после выполнения вышеописанных действий:

Шапка страницы, содержащая описание проекта

Раздел About

Для начала посмотрим на то, что мы хотим создать. Перед вами — раздел страницы со сведениями о веб-разработчике.

Раздел About

Здесь мы воспользуемся возможностями сетки Bootstrap для того, чтобы сформировать макет раздела, состоящий из двух частей. Приступим к работе, добавив к родительскому элементу раздела класс .row:

```
<div class="row"></div>
```

Первая часть макета будет расположена слева, она будет содержать фотографию. Вторая часть, расположенная справа, вместит описание.

Вот как выглядит разметка левой части этого раздела:

```
<div class="row">
 // левая часть
<div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/team-3.jpg" class="img-fluid">
    <span class="text-justify">S.Web Developer</span>
 </div>
</div>
```

А вот что получится после того, как сюда будет добавлено описание правой части макета:

```
<div class="row">
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/team-3.jpg" class="img-fluid">
    <span class="text-justify">S.Web Developer</span>
   </div>
   <div class="col-lg-8 col-md-8 col-sm-12 desc">

    <h3><font color="#3AC1EF"> D.John</font></h3>
    <p>
       ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
     tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,
     quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo
     consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse
     cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non
     proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
    </p>
   </div>
  </div>
```

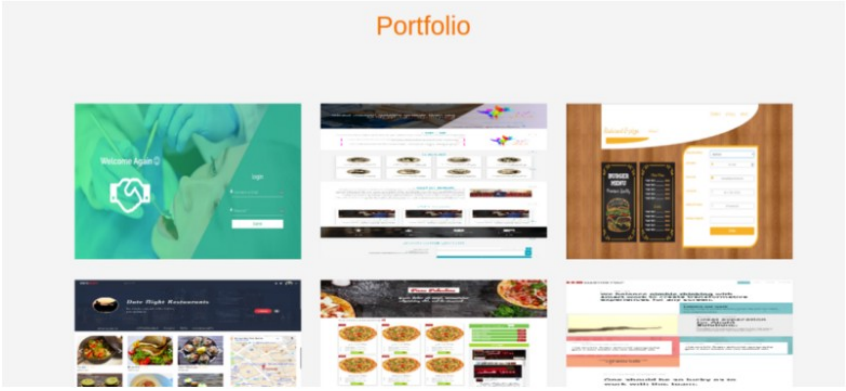
Обратите внимание на настройку ширины столбцов с использованием вышеописанных классов col-lg-*, col-md-* и col-sm-*.

Вот стили для всего этого:

```
.about{
 margin: 4em 0;
 padding: 1em;
 position: relative;
}
.about h1{
 color:#F97300;
 margin: 2em;
}
.about img{
 height: 100%;
    width: 100%;
    border-radius: 50%
}
.about span{
 display: block;
 color: #888;
 position: absolute;
 left: 115px;
}
.about .desc{
 padding: 2em;
 border-left:4px solid #10828C;
}
.about .desc h3{
 color: #10828C;
}
.about .desc p{
 line-height:2;
 color:#888;
}
```

Раздел Portfolio

Займёмся теперь разделом, в котором будет представлено портфолио разработчика. Он будет содержать галерею работ.



Раздел Portfolio

При формировании макета этого раздела применяются те же принципы работы с сеткой, которые мы рассматривали выше:

```
<!-- portfolio -->
<div class="portfolio">
     <h1 class="text-center">Portfolio</h1>
 <div class="container">
  <div class="row">
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port13.png" class="img-fluid">
   </div>
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port1.png" class="img-fluid">
   </div>
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port6.png" class="img-fluid">
   </div>
<div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port3.png" class="img-fluid">
   </div>
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port11.png" class="img-fluid">
   </div>
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/electric.png" class="img-fluid">
   </div>
<div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/Classic.jpg" class="img-fluid">
   </div>
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port1.png" class="img-fluid">
   </div>
   <div class="col-lg-4 col-md-4 col-sm-12">
    <img align="center" src="images/portfolio/port8.png" class="img-fluid">
   </div>
  </div>
 </div>
</div>
```

Добавление класса .img-fluid к каждому из изображений делает их отзывчивыми.

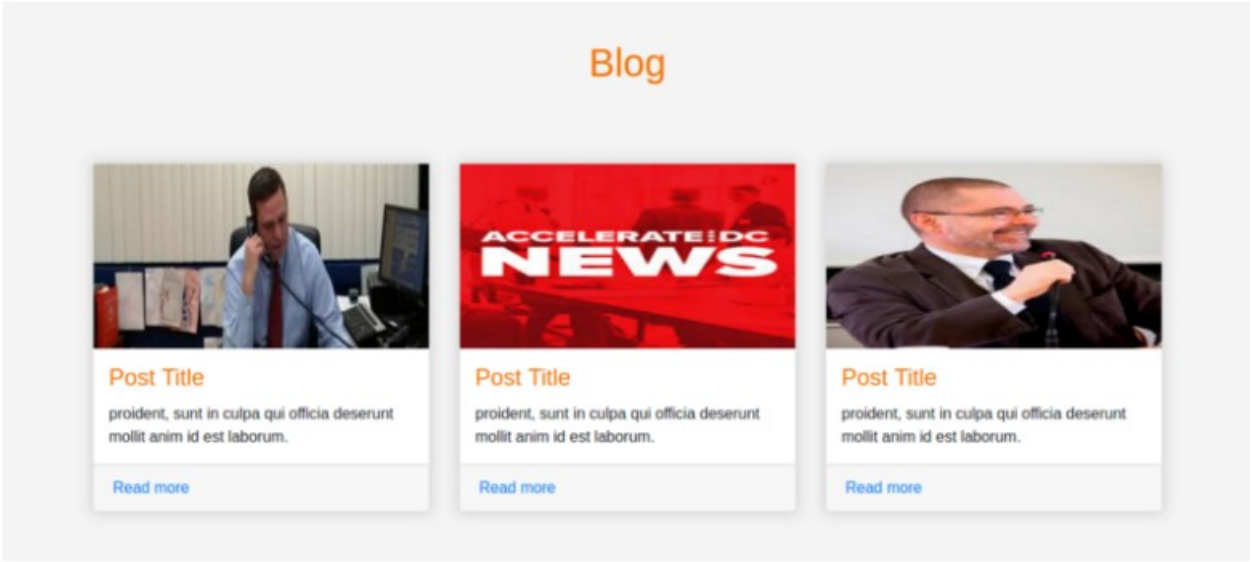
Каждый элемент в нашей галерее, на средних и больших экранах, занимает 4 колонки (напомним — класс col-sm-12 используется для устройств с маленькими экранами, класс col-md-4 используется для средних экранов, col-lg-4 — для устройств с большими экранами). Как результат, на больших и средних экранах на один элемент придётся примерно 33.3% элемента-контейнера, на маленьких устройствах каждый элемент будет занимать весь экран (12 колонок).

Стилизуем галерею работ:

```
/*Portfolio*/
.portfolio{
 margin: 4em 0;
    position: relative;
}
.portfolio h1{
 color:#F97300;
 margin: 2em;
}
.portfolio img{
  height: 15rem;
  width: 100%;
  margin: 1em;
}
```

Раздел Blog и работа с карточками

Поговорим о создании раздела, в котором содержатся анонсы материалов из блога, который ведёт наш условный веб-разработчик.



Раздел Blog

Для создания этого раздела нам понадобятся так называемые карточки (cards в терминологии Bootstrap).

Для того чтобы создать карточку, нужно включить в макет элемент и добавить к нему класс .card. Для настройки различных элементов карточки можно использовать следующие классы:

- .card-header: шапка
- .card-body: основное содержимое
- .card-title: заголовок
- .card-footer: подвал
- .card-image: изображение

HTML-разметка этого раздела будет выглядеть так:

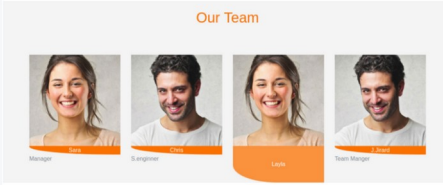
```
<!-- Posts section -->
<div class="blog">
 <div class="container">
 <h1 class="text-center">Blog</h1>
  <div class="row">
   <div class="col-md-4 col-lg-4 col-sm-12">
    <div class="card">
     <div class="card-img">
      <img align="center" src="images/posts/polit.jpg" class="img-fluid">
     </div>

     <div class="card-body">
     <h4 class="card-title">Post Title</h4>
      <p class="card-text">

       proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
      </p>
     </div>
     <div class="card-footer">
      <a href="" class="card-link">Read more</a>
     </div>
    </div>
   </div>
   <div class="col-md-4 col-lg-4 col-sm-12">
    <div class="card">
     <div class="card-img">
      <img align="center" src="images/posts/images.jpg" class="img-fluid">
     </div>

     <div class="card-body">
        <h4 class="card-title">Post Title</h4>
      <p class="card-text">

       proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
      </p>
     </div>
     <div class="card-footer">
      <a href="" class="card-link">Read more</a>
     </div>
    </div>
   </div>
   <div class="col-md-4 col-lg-4 col-sm-12">
    <div class="card">
     <div class="card-img">
      <img align="center" src="images/posts/imag2.jpg" class="img-fluid">
     </div>

     <div class="card-body">
     <h4 class="card-title">Post Title</h4>
      <p class="card-text">

       proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
      </p>
     </div>
     <div class="card-footer">
      <a href="" class="card-link">Read more</a>
     </div>
    </div>
   </div>
  </div>
 </div>
</div>
```

Вот стили для карточек:

```
.blog{
 margin: 4em 0;
 position: relative;
}
.blog h1{
 color:#F97300;
 margin: 2em;
}
.blog .card{
 box-shadow: 0 0 20px #ccc;
}
.blog .card img{
 width: 100%;
 height: 12em;
}
.blog .card-title{
 color:#F97300;

}
.blog .card-body{
 padding: 1em;
}
```

Раздел Team

В этом разделе будут размещены сведения о команде проекта.



Раздел Team

Для формирования этого раздела мы воспользуемся сеткой, поровну разделив имеющееся пространство между изображениями. Каждое изображение (на больших и средних экранах) будет занимать 3 колонки сетки, что составляет 25% общего пространства.

Вот HTML-разметка этого раздела:

```
<!-- Team section -->
<div class="team">
 <div class="container">
    <h1 class="text-center">Our Team</h1>
  <div class="row">
   <div class="col-lg-3 col-md-3 col-sm-12 item">
    <img align="center" src="images/team-2.jpg" class="img-fluid" alt="team">
    <div class="des">
      Sara
     </div>
    <span class="text-muted">Manager</span>
   </div>
   <div class="col-lg-3 col-md-3 col-sm-12 item">
    <img align="center" src="images/team-3.jpg" class="img-fluid" alt="team">
    <div class="des">
       Chris
     </div>
    <span class="text-muted">S.enginner</span>
   </div>
   <div class="col-lg-3 col-md-3 col-sm-12 item">
    <img align="center" src="images/team-2.jpg" class="img-fluid" alt="team">
    <div class="des">
      Layla
     </div>
    <span class="text-muted">Front End Developer</span>
   </div>
   <div class="col-lg-3 col-md-3 col-sm-12 item">
    <img align="center" src="images/team-3.jpg" class="img-fluid" alt="team">
     <div class="des">
      J.Jirard
     </div>
    <span class="text-muted">Team Manger</span>
   </div>
  </div>
 </div>
</div>
```

А вот — стили:

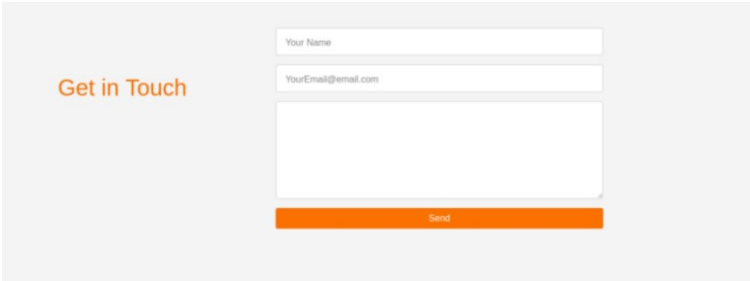
```
.team{
 margin: 4em 0;
 position: relative;
}
.team h1{
 color:#F97300;
 margin: 2em;
}
.team .item{
 position: relative;
}
.team .des{
 background: #F97300;
 color: #fff;
 text-align: center;
 border-bottom-left-radius: 93%;
 transition:.3s ease-in-out;
}
```

Украсим этот раздел анимацией, появляющейся при наведении указателя мыши на изображения. Для того чтобы достичь такого эффекта, добавим в main.css следующие стили:

```
.team .item:hover .des{
 height: 100%;
 background:#f973007d;
 position: absolute;
 width: 89%;
 padding: 5em;
 top: 0;
 border-bottom-left-radius: 0;
}
```

Форма обратной связи

В данном разделе страницы будет находиться форма, с помощью которой посетители сайта могут отправлять владельцу сайта сообщения. Тут, как обычно, для того, чтобы стилизовать элементы и обеспечить их отзывчивость, мы воспользуемся возможностями Bootstrap.



Форма обратной связи

Как и в Bootstrap 3, в Bootstrap 4 используется класс .form-control для полей ввода, но теперь тут имеется кое-что новое. Например — вместо устаревшего класса .input-group-addon используется новый класс .input-group-prepend (для значков и меток). Подробности об этом можно найти в документации к Bootstrap 4. В нашем случае каждое поле ввода будет помещено в элемент , которому назначен класс .form-group.

Добавим в файл index.html следующее:

```
<!-- Contact form -->
<div class="contact-form">
 <div class="container">
  <form>
   <div class="row">
    <div class="col-lg-4 col-md-4 col-sm-12">
      <h1>Get in Touch</h1>
    </div>
    <div class="col-lg-8 col-md-8 col-sm-12 right">
       <div class="form-group">
         <input type="text" class="form-control form-control-lg" placeholder="Your Nam
e" name="">
       </div>
       <div class="form-group">
         <input type="email" class="form-control form-control-lg" placeholder="YourEmai
l@email.com" name="email">
       </div>
       <div class="form-group">
         <textarea class="form-control form-control-lg">

         </textarea>
       </div>
       <input type="submit" class="btn btn-secondary btn-block" value="Send" name="">
    </div>
   </div>
  </form>
 </div>
</div>
```

Вот стили для раздела с формой обратной связи, которые надо поместить в файл main.css:

```
.contact-form{
 margin: 6em 0;
 position: relative;
}
.contact-form h1{
 padding:2em 1px;
 color: #F97300;
}
.contact-form .right{
 max-width: 600px;
}
.contact-form .right .btn-secondary{
 background:  #F97300;
 color: #fff;
 border:0;
}
.contact-form .right .form-control::placeholder{
 color: #888;
 font-size: 16px;
}
```

Шрифты

Стандартные шрифты подходят далеко не всем. Мы, воспользовавшись возможностями Google Font API, применим в нашем проекте шрифт Raleway. Он будет смотреться здесь очень хорошо. Для импорта шрифта добавим в файл main.css следующую директиву:

```
@import url('https://fonts.googleapis.com/css?family=Raleway');
```

Затем зададим глобальные стили для различных HTML-тегов:

```
html,h1,h2,h3,h4,h5,h6,a
 font-family: "Raleway";
}
```

Эффекты прокрутки

Для того, чтобы страница, при щелчках по ссылкам навигационной панели, плавно прокручивалась к нужному разделу, нам понадобится прибегнуть к возможностям jQuery. Если вы не очень хорошо знакомы с этой библиотекой, знайте, что ничего сложного тут нет — просто добавьте нижеприведённый код в файл main.js:

```
$(".navbar a").click(function(){
  $("body,html").animate({
   scrollTop:$("#" + $(this).data('value')).offset().top
  },1000)

 })
```

После этого добавьте атрибут data-value к каждой из ссылок в навигационной панели и приведите разметку к следующему виду:

```
<li class="nav-item">
          <a class="nav-link" data-value="about" href="#">About</a>
        </li>
        <li class="nav-item">
          <a class="nav-link " data-value="portfolio" href="#">Portfolio</a>
        </li>
        <li class="nav-item">
          <a class="nav-link " data-value="blog" href="#">Blog</a>
        </li>
        <li class="nav-item">
          <a class="nav-link " data-value="team" href="#">
          Team</a>
        </li>
        <li class="nav-item">
          <a class="nav-link " data-value="contact" href="#">Contact</a>
        </li>
```

Для того чтобы всё это, наконец, заработало, осталось лишь добавить атрибут id к основному элементу каждого из разделов страницы. При этом нужно проследить, чтобы его значение было идентично тому, которое задано в атрибуте data-value соответствующей ссылки. Например, вот соответствующий атрибут для раздела About:

```
<div class="about" id="about"></div>
```

На этом наш пример завершён. https://habr.com/company/ruvds/blog/350758/

Контрольные вопросы

1. Что такое фреймворк
2. Преимущества фрейморков

Лабораторная работа №37

Использование фреймворка для создания сайта

Цель: получить практические навыки программирование сложных структур с использованием JSON при разработке веб-приложений.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Что из себя представляют PHP-Фреймворки

Работа с Фреймворком требует определенных знаний. Но это не так страшно, как может показаться на первый взгляд.

Существует множество информативных ресурсов, дающих стартовые знания, а когда вы поймете принцип их работы от вас потребуется только желание к творчеству и экспериментированию. Фреймворк можно назвать основой для будущего приложения, он является набором готового кода, позволяющего решать задачи, поставленные перед создателями сайтов.

Самые популярные PHP Фреймворки:

Zend Framework;

Yii;

Phalcon;

Codeigniter - изучите возможности этого фреймворка у профессионалов:

Laravel;

Symfony2.

Достоинства

1. Код созданный на Фреймворках «легкий», он отличается высокой производительностью. Скорость работы выше только у приложений, написанных на чистом PHP, с индивидуальной разработкой кодов.

2. Хорошая защищенность кода. Фреймворки пишут опытные программисты, код проходит тестирование сообществом разработчиков. Недостатки и уязвимости своевременно устраняются.Возможность применения нестандартных решений значительно снижает вероятность взлома.

3. Универсальность. Используя готовые классы и библиотеки Фреймворков можно решить любую поставленную задачу.

Трудности, с которыми вы можете столкнуться

1. Новичку, первое время будет довольно тяжело разобраться в коде. Но нужно заметить, что язык PHP довольно прост в изучении и интуитивно понятен. Освоив азы, вы довольно быстро разберетесь и со сложными заданиями.

2. Для разработки сайта требуется определенное время. Хотя в библиотеках Фреймворков есть все необходимые коды, собрать из них функционирующий сайт не так уж и просто. Зато полученный результат того стоит. У вас будет уникальный, не похожий на другие, сайт.

3. В готовом сайте будет отсутствовать административный модуль. Его нужно собирать самостоятельно. Это не тяжело, если у вас есть хотя бы небольшие знания по PHP. К счастью, есть множество почти готовых решений, требующих незначительной доводки.

4. Считается, что сайты, созданные на основе Фреймворков, обходятся дорого в обслуживании. Причина кроется в уникальности. Даже опытный программист через полгода может забыть, какие коды он использовал при создании сайта. Чтобы упростить работу необходимо оставлять комментарии для каждого используемого элемента.

Системы управления контентом (CMS)

Если говорить простым и понятным языком CMS – это сердце сайта. В системе присутствует движок, отвечающий за правильную подачу данных и админ панель, позволяющая изменять конфигурацию страниц и разделов.

Основной задачей таких систем является сбор и объединение данных в единое целое. Источники при этом могут находиться как на самом ресурсе, так и вне его. Используя CMS, можно быстро создавать сайты, наполнять их содержимым.

Современные CMS - комплексное решение, имеющее огромный набор функциональных возможностей.

Самые популярные CMS:

Joomla!

Drupal;

WordPress.

Достоинства

1. Самое главное достоинство – это минимальное время, требующееся для разработки сайтов.

2. Множество готовых дополнений в виде легко устанавливаемых расширений.

3. Изучение, настройка - несложные, интуитивно понятные.

4. Дают возможность создать свой сайт, даже если вы совершенно не владеете языками программирования.

5. Схожесть устройства админ панели и функционала различных CMS упрощают работу вебмастеров.

На что нужно обратить внимание

1. У многих CMS есть некоторая ограниченность в возможностях. Встречаются системы с четко очерченными задачами. Что касается монстров как WordPress или Joomla! – здесь все решается путем установки расширений, в результате получаются сайты с большими возможностями.

2. Более низкая производительность, чем у сайтов, созданных на Фреймворках, требует больше ресурсов (не имеет значения, при мощном серверном оборудовании).

3. В некоторых случаях имеют избыточные функции. Это издержки универсальных решений, с которыми остается только смириться. К тому же никогда неизвестно что может понадобиться через месяц.

Выводы

Вариант выбора решения зависит от предполагаемого рода деятельности сайта, готовности к вложению средств в разработку или объема своих знаний.

Если вам не важна уникальность, а сайт требуется создать в кратчайшие сроки - нужна CMS.

Когда важен момент неповторимости и позволяют возможности, выбирайте вариант, при котором сайт создается на Фреймворке. В любом случае окончательное решение зависит от вас.

Задание: Установить любой PHP -фреймворк.

Описание пошаговой инсталляции фреймворка Yii с минимум необходимых настроек и созданием тестовой страницы.

Создание серьёзных интернет сайтов на фреймворках это уже давно стало нормой. Да это и понятно. Зачем всё писать с нуля если всё уже написано. Также все модули протестированы многократно что сохранит вам уйму времени. Берём готовое приложение, подключаем простыми командами и наслаждаемся прекрасной его работай. Да к то муже сайты работающие на них: быстрые и безопасные. Один из лучших фреймворков это – Yii.

