Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

ПМ.09 ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И ОПТИМИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЙ
программы подготовки специалистов среднего звена

МДК 09.01 Проектирование и разработка веб-приложений

для студентов специальности
09.02.07 Информационные системы и программирование

КВАЛИФИКАЦИЯ – РАЗРАБОТЧИК ВЕБ И МУЛЬТИМЕДИЙНЫХ ПРИЛОЖЕНИЙ

Магнитогорск, 2020

ОДОБРЕНО
Предметно-цикловой комиссией
«Информатики и вычислительной
техники»
Председатель И.Г.Зорина
Протокол № 7 от 17.02.2020

Методической комиссией МпК

Протокол №3 от «26» февраля 2020г

Составитель:
преподаватель ФГБОУ ВО «МГТУ им. Г.И. Носова» МпК Р.А. Закирова

Методические указания по выполнению лабораторных работ разработаны на основе рабочей программы учебной дисциплины ПМ.09. Проектирование, разработка и оптимизация веб-приложений.

Содержание лабораторных работ ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование и овладению общими компетенциями.

Методические указания по выполнению лабораторных работ разработаны на основе рабочей программы ПМ.09. Проектирование, разработка и оптимизация веб-приложений (очно), МДК 09.01 Проектирование и разработка веб-приложений, МДК 09.02 Оптимизация веб-приложений, МДК 09.03 Обеспечение безопасности веб-приложений.

Содержание лабораторных работ ориентировано на формирование общих и профессиональных компетенций по программе подготовки специалистов среднего звена по специальности 09.02.07 Информационные системы и программирование.

СОДЕРЖАНИЕ

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Состав и содержание лабораторных занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью лабораторных занятий является формирование профессиональных практических умений (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности) или учебных практических умений (умений решать задачи по математике, физике, химии, информатике и др.), необходимых в последующей учебной деятельности.

В соответствии с рабочей программой учебной дисциплины ПМ.09. Проектирование, разработка и оптимизация веб-приложений (очно), МДК 09.01 Проектирование и разработка веб-приложений, МДК 09.02 Оптимизация веб-приложений, МДК 09.03 Обеспечение безопасности веб-приложений предусмотрено проведение лабораторных занятий. В рамках лабораторного занятия обучающиеся могут выполнять одну или несколько лабораторных работ.

В результате их выполнения, обучающийся должен:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;
У2. осуществлять оптимизацию веб-приложения с целью повышения его рейтинга в сети Интернет;
У3. разрабатывать и проектировать информационные системы;
У4. проводить анкетирование;
У5. проводить интервьюирование;
У6. оформлять техническую документацию;
У7. осуществлять выбор одного из типовых решений;
У8. работать со специализированным программным обеспечением для планирования времени и организации работы с клиентами;
У9. использовать язык разметки страниц веб-приложения;
У10. оформлять код программы в соответствии со стандартом кодирования;
У11. использовать объектные модели веб-приложений и браузера;
У12. использовать открытые библиотеки (framework);
У13. использовать выбранную среду программирования и средства системы управления базами данных;
У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений;
У16. использовать объектные модели веб-приложений и браузера;
У17. разрабатывать анимацию для веб-приложений для повышения его доступности и визуальной привлекательности (Canvas);
У18. подключать и настраивать системы мониторинга работы веб-приложений и сбора статистики его использования;
У19. устанавливать и настраивать веб-сервера, СУБД для организации работы веб-приложений;
У20. работать с системами Helpdesk;
У21. анализировать и решать типовые запросы заказчиков;
У22. выполнять регламентные процедуры по резервированию данных;
У23. устанавливать прикладное программное обеспечение для резервирования веб-приложений;
У24. выполнять отладку и тестирование программного кода;
У25. выполнять оптимизацию и рефакторинг программного кода;
У26. тестировать веб-приложения с использованием тест-планов;
У27. применять инструменты подготовки тестовых данных;
У28. выбирать и комбинировать техники тестирования веб-приложений;
У29. работать с системами контроля версий в соответствии с регламентом использования системы контроля версий;

У30. выполнять проверку веб-приложения по техническому заданию;
У31. выбирать хостинг в соответствии с параметрами веб-приложения;
У32. составлять сравнительную характеристику хостингов;

Содержание лабораторных работ ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению ***профессиональными компетенциями:***

ПК 9.1 Разрабатывать техническое задание на веб-приложение в соответствии с требованиями заказчика

ПК 9.2 Разрабатывать веб-приложение в соответствии с техническим заданием

ПК 9.3 Разрабатывать интерфейс пользователя веб-приложений в соответствии с техническим заданием

ПК 9.4 Осуществлять техническое сопровождение и восстановление веб-приложений в соответствии с техническим заданием

ПК 9.5 Производить тестирование разработанного веб приложения

ПК 9.6 Размещать веб приложения в сети в соответствии с техническим заданием

А также формированию ***общих компетенций***:

ОК 1. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам

ОК 2. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 3 Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 4 Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 5 Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 6 Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей

ОК 7 Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.

ОК 8 Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности

ОК 9 Использовать информационные технологии в профессиональной деятельности.

ОК 10 Пользоваться профессиональной документацией на государственном и иностранном языке

2 ПЕРЕЧЕНЬ ЛАБОРАТОРНЫХ ЗАНЯТИЙ

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Разделы/темы	**Темы лабораторных занятий**	**Количество часов**	**Требования ФГОС СПО (уметь)**
Раздел 1. **Проектирование и разработка веб-приложений**			
Тема 1.1 Разработка сетевых приложений (клиентская часть)		**32**	
	Л.Р.№ 1 «Использование языка сценариев JavaScript при создании web-сайта»	16	У1, У3, У4, У5, У6, У7, У8, У9, У10, У11, У12, У16, У17, У20, У21, У24, У25, У26, У27, У28, У30, У31, У32, У01.2, У01.9, У02.4, У05.3, У09.1, У09.2, У10.1, У10.6, У03.2
	Л.Р.№ 2«Применение технологии AJAX»	8	
	Л.Р.№3 «Использование библиотеки jQuery»	8	
Тема 1.2 Разработка сетевых приложений (серверная часть)		**50**	
	Л.Р.№ 4 «Создание серверных сценариев с использованием технологии PHP»	4	У1, У3, У4, У5, У6, У7, У8, У9, У10, У11, У12, У13, У14, У16, У17, У18, У19, У20, У21, У22, У23, У24, У25, У26, У27, У28, У29, У30, У31, У32, У01.2, У01.9, У02.4, У05.3, У09.1, У09.2, У10.1, У10.6, У03.2
	Л.Р.№5 «Обработка данных на форме»	4	
	Л.Р.№6 «Организация файлового ввода-вывода»	4	
	№7 «Организация поддержки базы данных в PHP»	4	
	Л.Р.№8 «Отслеживание сеансов (session)»	4	
	Л.Р.№9 «Создание проекта «Регистрация»	4	
	Л.Р.№10 «Создание проекта «Интернет магазин»	4	
	Л.Р.№11 «Составление схем XML-документов»	2	
	Л.Р.№12 «Отображение XML-документов различными способами»	2	
	Л.Р.№13 «Разработка Web-приложения с помощью XML»	2	
	Л.Р.№14 «Использование фреймворка для создания сайта»	6	
	Л.Р.№15 «Создание сайта на CMS»	4	
	Л.Р.№16 «Администрирование	4	

	сайта»		
	Л.Р.№17 «Публикация сайта на бесплатном хостинге»	2	
ИТОГО		**82**	

3 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.1 Разработка сетевых приложений (клиентская часть)

Лабораторная работа № 1
«Использование языка сценариев JavaScript при создании web-сайта»

Цель: Ознакомить с основами языка JavaScript**.** Ознакомить студентов с принципами работы с формами, иерархией объектов веб-страницы. Рассмотреть приемы создания документов, содержащих фреймы. Ознакомиться с принципами использования адресации для фреймов. Ознакомиться с принципами работы с окнами и динамическим управлением документами, научить использовать свойства окон при создании веб-страниц и создавать документы, изменяющие свойства других документов

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У3. разрабатывать и проектировать информационные системы;

У9. использовать язык разметки страниц веб-приложения;

У10. оформлять код программы в соответствии со стандартом кодирования;

У11. использовать объектные модели веб-приложений и браузера;

У12. использовать открытые библиотеки (framework);

У13. использовать выбранную среду программирования и средства системы управления базами данных;

У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание:

1 Ознакомьтесь с теоретическими аспектами темы.

2 Создайте простую веб-страницу с использованием JavaScript согласно методическим указаниям.

3 Создайте веб-страницу с формой и кнопкой на основе JavaScript согласно методическим указаниям.

4 Напишите скрипт, печатающий текст «Добро пожаловать на мою страницу! Это JavaScript» три раза подряд. Инструкция по выполнению задания представлена в методических указаниях.

5 Создайте веб-страницу с использованием функции calculation().

6 Ознакомиться с аспектом темы создания объектов.

7 Создайте веб-страницу с использованием принципов иерархии объектов.

8 Создайте документ с использованием объектов.

9 Ознакомиться с теоретическими аспектами темы окна и динамическое управление документами

10 Создайте веб-страницу с несколькими ссылками в одном фрейме.

11 Создайте веб-страницу, в которой в одном фрейме создайте несколько ссылок, но если посетитель активирует какую-либо из них, соответствующая страница будет помещена не в тот же самый фрейм, а в соседний.

12 Ознакомиться с теоретическими аспектами темы.

13 Создайте веб-страницу, в которой в новое окно с помощью метода open() записывается другая страница.

14 Создайте веб-страницу, в которой производится создание нового окна фиксированного размера.

15 Создать документ с использованием методов объекта window.

16 Создать документ с использованием команд генерации нового документа.

Краткие теоретические сведения:

JavaScript - новый язык для составления скриптов, разработанный фирмой Netscape. С помощью JavaScript Вы можете легко создавать интерактивные Web-страницы.

Для запуска скриптов, написанных на языке JavaScript, нужен браузер, способный работать с JavaScript - например Netscape Navigator (начиная с версии 2.0) или Microsoft Internet Explorer (MSIE - начиная с версии 3.0). С тех пор, как оба этих браузера стали широко распространеными, множество людей получили возможность работать со скриптами, написанными на языке JavaScript. Код скрипта JavaScript размещется непосредственно на HTML-странице. Все, что стоит между тэгами <script> и </script>, интерпретируется как код на языке JavaScript. Инструкция*document.write()* - одна из наиболее важных команд, используемых при программировании на языке JavaScript. Команда *document.write()* используется, когда необходимо что-либо написать в текущем документе (в данном случае таком является наш HTML-документ).

События и обработчики событий являются очень важной частью для программирования на языке JavaScript. События, главным образом, инициируются теми или иными действиями пользователя. Если он щелкает по некоторой кнопке, происходит

событие *"Click"*. Если указатель мыши пересекает какую-либо ссылку гипертекста - происходит событие *MouseOver*. Существует несколько различных типов событий. Мы можем заставить нашу JavaScript-программу реагировать на некоторые из них. И это может быть выполнено с помощью специальных программ обработки событий. Так, в результате щелчка по кнопке может создаваться выпадающее окно. Это означает, что создание окна должно быть реакцией на событие щелка - *Click*. Программа - обработчик событий, которую мы должны использовать в данном случае, называется *onClick*. И она сообщает компьютеру, что нужно делать, если произойдет данное событие.

Вы можете использовать в скрипте множество различных типов функций обработки событий. В большинстве случаев функции представляют собой лишь способ связать вместе нескольких команд. Функции могут также использоваться в совместно с процедурами обработки событий.

Порядок выполнения работы

Методические рекомендации к выполнению задания 1-2

1. Запустите блокнот
2. Введите текст

```
<html>
<body>
<br>
Это обычный HTML документ.
<br>
<scriptlanguage="JavaScript">
    document.write("Аэто JavaScript!")
</script>
<br>
Вновь документ HTML.
</body>
</html>
```

3. Сохраните документ в формате html
4. Запустите страницу в окне браузера.

Если браузер не поддерживает JavaScript, то он проигнорирует тег <script>. В этом случае измените исходный текст:

```
<html>
<body>
<br>
Это обычный HTML документ.
<br>
<script language="JavaScript">
<!-- hide from old browsers
    document.write("Аэто JavaScript!")
// -->
</script><br>
Вновь документ HTML.
</body>
</html>
```

В этом случае использован тег комментария из HTML - <!-- -->. В результате новый вариант нашего исходного кода будет выглядеть как:

Это обычный HTML документ.

Вновь документ HTML.

Методические рекомендации к выполнению задания 3

1. Создайте новый документ html.
2. Вставьте следующий код

```
<form>
<input type="button" value="Click me" onClick = "alert('Yo')"> </form>
```

3. Просмотрите результат.

В данном примере создается некая форма с кнопкой. Первая новая особенность - onClick="alert('Yo')" в тэге <input>. Таким образом, если имеет место событие *Click*, компьютер должен выполнить вызов alert('Yo'). Это и есть пример кода на языке JavaScript.

Функция *alert()* позволяет Вам создавать выпадающие окна. При ее вызове Вы должны в скобках задать некую строку. В нашем случае это *'Yo'*. И это как раз будет тот текст, что появится в выпадающем окне. Таким образом, когда читатель когда щелкает на кнопке, наш скрипт создает окно, содержащее текст *'Yo'*.

Еще одна особенность данного примера: в команде document.write() мы использовали двойные кавычки ("), а в конструкции alert() - только одинарные.

В большинстве случаев Вы можете использовать оба типа кавычек. Однако в последнем примере мы написали onClick="alert('Yo')" - то есть мы использовали и двойные, и одинарные кавычки. Если бы мы написали onClick="alert("Yo")", то компьютер не смог бы разобраться в нашем скрипте, поскольку становится неясно, к которой из частей конструкции имеет отношение функция обработки событий onClick, а к которой - нет. Поэтому мы вынуждены использовать оба типа кавычек. Не имеет значения, в каком порядке Вы использовали кавычки - сначала двойные, а затем одинарные или наоборот. То есть Вы можете точно так же написать и onClick='alert("Yo")'.

Методические рекомендации к выполнению задания 4

1. Напишите скрипт, печатающий текст «Добро пожаловать на мою страницу! Это JavaScript» три раза подряд. Для начала рассмотрим простой подход:

```
<html>
<script language="JavaScript"> <!-
- hide
document.write("Добро пожаловать на мою страницу!<br>");
document.write("Это JavaScript!<br>"); document.write("Добро
пожаловать на мою страницу!<br>"); document.write("Это
JavaScript!<br>"); document.write("Добро пожаловать на мою
страницу!<br>");document.write("Это JavaScript!<br>");
// -->
</script
>
</html>
```

2. Если посмотреть на исходный код скрипта, то видно, что для получения необходимого результата определенная часть его кода была повторена три раза. Эту же задачу можно решить несколько иначе. Введите изменения:

```
<html>
<scriptlanguage="JavaScript">
<!-- hide functionmyFunction() {
```

```
document.write("Добро пожаловать на мою страницу!<br>");
document.write("Это JavaScript!<br>");}
myFunction();
myFunction();
myFunction();
// --> </script>
</html>
```

В этом скрипте мы определили некую функцию, состоящую из следующих строк:

```
function myFunction() {
  document.write("Добро пожаловать на мою страницу!<br>");
document.write("Это JavaScript!<br>");}
```

Все команды скрипта, что находятся внутри фигурных скобок - {} - принадлежат функции *myFunction()*. Это означает, что обе команды document.write() теперь связаны воедино и могут быть выполнены при вызове указанной функции. И действительно, в нашем примере есть три вызова этой функции В свою очередь, это означает, что содержимое этой функции (команды, указанные в фигурных скобках) было выполнено трижды.

Методические рекомендации к выполнению задания 5

Создайте новую веб-страничку:

```
<html>
<head>
<script language="JavaScript">
<!-- hide
function calculation() {

  var x= 12;
  var y= 5;
  var result= x + y;
  alert(result);}

// -->
</script
```

```
></head
>

<body>
<form>
<input type="button" value="Calculate" onClick="calculation()">
</form>
</body>
</html>
```

Здесь при нажатии на кнопку осуществляется вызов функции *calculation().* Как можно заметить, эта функция выполняет некие вычисления, пользуясь переменными *x, y* и *result*. Переменную мы можем определить с помощью ключевого слова *var*. Переменные могут использоваться для хранения различных величин - чисел, строк текста и т.д. Так строка скрипта var result= x + y; сообщает браузеру о том, что необходимо создать переменную *result* и поместить туда результат выполнения арифметической операции *x + y* (т.е. 5 + 12). После этого в переменный *result* будет размещено число *17*. В данном случае команда alert(result) выполняет то же самое, что и alert(17). Иными словами, мы получаем выпадающее окно, в котором написано число 17.

Вопросы для самоконтроля:

1 Для чего предназначен язык JavaScript?

2 Что называют инструкциями?

3 В чем отличие процедур от событий?

Методические рекомендации к выполнению задания 6

В языке JavaScript все элементы на web-странице выстраиваются в иерархическую структуру. Каждый элемент предстает в виде объекта. И каждый такой объект может иметь определенные свойства и методы. В свою очередь, язык JavaScript позволяет легко управлять объектами web-страницы, хотя для этого очень важно понимать иерархию объектов, на которые опирается разметка HTML.

С точки зрения языка JavaScript окно браузера - это некий объект window. Этот объект также содержит в свою очередь некоторые элементы оформления, такие как строка состояния. Внутри окна можно разместить документ HTML или файл какого-либо другого типа. Такая страница является объектом document. Это означает, что объект document

представляет в языке JavaScript загруженный на настоящий момент документ HTML. Объект document является очень важным объектом в языке JavaScript. К свойствам объекта document относятся, например, цвет фона для web-страницы. Все без исключения объекты HTML являются свойствами объекта document. Примерами объекта HTML являются, к примеру, ссылка или заполняемая форма.

Чтобы иметь возможность получать информацию о различных объектах этой иерархии и управлять ею мы должны знать, как в языке JavaScript организован доступ к различным объектам. Каждый объект иерархической структуры имеет свое имя. Следовательно, если нужно узнать, как можно обратиться к первому рисунку на HTML-странице, то обязаны сориентироваться в иерархии объектов. И начать нужно с самой вершины. Первый объект такой структуры называется *document*. Первый рисунок на странице представлен как объект *images[0]*. Это означает, что отныне мы можем получать доступ к этому объекту, записав в JavaScript *document.images[0]*. Если же, например, нужноузнать, какой текст ввел читатель в первый элемент формы, то нужно выяснить, как получить доступ к этому объекту. И снова начинаем с вершины нашей иерархии объектов. Затем нужно проследить путь к объекту с именем *elements[0]* и последовательно записать названия всех объектов, которые минуем. В итоге выясняется, что доступ к первому полю для ввода текста можно получить, записав: *document.forms[0].elements[0]*

Чтобы узнать текст, введенный читателем нужно написать на языке JavaScript строку:

name= document.forms[0].elements[0].value;

Полученная строка заносится в переменную *name*. Следовательно, теперь мы можем работать с этой переменной, как нам необходимо. Например, мы можем создать выпадающее окно, воспользовавшись командой alert("Hi " + name). В результате, если читатель введет в это поле слово *'Stefan'*, то по команде alert("Hi " + name) будет открыто выпадающее окно с приветствием *'Hi Stefan'*.

Если Вы имеете дело с большими страницами, то процедура адресации к различным объектам по номеру может стать весьма запутанной. Например, придется решать, как следует обратиться к объекту *document.forms[3].elements[17]document.forms[2].elements[18]*? Во избежание подобной

проблемы, можно самим присваивать уникальные имена различным объектам. Посмотрим на примере, как это делается:

```
<formname="myForm">
Name:
<input type="text" name="name" value=""><br>...
```

Эта запись означает, что объект *forms[0]* получает теперь еще и второе имя - *myForm*. Точно так же вместо *elements[0]* можно писать *name* (последнее было указано в атрибуте name тэга <input>). Таким образом, вместо

```
name= document.forms[0].elements[0].value; Вы
можете записать

name= document.myForm.name.value;
```

Это значительно упрощает программирование на JavaScript, особенно в случае с большими web-страницами, содержащими множество объектов. При написании имен нужно следить и за положением регистра - то есть нельзя написать *myform* вместо *myForm*. В JavaScript многие свойства объектов доступны не только для чтения. Имеетсявозможность записывать в них новые значения. Например, посредством JavaScript можно записать в упоминавшееся поле новую строку.

Пример кода на JavaScript, иллюстрирующего такую возможность - интересующий нас фрагмент записан каксвойство onClick второго тэга <input>:

```
<form name="myForm">
<input type="text" name="input" value="Привет!!!">

<input type="button" value="Write"
  onClick="document.myForm.input.value= 'Yo!'; ">
```

Кроме объектов *window* и *document* в JavaScript имеется еще один важный объект - *location*. В этом объекте представлен адрес загруженного

HTML-документа. Например, если Вы загрузили страницу *http://www.xyz.com/page.html,* то значение *location.href* как раз и будет соответствовать этому адресу.

В *location.href* можно записывать свои новые значения. Например, в данном примере кнопка загружает в текущее окно новую страницу:

```
<form>

<input type=button value="Yahoo"
  onClick="location.href='http://www.yahoo.com'; ">
</form>
```

Методические рекомендации к выполнению задания 7

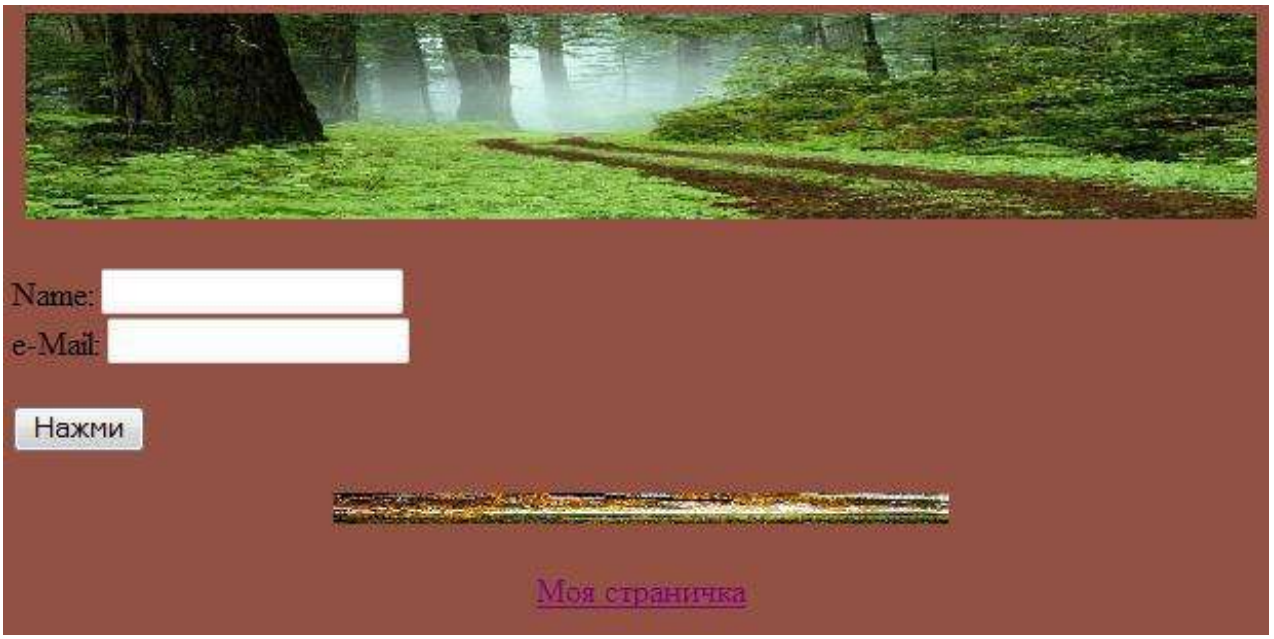
1. Создайте два произвольных рисунка формата gif. Сохраните их в отдельной папке на рабочем столе под названиями home.gif и ruler.gif.

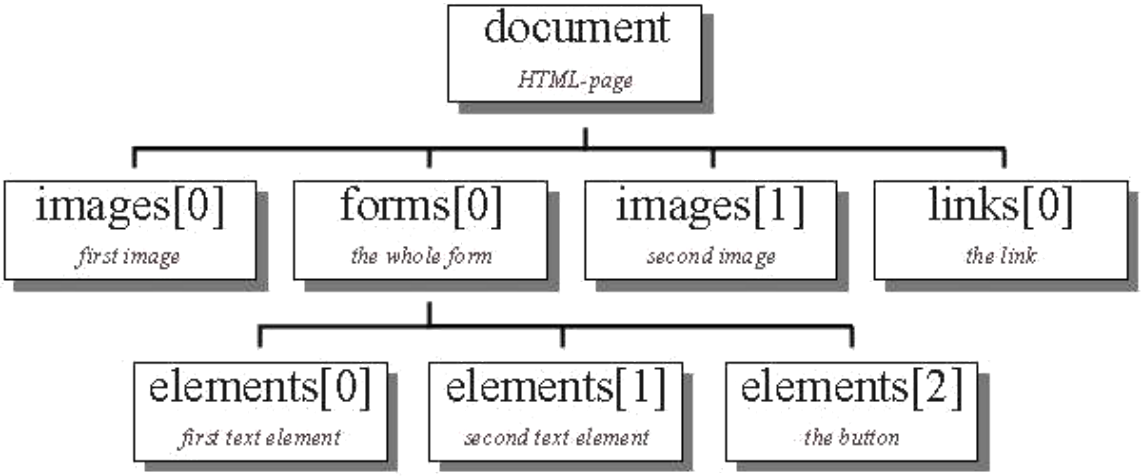
2. Создайте HTML-страницу:

```
<html>
<head>
</head>
<body bgcolor=#925142>

<center>
<img src="home.gif" name="pic1" width=600 height=100>
</center>
<p>
<form name="myForm">
Name:
```

```
<input type="text" name="name" value=""><br> e-
Mail:

<input type="text" name="email" value=""><br><br> <input
type="button" value="Нажми" name="myButton"
onClick="alert('Спасибо!')">

</form>

<p>
<center>

<img src="ruler.gif" name="pic4" width=300 height=15> <p>

<a href="http://rummelplatz.uni-mannheim.de/~skoch/">Моястраничка</a>
</center>

</body>
</html>
```

3. Проверьте результат.





