

Учреждение образования
БЕЛОРУССКАЯ ГОСУДАРСТВЕННАЯ АКАДЕМИЯ СВЯЗИ
Кафедра ПОСТ

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
ПО ДИСЦИПЛИНЕ
«ПРОГРАММИРОВАНИЕ ДЛЯ ИНТЕРНЕТ
для учащихся дневной формы получения образования
по специальности
2-40 01 31 – Тестирование программного обеспечения

Составитель: Лущик Н. И.

Утвержден на заседании кафедры ПОСТ
протокол № 1 от 30.08.2018 г.

Зав. кафедрой ПОСТ  Рыбак В.А.

Минск,
2018 г.

СОДЕРЖАНИЕ

Лабораторная работа №1 Верстка макета web-страницы	3
Лабораторная работа №2 Создание XML-документов.....	7
Лабораторная работа №3 Создание сайта на основе CMS.....	12
Лабораторная работа №4 Обработка событий в Java Script.....	19
Лабораторная работа №5 Взаимодействие с элементами формы	23
Лабораторная работа №6 Обработка массивов в Java Script	27
Лабораторная работа №7 Создание функциональных элементов web-страницы	30
Лабораторная работа №8 Создание web-страниц на PHP	34
Лабораторная работа №9 Получение и обработка данных из пользовательских форм.....	38
Лабораторная работа №10 Отправка HTTP заголовков. Сессий и cookies	40
Лабораторная работа №11 Средства языка PHP для работы с СУБД MySQL.....	45
Лабораторная работа №12 Работа с графикой в PHP	50
Лабораторная работа №13 Объектно-ориентированное программирование в PHP.....	52
Лабораторная работа №14 Взаимодействие сервлета и jsp-страниц	55
Лабораторная работа №15 Создание web-страниц с использованием сессий	62
Лабораторная работа №16 Взаимодействие с библиотекой JSTL.....	67
Лабораторная работа №17 Создание web-страниц	77
Лабораторная работа №18 Взаимодействие с xml-документом.....	84
Лабораторная работа №19 Средства для работы с БД	93
Лабораторная работа №20 Современные технологии продвижения web-проекта.....	100

Лабораторная работа №1

Верстка макета web-страницы

Цель работы: закрепить знания и навыки необходимые для разработки макета web-страницы.

Теоретические сведения

Версткой веб-страниц называют создание такого HTML-кода, который позволяет размещать элементы веб-страницы (изображения, текст, линии и т.д.) в нужных местах документа и отображать их в окне браузера согласно разработанному макету. При этом следует принимать во внимание ограничения присущие HTML и CSS, учитывать особенности браузеров и знать приемы верстки, которые дают желаемый результат.

Табличная верстка

Таблицы довольно долго властвовали в области верстки, поскольку предлагали достаточно простые методы для размещения разных элементов на веб-странице при отсутствии явных конкурентов. Благодаря наличию большого числа параметров, особенно границе нулевой толщины, таблица выступает в роли невидимой модульной сетки, относительно которой добавляется текст, изображения и другие элементы. Удобство и широкие возможности верстки – вот основной реверанс в пользу таблиц.

Таблицы удачно подходят для «резинового» макета, ширина которого привязана к ширине окна браузера. Благодаря тому, что размер таблицы можно задавать в процентах, она занимает все отведенное ей свободное пространство. Также можно регулировать и высоту содержимого.

Рисунки часто разрезают на отдельные фрагменты, а затем собирают их вновь в одно целое, выкидывая одни фрагменты или заменяя их другими изображениями. Это требуется для различных дизайнерских изысков вроде создания эффекта перекатывания, анимации или уменьшения объема файлов. Таблицы позволяют легко обеспечить «склейку» нескольких рисунков в одно изображение. Каждая картинка помещается в определенную ячейку, параметры таблицы при этом устанавливаются такими, чтобы не возникло стыков между отдельными ячейками.

Браузеры достаточно вольно толкуют некоторые параметры CSS, поэтому создание универсального кода с применением слоев может стать настоящей головной болью для разработчиков. В этом смысле таблицы отображаются в разных браузерах практически одинаково, поэтому создание веб-страниц упрощается.

Недостатки таблиц:

1. Долгая загрузка.

Особенность таблиц такова, что пока последнее слово в самом низу таблицы не загрузится, на экране содержимое ячеек отображаться не будет. Браузеры используют такой подход, чтобы получить всю информацию о таблице для правильного форматирования ее содержимого. Но если таблица велика по высоте, может пройти достаточно много времени, прежде чем мы увидим нужную информацию. Существуют и способы обхода этого свойства, в частности, разбиение одной большой таблицы на несколько таблиц поменьше, а также использование стилевого свойства table-layout.

2. Громоздкий код.

Таблицы содержат сложную иерархическую структуру вложенных тегов, которая увеличивает объем кода, и повышает сложность изменения отдельных параметров. В некоторых случаях для достижения желаемого результата приходится вкладывать одну таблицу внутрь другой.

3. Плохая индексация поисковиками.

За счет того, что текст располагается в отдельных ячейках таблицы, в коде он может находиться достаточно далеко друг от друга. Такая раздробленность информации, а также значительная вложенность тегов затрудняет правильное индексирование страницы поисковыми системами. Как результат документ не попадает в первую десятку выдачи запроса по ключевым словам.

4. Нет разделения содержимого и оформления.

В идеале HTML-код должен содержать только теги с указанием стилевого класса или идентификатора. А все оформление вроде цвета текста и положения элемента выносится в CSS и модифицируется отдельно. Такое разделение позволяет независимо править код страницы и менять вид отдельных ее элементов. Хотя к таблицам стиль легко добавляется, но обилие «лишних» тегов не позволяет действительно просто и удобно управлять видом отдельных компонентов страницы.

5. Несоответствие стандартам.

В последнее время стандарты HTML и CSS прочно засели в умах веб-разработчиков. Этому способствует развитие XHTML и XML, которые более «жестко» относятся к коду документа, появление новых версий браузеров, придерживающихся спецификации, и мода на верстку слоями. Таблицы в первую и последнюю очередь нужны для размещения табличных данных.

Пример использования табличной верстки для макета, представленного на рисунке 1.

```
<!DOCTYPE html>
<html>
<head>
  <title>Табличная вёрстка</title>
</head>
<body>
<table border="1" cellpadding="0" cellspacing="0" width="100%">
<tr>
<th colspan=2>шапка сайта (логотип, слоган, телефон)</th>
</tr>
<tr>
<th width="20%">навигация</th>
<th width="80%">заголовок</th>
</tr>
<tr>
<td width="20%">
<ul>
<li><a href="index.html" title="Ссылка 1">Ссылка 1</a></li>
<li><a href="index.html" title="Ссылка 2">Ссылка 2</a></li>
<li><a href="index.html" title="Ссылка 3">Ссылка 3</a></li>
</ul>
</td>
<td width="80%">контент</td>
</tr>
<tr>
<td colspan=2>Низ сайта (баннеры, счетчики, информация)</td>
</tr>
</table>
</body>
</html>
```

шапка сайта (логотип, слоган, телефон)	
навигация	заголовок
<ul style="list-style-type: none">• Ссылка 1• Ссылка 2• Ссылка 3	контент
Низ сайта (баннеры, счетчики, информация)	

Рисунок 1 – Макет использования табличной верстки

Верстка с помощью слоев (блоков)

Слои в большинстве случаев являются независимыми друг от друга, за счет чего они как отдельные блоки могут добавляться или удаляться в макете веб-страницы. За такое поведение верстка с помощью слоев получила название «блочная верстка». Слои допустимо вкладывать один в другой для формирования желаемого декоративного элемента. Поэтому под именем «блок» подразумевается не столько отдельный слой, сколько их совокупность.

По умолчанию содержимое контейнеров <DIV> на веб-странице располагаются по вертикали, вначале идет один слой, ниже располагается следующий и т.д. При создании колонок требуется располагать слои рядом по горизонтали, для чего применяется несколько методов. Одним из распространенных является использование стилевого параметра float. Хотя он предназначен для создания обтекания вокруг элемента, с тем же успехом float устанавливает и колонки. Но здесь следует учесть одну особенность. При уменьшении окна браузера до некоторой критической ширины, колонки перестают располагаться горизонтально и перестраиваются друг под другом по вертикали.

По умолчанию ширина блока вычисляется автоматически и занимает все доступное пространство. Например, если тег <DIV> в коде документа присутствует один, то он занимает всю свободную ширину окна браузера и ширина блока будет равна 100%. Стоит поместить один тег <DIV> внутрь другого, как ширина внутреннего тега начинает исчисляться относительно его родителя, т.е. внешнего контейнера.

Некоторые браузеры достаточно свободно трактуют понятие ширины, хотя в спецификации CSS четко указано, что ширина складывается из суммы следующих параметров: ширины самого блока (width), отступов (margin), полей (padding) и границ (border).

Пример 1. CSS для блочной верстке:

```
<style type="text/css">
DIV {
width: 300px; /* Ширина слоя */
margin: 7px; /* Значение отступов */
padding: 10px; /* Поля вокруг текста */
border: 4px solid black; /* Параметры границы */
background: #fc0; /* Цвет фона */
}
</style>
```

Примере 2. Создание трех слоев, ширина которых определяется в процентах:

```
<html>
<head>
<title>Ширина</title>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">
<style type="text/css">
#layer1 {
width: 100%; /* Ширина первого слоя */
padding: 10px; /* Поля вокруг текста */
background: #fc0; /* Цвет фона */
}
#layer2 {
width: 100%; /* Ширина второго слоя */
background: #cc0; /* Цвет фона */
}
#layer2 {
padding: 10px; /* Поля вокруг параграфа */
}
#layer3 {
background: #3ca; /* Цвет фона третьего слоя */
padding: 10px; /* Поля вокруг текста */
}
}
```

```

</style>
</head>
<body>
<div      id="layer1">Lorem      ipsum      dolor      sit      amet...</div>
<div      id="layer2"><p>Lorem      ipsum      dolor      sit      amet...</p></div>
<div      id="layer3">Lorem      ipsum      dolor      sit      amet...</div>
</body>
</html>

```

Цвет фона элемента проще всего устанавливать через универсальный параметр background. Фоном при этом заливается область, которая определяется значениями атрибутов width, height и padding.