В данном материале я подробно описал установку фреймворка Yii двумя способами: из архива и через Composer из Git репозитория. Ещё я описал минимальным набор необходимых настроек в конфигурационном файле. Это подключение к базе данных и “routing” (менеджер URL). И для проверки фрейморка будет создана одна

простая тестовая страница, и пункт меню с ссылкой на её. В данной статье речь будет идти о версии Yii – 1.1 так-как даже на ихнем сайте написано что она стабильная.

Фреймворк Yii сраз идёт как готовое приложение и после окончания установки его, у вас появляется практически готовый сайт, с несколькими страницами примеров, с меню, а также с рабочей формой авторизацией пользователей. Уже на этом этапе можно начинать заполнять контентом страницы по аналогии примеров. А если сделать еще несколько настроек, и подключить к базе данных, то у вас уже есть вполне функциональный и полноценный сайт.

Итак что же нужно для установки данного фреймворка?

Минимальные требования для работы с Yii это PHP не ниже 5.1 и веб-сервер Apache, ну и желательно ещё иметь базу данных MySQL или PostgreSQL.

Условно весь материал я разделил на небольшие блоки которые ниже расписал подробно:

1. Установка Yii фреймворка классическим способом из архива.
2. Установка Yii фреймворка через Composer из Git репозитория.
3. Установка Yii самого приложения (визуальная оболочка с тестовыми страницами).
4. Убираем index.php из адресной строки через файл htaccess.
5. Подключение базы данных MySQL и роутинга.
6. Создание первой тестовой страницы.
7. Добавляем пункт меню к новой созданной странице.

1. Установка Yii 1.1 фреймворка классическим способом из архива.

Скачивает дистрибутив (архив ZIP) с официального сайта фреймворка: http://www.yiiframework.com/download/

У меня получился файл с названием yii-1.1.17.467ff50.tar.gz у вас может быть другая версия и соответственно друго название.

Из архива распаковываем тольо одну папку framework и переносим в папку где будет распологатся сайт на фреймворке yii. Для данного примера я использую локальный сервер на ОС “Windows” и физически распоковал папку framework в папку yii-site, путь к которой выглядеть вот так – “C:\OpenServer\domains\yii-site\”

Если данным способом не получилось получить папку фрймворка или вы хотите версию с репозитория Git, то в следующем пункте описан процесс установки через “Composer”.

2. Установка Yii фреймворка в консоли через Composer с Git репозитория.

Первым делом устанавливаем сам клеит – менеджер пакетов Composer. На официальном сайте подробно описано как его установить, поэтому заострять внимание на этом не буду, а опушу с уже установленным composer.

Скачал файл composer.phar и положил его в папку где будет лежать сайт на yii. У меня папка локальная – “yii-site”.

Следующим действием создадим файл composer.json и в него добавляем несколько строк:

```
{
    "require": {
                    "yiisoft/yii": "dev-master"
     }
}
```

Можно указать конкретную версию:

```
{
    "require":{
        "yiisoft/yii": "1.1.16"
    }
}
```

Далее переходим в консоль (SHIFT + правая кнопка мыши и выбираем “Открыть окно команд”).

Можно сделать апдейт composer.phar (если вы запускали уже composer.phar install). в консоле набираем строчку php composer.phar update

И последнне запускаем саму инсталляцию: в консоле php composer.phar install

3. Установка Yii самого приложения (визуальная оболочка с тестовыми страницами).

Если у вас в консоль не запускается команды php то нужно прописать путь к файлу php.exe.

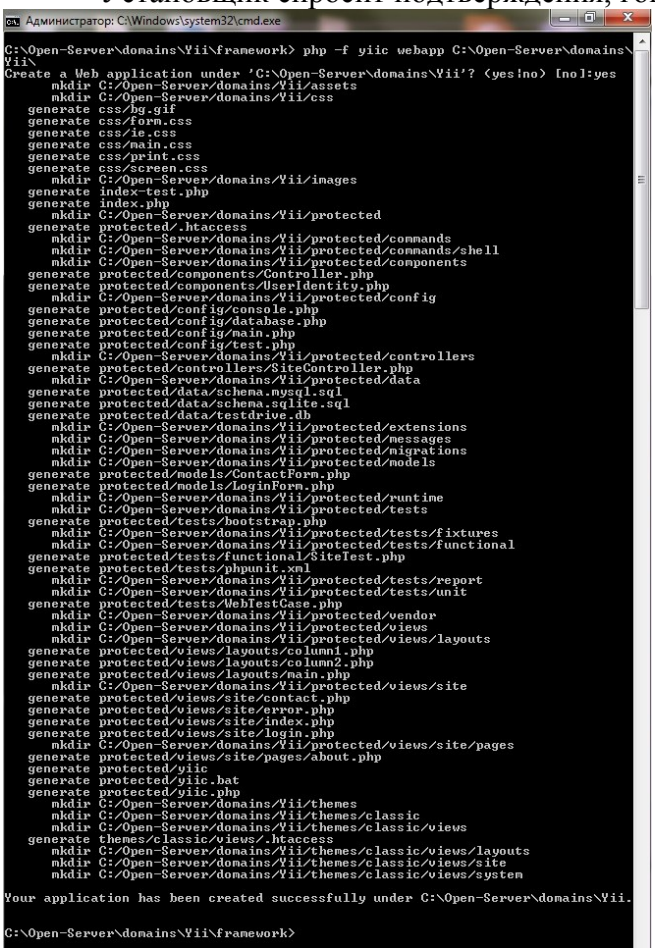
Заходим в папку распакованного фреймворка (yii-site\framework\ ) и переходим в консоль.

Далее пишим php yiic webapp C:\OpenServer\domains\yii-site\ (соответственно у вас может быть другой путь).

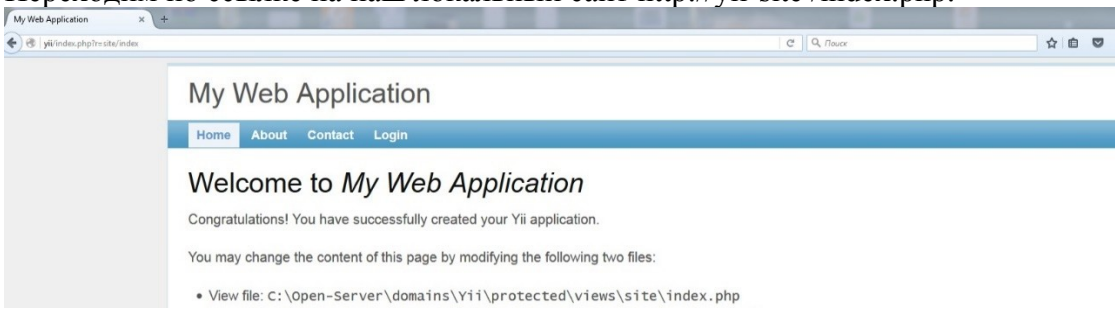
или

php -f yiic webapp C:\OpenServer\domains\yii-site\ и нажимаем Enter.

Установщик спросит подтверждения, говорим yes и Enter.



Переходим по ссылке на наш локальный сайт http://yii-site /index.php.



Если вы видите такую же картинку то это значит что фреймворк Yii1.1 установлен успешно!

4. Убираем index.php из адресной строки.

По умолчанию в адресной строке (URL) стоит слово (index.php). Его желательно убрать из адресной строки, чтобы наш URL стал более коротким и более привлекательным.

Для этого нужно сделать два действия: добавить строку в конфигурационный файл main.php в поле где указаны правила URL **'urlManager'=>array('urlFormat'=>'path',** после этой строки добавить: **'showScriptName'=>'false',**

и создать файл **.htaccess**, который должен находится в корне сайта, с содержанием следующих строк:

```
Options +FollowSymLinks
IndexIgnore */*
RewriteEngine on
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . index.php
```

Посе этой операции можно не писать (….index.php/s=?/…) а пишем только три пункта: само название сайта / контроллер / и экшен.

5. Настройка базы данных и роутинга.

Создаём базу данных. Я назвал её "yii_site".

Далее заходим в конфигурационный фал "main.php". У меня путь к ниму: "C:\ OpenServer\domains\yii-site\protected\config\".

В массиве компонентов "components" прописываем 2 массива:

```
'db'=>array(
        'connectionString' => 'mysql:host=localhost;dbname=yii_site,
        'emulatePrepare' => true,
        'username' => 'root',
        'password' => '',
        'charset' => 'utf8',
),
```

Следующим массив делает правильный роутинг. Это правила переходов по URL. Данное правило говорит что с начала идёт контроллер, потом через слеш экшен, а команда 'urlFormat'=>'path' говорит что урл будет считывается как (/…/…/).

```
urlManager'=>array(
        'urlFormat'=>'path',
        'rules'=>array(
                '<controller:\w+>/<id:\d+>'=>'<controller>/view',
                '<controller:\w+>/<action:\w+>/<id:\d+>'=>'<controller>/<action>',
                '<controller:\w+>/<action:\w+>'=>'<controller>/<action>',
        ),
```

Поумолчанию обычно включена БД лайт её следует закомментировать или удалить.

// 'db'=>require(dirname(__FILE__).'/database.php'),

6. Создание несшей страницы на сайте под yii.

Yii framework – основан по принципу MVC (модель, вью и контроллер) поэтому и логика разбросана по отдельным папкам. Поумолчанию дается контроллер "SiteController". Его мы и будем использовать в нашем примере. Он расположен в папке "C:\OpenServer\domains\yii-site\protected\controllers\" в файле "SiteController.php". В класс "SiteController" нужно добавить наш "экшен":

```php
public function actionTest()
{
        $this->render('test');
}
```

Данный экшен рендерит из файла вьюшки ‘test.php’.

Отображение мы добавим в папку этого же контроллера “C:\OpenServer\domains\yii-site\protected\views\site\”. Создадим наш файл “test.php”.

В файл test.php можно добавлять любой контент. В в нём могут работать все PHP функции, а также и все доступные встроенные методы Yii. Смело пишим (Hello World!).

Данную страницу теперь можно посмотреть набравши в адресной строке – http://yii-site/site/test

7. Добавим пункт меню нашей созданной странице страницы

По умолчанию, если мы подключаем встроенный layouts, то верхнее меню описано в файле C:\OpenServer\domains\yii-site\protected\views\layouts\main.php. В нём прописан виджет в котором вписаны все видимые пункты меню:

```
$this->widget('zii.widgets.CMenu',array(
'items'=>array(
….
array('label'=>'About', 'url'=>array('/site/page', 'view'=>'about')),
array('label'=>'Contact', 'url'=>array('/site/contact')),
….
));
```

Мы добавим наш пункт меню в самый низ посе других пунктов: array(‘label’=>’Моя тестовая страница’, ‘url’=>array(‘/site/test’)),

После этого появится наш пункт меню который будет виден со всех страниц.

Содержание отчета

1. Цель
2. Программные листинги (выдержки)
3. Выводы

Контрольные вопросы

1. Что такое фреймворк
2. Преимущества фрейморков
3. Что такое CMS, как они работают?

Лабораторная работа №38
Создание сайта на CMS

Цель: получить практические навыки создания сайта на CMS WordPress.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Что такое CMS?

CMS, от английского Content Management System (система управления контентом), - это программное обеспечение, позволяющее пользователям размещать или изменять уже размещенную на сайте информацию без привлечения разработчиков сайта. Это значит, что пользователю не обязательно обладать навыками программирования или знаниями языка HTML, чтобы, например, опубликовать на своем сайте новость, статью или добавить изображение. Часто наряду с термином CMS можно услышать также термин "движок сайта", которым обычно пользуются веб-мастера в своем профессиональном сленге. Большинство современных CMS имеют модульную архитектуру, что позволяет администратору самому выбирать и настраивать те компоненты, которые ему необходимы.

Состоят CMS обычно из двух частей:

- back-office - это часть системы, отвечающая за функциональность и хранение информации;
- front-office - это часть системы, обеспечивающая интерфейс с пользователем.

Как работают CMS?

Принцип работы всех CMS основан на разделении контента (содержания) и дизайна (оформления) сайта. Обычно дизайн сайта меняется редко, тогда как изменения контента могут происходить не только каждый день, но и даже каждый час. Поэтому в своей работе CMS используют так называемые шаблоны - специальные "пустые" заготовки страниц, в которых дизайн сайта уже прописан и осталось лишь наполнить их информацией. Пользователю достаточно воспользоваться специальным WYSIWYG-редактором. Этот редактор по внешнему виду очень похож на привычные текстовые редакторы офисных приложений, поэтому пользователю не составляет особого труда освоить его. А наличие в системе большого количества готовых шаблонов дает возможность выбрать подходящий дизайн буквально в считанные минуты.

Сайты, организованные посредством системы управления контентом, основаны на следующих технологиях: веб-сервер, хранилище данных (зачастую СУБД, например такие как MySQL или PostgreSQL, и другие), веб-приложение для обеспечения работы самой системы, визуальный редактор страниц, файловый менеджер с веб-интерфейсом для управления файлами сайта, система управления правами пользователей и редакторов сайта.

Информация хранится в базе данных, например, в MySQL и вызывается из нее при загрузке страниц сайта.

Работа CMS не требует установки дополнительного ПО, поскольку сама система находится на сервере, а доступ к ней осуществляется через обычный интернет-браузер.

Что такое Denwer?

Denwer, по-другому его называют Джентельменский набор WEB разработчика. Денвер это программа, предназначенная для имитации WEB сервера на домашнем компьютере. Используется она для тестового запуска и отладки WEB сайтов. На Denver можно запустить практически любой сайт.

В стандартную комплектацию этого так называемого программного комплекса входят: WEB сервер Apache, интерпретатор языка программирования PHP, интерпретатор языка PERL, база данных MySQL, имитация сервера Email почты, по умолчанию встроен движок phpMyAdmin для управления базами данных MySQL. Это всё что нужно чтобы запустить практически любую современную CMS систему.

Для создания сайта с помощью CMS необходимо иметь возможность размещения его на сервере. Хостинг – это услуга по предоставлению ресурсов для размещения информации на сервере, постоянно находящемся в сети (обычно Интернет). Обычно хостинг входит в пакет по обслуживанию сайта и подразумевает как минимум услугу размещения файлов сайта на сервере, на котором запущено программное обеспечение, необходимое для обработки запросов к этим файлам (веб-сервер). Как правило, в обслуживание уже входит предоставление места для почтовой корреспонденции, баз данных, DNS, файлового хранилища на специально выделенном файл-сервере и другое, а также поддержка функционирования соответствующих сервисов.

Для обучения развертыванию сайта с поомщью CMS можно использовать локальный сервер, который будет установлен на ваш компьютер. После создания сайта его можно будет перенести на любой другой сервер.

Рассмотрим создание сайта с помощью CMS на примере WordPress.

WordPress - это мощная платформа для создания персонального сайта или официального сайта организации. Она содержит отличный набор возможностей для того, чтобы максимально упростить процесс создания онлайн-публикаций, сделать его приятным и удобным. Распространяется на бесплатной основе под лицензий GNU GPL. Написана на языке PHP, использует базы данных MySQL. Официальная дата создания – 12 июня 2001 года. Последняя на сегодняшний день версия – 3.0 с системой мультиблогинга.

Особенности

- Wordpress устанавливается за несколько минут. Достаточно загрузить дистрибутив на FTP сервер, указать данные доступа к базе данных и выполнить настройку учетной записи администратора. CMS переведена на большую часть языков. Имеется официальный русский дистрибутив.
- Wordpress поддерживает технологии RSS, Ping, Trackback, ATOM и ЧПУ.

Расширяется с помощью плагинов, для нее разработано большое количество шаблонов и виджетов. Удобная архитектура CMS позволяет создавать визитки, блоги, магазины, галереи, социальные сети, портфолио и другие проекты.

- Большинство шаблонов и расширений для wordpress, в том числе распространяемых бесплатно, представлены в репозитории, поиск по которому осуществляется из панели администратора.

Достоинства

- контент сайта доступен при использовании обычного браузера. Основным редактором wordpress является модель WYSIWYG (от анг. What You See Is What You Get). При создании и изменении страницы web-мастер сразу видит конечный результат.
- меню и страницы меняются автоматически при добавлении новых материалов.
- для wordpress написано более 7 тысяч различных плагинов. Модуль достаточно закачать на сайт через FTP и активировать при помощи web-интерфейса.
- в ходе продвижения сайта его дизайн можно изменять любое количество раз, не оказывая влияния на CSM.
- посетители сайта, созданного на wordpress, могут оставлять комментарии к статьям, что улучшает интерактивность ресурса (взаимодействие авторов и целевой аудитории).
- сайт на wordpress может иметь любое количество страниц, что упрощает работы по его поисковой оптимизации. Каждой записи может быть присвоена определенная рубрика, для чего предусмотрена отдельная закладка. Категории могут иметь различную

древовидную иерархию. Для дополнительной тематической группировки используются метки.

Для безопасности сайта и снижения нагрузки на сервер для wordpress написан антивирусный плагин, предусмотрено кэширование данных. Различные уязвимости устраняются разработчиками достаточно оперативно.

Установка локального сервера на компьютер

Для использования WordPress на своем компьютере, сначала нужно скачать бесплатное программное обеспечение XAMPP. Это среда PHP разработки, хоть и есть много других возможностей для использования WordPress локально, но это является лучшим.

XAMPP - кроссплатформенная сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер.

Apache – это один из самых популярных веб-серверов. Веб-сервер – сервер, принимающий запросы от клиентов, обычно веб-браузеров, и выдающий им ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-потоком или другими данными.

MySQL – это одна из самых популярных и самых распространенных СУБД (система управления базами данных) в интернете. Она не предназначена для работы с большими объемами информации, но ее применение идеально для интернет сайтов, как небольших, так и достаточно крупных. MySQL отличатся хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей. Поддержка сервера MySQL автоматически включается в поставку PHP. Немаловажным фактором является ее бесплатность. MySQL распространяется на условиях общей лицензии GNU.

PHP— скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

Практическая часть. Вариант 1

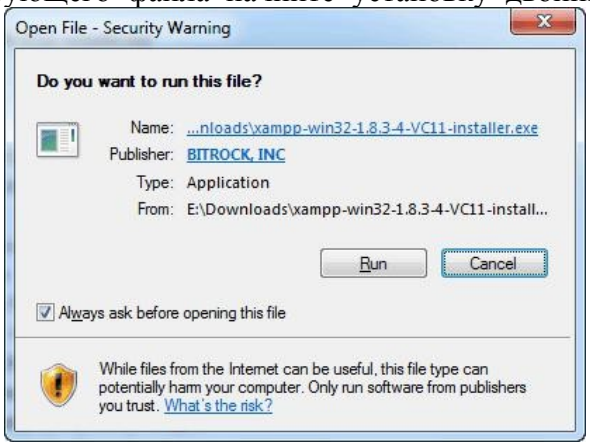
Шаг 2. устанавливаем XAMPP.

1. Загружаем XAMPP по ссылке https://www.apachefriends.org/ru/index.html

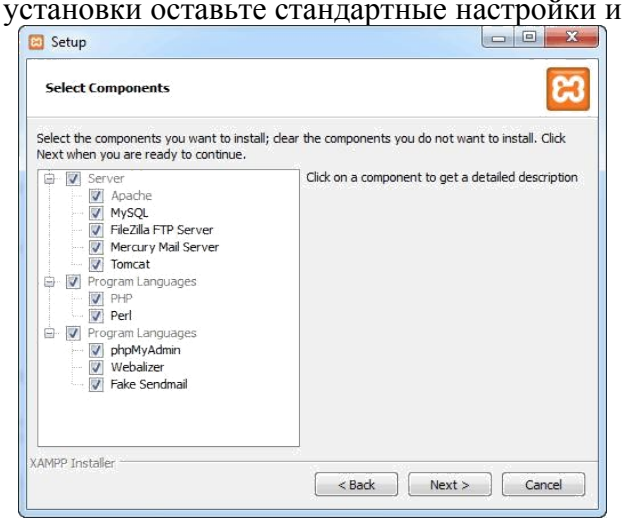
Выберите подходящий вариант, в зависимости от операционной системы на вашем компьютере и скачайте файл.

- Устанавливаем XAMPP

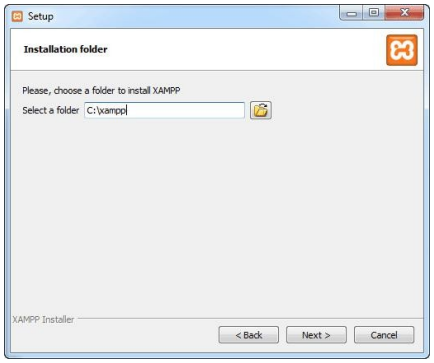
После загрузки соответствующего файла начните установку двойным щелчком мыши.



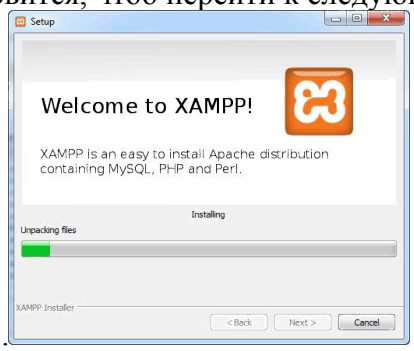
В процессе установки оставьте стандартные настройки и нажмите «Далее».



Когда дойдет до выбора папки для сохранения XAMPP, можно выбрать папку по умолчанию либо создать новую. Если вы не знаете, как лучше сделать, выберите первый вариант.



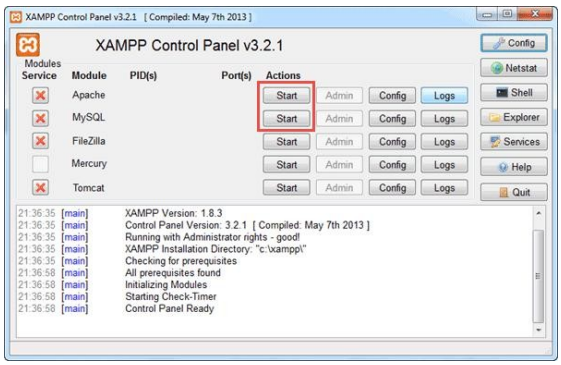
Ждем, пока XAMPP установится, чтоб перейти к следующему шагу



Шаг 3. Запускаем XAMPP

Когда XAMPP установлен на ваш компьютер, выберите «Начать загрузку панели управления сейчас» и нажмите «Готово».