Итак, мы имеем два рисунка, одну ссылку и некую форму с двумя полями для ввода текста и одной кнопкой. На следующем рисунке иллюстрируется иерархия объектов, создаваемая HTML-страницей из нашего примера:

Методические рекомендации к выполнению задания 8

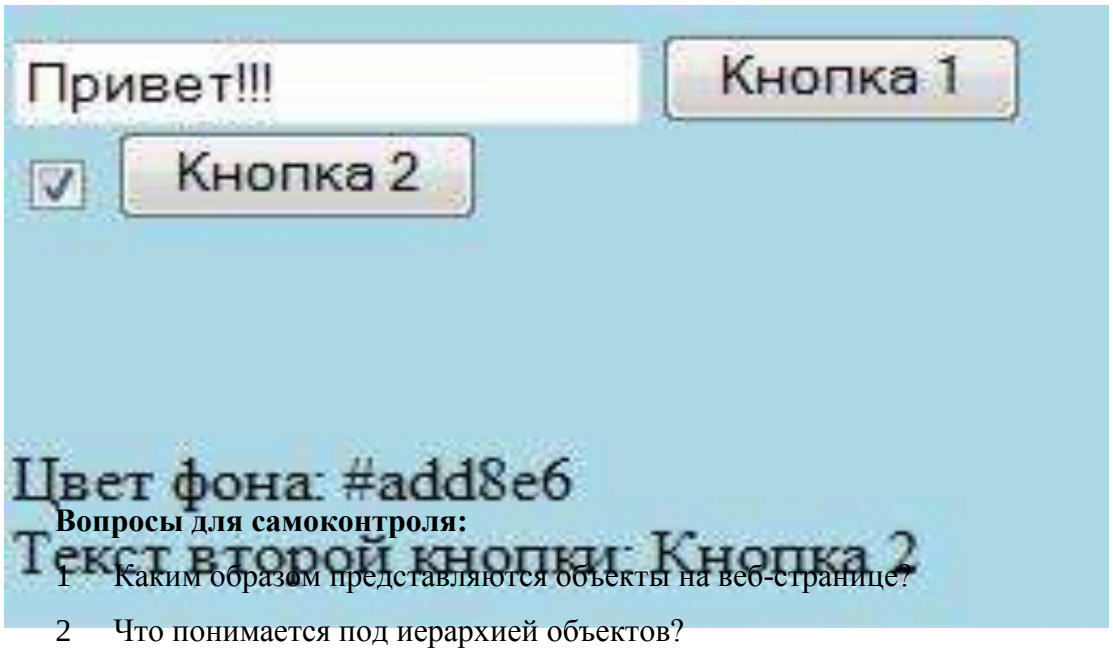
1. Создайте исходный код скрипта:

```
<html>
<head>
<title>Objects</title>

<script language="JavaScript"> <!-- hide

function first() {

// создает выпадающее окно, где размещается // текст,
введенный в поле формы

alert("Текст: " + document.myForm.myText.value); } function
second() {

// данная функция проверяет состояние переключателей varmyString=
"Переключатель ";

// переключатель включен, или нет?
```

```
if (document.myForm.myCheckbox.checked) myString+= "включен"
    elsemyString+= "отключен";

// вывод строки на экран
alert(myString); }

// --> </script>
</head>

<body   bgcolor=lightblue>   <form
name="myForm">

<input type="text" name="myText" value="Привет!!!">

<input type="button" name="button1" value="Кнопка 1"
  onClick="first()">

<br>

<input type="checkbox" name="myCheckbox" CHECKED> <input
type="button" name="button2" value="Кнопка 2"

  onClick="second()">
</form><p><br><br>

    <script language="JavaScript">
    <!-- hide
    document.write("Цветфона: ");
    document.write(document.bgColor + "<br>");
    document.write("Текст второй кнопки: ");
    document.write(document.myForm.button2.value);
```

```
// -->
</script
>
</body>
</html>
```

3. Сравните результат



Вопросы для самоконтроля:

1 Каким образом представляются объекты на веб-странице?

2 Что понимается под иерархией объектов?

3 Для чего предназначен объект document?

4 Что относится к свойствам объекта document?

5 Приведите примеры объектов HTML.

6 Каким образом организуется управление иерархией объектов?

7 Какой скрипт нужно написать для того, чтобы узнать текст, введенный читателем?

8 Какие функции выполняет объект location?

9 Для чего предназначен объект window?

10 К какому виду объектов принадлежит окно браузера с точки зрения JavaScript?

Методические рекомендации к выполнению задания 9

В общем случае окно браузера может быть разбито в несколько отдельных фреймов. Это означает, что фрейм определяется как некое выделенное в окне браузера поле

в форме прямоугольника. Каждый из фреймов выдает на экран содержимое собственного документа. Таким образом, можно, к примеру, создать два фрейма. В первый такой фрейм загрузить "домашнюю страницу" фирмы Netscape, а во второй - фирмы Microsoft. Для создания фреймов необходимы два тэга: <frameset> и <frame>. HTML-страница, создающая два фрейма, в общем случае может выглядеть следующим образом:

```
<html>
<frameset rows="50%,50%">

<frame    src="page1.htm"    name="frame1">
<frame    src="page2.htm"    name="frame2">
</frameset>
</html>
```

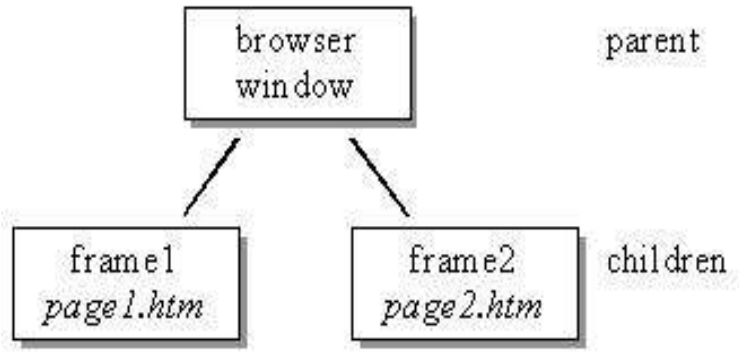
В результате будут созданы два фрейма. Во фрейме <frameset> мы используем свойство *rows*. Это означает, два фрейма будут расположены друг над другом. В верхний фрейм будет загружена HTML-страница *page1.htm*, а в нижнем фрейме разместится документ *page2.htm*.

Если нужно, чтобы документы располагались не друг над другом, а рядом, то следует в тэге <frameset> писать не *rows*, а *cols*. Фрагмент *"50%,50%"* сообщает, насколько велики должны быть оба получившихся окна. Вы имеете также возможность записать *"50%,*"*, если не хотите утруждать себя расчетами, насколько велик должен быть второй фрейм, чтобы в сумме получалась все те же 100%. Вы можете также задать размер фрейма в пикселях, для чего достаточно после числа не ставить символ %.

Любому фрейму можно присвоить уникальное имя, воспользовавшись в тэге <frame> атрибутом *name*. Такая возможность пригодится в языке JavaScript для доступа к фреймам.

При создании web-страниц можно использовать несколько вложенных тэгов <frameset>.

JavaScript организует все элементы, представленные на web-странице, в виде некой иерархической структуры. То же самое относится и к фреймам. На следующем рисунке показана иерархия объектов, представленных в первом примере:



В вершине иерархии находится окно браузера (browser window). В данном случае он разбито на два фрейма. Таким образом, окно, как объект, является родоначальником, родителем данной иерархии (parent), а два фрейма - соответственно, его потомки (children). Мы присвоили этим двум фреймам уникальные имена - *frame1* и *frame2*. И с помощью этих имен можно обмениваться информацией с двумя указанными фреймами.

С помощью скрипта можно решить следующую задачу: допустим посетитель активирует некую ссылку в первом фрейме, однако соответствующая страница должна загружаться не в этот же фрейм, а в другой. Примером такой задачи может служить составление меню (или навигационных панелей), где один фрейм всегда остается неизменным, но предлагает посетителю несколько различных ссылок для дальнейшего изучения данного сайта.

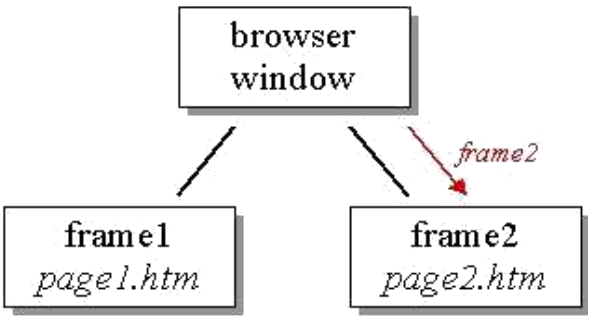
Чтобы решить эту задачу, нужно рассмотреть на три случая:

- главное окно/фрейм получает доступ к фрейму-потомку
- фрейм-потомок получает доступ к родительскому окну/фрейму
- фрейм-потомок получает доступ к другому фрейму-потомку

С точки зрения объекта "окно" (window) два указанных фрейма называются *frame1* и *frame2*. Как можно видеть на предыдущем рисунке, существует прямая

взаимосвязь между родительским окном и каждым фреймом. Так образом, если Вы пишете скрипт для родительского окна - то есть для страницы, создающей эти фреймы - то можете обращаться к этим фреймам, просто называя их по имени. Например, можно написать:

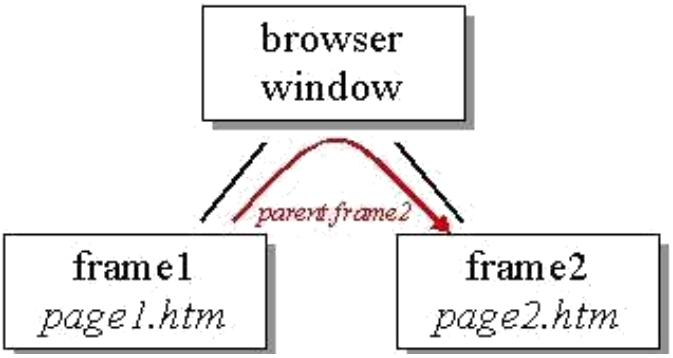
frame2.document.write("Это сообщение передано от родительского окна.");



В некоторых случаях понадобится, находясь во фрейме, получать доступу к родительскому окну. Например, это бывает необходимо, если нужно при следующем переходе избавиться от фреймов. В таком случае удаление фреймов означает лишь загрузку новой страницы вместо содержавшей фреймы.

В нашем случае это загрузка страницы в родительское окно. Сделать это нам поможет доступ к родительскому- *parent* - окну (или родительскому фрейму) из фреймов, являющихся его потомками. Чтобы загрузить новый документ, мы должны внести в *location.href* новый адрес URL. Поскольку нужно избавиться от фреймов, следует использовать объект location из родительского окна. (Напомним, что в каждый фрейм можно загрузить собственную страницу, то мы имеем для каждого фрейма собственный объект location). Итак, можно загрузить новую страницу в родительское окно с помощью команды:

parent.location.href= "http://...";



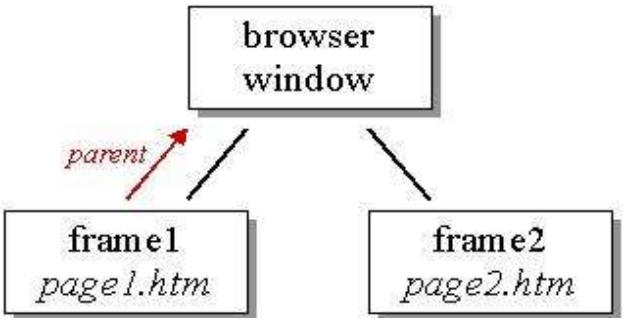
И наконец, очень часто Вам придется решать задачу обеспечения доступа с одного фрейма-потомка к другому такому же фрейму-потомку. Итак, как можно, находясь в первом фрейме, записать что-либо во второй - то есть, которой командой следует воспользоваться на HTML-странице *page1.htm*? Как можно увидеть на рисунке, между двумя этими фреймами нет никакой прямой связи. И потому мы не можем просто так вызвать *frame2*, находясь в фрейме *frame1*. С точки же зрения родительского окна второй фрейм действительно существует и называется *frame2*, а к самому родительскому окну можно обратиться из первого фрейма по имени *parent*. Таким образом, чтобы получить доступ к объекту document, разместившемуся во втором фрейме, мы должны написать следующее,:

```
parent.frame2.document.write("Привет, это вызов из первого фрейма.");
```

Предположим, мы имеем три фрейма с именами *frame1, frame2* и *frame3*.Допустим посетитель активирует ссылку в *frame1*. Нужно, чтобы при этом в два других фрейма загружались две различные web-страницы. В качестве решения этой задачи Вы можете, например, воспользоваться функцией:

```
function loadtwo() {

  parent.frame1.location.href= "first.htm";
  parent.frame2.location.href= "second.htm";}
```

Если же Вы хотите сделать функцию более гибкой, то можете воспользоваться возможностью передачи переменной в качестве аргумента. Результат будет выглядеть как:



function loadtwo(url1, url2) {

parent.frame1.location.href= url1;
parent.frame2.location.href= url2;}

Послеэтогоможноорганизоватьвызовфункции: loadtwo("first.htm", "second.htm") или loadtwo("third.htm", "forth.htm"). Очевидно, передача аргументов делает Вашу функцию более гибкой. В результате Вы можете использовать ее многократно и в различных контекстах.

Методические рекомендации к выполнению задания 10

1 Создайте документ html.

2 Создайте произвольную html-страничку и сохраните ее в отдельной папке на рабочем столе под названием cell.htm.

3 Создайте второй html-документ. Введите:

<frameset cols="50%,50%">
<frameset rows="50%,50%">
<frame src="cell.html">
<frame src="cell.html">
</frameset>
<frameset rows="33%,33%,33%">
<frame src="cell.html">
<frame src="cell.html">
<frame src="cell.html">
</frameset>
</frameset>

Можно задать толщину границы между фреймами, воспользовавшись в тэге <frameset> параметром *border*. Запись border=0 означает, что между тэгами нет какой-либо границы.

Методические рекомендации к выполнению задания 11

1 В одном фрейме создайте несколько ссылок. Однако, если посетитель активирует какую-либо из них, соответствующая страница будет помещена не в тот же самый фрейм, а в соседний. Сперва необходимо написать скрипт, создающий указанные фреймы:

```
<html>
<frameset rows="80%,20%">
<frame src="start.htm" name="main">
<frame src="menu.htm" name="menu">
</frameset>

</html>
```

Здесь *start.htm* - это та страница, которая первоначально будет показана в главном фрейме (*main*). У нас нет никаких специальных требований к содержимому этой страницы.

2 Сохраните документ под названием Frames.htm.

3 Создайте три документа с произвольным содержанием: first.htm, second.htm, third.htm.

4 Создайте документ ***menu.htm***:

```
<html>

<head>

<script language="JavaScript"> <!-
- hide
```

```
function load(url) {
  parent.main.location.href= url;
}

// -->
</script
>
</head>
<body>

<a href="javascript:load('first.htm')">first</a> <a
href="second.htm" target="main">second</a> <a
href="third.htm" target="_top">third</a> </body>

</html>
```

В данном примере использованы несколько способов загрузки новой страницы во фрейм *main*. В первой ссылке для этой цели используется функция *load()*. Давайте посмотрим, как это делается:

```
<a href="javascript:load('first.htm')">first</a>
```

Вместо явной загрузки новой страницы сдесь предлагается браузеру выполнить некую команду на языке JavaScript - для этого используется параметр *javascript:* вместо обычного *href*. Далее, внутри скобок можно увидеть *'first.htm'*. Эта строка передается в качестве аргумента функции *load()*. Сама же функция *load()* определяется следующим образом:

```
function load(url) {
parent.main.location.href= url;}
```

Здесь внутри скобок написано *url*. Это означает, что строка 'first1.htm' при вызове функции заносится в переменную *url*. И эту новую переменную теперь можно использовать при работе внутри функций *load().*

Во второй ссылке присутствует параметр *target*. Это одна из конструкций языка HTML. Как видно, здесь всего лишь указывается имя необходимого фрейма. Заметим, что в этом случае не обязательно ставить перед именем указанного фрейма слово *parent***.** Причина такого отступления от правил в том, что параметр *target* - это функция языка HTML, а не JavaScript.

На примере третьей ссылки можно увидеть, как с помощью *target* можно избавиться от фреймов. Если нужно избавиться от фреймов с помощью функции *load()*, то необходимо написать в ней лишь parent.location.href= url.

Вопросы для самоконтроля:

1 Что называют фреймом?

2 Какую роль в веб-документе играют фреймы?

3 Какие теги необходимы для создания фреймов?

4 Какие атрибуты и параметры можно задать относительно фреймов?

5 Можно ли присвоить имя фреймам?

6 Каким образом организуется обеспечение доступа с одного фрейма-потомка к другому такому же фрейму-потомку?

7 Для чего предназначена функция loadtwo?

Методические рекомендации к выполнению задания 12

Открытие новых окон в браузере - грандиозная возможность языка JavaScript. Можно либо загружать в новое окно новые документы, либо (динамически) создавать новые материалы. Также можно управлять самим процессом создания окна. Например, можно указать, должно ли новое окно иметь строку статуса, панель инструментов или меню. Кроме того, можно задать размер окна. Список свойств окна, которыми можно управлять:

Directories	Yes / no
Height	количество пикселов
Location	Yes / no
Menubar	Yes / no
Resizable	Yes / no
Scrollbars	Yes / no
Status	Yes / no
Toolbar	Yes / no
Width	количество пикселов

Как видите, открывая окна, мы должны использовать три аргумента:

myWin= open("cell.htm", "displayWindow", "width=400, height=300, status=no, toolbar=no, menubar=no");

Второй аргумент - это имя окна. Если оно известно, то можно загрузить туда новую страницу с помощью записи

<a href="cell.html" target="displayWindow">

При этом необходимо указать имя соответствующего окна (если же такого окна не существует, то с этим именем будет создано новое).

myWin - это не имя окна, но только с помощью этой переменной можно получить доступ к окну. И поскольку это обычная переменная, то область ее действия - лишь тот скрипт, в котором она определена. Имя окна (в данном случае это *displayWindow*) - уникальный идентификатор, которым можно пользоваться с любого из окон браузера.

Чтобы закрыть окно понадобится метод close(). Более того, можно таким же образом создавать и другие документы Web, такие как VRML-сцены и т.д. Для удобства можно размещать эти документы в отдельном окне или фрейме.

Методические рекомендации к выполнению задания 13

1 Создайте новый документ cell.html произвольного содержания.

2 Создайте новый документ html, скрипт показан ниже.

```
<html>
<head>
<script language="JavaScript">

<!-- hide
functionopenWin() {
  myWin= open("cell.htm");
}

// -->
</script
>
</head>
<body>
<form>

<input type="button" value="Открытьновоеокно" onClick= "openWin()" > </form>

</body>
</html>
```

В представленном примере в новое окно с помощью метода open() записывается страница *cell.htm*.

Методические рекомендации к выполнению задания 14

1. Создайте новый документ html

```
<html>

<head>
```

```
<script language="JavaScript"> <!-
- hide

function openWin2() {

  myWin= open("cell.htm", "displayWindow",
"width=400,height=300,status=no,toolbar=no,menubar=no,scrollbar=no");} // -->

</script>
</head>
<body>

<form>

<input type="button" value="Открытьновоеокно" onClick="openWin2()"> </form>

</body>
</html>
```

В этом скрипте открывается новое окно размером 400x300 пикселей.

Оно не имеет ни строки статуса, ни панели инструментов, ни меню. Как видите, свойства окна формулируются в строке

"width=400,height=300,status=no,toolbar=no,menubar=no".

Обратите внимание также и на то, что не следует помещать в этой строке символы пробела.

Методические рекомендации к выполнению задания 15

1. Создайтедокумент:

```
<html>
```

```
<script language="JavaScript"> <!-
- hide

function closeIt() {
  close();}
// -->
</script
>
<center>
<form>

<input type=button value="Close it" onClick="closeIt()">
</form>

</center>

</html>
```

Если теперь в новом окне нажать на кнопку, то окно будет закрыто. open() и close() - это методы объекта window. Следует писать не просто *open()* и *close()*, а *window.open()* и *window.close()*. В нашем случае объект *window* можно опустить - нет необходимости писать префикс window, если нужно вызвать один из методов этого объекта (и такое возможно только для этого объекта).

Методические рекомендации к выполнению задания 16

1. Создайте HTML-документ:

```
<html>
<head>

<script language="JavaScript"> <!-
- hide

function openWin3() {
```

```
  myWin= open("", "displayWindow",
    "width=500,height=400,status=yes,toolbar=yes,menubar=yes");

// открыть объект document для последующей печати
  myWin.document.open();

    // генерировать новый документ
myWin.document.write("<html><head><title>On-the-fly");

  myWin.document.write("</title></head><body>");
  myWin.document.write("<center><font size=+3>");
  myWin.document.write("This HTML-document has been created ");
  myWin.document.write("with the help of JavaScript!");
  myWin.document.write("</font></center>");
  myWin.document.write("</body></html>");

// закрыть документ - (но не окно!)
myWin.document.close(); }

// --> </script>
</head><body><form>

<input type=button value="On-the-fly" onClick="openWin3()"> </form>

</body>
</html>
```

Давайте рассмотрим функцию openWin3(). Очевидно, мы сначала открываем новое окно браузера. Поскольку первый аргумент функции open() - пустая строка (""), то это значит, что мы не желаем в данном случае указывать конкретный адрес URL. Браузер должен создать дополнительно новый документ. В скрипте мы определяем переменную *myWin*. И с ее помощью можем получать доступ к новому окну. Обратите внимание, что в данном случае нельзя воспользоваться для этой цели

именем окна (*displayWindow*). После того, как открыли окно, наступает очередь открыть для записи объект document. Делается это с помощью команды:

```
// открыть объект document для последующей печати
myWin.document.open();
```

Здесь мы обращаемся к open() - методу объекта document. Но это совсем не то же самое, что метод open() объекта window. Эта команда не открывает нового окна - она лишь готовит document к предстоящей печати. Кроме того, мы должны поставить перед *document.open()* приставку *myWin*, чтобы получить возможность писать в новом окне. В последующих строках скрипта с помощью вызова *document.write()* формируется текст нового документа:

```
// генерироватьновыйдокумент
myWin.document.write("<html><head><title>On-the-fly");
myWin.document.write("</title></head><body>");
myWin.document.write("<center><font size=+3>");
myWin.document.write("This HTML-document has been created ");
myWin.document.write("with the help of JavaScript!");
myWin.document.write("</font></center>");
myWin.document.write("</body></html>");
```

Как видно, здесь мы записываем в документ обычные тэги языка HTML. То есть мы фактически генерируем разметку HTML. При этом можно использовать абсолютно любые тэги HTML. По завершении этого нужно вновь закрыть документ. Это делается следующей командой:

```
// закрыть документ - (но не окно!)
myWin.document.close();
```

Вопросы для самоконтроля:

1 Перечислите свойства окна, которыми можно управлять.

2 Какие три аргумента нужно использовать при открытии окна?

3 При помощи какой переменной осуществляется доступ к окну?

4 Какой метод применяется для закрытия окна?

5 При помощи какой команды осуществляется открытие объекта document для последующей печати?

6 При помощи какой команды осуществляется закрытие документа?

7 При помощи какой команды осуществляется закрытие окна?

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.1 Разработка сетевых приложений (клиентская часть)

Лабораторная работа № 2
«Использование библиотеки jQuery»

Цель работы: изучить основы библиотеки jQuery.

Выполнив работу, Вы будете:
уметь:
У10. оформлять код программы в соответствии со стандартом кодирования;
У11. использовать объектные модели веб-приложений и браузера;
У12. использовать открытые библиотеки (framework);
У13. использовать выбранную среду программирования и средства системы управления базами данных;
У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание:

1 Ознакомится с теоретическими сведениями
2 Изучить работу скриптов, приведенных в теоретических сведениях.
3 Реализовать задание согласно варианта.
4 Составить отчет о выполнении работы.

jQuery

jQuery — библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript
4. HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API по работе с Ajax. Т.к. jQuery является более простой для изучения остановимся на ней подробнее.

Возможности:
1. переход по дереву DOM, включая поддержку Xpath как плагина;
2. события;
3. визуальные эффекты;
4. AJAX-дополнения;
5. JavaScript-плагины

Философия

Точно так же, как CSS отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки, управление передаётся JQuery,

идентифицирующей кнопки и затем преобразовывающий его в обработчик события клика. Такое разделение поведения и структуры также называется принципом ненавязчивого JavaScript.

Библиотека jQuery содержит функционал, полезный для максимально широкого круга задач. Тем не менее, разработчиками библиотеки не ставилась задача совмещения в jQuery функций, которые подошли бы всюду, поскольку это привело бы к большому коду, большая́ часть которого не востребована. Поэтому была реализована архитектура компактного универсального ядра библиотеки и плагинов. Это позволяет собрать для ресурса именно тот JavaScript-функционал, который на нём был бы востребован.

Для использования jQuery достаточно скачать с сайта http://jquery.com/ последнюю версию фремворка и подключить его в свой проект.

Шаг 1. Пример подключения jQuery как один внешний файл

```
<head>
<script type="text/javascript" src="путь/к/jQuery.js"></script>
</head>
```

Вся работа с jQuery ведётся с помощью функции $. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним — jQuery. Второй способ считается более правильным, а чтобы код не получался слишком громоздким можно писать его следующим образом:

Шаг 2. Использование функции $

```
jQuery(function($) {
//  Тут код скрипта, где в $ будет jQuery
})
```

Работу с jQuery можно разделить на 2 типа:

4. Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта.
5. Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора CSS, что возвращает объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery. Например,

Шаг 3.

```
$("div.test").add("p.quote").addClass("blue").slideDown("slow");
```

находит все элементы с классом test, а также все элементы <p> с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз. Методы, начинающиеся с $., удобно применять для обработки глобальных объектов. Например:

Шаг 4.

```
$.each([1,2,3], function() {
document.write(this + 1);
});
```

Для примера немного изменим предыдущие файлы и добавим в них использование фреймворка jQuery. Подключаем фреймворк в файл index.html:

Шаг 5. Подключение фрейморка jQuery

```
<script type="text/javascript" src="jquery.js"></script>
```

Добавим в нашу форму дополнительное поле, в котором будем изменять значение параметра учета количества сделанных изменений:

Шаг 6. Добавления дополнительного поля в форму

```
<form>
<p>City: <input type="text" name="city" id="city" size="25"
onChange="getResult();" /></p>
<p>ZipCode: <input type="text" name="zipCode" id="zipCode" size="5"
readonly /> получен с помощью AJAX запроса</p>
<p>Значение было изменено <input type="text" name="jq" id="jq"
size="3" value="0" readonly /> раз с помощью jQuery</p> </form>
```

Дополним функцию updatePage() вызовом дополнительной функции jq(), которая, используя механизмы jQuery, будет изменять данные в поле с id=”jq”, в котором указывается количество обновлений страницы:

Шаг 7. Исходный код функции jq()

```
function jq() {
//Записываем в переменную X значение поля с id='jq'
$x = $("#jq").attr("value");
//Увеличиваем переменную на 1
$x++;
//Обновляем значение атрибута value поля с id='jq'
$("#jq").attr("value", $x);
}
```

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.1 Разработка сетевых приложений (клиентская часть)

Лабораторная работа № 3
«Применение технологии AJAX»

Цель работы: изучить основы технологии AJAX. Написать простейшие клиентское и серверное приложение обменивающиеся ассинхронными данными.

Выполнив работу, Вы будете:

уметь:

У13. использовать выбранную среду программирования и средства системы управления базами данных;

У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание:

5 Ознакомится с теоретическими сведениями

6 Изучить работу скриптов, приведенных в теоретических сведениях.

7 Реализовать задание согласно варианта.

8 Составить отчет о выполнении работы.

9 Проверка логина при регистрации – проверка на существование вводимого логина в базе данных пользователей, при наборе логина в соответствующем поле.

10 Корзина товаров в онлайн магазине. Добавление/удаление.

11 Города и области – выборка области по выбранному городу. Подсчет хеша при заполнении строки.

12 Транслитерация.

13 Простейший почтовой клиент – выборка темы письма по запросу и по клику на теме письма доставка его тела.

14 Мониторинг сервисов на ПК. Опрос списка запущенных процессов, и асинхронное обновление.

15 Гостевая книга, Комментарии – добавление/удаление.

16 Чат.

17 Доработка заданий 3 и 4 лабораторной работы.