Порядок выполнения

1. Ознакомиться с теоретическими сведениями.
2. Выполнить задание для самостоятельной работы.
3. Оформить отчет, содержащий результаты выполнения (скриншоты страниц, содержание html- и css-страниц) индивидуальное задание и ответы на контрольные вопросы.

Задание для самостоятельной работы

Создать сайт на основе HTML-кода, состоящего из 3-5 страниц, имеющих различные функциональные возможности (содержать элементы: кнопка, поле для ввода, раскрывающейся список и т.д.). Оформление стилем осуществить с помощью CSS. Наполнить сайт информацией согласно варианту.

Вариант 1: сайт учреждения образования.

Вариант 2: сайт-витрина магазина одежды.

Вариант 3: сайт спортивных новостей.

Вариант 4: блог о правильном питании.

Вариант 5: сайт достопримечательностей Республики Беларусь.

Вариант 6: каталог учреждений образования в Республике Беларусь.

Вариант 7: визитка ботанического сада.

Вариант 8: книжный интернет-магазин.

Вариант 9: блог о рыбалке.

Вариант 10: сайт политических новостей.

Контрольные вопросы

1. В чем схожесть и отличия HTML-документа и CSS-файла?
2. Способы взаимодействия CSS-стилей и HTML-документа?
3. Охарактеризуйте виды версток web-страниц.

Лабораторная работа №2

Создание XML-документов

Цель работы: закрепить знания и навыки необходимые для разработки xml-документа, изучить и применить на практике теоретические сведения об анализаторе типа DTD.

Теоретические сведения

DTD – это язык описания, который позволяет определить, какие элементы должны быть в XML-документе, сколько раз они должны повторяться, какие атрибуты должны быть у этих элементов, какие атрибуты обязательные и какие не обязательные, а также какие сущности могут использоваться в документе.

Если говорить кратко, то DTD в XML используется для проверки грамматики документа и соответствия его стандарту. Это позволяет парсеру (обработчику) на этапе обработки определить, соответствует ли документ нашим требованиям. То есть, проходит валидация XML-документа.

Необходимость проверки грамматики XML-документов заключается в следующем:

- XML-документ может быть предназначен не для требуемой системы;
- XML-документ может содержать неправильные данные;
- XML-документ может содержать ошибки в структуре.

Недостатки DTD:

- отличный от xml синтаксис языка. это вызывает множество проблем, таких как, например, проблемы с кодировкой или невозможность отслеживать ошибки;
- нет проверки типов данных. в dtd есть только один тип – строка;
- в dtd нет пространств имен. нельзя поставить в соответствие документу два и более dtd описаний.

Для объявления элементов, атрибутов и сущностей в DTD используются специальные декларации и модификаторы.

Определение элемента XML и последовательности элементов XML:

<!ELEMENT название элемента (что может содержать)>

Пример:

<!ELEMENT book (title, author, price, description)>

Элемент book содержит по одному элементу title, author, price и description.

Альтернативы элементов:

<!ELEMENT название элемента (элемент1, элемент2, (элемент3 | элемент4 | элемент5))>

Пример:

<!ELEMENT pricelist (title, price, (author | company | sample))>

Элемент pricelist содержит элементы title, price и один элемент из трех на выбор – author, company либо sample.

Объявление атрибута:

<!ATTLIST элемент
атрибут 1 CDATA #REQUIRED
атрибут 2 CDATA #IMPLIED
>

Пример:

<!ATTLIST pricelist
id CDATA #REQUIRED
name CDATA #IMPLIED
>

Элемент pricelist может содержать два атрибута – атрибут id и атрибут name. При этом атрибут id является обязательным, так как указано #REQUIRED, а атрибут name – не обязательным (указано #IMPLIED). В свою очередь CDATA указывает обработчику, что разбирать содержимое атрибутов не нужно.

Определение сущностей:

<!ENTITY сущность "что подставлять">

Пример:

```
<!ENTITY myname "Надежда Ивановна">
```

Если встретится сущность «&myname;», то вместо нее автоматически подставится «Надежда Ивановна».

Модификаторы (объясняют повторения элементов):

* – ноль или много,

? – ноль или один,

+ – один или много.

Пример:

```
<!ELEMENT books (book+)>
```

Элемент books может содержать один или более элементов book.

Создание DTD-файла для валидации XML-документа на примере прайс-листа книг. XML-документ выглядит следующим образом.

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<pricelist>
```

```
<book>
```

```
<title>Книга 1</title>
```

```
<author>&myname;</author>
```

```
<price>Цена 1</price>
```

```
<description>Описание</description>
```

```
</book>
```

```
</pricelist>
```

В данном примере корневой элемент pricelist содержит вложенные элементы book. Внутри элементов book находятся элементы title, author, price и description, которые могут содержать какие-то текстовые данные.

Для валидации данного прайс-листа можно использовать DTD-документ следующего содержания.

```
<!ELEMENT pricelist (book+)>
```

```
<!ELEMENT book (title, author+, price, description?)>
```

```
<!ELEMENT title (#PCDATA)>
```

```
<!ELEMENT author (#PCDATA)>
```

```
<!ELEMENT price (#PCDATA)>
```

```
<!ELEMENT description (#PCDATA)>
```

```
<!ENTITY myname " Надежда Ивановна">
```

<!ELEMENT pricelist (book+)> – декларируется корневой элемент books и в скобках указывается, что он может содержать. В данном случае он может содержать один или более элементов book (+ означает один или более).

<!ELEMENT book (title, author+, price, description?)> – определяется элемент book. Элемент book может содержать один элемент title, один или более элементов author, один элемент price и один или ни одного элемента description.

<!ELEMENT title (#PCDATA)> – определяется элемент title. В качестве содержимого элемента указывается #PCDATA. Это означает, что анализатор обязан разбирать то, что находится внутри этого элемента.

Внешнее определение DTD – подключение DTD-документа

Суть данного метода состоит в том, чтобы подключить к XML-документу файл DTD при помощи следующей конструкции.