После этого нажмите «Start» для Apache и MySql:



На этом этапе обычно приходит оповещение о системе безопасности от Windows, но вы можете ни о чем не беспокоиться. Убедитесь, что вы нажали на «Разблокировать», и ваш локальный сервер XAMPP будет готов к использованию.

Чтоб проверить, все ли сделано правильно, откройте браузер и перейдите по этому адресу: http://localhost

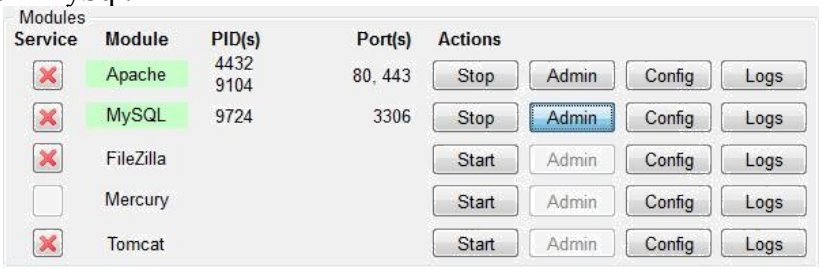
Если XAMPP настроен правильно, то вы должны увидеть следующее:

Если такого изображения не появляется, попробуйте отключить антивирусную программу и снова нажать на http://localhost.

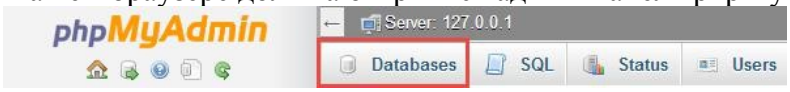
Шаг 3. Создаем базу данных для WordPress

Так как WordPress использует базу данных, то нам нужно ее создать перед установкой WordPress. Это делается очень просто и всего в несколько шагов.

Для начала переключитесь на панель управления, нажмите на кнопку «Админ», а затем на «Старт» для MySql:



После этого в вашем браузере должна открыться админ-панель phpMyAdmin:

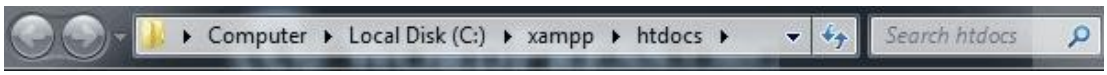


Здесь нажмите на «Databases», введите название вашей БД (любой набор латинских символов) и нажмите на кнопку «Создать». Так как вы можете создать несколько БД и произвести несколько установок WordPress на свой компьютер. Главное, дайте такое имя для БД, чтобы потом не перепутать, к какому сайту оно относится.

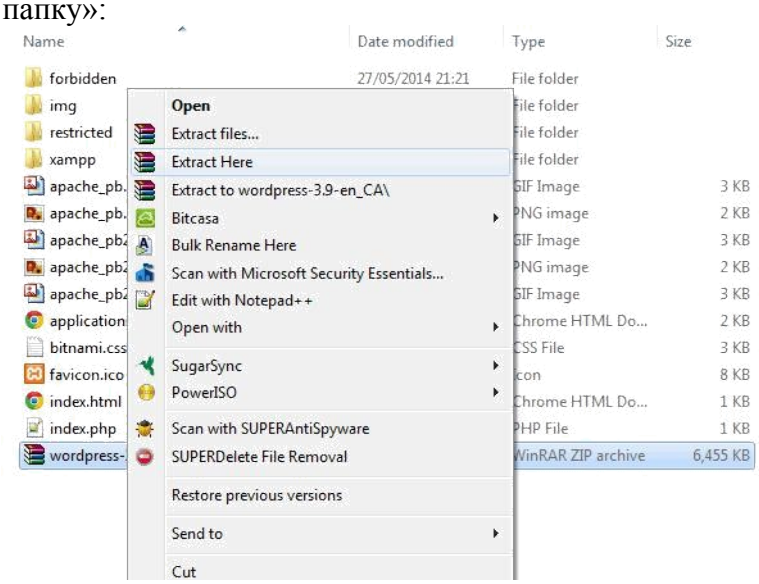
Когда вы увидите сообщение, подтверждающее создание базы данных, можете переходить к следующему шагу.

Шаг 4. Загружаем WordPress

Перейдите на WordPress.org и скачайте последнюю версию WordPress. После загрузки zip-файла, перенесите его туда, где вы установили XAMPP и сохраните в папке «htdocs».



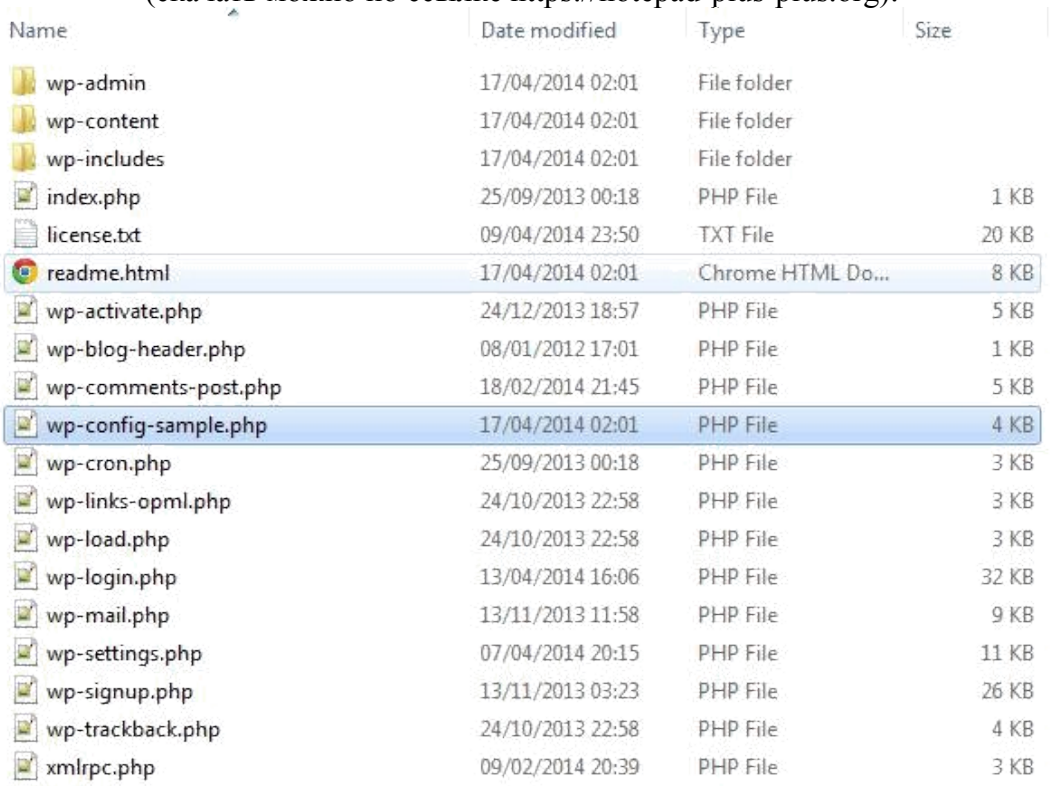
Затем разархивируйте этот файл, кликнув на него правой кнопкой мыши и выбрав «Извлечь в текущую папку»:



Можете дать папке уникальное название, чтоб можно было отличить ее от других, которые вы будете создавать в дальнейшем.

Шаг 5. Настраиваем WordPress

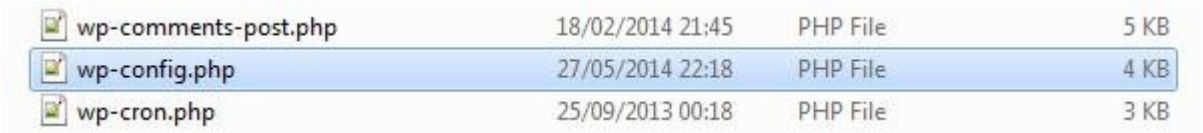
Мы практически все сделали. Осталось только отредактировать и переименовать один файл. В каталоге WordPress, который вы только что создали, найдите файл wp-config-sample.php и откройте его с помощью текстового редактора, например Notepad++ (скачать можно по ссылке https://notepad-plus-plus.org):



Отредактируйте данные в выделенной области в соответствии с названием базы данных, которую вы создали в шаге 3.

```
// ** MySQL settings - You can get this info from your web host **
/** The name of the database for WordPress */
define('DB_NAME', 'wordpresslocal');

/** MySQL database username */
define('DB_USER', 'root');

/** MySQL database password */
define('DB_PASSWORD', '');

/** MySQL hostname */
define('DB_HOST', 'localhost');
```

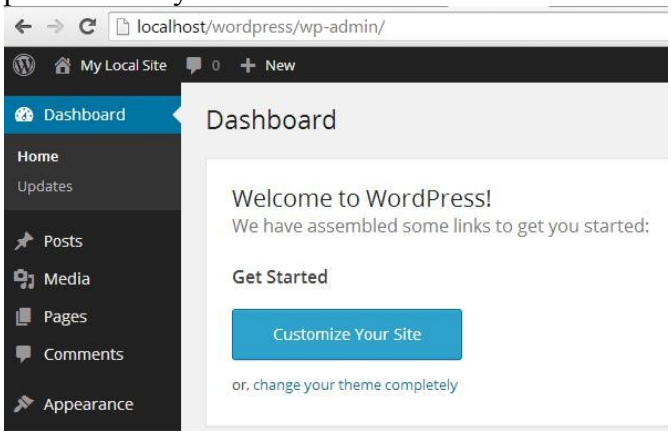
Сохраните файл и переименуйте его на wp-config.php, удалив –sample из его названия.



Шаг 6. Устанавливаем WordPress

Чтобы установить WordPress, введите в браузере адрес: http://localhost/WordPress/, где «WordPress» является названием папки, которую вы создали в шаге 5. После загрузки страницы отобразится следующее

Затем введите необходимые данные, нажмите на «Установить WordPress», после чего на вашем компьютере появится установленный WordPress в автономном режиме.



Советы по использованию WordPress в автономном режиме

Если вы хотите установить темы или плагины, вы можете сделать это через админку WordPress, как бы вы это делали в онлайн режиме, или же можно скопировать файл в соответствующие папки на вашем компьютере:

/ Темы: c:\xampp\htdocs\wordpress\wp-content\themes

/ Плагины: c:\xampp\htdocs\wordpress\wp-content\plugins

Если вы не можете получить доступ к WordPress автономно по этим ссылкам, убедитесь, что XAMPP активен и запущены Apache и MySql.

Шаг 7. Техническая настройка WordPress после установки

Настройку WordPress можно условно разделить на два этапа – техническая настройка (настройка административной консоли, настройка человекочитаемых ссылок (ЧПУ), sitemap.xml, robots.txt) и визуальная настройка (установка плагинов для отображения фото и видео, настройка постраничной навигации и «хлебных крошек»).

1. Удаляем лишнюю информацию в WordPress

После того, как вы установили CMS WordPress на хостинг и вошли в административную панель (введя свой логин и пароль, заданные при регистрации) – вам необходимо удалить лишнюю информацию, которая автоматически задается системой в качестве предустановки. Чтобы войти в админскую панель наберите в браузере: http://site.ru/wp-login.php.

Речь идет о приветственной записи, комментарии к этой записи и приветственной странице. Удалить их не сложно: в левой колонке у вас есть вкладки «Записи» и «Страницы». Достаточно подвести мышкой к этим записям и страницам и у вас высветится контекстное меню, в котором будет присутствовать слово «удалить». Его и нажимаем для удаления приветственной записи, страницы и комментария к записи.

Также в разделе «Записи – Рубрики» у вас будет предустановленная рубрика «Без рубрики». Ее удалить нельзя, но можно переименовать (но это лучше сделать позже, после настройки ЧПУ, поскольку рубрика иначе может работать не корректно).

/ Установка базового «seo-пакета» плагинов для WordPress

Что такое плагин для WordPress? Это специальные программы-расширения, позволяющие оптимизировать ваш сайт (блог), расширить его функционал и возможности, улучшить внешний вид сайта как в глазах посетителей, так и поисковых систем.

У каждого вебмастера, использующего WordPress в качестве CMS для сайта – свой собственный набор плагинов, которые подлежат обязательной установке. Здесь речь идет обазовом наборе плагинов.

RusToLat – плагин позволяет настраивать внешний вид URL ваших записей и страниц, придавая им «человечность» и «читабельность». В общем то, поэтому адреса, обработанные таким плагином называются ЧПУ. Адрес страниц в браузере – адрес страницы соответствует названию статьи и прописан латиницей через дефисы.

Google XML Sitemaps – плагин позволяет создавать в автоматическом режиме карту вашего сайта в формате .xml, что позволяет передавать поисковым системам содержание вашего сайта и упрощать процедуру индексации сайта поисковыми роботами.

Platinum SEO Pack – позволяет задавать keywords и description для каждой записи (страницы), главной страницы сайта (блога), управлять индексацией отдельных записей в блоге.

Эти три плагина – достаточно мощный инструмент, позволяющий настроить ваш сайт (блог). В дальнейшем могут понадобиться дополнительные плагины.

Настройка отображения информации в CMS WordPress

После того как установили и активировали основные плагины – перейдите во вкладку левого меню CMS WordPress «Параметры».

3.1. общие – в этой вкладке вам необходимо задать название сайта и его описание (изменить, если эти данные задавались про установке WordPress на хостинг), указать как отображать ваш сайте в результатах поиска и браузерах (с www или без www), указать свой часовой пояс, формат даты и времени отображения записей на страницах блога (сайта);

3.2. чтение – в этой вкладке вы можете задать какую из страниц вашего сайта отображать на главной, какая из страниц сайта (блога) будет выполнять функции «ленты последних записей», настроить количество выводимых на странице записей, количество записей поступающих в RSS-ленту;

3.3. обсуждение – в этой вкладке вы можете разрешить или запретить комментирование ваших записей (страниц), а также формат публикации соответствующих комментfриев к постам (записям на сайте);

3.4. постоянные ссылки – в этой вкладке вы можете настроить внешний вид адресов ваших страниц (желательно настраивать после активации плагина RusToLat).

Если вы хотите, чтобы адреса ваших страниц отображались в формате www.site.ru/nazvanie-stranicy/ - введите в поле «Произвольно» следующую строку: /%postname%/

Если вы хотите придать страницам олд-скульный вид и «постоянство» – .html на конце записей – введите в поле «Произвольно» следующую строку: /%postname%.html

3.5. приватность. Пока вы не окончили настройку вашего сайта (блога), рекомендую установить флажок на пункте «Попросить поисковые системы не индексировать сайт». Нежелательно, чтобы поисковые системы видели сайт недоделанным – первое впечатление может быть плохим во всех смыслах этого слова.

Желательно сайт полностью настроить, добавить контент и только после этого открывать его для индексации.

4. Настройка robots.txt для WordPress

CMS WordPress как и многие другие аналогичные платформы – создает внутренние дубли страниц, от которых желательно избавляться.

Шаг 8. Первые шаги работы в WordPress (визуальная настройка)

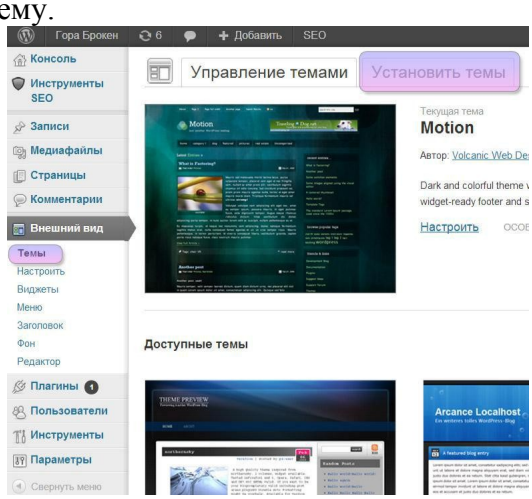
1. Вход в административную панель

Для входа в панель управления нужно написать после названия сайта в строке ввода веб адреса интернет обозревателя wp-login.php. Ввести ранее зарегистрированные данные – имя пользователя и пароль, затем нажать войти.

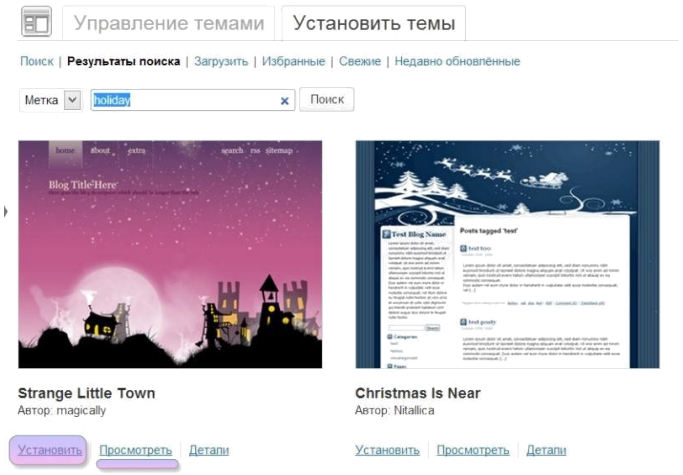
После чего, открывается консоль управления своим сайтом.

2. Настраиваем внешний вид сайта

Для этого заходим в панель администратора **WordPress** и в левом меню нажимаем **Внешний вид -> Темы,** далее заходим **Установить темы** и с помощью фильтров подбираем светлую, темную или любую другую тему. На скриншоте выбрана тёмная. И нажимаем на кнопку найти тему.



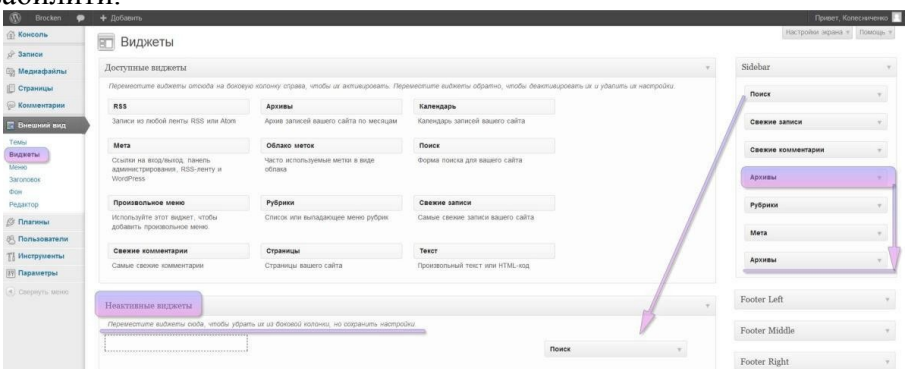
Перед установкой шаблона можно его предварительно просмотреть. После того как определитесь с темой, нажмите на ссылку установить, а затем активировать, теперь выбранная тема действует. Проверьте, зайдите на сайт.



Вообще для большинства сайтов лучше выбирать светлые темы, где обычно черным шрифтом текст расположен на белом фоне, так его легче читать. Желательно чтобы тема простая, без лишних объектов.

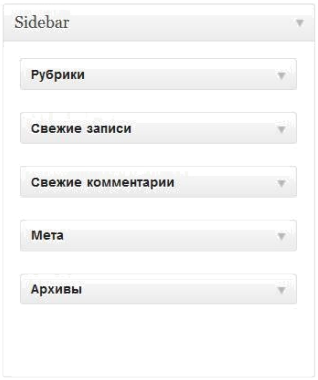
3. Настраиваем виджеты

Далее настроим виджеты. Виджеты – это такие вспомогательные меню, которые находятся обычно в сайдбаре (в правой или левой части сайта). Они отображают свежие записи, новые комментарии, архивы, рубрики и другие полезные элементы меню так называемого юзабилити.



Настройте виджеты, как считаете нужным (удалить, поменять местами, добавить). Например, можно убрать поиск из сайтбара, так как он уже есть в шапке сайта сверху. Все остальные виджеты пока можно оставить. Чтобы поменять местами виджеты, возьмите любой виджет из сайдбара и нажимайте на него левой кнопкой мышки и не отпуская перетащите в то место или очередность какую нужно. Изменения появятся сразу на сайте.

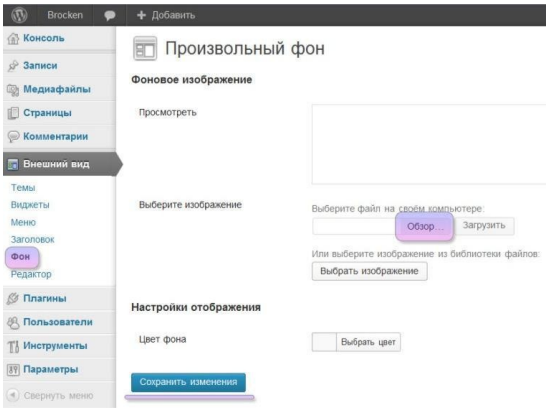
Может быть так:



- Рубрики – это перечисление ваших категорий, к которым относятся статьи
- Свежие записи – перечисление новых поступивших статей.

- Свежие комментарии – новые комментарии.
- Мета – это список тегов/меток относящихся к тем или иным статьям или темам.
- Архивы – количество статей за определенный месяц или срок.