Краткие теоретические сведения

AJAX — это модное название для набора техник разработки веб-интерфейсов, позволяющих делать динамические запросы к серверу без видимой перезагрузки веб-страницы: пользователь не замечает, когда его браузер запрашивает данные.

AJAX обеспечивает динамичность и асинхронность web-разработок при отсутствии необходимости обновления страниц.

Пожалуй, любой разработчик мечтает о том, чтобы превратить обычную, неновую web-страничку во что-то более захватывающее. Сейчас можно попробовать вдохнуть немного жизни в web-технологии десятилетней давности, для этого познакомся с AJAX.

При использовании Google или web-клиента Gmail вам уже приходилось сталкиваться с решением, основанном на AJAX. Это технология, которая обеспечивает динамическое и асинхронное поведение, где исключается обновление страниц. Посредством AJAX пользователь может взаимодействовать с web-страницами, подобно работе клиентов с более богатыми возможностями.

В настоящее время многие говорят о AJAX. Технология, следующая за AJAX, вряд ли является новшеством, однако недавно стали появляться некоторые мощные новые приложения, использующие объект XMLHttpRequest, они вдохнули новую жизнь в концепцию обновления образа клиентской части.

Самым примечательным из этих новых приложений является Google Maps. Пользуясь им, можно находить определенную местность на карте планеты, затем переходить к более мелким объектам, прокручивать перетягивать карту без необходимости обновления страницы.

AJAX — это аббревиатура от Asynchronous JavaScript and XML (и DHTML, и т.д.). (как это представил Джис Джеймс Гаррет (Jesse James Garrett), он первым ввел термин 'AJAX' для асинхронного JavaScript + XML). Т.е. это коллекция технологий, существующих с момента появления Web.

Возможности, предоставляемые AJAX:

Стандартно-базированная презентация с использованием XHTML и CSS;

Динамическое отображение и взаимодействие с использованием объектной модели документа;

Взаимообмен данными и манипуляция с задействованием XML и XSLT;

Асинхронное извлечение данных с использованием XMLHttpRequest;

JavaScript, связывающий все вместе.

Вкратце AJAX позволяет писать быстрореагирующие веб-приложения, в которых не нужно постоянно обновлять страницы. AJAX — простая технология, поддерживаемая всеми основными браузерами. Как можно вкратце отметить, единственным предварительным условием для внедрения AJAX является знание JavaScript.

Как работает AJAX

Если вы когда-либо пользовались веб-клиентом Gmail или Google Maps, то замечали возможность проверки правописания и прокрутки по всему изображению, соответственно, без обновления страниц. AJAX — это технология, которая обрабатывает операции в JavaScript и асинхронно запускает на стороне сервера операции, предоставляющие желаемый результат.

5. основе технологии AJAX лежит объект XMLHttpRequest. Изначально он появился в Internet Explorer, а затем — в Mozilla/Safari. На момент написания этой статьи уже была поставлена 8-я совместимая версия Opera. Однако, Opera в свое время отличилась нестабильностью с точки зрения реализации XMLHttpRequest.

Объект XMLHttpRequest

Первый объект, о котором вы хотите узнать, возможно, самый новый для вас; он называется XMLHttpRequest. Это объект JavaScript, и он создается так же просто, как показано в листинге 1.

Порядок выполнения работы

Шаг 1. Создание нового объекта XMLHttpRequest

<script language="javascript" type="text/javascript">
var request = new XMLHttpRequest(); </script>

Это объект, который управляет всем вашим взаимодействием с сервером – это технология JavaScript в объекте XMLHttpRequest, который общается с сервером. Это не обычный ход работы приложения, и именно здесь заключается почти вся магия Ajax.

А нормальных Web-приложениях пользователи заполняют поля форм и нажимают кнопку *Submit* (подтвердить). Затем форма передается на сервер полностью, сервер обрабатывает сценарий (обычно PHP или Java, возможно, CGI-процесс или что-то в этом роде), а потом передает назад всю новую страницу. Эта страница может быть HTML-страницей с новой формой с некоторыми заполненными данными, либо страницей подтверждения, либо, возможно, страницей с какими-то выбранными вариантами, зависящими от введенных в оригинальную форму данных. Естественно, пока сценарий или программа на сервере не обработается и не возвратится новая форма, пользователи должны ждать. Их экраны очистятся и будут перерисовываться по мере поступления новых данных от сервера. Вот где проявляется низкая интерактивность – пользователи не получают немедленной обратной реакции и определенно чувствуют себя не так, как при работе с настольными приложениями.

Ajax по существу помещает технологию JavaScript и объект XMLHttpRequest *между* вашей Web-формой и сервером. Когда пользователи заполняют формы, данные передаются в какой-то JavaScript-код, а не прямо на сервер. Вместо этого JavaScript-код собирает данные формы и передает запрос на сервер. Пока это происходит, форма на экране пользователя не мелькает, не мигает, не исчезает и не блокируется. Другими словами, код JavaScript передает запрос в фоновом режиме; пользователь даже не замечает, что происходит запрос на сервер. Более того, запрос передается асинхронно,

//это означает, что ваш JavaScript-код (и пользователь) не ожидают ответа сервера. То есть, пользователи могут продолжать вводить данные, прокручивать страницу и работать с приложением.

Затем сервер передает данные обратно в ваш JavaScript-код (все еще находящийся в вашей Web-форме), который решает, что делать с данными. Он может обновить поля формы "на лету", придавая свойство немедленности вашему приложению – пользователи получают новые данные без подтверждения или обновления их форм. JavaScript-код может даже получить данные, выполнить какие-либо вычисления и передать еще один запрос, и все это без вмешательства пользователя! В этом заключается мощь XMLHttpRequest. Он может общаться с сервером по своему желанию, а пользователь даже не догадывается о том, что происходит на самом деле. В результате мы получаем

динамичность, чувствительность, высокую интерактивность настольного приложения вместе со всеми возможностями интернет.

Добавление JavaScript-кода

После того, как вы разберетесь с XMLHttpRequest, оставшийся JavaScript-код превращается в рутинную работу. Фактически, вы будете использовать JavaScript-код для небольшого числа основных задач:

Получить данные формы: JavaScript-код упрощает извлечение данных из вашей HTML-формы и передает их на сервер.

Изменить значения в форме: Форма обновляется тоже легко, от установки значений полей до замены изображений "на лету".

Выполнить анализ HTML и XML: Вы будете использовать JavaScript-код для управления DOM и для работы со структурой вашей HTML-формы и всеми XML-данными, возвращаемыми сервером.

Для выполнения первых двух задач вы должны очень хорошо знать метод getElementById(), приведенный в листинге 2.

Шаг 2. Сбор и установка значений полей при помощи JavaScript-кода

4. Получить значение поля "phone" и записать его в переменную phone

```
var phone = document.getElementById("phone").value;
```

5. Установить значения в форме, используя массив response

```
document.getElementById("order").value = response[0];
document.getElementById("address").value = response[1];
```

Вы должны начать понимать, что нет ничего чрезмерно сложного во всем этом. Как только вы освоите XMLHttpRequest, оставшаяся часть вашего Ajax-приложения будет простым JavaScript-кодом, похожим на приведенный в листинге 2, смешанным с немного более умным HTML. К тому же, время от времени есть немного работы с DOM. Итак, давайте рассмотрим это.**Получение объекта Request**

Поскольку XMLHttpRequest является центральным для Ajax-приложений (и, возможно, нов для многих из вас) начнем с него. Как вы видели в листинге 1, создать этот объект и использовать его должно быть просто, не правда ли? Подождите минуточку.

Помните те ужасные войны браузеров, происходившие несколько лет назад, и как ничто не работало одинаково в разных браузерах? Поверите вы или нет, но те войны продолжаются до сих пор, хотя и с намного меньшим масштабом. И, сюрприз: XMLHttpRequest – это одна из жертв этих войн. Поэтому вы должны выполнить несколько различных действий для обеспечения возможности работы XMLHttpRequest.

Работа с браузерами Microsoft

Браузер Microsoft Internet Explorer для обработки XML использует анализатор MSXML. Поэтому, когда вы пишете Ajax-приложения, которые должны работать в Internet Explorer, необходимо создать объект особым способом.

Однако, это не так то и легко. На самом деле в ходу две различных версии MSXML. Версия MSXML зависит от версии технологии JavaScript, установленной в Internet Explorer, поэтому вам нужно написать код, подходящий для обеих версий. Взгляните на листинг 3, в котором приведен код для создания XMLHttpRequest в браузерах Microsoft.

Шаг 3. Создание объекта XMLHttpRequest в браузерах Microsoft

```
var request = null;

function createRequest () {
try {
request = new ActiveXOblect("Msxml2.request");
} catch (othermicrosoft)
{ try {
request = new ActiveXObject("Microsoft.request"); }
catch (failed) {

request = null;
}
}
}
```

Все это пока может не иметь смысла, но это нормально. Пока вы должны записать в своей голове две основных строки:

request = new ActiveXObject("Msxml2.request");

и

request = new ActiveXObject("Microsoft.request");.двух словах, этот код пытается создать объект, используя одну версию MSXML; если это не получится, создается объект для второй версии. Изящно, да? Если ничего не сработало, переменная request устанавливается в false, для того чтобы указать вашему коду, что что-то не так. В этом случае вы, возможно, работаете с браузером не от Microsoft и должны использовать другой код для выполнения работы.

Работа с Mozilla и браузерами не от Microsoft

Если Internet Explorer не ваш браузер, либо вы пишете код для браузеров не от Microsoft, вам нужен другой код. Фактически, это простая строка, которую вы видели в листинге 1:

var request = new XMLHttpRequest object;.

Эта намного более простая строка создает объект XMLHttpRequest в Mozilla, Firefox, Safari, Opera и в большой степени в каждом браузере не от Microsoft, поддерживающем Ajax в любой форме или разновидности.

Объединение

Мы хотим поддерживать *все* браузеры. Кто хочет писать приложение, работающее только в Internet Explorer, или приложение, работающее только во всех остальных браузерах? Еще хуже, хотите ли вы написать ваше приложение дважды? Конечно, нет! Итак, объединим поддержку для Internet Explorer и для остальных браузеров. В листинге 4 приведен код, делающий это.

Шаг 4. Создание объекта XMLHttpRequest для всех браузеров

```
var request = null;

function createRequest () {

try {
request = new XMLHttpRequest();
}                    catch
(trymicrosoft) { try {
request = new ActiveXOblect("Msxml2.request"); }
catch (othermicrosoft) {

try {
request = new ActiveXObject("Microsoft.request");
} catch (failed)
{ request = null;

}
}
}

if (request == null) alert("Ошибка создания XMLHttpRequest объекта!"); }
```

Основу этого кода можно разделить на три шага:

Создаем переменную request для ссылки на объект XMLHttpRequest, который вы создадите.

Создаем объект для неMicrosoft браузеров, если это не получается, то переходим к следующему пункту.

В блоке try создайте объект в браузерах Microsoft:

В блоке try создайте объект с использованием объекта Msxml2.request.

Если это не получится, В блоке try создайте объект с использованием объекта Microsoft.request.

// конце этого процесса request должен ссылаться на корректный объект XMLHttpRequest, независимо от используемого пользователем браузера.

Запрос/ответ в мире Ajax

Итак, вы уже знакомы с Ajax и имеете базовое представление об объекте XMLHttpRequest и о том, как создать его. Если вы читали внимательно, то вы даже понимаете, что это технология JavaScript общается с любым Web-приложением на сервере, а не ваша HTML-форма, которую вы подтвердили напрямую.

Что мы пропустили? Как на самом деле использовать XMLHttpRequest. Поскольку это критический код, который вы будете использовать в некоторых формах

4*каждом* вашем Ajax-приложении, рассмотрим коротко, как выглядит базовая модель запрос/ответ в Ajax.

Выполнение запроса

У вас есть ваш новый объект XMLHttpRequest; приведем его в движение. Во-первых, нам понадобится JavaScript-метод, который ваша Web-страница может вызвать (например, когда пользователь вводит текст или выбирает вариант из меню). Затем, нужно следовать одной и той же основной схеме практически во всех ваших Ajax-приложениях:

Получить какие-либо данные из Web-формы.
Создать URL для подключения.
Открыть соединение с сервером.
Установить функцию для сервера, которая выполнится после его ответа.

Передать запрос.

В листинге 5 приведен пример Ajax-метода, который выполняет именно эти операции и именно в этом порядке:

Шаг 5. Выполнить запрос с Ajax

```
function getResult() {

//Создать URL для подключения
var url = "test.php";
//Создаем объект XMLHttpRequest
createRequest();

//Открыть соединение с сервером
request.open("GET", url, true);

//Установить функцию для сервера, которая будет выполнена после его ответа
request.onreadystatechange = updatePage;

//Отправить запрос
request.send(null);
}
```

Многое из этого не требует пояснений. Код устанавливает PHP-сценарий в качестве URL для подключения. Затем открывается соединение; это первое место, где вы опять увидели в действии XMLHttpRequest. Указывается метод соединения (GET)

вURL. Последний параметр, когда установлен в true, запрашивает асинхронное соединение (то есть, делает это способом, сооветствующим названию Ajax). При использовании false код ждал бы выполнения запроса и не продолжал бы работу до

получения ответа. При использовании true ваши пользователи могут работать с формой (и даже вызывать другие JavaScript-методы) пока сервер обрабатывает этот запрос в фоновом режиме.

Свойство onreadystatechange request (вспоминайте, это ваш экземпляр объекта XMLHttpRequest) позволяет вам информировать сервер о том, что следует делать после завершения работы (что может быть через пять минут или через пять часов). Поскольку код не собирается ждать сервер, вы должны дать серверу знать, что делать, так чтобы вы смогли среагировать. В данном случае будет инициирован конкретный метод (называемый updatePage()) после завершения сервером обработки вашего запроса.

Наконец, вызывается send() со значением null. Поскольку вы добавили данные для передачи на сервер в URL запроса, вам не надо передавать что-либо в запросе. Таким образом, передается ваш запрос, и сервер может делать то, что вы указали ему делать.

Если вы кроме этого ничего больше не делаете, обратите внимание на то,

насколько все просто и понятно! В отличие от осознания вами асинхронной природы Ajax, все это действительно простые вещи. Вы оцените то, как это освобождает вас для концентрации внимания на крутых приложениях и интерфейсах, а не на сложном HTTP-коде запроса/ответа.

Код в листинге 5 очень прост. Данные являются простым текстом и могут быть включены как часть URL-запроса. GET посылает запрос вместо более сложного POST. Не добавляется XML, заголовки контента, не передаются данные в теле запроса.

На странице test.php, которую мы указываем в запросе производится либо

обработка принятой информации, если таковая передавалась в запросе, например, как test.php?city=chernihiv, либо просто возвращается результат.

Шаг 5.1 пример test.php

```
<?php

//Sending zipCode
echo rand(10000, 99999);

?>
```

Обработка ответа

Теперь вы должны разобраться с ответом сервера. Пока вы должны знать только два момента:

- Не делать ничего, пока свойство request.readyState не будет равно 4.

- Сервер будет записывать свой ответ в свойстве request.responseText.

Первый момент (состояния готовности) – о нем вы должны сами разобраться,

т.е. уточнить какие бывают стадии HTTP-запроса. Пока вы просто проверяйте на равенство определенному значению (4), и все будет работать. Второй момент (использование свойства request.responseText для получения ответа от сервера) является простым. В листинге 6 приведен пример метода (который сервер может вызвать), основанного на значениях, переданных в листинге 5.

Шаг 6. Обработка ответа от сервера

```
function updatePage() {
if(request.readyState == 4) {
var response = request.responseText;
//Что-то делаем с пришедшими нам данными, например, присваиваем
их полю с id="zipCode"
document.getElementById("zipCode").value = response;
}
}
```

Опять же, код не является трудным или сложным. Он ожидает, пока сервер не вызовет его с нужным состоянием готовности, и затем использует значение, которое сервер возвращает, для установки другого поля формы. В результате поле zipCode неожиданно появляется с ZIP-кодом, но пользователь *ни разу не щелкнул по кнопке*! Это поведение настольного приложения, о чем мы говорили ранее.

Вы, возможно, заметили, что поле zipCode является обычным текстовым полем. После возврата сервером ZIP-кода и установки этого поля методом updatePage() в значение ZIP-кода города/штата пользователи *могут* переопределить это значение. Так сделано умышленно по двум причинам: сохранить этот пример простым и показать вам, что иногда нужно, чтобы пользователи имели возможностьпереопределить значения, возвращенные сервером. Помните об обоих моментах; они важны при хорошем дизайне пользовательского интерфейса.

Перехват в Web-формах

Что нам осталось? В сущности, не много. Вы имеете JavaScript-метод, собирающий введенную пользователем в форму информацию, передаете ее серверу, предоставляете еще один JavaScript-метод для обработки ответа и даже устанавливаете значение поля, когда этот ответ приходит. Все что осталось на самом деле – *вызвать* этот первый метод и запустить полный процесс. Вы могли бы, очевидно, добавить кнопку в вашу HTML-форму, но это же старый, добрый 2001 год, не так ли? Воспользуемся возможностями технологии JavaScript, как показано в листинге 7.

Шаг 7. Запуск Ajax-процесса

```
<form>
<p>City: <input type="text" name="city" id="city" size="25"
onChange="getResult();" /></p>
<p>Zip Code: <input type="text" name="zipCode" id="zipCode" size="5"
readonly /></p>
</form>
```

Для более легкой работы с HTML при помощи JavaScript были разработаны несколько фреймворков, которые упрощают повседневные рутинные функции и помогают программисту не задумываться о реализации примитивов, например, доступа к полям DOM-а, их изменение стандартными средствами JavaScript, а предоставляют удобные инструменты для данных функций. Из наиболее известных на сегодняшний день стоит отметить два фреймворка: Ext JS и jQuery.

ExtJS

Ext JS — библиотека JavaScript для разработки веб-приложений и пользовательских интерфейсов, изначально задуманная как расширенная версия Yahoo! UI Library, однако преобразовавшаяся затем в отдельный фреймворк. Использует адаптеры для доступа к библиотекам Yahoo! UI Library, jQuery или Prototype/script.aculo.us. Поддерживает технологию AJAX, анимацию, работу с DOM, реализацию таблиц, вкладок, обработку событий и все остальные новшества «Web 2.0».

3. версии 2.1 библиотека ExtJS распространяется по условиям трёх лицензий: Commercial License, Open Source License и OEM / Reseller License.

Начиная с версии Ext JS 3.0 библиотека разбивается на две части: Ext Core (набор JavaScript функций, позволяющий создавать динамические веб-страницы и рапространяемый по MIT-лицензии) и Ext JS (набор виджетов для создания пользовательских интерфейсов, распространяемый аналогично Ext JS 2.1 по условиям трёх лицензий).

Форма представления результата:

Предоставить скрипт заданий

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.2 Разработка сетевых приложений (серверная часть)

Лабораторная работа № 4
«Создание серверных сценариев с использованием технологии PHP»

Цель:

Освоение базовых конструкций языка PHP:

- организация циклов;
- операторы ветвления.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;
У3. разрабатывать и проектировать информационные системы;
У9. использовать язык разметки страниц веб-приложения;
У10. оформлять код программы в соответствии со стандартом кодирования;
У13. использовать выбранную среду программирования и средства системы управления базами данных;
У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений;

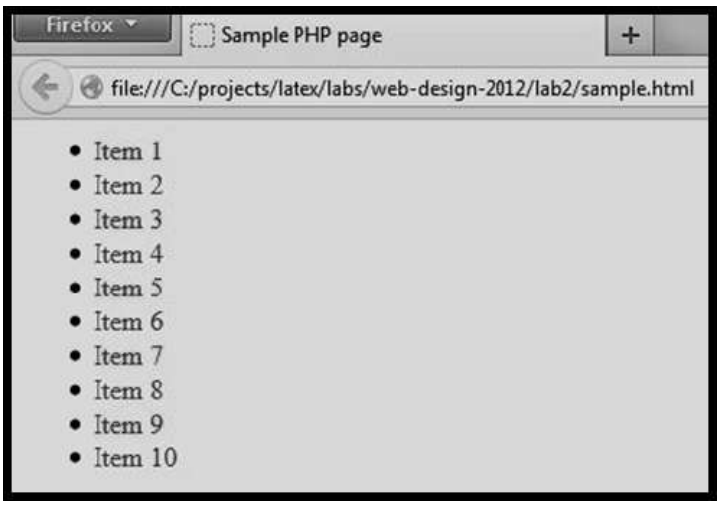
Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: изучить основы языка PHP.

Порядок выполнения работы

Предположим, нам необходимо сделать страницу, показанную на рисунке ниже. У всех элементов списка есть закономерности. Во-первых, номер элемента указывается в тексте элемента. Во-вторых, четные элементы списка выводятся зеленым, а нечетные красным.



Можно написать такую страницу вручную, а можно прибегнуть к помощи интерпретатора PHP. Одно из главных преимуществ, которое дает этот интерпретатор, – это возможность писать страницу напрямую в формате HTML (эти фрагменты интерпретатор передает серверу в качестве ответа сразу же без обработки), вынеся в PHP только те фрагменты, которые генерируются динамически:

```
<html>
   <head><title>Sample PHP page</title></head>
   <body>
      <ul>
         <<PHP fragment goes here>>
      </ul>
   </body>
</html>
```

Поскольку большая часть страницы остается неизменной и всегда должна выдаваться в одном и том же формате, она записана напрямую в HTML. В PHP записано только то, что нужно сгенерировать программно.

Фрагмент PHP – вариант первый

Рассмотрим простейшую реализацию этого списка из 10 элементов с помощью цикла с условиями:

```
<?php
for ($i = 1; $i <= 10; $i++) {
   // Вывод тэга элемента
   echo "<li>";
   // Вывод тэга шрифта в зависимости от четности
   if ($i%2 == 0) echo "<font color = \"green\">";
   else echo "<font color = \"red\">";
   // Вывод содержимого элемента
   echo "Item ".$i;
   // Вывод закрывающего элемента шрифта
   echo "</font>";
   // Вывод закрывающего элемента списка с переводом на новую строку
   echo "</li>\n";
}
?>
```

Фрагмент PHP – вариант второй

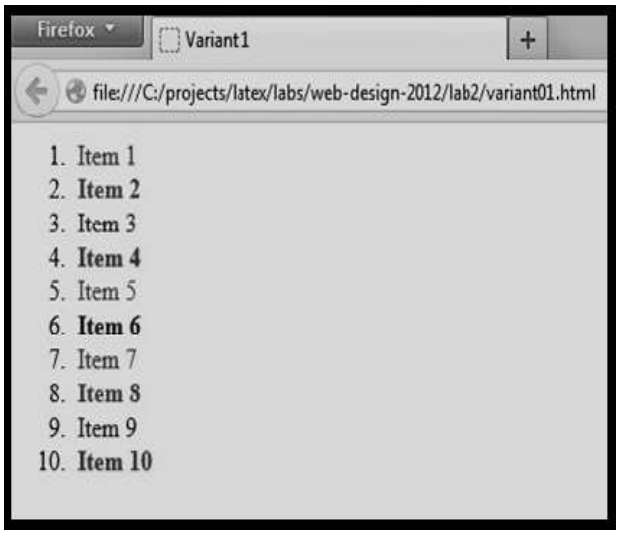
Предыдущий вариант реализует то, что нужно, однако он выглядит не очень понятно. PHP позволяет работать с выводом строк гораздо более изящно, указывая переменные напрямую в строке без всякой конкатенации:

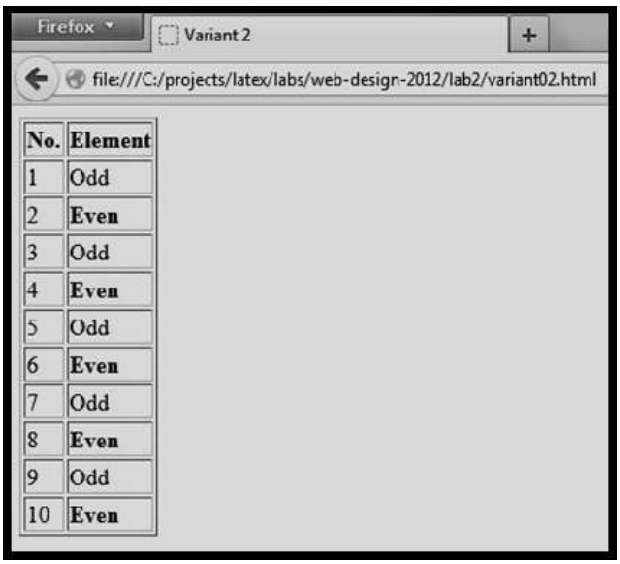
```
<?php
for ($i = 1; $i <= 10; $i++) {
   // Установка переменной цвета
   if ($i%2 == 0) $color = "green";
   else $color = "red";
   // Вывод содержимого в совокупности
   echo "<li><font color = \"$color\">Item $i</font></li>\n";}
?>
```

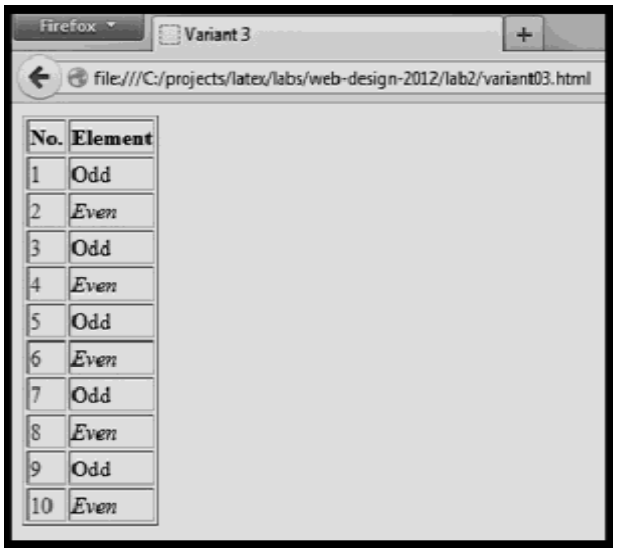
Варианты заданий

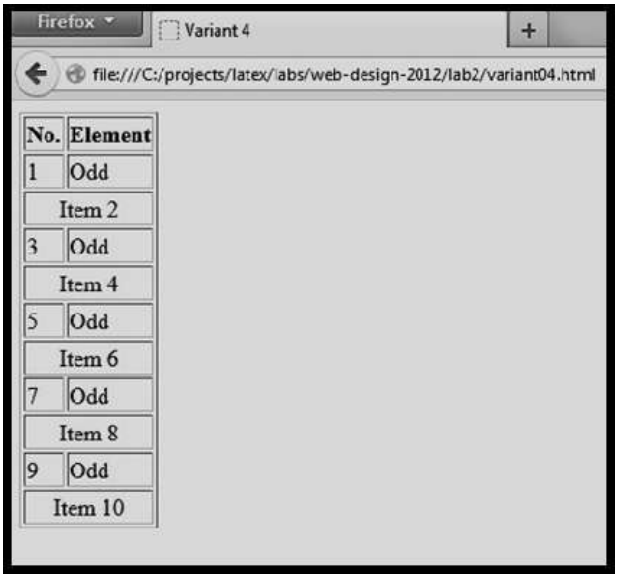
Имеется 10 вариантов заданий, представленных ниже. Необходимо проанализировать каждую страницу, выявить, какие тэги требуются для ее реализации, и написать исходный код этой страницы, после чего убедиться, что браузер отображает эту страницу именно так, как она выглядела

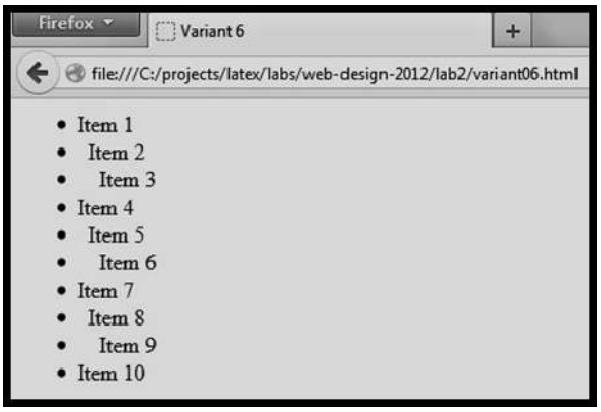
изначально. Для каждого варианта следует проставить заголовок страницы, который отображает браузер на рисунке.