```
<!DOCTYPE DOCUMENT SYSTEM "file.dtd">
```

где DOCUMENT – указывается корневой элемент XML-документа, file.dtd – ссылка на файл DTD.

Пример:

XML-документ

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<!DOCTYPE pricelist SYSTEM "file.dtd">
```

```
<pricelist>
```

```
<book>
```

```
<title>Книга 1</title>
```

```
<author>Автор 1</author>
```



```
</book>
</pricelist>
DTD файл
<!ELEMENT pricelist (book+)>
<!ELEMENT book (title, author+, price, description?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```

Порядок выполнения

1. Ознакомиться с теоретическими сведениями;
2. Выполнить индивидуальное задание согласно варианту;
3. Оформить отчет, содержащий выполненное индивидуальное задание и ответы на контрольные вопросы.

Задание для самостоятельной работы

Создать внешнее определение типа документа (файл.dtd), которое определяло бы XML-совместимый формат для хранения данных, согласно варианту. Затем на основе созданного DTD-документа создайте файл.xml, который должен содержать в себе информацию согласно варианту и иметь описание не менее 6 элементов.

Вариант 1

Создать XML-документ типа телефонная книга, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Record;
- у каждого элемента Record должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 2

Создать XML-документ типа студенческая группа, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Student;
- у каждого элемента Student должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 3

Создать XML-документ типа комната, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Element;
- у каждого элемента Element должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 4

Создать XML-документ расписание, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Lesson;
- у каждого элемента Lesson должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 5

Создать XML-документ типа аптека, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Drug;
- у каждого элемента Drug должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 6

Создайте XML-документ типа музыкальный альбом, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Song;
- у каждого элемента Song должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 7

Создайте XML-документ типа гербарий, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Herb;
- у каждого элемента Herb должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 8

Создайте XML-документ типа кулинарная книга, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Dish;
- у каждого элемента Dish должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 9

Создайте XML-документ типа гардероб, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом Thing;
- у каждого элемента Thing должен быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Вариант 10

Создайте XML-документ типа зоопарк, при этом обеспечьте выполнение следующих условий:

- каждая запись должна быть представлена элементом `Animal`;
- у каждого элемента `Animal` должны быть обязательные атрибуты, вложены обязательные и необязательные элементы. Некоторые элементы могут встречаться неограниченное количество раз, а другие только один раз;
- некоторые из элементов должны содержать смысловые атрибуты с заданными значениями.

Контрольные вопросы

1. В чем сходство и отличия HTML-документа с XML-документом?
2. Из чего состоит XML-документ?
3. Почему и для чего используется DTD?
4. Перечислите все составные части DTD-документа и их возможные значения.

Лабораторная работа №3

Создание сайта на основе CMS

Цель работы: приобрести навыки, необходимые для установки локального web-сервера, закрепить теоретические знания об установке CMS Joomla, настройке и созданию сайта на ее основе.

Теоретические сведения

Установка локального сервера

Сервер на базе Денвер-3 предназначен для создания собственных доменов на локальном компьютере, а также позволяет создавать сайты и тестировать их работоспособность, не имея доступа в Интернет.

Для установки программного продукта Денвер-3 (Джентльменский набор Web-разработчика) необходимо запустить установочный файл. Во время установки необходимо выбрать следующие опции:

- установку осуществить в папку C:\WebServer, а не в привычную папку C:\Program Files;
- выбрать вариант запуска вручную, а не как службу.

После установки Денвера его следует запустить вручную и проверить работоспособность. Для этого в браузере необходимо набрать адрес тестовой страницы <http://localhost/denwer/>. В случае успешной установки на экране появится страничка «Ура заработало!».

Эта программа помогает установить и настроить компоненты Web-сервера, необходимые для работы. Необходимо внимательно читать и отвечать на все вопросы, задаваемые программой. Можно прервать выполнение программы в любой момент, нажав Ctrl+Break.

После запуска установочного файла проверяется наличие необходимых драйверов, осуществляется поиск конфликтных файлов.

Далее необходимо указать имя директории, в которую будет установлен Денвер. Если устанавливать Денвер на флэш-накопитель, то удобнее всего указать здесь просто имя диска в качестве пути установки (без директории). В этом случае Денвер не "привязывается" к букве диска, и можно сразу же его использовать, просто вставив накопитель в любой компьютер.

Затем инсталлятор создает отдельный виртуальный диск, который необходим для функционирования всех компонентов системы. Отдельный диск сильно упрощает работу с Web-инструментарием, позволяя устроить на машине нечто вроде "маленького Unix".

Виртуальный диск - это просто синоним для одной из директорий на вашем диске. После того как он будет создан, вся работа с виртуальным диском будет в действительности происходить с указанной вами папкой. Чтобы создать диск, необходима утилита subst, входящая в Windows. При переходе на следующий шаг производится поиск утилиты subst.

Далее необходимо определиться с именем нового диска. Как оптимальный вариант предлагается диск Z: - маловероятно, что он уже занят. Впрочем, можно ввести и любую другую букву диска, который еще не занят. Указывать существующие диски запрещено.

Денвер может запускаться в двух режимах:

1. Виртуальный диск создается ПРИ ЗАГРУЗКЕ ОС. Запуск серверов осуществляется с помощью ярлыка на Рабочем столе. При завершении работы Денвера виртуальный диск НЕ отключается. Этот режим рекомендуется использовать, если вы собираетесь использовать виртуальный диск, не запуская серверов (например, хотите запускать Perl-скрипты не только из браузера, но и из командной строки).

2. При загрузке ОС виртуальный диск НЕ создается. На Рабочем столе также, как и в пункте 1, создаются ярлыки для запуска и останова серверов. При запуске серверов вначале создается виртуальный диск, после останова - диск отключается. Необходимо помнить, что в этом режиме при неактивном Денвере не будет доступа к виртуальному диску (в частности, к

Perl). Кроме того, некоторые версии Windows не умеют правильно отключать виртуальный диск (требуется перезагрузка).

При правильном выполнении всех пунктов получится результат, представленный на рисунке 1:

Денвер успешно установлен

Чтобы начать использовать Денвер, проделайте следующие действия:

1. Запустите Денвер, воспользовавшись ярлыком **Start Denwer** на Рабочем столе. Если вы не создавали ярлыки, то можно запустить Денвер по команде `C:\WebServers\denwer\Run.exe`.
2. Откройте браузер и перейдите по адресу <http://localhost>.
3. Вы должны увидеть главную страницу Денвера.
4. Если после запуска Денвера <http://localhost> не открывается, проверьте, не блокируется ли Денвер вашим антивирусом или фаерволом. Например, были замечены проблемы с NOD32 в Windows XP (в нем нужно добавить процесс `X:\usr\local\apache\bin\httpd.exe` в список исключений, это можно сделать в окне ИМОН/Настройка Разное/Исключение).



Внимание: если вы используете **Skype**, убедитесь, что он не занимает порты 80 и 443, необходимые для работы Apache в Денвере ("Инструменты - Настройки - Дополнительно - Соединение - Использовать порты 80 и 443 в качестве входящих альтернативных" должно быть отключено).

Если по каким-то причинам Денвер не заработал, свяжитесь, пожалуйста, с разработчиками: <http://forum.dklab.ru/denwer/bugs/>. Прикрепите к сообщению следующую информацию:

1. При каких условиях проявился баг? Что вы сделали перед тем, как его зафиксировали?
2. Точную версию Вашей OS (можно получить по команде `winver`, запущенной в Командной строке).
3. Файл `netstat.txt`, получившийся в результате работы команды `netstat -nb > C:\netstat.txt` (кстати, этот файл не содержит персональной информации или сведений, подрывающих безопасность системы, хотя на неуклюжий взгляд он и может показаться подозрительным).
4. Значимые сообщения из конца файла `/usr/local/apache/logs/error.log`.

Спасибо за использование Денвера!

Рисунок 1 – Результат выполнения всех действий

Установка CMS:

Joomla представляет собой бесплатную систему для создания веб-сайтов. Позволяет создавать веб-ресурсы разного уровня сложности: от сайта-визитки или личного блога до большого информационного портала или интернет-магазина. В своей работе CMS использует стандартный набор компонентов (PHP/MySQL) и может быть установлена даже на недорогой хостинг.

Сразу после инсталляции пользователь получает основу сайта с удобной административной панелью – остаётся только настроить его под свои нужды и заполнить контентом. Практически всё, что Joomla не может делать по умолчанию, решается путём установки расширений, которых под неё написано немало.

Необходимо распаковать архив с Joomla-файлами в папку с именем сайта по пути: имя_диска: \home\localhost\www\название_сайта.