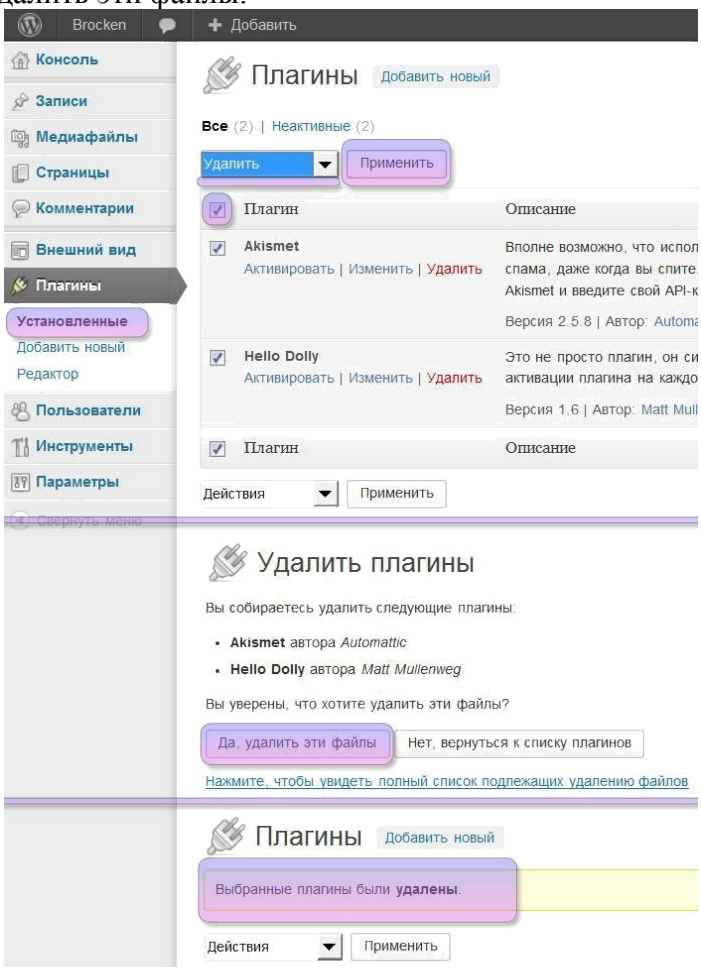
В любом случае, все можно поменять двумя кликами. Также в разделе Фон слева – можно выбрать любой подходящий фон для сайта. Для этого нужно нажать Обзор и загрузить файл с компьютера. Это можно сделать если в шаблоне нет фона или нужно поменять фон.



4. Настраиваем плагины

Следующее что нужно сделать — это установить некоторые нужные плагины (часть из них рекомендовались выше). Плагин – это специальное дополнение или расширение для сайта, набор новых функций или возможностей. Много плагинов устанавливать не рекомендуется, они могут тормозить сайт.

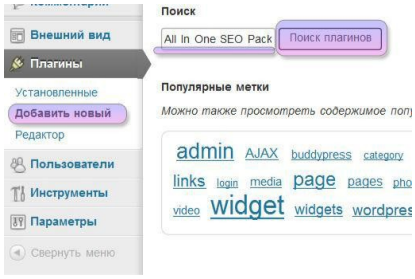
Плагины установленные по умолчанию можно удалять. Для этого их нужно выделить и в всплывающем меню Действия выбрать удалить и нажать Применить. После нажать на кнопку Да, удалить эти файлы.



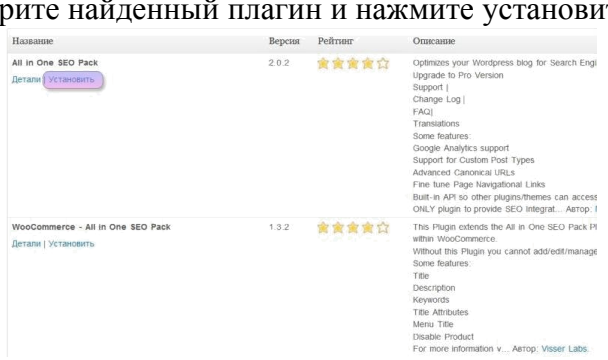
После удаления, установим новые плагины. Установим три плагина:

- ❑ All In One SEO Pack – этот плагин прежде всего нужен для установки заголовков, описания и ключевых слов к вашим статьям для поисковых систем;
- ❑ CheckBot – защищает от спамеров в комментариях ("капча");
- ❑ The Simplest Favicon – установка маленькой иконки 16*16 мм. Она отображается рядом с названием сайта в некоторых поисковиках. Например таких как Яндекс. А также отображается в заголовках шапки браузеров.

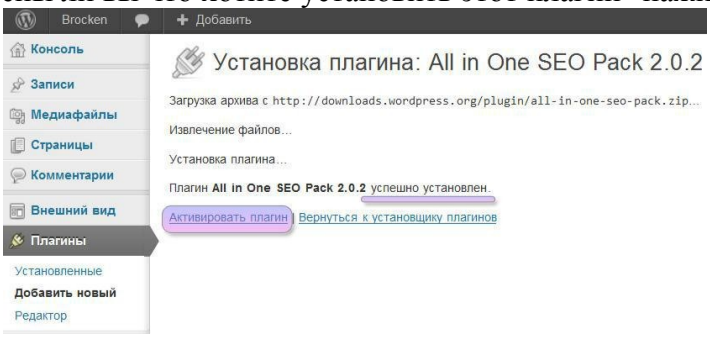
Нажимаете на Добавить новый и в форму поиска впишите название и нажимите на кнопку Поиск плагинов.



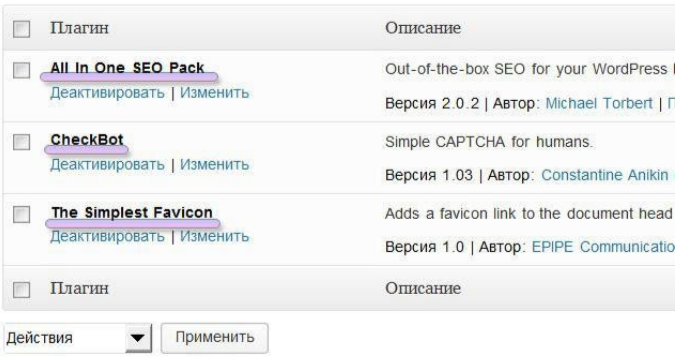
В результатах поиска выберите найденный плагин и нажмите установить.



На вопрос "уверены ли вы что хотите установить этот плагин" нажимите ОК.



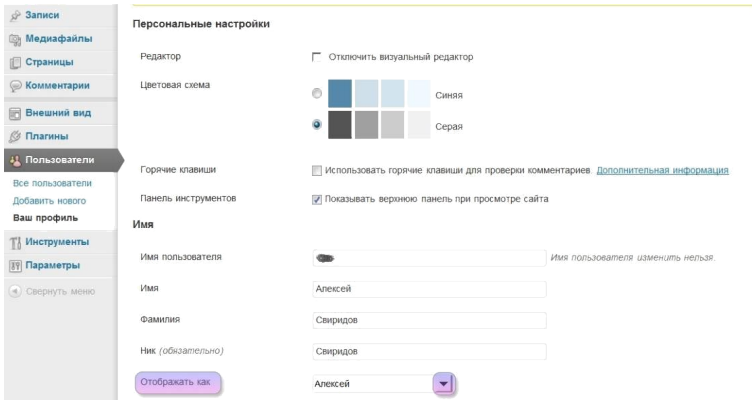
После успешной установки – нажимите **Активировать плагин**. Далее таким же способом устанавливаем остальные два плагина.



5. Настройка пользователей

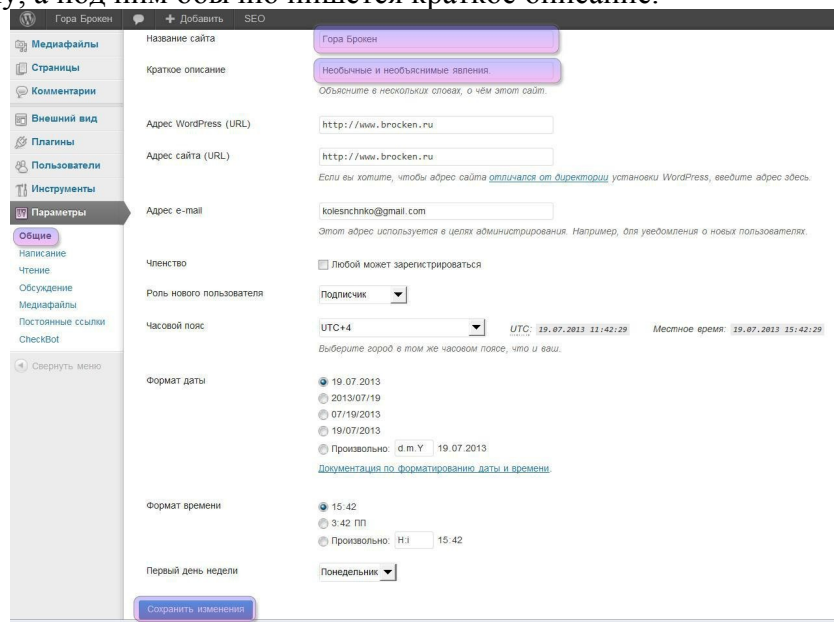
Следует зайти в раздел пользователи для того чтобы дополнить или изменить свой профиль. Тут можно добавить свои данные, такие как имя, фамилию, ник и другие. И можно выбрать, как будет отображаться на сайте имя автора статьи.

Добавьте еще пользователя, который будет модератором сайта, если это необходимо.



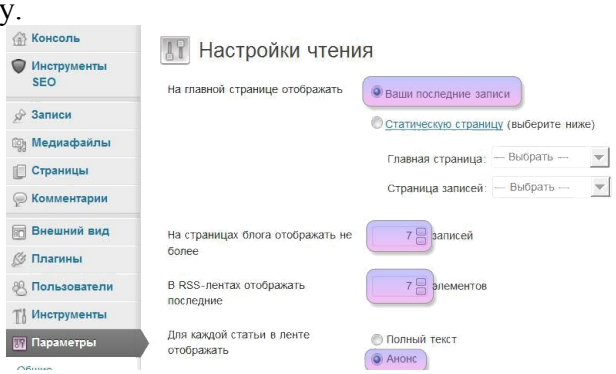
6. Настройка параметров

Нажмите вкладку Параметры, далее первую вкладку Общие. Поменяйте название сайта и краткое его описание. Название сайта - это то название, которое высвечивается в шапке сайта сверху, а под ним обычно пишется краткое описание.



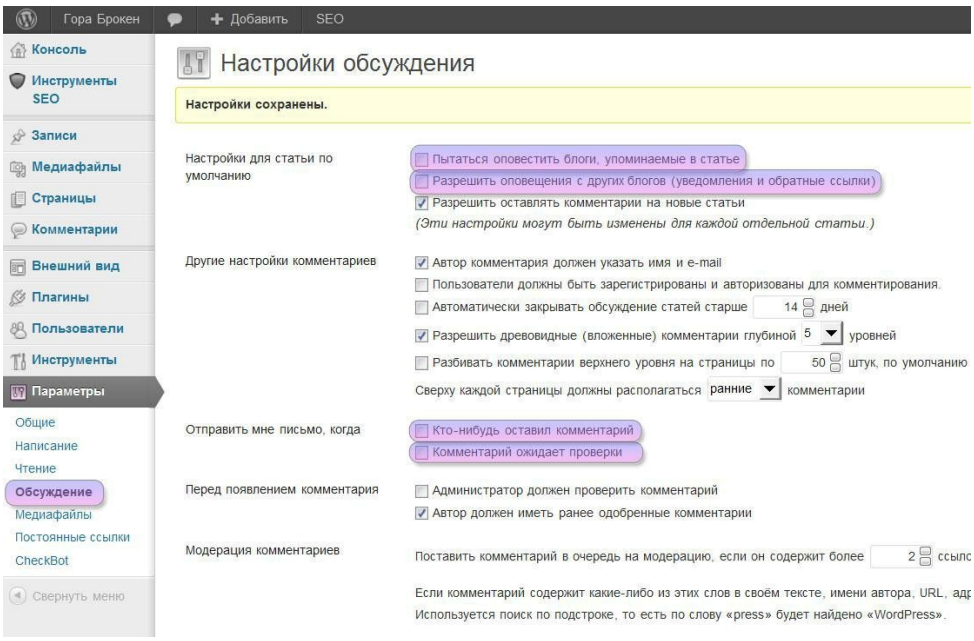
Тут можно поменять свой адрес электронной почты, который используется в целях администрирования. Формат даты, времени, часовой пояс и другие параметры..

В разделе Чтение можно настроить стиль и хронологию отображения статей. Например, есть настройка которая выводит последние записи, а можно сделать статическую страницу.



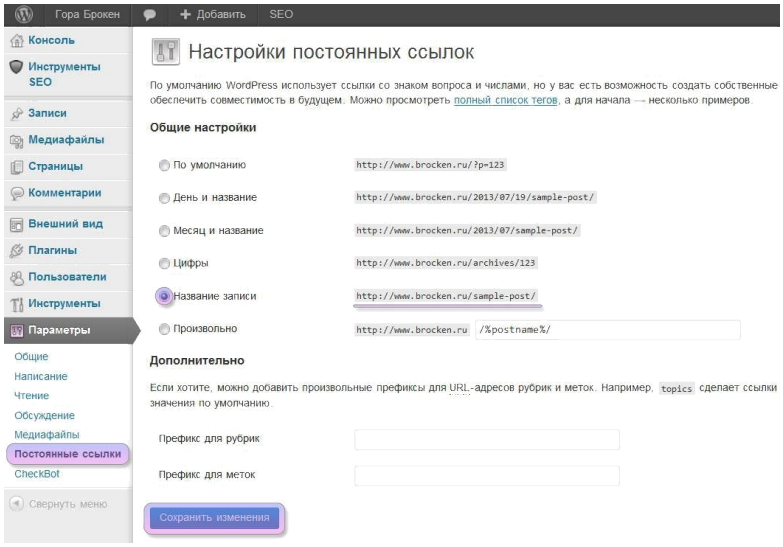
На страницах сайта, на примере, оставлены для отображения 7 записей, для RSS – ленты 7 элементов + в rss отображать Анонс, а не полный текст.

Параметры Обсуждение - уберите галочки с "Пытаться оповестить блоги", "Упоминаемые в статье" и "Разрешить оповещения с других блогов".

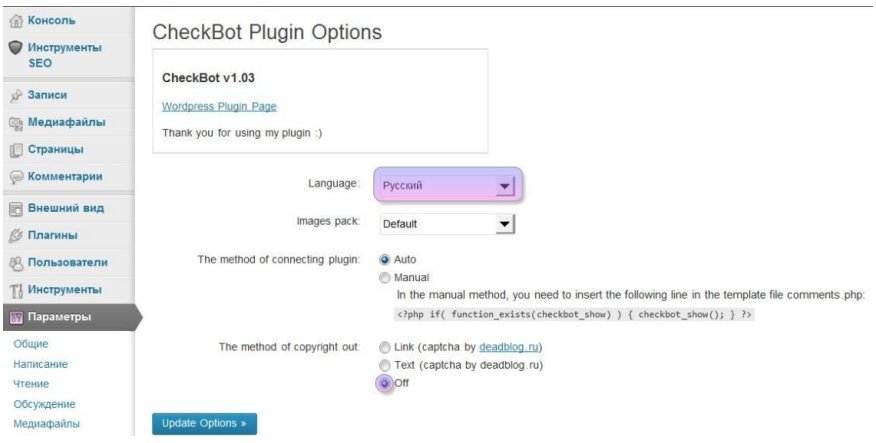


Уберите галочки в разделе "Отправить мне письмо", когда пользователь оставил комментарий и комментарий ожидает проверки. Прочитать комментарии можно зайдя в панель администратора WordPress. Нажмите кнопку Сохранить изменения.

Постоянные ссылки. По умолчанию ссылки выглядят как - ?p=3545. Чтобы они выглядели как заголовок на английском языке, нажмите Название записи и Сохранить изменения.



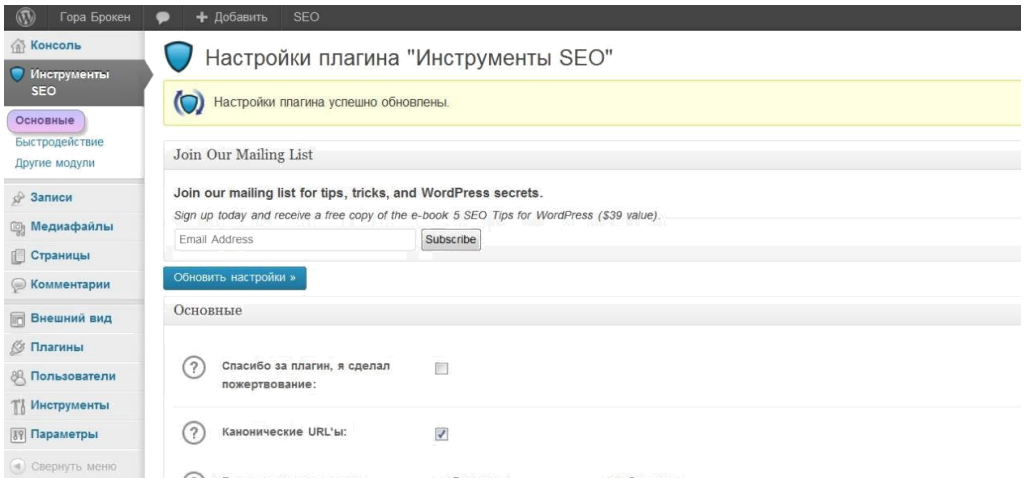
Еще одна настройка - антиспам плагина **CheckBot**. Выберите Русский и убирите ссылку автора плагина (или оставляем ее в знак благодарности). Нажите на кнопку Update Options.



7. Настройка инструментов SEO

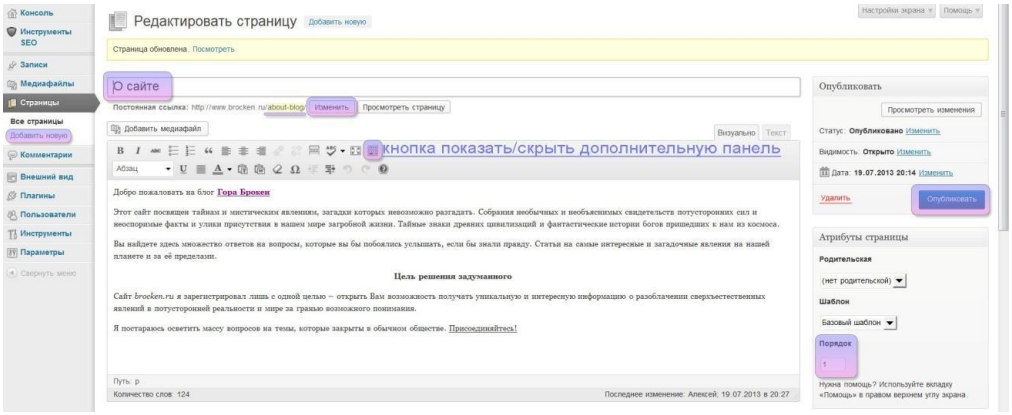
Настройка в плагине **All in One SEO Pack**. В настройках главной страницы необходимо задать Заголовок, описание и ключевые слова. Это делается для поисковиков и SEO2, они должны быть близки тематике сайта и отражать его направление.

SEO расшифровывается как Search Engine Optimization, что в переводе означает поисковая оптимизация или же оптимизация под поисковые машины. Смысл этих трех слов – это оптимизация сайта для дальнейшего продвижения сайта в рейтинге поисковых систем.



8. Создадим несколько страницы

Страницы - это статичные страницы, например, *О сайте*, *Об авторе* или *Контакты*. Находятся вверху сайта, в горизонтальном меню. Сделаем три такие страницы.

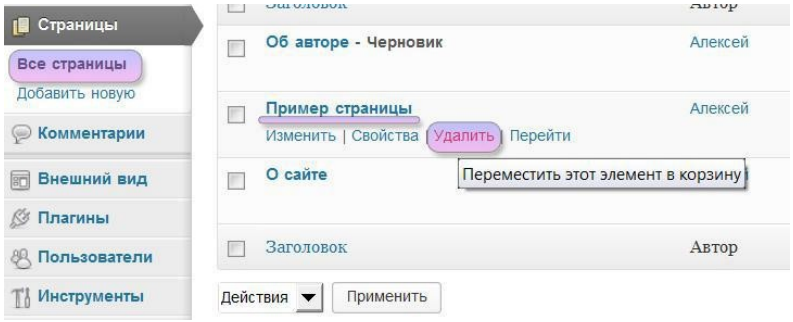


Нажмите Добавить новую в разделе Страницы. впишите заголовок этой страницы – “О сайте”.

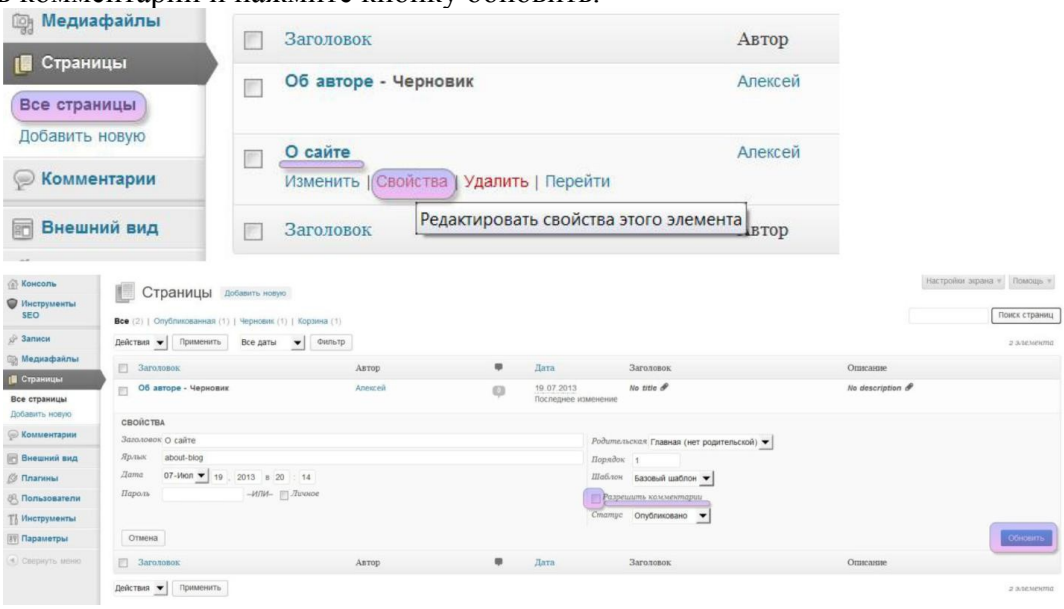
В визуальном редакторе откройте дополнительную панель, щелкнув по кнопке показать/скрыть дополнительную панель. В впишите текст в вашем сайте.