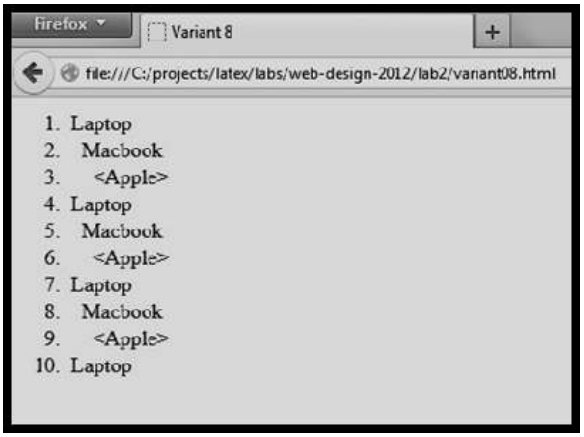


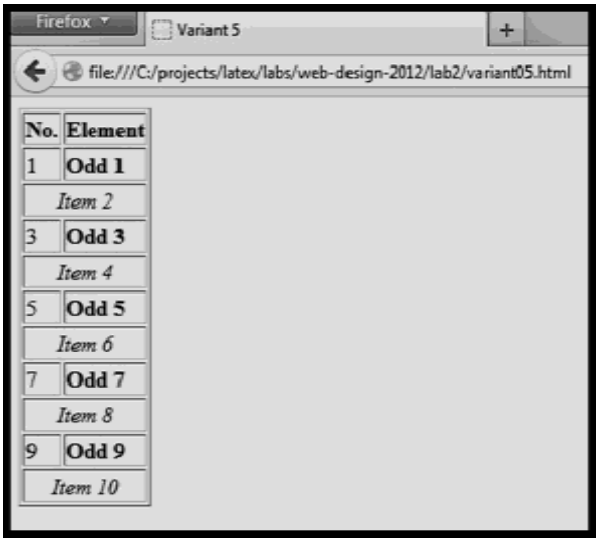


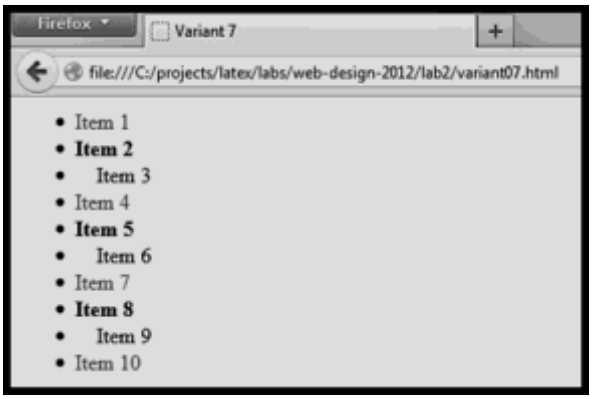


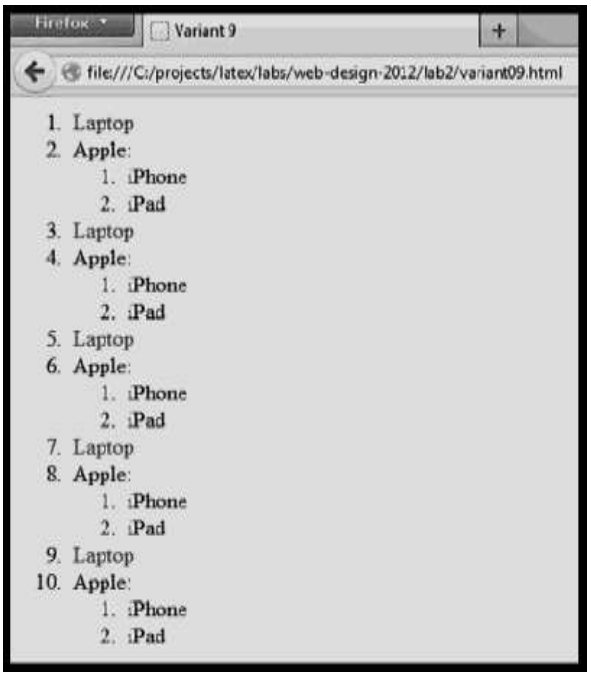

Firefox
Variant 4
file:///C:/projects/latex/labs/web-design-2012/lab2/variant04.html
No. Element
1 Odd
Item 2
3 Odd
Item 4
5 Odd
Item 6
7 Odd
Item 8
9 Odd
Item 10


Firefox
Variant 6
file:///C:/projects/latex/labs/web-design-2012/lab2/variant06.html
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Item 10


Firefox
Variant 8
file:///C:/projects/latex/labs/web-design-2012/lab2/variant08.html
1. Laptop
2. Macbook
3. <Apple>
4. Laptop
5. Macbook
6. <Apple>
7. Laptop
8. Macbook
9. <Apple>
10. Laptop


Firefox
Variant 5
file:///C:/projects/latex/labs/web-design-2012/lab2/variant05.html
No. Element
1 Odd 1
Item 2
3 Odd 3
Item 4
5 Odd 5
Item 6
7 Odd 7
Item 8
9 Odd 9
Item 10


Firefox
Variant 7
file:///C:/projects/latex/labs/web-design-2012/lab2/variant07.html
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9
Item 10


Firefox
Variant 9
file:///C:/projects/latex/labs/web-design-2012/lab2/variant09.html
1. Laptop
2. Apple:
1. iPhone
2. iPad
3. Laptop
4. Apple:
1. iPhone
2. iPad
5. Laptop
6. Apple:
1. iPhone
2. iPad
7. Laptop
8. Apple:
1. iPhone
2. iPad
9. Laptop
10. Apple:
1. iPhone
2. iPad

Форма представления результата:

1. Цель работы.
2. Вариант задания. Изображение требуемой страницы.
3. Исходный текст страницы.
4. Изображение полученной страницы.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.2 Разработка сетевых приложений (серверная часть)

Лабораторная работа № 5
«Обработка данных на форме»

Цель: Знакомство с GET- и POST-запросами от клиента к серверу и создание простейших сценариев взаимодействия

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: Выполнить обработку данных на форме.

Краткие теоретические сведения:

Сайты работают по схеме:

- существуют страницы *просмотра*, на которых пользователь может просмотреть какие-либо сведения, которые извлека- ются из базы и выдаются сервером в удобном для пользова- теля виде;
- существуют страницы *добавления/редактирования* инфор- мации, на которых пользователь может добавить новые дан- ные либо модифицировать уже имеющиеся.

Очевидно, что оба типа страниц работают с содержимым базы, первый тип осуществляет запросы на выборку данных из базы. Вто- рой – осуществляет запросы на добавление/редактирование/удаление записей в базе. Кроме того, страницы могут быть смешанного типа, т. е. отображать какие-то сведения и предоставлять функционал по добавле- нию новых сведений. Вот наиболее распространенные примеры таких страниц:

- форум. Пользователь после авторизации может зайти в любую ветвь, просмотреть ее содержимое и добавить новое сообщение в конце ветви;

- блог. Пользователь может добавлять новые статьи в соб- ственный блог и просматривать остальные блоги.

В данной лабораторной работе рассматривается инструментарий для взаимодействия клиента с сервером, даются ответы на вопросы:

- как разместить в HTML-странице элементы, которые позво- лили бы пользователю вводить данные и отправлять их серверу?
- Как сервер обрабатывает данные от пользователя?

Существуют два основных способа передачи данных от клиента к серверу – это GET- и POST-запросы (есть и другие варианты, но они здесь не рассматриваются).

GET-запросы

В предыдущих лабораторных работах упоминалось, что браузер клиента общается с сервером, формируя пакет с запросом, который отправляется серверу. Сервер, получив такой пакет, отправляет назад пакет с ответом. В пакете запроса содержится многое, в первую очередь адрес страницы на сервере, к которой нужно обратиться. Помимо этого пакет может содержать дополнительные данные. Рассмотрим это на примере простой ссылки:

```
<a href = "remove.php?article_id=96&confirmed=1">Удалить статью</a>
```

Что происходит, когда пользователь нажимает на эту ссылку? Браузер отправляет содержимое ссылки (значение атрибута **href**) в виде GET-запроса на сервер. В качестве адреса сервер использует то, что идет в ссылке до знака вопроса, т. е. адрес **remove.php** (учтите, что этот адрес вычисляется относительно местонахождения страницы, на которой расположена ссылка). То что идет после знака вопроса – это параметры GET-запроса, которые представляются в формате **ключ=значение.** Такие пары разделяются знаком амперсенда. То есть в данном случае в пакете будет передано помимо самого адреса еще два параметра – **article_id** (со значением 96) и **confirmed** (со значением 1).

Следует учитывать, что эти параметры передаются в виде строковых значений. Apache, получая такой пакет, вызывает интерпретатор PHP для

страницы **remove.php** и передает ему параметры GET- запроса. Интерпретатор может обратиться к ним через глобальную переменную-массив

```
<?php
$article = $_GET['article_id'];
$confirmed = $_GET['confirmed'];
// Произвести далее операцию удаления
?>
```

Недостатки такого способа передачи следующие:

- дополнительные параметры указываются явно в ссылке, что зачастую неправильно (приходится отображать пользовате- лю служебные данные);
- проблемы с кодировкой – символы, которые не являются символами английского алфавита, приходится кодировать с помощью шестнадцатеричной нотации, которой предшеству- ет знак процента. Например, знак пробела (код 32) кодирует- ся как "%20". Это приводит к тому, что адресная строка мо- жет быть очень большой. Кроме того, часть символов (кото- рая не может быть закодирована в ISO Latin 1) нельзя передать таким запросом;
- невозможность передавать в адресной строке содержимое файлов;
- многие серверы содержат ограничения по длине принимае- мой адресной строки и слишком длинную адресную строку обрабатывать просто не будут.

POST-запросы

Для решения подобного рода проблем были созданы POST-зап- росы. В них содержимое параметров хранится в пакете запроса, но не в адресной строке, а в отдельном поле сообщения, которое может быть потенциально очень большим, и оно не отображается при запро- се в адресной строке. Для того чтобы отправить такое содержимое, нужно разместить необходимые элементы (с помощью которых поль- зователь вводит данные) в *форму*. Форма задается с помощью следующих тэгов:

```
<form method = "POST" action = "admin/test.php">
<!-- Содержимое формы -->
</form>
```

Теперь при отправке данных пользователем через эту форму будет создан POST-запрос с адресом **admin/test.php**, где в теле сообщения (но не в адресной строке) будет передано содержимое формы. А содержимое может включать в себя следующие поля (рассмотрены наиболее распространенные).

Флаги

Чаще всего на формах в виде опций предоставляются флаги – галочки, которые пользователь может пометить. Чтобы ее создать, нужно использовать следующий тэг:

```
<input name = "[Имя флага]" type = "checkbox" value = "[Значение флага]"/>
```

Если пользователь выбрал флаг, то при отправке формы в тело сообщения пакета будет добавлена пара **Имя флага=Значение флага**. Если пользователь не выбрал флаг, то параметр с его именем просто не будет добавлен. По умолчанию флаг отображается как не выбранный пользователем.

Переключатели

Переключатели **radio** предлагают пользователю ряд вариантов, причем выбрать нужно один из них:

```
<input name = "[Имя переключателя]" type = "radio" value = "[Значение]"/>
```

При выборе пользователем одного из *группы* переключателей (группа – это все переключатели с одинаковым именем) и отправке данных через форму в тело сообщения пакета будет добавлена пара **Имя переключателя=Значение выбранного переключателя.**

Кнопка сброса формы

При нажатии на кнопку сброса все элементы формы будут установлены в то состояние, которое было задано в атрибутах по умолчанию, причем отправка формы не производится

```
<input type = "reset"  value = "Очистить форму"/>
```

Выпадающий список

Тэг **select** позволяет отображать на форме выпадающие списки, из которых пользователь может выбрать одно или несколько значений

```
<select name = "[Имя списка]">
<option value = "[Значение первого элемента]">Первый элемент</option>
<option value = "[Значение второго элемента]">Второй элемент</option>
</select>
```

При отправке пользователем данных через форму в случае, когда в списке что-то выбрано, в тело сообщения будет добавлена пара **Имя списка=Выбранное значение**. Естественно, содержимое списка можно генерировать динамически с помощью PHP, как и весь остальной код HTML.

Текстовое поле

Данные поля позволяют пользователю вводить различную текстовую информацию:

При вводе пользователем значения в тело сообщения будет добавлена пара **Имя поля=Введенный текст**. При этом производится шифрование в многобайтовой кодировке, поэтому таким способом можно передать любые символы.

Поле ввода пароля

Практически идентично текстовому полю, за тем исключением, что вводимый пользователем текст не отображается в поле (точнее, отображается в виде зведочек)

```
<input type = "password" name = "[Имя поля]"/>
```

Скрытое текстовое поле

Помимо всего прочего в форме могут располагаться скрытые поля, которые не отображаются пользователю, но при этом все равно передаются в случае отправки данных формы

```
<input type = "hidden" name = "[Название поля]" value = "[Значение поля]"/>
```

Кнопка отправки формы

Отправить данные в форме можно посредством специального элемента – кнопки отправки:

```
<input type = "submit" name = "[Название кнопки]" value = "[Текст кнопки]"/>
```

При нажатии на эту кнопку все данные формы кодируются в виде пакета и отправляются серверу. Если у кнопки есть имя, то в тело сообщения будет добавлена пара **Название кнопки=Текст кнопки**. Это необходимо, когда в форме несколько кнопок отправки, чтобы сервер смог определить, что именно ему отправляют.

Примеры обработки POST-запросов

Пример 1. Первая страница содержит в себе два текстовых поля, которые будет отображать вторая страница:

```
<form action = "action.php" name = "myform" method = "post">
  <input type = "text" name = "mytext"/><br/>
  <textarea name = "msg" cols = "20" rows = "10"/>
  <input type = "submit" value = "Отправить данные"/>
</form>
```

Вот как выглядит код страницы **action.php**:

```
<?php
$text = $_POST['mytext'];
$msg = $_POST['msg'];
echo "$text<br/>$msg";
?>
```

Пример 2. На первой странице генерируется динамически список выбора по годам:

```
<form action="action.php" name = "myform" method = "post">
  <select name = "year">
<?php
for ($i = 2000; $i <= 2015; $i++)
  echo "<option value = $i>$i</option>\n";
?>
  </select>
  <input type = "submit" value = "Отправить данные"/>
</form>
```

Вот как выглядит код страницы **action.php**:

```
<?php
$year = $_POST['year'];
echo "Selected value is: $year";
?>
```

Порядок выполнения работы:

Задания для данной лабораторной работы являются логическим развитием предыдущей лабораторной. Страницы добавления/редактирования сущностей должны позволять редактировать все поля записи (за исключением первичного ключа). Ссылки (внешние ключи) представляются с помощью выпадающих списков, все остальное – с помощью текстовых полей. Страница редактирования должна хранить в себе первичный ключ сущности (в виде скрытого поля). Страницы добавления/редактирования должны производить валидацию и в случае ошибки оставлять пользователя на той же странице. Каждый таб личный список содержит в качестве первого столбца "№ п/п".

Варианты заданий следующие.

1. Страница просмотра списка фирм отображает в виде табли- цы со столбцами "Название", "Просмотр". В первом – название фир- мы. Во втором – ссылка, которая ведет на страницу просмотра сведе- ний о фирме (описана детально ниже). Внизу страницы просмотра списка фирм имеется кнопка "Добавить", при нажатии на которую пользователь переходит на страницу добавления сведений о фирмах.

Страница просмотра отдельной фирмы отображает следующие сведения: название фирмы, маркированный список автомобилей, где каждый элемент списка – это ссылка с названием автомобиля. Нажав на эту ссылку, пользователь попадает на страницу просмотра сведений об автомобиле. Внизу страницы есть кнопки "Редактировать" и "Удалить" (эта кнопка отключена, если у фирмы есть автомобили).

При нажатии на кнопку "Редактировать" пользователь попадает на страницу редактирования сведений о фирме, которая описана ниже, при нажатии на кнопку "Удалить" – на страницу удаления записи о фирме, описанной ниже. Страница просмотра ожидает в качестве параметра GET-запроса параметр **id**, который хранит идентификатор фирмы. Если идентификатор не задан или такой фирмы в базе нет, то страница должна остановить выполнение и выдать сообщение об ошибке. Внизу страницы имеется ссылка "Назад", ведущая на страницу просмотра списка фирм.

Страница просмотра сведений об автомобиле отображает следующие сведения: название фирмы, название автомобиля и цвет. Никаких опций добавления/редактирования/удаления не нужно. Внизу страницы имеется ссылка

"Назад к фирме", которая будет переводить пользователя назад на страницу просмотра сведений о фирме- производителе данного автомобиля. Страница просмотра сведений ожидает в качестве параметра GET-запроса параметр **id**, который хранит идентификатор автомобиля. Если идентификатор не задан или такого автомобиля в базе нет, то страница должна остановить выполнение и выдать сообщение об ошибке.

Страница добавления новой фирмы содержит заголовок "Добавление новой фирмы", одно текстовое поле, в которое пользователь введет название. Внизу страницы имеются две кнопки "Добавить" и "Отмена", нажатие на них возвращает назад на эту же страницу, но обрабатывать эти события страница должна по-разному. Если нажата кнопка "Отмена", то происходит принудительный возврат пользователя на общую страницу просмотра списка фирм. Если нажата кнопка "Добавить", то производится валидация. Если она пройдена успешно, то фирма добавляется в базу и осуществляется принудительный переход на страницу просмотра сведений для этой фирмы. Валидация не проходит в следующих случаях: название фирмы пустое, название фирмы превышает допустимую длину текстового поля в базе, фирма с таким названием уже есть. Если валидация не проходит, то пользователь остается на странице добавления и под текстовым полем ему выводится красным цветом сообщение об ошибке – что именно неправильно.

Страница редактирования сведений о фирме выглядит практически так же, как предыдущая. Отличия следующие: заголовок "Редактирование сведений о фирме [Название фирмы]", текстовое поле будет изначально не пустым (в нем название фирмы, извлеченное из базы). При нажатии на кнопку "Отмена" пользователь возвращается на страницу просмотра сведений о фирме. Вместо кнопки "Добавить" будет кнопка "Сохранить". Страница ожидает в качестве параметра POST-запроса параметр **id**, который хранит идентификатор фирмы. Если идентификатор не задан или такой фирмы в базе нет, то страница останавливает выполнение и выдает сообщение об ошибке. Этот параметр должен сохраняться в форме в виде скрытого поля, чтобы при отправке данных из формы это поле снова присутствовало в POST-запросе.

Страница удаления фирмы ожидает в качестве параметра POST- запроса параметр **id**, который хранит идентификатор фирмы. Если идентификатор не задан или такой фирмы в базе нет или у этой фирмы есть автомобили, то страница останавливает выполнение и выдает сообщение об ошибке. В противном случае

страница удаляет записи из базы и выдает сообщение "Фирма [Название фирмы] удалена успешно". Внизу страницы имеется ссылка "Назад", ведущая на страницу просмотра списка фирм.

2. Страница просмотра списка групп должна быть представлена в табличной форме со столбцами "Название", "Просмотр", "Удалить". Название будет красного цвета, если в группе меньше пяти студентов. В столбце "Просмотр" находится ссылка, нажав на которую пользова- тель попадает на страницу просмотра детальных сведений о группе, которая описана ниже. В столбце "Удалить" должны быть флаги, ко- торые пользователь может пометить (т. е. удалить за один раз можно несколько групп). Внизу страницы кнопка "Удалить" переводит на страницу удаления групп. Флаги должны быть только для тех групп, у которых нет студентов, т. е. для группы со студентами в столбце "Удалить" не должно быть ничего.

Страница просмотра сведений о группе содержит следующие сведения: название группы, затем в виде маркированного списка идет список студентов, принадлежащих к данной группе, отсортированный в алфавитном порядке. Внизу страницы имеются кнопка "Редактировать", ведущая на страницу редактирования сведений о группе, а так- же ссылка "Назад", ведущая к странице просмотра списка групп. Страница ожидает в качестве параметра GET-запроса параметр **id**, который хранит идентификатор группы. Если идентификатор не задан или такой группы в базе нет, то страница должна остановить свое выполнение и выдать сообщение об ошибке.

Страница редактирования сведений о группе выглядит следующим образом: заголовок "Редактирование сведений о группе [Название группы]", далее текстовое поле, в котором изначально хранится название группы, извлеченное из базы. Внизу страницы две кнопки "Сохранить" и "Отмена", нажатие на которые ведет назад на эту же страницу, но обрабатывать эти события страница должна по-разному. Если нажата кнопка "Отмена", то осуществляется принудительный возврат пользователя на страницу просмотра сведений о группе. Если нажата кнопка "Сохранить", то производится валидация. Если валидация пройдена успешно, запись обновляется в базе, и происходит принудительный переход на страницу просмотра сведений для этой группы.

Валидация не проходит в следующих случаях: название группы пустое, название группы превышает допустимую длину текстового поля в базе, есть

другая группа с таким же названием. Если валидация не проходит, то пользователь остается на странице редактирования, и под текстовым полем ему выводится красным цветом сообщение об ошибке – что именно неправильно. Страница ожидает в качестве параметра POST-запроса параметр **id**, который хранит идентификатор группы. Если идентификатор не задан или такой группы в базе нет, то страница останавливает выполнение и выдает сообщение об ошибке. Этот параметр должен сохраняться в форме в виде скрытого поля, чтобы при отправке данных из формы это поле снова присутствовало в POST-запросе.

Страница удаления групп ожидает в качестве параметра POST- запроса параметр **id**, который хранит массив идентификаторов групп. Для каждого идентификатора в массиве проводится проверка: если в группе нет студентов, то группа удаляется из базы и выводится новая строка с сообщением "Группа [Название группы] успешно удалена". Если в группе есть студенты (недопустимая ситуация, но все же стоит проверить), то страница должна остановить выполнение и выдать сообщение об ошибке. Внизу страницы должна быть ссылка "Назад", ведущая на страницу просмотра списка групп.

3. Страница просмотра стран строится в табличной форме со столбцами "Название", "Просмотр", "Удалить". Название выделяется курсивом, если у страны нет отелей. В столбце "Просмотр" имеется ссылка, нажав на нее пользователь попадает на страницу просмотра детальных сведений о стране, которая описана ниже. В столбце "Уда- лить" должны быть флаги, которые пользователь может пометить (т. е. удалить за один раз можно несколько стран). Внизу страницы рас- положена кнопка "Удалить", которая переводит на страницу удаления выбранных стран. Флаги отмечают только те страны, у которых нет отелей, т. е. для страны с отелями в столбце "Удалить" не должно быть ничего.

Страница удаления стран ожидает в качестве параметра POST- запроса параметр **id**, который хранит массив идентификаторов стран.

Для каждого идентификатора в массиве проводится проверка: если в стране нет отелей, то она удаляется из базы и выводится новая строка с сообщением "Страна [Название страны] успешно удалена". Если в стране есть отели (недопустимая ситуация, но все же стоит проверить), то страница должна остановить выполнение и выдать сообщение об ошибке. Внизу страницы должна быть ссылка "Назад", ведущая на страницу просмотра списка стран.

Страница просмотра сведений о стране выглядит следующим образом: сначала название страны, затем в виде маркированного списка названия отелей, отсортированные в алфавитном порядке. Каждое название – это ссылка, нажав на которую пользователь попадает на страницу просмотра детальных сведений об отеле. Внизу страницы расположены кнопка "Добавить новый отель", нажав на которую пользователь попадает на страницу добавления нового отеля для данной страны, а также ссылка "Назад", ведущая на страницу просмотра списка стран. Страница ожидает в качестве параметра GET- запроса параметр **id**, который хранит идентификатор страны. Если идентификатор не задан или такой страны в базе нет, то страница должна остановить выполнение и выдать сообщение об ошибке.

Страница просмотра сведений об отдельном отеле выглядит следующим образом: сначала название отеля, затем название страны, далее название фирмы-владельца и стоимость проживания в день. Внизу страницы должна быть ссылка "Назад к стране", ведушая на страницу просмотра сведений о стране, к которой принадлежит отель. Страница ожидает в качестве параметра GET-запроса параметр **id**, который хранит идентификатор отеля. Если идентификатор не задан или такого отеля в базе нет, то страница должна остановить выполнение и выдать сообщение об ошибке.

Страница добавления сведений об отеле выглядит следующим образом: заголовок "Добавление отеля для страны [Название страны]", далее текстовое поле с названием отеля, текстовое поле со стоимостью и выпадающий список фирм. Внизу страницы есть две кнопки "Добавить" и "Отмена", при нажатии на которые пользователь попадает на эту же страницу, но она обрабатывает эти два события по- разному. При нажатии на кнопку "Отмена" пользователь возвращается на страницу просмотра сведений о стране. При нажатии на кнопку "Добавить" производится валидация, если она прошла успешно, то запись добавляется в базу и происходит принудительный переход на страницу просмотра сведений о новом отеле.

Валидация может не пройти в следующих случаях: название отеля пустое, название отеля слишком длинное, отель с таким названием уже есть. Если валидация не пройдена, то пользователь остается на этой же странице, и под текстовым полем ему выводится сообщение об ошибке – что именно неправильно. Страница ожидает в качестве параметра POST-запроса параметр **id**, который хранит идентификатор страны. Если идентификатор не задан или такой страны в базе нет, то страница должна остановить выполнение и выдать

сообщение об ошибке. Этот параметр должен сохраняться в форме в виде скрытого поля, чтобы при отправке данных из формы это поле снова присутствовало в POST-запросе.

4. Страница просмотра списка улиц в виде таблицы со столбцами "Название", "Просмотр". При нажатии на ссылку "Просмотр" пользователь попадает на страницу просмотра сведений об улице. Внизу после таблицы есть кнопка "Добавить", нажав на которую пользователь попадает на страницу добавления новой улицы.

Страница добавления новой улицы содержит одно текстовое поле, в которое пользователь вводит название новой улицы. Внизу кнопка "Добавить" и ссылка "Назад", которая возвращает пользователя на страницу просмотра списка улиц. При нажатии на кнопку "Добавить" пользователь оказывается на этой же странице и производится вали- дация. Если валидация прошла успешно, то улица добавляется в базу, и осуществляется принудительный переход на страницу просмотра сведений о новой улице. Валидация не проходит в следующих случаях: название пустое, название слишком длинное, улица с таким названием уже есть. Если валидация не проходит, то пользователь остается на странице добавления, и под текстовым полем ему выводится красным цветом сообщение об ошибке – что именно неправильно.