Для этого при запущенном DENWER либо в адресной строке браузера необходимо ввести **http://localhost/название_сайта** и выполнить все шаги установки Joomla: выбор языка → начальная проверка → лицензия → конфигурация БД → конфигурация FTP → конфигурация сайта → завершение установки.

На шаге конфигурация БД необходимо заполнить поля согласно рисунку 2:

Рисунок 2 – Поля для заполнения

На шаге конфигурация FTP поля для заполнения оставить пустыми. На шаге завершение установки **ОБЯЗАТЕЛЬНО** удалить директорию «installation».

Теперь в верхнем правом углу страницы можно перейти на «Панель управления» для администрирования сайта либо на «Сайт» для предварительного просмотра (рисунок 3).

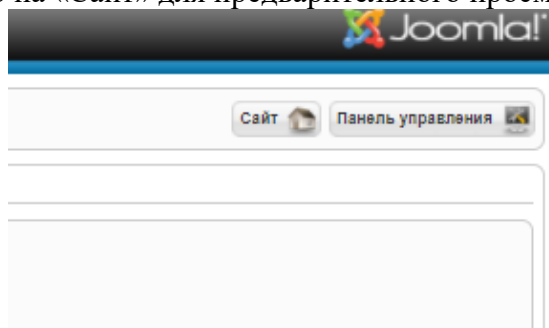


Рисунок 1 – Расположение кнопок

Joomla располагает мощной, функциональной, но при этом довольно понятной панелью администратора, с помощью которой можно менять дизайн и функциональность сайта, а также наполнять его контентом и настраивать через удобный графический интерфейс.

Чтобы попасть в панель управления движком, достаточно перейти по адресу вида **адрес_сайта/administrator/**, на отобразившейся странице авторизоваться (ввести указанные во время установки движка логин и пароль) и нажать кнопку **Войти**.

Для русификации «Панели управления» необходимо подключить расширению, перейдя на вкладку согласно рисунку 4 и выбрать файл-расширение.

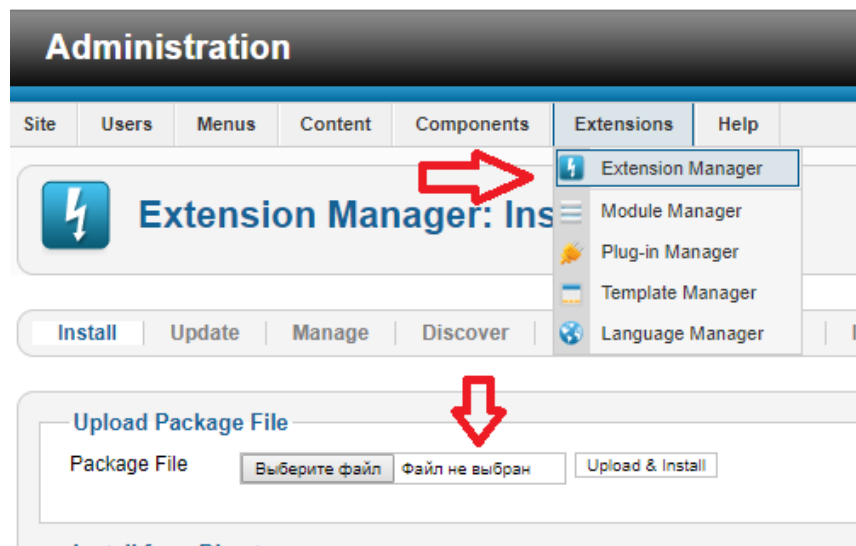


Рисунок 2 – Переход на вкладку для подключения расширения

Для включения файла-расширения необходимо перейти на вкладки «installed-site» «installed-administrator» и выбрать язык, перенеся звезду на язык по умолчанию (рисунок 5).

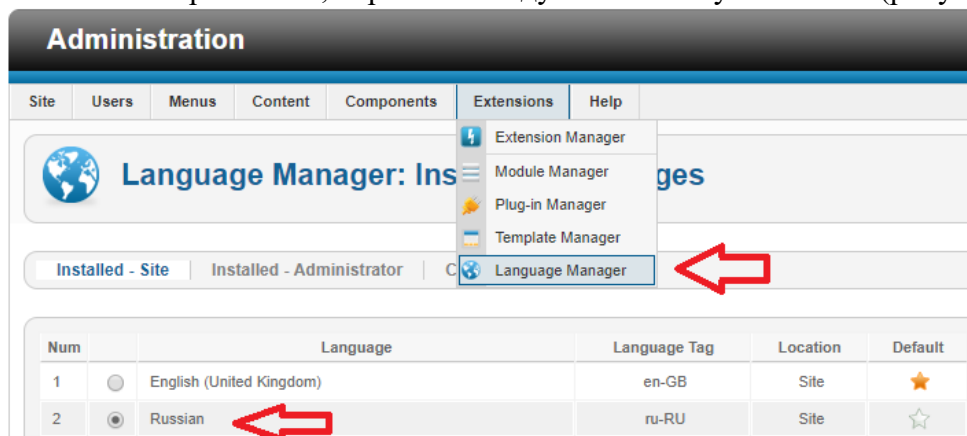


Рисунок 3 – Переключение языка

После этого содержимое сайта и панели управления будет отображено на русском языке. Затем необходимо установить следующие настройки:

- сайт → Общие настройки → Сайт → Настройки SEO и отключить ЧПУ;
- сайт → Общие настройки → Система → Настройки сессии и установить время жизни сессии 1000. Сохранить изменения.

Включить шаблон по умолчанию (рисунок 6).

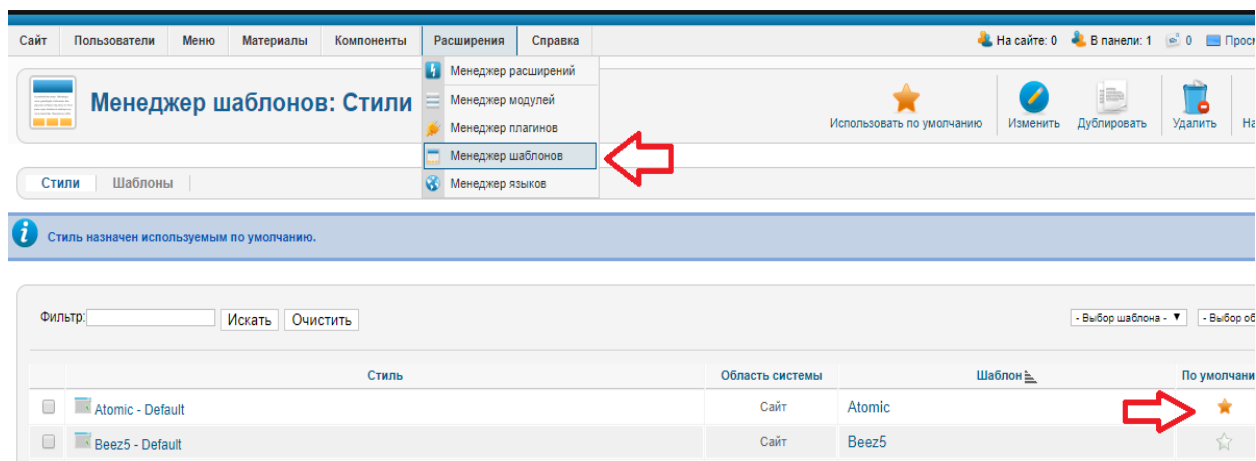


Рисунок 4 – Включение шаблона по умолчанию

Вкладка «Сайт» содержит следующие элементы:

- **панель управления.** Этот пункт позволяет перейти на её главную страницу. К сожалению, доступен он не всегда: если пункт Панель управления недоступен, то вы, вероятно, открыли одну из страниц редактирования, для выхода из которого достаточно нажать кнопку Закрыть;
- **общие настройки.** Обратите внимание, что они разбиты на вкладки. Здесь можно изменить заданное во время инсталляции Joomla имя ресурса, закрыть сайт на время выполнения технических работ (при этом настраивается видимое пользователям сообщение и картинка), убрать показ версии Joomla и настроить другие параметры;
- **обслуживание.** Позволяет разблокировать тот или иной материал, удалять временные файлы, создаваемые CMS для ускорения загрузки страниц сайта. Особенно рекомендуется очищать кэш после внесения серьёзных или многочисленных изменений. Также содержит сведения о Joomla и о сервере, на котором она работает;

Вкладка «Пользователи» (предназначено для работы с учётными записями) содержит следующие элементы:

- **менеджер пользователей.** Позволяет управлять отдельно взятыми учётными записями: активировать, блокировать, удалять и т. д. Есть функции пакетной обработки на случай, если регистраций было чересчур много. Подраздел Создать пользователя даёт возможность добавить учётную запись непосредственно из админ-панели, не регистрируя её на сайте стандартным способом;
- **группы.** Если предыдущий пункт обеспечивает индивидуальный подход, позволяя работать с отдельно взятым пользователем, то здесь работа ведётся с группами учётных записей, которых в Joomla насчитывается около десятка. Категории напрямую связаны с привилегиями. Больше всего прав имеет группа пользователей Super Users – они могут делать на сайте всё, что угодно. Самые бесправные – гости (Guest), они могут только просматривать сайт. Если существующих групп мало, можно создавать собственные;
- **уровни доступа.** Здесь можно управлять привилегиями пользователей. Разбиение на группы позволяет разрешать какие-то действия одним и запрещать их же другим. Уровнями доступа имеет смысл пользоваться на крупных сайтах, где работает множество людей, среди которых есть копирайтеры, администраторы, контент-менеджеры;
- **заметки о пользователях.** К каждой учётной записи можно прикрепить примечание, но эта функция целесообразна только для больших порталов с внушительным количеством регистраций;
- **категории заметок.** Примечания, как и пользователи, группируются, и это подменю позволяет категориями управлять, но на начальном этапе создания сайта функция не нужна;
- **массовая рассылка сообщений.** Позволяет оповестить о чём-либо сразу всех пользователей или отдельную их группу.

Вкладка «Материалы» служит для управления статьями, их категориями, а также медиа-контентом: картинки, звук, видео.

Вкладка «Компоненты» содержит следующие элементы:

- **баннеры.** Добавление рекламных баннеров на страницы ресурса и получение по ним исчерпывающей статистики в дальнейшем: просмотры, клики – всё, как полагается;
- **контакты.** Предназначен для создания страницы с формой обратной связи и, собственно, контактными данными владельца сайта;
- **ленты новостей.** С помощью этого компонента можно легко наполнить свой сайт чужим контентом, выводя новости из сторонних RSS-лент;
- **обновление Joomla.** Расширение, позволяющее устанавливать свежие версии движка. Заглядывать в него необязательно, так как сразу после нового релиза CMS информация об этом и кнопка обновления будут ждать вас на главной странице админки;
- **перенаправления.** Настройка переадресаций с одного сайта на другой;
- **поиск.** Познакомит вас с анализом поисковых запросов;
- **сообщения.** Предназначен для чтения и отправки приватных сообщений;
- **умный поиск.** Компонент, который помогает пользователю ввести запрос, дополняя незаконченное слово, как это делают «Яндекс» или Google. Использование компонента увеличивает нагрузку на сервер, потому как этот интеллектуальный поиск не работает без индексации страниц.

Вкладка «**Расширения**» (управление статьями, их категориями, а также медиа-контентом) содержит следующие элементы:

- **менеджер расширений.** Позволяет установить всё: язык, шаблон, компонент, плагин или модуль, а после инсталляции – всем этим управлять;
- **менеджер модулей.** Позволяет управлять видом расширений, которые выводятся в специально отведённых для этого местах темы оформления и внутри материалов;
- **менеджер плагинов.** Плагины расширяют функционал Joomla, а это подменю позволяет ими управлять;
- **менеджер шаблонов.** Отвечает за управление темами оформления движка;
- **менеджер языков.** Здесь можно увидеть, какие пакеты уже установлены, при необходимости загрузить новые и настроить языки.

Вкладка «**Справка**» содержит ссылки на справочный материал. К сожалению, большинство разделов этого меню не переведены на русский.

Установка шаблона: скачать архив темы оформления. Открыть архив и убедиться, что в его корне находится тема оформления. Войти в панель управления Joomla и открыть **Расширения** → **Менеджер расширений**. Сообщение проинформирует об успешной установке. Шаблон инсталлирован, осталось его активировать с помощью **Расширения** → **Менеджер шаблонов**. На появившейся странице выбрать «звёздочку» справа от имени шаблона, который необходимо установить по умолчанию. После появления сообщения «Стиль назначен используемым по умолчанию» (или что-то вроде этого), открыть сайт и посмотреть, как изменился его внешний вид. Если шаблон был установлен без демонстрационных данных, то все элементы, включая меню, модули, компоненты и контент.

Создание страницы: с помощью **Материалы** → **Менеджер материалов**. В поле **Заголовок** введите название статьи. В основное поле вставьте её текст. Joomla предлагает для этой цели функциональный и удобный визуальный редактор. Самое популярное действие, кроме правки текста – добавление в него картинок. Эта функция в расширенном виде доступна на вкладке **Изображения** и ссылки.

Создание пункта меню: откройте **Меню** -> **Менеджер меню**.

Порядок выполнения

1. Ознакомиться с теоретическими сведениями.
2. Установить локальный сервер и cms.
3. Выполнить задание для самостоятельной работы.
4. Оформить отчет, содержащий результаты выполнения (скриншоты страниц индивидуальное задание и ответы на контрольные вопросы).

Задание для самостоятельной работы

Создать сайта на основе CMS, состоящего из 4-6 страниц, имеющих различные функциональные возможности. Шаблон для сайта самостоятельно найти и установить. Наполнить контентом согласно варианту.

Вариант 1: достопримечательности Республики Беларусь.

Вариант 2: структура УО «БГАС».

Вариант 3: языки программирования.

Вариант 4: толковый словарь.

Вариант 5: кулинарная книга.

Вариант 6: столицы Олимпийских игр.

Вариант 7: областные города Республики Беларусь.

Вариант 8: реки Республики Беларусь.

Вариант 9: озера Республики Беларусь.

Вариант 10: наиболее популярные CMS.

Контрольные вопросы

1. В чем заключаются достоинства и недостатки использования cms?
2. Охарактеризуйте основные вкладки панели управления.

Лабораторная работа №4

Обработка событий в Java Script

Цель работы: научиться добавлять обработчики событий к html-тегам, описывать функции-обработчики событий, получать доступ и менять значения атрибутов тегов средствами JavaScript.

Теоретические сведения

Код на языке JavaScript помещается в теги `<script></script>` и выполняется интерпретатором языка JavaScript, встроенным в браузер.

Общий формат синтаксиса описания и прикрепления к тегу обработчика события:

```
<html><head>
  <script>
    function имя_функции(параметры) {
      тело функции
    }
  </script>
</head><body>
  <тег имя_события="имя_функции(параметры);">
</body></html>
```

Пример 1. По событию «нажать на кнопку» вывести окно с надписью «Hello!».

```
<html><head>
<script>
  function pressButton(text) {
    alert(text);
  }
</script>
</head><body>
  <INPUT TYPE="button" VALUE="Click" onClick = "pressButton('Hello!')">
</body></html>
```

Имя события `onClick`, момент его наступления – щелчок левой кнопкой мыши по данной кнопке. Обрабатывает событие функция `pressButon()`. В примере использована функция `alert(text)`, которая выводит окно и печатает текст, переданный в качестве параметра. Данная функция полезна при отладке скриптов для определения точки останова интерпретатора, а также для вывода текущего состояния переменных.

Для того чтобы изменилась html-страница, нужно изменить html-код, содержимое и параметры html-тегов. Средства языка JavaScript и позволяют это сделать. Для того чтобы было

понятно, с каким именно тегом будем работать в программном коде, тегам нужно присвоить названия. Это можно сделать следующим образом:

```
<тег атрибут=" значение_параметра" id="идентификатор_тега">
```

Таблица 1 – Присвоение тегам атрибутов названия

<pre><p>Do you know JavaScript?</p> <p>Yes.</p> </pre>	<pre><p id="question" >Do you know JavaScript?</p> <p id="answer">Yes.</p> </pre>
---	---

Доступ к атрибутам тега и его содержимому осуществляется (по таблице 1):

```
document.getElementById("идентификатор_тега").атрибут
```

т. е. `alert(document.getElementById("question").innerHTML);` выведет «Do you know JavaScript?»

```
alert(document.getElementById("picture").width);
```

 выведет 10.

Значение свойства `innerHTML` соответствует информационному содержимому тега, от открывающего до закрывающего дескриптора.

Изменение значений атрибутов тегов осуществляется:

```
document.getElementById("идентификатор_тега").атрибут=
новое_значение_атрибута;
```

После выполнения команды:

```
document.getElementById("answer").innerHTML="No. ";
```

страница изменится, и ответ на вопрос будет «No.»

Рассмотрим **пример** программного кода:

```
<html><head>
<script>
function getValue () {
alert(document.getElementById("testField").value);
}
function setValue (text) {
document.getElementById("testField").value=text;
}
</script></head><body>
<INPUT TYPE="text" id="testField" value="default text">
<INPUT TYPE="button" VALUE="Получить" onClick ="getValue()">
<INPUT TYPE="button" VALUE="Установить" onClick ="setValue('Hello!')">
```

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

Значение поля ввода хранится в атрибуте value тега <INPUT>. По нажатию на кнопку «Получить» будет выведено значение, находящееся в текстовом поле ввода, которое помечено идентификатором id="testField". По нажатию на кнопку «Установить» в текстовое поле ввода, помеченное идентификатором id="testField", будет помещено значение «Hello!».

Пример 2. Запросить у пользователя его имя и вывести на страницу в формате «Добро пожаловать, XXXX!»

```
<html><head>
<script>
function inputName() {
    var name;
    name=prompt("Введите имя","");
    document.getElementById("userName").innerHTML="Добро пожаловать, "+name+"!";
}
</script></head><body onload=" inputName()" >
<P id="userName"></P>
</body></html>
```

В данном примере после загрузки страницы, то есть по событию onload, вызывается функция inputName(). Для запроса имени пользователя используется специальное окно ввода prompt. Функция prompt() помещает введенное пользователем имя в переменную name. Следующая команда объединяет в одну строку фразу "Добро пожаловать" и имя, хранящееся в переменной name, и помещает ее в тег с идентификатором userName.

Порядок выполнения работы

1. Изучить теоретические сведения по теме: «Создание обработчиков событий на javascript».
2. Выполнить примеры, приведенные в теоретическом материале.
3. Создать веб-страницу и поместить на нее кнопку, по нажатию на которую выводится надпись «Я знаю javascript!».
4. Реализовать на веб-странице возможность формирования бланка заказа. Интерфейс приведен на рисунке 1.

№	Название	Цена за ед.	Кол-во	Сумма
1	Ручка	2 000	<input type="text"/>	
				Итого

Рисунок 1 – Вид бланка заказа

Пользователь вводит количество единиц товара, которое он хочет заказать, скрипт вычисляет сумму, которую нужно заплатить за каждый товар с учетом количества заказанных

единиц и итоговую сумму. Пересчет будет производиться после того как введено количество, т. е. по событию *onBlur* у тега *<INPUT>*.

5. Добавить возможность заказа карандашей стоимостью 1600.
6. Создать отчет, включив в него описание работы по пунктам 2–5, ответы на контрольные вопросы.

Контрольные вопросы

1. Каким образом добавить событие в html-тег?
2. В каком месте документа и каким образом описать функцию – обработчик события?
3. Как получить доступ к атрибутам тега?
4. Каким образом изменить значение атрибута тега?

Лабораторная работа №5

Взаимодействие с элементами формы

Цель работы: научиться создавать обработчики событий, которые получают данные из полей ввода html-форм, обрабатывают их с помощью условных операторов и выводят результат на html страницу.

Теоретические сведения

Доступ к элементам формы осуществляется:

document.имя_формы.имя_элемента.дескриптор

Для обеспечения более детальной обработки данных, введенных пользователем через формы, в JavaScript используются условные операторы.

Структура оператора if:

if (условие) {блок операторов 1 }

[else { блок операторов 2 }]

Пример 1. Создать обработчик, который будет определять, какое из двух введенных в поля ввода чисел больше.

```
<html><head>
<title>Оператор if</title>
<script>
function compare(){
if(parseInt(document.f.a.value)>parseInt(document.f.b.value)) {
document.getElementById("answer").innerHTML="А больше Б.";}
else if(parseInt(document.f.a.value) < parseInt(document.f.b.value)){
document.getElementById("answer").innerHTML="Б больше А.";}
else document.getElementById("answer").innerHTML="Значения равны.";
}
</script></head><body>
<form name="f"><table><tr>
<td>A:<br></td>
<td><input type="text" value="" size="5" name="a"></td>
<td>B:<br></td>
<td><input type="text" value="" size="5" name="b" ></td>
<td><input type="button" value="Сравнить" onclick="compare()"></td>
</tr></table></form>
<span id="answer"></span>
</body></html>
```

Поступающие через поля ввода html-формы значения имеют строковый тип данных. Для корректного выполнения математических операции их надо преобразовать к одному из числовых тип. Для этого используется функция parseInt(), которая преобразует аргумент к целочисленному типу.

Структура оператора выбора:

```
switch(выражение){
case "значение 1": операторы;break;
case "значение 2": операторы;break;
default: операторы;
}
```

Пример 2. Разработать обработчик, который в зависимости от выбранной поры года выводит ее продолжительность в днях.

```
<html><head>
```

```

<title>Оператор case</title>
<script>
function result(){
    switch(document.f.season.value){
        case "": answer.innerHTML="";break;
        case "0": answer.innerHTML="90 ";break;
        case "1":
        case "2": answer.innerHTML="92 ";break;
        case "3": answer.innerHTML="91 ";break;
    }
}
</script></head><body>
<form name="f"><table><tr>
    <td>Выберите пору года </td>
    <td><select name="season" onchange="result()">
        <option value="">--
        <option value="0">зима
        <option value="1">весна
        <option value="2">лето
        <option value="3">осень
    </select></td></tr></table>
</form>
<span id="answer" ></span>
</body></html>

```

Важной задачей в современном веб-программировании является проверка заполнения полей формы перед отправкой на сервер.

Создание HTML-макет формы:

```

<p id=msg>&nbsp;</p>
<form name="f1" action="" method="POST" onsubmit="return check();">
    Номер счета<input type="text" name="inputField" value=""><br>
    Вид операции <SELECT name="selectField">
        <OPTION value=0>все
        <OPTION value='1' >Значение_1
        <OPTION value='2' >Значение_2
    </SELECT>
    <input type="submit" name="button" value="Добавить">
</form>

```

Чтобы реализовать проверку данных, необходимо создать функцию check(), которая сработает по событию onsubmit и вернет либо true, если данные верны и их можно отправлять на сервер, либо false, в случае неправильного заполнения какого-либо поля, что будет означать остановку процесса передачи данных на сервер. Следует обратить внимание, что перед вызовом функции написано return.