Обязательно измените название Постоянной ссылки, чтобы она была написана латиницей (на английском), иначе могут быть проблемы в поисковиках и в некоторых местах при формировании ссылки. Всегда изменяйте Постоянную ссылку. Указываем порядок – “1”. Теперь в горизонтальном меню она будет первая. И жмем кнопку Опубликовать.

На сайте есть первая страница. Теперь можно удалить стандартно созданную страницу – “Пример страницы”. Для этого щелкните Все страницы в разделе Страницы и подведите мышку к Пример страницы. Снизу появится меню << Изменить | Свойства | Удалить | Перейти >> выберите удалить.

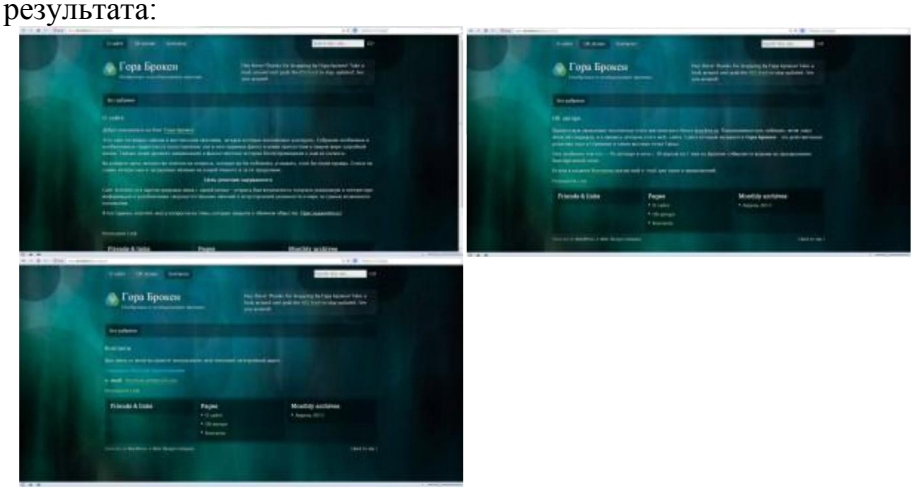


WordPress к каждой странице или записи создает по умолчанию форму для комментирования. Необходимо форму убрать, так как для этой страницы она не нужна. Для этого щелкните Все страницы в разделе Страницы и подведите мышку к странице О сайте, в появившемся меню нажмите Свойства. В свойствах уберите галочку напротив Разрешить комментарии и нажмите кнопку обновить.



После создания страницы О сайте, создайте страницы Об авторе и Контакты, при этом не забывая указывать их порядок (Об авторе(2-вторая), Контакты(3-третья)). В странице Об авторе напишите пару строк о себе. В странице Контакты впишите ваш адрес электронной почты и о том как связаться с вами.

Пример результата:



9. Настройка файла Robots.txt

Перед тем как начать заполнять сайт контентом и делать записи в него необходимо оградить его от такого понятия как дублирование контента - как только на сайте будет некоторое количество статей, CMS создаст массу ссылок на статью и она проиндексируется в поисковой выдаче, может появится множество лишних ссылок на

сайт вместо одной нужной, той которая ведет на искомую статью. Это может привести к плохим отношениям с поисковыми системами.

Для того, чтобы не было повторов ссылок, необходимо специальный файл который называется Robots.txt. Он регулирует или помогает поисковым системам правильно просканировать web-сайт и добавить те материалы которые действительно нужны, то есть основные статьи. Пример содержимого файла, можно посмотреть по ссылке — http://www.white-windows.ru/robots.txt.

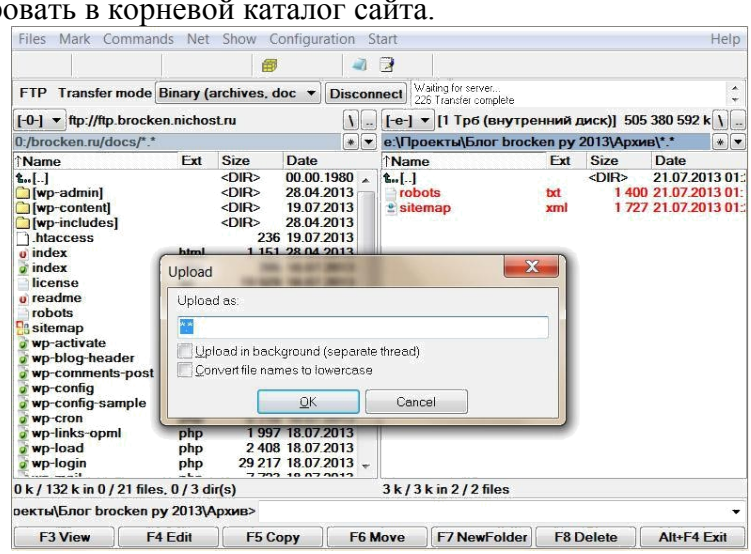
Создайте обычный **txt** файл с именем **robots.txt** маленькими буквами. Копируете в него найденный и правильный на ваш взгляд шаблон. Перепишите в нем следующее:

28 строка Host: — напишите адрес своего сайта.

69 последняя строка **Sitemap** – сюда нужно прописать ссылку на файл **sitemap.xml** (этот файл нужен для помощи поисковикам в сканировании сайта, он создается

отдельными сервисами, например этим можно сделать — http://htmlweb.ru/analiz/sitemap.php потом этот файл также скопировать в корень своего сайта вместе с **robots.txt**)

Эти файлы **robots.txt** и **sitemap.xml** (на первом этапе, когда мало статей, он не так важен) нужно скопировать в корневой каталог сайта.



10. Контент

Контент – это собственно тот интересный и уникальный материал, который публикуется на сайте (статьи, записи, обзоры, новости, инструкции). Но как же сделать новый и интересный контент.

Писать самому – но не все мы писатели.

Сделать рерайт – переделать чужой материал и улучшить его.

Скопировать – скопировать и вставить чужую статью. Это самый плохой вариант, так как поисковики могут заблокировать сайт.

Купить готовые статьи на бирже статей.

Могу посоветовать некоторые биржи статей: http://advego.ru/, www.textsale.ru, www.texchange.ru.

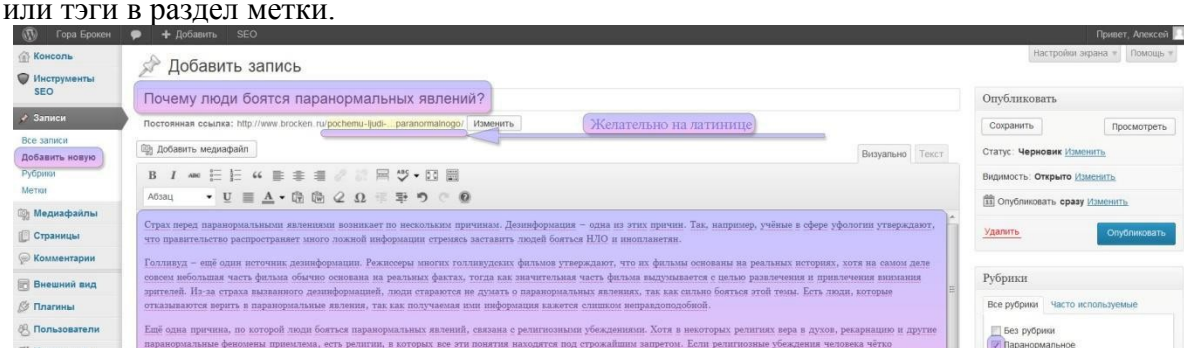
Создадим рубрику. Для добавления рубрики зайдите в **Записи -> Рубрики**. Придумайте название рубрики, название ярлыка – на английском языке и краткое описание.

Рубрика создается для наполнения статей.

11. Записи

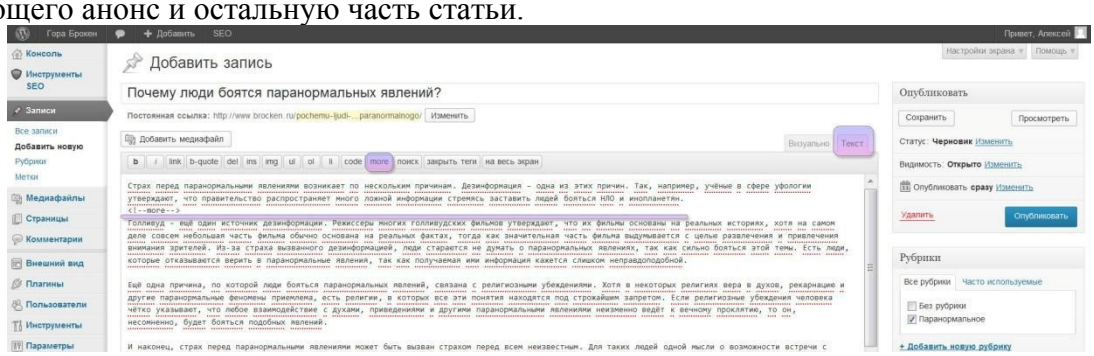
Для того чтобы добавить новую запись(статью или обзор) зайдите в раздел Записи -> Добавить новую.

Введите заголовок статьи. Далее изменить постоянную ссылку на ссылку в виде латиницы(на английском). Далее вставляем основной текст в визуальный редактор. Справа выбираем, в какую рубрику поместить статью. Можно также добавить ключевые слова или тэги в раздел метки.



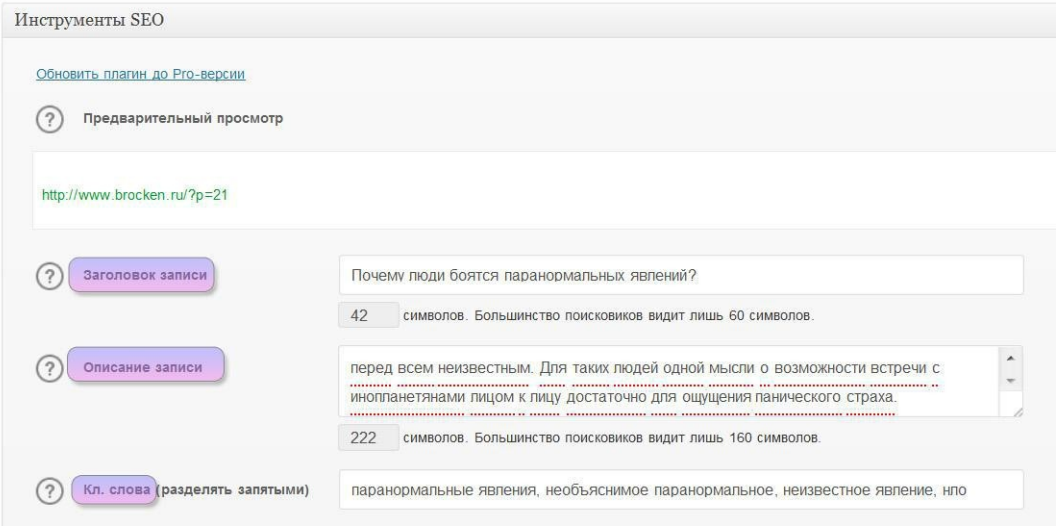
Сделайте так чтобы статья не выводилась на основной странице блога полностью. То есть выводился заголовок и анонс статьи пару начальных строк, а дальше посетитель должен будет открыть статью полностью и тем самым посмотреть полный текст.

Для этого перейдите из визуального редактора в текстовый. И после первого абзаца, поставьте в конце курсор и нажамите на кнопку **more** для вставки тега разделяющего анонс и остальную часть статьи.

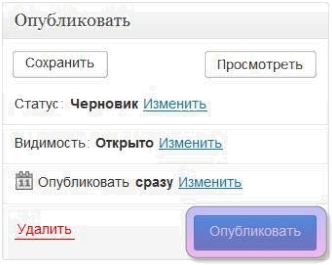


Старайтесь разделять его на абзацы, разделы, подзаголовки, выделять важные места подчеркиванием или жирным шрифтом, выделять что-то другим цветом, формировать списки, цитаты, делать текст интересным и красивым.

Инструменты SEO. В целом это нужно только для поисковиков. Заголовок можно скопировать из заголовка статьи. А в описание скопировать часть текста из статьи в виде пару интересных предложений.



Нажите кнопку — **Опубликовать**

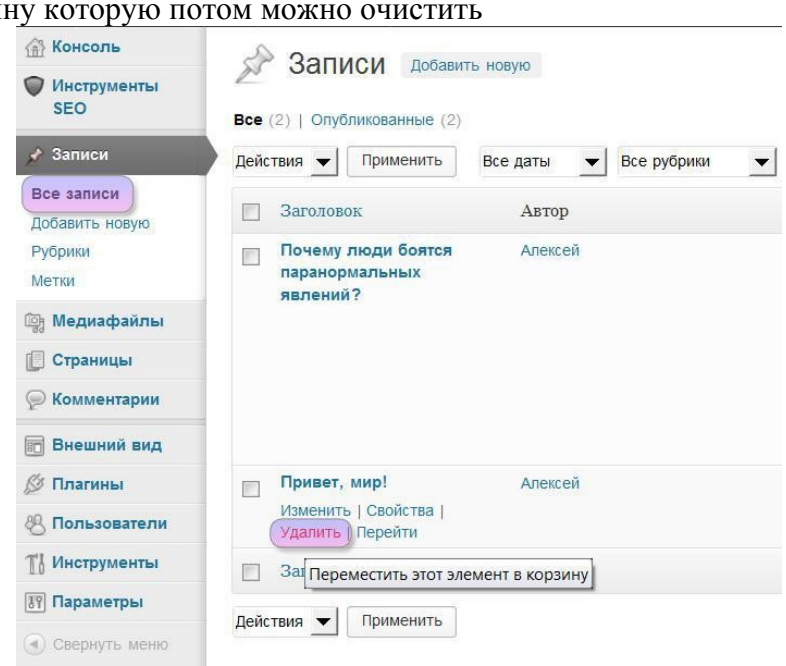


.

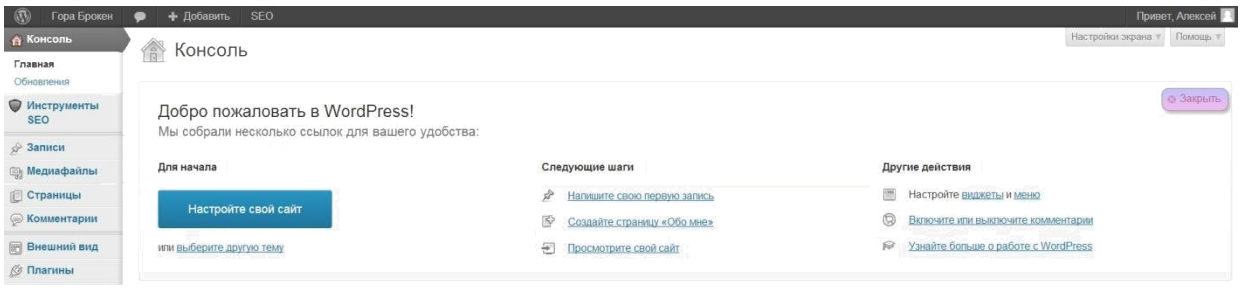
Посмотрите ваш сайт через браузер.

Можно удалить созданную по умолчанию запись **Привет, мир!** Для этого зайдите

2. **Записи -> Все записи** и подведите курсор мышки к записи Привет, мир! Снизу появится меню в котором нужно выбрать **Удалить**. Запись будет удалена и перемещена в корзину которую потом можно очистить



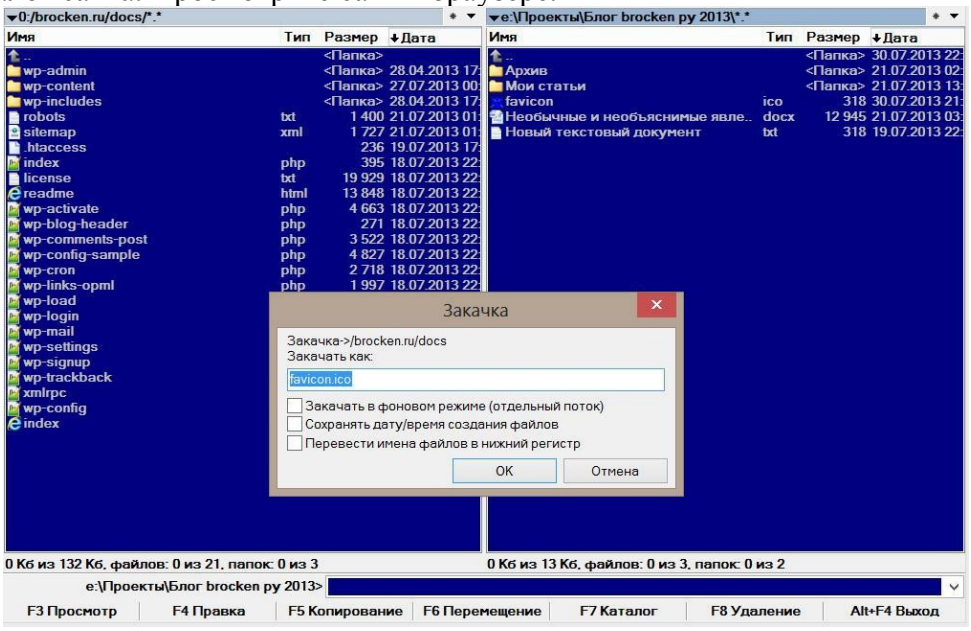
После данных манипуляций уберите окно начальных настроек. Для этого нужно нажать в правом верхнем углу на надпись **Закрыть**.



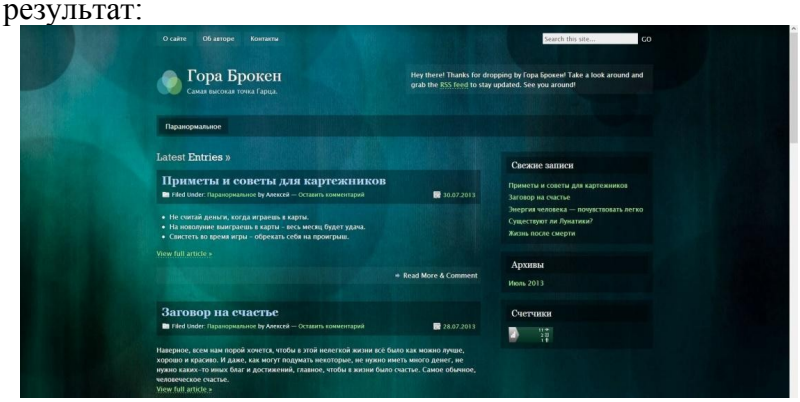
3. Favicon – мини иконка для сайта

Фавиконка – это маленькая иконка размером 16*16 пикселей, она отображается в поиске рядом с ссылкой сайта, показывается в браузере сверху в заголовке текущего окна. Для её создания можно использовать следующие web-сервисы:favicon.ru или www.favicon.cc.

Сохраните иконку на компьютер с названием favicon.ico, этот файл скопируйте корневой каталог сайта. Просмотрите сайт в браузере.



Примерный результат:



Шаг 9. Работа с удаленным сервером

Настроив сайт на локальном сервере, вы можете разместить его на удаленном сервере. Для обучения, можно использовать сайт http://www.biz.nf/, который предоставляем 250 Мб дискового пространства и все необходимы ресурсы для работы сайта (Apache, PHP, MySQL).

Попробуйте разместить ваш сайт самостоятельно, пользуясь справкой портала.

Шаг 10. Отчетность

Предоставьте отчет по практической работе в произвольной форме преподавателю.

Дополнительные материалы для изучения

https://ru.wordpress.com/ - русскоязычный сайт для создания сетевых ресурсов на WordPress.

https://ru.wordpress.org/ - справка на русском языке по WordPress.

Практическая часть. Вариант 2

Подготовка

1 Скачиваем Денвер с сайта http://www.denwer.ru/

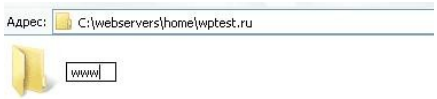
2 Устанавливаем Денвер (подробней про установку Вы можете прочитать тут - http://www.denwer.ru/base.html).