Страница просмотра сведений об улице выглядит следующим образом: сначала название улицы, затем в виде маркированного списка дано перечисление адресов на этой улице. Внизу страницы после списка расположена кнопка "Добавить адрес", нажав на которую пользователь переходит на страницу добавления адреса. Также имеется ссылка "Назад", которая возвращает пользователя на страницу просмотра списка улиц. Страница ожидает в качестве параметра GET-запроса параметр **id**, который хранит идентификатор улицы. Если идентификатор не задан или такой улицы в базе нет, то страница должна остановить выполнение и выдать сообщение об ошибке.

Страница добавления сведений об адресе выглядит следующим образом: сначала заголовок "Добавление адреса для улицы [Название улицы]", затем идут два текстовых поля, в которые пользователь вводит номера дома и квартиры. Внизу есть кнопка "Добавить". Нажав на нее, пользователь попадает на эту же страницу и производится вали- дация. Если валидация прошла успешно, то запись об адресе добавляется в базу, и осуществляется принудительный переход на

страницу просмотра сведений об улице. Валидация не проходит в следующих случаях: номера пустые, номера слишком длинные, на этой улице уже есть такой адрес. Если валидация не проходит, то пользователь оста ется на странице добавления, и под текстовым полем ему выводится красным цветом сообщение об ошибке – что именно неправильно. Страница ожидает в качестве параметра POST-запроса параметр **id**, который хранит идентификатор улицы. Если идентификатор не задан или такой улицы в базе нет, то страница должна остановить выполнение и выдать сообщение об ошибке. Этот параметр сохраняется в форме в виде скрытого поля, чтобы при отправке данных из формы это поле снова присутствовало в POST-запросе.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.2 Разработка сетевых приложений (серверная часть)

Лабораторная работа № 6
«Организация файлового ввода-вывода»

Цель: Знакомство с организацией файлового ввода-вывода информации.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: Выполнить задание, представленное ниже.

Порядок выполнения работы:

Часто требуется включать в скрипты необходимые файлы. Две инструкции invlude() и require() позволяют это сделать. В функциональном плане они практически одинаковы. Поэтому мы рассмотрим несколько примеров связанных с одной из них, а затем поясним отличие функций друг от друга. На листинге 6 представлен файл a.php, в котором определяются значения массива $mass. В другом PHP-файле (b.php на листинге 1) производится его подключение. Результат запроса к файлу a.phpв окне браузера представлен на рис. 7.

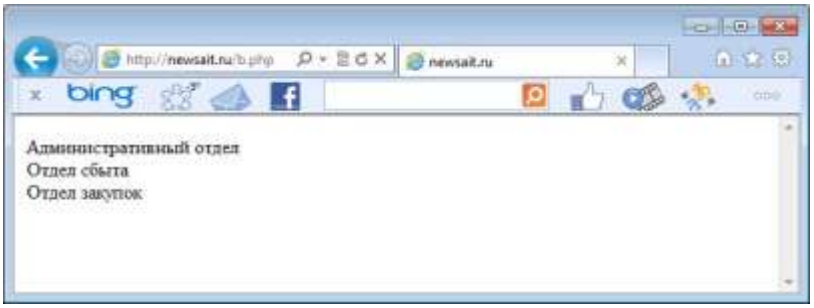


Рис.1

Листинг 1. Подключаемый файл a.php(его выполнять не надо!)

```
<?php
$mass[0]="Административный отдел";

$mass[1]="Отдел сбыта";
$mass[2]="Отдел закупок";
$cols=3;
?>
```

Листинг 2. Файл b.php (выполняемый файл)

```
<?php
include ("a.php");
for ( $i = 0 ; $i < $cols ; $i++) echo $mass[$i]."<BR>";
?>
```

Включаемый функцией include() файл может быть не только PHP-скриптом, он может содержать простой текст.

Упоминаемая выше функция require()отличается от include()реакцией на отсутствие подключаемого файла. Отсутствие файла указанного с помощью директивы require()приводит к остановке обрабатываемого скрипта. В аналогичной ситуации использование include()приводит в выводу на экран предупреждающего сообщения.

Существуют еще две функции аналогичные рассмотренным. Это include_once()и require_once(). Они позволяют в документ подключить файл только один раз, даже если подключений будет несколько.

Открытие файла

Любая работа, связанная с содержимым файла начинается с его открытия. Открытие файла производится с помощью функции fopen(), которая имеет следующий синтаксис:

int fopen(string filename, sring mode)

Первый аргумент filename – относительный или абсолютный пункт к файлу. Второй аргумент задает режим работы (табл. 2) с открываемым файлом.

Таблица 2. Режимы работы с файлом

Значение	Комментарий
r	Открытие файла для чтения, после открытия указатель файла устанавливается в начало файла. Если файл не существует, то функция вернет false.
r+	Открытие файла для чтения и записи, после открытия указатель файла устанавливается в начало файла. Если файл не существует, то функция вернет false.
w	Создание нового пустого файла только для записи; если файл с таким именем уже есть, вся информация в нем уничтожается. После открытия файла указатель устанавливается на начало файла.
w+	Создание нового пустого файла только для чтения и записи; если файл с таким именем уже есть, вся информация в нем уничтожается.
a	Открытие файла для дополнения. Данные записываются в конец файла. Если файл не существует, то функция вернет false.
a+	Открытие файла для дополнения и чтения данных. Данные записываются в конец файла. Если файл не существует, то он будет создан.

После указания режима может следовать модификатор:

- b– файл будет открыт в бинарном режиме (по умолчанию);
- t – файл будет открыт в текстовом режиме.

В случае удачного открытия файла, функция fopen() возвращает дескриптор файла, а в случае неудачи – false.

ПРИМЕЧАНИЕ

Дескриптор файла представляет собой указатель на открытый файл, который используется операционной системой для поддержки операций с этим файлом и представляет уникальное число.

Возвращенный функцией дескриптор файла необходимо указывать во всех функциях, которые используются для работы с файлом.

Листинг 3 содержит пример создания файла a.txt в том каталоге из которого запущен скрипт.

Листинг 3. Создание файла a.txt

```
<?php
$fd = fopen("a.txt","w");
if (!$fd) exit("Ошибка открытия файла a.txt");
?>
```

Открытие двоичного файла, к примеру, рисунка, происходит таким же образом, только с флагом b.

Листинг 4 содержит пример открытия файла с рисунком image.jpgдля чтения.

Листинг 4. Открытие двоичного файла image.jpgдля чтения

```
<?php
$fd = fopen("image.jpg ,"rb");
if (!$fd) exit("Ошибка открытия файла");
?>
```

В рассмотренных примерах считается, что файлы создаются и открываются в том же каталоге, где расположен скрипт. Если же нужно открыть файл из произвольного каталога, то необходимо указать относительный путь.

При формировании относительных путей следует учитывать, что текущий каталог обозначается одной точкой (.), а каталог, расположенный на один уровень выше, обозначается двумя точками (..). Например, если скрипт расположен в каталоге C:/main/test/, а файл необходимо создать в каталоге C:/main/files/, то скрипт должен выглядеть так, как показано на листинге 11. В этом случае мы поднимается на один каталог вверх, а затем опускаемся в каталог files.

Листинг 5. Открытие файла с указанием относительного пути

```
<?php
$fd = fopen("../files/file.txt","w");
if (!$fd) exit("Ошибка открытия файла a.txt");
?>
```

Закрытие файла

После того как работа с файлом завершена, его необходимо закрыть. Закрытие файла осуществляется с помощью функции fclose(), которая имеет следующий синтаксис

int fclose(int fd);

Аргумент fd представляет собой дескриптор файла, который необходимо закрыть.

Если скрипт не закрывает файл, то его закрытие производится автоматически при завершении работы скрипта. По возможности следует закрывать файл сразу же, как с ним прекращена работа, т. к. в то время когда файл открыт, с ним не может работать другой скрипт. Листинг 6 содержит пример закрытия файла.

Листинг 6. Закрытие файла

```
<?php
$fd = fopen("file.txt","w");
if (!$fd) exit("Ошибка открытия файла"); fclose($file);
?>
```

Запись в файл

Рассмотрим теперь как происходит запись информации в файл. Для этого следует использовать практически идентичные функции:

fputs (дескриптор файла, строка, [длина строки]), fwrite (дескриптор файла, строка, [длина строки]).

Первый аргумент представляет собой дескриптор файла, который возвращается функцией fopen(). Второй аргумент является строкой, которая должна быть записана в файл. В этих функциях возможен третий аргумент, который является необязательным и

задает количество символов в строке, которые должны быть записаны. Если третий аргумент не указан, записывается вся строка.

На листинге 7 приведен пример открытия файла, записи в него информации и последующего закрытия. Запустите этот скрипт и посмотрите на присутствие в том же каталоге файла prim1.txt и его содержимое.

Листинг 7. Запись в файл строки

```
<?php
$file = fopen("prim1.txt","w");
fputs($file,"Пример работы с файлами"); fclose($file);
?>
```

Чтение из файла

Чтение содержимого файла можно осуществить при помощи fread(), которая имеет следующий синтаксис:

string fread (дескриптор файла, длина в байтах).

Эта функция принимает в качестве первого аргумента дескриптор открытого файла, а в качестве второго длину строки, которую требуется прочитать из файла (в байтах).

Для чтения всего файла целиком, часто прибегают к функции filesize(), которая возвращает число байтов, содержащихся в файле, имя которого передается функции в качестве аргумента.

Листинг 8. Чтение из файла

```
<?php
$filename="prim1.txt";
// Открываем файл для чтения
$file = fopen($filename,"r");
// Читаем содержимое файла в переменную $bufer
$bufer =fread($file,filesize($filename));
// Закрываем файл fclose($file); echo $bufer;
?>
```

Следующий скрипт на листинге 9 позволяет открыть существующий файл и прочитать из него первые 7 символов.

Листинг 9. Чтение из файла 7-ми символов

```
<?php

$file = fopen("prim1.txt","r");
$inform=fread($file,7); fclose($file);
echo $inform;
?>
```

Для излечения информации из файла также используется функция

fgets(дескриптор),

которая позволяет считывать из файла по одной строке за раз. Считывание будет выполняться до тех пор, пока не встретится символ новой строки (/n) или символ конца файла.

Иногда удобнее прочитать файл посимвольно. В этом случае необходимо контролировать – достигнут ли конец файла? Для этого предназначена функция

feof(дескриптор).

В случае если достигнут конец файла, то функция возвращает true. Рассмотрим пример посимвольного чтения информации из файла (листинг 10).

Листинг 10. Посимвольное чтение из файла

```
<?php
$file = fopen("prim1.txt","r");
$s="";
while (!feof($file))
{
$s=$s.fread($file,1);
}
echo $s;
```

Рассмотрим пример построчного чтения информации из файла (листинг 11).

Листинг 11. Построчное чтение из файла

```
<?php
```

```
$file = fopen("prim1.txt","r"); while (!feof($file))
{
echo fgets($file);
}
fclose($file);
?>
```

Здесь для чтения информации из файла нам пришлось использовать несколько строк кода. Вместо этого можно воспользоваться функцией file(), которая читает файл в массив. Каждый элемент массива – это прочитанная строка из файла. Пример организации построчного чтения из файла:

```
$s=file('file.txt');
```

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Тема 1.2 Разработка сетевых приложений (серверная часть)

Лабораторная работа № 7
«Организация поддержки базы данных в PHP»

Цель: Знакомство с базами данных и способами извлечения данных на уровне PHP.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: Выполнить обработку данных на форме.

Порядок выполнения работы:

Шаг 1. Создать таблицу

Рассмотрим следующую задачу. Пусть нам нужно хранить сведения о студентах. Для этого создадим две таблицы: первая таблица о группах, вторая – о самих студентах (со ссылкой на группу). Предположим, что у нас уже есть база, нужная нам, и требуется создать только сами таблицы

```
CREATE TABLE `group` (
   `id` INT(10) NOT NULL AUTO_INCREMENT,
   `name` VARCHAR(250) NOT NULL,
   PRIMARY KEY (`id`)
) ENGINE = InnoDB DEFAULT_CHARSET = utf8;

CREATE TABLE `student` (
   `id` INT(10) NOT NULL AUTO_INCREMENT,
   `name` VARCHAR(250) NOT NULL,
   `group` INT(10) NOT NULL,
   PRIMARY KEY (`id`),
   FOREIGN KEY (`group`) REFERENCES `group` (`id`) ON DELETE CASCADE
) ENGINE = InnoDB DEFAULT_CHARSET = utf8;
```

Шаг 2. Создать скрипт для заполнения таблицы

Поскольку здесь необходимо проставлять ссылки на таблицу групп, идентификаторы у нее нужно задавать явно. Для второй таблицы идентификаторы можно не задавать, они будут сгенерированы автоматически

```
INSERT INTO ‘group‘ (‘id‘, ‘name‘) VALUES
   (1, ’Group A’), (2, ’Group B’);

INSERT INTO ‘student‘ (‘name‘, ‘group‘) VALUES
   (’Ivanov’, 1), (’Petrov’, 1), (’Sidorov’, 2);
```

Шаг 3. Создать скрипт для удаления таблицы

Заодно на случай переустановки базы необходимо сделать скрипт для удаления таблиц

```
DROP TABLE ‘student‘;
DROP TABLE ‘group‘;
```

Шаг 4. Работа со страницей с данными

Страница будет выглядеть несколько иначе, чем в предыдущих работах. Первым в ней будет фрагмент попытки подключения к базе. Если этого сделать не удалось, то выдается сообщение об ошибке. Затем – фрагмент HTML. Внутри него есть фрагмент прохода по выборке из базы. После фрагмента HTML идет фрагмент очистки соединения.

Шаг 5. Подключение к базе данных

Вот как выглядит этот фрагмент:

```
<?php
$host = 'localhost';
$user = 'root';
$pass = '123';

$link = mysql_connect($host, $user, $pass)
   or die('Failed to connect to the database. Error: '
      . mysql_error());
mysql_select_db('test') or
   die('Failed to select database. Error: ' . mysql_error());
$query = "SELECT student.name as name, `group`.name as gname"
   . "FROM student LEFT JOIN `group`"
   . "ON student.`group` = `group`.id";
$rs = mysql_query($query) or
   die("Failed to execute query. Error: "
      . mysql_error());
?>
```

По итогам этого фрагмента мы имеем выборку данных и подключение к базе.

Вот как выглядит страница. Это обычный одноуровневый список:

```
<html>
   <head><Title>Database</title></head>
   <body>
      <ul>
      <!-- PHP Fragment goes here-->
      </ul>
   </body>
</html>
```

Шаг 6. Проход по выборке

Вот так выглядит проход по выборке:

```
<?php
while ($row = mysql_fetch_assoc($rs)) {
   $name = $row['name'];
   $groupname = $row['gname'];

   echo "<li>Student - $name. Group - $groupname.</li>\n";
}
?>
```

Шаг 7. Освобождение ресурсов

После того как данные выведены и соединение более не нужно, необходимо освободить ресурсы, чтобы база могла получить соединение назад

```
<?php
mysql_free_result($rs);
mysql_close($link);
?>
```

Варианты заданий

В каждом варианте надо написать скрипт для создания таблиц базы, наполнения базы данными и удаления таблиц базы. Заодно нужно написать страницу согласно заданию. Каждая таблица в базе должна содержать нормальный первичный ключ, который генерируется автоматически. Если таблица ссылается на другую, следует прописать нормальный внешний ключ.

Необходимо создать две таблицы. Первая хранит в себе све- дения о фирмах, а именно название фирмы (до 250 символов). Вторая – сведения об автомобилях, производимых фирмами: название автомо- биля (до 150 символов), его стоимость (целое число) и ссылка на фирму. Далее вывести сведения о фирмах и автомобилях в виде двух- уровневого списка. Первый уровень содержит названия фирм в алфа-

витном порядке. На втором расположены сведения об автомобилях для данной фирмы, отсортированные по названиям автомобилей в алфавитном порядке. Все данные страница должна извлечь из базы одним запросом, т. е. нельзя сделать запрос чисто для фирм, а потом для каждой фирмы извлекать дополнительным запросом сведения об автомобилях. Выглядеть страница должна так:

```
1. Ford.
   1. Name - extended. Cost - 2500.
   2. Name - T-model. Cost - 2700.
2. Mercedes.
   1. Name - advanced. Cost - 2800.
   2. Name - Basic. Cost - 2900.
```

Необходимо создать две таблицы. Первая хранит в себе сведе- ния о фирмах, а именно название фирмы (до 100 символов). Вторая – сведения об автомобилях, производимых фирмами: название автомо- биля (до 150 символов), его стоимость (целое число) и ссылка на фирму. Далее вывести сведения о фирмах и автомобилях в виде двух- уровневого списка. Первый уровень содержит названия фирм в алфа- витном порядке. На втором расположены сведения об автомобилях для данной фирмы, отсортированные по стоимости автомобилей в убывающем порядке. Все данные страница должна извлечь из базы одним запросом, т. е. нельзя сделать запрос чисто для фирм, а потом для каждой фирмы извлекать дополнительным запросом сведения об автомобилях. Выглядеть страница должна так:

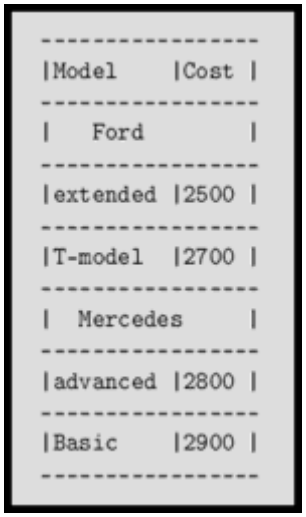
```
* Ford.
   1. Name - T-model. Cost - 2700.
   2. Name - extended. Cost - 2500.
* Mercedes.
   1. Name - Basic. Cost - 2900.
   2. Name - advanced. Cost - 2800.
```

Необходимо создать две таблицы. Первая хранит в себе сведения о фирмах, а именно название фирмы (до 150 символов). Вторая – сведения об автомобилях, производимых фирмами: название автомобиля (до 200 символов), его стоимость (целое число) и ссылка на фирму. Далее вывести сведения о фирмах и автомобилях в виде

одноуровневого списка. Каждый элемент списка должен содержать сведения о фирме, названии модели и стоимости. Отсортировывают элементы по названию фирмы и автомобиля. Выглядеть страница должна так:

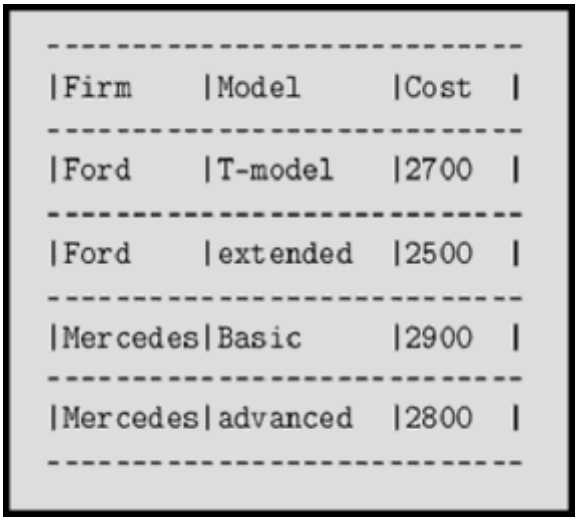
```
1. Firm - Ford. Name - extended. Cost - 2500.
2. Firm - Ford. Name - T-model. Cost - 2700.
3. Firm - Mercedes. Name - advanced. Cost - 2800.
4. Firm - Mercedes. Name - Basic. Cost - 2900.
```

Необходимо создать две таблицы. Первая хранит в себе сведения о фирмах, а именно название фирмы (до 250 символов). Вторая – сведения об автомобилях, производимых фирмами: название автомобиля (до 150 символов), его стоимость (целое число) и ссылка на фирму. Далее вывести сведения о фирмах и автомобилях в виде таблицы из двух столбцов. Сведения о каждой фирме даны в отдельных строках, объединяющих сразу два столбца. Эти строки отсортировывают в алфавитном порядке. Далее для каждой фирмы следуют сведения об автомобилях, остортированные по названиям автомобилей. Все данные страница должна извлечь из базы одним запросом, т.е. нельзя сделать запрос чисто для фирм, а потом для каждой фирмы извлекать дополнительным запросом сведения об автомобилях. Выглядеть страница должна так:



Необходимо создать две таблицы. Первая хранит в себе сведения о фирмах, а именно название фирмы (до 250 символов), вторая – сведения об автомобилях, производимых фирмами: название

автомобиля (до 150 символов), его стоимость (целое число) и ссылка на фирму. Необходимо вывести сведения о фирмах и автомобилях в виде таблицы из трех столбцов. Сведения отсортировывают по названию фирмы (в алфавитном порядке) и стоимости автомобиля (в порядке убывания). Выглядеть страница должна так:



Firm	Model	Cost
Ford	T-model	2700
Ford	extended	2500
Mercedes	Basic	2900
Mercedes	advanced	2800

Необходимо создать две таблицы для хранения сведений об отелях. Первая хранит сведения о фирмах – название фирмы (до 150 символов). Вторая – сведения об отелях: название отеля (до 150 сим- волов), стоимость проживания в отеле и название страны, где распо- ложен отель (до 150 символов), а также ссылка на фирму. Сведения об отелях выводятся в виде двухуровневого списка. На первом уровне должны располагаться названия стран в алфавитном порядке. На вто- ром – сведения об отелях для данной страны, отсортированные по названию фирмы и отеля в алфавитном порядке. Все данные страница должна извлечь из базы одним запросом, т. е. нельзя сделать запрос чисто для стран, а потом для каждой страны извлекать дополнительным запросом сведения об отелях. Выглядеть страница должна так:

```
1. Egypt.
   1. Firm - Arch. Hotel - Awesome. Cost - 1000.
   2. Firm - Arch. Hotel - Best. Cost - 800.
   3. Firm - Tour. Hotel - Epic. Cost - 100.
2. France.
   1. Firm - Arch. Hotel - Ritz. Cost - 1200.
   2. Firm - Arch. Hotel - Zimmer. Cost - 700.
   3. Firm - Tour. Hotel - Fail. Cost - 50.
```

Необходимо создать две таблицы для хранения сведений об отелях. Первая хранит сведения о фирмах – название фирмы (до 150 символов). Вторая – сведения об отелях: название отеля (до 150 сим- волов), стоимость проживания в отеле и название страны, где распо- ложен отель (до 150 символов), а также ссылка на фирму. Сведения об отелях выводятся в виде двухуровневого списка. На первом уровне должны располагаться названия фирм в алфавитном порядке. На вто- ром – сведения об отелях для данной фирмы, отсортированные по названию страны и отеля в алфавитном порядке. Все данные страница должна извлечь из базы одним запросом, т. е. нельзя сделать запрос чисто для фирм, а

потом для каждой фирмы извлекать дополнительным запросом сведения об отелях. Выглядеть страница должна так:

```
* Arch.
   1. Country - Egypt. Hotel - Awesome. Cost - 1000.
   2. Country - Egypt. Hotel - Best. Cost - 800.
   3. Country - France. Hotel - Ritz. Cost - 1200.
   4. Country - France. Hotel - Zimmer. Cost - 700.
* Tour.
   1. Country - Egypt. Hotel - Epic. Cost - 100.
   2. Country - France. Hotel - Fail. Cost - 50.
```

Необходимо создать две таблицы для хранения сведений об отелях. Первая хранит сведения о фирмах – название фирмы (до 150 символов). Вторая – сведения об отелях: название отеля (до 150 сим- волов), стоимость проживания в отеле и название страны, где распо- ложен отель (до 150 символов), а также ссылка на фирму. Необходи- мо вывести сведения об отелях в виде таблицы из трех столбцов. Све- дения о фирмах должны идти в алфавитном порядке в строках, объ- единяющих все три столбца. Для каждой фирмы далее идут строки со сведениями об отелях, отсортированные по названию страны (в алфа- витном порядке), и стоимости отеля (в порядке возрастания). Все данные страница должна извлечь из базы одним запросом, т. е. нельзя сделать запрос чисто для фирм, а потом для каждой фирмы извлекать дополнительным запросом сведения об отелях. Выглядеть страница должна так:

Country	Hotel	Cost
Arch		
Egypt	Best	800
Egypt	Awesome	1000
France	Zimmer	700
France	Ritz	1200
Tour		
Egypt	Epic	100
France	Fail	50

Необходимо создать две таблицы для хранения сведений об отелях. Первая хранит сведения о фирмах – название фирмы (до 150 символов). Вторая – сведения об отелях: название отеля (до 150 симво- лов), стоимость проживания в отеле и название страны, где расположен отель (до 150 символов), а также ссылка на фирму. Сведения об отелях выводятся в виде таблицы из трех столбцов. Сведения о странах должны идти в алфавитном порядке в строках, объединяющих все три столбца. Для каждой страны далее идут строки со сведениями об отелях, отсорти- рованные по названию фирмы (в алфавитном порядке) и стоимости отеля (в порядке возрастания). Все данные страница должна извлечь из базы одним запросом, т. е. нельзя сделать запрос чисто для стран, а потом для каждой страны извлекать дополнительным запросом сведения об отелях. Выглядеть страница должна так:

Firm	Hotel	Cost
Egypt		

Необходимо создать две таблицы для хранения сведений об отелях. Первая хранит сведения о фирмах – название фирмы (до 150 символов). Вторая – сведения об отелях: название отеля (до 150 символов), стоимость проживания в отеле и название страны, где расположен отель (до 150 символов), а также ссылка на фирму. Сведения об отелях выводятся в виде таблицы из четырех столбцов. Сведения отсортировывают по названиям стран, фирм и отелей. Выглядеть страница должна так:

Форма представления результата:

Предоставить отчет, который будет включать в себя:

6. Цель работы.
7. Вариант задания.
8. Исходный текст программы на PHP.
9. Скриншоты сгенерированных страниц.
10. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 8
«Отслеживание сеансов (session)»

Цель: Изучение технологии и получение практических навыков отслеживания сеанса пользователя.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: Модифицировать результаты лабораторной работы №7 путем добавления возможности формирования корзины товаров. Для этого необходимо напротив каждого товара установить кнопку (ссылку) «Добавить в корзину». Предусмотреть возможность просмотра корзины и ее модификации (добавление, удаление товаров). Осуществлять контроль над количеством каждого товара в корзине необязательно.

Краткие теоретические сведения

Сеансом (session) называется период времени, который начинается с момента прихода пользователя на сайт и завершается, когда пользователь покидает сайт. В течение сеанса часто возникает необходимость в сохранении различных переменных, которые бы «сопровождали» пользователя при перемещениях на сайте, чтобы вам не приходилось вручную кодировать многочисленные скрытые поля или переменные, присоединяемые к URL.

При входе на сайт пользователю присваивается уникальный идентификатор сеанса (SID), который сохраняется на компьютере пользователя в cookie с именем PHPSESSJD. Если использование cookie запрещено или cookie вообще не поддерживаются, SID автоматически присоединяется ко всем локальным URL на протяжении сеанса. В то же время на сервере сохраняется файл, имя которого совпадает с SID. По мере того как пользователь перемещается по сайту, значения некоторых параметров должны сохраняться в виде сеансовых переменных. Эти переменные сохраняются в файле пользователя. При последующем обращении к сеансовой переменной сервер открывает сеансовый файл пользователя и ищет в нем нужную переменную. В сущности, в этом и заключается суть

отслеживания сеанса. Конечно, информация с таким же успехом может храниться в базе данных или в другом файле.

Рассмотрим процесс отслеживания сеанса более подробно. Первое - сеанс инициируется функцией session_start( ).