```

<script>
function check(){
    if(document.f1.inputField.value.length==0){ document.f1.inputField.focus();
        alert("Поле ввода не заполнено!");
        return false;
    }
    var reg= /^[A-Z]\-\d{2,3}$/i;

```



```

if(!reg.test(document.f1.inputField.value)){ document.f1.inputField.focus();
    alert('Неправильный формат в поле ввода!');
    return false;
}
if(document.f1.selectField.value==0){ document.f1.selectField.focus();
    msg.innerHTML="Не выбрано значение!";
    return false;
}
return true;
}
</script>

```

Данные можно проверять на соответствие шаблону. Шаблон задается с помощью регулярных выражений.

Порядок выполнения работы

1. Изучить теоретические сведения по теме «взаимодействие с элементами формы».
2. Разобрать и выполнить примеры программ, приведенных в теоретическом материале.
3. Разработать скрипт для решения следующей задачи:

В спортивном клубе существуют четыре программы «вт. 18-00, пт. 18-00» – стоимость 300, «пн. 9-00, чт. 9-00» – стоимость 200, «пн. 18-00, чт. 18-00» – стоимость 300, «ср. 18-00, вс. 9-00» – стоимость 250, на которые клиенты могут покупать абонементы. Клиентам, имеющим дисконтную карту, предоставляется на абонемент скидка 10 %. Создать скрипт, позволяющий рассчитать стоимость и сформировать бланк заказа абонемента. Интерфейс пользователя приведен на рисунке 1.

Выбрать программу ▾	
Стоимость	XXXX
<input checked="" type="checkbox"/> Наличие дисконтной карты	
Размер скидки	XXXX
Итого	XXXX

Рисунок 1 – бланк заказа абонемента

Подсказка. Для того чтобы узнать установлен ли флаг checkbox, необходимо проанализировать свойство checked:

```

<script>
Function f(){
    if(document.f.testcheckbox.checked){alert("установлен");}
    else {alert("снят");}
}
</script>
<input type="checkbox" name="testcheckbox" checked onclick="f();">

```

модифицировать html-страницу и разработанный скрипт в связи с изменившимися условиями.

В спортивном клубе проводится рекламная акция: при покупке нескольких абонементов на второй и последующий абонементы предоставляется скидка 10 % (наличие скидки по дисконтной карте сохраняется). Интерфейс приведен на рисунке 2.

Выбрать программу	
Стоимость	XXXX
Количество абонементов	<input type="text" value="1"/>
<input checked="" type="checkbox"/> Наличие дисконтной карты	
Размер скидки	XXXX
Итого	XXXX

Рисунок 2 – бланк заказа абонемента на время проведения акции

Тестовый пример. Программа «вт. 18-00, пт. 18-00», кол-во абонементов – 2, дисконтная карта имеется. Размер скидки – 90. Итого: 510.

Программа «пн. 9-00, чт. 9-00», кол-во абонементов – 3, дисконтной карты нет. Размер скидки – 40. Итого: 560.

реализовать ограничение: клиент может заказать не более трех абонементов. При заказе более трех абонементов выводить сообщение об ошибке, представленное на рисунке 3.

Вы не можете заказать более 3-х абонементов!	
Количество абонементов	<input type="text" value="4"/>
(не более 3)	

Рисунок 3 – сообщение об ошибке

4. Создать отчет, включив в него описание работы по пунктам 2–3 и ответы на контрольные вопросы.

Контрольные вопросы

1. Каким образом извлечь данные из полей ввода?
2. Для чего используется функция `parseInt()`?
3. Какие операторы используются для реализации условных алгоритмов в javascript?
4. Каким образом проверить данные перед отправкой на сервер?

Лабораторная работа №6

Обработка массивов в Java Script

Цель работы: научиться создавать функции обработчики массивов данных, на основе параметров, введенных через поля html-форм, а также создавать элементы html-страниц на основе данных массивов.

Теоретические сведения

На практике на веб-страницах можно представлять данные, полученные из каких-либо систем. Например, CRM-система, на предприятии генерирует данные для представления на сайте и сохраняет их во внешнем файле. В зависимости от этих данных генерируется страница. Для хранения данных в скрипте в таких случаях используются массивы.

Задать массив можно следующим образом:

```
var имя_массива=[значение1,значение2 ...];
```

```
var x=[50,100,150,250];
```

Обращаться к элементам массива можно через индексные имена:

```
имя_массива[номер_элемента];
```

```
x[1];
```

Нумерация элементов начинается с 0.

Пример 1. Создать функциональный элемент, который будет показывать температуру воздуха по дням недели.

```
<html><head>
<title>Прогноз погоды </title>
<script>
function showTemperature(idSelect,idSpan){
var x=["",10,15,25];
document.getElementById(idSpan).innerHTML=x[idSelect.value];
}
</script></head><body >
  <select onChange="showTemperature(this,'temperature')">
    <option value="0">выберите день
    <option value="1"> сегодня
    <option value="2"> завтра
    <option value="3"> послезавтра
  </select>
  Температура воздуха <span id="temperature"></span>
</body></html>
```

В данном примере при выборе дня недели в элементе SELECT (по событию onChange) будет срабатывать функция showTemperature(), которая принимает два параметра идентификатор элемента выбора и идентификатор тега, в который необходимо выводить температуру. В функции описан массив X, элементы которого содержат значения температуры воздуха. Каждый элемент массива соответствует "опции" тега SELECT. Данная функция выводит соответствующий элемент массива X в тег с идентификатором id=temperature.

Проход по массиву организуется оператором цикла for. Синтаксис:

```
for(инициализация;условие выхода;изменение параметра цикла){
    операторы,                выполняемые                в                цикле
}
```

Пример 2. На основании данных массива X сгенерировать таблицу.

```
<html><head>
<title>Вывод элементов массива </title>
```

```

    </head><body>
<script>
var      x=[50,100,150,250];           //      описание      массива      X
document.write("<table border=1> <tr><td>i</td><td>X[i]</td></tr>"); // вывод заголовка
таблицы
for(i=0;i<x.length;i++){           //      вывод      строки      таблицы
document.write(           "<tr>      ><td>"+(i+1)+"</td><td>"+x[i]+"</td></tr>");
}//      конец      вывода      строки      таблицы
document.write("</table>");           //      вывод      окончания      таблицы
</script></body></html>

```

Пример 3. Найти сумму элементов массива больших числа Z. Четные ячейки выделить цветом.

```

<html><head>
<style type="text/css">
.even {/*стиль для четной ячейки*/
background-color: #FFFFFF
}
.odd {/*стиль для нечетной ячейки*/
background-color: #ccccff
}
</style>
<script>
function sumOfElemMoreZ(z){
    var S=0; //сумма
    var k=0; //счетчик элементов больших z
    for(i=0;i<x.length;i++){
        if(x[i]>parseInt(z)){//если нашли элемент больших z
            S=S+x[i]; //увеличиваем сумму
            k++; //увеличиваем счетчик
        }
    }
    //выводим результат
    document.getElementById("answer").innerHTML=
(k==0)?"Элементов больших Z в массиве нет!":"Сумма элементов массива больших Z:
"+S;
}
</script></head><body>
Массив: <br>
<script>
    var x=[50,100,150,250,300,350,300,250]; // описание массива X
    document.write("<table border=1><tr>"); // вывод заголовка таблицы
    for(i=0;i<x.length;i++){ // вывод строки таблицы
        document.write( "<td ");
        // определение четности ячейки и установка нужного стиля
        if (i%2) document.write(" class='odd' ");
        else document.write(" class='even' ");
        document.write(">"+x[i]+"</td>");
    } // конец вывода строки таблицы
    document.write("</tr></table>"); // вывод окончания таблицы
</script>
<br>Z: <input type="text" value="" size="4" id=z>
<input      type="button"      value="Посчитать"      onclick
="sumOfElemMoreZ(getElementById('z').value)"><br>
<span id="answer"></span></body></html>

```

Порядок выполнения работы

1. Изучить теоретические сведения по работе с массивами в javascript.
2. Разобрать и выполнить примеры программ, приведенных в теоретическом материале.
3. В массиве хранится прибыль фирмы по месяцам. Реализовать возможность вывода прибыли за выбранный месяц.
4. Создать функцию, определяющую количество месяцев, когда фирма работала в убыток.
5. Создать функцию, определяющую общую прибыль за год, на основании ее вывести сообщение о результате работы фирмы: прибыльная или убыточная. При выводе массива отрицательную прибыль выделить синим цветом, а положительную – красным.
6. Задан массив, содержащий названия предметов. Вывести значения в виде маркированного нумерованного списка.
7. В массиве pagesname содержатся названия html-страниц, в массиве pagesurl их URL (соответствие по индексу). Сформировать и вывести на экран меню.
8. Создать отчет, включив в него описание работы по пунктам 2–7, ответы на контрольные вопросы.

Контрольные вопросы

1. Каким образом можно объявить массив в языке javascript?
2. Пусть задан массив нечетной длины. Каким образом вывести на экран значение центрального элемента массива?

Лабораторная работа №7

Создание функциональных элементов web-страницы

Цель работы: научиться создавать функциональные элементы веб-страниц с использованием встроенных объектов, а также основанных на взаимодействии с CSS.

Теоретические сведения

В JavaScript существует ряд встроенных объектов: String – для обработки строк, Math содержит математические функции, Date – для работы с датой и временем.

В результате выполнения программного кода:

```
var a="Test text."; var n=Math.random();
```

```
alert(a.substring(0,Math.floor(n*10)));
```

будет выведена подстрока строки "Test text." случайной длины.

Создание функции, которая обеспечит работу часов на веб-странице:

```
function clock() { // часы
```

```
    date = new Date();//функция установки текущей даты
```

```
    hour = date.getHours();//определяем сколько часов
```

```
    minute = date.getMinutes();
```

```
    second = date.getSeconds();
```

```
    time = hour+":"+minute+":"+second;//формируем текущее время
```

```
    document.getElementById("fieldTime").value = time;
```

```
    setTimeout("clock();",1000);} 
```

На веб-странице функция будет срабатывать после загрузки страницы, то есть по событию onload в теге <body>:

```
<body onload="clock()"></body>
```

Текущее время устанавливается в тег, с идентификатором id=fieldTime, поэтому необходимо его включить в состав страницы.