3 Далее скачиваем последнюю версию WordPress с сайта - http://ru.wordpress.org/

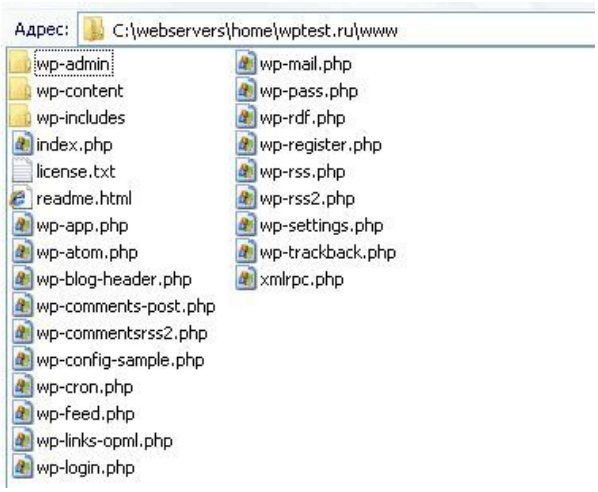
I. Установка необходимого ПО

Запускаем Денвер для этого открываем папку C:\WebServers\etc, и запускаем Run.exe.
Далее в папке C:\WebServers\home\ создаем папку с названием Вашего сайта - например wptest.ru

переходим в эту папку и создаем в ней папку www. В итоге получаем такой путь: C:\WebServers\home\wptest.ru\www

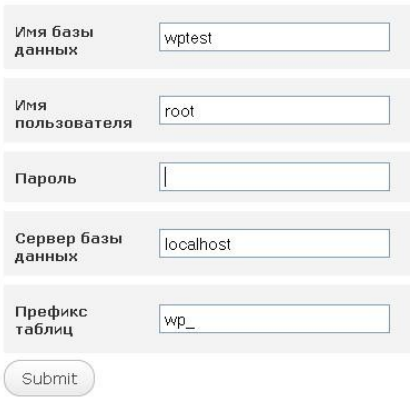


Распакуйте архив с последней версией WordPress в папку C:\WebServers\home\ wptest.ru\www



Создаем базу данных, для этого в адресной строке Вашего браузера набираем http://localhost/Tools/phpMyAdmin/ вводим логин и пароль (по умолчанию имя пользователя - root, пароля нет). В открывшимся окне в поле "Создать новую БД" пишем название базы данных (например wptest), в соседнем поле выбираем сравнение с utf8_general_ci и нажимаем кнопку "создать".

2. Далее запускаем и устанавливаем WordPress - для этого в адресной строке Вашего браузера набираем http://wptest.ru - в появившимся окне нажимаем кнопку "создать файл настроек", далее кнопку "Вперед!" в появившимся окне заполняем все поля.



/ поле "Имя базы данных" вводим название базы данных, в нашем случаи wptest.

/ поле "Имя пользователя" вводим имя пользователя базы данных, в нашем случаи root.

/ поле "Пароль" вводим пароль баззы данных, в нашем случаи оставляем его ПУСТЫМ.

/ поле "Сервер базы данных" оставляем localhost.

/ поле "Префикс таблиц" оставляем wp_ далее нажимаем кнопку Sumbit, затем в появившимся окне нажимаем кнопку "Запустить установку". Далее вводим заголовок Вашего сайта и Ваш емэйл (на него Вам придет пароль для доступа к админке, а также будет приходить информация о новых комментариях на сайте) и нажимаем кнопку "Установить WordPress". На этом установка WordPress завершена. Переписываем или копируем логин и пароль и входим на Ваш новый сайт - http://wptest.ru для перехода в админку http://wptest.ru/wp-admin/

На этом установка завершена.

/ Ознакомление с интерфейсом WordPress

Ознакомьтесь с интерфейсом системы управления контентом WordPress (WP). Посмотрите какие элементы есть в **Консоли** WP, для чего они служат. Также ознакомьтесь с **Настройками** WP.

В настройках найдите опцию **Видимость для поисковых систем** и поставьте «галочку» **Попросить поисковые системы не индексировать сайт** (Данная опция не даёт гарантий, что ПС не будут индексировать ваш ресурс, но на этапе создания сайта её можно включить).

Поставьте в настройках программы необходимый часовой пояс (UTC +9) и выберите подходящий для вас формат даты.

2. Установка темы для вашего сайта (блога)

пункте меню **Внешний вид** выберите **Темы** и загрузите предложенный преподавателем шаблон (тему). Активируйте тему и ознакомьтесь с её настройками (для разных тем могут быть разные настройки). В настройках темы заполните **Свойства сайта** (**Название сайта** и **Краткое описание**). Добавлением виджетов и меню мы займёмся позже (Прежде чем добавить меню, необходимо создать **Страницы** для вашего сайта).

3. интернете можно найти множество бесплатных и платных **тем** для WP, так данная система управления контентом является очень популярной, так же вы можете создавать свои темы. Один из качественных ресурсов, где вы можете скачать темы для WP: wp-templates.ru

IV. Добавление страниц и создание меню

Следующий шаг в создании вашего сайта – добавление **Страниц** и в дальнейшем создание **Меню**. Вы должны определиться какие страницы у вас будут основными, т.е. будут главными элементами меню сайта. Для создания страниц вам нужно определиться с предметной областью (тематикой) вашего сайта. В практической работе лучше выбрать тему личного сайта или сайта портфолио, также можете сделать сайт вашей группы.

Создайте 3-4 страницы для ваше сайта, причём одну страницу сделайте дочерней по отношению к другой (для того чтобы был элемент выпадающего меню). Например, страница Портфолио – для неё дочерняя страница Мои достижения или Обо мне.

В зависимости от выбранной тематики сайта наполните страницы информацией. Добавьте тест и картинки. Для добавления картинок их сначала нужно загрузить в **Медиафайлы.** Для каждой картинки добавляйте **Описание** и **Атрибут alt** (это является частью поисковой оптимизации вашего ресурса). При добавление информации на страницы обратите внимание что во встроенном редакторе есть 2 режима: **Визуально** и **Текст**. Режим **Текст** показывает теги и даёт возможность вручную их добавлять и менять.

После добавления страниц создайте меню для вашего сайта. Для этого в **Консоли** выберите **Внешний вид** – **Меню**. В Структуре меню вы можете вручную перетаскивать элементы меню в нужное место и создавать дочерние элементы (многоуровневое меню). После создания структуры меню в **Настройках меню** поставьте «галочку» **Главное меню**.

Когда вы создадите меню обязательно проверьте добавилось ли оно и как работает.

Содержание отчета

1. Цель
2. Ход работы
3. Выводы

Контрольные вопросы

1. Что такое CMS, как они работают?
2. Что такое Denwer и для чего используется?
3. Как установить WordPress на локальный сервер Denwer?
4. Какие вы знаете основные элементы Консоли WP?
5. Чем CMS отличаются от конструкторов сайтов?
6.

Лабораторная работа №39

Администрирование сайта

Цель: получить практические навыки администрирования сайта с помощью CMS.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Администрирование сайта – обслуживание сайта, включающее себя управление категориями, меню и контентом

Администрирование сайта – ряд мероприятий по запуску и поддержке четкого функционирования сайта. Владельцы всех веб-ресурсов знают, что просто написать сайт, опубликовать его, заполнив минимальным контентом совершенно недостаточно. И не важно, будет это интернет-магазин, сайт-визитка, сайт большой корпорации или крупный портал, он требует ухода, присмотра и обновления. Именно постоянное администрирование позволяет сайту удерживать свои позиции, занятые им при раскрутке.

Как правило, информационная поддержка включает в себя добавление новостей, объявлений, новых актуальных и уникальных статей. Если это магазин, то это будут новости про расширение ассортимента товаров или услуг, изменение цен, размещение информации о скидках. Чем больше всевозможного текстового и графического контента будет предоставлено пользователям, тем меньше конкурентов у сайта останется, и тем больше посетителей зайдет на ресурс.

Администрирование делится на информационное и техническое.

Администрирование сайта информационное

Заключается в работе по заполнению ресурса. Это постоянная, монотонная работа, которая делится на несколько составляющих:

- Добавление новостей и новых материалов на ресурс.
- Изменение уже имеющихся материалов, исправление ошибок, повышение удобства использования контента.
- Удаление и помещение в архив устаревшей информации.
- Поддержка актуальности и практической значимости материалов сайта.

Администрирование сайта техническое (техподдержка)

Техническое администрирование призвано обеспечить постоянный доступ к сайту любому пользователю. Техподдержка заключается в следующих мероприятиях:

- Выборе хостинга и его настройке.
- Круглосуточный контроль над состоянием сервера.
- Круглосуточная поддержка сайта и всех его элементов в рабочем состоянии.

Каждый владелец сайта может администрировать его самостоятельно, либо передать эту работу профессионалам. В последнем случае, функция держателя будет заключаться лишь в предоставлении и поддержке новых материалов и тем для опубликования, а также указаниям по желаемым изменениям. Такие функции могут осуществляться при непосредственном контакте с техподдержкой или дистанционно через Интернет

Администрирование сайтов работающих на CMS WordPress в большинстве случаев не требует использования FTP и MySQL каналов доступа к сайту, и осуществляется непосредственно из окна браузера.

Панель администрирования WordPress.

1. Навигационное меню *админки WordPress* находится слева. Впервые зайдя туда, мы попадем в пункт «Консоль», в котором можно указать описание сайта, установить часовой пояс, разрешить или отключить комментарии и произвести другие необходимые настройки ВордПресс.
2. Следующим пунктом у нас идут обновления. Если появится новая версия WordPress, установленных тем или используемых плагинов, то в этом меню их можно обновить одним щелчком мыши. Все произойдет в автоматическом режиме.
3. Следующий пункт «Записи» наверно один из самых важных для нас. Именно там мы будем писать свои посты, создавать рубрики.

4. Медиафайлы — следующий пункт. Здесь хранятся все наши изображения, аудио и видеозаписи, которые мы вставляли в блог WordPress. Вы всегда сможете удалить лишнее, неиспользуемое, или что-то подредактировать.
5. Пункт «Ссылки» поначалу не очень важен, на мой взгляд, но позднее очень может пригодится. Суть в том, что в сайдбаре вашего блога вы сможете разместить блок с ссылками, например для обмена ими с какими-то другими сайтами в целях наращивания ссылочной массы.
6. Пункт «Страницы» позволяет создавать нам статичные страницы, такие, например, как страница контактов, или карта сайта, либо же какие-то страницы с данными о себе или свои услугах. Комментарии — здесь вы можете администрировать все комментарии на вашем блоге WordPress.
7. Внешний вид тоже один из важнейших пунктов нашего блога. С его помощью можно полностью преобразить внешний вид WordPress и настроить его по своему вкусу
8. Плагины — ну тут в общем интуитивно все понятно. Здесь отображается список всех установленных необходимых плагинов wordpress, мы можем включать и выключать ненужные, а так же устанавливать новые или удалять неактуальные плагины
9. В меню «Пользователи» можно администрировать всех пользователей. Добавлять, удалять, менять профили и так далее.
10. Инструменты.
11. И последний пункт меню админки WordPress — «Параметры». Тоже очень важный пункт, в котором производится много различных настроек — от внешнего вида до работы плагинов

ПОШАГОВАЯ ИНСТРУКЦИЯ ПО ОСНОВНЫМ МОМЕНТАМ УПРАВЛЕНИЯ CMS WORDPRESS 3.3 ДЛЯ НАЧИНАЮЩИХ АДМИНИСТРАТОРОВ

Для того что бы попасть в административную зону сайта созданного на основе CMS WordPress необходимо к адресу сайта в строке браузера дописать “wp-admin” соответственно, если название вашего сайта имеет вид “http://название-сайта.ru”, то адрес административной зоны будет располагаться по адресу “http://название-сайта.ru/ wp-admin /”. На открывшейся странице вводим логин и пароль администратора, указанные при установке CMS.

СМЕНА ПАРОЛЯ АДМИНИСТРАТОРА В CMS WORDPRESS 3.3

В административной зоне сайта идем в “Пользователи” “Все пользователи” В таблице с пользователями наводим на строчку с Администратором и жмем “Изменить” Спускаемся вниз, вводим новый пароль в поле “Новый пароль” и дублируем в следующим за ним поле Внизу страницы жмем “Обновить профиль”

СМЕНА E-MAIL АДРЕСА АДМИНИСТРАТОРА В CMS WORDPRESS 3.3

В административной зоне сайта идем в “Пользователи” “Все пользователи” В таблице с пользователями наводим на строчку с Администратором и жмем “Изменить” В группе полей “Контакты” меняем значение поля E-mail на новое значение Внизу страницы жмем “Обновить профиль”

ДОБАВЛЕНИЕ НОВОГО ПОЛЬЗОВАТЕЛЯ В CMS WORDPRESS 3.3

В административной зоне сайта идем в “Пользователи” “Добавить нового” Вводим индивидуальные данные нового пользователя Внизу страницы жмем “Добавить нового пользователя”

ДОБАВЛЕНИЕ НОВОЙ СТАТЬИ В CMS WORDPRESS 3.3

В CMS WordPress иерархия статей ведется путем добавления Рубрик, поэтому если вы хотите иметь на сайте несколько тематических разделов, то прежде чем добавить статью, для нее следует добавить Рубрику, для этого:

1. В административной зоне сайта идем в “Записи” “Рубрики” В левой половине странице под надписью “Добавить новую рубрику” в поле “Название” вводим имя для нашей Рубрики Внизу страницы жмем “Добавить новую рубрику”
2. В административной зоне сайта идем в “Записи” “Добавить новую” Вводим название и текст статьи, также вы можете прикрепить к статье любой медиа объект, поставив курсор в любое место статьи и нажав “Загрузить / Вставить” (в открывшемся окне заполняем все необходимые поля и жмем “Вставить в запись” Далее в правом блоке “Рубрики” отмечаем к какой именно Рубрике (Рубрикам) будет относиться данная статья Далее в правом верхнем углу мы можем либо нажать “Сохранить” и тогда статья сохраниться в черновиках, либо “Опубликовать” и тогда статья автоматически сохраниться и попадет на страницы сайта в соответствующий раздел-Рубрику.

РЕДАКТИРОВАНИЕ СУЩЕСТВУЮЩЕЙ СТАТЬИ В CMS WORDPRESS 3.3

В административной зоне сайта идем в “Записи” “Все записи” В таблице со статьями сайта, наводим курсор на ту, которую намерены отредактировать и жмем “Изменить” Вносим необходимые правки В правом верхнем углу жмем “Обновить”

ДОБАВЛЕНИЕ НОВОГО МЕНЮ В CMS WORDPRESS 3.3

В административной зоне сайта идем в “Внешний вид” “Меню” В правом верхнем углу вводим имя для нового меню в поле “Заголовок меню” и жмем “Создать меню” Далее в блоке “Области темы” выбираем из списка добавленное нами меню и жмем “Сохранить”

ДОБАВЛЕНИЕ НОВОГО МЕНЮ В CMS WORDPRESS 3.3

1. В административной зоне сайта идем в “Внешний вид” “Меню” В правом верхнем углу вводим имя для нового меню в поле “Заголовок меню” и жмем “Создать меню”
2. В административной зоне сайта идем в “Внешний вид” “Виджеты” В левой части находим виджет “Произвольное меню” и перетаскиваем его в один из блоков сайта (расположены справа) После размещения данного виджета в одном из блоков вам предложат его назвать и выбрать для него меню, вводим необходимые данные и жмем “Сохранить”

ДОБАВЛЕНИЕ НОВОГО ПУНКТА МЕНЮ (СТРАНИЦЫ) В CMS WORDPRESS 3.3

В административной зоне сайта идем в “Внешний вид” “Меню” В правом верхнем блоке, выбираем из вкладок меню, в которое будем добавлять новый пункт Далее нам становятся доступны блоки расположенные в правой половине странице:

Блок “Произвольные ссылки” позволяет добавлять любые ссылки, хоть внешние, хоть внутренние с указанием заголовка для пункта меню.

Блок “Страницы” позволяет добавлять пункты, указывающие на конкретные страницы расположенные на сайте.

Блок “Рубрики” создает ссылки ведущие в разделы сайта, на таких страницах все статьи из данной рубрики выводятся в виде блога.

После того как вы определились с форматом вашего пункта или пунктов, в правом блоке вы можете настроить порядок их отображения простым перетаскиванием В правом верхнем углу жмем “Сохранить меню”

ДОБАВЛЕНИЕ СТРАНИЦЫ “КОНТАКТЫ / ОБРАТНАЯ СВЯЗЬ” В CMS WORDPRESS 3.3

Добавление страницы “Контакты” в CMS WordPress осуществляется аналогично с добавлением простой статьи, в которой вы можете указать свои контактные данные. Форма отправки писем с сайта устанавливается как стороннее расширение и не входит в стандартный набор программных модулей.

НАСТРОЙКА ЧПУ ССЫЛОК В CMS WORDPRESS 3.3

Для того чтобы ссылки на страницы вашего сайта имели максимально понятный вид как для человека так и для поискового бота, нужно произвести небольшие настройки в ВордПрессе, а именно:

В административной зоне сайта идем в “Параметры” “Постоянные ссылки” Отмечаемся напротив “Название записи” Внизу страницы “Сохранить изменения”

ИЗМЕНЕНИЕ НАСТРОЕК САЙТА В CMS WORDPRESS 3.3

Для того, что бы изменить специфические настройки сайта, такие как название сайта, слоган, ключевые слова и прочее в административной зоне сайта идем в “Параметры” “Общие” Вносим необходимые изменения, особое внимание стоит уделить заголовку и краткому описанию (для поисковых систем) Внизу страницы “Сохранить изменения”

Задание: Выполнить администрирование сайта, разработанного на предыдущей лабораторной работе, в соответствии с инструкцией

Содержание отчета

1. Цель
2. Ход работы
3. Выводы

Контрольные вопросы

1. В чем заключается администрирование сайта
2. Функциональные рвзделы администрирования
3.

Лабораторная работа №40

Публикация сайта на бесплатном хостинге

Цель: получить практические навыки публикации сайта на бесплатном хостинге.

Ход работы:

1. Изучить теоретический материал.
2. Выполнить задания в соответствии с указаниями.
3. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
4. Подготовить ответы на контрольные вопросы (устно)

Теоретический материал:

Хостинг сайтов – это онлайн услуга, которая позволяет публиковать ваш веб-сайт или веб-приложение в интернете. Когда вы подписываетесь на услугу хостинга, вы обычно арендуете пространство на сервере, на котором вы можете хранить все файлы и данные, необходимые для правильного функционирования вашего сайта.

Сервер – это физический компьютер, который работает без перерывов, чтобы ваш сайт был доступен всё время для тех, кто хочет его посетить. Ваш хостинг отвечает за поддержание работы сервера, защиту его от вредоносных атак и передачу вашего контента (текста, изображений, файлов) с сервера в браузеры ваших посетителей.

Различные типы хостинга

Большинство провайдеров предлагают несколько типов хостинга для удовлетворения различных потребностей клиентов. Вот наиболее часто предоставляемые типы хостинга:

- Общий хостинг (Shared Hosting)
- VPS (**V**irtual **P**rivate **S**erver – виртуальный приватный сервер) хостинг
- Облачный хостинг (Cloud Hosting)
- WordPress хостинг
- Хостинг выделенных серверов

Чем больше ваш сайт, тем больше требуется пространство на сервере. Лучше всего начинать с малого – с тарифного плана общего хостинга, и когда ваш сайт станет больше, перейти на более расширенный тарифный план или сменить тип хостинга.

Хостинг провайдеры обычно предлагают более одного тарифного плана хостинга для каждого типа хостинга.

Hostinger – отличный провайдер с низкими ценами и хорошей технической составляющей. Хостинг простой в обращении, имеет множество элементов для облегчения работы с ним, из-за чего даже новички могут быстро начать работу. Быстрая и грамотная техническая поддержка. У провайдера есть различные акции и бонусы, которые заинтересуют многих.

Для загрузки веб-сайта вы можете выполнить следующие 6 простых шагов:

Прежде чем начать следовать советам из этого руководства, вам понадобится следующее:

- Доступ к **панели управления** вашей учётной записи хостинга.
- **Файлы вашего сайта** (желательно в архиве .zip или .tar.gz) и **базы данных** (если используются).
- FTP-клиент, такой как FileZilla и данные для входа в FTP (необязательно).

Шаг 1: выберите надёжный веб-хост

Прежде всего, вам нужно найти правильный веб-хост. Создание веб-сайта — это не то, что вам нужно сделать легкомысленно. Таким образом, вы должны выбрать первоклассный хост, у которого есть все ключевые функции, чтобы запустить ваш веб-проект.

Вот несколько важных вещей, которые вы должны искать в веб-хостинге:

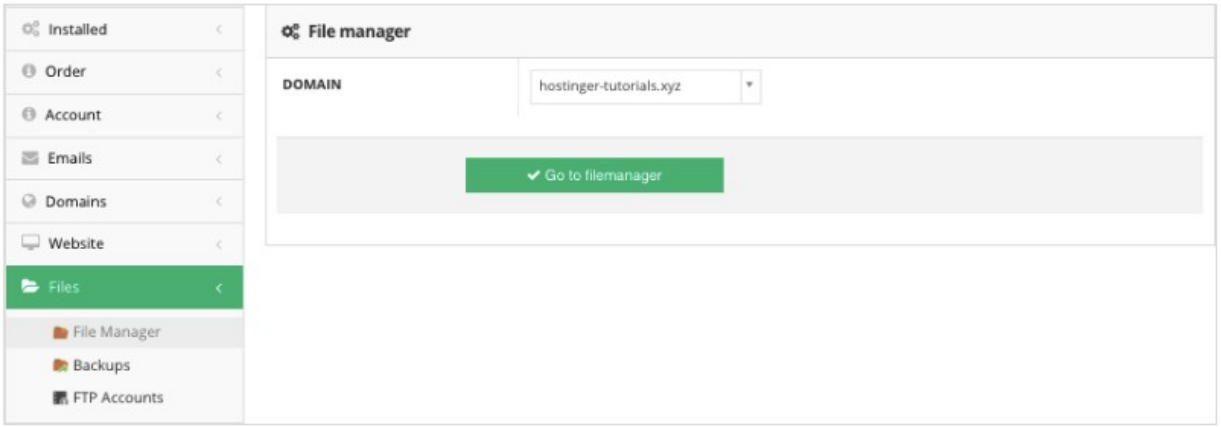
1. **Онлайн поддержка.** Нет ничего хуже, чем застрять и узнать, что вам некому помочь. Если веб-хост не предлагает чат или телефонную горячую линию, вы можете оказываться в трудном положении время от времени.
2. **Контролируйте свой веб-хостинг.** Чем меньше вы контролируете свой аккаунт, тем больше вероятность возникновения трудностей, когда ваш сайт начнёт расти. Хорошим примером может быть сравнение WordPress.com с WordPress.org (англ) (версия для самостоятельного размещения).
3. **Место для роста.** Самые успешные веб-сайты играют в долговременную игру. Прежде чем вы присоединитесь к веб-хосту, убедитесь, что у них есть масштабируемые решения, если ваш веб-сайт начинает требовать больше огневой мощи (например, виртуальные частные серверы или облачный хостинг).
4. **Гарантия возврата денег.** Никто не любит плохие инвестиции, поэтому не забудьте проверить политику возврата (англ). Это даст вам время, чтобы проверить всё, прежде чем полностью использовать эту услугу.