Функция session_start( ) имеет двойное назначение. Сначала она проверяет, начал ли пользователь новый сеанс, и если нет — начинает его. Синтаксис функции session_start( ):

boolean session_start()

Если функция начинает новый сеанс, она выполняет три операции: назначение пользователю SID, отправку cookie и создание файла сеанса на сервере. Второе назначение функции заключается в том, что она информирует ядро PHP о возможности использования в сценарии, в котором она была вызвана, сеансовых переменных.

Если сеанс можно создать, значит, его можно и уничтожить. Это делается функцией session_destroy().

Функция session_start( ) возвращает TRUE независимо от результата. Следовательно, проверять ее в условиях if или в команде die( ) бессмысленно.

boolean session_destroy( )

Пример использования функции:

```
<?
session_start( );
// Выполнить некоторые действия для текущего сеанса
session_destroy( ):
?>
```

Теперь вы умеете уничтожать сеансы, и мы можем перейти к работе с сеансовыми переменными. Возможно, самой важной сеансовой переменной является SID (идентификатор сеанса). Его легко можно получить при помощи функции session_id( ).

session_id( )

Функция session_id( ) возвращает SID для сеанса, созданного функцией session_start( ). Синтаксис функции session_id( ):

string session_id ([string sfd])

Если в необязательном параметре передается идентификатор, то значение SID текущего сеанса изменяется. Однако следует учитывать, что cookie при этом заново не пересылаются. Пример:

```
<?
session_start()
print "Идентификатор сессии = " . sessionjd( ):
```

session_destroy( ):

?>

Результат, выводимый в браузере, выглядит примерно так:

Идентификатор сессии = 067d992a949114ee9832flcllcafc640

Сеансовая переменная создается с помощью функции session_register( ).

session_register( )

Функция session_register( ) регистрирует имена одной или нескольких переменных для текущего сеанса. Синтаксис функции session_register( ):

boolean session_register (mixed имя_переменной1 [, mixed имя_переменной2...])

Следует помнить, что регистрируются не переменные, а их имена. Если сеанс не существует, функция session_register( ) также неявно вызывает session_start( ) для создания нового сеанса.

Проверить зарегистрирована ли переменная в сессии можно функцией session_is_registered(). session_is_registered( )

Часто требуется определить, была ли ранее зарегистрирована переменная с заданным именем. Задача решается при помощи функции session_is_registered( ), имеющей следующий синтаксис:

Порядок выполнения работы:

boolean session_is_registered (string имя_переменной)

Применение функций session_register( ) и session_is_registered( ) будет продемонстрировано на классическом примере использования сеансовых переменных — счетчике посещений (листинг 1).

Листинг 1. Счетчик посещений сайта пользователем

<?

session_start( ):

if (! sessionjs_registered('hits')) :

session_register('hits') ;

endif ;

$hits++:

print "You've seen this page $hits times.

?>

Сеансовые переменные удаляются применением функции session_unregister( ).

session_unregister( )

Ниже приведен ее синтаксис:

boolean session_unregister (string имя_переменной')

При вызове функции передается имя сеансовой переменной, которую вы хотите уничтожить.

```
<?
session_start()
session_register('username');
// Использовать переменную $username.
// Когда переменная становится ненужной - уничтожить ее.
session_unregister('username');
session_destroy();
?>
```

Как и в случае с функцией session_register указывается не сама, а имя переменной.

session_encode( )

Функция session_encode( ) обеспечивает чрезвычайно удобную возможность форматирования сеансовых переменных для хранения (например, в базе данных). Синтаксис функции session_encode( ):

boolean session_encode( )

В результате выполнения этой функции все сеансовые данные форматируются в одну длинную строку, которую можно сохранить в базе данных.

Пример использования session_encode( ) приведен в листинге 2. Предположим, что на компьютере «зарегистрированного» пользователя имеется cookie, в котором хранится уникальный идентификатор этого пользователя. Когда пользователь запрашивает страницу, содержащую листинг 2, UID читается из cookie и присваивается идентификатору сеанса. Мы создаем несколько сеансовых переменных и присваиваем им значения, после чего форматируем всю информацию функцией session_encode( ) и заносим в базу данных MySQL.

Листинг 2. Использование функции session_encode( ) для сохранения данных в базе данных MySQL

```
<?
// Инициировать сеанс и создать сеансовые переменные
session_register('bgcolor');
session_register('fontcolor');
// Предполагается, что переменная $usr_id (с уникальным идентификатором пользователя) хранится в cookie
// на компьютере пользователя.
// При помощи функции session_id( ) присвоить идентификатору
```

```
//сеанса уникальный идентификатор пользователя (UID), хранящийся в cookie.
$id = session_id($usr_id);
// Значения следующих переменных могут задаваться пользователем на форме HTML
$bgcolor = "white"; $fontcolor = "blue";
// Преобразовать все сеансовые данные в одну строку
$usr_data = session_encode( );
// Подключиться к серверу MySQL и выбрать базу данных users
@mysql_pconnect("localhost", "root", "") or die("Could not connect to MySQL server!");
@mysql_select_db("users") or die("Could not select user database!");
// Обновить пользовательские параметры страницы
$query = "UPDATE user_info set page_data='$usr_data' WHERE user_id= '$id'";
$result - mysql_query($query) or die("Could not update user information!");
?>
```

Как видно, быстрое преобразование всех сеансовых переменных в одну строку избавляет от необходимости создавать несколько полей для хранения/загрузки данных, а также несколько уменьшает объем программы.

session_decode( )

Все сеансовые данные, ранее преобразованные в строку функцией session_encode( ), восстанавливаются функцией session_decode( ). Синтаксис:

string session_decode (string сеансовые_данные)

В параметре сеансовые_данные передается преобразованная строка сеансовых переменных, возможно — прочитанная из файла или загруженная из базы данных. Строка восстанавливается, и все сеансовые переменные в строке преобразуются к исходному формату.

В листинге 3 продемонстрировано восстановление закодированных сеансовых переменных функцией session_decode( ).

Предположим, таблица MySQL с именем **user_info** состоит из двух полей: **user_id** и **page_data**. Пользовательский UID, хранящийся в cookie на компьютере пользователя, применяется для загрузки сеансовых данных, хранящихся в поле **page_data.** В этом поле хранится закодированная строка переменных, одна из которых ($bgcolor) содержит цвет фона, выбранный пользователем.

Листинг 4. Восстановление сеансовых данных, хранящихся в базе данных MySQL

```
<?
// Предполагается, что переменная $usr_id (с уникальным идентификатором пользователя) хранится в cookie
```

```
// на компьютере пользователя.
$id = session_id($usr_id);
// Подключиться к серверу MySQL и выбрать базу данных users
@mysq]_pconnect("localhost", "web", "4tf9zzzf") or die("Could not connect to MySQL server!");
@mysql_select_db("users") or die("Could not select company database!");
// Выбрать данные из таблицы MySQL
$query = "SELECT page_data FROM user_info WHERE user_id= '$id'",
$result = mysql_query($query);
$user_data = mysql_result($result, 0, "page_data");
// Восстановить данные
session_decode($user_data):
// Вывести одну из восстановленных сеансовых переменных
print "BGCOLOR: $bgcolor";
?>
```

Как видно из двух приведенных листингов, функции session_encode() и session_decode() обеспечивают очень удобные и эффективные сохранение и загрузку сеансовых данных.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 9
«Создание проекта «Регистрация»

Цель: Изучение технологии и получение практических навыков отслеживания сеанса пользователя.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: создать форму регистрации на сайте.

Порядок выполнения работы:

Включение формы делает страничку интерактивной. Фактически в этом случае пользователю обеспечивается диалог с Web-сервером. Посетитель вводит запрашиваемую информацию. Это могут быть - фамилия, адрес, банковские реквизиты.

Важно заметить, что форма является элементом языка *HTML*, а не *PHP*. Технология *PHP* предоставляет только механизм работы. В дальнейшем тексте мы практически в каждой разработке будем использовать формы.

Форма в HTML-документе реализуется *тегом-контейнером* <FORM>, в котором задаются необходимые управляющие элементы - поля ввода, кнопки и.т.д.

Если управляющие элементы указаны *вне содержимого тега* FORM, то они не создают форму, а используются для построения пользовательского интерфейса на HTML-странице.

Обработка элементов формы производится с помощью скриптов (PHP- сценариев). Имена элементам формы присваиваются через их атрибут name.

На листинге 19 представлен файл формы (*19.html*). Это обыкновенный HTML-файл без скриптов и в окне браузера он приводит к отображению формы (рис. 10). Здесь перед пользователем открывается поле ввода и кнопка с помощью которой информация от элементов формы отправляется на сервер.

Листинг 19. Файл 19.html, обеспечивающий Web-страницу на рис. 10

```
<HTML> <HEAD><TITLE> Форма </TITLE>

</HEAD><BODY>

<FORM action = "20.php">

<P>Ваша фамилия:

<INPUT type = "text" name = "fio" value="" title="Поле для фамилии">
</P>

<INPUT type="submit" value="Передать данные на сервер" name="OK"
title="Отправка информации на сервер">
</FORM>

</BODY></HTML>
```

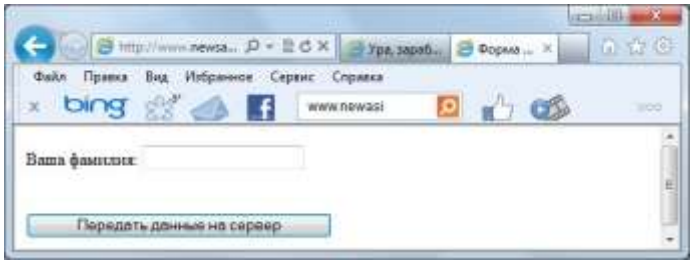


Рис. 10.

Начнем рассмотрение листинга 19 с ключевого тега – <FORM>. Фактически он объявляет форму. Для окончания объявленной формы существует соответствующий закрывающий тег </FORM>. У тега <FORM> имеется несколько параметров из которых на листинге 19 мы использовали только один. Отметим только, что задать имя формы можно с помощью еще одного параметра – name. В большинстве случаев

этот параметр устанавливать не требуется, и в наших примерах мы так и будем поступать.

В примере на листинге 19 указано значение параметра action action
= "20.php",
которое представляет имя программы (скрипта). Эта программа будет обрабатывать данные на Web-сервере полученные от элементов формы.

ПРИМЕЧАНИЕ

Отправка формы – это обычный запрос к серверу, только вместе с запросом передается еще и дополнительная информация – данные от формы.

Если параметр action не указать, то данные будут переданы текущей программе (самой себе). Она будет выполнена на сервера повторно, но уже с принятыми значениями от элементов формы. Такая ситуация достаточно распространена.

Еще один параметр – method, и он определяет способ отправки формы на сервер. Этих способов два – GET и POST, различия между ними мы рассмотрим позже. Заметим только, что если способ отправки не указан, то предполагается использование GET.

Вернемся к рис. 10. Если пользователь внес информацию в текстовое поле и нажал на кнопку Передать данные сервер, то к указанному в разделе action скрипту 20.php произойдет обращение на сервере.

Как уже сказано, основное назначение формы заключается в том, чтобы быть контейнером для различных элементов. Наиболее распространенным элементом является *тестовое поле* (его также часто называют – *поле для ввода*). На нашей форме этот элемент объявлен следующим образом:

<INPUT type = "text" name = "fio" value="" title="Поле для
фамилии">.

Значение параметра type равное text как раз и определяет тип элемента – *поле для ввода*. Параметр name – *имя элемента*, и когда информация будет отправляться на сервер, то данные от этого элемента передается в виде:

Значение **name** = Значение **value.**

Указанный здесь параметр value определяет текст, который будет отображаться в поле ввода. Еще один параметр title – это всплывающая подсказка, которая будет отображаться при наведении мыши на поле для ввода.

Имеется еще один параметр, который в приведенном примере не указан:

size - определяет длину поля для ввода (в качестве значения данного параметра необходимо указать число).

В качестве альтернативы объявлению поля ввода на листинге 6.10 можно использовать более информативную запись:

<INPUT type = "text" name = "fio"

title="Поле для фамилии" value=”Здесь необходимо ввести свою фамилию”
size=40>

На листинге 19 расположен еще один элементу управления – *кнопка* с помощью которой данные отправляются на сервер. Программно *кнопка* создается также с использованием тега <INPUT>, но в качестве значения параметра type следует указать submit. В рассматриваемом примере мы использовали полный набор параметров тега определяющего на форме кнопку:

<INPUT type="submit" value="Передать данные на сервер" name="OK"
title="Отправка информации на сервер">

Рассмотрим, что представляет собой передача данных от элементов формы скрипту на сервере - скрипту 20.php.

Для того, чтобы другая программа могла получить данные формы можно использовать один двух методов:

- GET – в этом случае данные будут передаваться присоединенными к имени программы указанной в параметре action;
- POST–данные в этом случае будут отправляться незаметно для пользователя.

В рассмотренном примере мы не указывали метод передачи данных скрипту на сервере:

<FORM action = "20.php">

В этом случае вступает правило – по умолчанию считается, что данные передаются методом GET.

На листинге 20 приведен пример скрипта, который обрабатывает данные полученные от формы на рис. 10.

Листинг 20. Файл 20.phpдля обработки файла с формой

```
<html><body>

    <?php

    echo "Добрый день,    ".$_GET['fio']."!";

    ?>

    </body></html>
```

В данном тексте учтено, что для имени поля ввода было использовано fio. В связи с этим значение, которое ввел пользователь на сервере можно получить с помощью:

$_GET['fio']

Результат выполнения данного скрипта показан на рис. 11. Если взглянуть на адресную строку, то мы увидим, что к имени скрипта добавляется строка символов с передаваемой информацией. Если вы в Интернете увидите подобную строку (содержащую знак вопроса), то перед вами передача параметров методом GET. Как раз знак вопроса (?) после имени переменной означает начало передачи данных методом GET. После него следуют пары имя=значение, разделенные символом амперсанда (&). В обрабатывающем скрипте доступ к GET-параметрам можно получить через суперглобальный массив $_GET, указав в квадратных скобках после него имя параметра.

Рис. 11.

Передача методом GET часто не удобна по ряду причин:

- пользовать видит в адресной строке параметры и при желании может изменить их значения;
- объем передаваемой информации ограничен.

Чтобы сервер мог данные отличить, они присоединяются по специальным правилам. Сначала после имени программы-обработчика ставится знак вопроса (?), что указывает передачу данных. После этого указывается имя первого объявленного элемента на форме (в нашем случае это текстовое окно), далее знак равенства и значение. Если элемент один, то формирование запроса считается оконченным. После этого он отправляется на сервер. Если же элементов несколько, то информация о следующем отделяется от предыдущего знаком &. Формат записи очередного элемента строится по описанной схеме: имя элемента, знак равно, значение элемента.

ПРИМЕЧАНИЕ

Стоит отметить, что русские буквы передаются в закодированном виде, так что вы не увидите русских букв – они будут заменены кодами.

Поясним теперь в чем отличительная особенность метода POST. Для того, чтобы использовать данный метод передачи необходимо указать у формы в качестве значения параметра action вариант POST. В этом случае для извлечения значений параметров формы следует использовать суперглобальный массив $_ POST.

Для иллюстрации данного метода передачи изменим файл с формой (новый вариант на листинге 21) и скрипт обработки формы на сервере (листинг 22).

Листинг 21. Файл 21.html для формы

```
<HTML> <HEAD><TITLE> Форма  </TITLE>

</HEAD><BODY>

<FORM method="POST" action = "22.php"> Ваша фамилия:
<INPUT type = "text" name = "fio" value="" title="Поле для фамилии">
```


<INPUT type="submit" value="Передать данные на сервер" name="OK" title="Отправка информации на сервер">

</FORM>

</BODY></HTML>

Листинг 22. Файл 22.php обработки данных от формы

```
<html><body>

<?php

echo "Добрый день,    ".$_POST['fio']."!";

?>

</body></html>
```

Как видно (рис. 12) теперь в адресной строке нет никаких параметров - при использовании метода POSTданные отправляются незаметно для пользователя в теле запроса.

Рис. 12.

Элемент формы для пароля

В плане работы с формой мы использовали текстовое окно и кнопку, но ими ассортимент элементов не исчерпывается. Здесь мы приведем примеры использования других элементов в программе. Начнем с поля для ввода пароля, которое объявляется точно также как и поле ввода. Только в параметре type необходимо указать значение password.Например, следующая запись:

<INPUT type=”password”

name=”pass” title=”Поле для ввода имени”>

создает на форме поле для ввода пароля (все вводимые символы будут выглядеть звездочками). Продемонстрируем сказанное на примере. На листинге 23 представлен файл, который позволяет создать на странице форму с необходимостью указания пароля (рис. 13).

Листинг 23. Файл 23.html, обеспечивающий форму с указанием пароля

```
<HTML> <BODY>

<FORM ACTION = "24.php">

Введите пароль:

<INPUT type = "password" name = "pass"

value=""                title="Необходимо ввести пароль">
<INPUT TYPE ="submit" value="Отправка значения пароля на сервер" name="OK">
</FORM> </BODY></HTML>
```

Рис. 13.

В этом случае вводимый пользователем текст будет скрыт от взгляда посторонних, а отправляемый серверу введенный пароль проверяется на соответствие истинному (условно в качестве правильного пароля принята комбинация *123*). Файл, обрабатывающий данный запрос представлен на листинге 24.

Листинг 24. Скрипт 24.php, обрабатывающий форму на рис. 13

```
<HTML><BODY>

<?php

if ($_GET['pass'] == "123")

echo "Привет! Вы прошли авторизацию."; else
echo "Ошибка при указании пароля.";

?>

</BODY></HTML>
```

Результат работы скрипта при правильном указании пароля продемонстрирован на рис. 14.

Рис. 14.

Скрытое поле

Важный элемент управления – *скрытое поле*. Данный элемент нельзя увидеть и его назначение заключается в передаче данных, которые не должны изменяться пользователем.

Скрытое поле создается следующим образом:

```
<input name=”имя” type=”hidden” value=”значение”>
```

Форма для регистрации вопросов на сайте

В этом разделе мы рассмотрим создание формы, которая позволяет посетителям сайта оставить вопрос администратору, либо другому руководящему сотруднику. Разумеется все это можно сделать и с помощью средств электронной почты, однако рассматриваемый механизм интереснее и кроме того он поможет нам попрактиковаться в PHP.

На рис. 15 представлена форма, которую нам следует реализовать. Здесь расположены три поля вода и кнопка для отправки сведений на сервер. Для формирования данных элементов на странице формы нам потребуется разработать HTML-файл (листинг 25).

Подобная форма регистрации посетителя представляет механизм общения посетителя сайта с его группой администрирования.

Листинг 25. Файл для регистрации посетителя 25.html

```
<form action="26.php" method="post">
Ваша фамилия и имя <br> <input type="text" name="fam"><br>

Адрес электронной почты или

контактный телефон <br> <input type="text" name="adr">

<br> Сообщение:<br> <br>

<textarea name="info" cols=10 rows=5>

</textarea> <br>

<input type="submit" name="ok" value="OK">

</form>
```

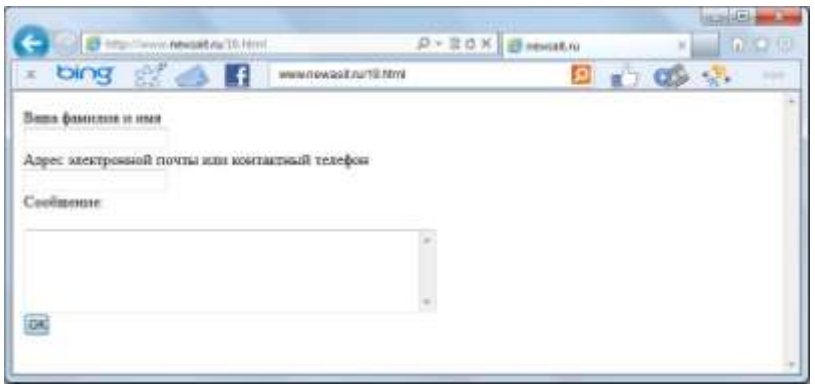


Рис. 15.

В заголовке формы

<form action="26.php" method="post">

указано название скрипта, который должен обрабатывать данные формы. Этот скрипт приведен на листинге 26. В нем в первую очередь проверяется наличие в полученных сведений информации от элемента с именем ok – запущен ли скрипт в результате щелчка по кнопке с указанным именем. Только при положительном ответе на данный вопрос производятся действия по открытию файла и внесения в него информации.

В результате открытия файла

$a=fopen("info.txt","at")

в переменную $a записывается дескриптор открытого файла. В дальнейшем всяработа происходит через данную переменную и нам не требуется больше указывать имя файла и режим работы с ним. В следующей строке производится блокировка файла

flock($a,2)

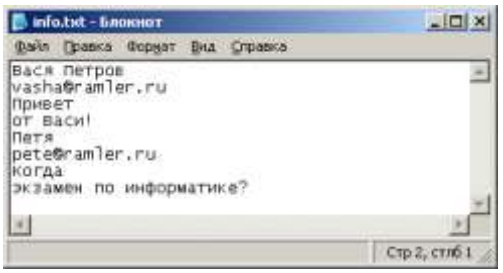
Благодаря этому другой посетитель не сможет внести изменения в данный файл. Далее в файл с помощью функции fputs() последовательно записывается информация о фамилии, адресе и самом сообщении. В результате в файл info.txt вносится информация (рис. 16) от очередного посетителя. После этого осуществляется разблокировка файла:

flock($a,3).

Листинг 26. Файл 26.phpдля обработки данных формы на рис. 15

```
<?php

if (isset($_POST['ok']))

{

$a=fopen("info.txt","at"); flock($a,2); fputs($a,$_POST['fam']."\n");
fputs($a,$_POST['adr']."\n");

fputs($a,$_POST['info']."\n"); flock($a,3);
```

```
fclose($a);

echo "Информация внесена в файл";

}

?>
```



Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 10
«Создание проекта «Интернет магазин»

Цель: Создание интернет магазина.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: выполнить задание, представленное ниже.

Порядок выполнения работы:

1. Создание внешней таблицы стилей

Создать на "Рабочем столе" каталог "lab4". Создать в каталоге "lab4" текстовый файл "styles.css", для этого

Запустить текстовый редактор, например, программу "Блокнот" (меню "Пуск"->"Программы"->"Стандартные"->"Блокнот")

В меню "Файл" выбрать пункт "Сохранить как..."

Указать каталог "lab4"

Указать имя файла "styles.css" (имя файла взять в двойные кавычки, для того, чтобы автоматически к названию файла не добавлялось расширение .txt)

Нажать на кнопку "Сохранить"

Начиная с первой строки файла "styles.css" между символами "/*" и "*/" расположить комментарий, указывающий на авторские права в произвольной форме на английском языке. Далее в файле "styles.css" вручную задать все необходимые свойства тегов, используемых в Ваших html-страницах.

Например, для тега <body> можно установить свойства отступов от границ окна, равными нулю и цвет фона. Для задания цвета и размера текста, размещаемого внутри тегов в файл стилей "styles.css" можно добавить: для задания свойств заголовков первого уровня

<h1></h1>: для задания свойств таблиц <table></table>: для задания свойств всех ячеек таблиц <td></td>. Далее для всех тегов, которые будут встречаться в html-коде Ваших трех страниц. Сохранить файл "styles.css" на диск.

2. Создание заглавной страницы "index.html"

Создать в каталоге "lab4" текстовый файл "index.html". В файле "index.html" вручную разместить html-код страницы.

Тег <link> подключает внешнюю таблицу стилей "styles.css". Между тегами <title></title> расположить название страницы, например, "Томск - Главная". Затем, между тегами <body></body> расположить таблицу, состоящую, например, из четырех строк, а в каждой строке по три ячейки, причем в первой, второй и четвертой строке три ячейки объединены в одну.

Такая таблица в этом случае должна выглядеть примерно так:

Наполнение страницы информацией (контентом)

В первой ячейке первой строки таблицы разместить герб и название города

Во первой ячейке второй строки таблицы поместить информацию о назначении Вашего сайта, например, "Информационный портал", "Учебный проект"

В первой ячейке третьей строки разместить три гиперссылки на каждую из Ваших html-страниц ("index.html", "info.html", "map.html")

Во второй ячейке третьей строки разместить описание сайта (это та первая информация, которую видит новый посетитель, по-этому она должна быть краткой и по сути дела), например, "На сайте представлена информация о городе Томск. Для получения информации воспользуйтесь меню в левой части окна."

В третьей ячейке третьей строки расположить ссылки на другие сайты похожей тематики, и другую информацию, которая может заинтересовать посетиля сайта

В первой ячейке четвертой строки поместить подпись разработчика

Задание размеров ячеек

Если посмотреть на таблицу, то видно, что она состоит из шести ячеек. По-этому можно в таблицу стилей добавить шесть стилей - для каждой ячейки свой стиль (высота, ширина, цвет).

Добавления в файл "styles.css"

Например, можно добавить в файл "styles.css" стиль первой ячейки "cell1" (высота ячейки: 100 пикселей): создать стиль второй ячейки (высота ячейки: 20 пикселей): для третьей (высота: 500, ширина: 200): и так для каждой ячейки: "cell4" (высота: 500), "cell5" (высота: 500; ширина: 200), "cell6" (высота: 30).
Сохранить файл "styles.css" на диск.

Добавления в файл "index.html"

В файле "index.html" в свойствах каждой ячеки указать название применяемого к ней стиля. Например, для первой ячейки в теге <td> добавить свойство class='cell1'

Для второй ячейки указать и так далее (нумерация ячеек идет сверху вниз и слево направо). В итоге в теге <td> каждой ячейки должен быть указан стиль. Сохранить файл "index.html" на диск.

Задание внешнего вида ячеек

При необходимости для каждой ячейки можно задать (добавлением свойств в стили "cell1", "cell2", ... "cell6")

цвет шрифта (например, "color: blue;")

размер шрифта (например, "font-size: 12pt;")

начертание шрифта (например, "font-weight: bold;")

тип шрифта (например, "font-family: Sans-Serif, Arial, Tahoma, Helvetica;")

выравнивание текста (например, "text-align: center;")

вертикальное выравнивание в ячейке (например, "vertical-align: top;")

фоновый цвет (например, "background-color: lightblue;")

фоновое изображение (например, "background-image: url('clover.png');")

3. Создание страницы с информацией о городе "info.html"

Создать текстовый файл "info.html" в каталоге "lab4"

Копировать весь текст из файла "index.html"

Поменять название страницы в теге <title>, например на, "Томск - Информация"

Поменять содержимое второй ячейки третьей строки на информацию о городе

Сохранить файл на диск

4. Создание страницы с картой города "map.html"

Создать текстовый файл "map.html" в каталоге "lab4"

Копировать весь текст из файла "index.html"

Поменять название страницы в теге <title>, например на, "Томск - Карта"

Поменять содержимое второй ячейки третьей строки на изображение карты города

Сохранить файл на диск

5. Размещение сайта на сервере

Создать на сервере в корневом каталоге подкаталог "lab3" (именно номер 3, для сохранения файлов предыдущей лабораторной работы)

Скопировать содержимое корневого каталога (файлов предыдущей лабораторной работы) в каталог "lab3" на сервере

Скопировать содержимое каталога "lab4" локального компьютера в корневой каталог на сервере

6. Оформление отчета по лабораторной работе

Отправить электронное письмо на адрес teacher@meteolab.ru. В теме письма обязательно указать номер лабораторной работы, например, "lab4". В тексте письма указать адрес сайта, а также Фамилию Имя Отчество (прикрепленные файлы не отправлять). Информацию о зачете лабораторной работы можно посмотреть в разделе "Список групп" рядом с описаниями лабораторных работ.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 11
«Составление схем XML-документов»

Цель: Создание интернет магазина.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: Создайть XML-документ, который будет содержать информацию по вашей специальности в других университетах (университет, проходной балл, план набора, город, в котором размещен университет). При выполнении задания используйтесss.