```
<input type="text" size=8 id= "fieldTime" disabled />
```

Функция setTimeout("имя_функции();",кол-во миллисекунд) позволяет организовать бесконечный повтор событий: через заданный промежуток времени вызывается функция, имя которой указано первым параметром.

Создание слайд-шоу. Создается html-каркас. В тело документа помещается тег с идентификатором id=pict, в котором и будет происходить смена изображений. В теге <body> указывается, что по событию onload, т. е. после загрузки документа, надо вызвать функцию обработчик, которая и будет производить смену изображений. Для того чтобы можно было останавливать и запускать слайд-шоу, помещается на страницу кнопка с идентификатором

id=startStopButton, и указывается, что по нажатию на нее должна будет вызваться функция-обработчик startStopShow().

```
<body onload="show();">
<img src="" id="pict" width=20 height=20>
<input type="button" id="startStopButton" value="Стоп" onClick ="startStopShow()" />
</body>
```

Далее описывается функция show() и startStopShow():

```
var count=3, flag=1;
function show(){// показ слайдов
    if(flag==1)
        eval("document.getElementById('pict').src='images/"
+parseInt(count%4+1)+".gif'");
        count++;
setTimeout("show();",500);}
function startStopShow(){// кнопка запуска\остановки шоу
if (flag==1){ flag=0;
        document.getElementById("startStopButton").value="Старт";}
else{   flag=1;
        document.getElementById("startStopButton").value="Стоп";}}
```

Алгоритм включения/выключения шоу основан на следующем принципе. Если ее текущее значение переменной flag =1, то в функции show() происходит смена изображений, иначе процесс замены останавливается. Следовательно, в функции startStopShow() нужно соответственно менять значение переменной flag.

Средствами JavaScript можно изменить стилевое оформление тега. Для этого можно создать несколько разных классов и присваивать объекту нужный класс с помощью следующей команды:

```
document.getElementById("идентификатор_тега").className=имя;
```

Можно получить доступ и изменить CSS-свойство:

```
document.getElementById("идентиф_тега").style.свойство=значен;
```

Пример создания модельного ряда. Создается html-страницу, на которую помещаются изображения и блок, с идентификатором id= inf, в котором будет отображаться описание элемента. При загрузке страницы блок inf скрыт.

```
<body>

<div class="hide" id="inf">Выберите фигуру.</div>
</body>
```

Создаем стилевое оформление веб-страницы.

```
<style type="text/CSS">
img.p{
border:4px #fff solid;}
img.p:hover{
border:4px #fc0 solid;}
div.hide{// скрыт
visibility: hidden}
div.vis{//ВИДИМЫЙ
position: absolute;
left: 10px;
top: 60px;
width: 80px;
height: 40;
border: 6px #fc0 solid;
visibility: visible}
</style>
```

Классы vis и hide будут присваиваться тегу inf, в зависимости от того, каким он должен быть в данный момент: скрыт или видим. Далее создаются функции-обработчики, которые будут показывать и скрывать описание элемента в информационном блоке.

```
<script>
Information=["треугольник","квадрат"];
function over(i){ //функция показывает описание
document.getElementById("inf").className="vis";
document.getElementById("inf").innerHTML=Information[i];}
function out(){ //функция скрывает описание
document.getElementById("inf").innerHTML="Выберите фигуру.";
document.getElementById("inf").className="hide";}
</script>
```

На веб-странице часто встречается ситуация, когда по нажатию на кнопку открываются дополнительные блоки. Реализовать такого рода функциональную возможность можно путем установки значения свойства display.

```
<div>
<a href="javascript:displayElement('id1')" title="" >Объявление1</a>
<div id="id1" style="display: none">
    <p>Текст объявления 1</p>
</div>
```


</div>

```
function displayElement(id){  
document.getElementById(id).style.display =  
(document.getElementById(id).style.display=='none')?"block":"none";}
```

По нажатию на гиперссылку, срабатывает функция-обработчик displayElement(id), которая меняет значение свойства display.

Порядок выполнения работы

1. Разобрать и выполнить пример программы по созданию часов, приведенных в теоретическом материале.
2. Создать функциональный элемент, который будет выводить на экран приветствие в зависимости от времени суток на компьютере пользователя.
3. Разобрать и выполнить пример, реализующий слайд-шоу.
4. Создать слайд-шоу на тему «человек занимается физкультурой».
5. Разобрать и выполнить пример, реализующий модельный ряд.
6. Создать модельный ряд на заданную преподавателем тематику.
7. Разобрать и выполнить пример с выпадающими блоками.
8. Создать глоссарий на заданную преподавателем тематику.
9. Создать отчет, включив в него описание работы по пунктам 1–8, ответы на контрольные вопросы.

Контрольные вопросы

1. Какие средства для работы с датой и временем есть в javascript?
2. С помощью какого приема организовывается в javascript бесконечный цикл? Что произойдет, если для этих целей использовать оператор цикла while?
3. Как средствами javascript изменить стилевое оформление тега?
4. Какими средствами организовывается показ и сокрытие блоков или элементов веб-страницы?

Лабораторная работа №8

Создание web-страниц на PHP

Цель работы: научиться создавать веб-страницы на языке PHP, в том числе с использованием массивов и пользовательских функций.

Теоретические сведения

Программный код на языке PHP заключается в `<? ?>` и выполняется интерпретатором языка PHP на веб-сервере. В результате выполнения программного кода:

```
<? echo "<p>Язык программирования PHP.</p>";?>
```

на экран будет выведен текст:

Язык программирования PHP.

Для того чтобы запустить скрипт, необходимо создать файл с расширением «.php» (именно это расширение дает понять серверу, что в файле содержится PHP-скрипт) и скопировать туда предыдущий пример. Далее необходимо поместить файл в исполнимую директорию сервера Apache (обычно это директория `htdocs`). Для вызова скрипта нужно набрать в браузере соответствующий URL `http://localhost/имя_файла.php`.

Для вывода данных на экран используется оператор `echo`. Все переменные в PHP начинаются со знака `$`. Имена переменных чувствительны к регистру. Тип данных определяется автоматически. Операторы аналогичны операторам языков C, JavaScript.

Работа с массивами в PHP. Часто данные, на основе которых генерируется HTML-страница, хранятся в массивах. Задать массив можно с помощью функции `array()`:

```
$regions=array(1=>"Брест", "Витебск", "Гродно", "Гомель",  
"Могилев", "Минск_обл", "Минск");
```

В языке PHP часто используются ассоциативные массивы, т. е. массивы, в которых индексами могут быть не только числовые значения последовательного ряда, но и любые числа или строки. Например:

```
$fruits = array("банан"=>"желтый","помидор"=>"красный");
```

Элементами массива могут массивы.

```
$months = array("зима" => array( 'январь', 'февраль' ), "весна" =>  
array("март","апрель","май"));
```

Генерация таблицы на основании данных из массива.

```
<html><head>  
<title>Вывод элементов массива </title>  
</head><body>  
<? $x=array(50,100,150,250); // описание массива X  
// вывод заголовка таблицы  
echo"<table border=1> <tr><td>i</td><td>X[i]</td></tr>"; for($i=0;$i<count($x);$i++){  
// вывод строки таблицы
```

```

echo "<tr><td>".($i+1)."</td><td>".$x[$i]."</td></tr>";
} // конец вывода строки таблицы
echo "</table>"; // вывод окончания таблицы
?>
</body></html>

```

В языке PHP имеется ряд суперглобальных массивов:

`$_SERVER` – содержит переменные, установленные web-сервером либо напрямую связанные с окружением выполнения текущего скрипта;

`$_ENV` – содержит переменные окружения;

`$_REQUEST` – содержит переменные, передаваемые скрипту через методы GET, POST, cookies. Наличие и порядок включения переменных в этот массив определяется в соответствии с директивой `variables_order` конфигурационного файла PHP.

Например, определить IP-адрес посетителя, можно написав:

```
<?echo $_SERVER["REMOTE_ADDR"] ?>
```

Функции в PHP. Формат описания функции пользователя:

```

function имя_функции(параметры){
    тело функции
}

```

В определении функции можно указать аргумент функции по умолчанию, которое будет использоваться, если при вызове функции аргумент не указан. Значение по умолчанию должно указываться справа. Функция, которая сгенерирует тег `<SELECT>`:

```

function generate_select($select_array,$name_par,$size_par=1){
    //вывод открывающего тега
    echo "<SELECT name=\"\$name_par\" size=\"\$size_par\">
        <OPTION value=\"0\" > выбрать";
        foreach($select_array as $value=>$text)
            echo "<OPTION value=\"\$value\" > $text"; //вывод опций
    echo "</SELECT>"; //вывод закрывающего тега
}

```

Если внутри строки необходимо вывести на экран специальный символ, например, кавычки, то перед таким символом ставится символ «`\`».

Вызывать функцию можно:

```

generate_select($regions,"town");
generate_select($regions,"town",6);

```

Нужно учесть, что все переменные внутри функции являются локальными, действующими только в теле функции. Чтобы указать, что будем использовать глобальную переменную, необходимо перед ней написать `global`.