Шаг 2. Выберите способ, как загрузить свой сайт в интернет.

Следующая задача — выбрать правильный инструмент для выполнения задания. Вот четыре из наиболее часто используемых инструментов для загрузки сайта:

Файловые менеджеры

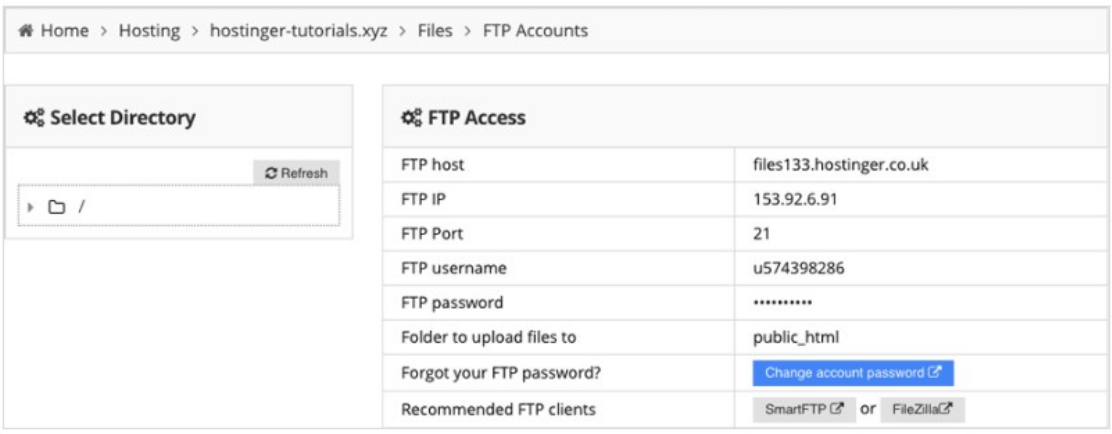
Браузерный инструмент со всеми ключевыми функциями, чтобы заботиться о ваших файлах и каталогах. Вы тоже получите его в Hostinger!



Однако одним из недостатков, с которым вы можете столкнуться, является ограничение на загрузку. Если резервная копия вашего сайта **превышает 256 МБ**, вы должны использовать FTP вместо этого.

Протокол передачи файлов (FTP)

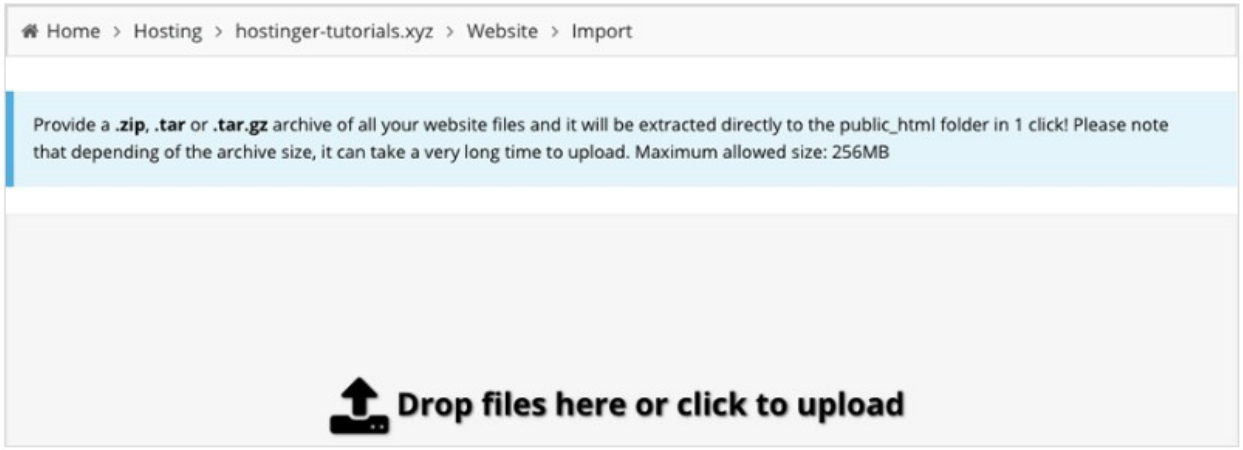
Поскольку каждый веб-хост включает FTP по умолчанию, вы можете использовать его для настройки FTP-клиента (например, FileZilla). Все необходимые данные будут размещены в **учётных записях FTP** в разделе **Файлы**.



Если вы решите загрузить свой сайт с помощью FTP, вы **не столкнётесь** с ограничениями по размеру. Это означает, что вы сможете импортировать резервный архив независимо от его размера.

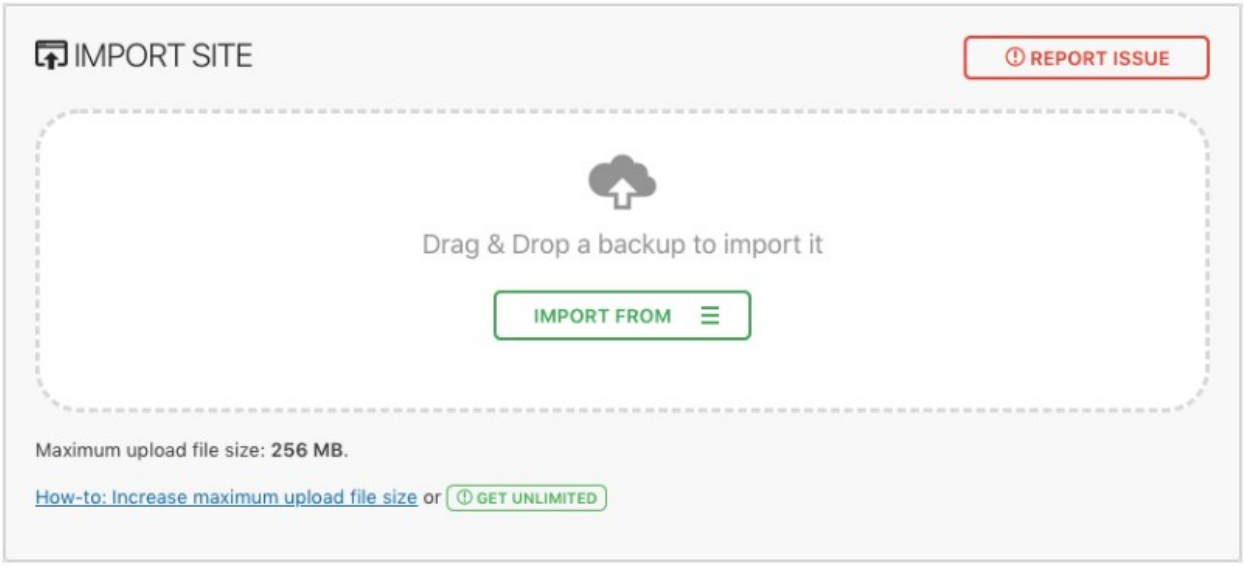
Автоматический импортёр сайтов

В Hostinger вы также можете найти функцию **Import Website**. Вы можете использовать его для извлечения архива сайта **до 256 МБ** непосредственно в каталог **public_html**.



Плагины миграции WordPress

Если вы используете WordPress, есть несколько способов перемещения вашего сайта. Один из самых простых способов — использовать плагин (например, All in One WP Migration (англ)), который позаботится обо всём.



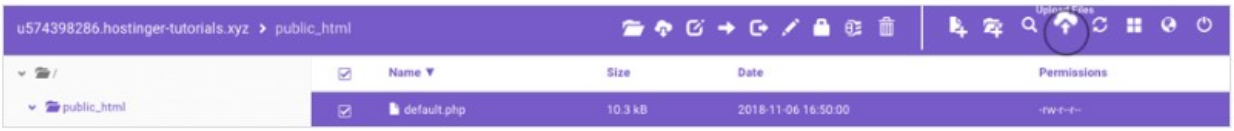
Однако он также имеет **ограничение 256 МБ**, которое можно увеличить, купив премиум-версию плагина.

Если веб-сайт больше, использование FTP, это лучший выбор. Подробные инструкции смотрите в нашем полном руководстве по миграции сайта WordPress.

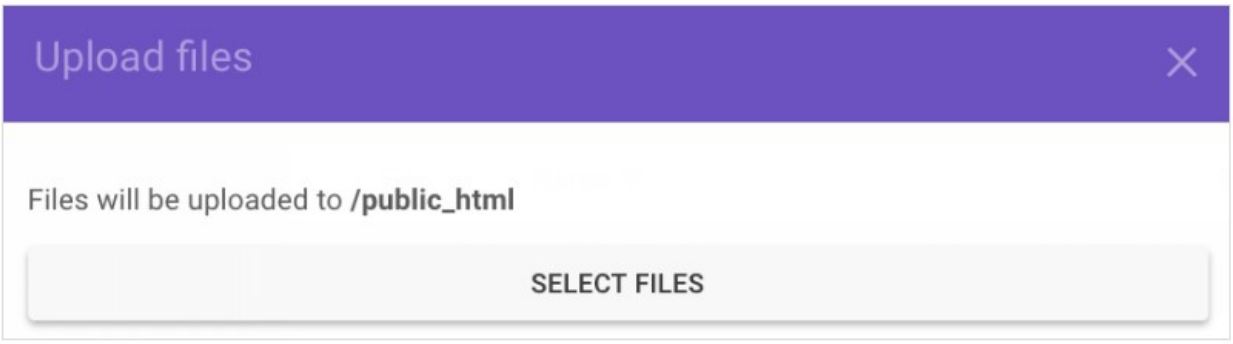
Шаг 3. Загрузите архив сайта и извлеките его.

Теперь, когда вы изучили лучшие инструменты и узнали, как загрузить свой сайт в интернет, пришло время засучить рукава и окунуться в процесс.

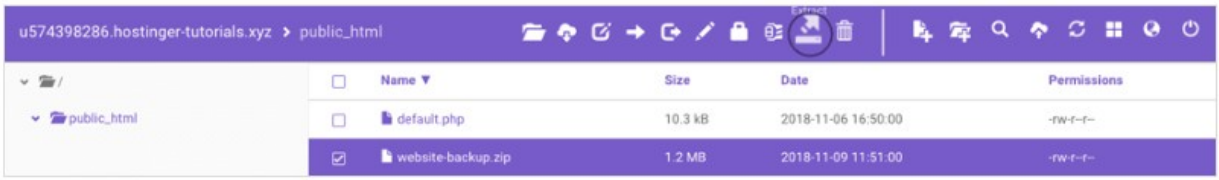
Начнём с **файлового менеджера Hostinger** — нашего пользовательского инструмента, предназначенного для облегчения рабочего процесса каждого веб-мастера. После его открытия выберите значок **Загрузить файлы** в правом верхнем меню.



Затем вам нужно выбрать архив веб-сайта через свой компьютер и импортировать его на наш сервер.

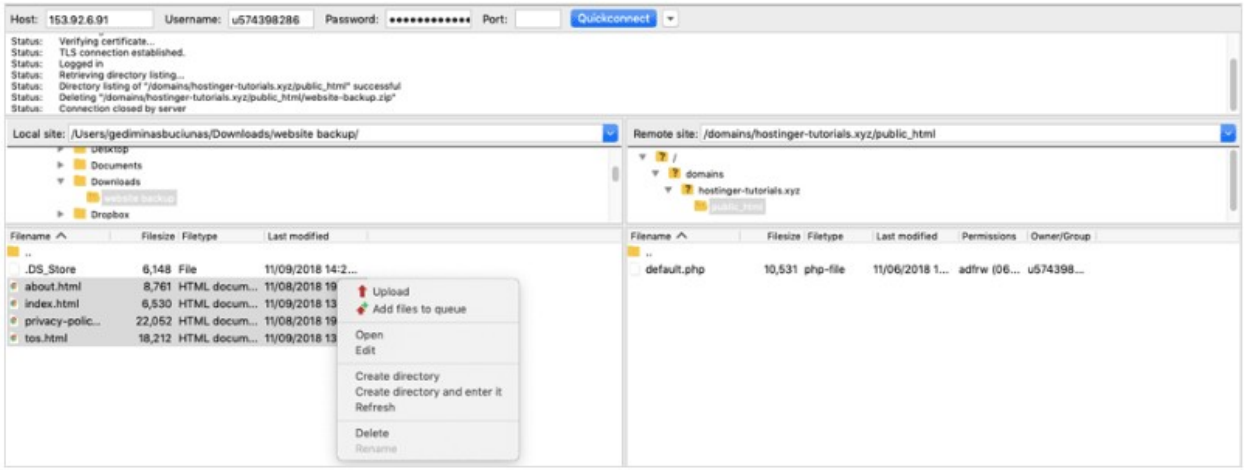


Затем используйте функцию **Извлечь** из верхнего меню, чтобы распаковать архив.



И это подводит итог этому шагу. Следующий будет посвящён тому, чтобы все файлы находились в правильной папке.

Если вы решили использовать **FTP с FileZilla** для загрузки своего сайта, мы рекомендуем предварительно распаковать архив на вашем локальном компьютере (поскольку FTP-клиент не имеет функции извлечения).

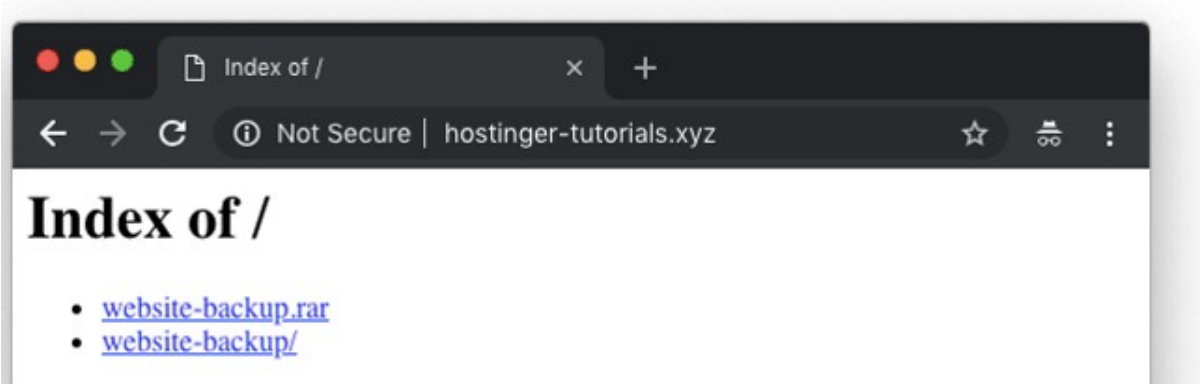


Делая это, вы сможете перенести все файлы непосредственно в **public_html** без дополнительной работы. В противном случае вам нужно будет подключиться через SSH позже и извлечь архив вручную.

Шаг 4. Убедитесь, что все файлы находятся в public_html

На этом этапе вам необходимо убедиться, что все файлы находятся в **корневом каталоге** вашего домена, который называется **public_html**.

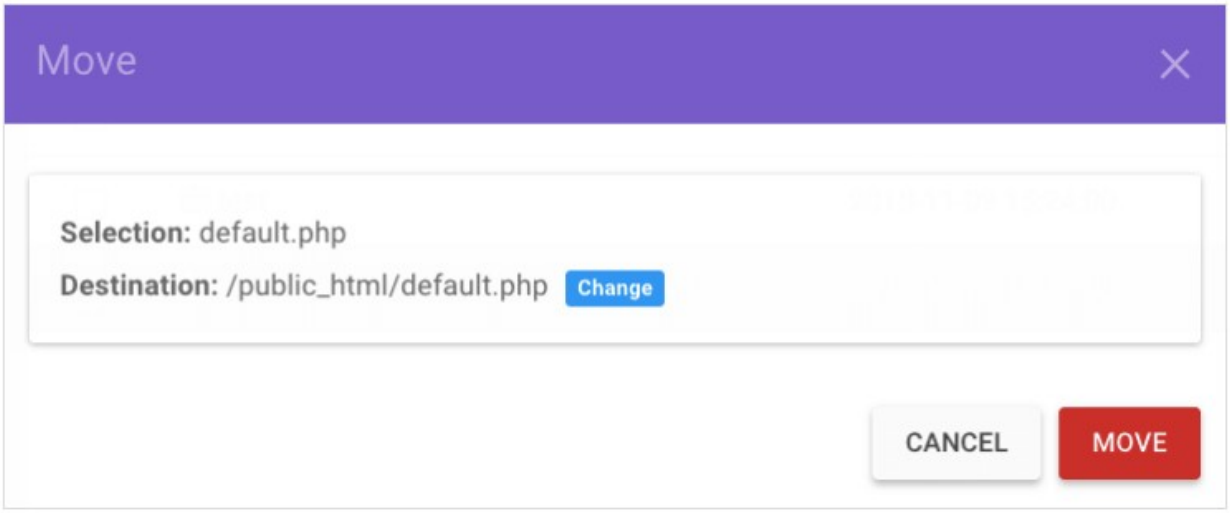
В некоторых случаях при извлечении резервной копии веб-сайта создаётся дополнительный каталог. Это может привести к открытию файлов вашего сайта через **example.com/something** вместо **example.com**



Сообщение “Index of /” означает, что ваши файлы не находятся в правильном каталоге.

Чтобы переместить свой веб-сайт из подпапки в базовый домен (англ), вы можете использовать File Manager или FTP. Выполните следующие короткие действия:

1. Войдите в каталог, в котором находятся ваши файлы.
2. Выберите все из них и нажмите на кнопку **Переместить**.
3. Задайте пункт назначения **public_html** и продолжайте.



Шаг 5. Импорт базы данных MySQL

Если ваш сайт использует базу данных, вам также придётся загрузить его. Например, WordPress использует базы.

Однако, если ваш сайт не использует базу данных MySQL, вы можете просто пропустить эту часть.

Быстрые шаги по импорту базы данных:

1. Создайте новую базу данных MySQL и пользователя.
2. Получите доступ к своей новой базе данных через **phpMyAdmin**.
3. Используйте раздел **Импорт** для загрузки файла резервной копии.
4. Обновите сведения о подключении базы данных MySQL (например, **имя базы данных**, **хост**, **пользователь**, **пароль**) в конфигурационных файлах.

Для более детального подхода ознакомьтесь с нашим руководством по восстановлению базы данных с помощью phpMyAdmin (англ).

Шаг 6: Проверьте, работает ли сайт

Как только файлы веб-сайта загружаются, всё, что вам нужно сделать, это проверить, работает ли сайт (англ). Если ваш домен уже указывает на Hostinger, вам нужно будет только открыть свой домен через браузер.

Имейте в виду, что если ваш домен был указан на наших серверах только недавно, вам может потребоваться до 24 часов, чтобы DNS полностью распространялся по всему миру.

Если домен указывает в другое место (англ), есть несколько способов проверить, всё ли будет работать:

1. **Использование файла hosts.** На вашем компьютере есть специальный файл, который вы можете настроить для эмуляции изменений DNS (англ) (если вы используете MacOS, проверьте это руководство (англ)).
2. **Проверка доступности через онлайн-инструменты** (англ). Их там много! Кроме того, они чрезвычайно просты в использовании. Просто вставьте своё имя домена, и инструмент сделает всё остальное.
3. **Использование плагина браузера.** Если вы настроили расширение, такое как Виртуальные Хосты, вы можете использовать его для проверки DNS-изменений. Всё, что вам нужно — это доменное имя и IP-адрес вашей учётной записи (запись A).

Если вы следовали всем инструкциям, как загрузить свой сайт в интернет, он должен появиться.

Задание: Опубликовать ранее созданный сайт в соответствии с инструкцией:

1. Откройте браузер. Перейдите по ссылке http://www.hostinger.ru/zakazat. Откроется страница для регистрации хостинга Hostinger (рис. 7.3.3).

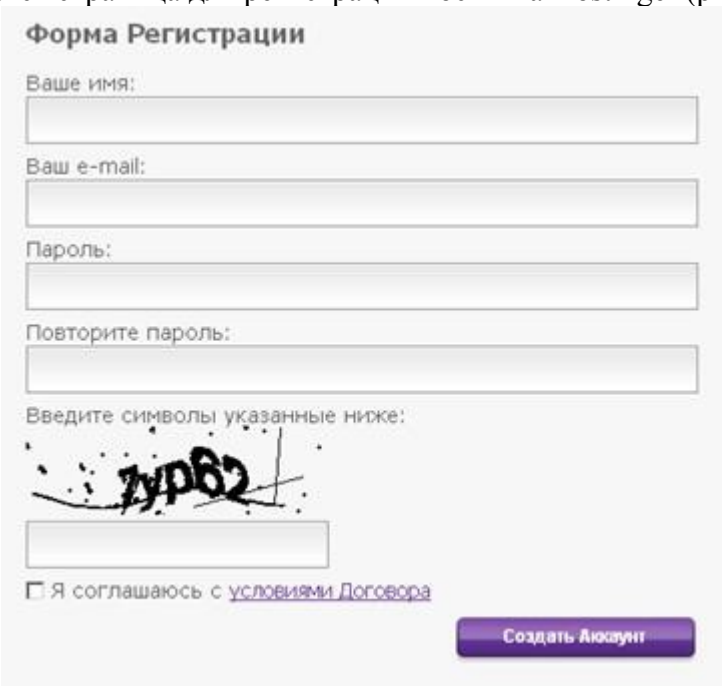


Рис. 7.3.3. Форма регистрации хостинга Hostinger.