Краткие теоретические сведения

XML (от англ. eXtensible Markup Language) – расширяемый язык разметки. Он создан для структурирования, хранения и передачи информации. XML – это общий инструмент передачи данных между всеми видами приложений. В языке XML нет предопределенных тегов, автор определет свои языковые теги и свою структуру документа.

Пример XML документа

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
<to>Анна</to>
<from>Дмитрий</from>
<heading>Напоминание</heading>
<body>Не забудь обо мне в эти выходные!</body>
</note>
```

Первая строка – это XML декларация. Здесь определяется версия XML (1.0). На следующей строке описывается корневой элемент документа: <note>. Следующие 4 строки описывают дочерние элементы корневого элемента: <to>Анна</to>, <from>Дмитрий</from>, <heading>Напоминание</heading>, <body>Не забудь обо мне в эти выходные!</body>. И, наконец, последняя строка определяет конец корневого элемента: </note>.

Исходя из этого примера, можно смело предположить, что в этом XML документе содержится заметка от Дмитрия к Анне. И все вполне понятно.

Правила синтаксиса XML- документа

XML-документы должны удовлетворять следующим правилам:

- все XML элементы должны иметь закрывающий тег;
- теги XML являются регистрозависимыми;
- все элементы обязаны соблюдать корректную вложенность;
- значения атрибутов должны заключаться в кавычки;
- XML документ должен содержать один корневой элемент, который будет родительским для всех других элементов;
- учитываются все символы форматирования (т.е. пробелы, переводы строк, табуляции не игнорируются, как в HTML).

Если XML документ составлен в соответствии с приведенными синтаксическими правилами, то говорят, что это «синтаксически верный» или «корректно сформированный» XML документ.

Комментарии в XML

<!—это комментарий -->.

Подключение файла таблицы стилей

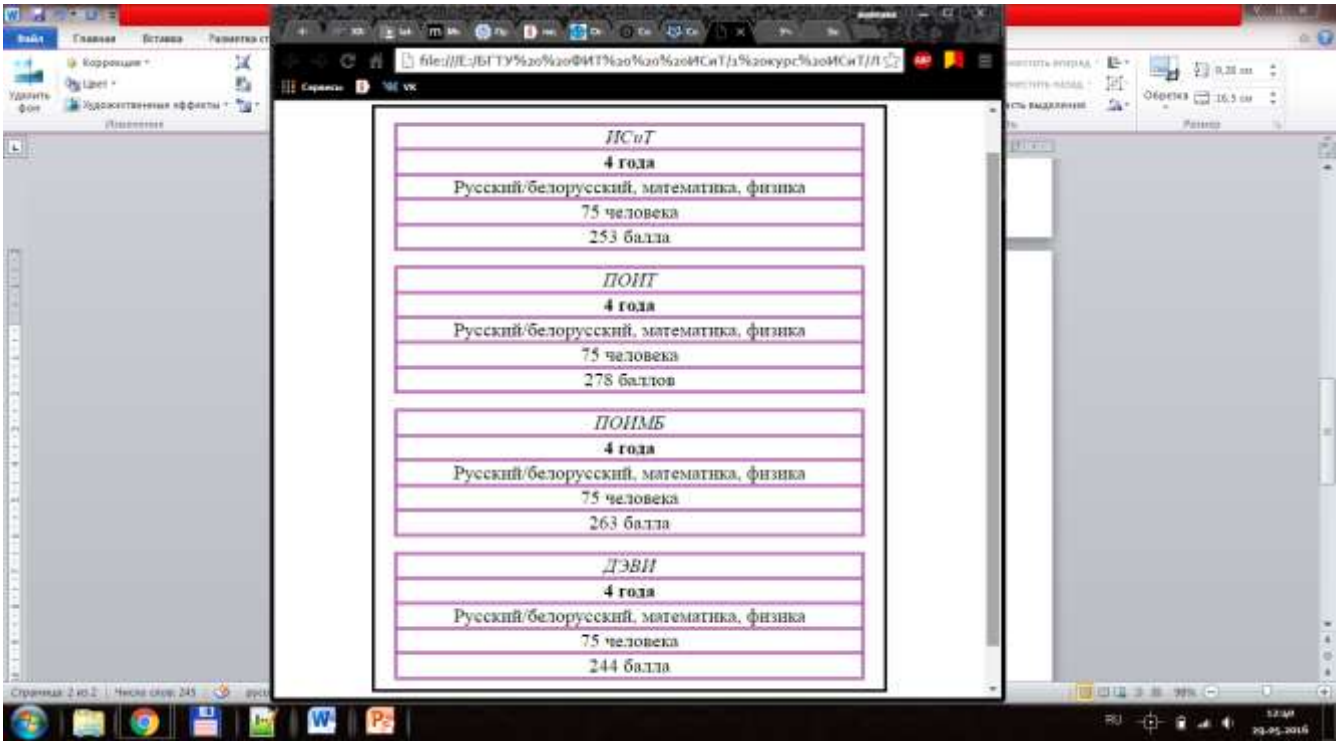
<?xml-stylesheet type="text/css" href="1.css"?>.

Порядок выполнения работы:

XML элементы должны следовать следующим правилам написания имен:

- имена могут содержать буквы, числа и другие символы;
- имена не могут начинаться с цифры или символа пунктуации;
- имена не могут начинаться с сочетания "xml" (или XML, или Xml и т.п.);
- имена не могут содержать пробельные символы.

В качестве имен можно использовать любые слова. Нет зарезервированных слов.



Данная таблица была создана XML-документом с подключением css. Структура файла представлена ниже:

<?xmlversion="1.0" encoding="UTF-8"?>

<!--FileName: 1.xml>

<?xml-stylesheet type="text/css" href="1.css"?>

<FACULTY>

<SPECIALIZATION>

<NAME>

<TIME>

<EXAM>

<PAGES>

<PASSING>

</SPECIALIZATION>

<SPECIALIZATION>

<NAME>

<TIME>

<EXAM>

<PAGES>

<PASSING>

</SPECIALIZATION>

<SPECIALIZATION>

<NAME>
<TIME>
<EXAM>
<PAGES>
<PASSING>
</SPECIALIZATION>
<SPECIALIZATION>
<NAME>
<TIME>
<EXAM>
<PAGES>
<PASSING>
</SPECIALIZATION>
</FACULTY>

В css-файле для каждого тэга описаны свойства:

```
SPECIALIZATION
{ display:block;
text-align:center;
margin:10px 20px 10px 20px;
margin-top:12pt;
border-style:double;
border-color:#8C0088;
font-size:15pt}
NAME
{ font-style:italic;
text-align:center}
TIME
{ display:block;
text-align:center;
font-weight:bold;
border-top:double;
border-color:#8C0088}
EXAM
{ display:block;
text-align:center;
```

border-top:double;

border-color:#8C0088}

PAGES

{ display:block;

text-align:center;

border-top:double;

border-color:#8C0088}

PASSING

{ display:block;

text-align:center;

border-top:double;

border-color:#8C0088}

FACULTY

{ display:block;

margin:50px 100px;

border-style:solid}

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 12
«Отображение XML-документов различными способами»

Цель: научиться работать с отображение XML-документов различными способами.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

У27. применять инструменты подготовки тестовых данных;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: на основе прошлой лабораторной работы составьте валидный XML-документ, используя 2 способа: с помощью DTD, а также XMLсхемы.

Порядок выполнения работы:

Валидные XML документы

Валидный XML документ не то же самое, что и синтаксически верный XML документ.

Первое правило для валидного XML документа то, что он должен быть синтаксически верным.

Второе правило – валидный XML документ должен соответствовать определенному типу документов.

Правила, определяющие допустимые элементы и атрибуты для XML документа, называются определениями документа или схемами документа.

С XML можно использовать различные типы определений документа:

- оригинальное определение типа документа (DTD)
- более новый тип определений, основанный на XML, – XML схема.

XML DTD

Цель DTD состоит в том, чтобы определить структуру XML документа. Это делается путем определения списка допустимых элементов.

В приведенном ниже примере

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
```

<note>

<to>Анна</to>

<from>Дмитрий</from>

<heading>Напоминание</heading>

<body>Не забудь обо мне в эти выходные!</body>

</note>

декларация DOCTYPE является ссылкой на внешний файл определений типов документа (DTD). Содержимое этого файла показано ниже.

<!DOCTYPE note [

<!ELEMENT note (to,from,heading,body)>

<!ELEMENT to (#PCDATA)>

<!ELEMENT from (#PCDATA)>

<!ELEMENT heading (#PCDATA)>

<!ELEMENT body (#PCDATA)>

]>

Приведенное выше DTD интерпретируется следующим образом:

- !DOCTYPE note определяет, что корневым элементом документа является *note;*
- !ELEMENT note определяет, что элемент *note* содержит четыре элемента: *to, from, heading, body;*
- !ELEMENT to определяет, что элемент *to* должен быть типа "#PCDATA";
- !ELEMENT from определяет, что элемент *from* должен быть типа "#PCDATA"
- !ELEMENT heading определяет, что элемент *heading* должен быть типа "#PCDATA"
- !ELEMENT body определяет, что элемент *body* должен быть типа "#PCDATA"

#PCDATA означает разбираемые текстовые данные.

С DTD XML файл может нести собственный формат и можно быть уверенным, что получаемые из внешних источников данные будут корректными.

Xml схема

Также, как и DTD, XML схемы описывают структуру XML документа. XML документ, прошедший проверку по XML схеме, является "синтаксически верным" и "валидным".

Пример XML схемы

<?xml version="1.0"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">

<xs:complexType>

<xs:sequence>

<xs:element name="to" type="xs:string"/>

<xs:element name="from" type="xs:string"/>

<xs:element name="heading" type="xs:string"/>

<xs:element name="body" type="xs:string"/>

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>

Приведенная выше схема интерпретируется следующим образом:

- *<xs:element name="note"*> определяет элемент "note";
- *<xs:complexType*>у элемента "note" комплексный тип;
- *<xs:sequence*>комплексный тип – эпоследовательность элементов;
- *<xs:element name="to" type="xs:string"*> у элемента "to" строковый тип (текст);
- *<xs:element name="from" type="xs:string"*> у элемента "from" строковый тип;
- *<xs:element name="heading" type="xs:string*"> у элемента "heading" строковый тип;
- *<xs:element name="body" type="xs:string*"> у элемента "body" строковый тип.

Как видно из примера, каждая XML схема состоит с корневого элемента «schema» и обязательного пространства имен «http://www.w3.org/2001/XMLSchema». Далее идет описание схемы и собственно сама схема. При этом очень часто в очень качественных схемах описание бывает куда большим, чем сама XML Schema.

XML схема мощнее DTD, так как:

- XML схема пишется на XML;
- XML схема легко расширяется;
- XML схема поддерживает типы данных;
- XML схема поддерживает пространства имен;
- XML схема поддерживает типы данных.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 13
«Разработка Web-приложения с помощью XML»

Цель: Разработать Web-приложение с помощью XML.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: Оформите задание лабораторной работы № 1 через подключение XSLT.

Порядок выполнения работы:

XSL является приложением XML, т. е. XSL-таблица представляет собой корректно сформированный XML-документ, который отвечает правилам XSL. Подобно любому XML-документу, XSL-таблица стилей содержит простой текст, и вы можете создать ее с помощью вашего любимого текстового редактора.

Связывание XSL-таблицы стилей с XML-документом. Вы можете связать XSL-таблицу стилей с XML-документом, включив в документ инструкцию по обработке xml-stylesheet, которая имеет следующую обобщенную форму записи:

<?xml-stylesheet type="text/xsl" href="3.xslt"?>

Таблица стилей при этом должна размещаться на том же домене, что и XML-документ, с которым вы ее связываете.

Если вы не связали XML-документ ни с CSS-таблицей, ни с XSL-таблицей стилей, Internet Explorer 5 отобразит документ с помощью встроенной XSL-таблицы, которая используется по умолчанию. Эта таблица стилей отображает исходный XML-текст в виде дерева с возможностью свертывания/развертывания уровней.

В отличие от CSS, содержащей правила, XSL-таблица стилей включает один или несколько шаблонов, каждый из которых содержит информацию для отображения в определенной ветви элементов в XML-документе.

Каждая XSL-таблица стилей должна иметь элемент Документ, известный как корневой элемент, является XML-элементом верхнего уровня, который содержит все остальные элементы.

<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheetversion="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

Элемент Документ xsl:stylesheet служит не только хранилищем других элементов, но также идентифицирует документ как XSL-таблицу стилей. Все XSL-элементы принадлежат пространству имен xsl – т. е. вы предваряете имя каждого XSL-элемента префиксом xsl:, обозначающим пространство имен.

Элемент Документ xsl:stylesheet XSL-таблицы стилей должен содержать один или несколько шаблонов элементов, которые для краткости будем называть шаблонами.

<xsl:template match="/">

<!-- дочерние элементы … -->

</xsl:template>

Браузер использует шаблон для отображения определенной ветви элементов в иерархии XML-документа, с которым вы связываете таблицу стилей. Атрибут match шаблона указывает на определенную ветвь.

Значение атрибута match носит название образца (pattern). Образец в данном примере ("/") представляет корневой элемент всего XML-документа.

Каждая XSL-таблица стилей должна содержать один и только один шаблон с атрибутом match, который имеет значение "/".

Корневой образец ("/") не представляет элемент Документ (или корневой элемент) XML-документа. Он представляет весь документ, для которого элемент Документ является дочерним. (Т. е. он аналогичен корневому узлу Document в объектной модели документа DOM)

Пример использования XSL-таблицы стилей для предоставления информации в виде таблицы:

<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<body>

<h2> Список специальностей факультета ИТ</h2>

<table border="1">

<tr bgcolor="#9acd32">

<th style="text-align:center">Специальность</th>

<th style="text-align:center">Срок обучения</th>

<th style="text-align:center">Предметы ЦТ</th>

<th style="text-align:center">План набора</th>

```
<th style="text-align:center">Проходной балл</th>
</tr>
<xsl:for-each select="FACULTY/SPECIALIZATION">
<tr>
<td><xsl:value-of select="NAME"></td>
<td><xsl:value-of select="TIME"></td>
<td><xsl:value-of select="EXAM"></td>
<td><xsl:value-of select="PAGES"></td>
<td><xsl:value-of select="PASSING"></td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Оператор пути в значении атрибута select относится к текущемуэлементу. Каждый контекст внутри XSL-таблицы стилей относится к текущему элементу. Если вы опустите атрибут select для XSL-элемента value-of, элемент будет осуществлять вывод текстового содержимого плюс текстовое содержимое всех дочерних элементов в текущий элемент.

Порядок элементов value-of в шаблоне определяет порядок, в котором браузер отображает эти элементы. Таким образом, даже из этой простой таблицы стилей вы можете понять, что XSL-таблица стилей является гораздо более гибкой, чем CSS, которая всегда отображает элементы в том порядке, в котором они следуют в документе.

Элемент **for-each**выполняет две основные задачи:

- осуществляет вывод блока элементов, содержащихся внутри элемента for-each, повторяя его для каждого XML-элемента в документе, отвечающего образцу, присвоенному атрибуту select элемента for-each;
- внутри элемента for-each задает текущий элемент, устанавливаемый атрибутом select элемента for-each.

Не нужно включать в XSL-шаблон элементы, представляющие элементы HTML или BODY, которые являются стандартными составными частями HTML-страницы, поскольку браузер сам эффективно их формирует.

Каждый из элементов, представляющих HTML-разметку, должен быть корректно сформированным XML-элементом, а также стандартным HTML-элементом. Не забывайте, что XSL-таблица стилей является XML-документом.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 14
«Использование фреймворка для создания сайта»

Цель: получить практические разработки модулей приложений с помощью Фреймворка CodeIgniter.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

У12. использовать открытые библиотеки (framework);

У13. использовать выбранную среду программирования и средства системы управления базами данных;

У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений

Материальное обеспечение:

ПК, текстовый редактор, браузер

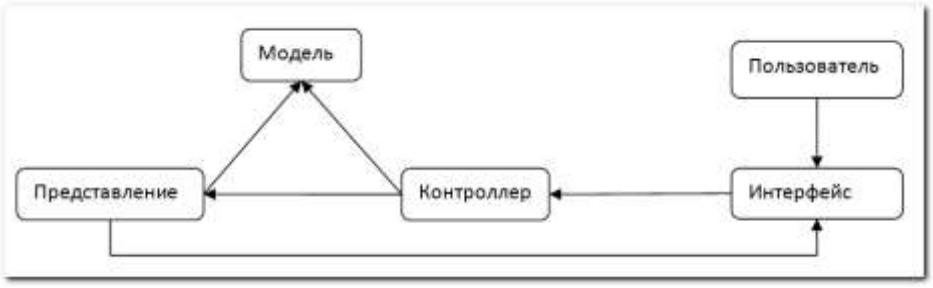
Задание:

1. Ознакомиться с теоретическим материалом.
2. Создать приложение согласно варианту (использовать элементы из таблицы 1 первого задания)
3. Разработать структуру приложения
4. Создать скрипт на php для ввода и выдачи результата, используя MVC
5. Оформить отчет согласно требованиям.

Краткие теоретические сведения

MVC Фреймворк

Model-view-controller (Паттерн модель-представление-контроллер) используется очень давно. Еще в 1979 году его описал Тригве Реенскауг в своей работе «Разработка приложений на Smalltalk-80: как использовать Модель-представление-контроллер». С тех пор паттерн зарекомендовал себя как очень удачная архитектура программного обеспечения.



Пользователь, работая с интерфейсом, управляет контроллером, который перехватывает действия пользователя. Далее контроллер уведомляет модель о действиях пользователя, тем самым изменяя состояние модели. Контроллер также уведомляет представление. Представление, используя текущее состояние модели, строит пользовательский интерфейс.Основой паттерна является отделение модели данных приложения, его логики и представления данных друг от друга. Таким образом, следуя правилу «разделяй и властвуй», удается строить стройное программное обеспечение, в котором, во-первых, модель не зависит от представления и логики, а во-вторых, пользовательский интерфейс надежно отделен от управляющей логики.

На данный момент паттерн MVC реализован в том или ином виде для большинства языков программирования используемых для разработки web-приложений. Самое большое количество реализаций имеет PHP, но и для Java, Perl, Python, Ruby есть свои варианты.

«M» или модель – часть MVC-системы, которая отвечает за запросы к базе данных (или другому внешнему источнику) и предоставление информации контроллеру. Можно было бы загружать необходимую модель в зависимости от запроса, но возможно немного стереть границы между моделью и контроллером, т.е. контроллер работает с БД непосредственно через библиотеку взаимодействия с БД, нежели чем через отдельную модель.

Codeigniter

CodeIgniter — открытый фреймворк написанный на PHP для разработки полноценных веб-систем и приложений. Разработан компанией EllisLab, а также Риком Эллисом (Rick Ellis) и Полом Бурдиком (Paul Burdick).

Фреймворк – это готовый каркас для приложений, которые будут строиться на его основе. В этот каркас включены наиболее часто используемые библиотеки. Можно при написании каждого нового приложения изобретать велосипед с распределением его основных модулей, структурой директорий, классами обработки основных компонентов и т.п., а можно воспользоваться готовым универсальным решением.

CodeIgniter является инструментарием для тех, кто строит веб-приложения на PHP. Его цель в том, чтобы позволить вам разрабатывать приложения быстрее, чем если бы вы писали код с нуля, предоставляя богатый набор библиотек для часто используемых задач, а также простой интерфейс и логическую структуру для доступа к этим библиотекам. CodeIgniter позволяет

творчески сосредоточиться на ваших проектах, используя минимальный объема кода, необходимый для той или иной задачи.

CodeIgniter обладает рядом значительных плюсов перед другими веб-фреймворками, например:

- используется модель MVC (Модель-Отображение-Контроллер), хорошо зарекомендовавшая себя при разработке приложений самой разной направленности;
- поддерживается множество баз данных (MySQL, PostgreSQL, MSSQL, SQLite, Oracle);
- отлично написанная документация с примерами позволит быстро освоить фреймворк;
- CodeIgniter очень быстр в работе. Его считают эталоном скорости генерации страниц.

Порядок выполнения работы:

Создание контроллера welcome.php:

```
<?php
class Welcome extends Controller {
function Welcome()
{
parent::Controller();
$this->load->model('User_model', 'users'); // указание подключаемой модели
}
function index()
{
$this->load->view(welcome_message'); // указание файла отображения приветствия codeigniter
}
function say_hello($id=1) {
$data['name'] = $this->users->name($id);
$this->load->view('hello', $data); // указание файла отображения views и данных
}
}
```

Создание модели user_model.php:

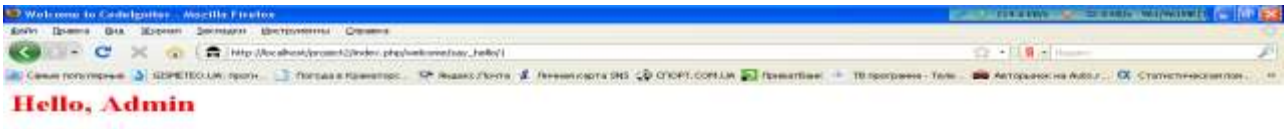
```
<?php
class User_model extends Model {
function name($key) {
```

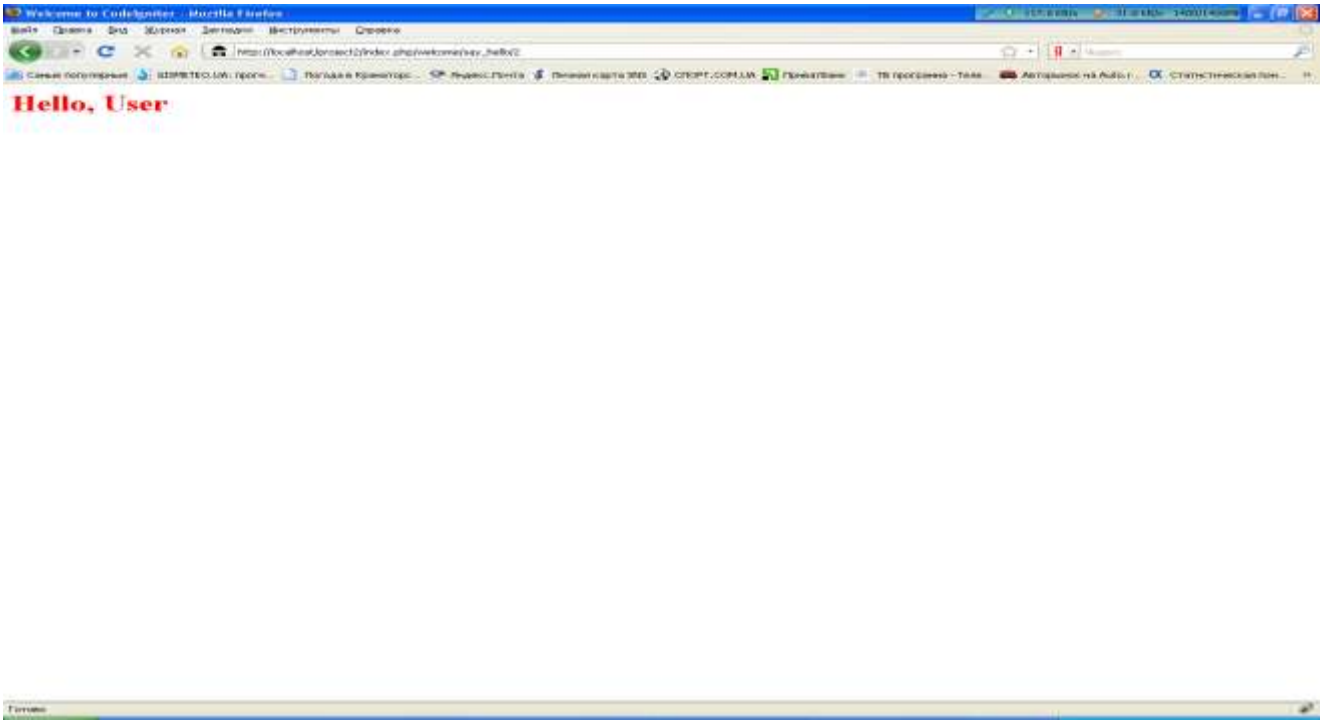
```
$names = $this->names();
return $names[$key];
}
function names() {
return array(1 => "Admin", 2 => "User"); // в зависимости от переданного параметра будет
} вывод имени соответствующего пользователя
}
?>
```

Создание view hello.php:

```
<html>
<head>
<title>Welcome to CodeIgniter</title> </head>
<body>
<h1><font color=red> Hello, <?=$name?></h1>
</body>
</html>
```

Запуск приложения:





Задание к работе:

Отчет должен содержать:

1. Название и цель работы.
2. Ход работы с детальным описанием выполненных действий с рисунками, листингом кода.
3. Экранные формы браузера с загруженными страницами.
4. Выводы о проделанной работе.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 15
«Создание сайта на CMS»

Цель: Создание интернет магазина.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

У12. использовать открытые библиотеки (framework);

У13. использовать выбранную среду программирования и средства системы управления базами данных;

У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: создать сайт, представленный ниже.

Краткие теоретические сведения:

Преимущества WordPress

Почему именно WordPress? Назовем семь преимуществ Вордпресс перед другими аналогичными конструкторами:

Бесплатность.

Гибкость: с помощью дополнительных модулей вы можете «собрать» уникальный сайт под ваши запросы.

Огромное количество шаблонов (тем), что позволит создать свой, неповторимый внешний вид.

Развитое сообщество и большой объем информации (в сети) по работе с конструктором.

Распределение ролей пользователей (можно разделить обязанности по поддержке сайта).

Не требуется никаких знаний из области программирования.

Это одна из несложных CMS, доступная для понимания даже новичку.

Порядок выполнения работы:

Домен и хостинг

WordPress бесплатен, но вам нужно будет оплатить хостинг – специальную услугу по предоставлению места на сервере (там, где физически будет находиться ваш проект). Сервер это что-то вроде компьютера, но он устроен немного не так, как наш домашний любимец, да и программы там установлены также специфические. Кроме хостинга потребуется еще доменное имя – адрес вашего сайта в сети интернет. Конечно, можно найти бесплатные хостинг и домен, но это подойдет, скорее, для тренировки, нежели чем для крупного проекта.

Несколько советов по выбору доменного имени:

Домен должен быть короткий, легко запоминающийся; в то же время он должен быть “профильным”.

Домен лучше регистрировать на латинице, поэтому избегайте в домене “сложных” звуков, таких как “ж”, “ш”, “щ” и т.д.

Установка WordPress

Если с доменом и хостингом придется немного повозиться, то с установкой сайта никаких проблем возникнуть не должно. Обычно на (нормальных) хостингах имеется автоинсталлятор. Вы заходите в данный раздел, выбираете из списка “WordPress”, записываете email, придумываете логин-пароль – и вуаля! – через пару минут ваш новый сайт установлен.

Логин и пароль запишите! Они вам еще пригодятся для входа в админ-панель вашего сайта.

Тут важно упомянуть, что WordPress состоит из двух частей – фронтенд (Front End) – та часть, что видят пользователи и бэкенд (Back End) или админ-панель – та часть, с которой работает администратор.

Чтобы перейти в админ-панель допишите к вашему сайту /wp-admin.

Получиться так: названиевашегосайта.ru/wp-admin

Пример: www.eduneo.ru/wp-admin

На этой странице записываем логин-пароль, которые указывали при регистрации и вы попадаете в пульт управления вашим блогом/сайтом. Обратите внимание, что консоль Вордпресс может быть свернута или развернута (управляется специальной кнопкой внизу).

Основные компоненты WordPress

Разберемся с основными компонентами WordPress:

Посты (записи) – относительно короткие, динамические (часто обновляемые) статьи; основа сайта (блога) – именно такие посты.

Страницы – статические (редко обновляемые) статьи; «О нас», «Контакты», «Схема проезда» – примеры таких статей.

Рубрики предназначены для группировки записей.

Метки (Теги) – еще один инструмент таксономизации, важный для создания перелинковки и связанных записей.

Виджеты – независимые блоки контента, которые можно разместить в специально предусмотренных областях шаблона.

Плагины – дополнительные модули, расширяющие базовый функционал WordPress.

Плагин Rus to Lat

После установки WordPress познакомимся с плагинами. Нам понадобиться плагин “Rus to Lat” который транскрибирует кириллицу в латиницу. Считается, что это необходимо для SEO – поисковой оптимизации и создания человекопонятных урлов (ЧПУ или friendly URL).

URL (Uniform Resource Locator) — единообразный локатор (определитель местонахождения) ресурса в интернете. Он является общепринятым стандартом во всемирной паутине и во всех браузерах. Другими словами – это всем знакомая ссылка в адресной строке на источник в интернете: страницу, сайт, запись и пр.

С помощью Rus to Lat ссылки в адресной строке на страницы вашего сайта станут понятными. Вот пример такой ссылки: https://www.eduneo.ru/uchebnyj-krossvord-kak-i-gde-sostavit/

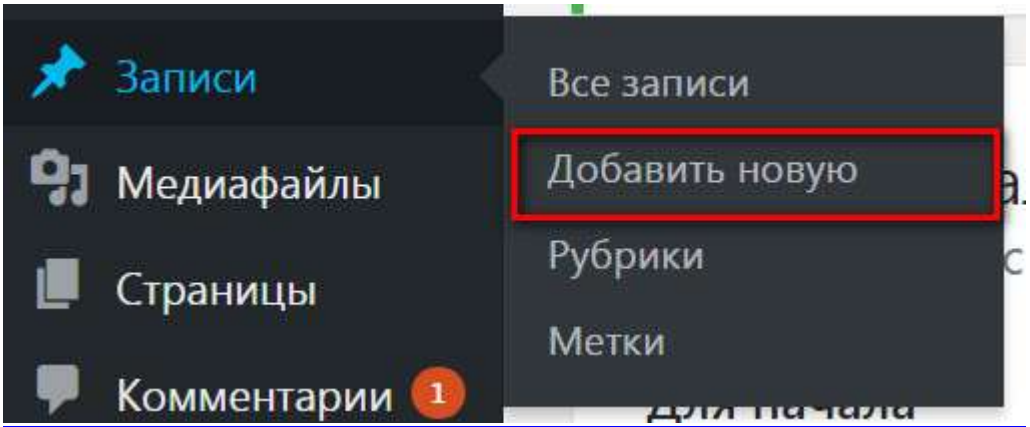
Все догадались как называется статья?

Для установки плагина выбираем вкладку “Плагины” (левое меню), “Добавить новый”. Мы можем воспользоваться поиском, чтобы найти и установить плагин, либо загрузить его с нашего компьютера. Первый способ предпочтительнее. После установки плагина активируем его (или сразу или на странице плагинов).

Для настройки “Rus to Lat” переходим в “Настройки”(левое меню), “Постоянные ссылки” и выбираем “Название записи”.

Как добавить первую запись на сайт

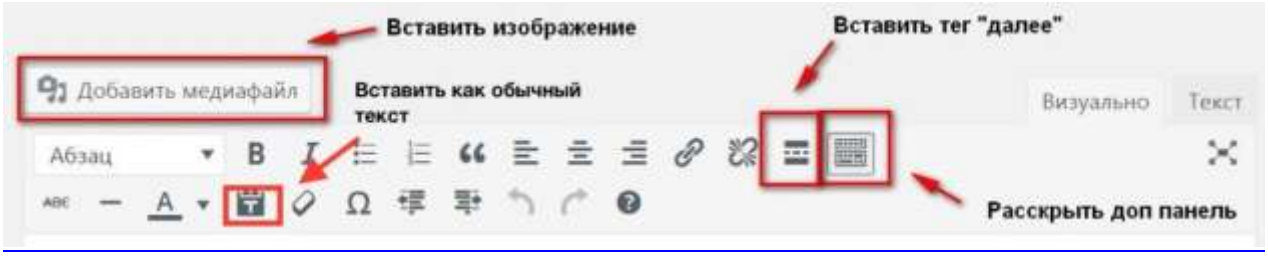
Теперь мы готовы к добавлению записей и страниц. Что добавить пост (запись) выберите на консоли пункт “Записи” – “Добавить новую”. Или перейдите на страницу «Все записи» и там нажмите кнопку “Добавить новую”.



Создание записи мало чем отличается от работы в офисных приложениях: записываем заголовок и набираем текст. Также можно вставить уже набранный текст из вашего текстового редактора.

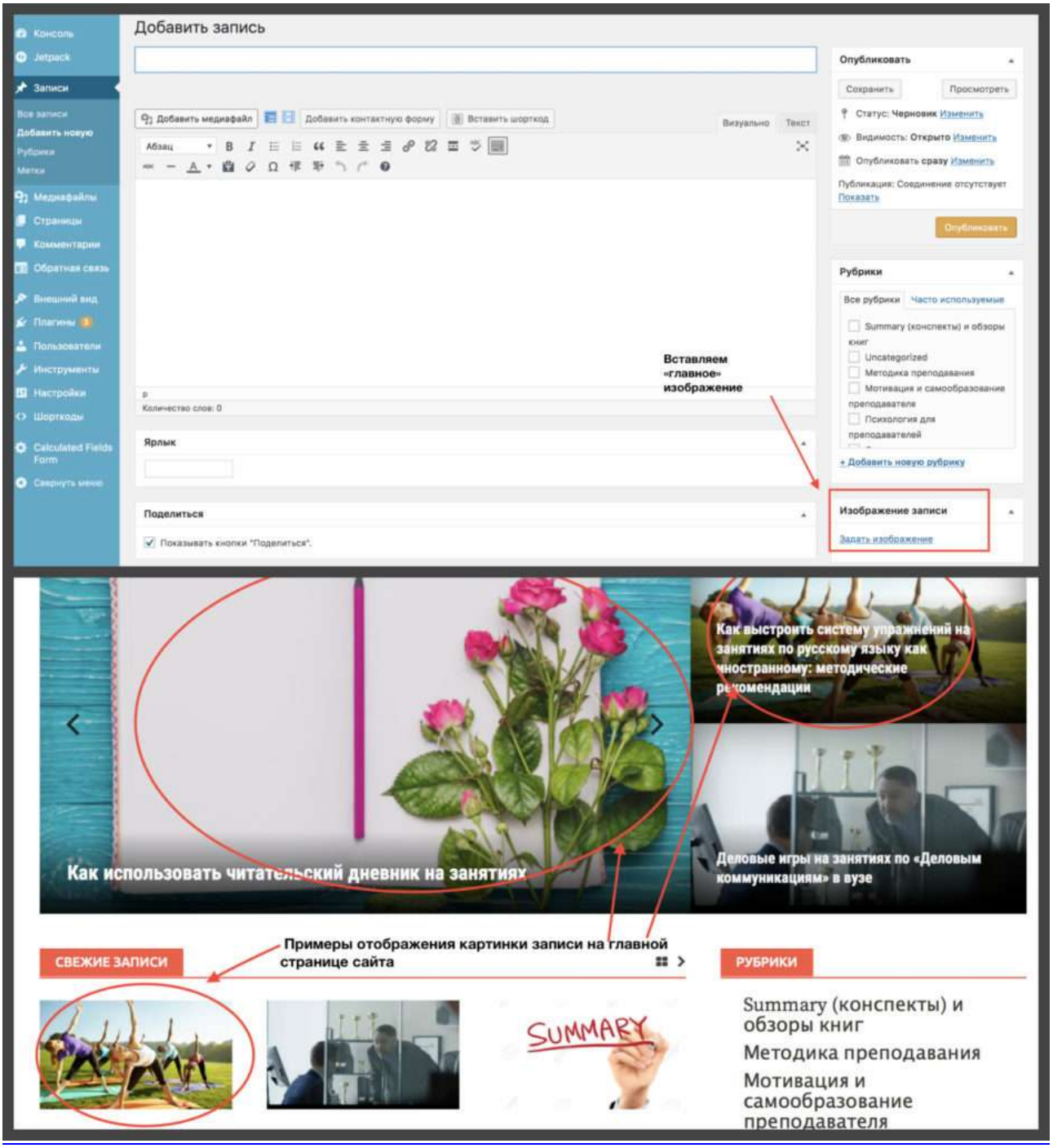
При вставке лучше использовать команду “Вставить как обычный текст” – так мы сможем избавиться от навязчивых стилей Word и использовать стиль текста, предлагаемый выбранной темой. Это поможет добиться единообразия и красоты на сайте (все тексты будут использовать определенный шрифт).

Если запись большая используйте кнопку Вставить тег “далее” – он отделяет анонс на главной странице от основного тела записи. Чтобы добавить изображение нажмите кнопку “Вставить медиафайл” и выберите вкладку “Загрузить файлы”. После загрузки справа в настройках отображения файла выберите выравнивание слева, размер и нажмите “Вставить в запись”.



Оформление записи

К записи можно добавить главное изображение. Оно будет отображаться в начале записи и на главной странице сайта. В правой колонке админ-панели зайдите в раздел “Изображение записи” и выберите “Задать изображение”. Загрузите необходимую картинку.



Записи и рубрики

Далее записываем метки, указываем (или добавляем новую) рубрику. Обратите внимание, что рубрику “Без рубрики” (Uncategorized) удалить нельзя – просто игнорируйте ее.

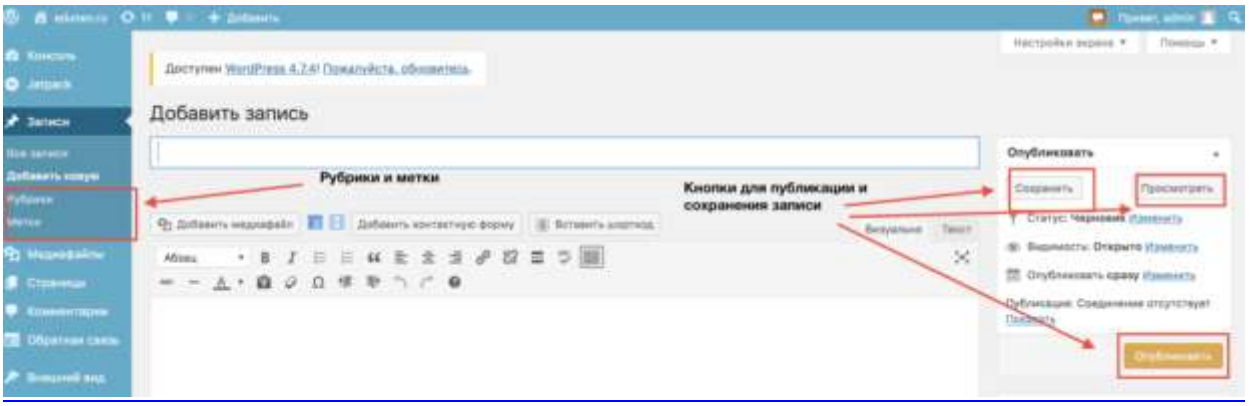
Запомните! Рубрики позволяют группировать записи. Рубрик должно быть существенно меньше чем записей, по сути, это крупные разделы вашего сайта. Поэтому не плодите их в большом числе.

Например, на сайте EduNeo.ru всего 6 рубрик: методика преподавания; технологии; мотивация и самообразование; свежие записи; психология для преподавателей и summary.

Публикация записи

Для публикации записи на всеобщее обозрение нажмите “Опубликовать”(правое меню, верхний угол) – наша первая запись опубликована!

Внимание! Кнопка “Сохранить” просто сохраняет ваши записи.“Просмотреть” позволяет посмотреть как будет выглядеть ваша запись перед публикацией. И только кнопка “Опубликовать”делает вашу запись доступной для пользователей.



Аналогичным образом мы можем опубликовать страницу. “Рубрики” и “Метки” находятся в пункте меню “Записи”.

Основные разделы админ-панели

Вкратце рассмотрим остальные параметры админ-панели.

Вкладка “Консоль” содержит быстрый доступ к основным параметрам сайта, а также показывает информацию об обновлении WordPress и плагинов. Вордперсс обновляется автоматически – просто нажмите соответствующую кнопку.

“Медиафайлы” предназначены для управления изображениями.

С помощью вкладки “Внешний вид” мы можем устанавливать новые темы (шаблоны), а также управлять виджетами. О том как это делать, мы поговорим в следующий раз.

Вкладка “Пользователи” содержит информацию о зарегистрированных пользователях. В частности, вы можете здесь изменить свой профиль (email, пароль и т.д.).

“Инструменты” не содержат важной информации – можете игнорировать это пункт меню.

В меню “Настройка” содержится много важных параметров сайта. Во вкладке “Общие настройки” вы можете изменить название и описание вашего сайта, а также изменить параметры регистрации новых пользователей.

“Написание” не содержит значимой информации, эту вкладку можно игнорировать.

С помощью “Чтения” мы можем изменить отображение главной страницы (записи или закрепленная страница). Вид главной страницы определяется целями сайта и особенностями шаблона WordPress.

“Обсуждение” управляет действиями с комментариями. Это достаточно нетривиальная тема, она связана с защитой от спам-ботов и требует отдельного рассмотрения. К сожалению, штатные средства Вордпресс плохо справляются со спамом.

Во вкладке “Медиафайлы” мы можем задавать размеры изображений для миниатюр, среднего и крупного размера.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 16
«Администрирование сайта»

Цель: получение навыков администрирования сайтов с использованием систем управления контентом на примере CMS WordPress.

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

У12. использовать открытые библиотеки (framework);

У14. осуществлять взаимодействие клиентской и серверной частей веб-приложений

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание: получение навыков администрирования сайтов с использованием систем управления контентом на примере CMS WordPress.

Задание выполняется с использованием пакета программ Денвер:

1. Скачайте установщик WordPress с сайта wordpress.org.
2. Распакуйте его в папку web-сервера Denwer (z:\home\localhost\Ваше название).
3. Создайте БД MySQL.
4. Наберите в браузере адрес сайта (localhost/Ваше название). Через **Параметры-Постоянные ссылки** включите ЧПУ.
5. Смените установленный по умолчанию шаблон. Настройте его.
6. Измените состав и расположение Виджетов.
7. Добавьте на сайт 2 страницы, записи в рубриках, рисунки.
8. Установите плагины, выполняющие следующие действия:
    - поддержка SEO;
    - создание кеша (например, Super Cache);
    - набор математических формул;
    - выполнение php;

- воспроизведение роликов YouTube;
- резервное копирование.

9. Добавьте на сайт записи, содержащие:
   - математический текст;
   - ролики YouTube;
   - исполняющийся код на php.
10. Добавьте код, выводящий содержимое некой таблицы из БД.

Форма представления результата:

Предоставить отчет, который будет включать в себя:

1. Цель работы.
2. Вариант задания.
3. Исходный текст программы на PHP.
4. Скриншоты сгенерированных страниц.
5. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.

МДК 09.01 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Лабораторная работа № 17
«Публикация сайта на бесплатном хостинге»

Цель: Познакомиться с сервисами размещения сайта в сети интернет. Научиться размещать сайт на бесплатном сервисе narod.ru

Выполнив работу, Вы будете:

уметь:

У1. разрабатывать программный код клиентской и серверной части веб-приложений;

У24. выполнять отладку и тестирование программного кода;

У10. оформлять код программы в соответствии со стандартом кодирования;

У16. использовать объектные модели веб-приложений и браузера;

У21. анализировать и решать типовые запросы заказчиков;

У31. выбирать хостинг в соответствии с параметрами веб-приложения;

У32. составлять сравнительную характеристику хостингов;

Материальное обеспечение:

ПК, текстовый редактор, браузер

Задание:

1. Найти административные интерфейсы коммуникационного и сетевого оборудования (видеокамеры, коммутаторы ЛВС, домашние Wi-Fi маршрутизаторы, и т.д.), подключенные к сети Интернет.
2. Известно, что адрес веб-интерфейса системы VMWare Horizon View HTML Access содержит строку portal/webclient/views/mainUI.html. Найти такие системы, доступные из сети Интернет.
3. Оценить количество коммутаторов Cisco Catalyst с админи- стративным веб-интерфейсом, подключенным к сети Интернет.

Порядок выполнения работы:

1. Введение

Знакомство с Интернетом часто начинают с поисковых систем. Не забывает новый Интернет-пользователь совершать экскурсии в мир WWW, где он встречается со множеством сайтов. Проходит какое-то время, и этот посетитель хочет принять участие в наполнении Интернета. Чаще всего это выражается в его желании разместить на просторах Интернета свой

сайт, то есть найти место в Интернете для его размещения. Это место – земля, -- на которой будет стоять ваш дом(сайт), называется **хостинг.**

Хостинг бывает двух типов: **бесплатный** и **платный**. Чтобы выяснить, какой лучше, попытаемся сравнить оба типа.

Платный хостинг	**Бесплатный хостинг**
Выдаётся своё доменное имя, например: ProstoSite.ru	Сайт будет называться так: ProstoSite.NAROD.ru
Надёжный сервер	Разные «неполадки» с сервером
Невозможно удаление сайта клиента из базы данных	Удаление сайта клиента из базы данных без объяснения причины
Оперативная служба поддержки	Медленная служба поддержки
Поддержка технологий PHP, MySQL и т.д.	Данные технологии поддерживаются не всегда

Преимущество платного хостинга перед бесплатным очевидно, но нет смысла тратить деньги на платный хостинг, когда сайт только в начале пути, поэтому поговорим о бесплатном хостинге.

Подобных хостингов существует множество. Немного позже мы рассмотрим все достоинства и недостатки основных из них. Но сначала о мышеловке, которая скрыта за этой «бесплатностью». Подвох заключается в следующем: при размещении на бесплатном хостинге сайт будет автоматически использован как рекламная площадка для этого хостинга.

Выражаться эта реклама может разными способами. Например, при заходе на сайт появляется всплывающее окно с рекламой, либо на каждой странице сайта будет добавлен сверху большой баннер и т.д.

А бывает, что так называемый бесплатный хостинг оказывается бесплатным только на определённый срок, по прошествии которого придётся либо заплатить, либо ваш сайт удалят из базы данных.

Но, несмотря на всё это, бесплатный хостинг – великая вещь, потому что есть возможность размещения своего сайта совершенно бесплатно. А это значит, что сайт будет доступен для просмотра любому человеку, выходящему в Интернет.

Итак, рассмотрим несколько популярных и, по словам пользователей, надёжных бесплатных хостингов.

Самый популярный бесплатный хостинг – **www.narod.ru**. Этот хостинг предоставляет под сайт 100 Мбайт. Есть чат, гостевая книга и анкета. Имеется много готовых шаблонов. К

сожалению, не поддерживает никаких «наворотов» типа PHP или SSI. Из рекламы – только небольшое окошко в правом верхнем углу, да и то с крестиком. Имя для сайта **Название_вашего_сайта.narod.ru.** Хостинг неплохой, поэтому и популярный. Настолько популярный, что имя для сайта придётся придумывать довольно долго, так как. практически все имена уже заняты.

Следующий хостинг -- www.boom.ru. Место под сайт – 50 Мбайт. Есть готовые шаблоны. Нет ни PHP, ни SSI, вообще ничего нет. Имя для сайта **Название_вашего_сайта.boom.ru.** Главный недостаток – это навязчивая реклама в виде большого баннера в нижней части окна, который при прокрутке остаётся неподвижным.

Хостинг www.by.ru. Под сайт даётся неограниченное место. Есть гостевая книга. Поддерживает SSI. Не имеет службы поддержки, частые сбои на сервере. Доменное имя **Название_вашего_сайта.by.ru.**

Хостинг www.newmail.ru . Под сайт даёт 16 Мбайт. Есть форум. Предоставляет несколько доменных имён:

Название_вашего_сайта. newmail.ru.

Название_вашего_сайта.orc.ru.

Название_вашего_сайта.hotmail.ru.

Название_вашего_сайта.nm.ru.

Для более-менее серьёзных проектов не подходит, так как. не поддерживает ни одну из технологий. Рекламы нет. Конечно, это не все бесплатные хостинги, существуют и другие.

Практическая часть выполнения работы.

Часть 1.

1. Используя программу WebSiteX5 создайте сайт, посвященный красотам родного края (о городе Улан-Удэ или Республике Бурятия или об озере Байкал или Тункинской долине и т.д.)

Внимание! При создании сайта указывайте доменное имя, полученное при регистрации на хостинге Yandex.Диск.

Например вы зарегистрировались на Yandex.Диске под логином name_2012@yandex.ru. Тогда доменное имя Вашего сайта будет следующим *name_2012.narod.ru* или *name_2012.narod2.ru*. Вот это доменное имя и используйте при создании сайта в программе WebSiteX5Free.

Часть 2.

Размещение сайта на хостинге www.narod.ru.

Внимание! Данный пункт используйте, если Вы не регистрировались в сервисе Яндекс.Диск или Яндекс-почта. Если у вас есть аккаунт на Яндекс, то просто войдите в него.

2.1. . Регистрация на хостинге:

Зайдём на сайт хостинга www.narod.ru.

Если же вы не зарегестрированы на Яндекс почте по прейдите по адресу Narod.ru. На главной странице в левом верхнем углу есть ссылка **Регистрация,** после щёлчка по которой, заполнить предлагаемую форму. В форме необходимо указать свой логин, пароль для доступа, фамилию, имя, e-mail, а также имя для сайта., которое нужно будет ввести в соответствующее поле латинскими буквами. **2. Переход по ссылке:**

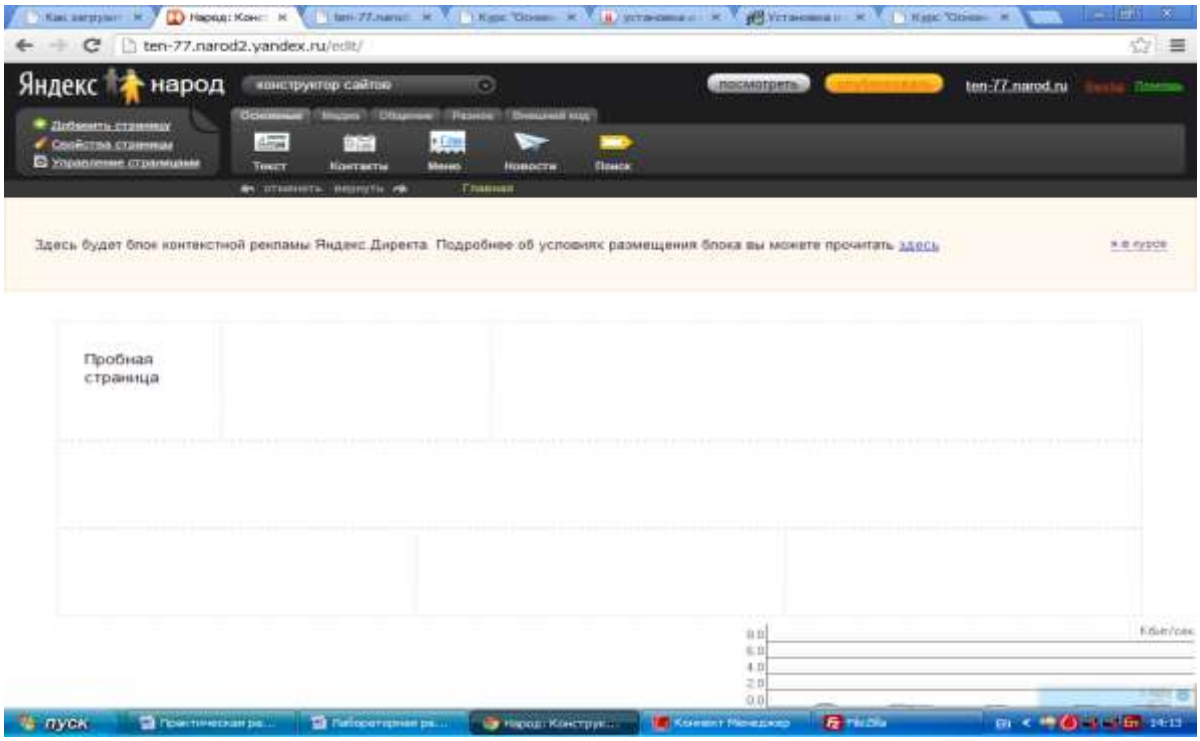
После завершения регистрации обычно высылается письмо на электронный почтовый ящик, который был указан. В письме повторно указывается логин и пароль и даётся ссылка на панель управления вашего сайта. Зайти по ссылке.

3. Загрузка файлов на хостинг:

На открывшейся странице необходимо найти ссылку **Создать сайт**. Перейти по данной ссылке на следующую страницу. На этой странице выберите стиль вашего сайта (выбирайте любой, т.к в дальнейшем мы установим свой сайт) и нажимаете **Далее**. Во вновь открывшейся странице выбираем структуру сайта и нажимаем **Далее**. На следующем шаге вводим любой текст и нажимает кнопку О**публиковать** (в левом верхнем углу окна**).**

Вот теперь мы переходим к самому главном- публикации Вашего сайта (созданного ранее в программе WebSiteX5Free).

В верху окна (в режиме конструктора) раскройте список



Выберите пункт **Управление файлами.**

В открывшейся странице создайте папки, такие же, как и у Вас на диске

С:.

Для этого используйте кнопку **Новая папка.**

Для загрузки файлов выберите пункт **Загрузить файл.** В появившемся окне нажимаете кнопку Выбрать файл и в открывшемся окне выбираете необходимый файл и нажимаете **ОК.**

Внимание! Загружайте файлы в те папки на хостинге, что и папки в которых они находились на диске С:

Файлы находящиеся вне

папок

загружаете непосредственно на страницу **Главная.**

5. Отладка работы сайта

Просмотрите результаты своей работы нажав кнопку **Опубликовать**

Форма представления результата:

Предоставить отчет, который будет включать в себя:

6. Цель работы.
7. Вариант задания.
8. Исходный текст программы на PHP.

9. Скриншоты сгенерированных страниц.
10. Выводы.

Критерии оценки:

Оценка "5" ставится: вся работа выполнена безошибочно и нет исправлений;

Оценка "4" ставится: допущены 1-2 вычислительные ошибки.

Оценка "3" ставится: допущены ошибки в ходе решения задачи при правильном выполнении всех остальных заданий или допущены 3-4 вычислительные ошибки, при этом ход анализа должен быть верным.

Оценка "2" ставится: допущены ошибки в ходе решения задачи и хотя бы одна вычислительная ошибка или при анализе и примеров допущено более 5 вычислительных ошибок.