2. Заполните поля формы на сайте компании Hostinger. Введите ваше имя и адрес вашего почтового ящика электронной почты. Придумайте легко запоминающийся пароль и дважды введите его в поля формы. Повторно пароль вводится, чтобы избежать ошибки. Пароль – это ключ к вашему аккаунту. Вводите пароль только латиницей. Чтобы пароль был более сложным, используйте прописные и строчные буквенные символы, и цифры. Введите символы, указанные в форме (они называются капча – полностью автоматизированный публичный тест Тьюринга для различения компьютеров и людей), прочитайте условия использования, поставьте соответствующую галочку *Я соглашаюсь с условиями Договора* и нажмите внизу формы регистрации *Создать Аккаунт*.

3. На сайте Hostinger появится информационное сообщение (рис. 7.3.4).

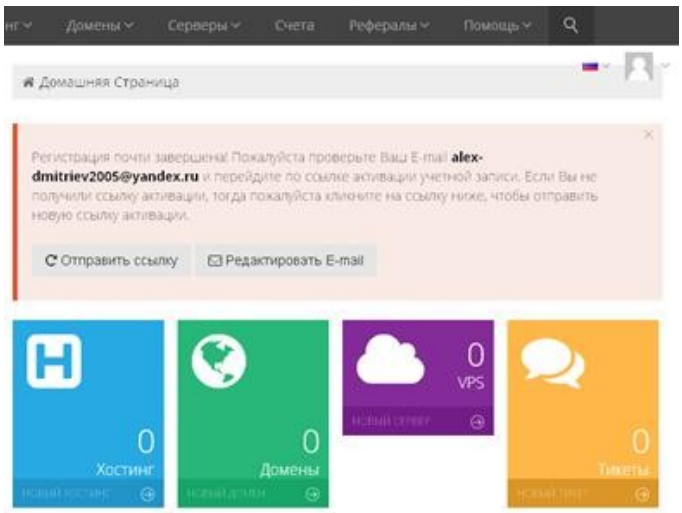


Рис. 7.3.4. Информационное сообщение на сайте Hostinger после создания аккаунта

4. Откройте ваш почтовый ящик, который Вы указывали при регистрации аккауната. Получите два письма от Hostinger.

5. Активируйте профайл. Для этого перейдите по ссылке для активации профайла (во втором письме). Откроется окно на сайте Hostinger (рис. 7.3.5).

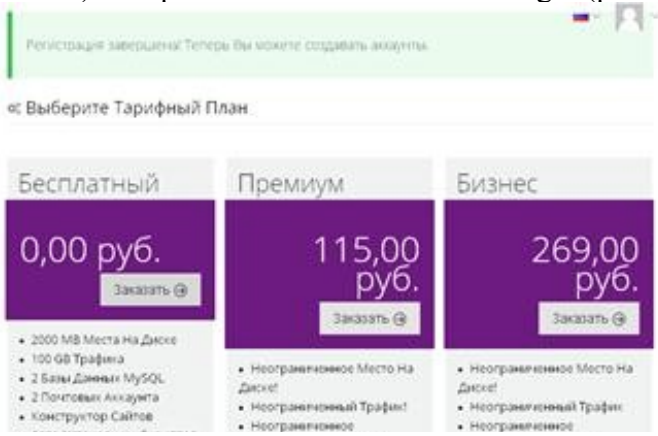


Рис. 7.3.5. Информационное сообщение на сайте Hostinger после активации профайла.

6. Теперь выберите бесплатный хостинговый план и бесплатный субдомен (рис. 7.3.6).

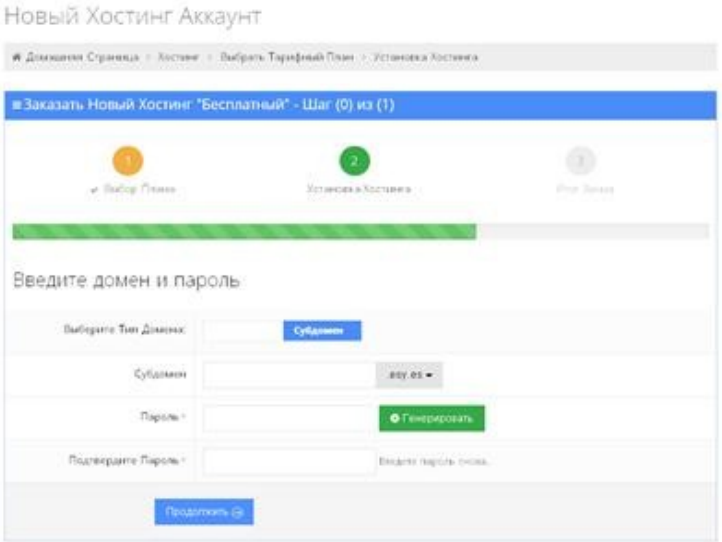


Рис. 7.3.6. Окно для ввода субдомена и пароля на сайте Hostinger.

7. Укажите (введите в соответствующее поле) имя субдомена английскими буквами: фамилию и инициалы, например, dmitrievai. Пример полного адрес вашего сайта: http://dmitrievai.esy.es. Далее в соответствующие поля введите пароль и нажмите кнопку *Продолжить*. В новом открывшемся окне введите символы, указанные на картинке капчи, затем нажмите кнопку *Заказать* (рис. 7.3.7).

Рис. 7.3.7. Окно для подтверждения заказа на сайте Hostinger.

8. Откройте почтовый ящик, который указали при регистрации аккаунта, и прочитайте письмо от компании Hostinger. Перейдите на созданный сайт по ссылке http://cpanel.hostinger.ru, указанной в письме (рис. 7.3.8).

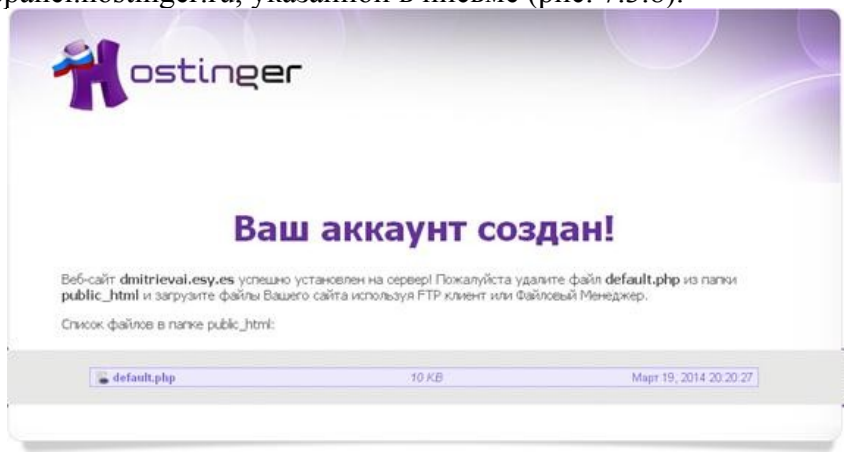


Рис. 7.3.8. Страница сайта Hostinger после завершения процедуры создания аккаунта.

9. Теперь нужно загрузить HTML страницу на сервер хостинга Hostinger. Для этого снова перейдите по ссылке в последнем письме в панель управления аккаунтом: http://cpanel.hostinger.ru. При необходимости авторизуйтесь (рис. 7.3.9).

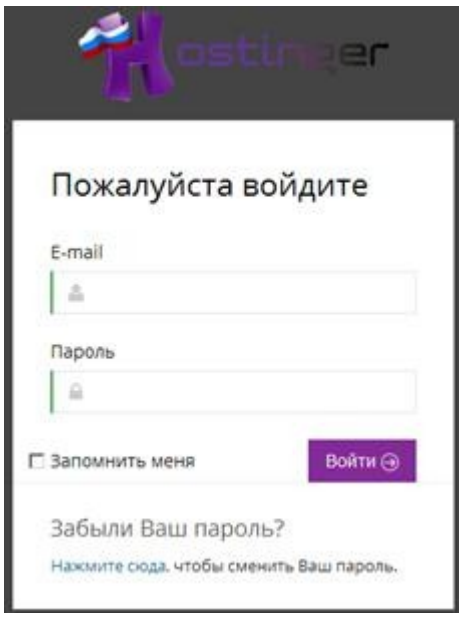


Рис. 7.3.9. Окно авторизации для входа в панель управления аккаунтом на сайте Hostinger.

10. Перейдите к списку хостингов. Для этого нажмите на ссылку *Хостинг* (рис. 7.3.10, рис. 7.3.11).

Рис. 7.3.10. Окно для перехода к списку хостингов на сайте Hostinger.

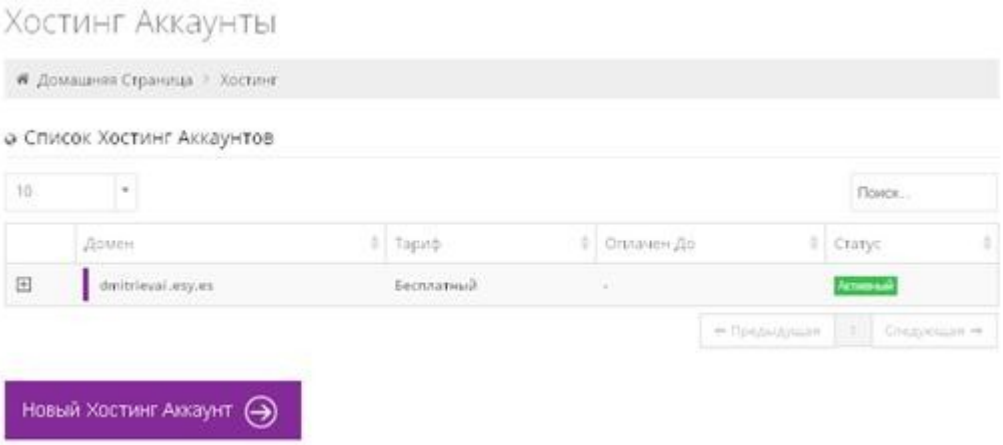


Рис. 7.3.11. Список хостинг аккаунтов.

11. Нажмите на ссылку хостинг аккаунта (строка с заказанным доменом). Затем перейдите по ссылке *Управление* (рис. 7.2.12). На рис. 7.2.13 изображена панель управления хостинг аккаунтом.

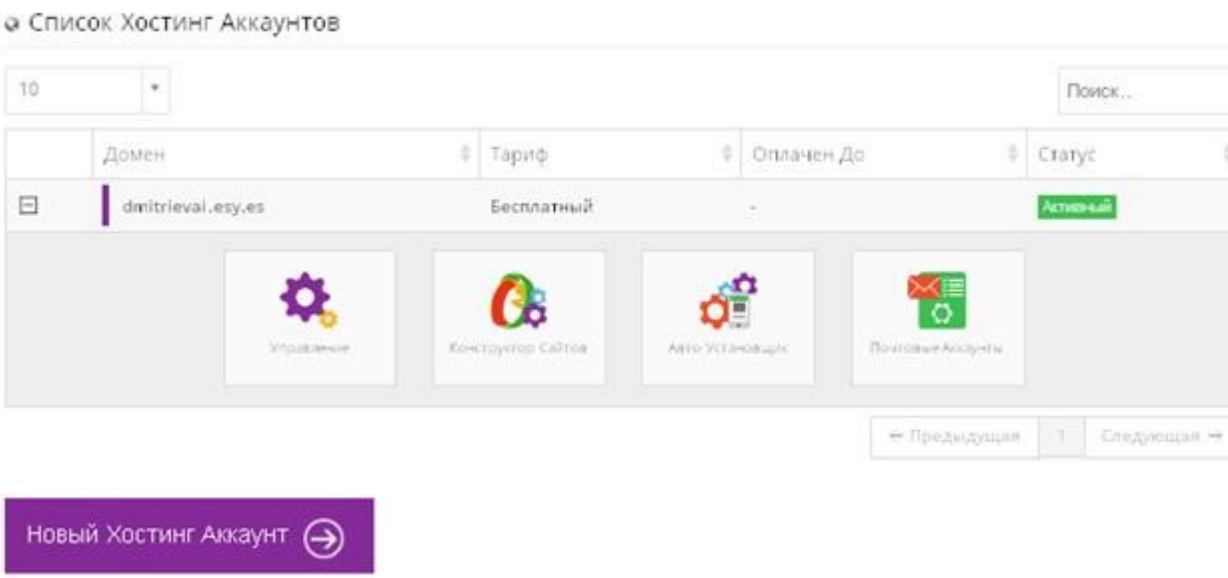


Рис. 7.3.12. Окно перехода в панель управления хостинг аккаунтом.

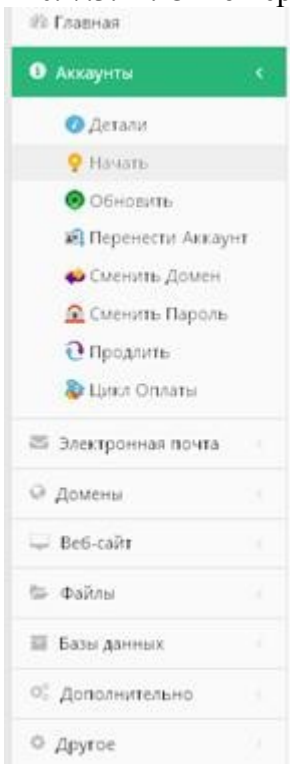


Рис. 7.3.13. Панель управления хостинг аккаунтом.

12. Для того чтобы приступить к загрузке HTML страницы на сервер хостинга нажмите по ссылке *Файлы* панели управления. Затем нажмите ссылку *Файловый менеджер* (рис. 7.3.14).

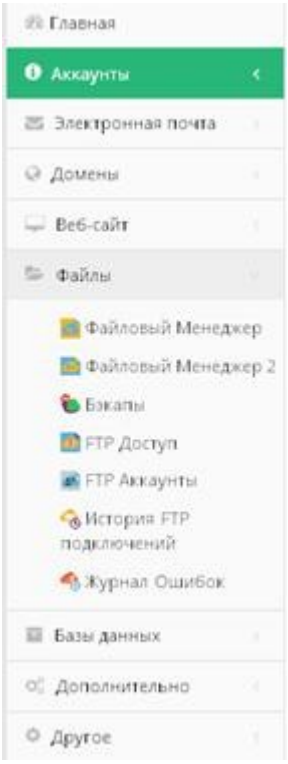


Рис. 7.3.14. Панель управления хостинг аккаунтом.

13. С помощью файлового менеджера удалите все имеющиеся файлы в папке на сервере хостинга Hostinger (рис. 7.3.15).

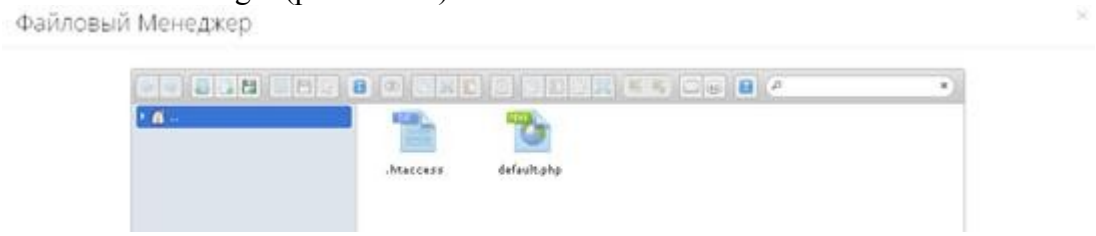


Рис. 7.3.15. Окно файлового менеджера на сервере хостинга Hostinger.

14. С помощью файлового менеджера загрузите HTML страницу и необходимые файлы к ней на сервер хостинга Hostinger (рис. 7.3.16). Закройте файловый менеджер.

Рис. 7.3.16. Окно файлового менеджера на сервере хостинга Hostinger.

15. Выйдете из аккаунта. В адресной строке браузера наберите http://сайт/html страница. Должна открыться страница, аналогичная приведенной на рис. 7.3.17 (http://dmitrievai.esy.es/dmitrievai.html).

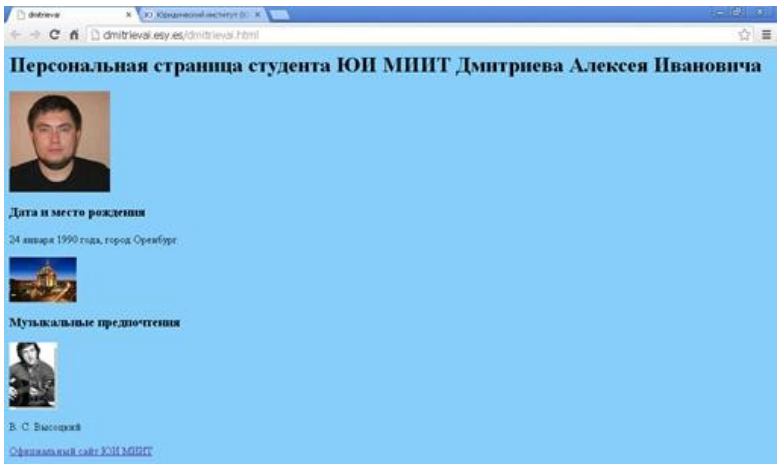


Рис. 7.3.17. HTML страница, открытая в браузере *Google Chrome*

16. Представить преподавателю для проверки ссылку на загруженную страницу.

Содержание отчета

1. Цель
2. Ход работы
3. Выводы

Контрольные вопросы

1. Что такое хостинг?
2. Какие виды хостинга существуют?
3. Какова последовательность публикации сайта на хостинге?
4. Для чего нужны протоколы FTP и HTTP?
5. Что такое FTP-клиент?
6. Какие существуют способы загрузки HTML страниц на сервер хостинга?
7. Что такое авторизация?
8.

Список литературы

1. Duckett J. Web Design with HTML, CSS, JavaScript and jQuery Set. – Wiley Publishing, 2014.
2. Клименко Р. А. Веб-мастеринг на 100%. 2-е изд. – " Издательский дом"" Питер""", 2015.
3. Квинт И. Создаем сайты с помощью HTML, XHTML и CSS на 100%. 3-е изд. – " Издательский дом"" Питер""", 2014.
4. Warren T. PHP Programming For Beginners: The Simple Guide to Learning PHP Fast!. – 2015.
5. Колисниченко Д. Н. PHP и MySQL. Разработка Web-приложений. 4-е изд. – БХВ-Петербург, 2013.
6. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд //СПб.: Питер. – 2013.
7. Mikowski M. S., Powell J. C. Single Page Web Applications //B and W. – 2013.
8. Лиза Г., Джейсон Г. Разработка веб-сайтов для мобильных устройств. – " Издательский дом"" Питер""", 2013.
9. Бессонов Л. В., Брагина И. Г. Операционные системы. Компьютерные сети: Пособие для студентов, обучающихся по дополнительной специальности «Компьютерная графика и веб-дизайн» // Саратов: Изд-во «Научная книга», 2009. – 44 с. – ISBN 978-5-9758-1063-2
10. Бессонов Л. В., Брагина И. Г. Информационные технологии в профессиональной деятельности преподавателя: Учеб. пособие. – Саратов: Изд-во «Научная книга», 2014. – 28 с. – ISBN 978-5-9758-1524-8
11. Бессонов Л. В. Формирование требований к официальному сайту вуза на примере официального сайта СГУ // Наука и образование в XXI веке. Сборник научных трудов по материалам Международной научно-практической конференции 30 января 2015 г.: в 5 частях. Москва, 2015, С. 105–108.
12. Шаталина А.В., Бессонов Л.В. О подходе к задаче автоматизации проверки корректности и полноты входящих в ООП документов // Фундаментальные прикладные исследования в современном мире. 2016. № 13-1. С. 68-71. 13. Бессонов Л.В., Сецинская Е.В., Кухарев В.В. Анализ требований к дизайну сайта вуза для лиц с ограниченными физическими возможностями // Фундаментальные и прикладные исследования в современном мире. 2015. №11-4. С. 85-87.
14. Бессонов Л.В., Тышкевич С.В., Панкратов Д.В. О подходе к формированию персональных страниц преподавателей на официальном сайте вуза // Фундаментальные и прикладные исследования в современном мире. 2015. №11-1. С. 120-123.
15. Дмитриев П.О., Никулин Б.Л. Моделирование системы представления олимпиадных задач на сайте вуза // Новые задачи технических наук и пути их решения: сборник статей Международной научно-практической конференции (10 апреля 2016 г., г. Пермь). Уфа: АЭТЕРНА. 2016. С. 26–28
16. Матершев И.В., Дмитриев П.О. О построении автоматизированной системы управления хостингом для обучающихся по IT-направлениям подготовки // Наука, образование и инновации: сборник статей Международной научно-практической конференции (25 июня 2016 г., г. Томск). В 4 ч. Ч.3. Уфа. 2016. С. 49–51.
17. Бессонов Л.В., Дмитриев О.Ю. Подход к написанию технического задания на разработку сайта поддержки олимпиадного движения по предмету // Фундаментальные и прикладные исследования в современном мире. 2016. №15-1. С. 154-157.
18. Амелин Р. В. Информационные технологии: учебное пособие для студентов механико-математического факультета, обучающегося по специальности 351400

«Прикладная информатика» (по областям). – Саратов: Изд-во Сарат. ун-та, 2006. – 52 с.
19. Амелин Р. В. Мировые информационные ресурсы: учебное пособие для студентов механико-математического факультета, обучающегося по специальности 351400 «Прикладная информатика» (по областям). – Саратов: Изд-во Сарат. ун-та, 2006. – 88 с.
20. Амелин Р.В. Правовой режим государственных информационных систем:монография / под ред. С.Е. Чаннова. М.: ГроссМедиа, 2016. – 338 с.
21. Амелин Р.В. Обязанность по представлению информации в федеральные информационные системы // Административное и муниципальное право. – 2015. – № 10. – С. 1081–1089.
22. Бессонов Л.В. Практикум по веб-программированию. Упражнения и практические задания: Учебно-методическое пособие для студентов, обучающихся по направлению подготовки бакалаврита 09.03.03 «Прикладная информатика» — Саратов: ООО Издательский Центр «Наука», 2016. — 40 с.