```

function decode_region($index){
    global $regions; // используем ранее описанный массив regions
    echo "Код $index регион ".$regions[$index].".";
}

```

Для соединения строк в PHP используется операция «.».

```
decode_region(7); // выведет Код 7 регион Минск
```

Функцию можно сделать значением переменной. Продолжим пример:

```
$action="decode_region";
```

```
echo $action(1); // выведет Код 1 регион Брест
```

В языке PHP разработано множество библиотек встроенных функций. Например:

phpinfo() – выводит на экран параметры конфигурации веб-сервера;

count(array arr) вычисляет размер массива;

sort(array arr) – сортирует массив по возрастанию;

print_r(array arr) – выводит на экран массив;

array_rand(array arr) – выполняет сортировку в случайном порядке элементов массива.

Необязательный параметр num_req задает количество сортируемых элементов в массиве;

in_array(mixed element, array arr) – возвращает true, если значение element найдено в массиве arr, и false иначе;

strtoupper(string \$str) – приводит str к верхнему регистру;

strtolower(string \$str) – приводит str к нижнему регистру;

substr(string \$str, int start, int length) – возвращает часть строки str, начиная с позиции start длиной length;

string str_replace(string from, string to, string str) – заменяет в строке str все вхождения from на to;

strlen(string \$str) – возвращает длину строки;

explode(string separator, string str) – делит строку str по разделителю separator. Результат возвращает в виде массива;

ltrim(string \$str) – удаляет начальные пробелы из строки;

rtrim(string \$str) – удаляет конечные пробелы из строки;

trim(string \$str) – удаляет начальные и конечные пробелы из строки;

ord(string \$str) – возвращает ASCII-код символа;

time() – возвращает количество секунд, прошедших с 00:00:00 1.01.1970;

mktime() – формирует временную метку;

getdate() – возвращает ассоциативный массив с ключами: seconds, minutes, hours, mday, wday, mon, year, yday, weekday, month, 0;

date() – позволяет форматировать дату.

Примеры программного кода:

```
$str="удовлетворительно;хорошо;отлично";
```

```
$array=explode(";", $str);
```

```
print_r($array);
```

Результат:

```
Array ( [0] => удовлетворительно [1] => хорошо [2] => отлично )
```

Часто возникает задача определить, сколько по времени исполнялся скрипт. Код для ее решения:

```
//текущее время на момент начала выполнения скрипта $part_time=explode(' ',microtime());
```

```
$begin_time=$part_time[1].substr($part_time[0],1);
```

```
... //скрипт
```

```
//текущее время на момент завершения выполнения скрипта
```

```
$part_time=explode(' ',microtime());
```

```
$end_time=$part_time[1].substr($part_time[0],1);
```

```
echo $end_time-$begin_time;// реальное время выполнения скрипта
```

Функция вывода календаря по заданным месяцу и году.

```
function show_kalendar($month,$year){
```

```
echo "<table><tr><td >";
```

```
for ($i=1; $i<= date("t",$time);$i++){
```

```

$time=mktime(0,0,0,$month,$i,$year);
if (date("D",$time)=="Mon") echo "<td valign=top>";
if((date("D",$time)=="Sun")||(date("D",$time)=="Sat")){
    echo "<font color=red>".date("d",$time)."</font><br>";}
else{echo "<font color=black>".date("d",$time)."</font><br>";
}
if (date("D",$time)=="Sun") echo "</td>";
}
echo "</tr></table>";
}?>

<?show_kalendar(12,2014);?>

```

Порядок выполнения работы

1. Изучить механизм создания и запуска PHP-скриптов.
2. Создать веб-страницу, на которой будет выведена надпись «Я знаю PHP!».
3. Изучить работу с массивами и выполнить примеры программ, приведенных в теоретическом материале.
4. Задан массив, содержащий названия предметов. Вывести значения в виде маркированного нумерованного списка.
5. Изучить работу с функциями в языке PHP и выполнить примеры программ, приведенных в теоретическом материале.
6. Создать функцию, которая формирует и выводит на экран меню. Входные данные – последовательности названий пунктов и соответствующих им URL. Использовать функцию в скрипте, для вывода меню.
7. Создать отчет, включив в него описание работы по пунктам 2–6, ответы на контрольные вопросы.

Контрольные вопросы

- 1) Привести алгоритм запуска скрипта на сервере?
- 2) Как задать массив в языке php?
- 3) Что такое ассоциативные массивы?
- 4) Привести формат описания функции пользователя?

Лабораторная работа №9

Получение и обработка данных из пользовательских форм

Цель работы: научиться получать на сервере данные из пользовательских форм, проводить на основе них расчеты и выводить результаты на html-страницу.

Теоретические сведения

Данные на сервер передаются посредством HTML-форм. Для взаимодействия с сервером форма должна иметь обязательные элементы: атрибут action, который указывает путь к серверному скрипту-обработчику; кнопку `<INPUT type="submit">`, инициализирующую передачу данных; в каждом поле должен быть указан атрибут name. Существует два метода передачи данных из формы в скрипт: GET и POST. Метод GET предполагает присоединение к URL имен и значения форм (`http:// URL/страница.PHP?имя=значение&имя=значение`). Метод POST, передает данные отдельным запросом.

Данные, переданные на сервер методом GET, хранятся в суперглобальном массиве `$_GET`, а данные, переданные методом POST, – в суперглобальном массиве `$_POST`. В скрипт значения передаются следующим образом: значение атрибута name становится индексом массива, а введенные в поле данные – значением этого элемента массива.

```
<html><head></head>
<body>
<form action="" method="POST">
Ваше имя <input type="text" name="name"><br>
Ваша фамилия <input type="text" name="surname"><br>
<input type="submit" name="button" value="Да!">
</form>
<?if (isset($_POST['button']))// если нажали на кнопку с именем button
echo "Поздравляю, ". $_POST['name']. " ".$_POST['surname']. "<br> скрипт, передающий
данные из формы, заработал! " ;?>
</body></html>
```

При использовании элемента `< INPUT type="checkbox">` в случае установки флажка в обработчик передается переменная с именем, соответствующим имени самого checkbox, со значением On. Если checkbox пуст, то в скрипте эта переменная не будет определена.

Проверить, какие данные пришли на сервер методом POST, можно с помощью следующей команды:

```
print_r($_POST);
Результат для предыдущего скрипта:
Array ( [name] => Алесь [surname] => Васильков [button] => Да! )
```

Порядок выполнения работы

1. Изучить теоретические сведения по теме «Передача данных из пользовательских форм».
2. Выполнить примеры программного кода, приведенного в теоретическом материале.
3. Создать PHP-страницу для решения следующей задачи. Банк предлагает три вида вкладов: «15 дней 5 % годовых», «30 дней 4,7 % годовых», «35 дней 4,5 % годовых». Пользователям, имеющим счета в банке, предоставляется бонус в размере 0,1 % годовых. Разработать скрипт для расчета суммы начисленных процентов и итоговой суммы к выдаче в зависимости от введенной пользователем суммы.

Формула для расчета суммы начисленных процентов:

$$P=i/36500*n*M$$

Формула для расчета суммы к выдаче:

$$S = (i/36500 * n + 1) * M$$

Где S – сумма к выдаче, P – сумма начисленных процентов, i – ставка(% годовых), n – количество дней вклада, M – сумма вклада.

4. Создать отчет, включив в него описание работы по пунктам 2–4, ответы на контрольные вопросы.

Контрольные вопросы

1. Приведите элементы формы. Какие из них обязательны для передачи данных на сервер?
2. В чем разница между методами передачи get и post?
3. Каким образом получить данные на сервере? Привести пример.
4. Как можно проверить, какие данные получены на сервере? Привести пример.

Лабораторная работа №10

Отправка HTTP заголовков. Сессий и cookies

Цель работы: научиться сохранять данные в течение сеанса работы пользователя с помощью механизмов COOKIES и сессий.

Теоретические сведения

Протокол TCP/IP не позволяет поддерживать постоянное соединение с веб-сервером. Имитировать постоянное соединение можно используя механизмы COOKIES или сессий. При использовании COOKIES данные записываются в специальные файлы, которые хранятся на компьютере пользователя. Недостаток COOKIES в том, что пользователь в параметрах браузера может запретить их использование. В этом случае не будут работать те функции сайта, которые используют COOKIES. Поэтому не рекомендуется ключевые функции сайта реализовывать с их помощью.

Регистрацию данных в COOKIES выполняет функция:

```
setcookie(имя, значение, срок действия, путь, домен, защищенность);
```

Функция должна быть вызвана до отправки каких-либо данных удаленному клиенту, в том числе «пустых» символов, т. е. пробелов, символов перевода строки и т. д. Все параметры, кроме имени cookie, являются необязательными. Значением, которое несет cookie, может быть любая строка ASCII символов. Например, установим cookie с именем и фамилией посетителя, которые он ранее ввел в поле формы.

```
$value = $_POST["name"]."||".$_POST["surname"];
```

```
setcookie("my_cookie", $value, time() + 3600 * 24 * 5, "/", ".somesrv.com", 1);
```

Для того чтобы сохранить в COOKIES массив, необходимо его сериализовать (упаковать строку) с помощью функции `serialize()`, а затем выполнить обратный процесс, применив функцию `unserialize()`.

Обращение к данным, сохраненным в cookie, происходит через суперглобальный массив `$_COOKIE`, используя в качестве индекса имя cookie. Например, продолжая пример выше, прочесть cookie можно следующим образом:

```
echo "У вас сохранены следующие данные:<br>";
```

```
echo $_COOKIE["my_cookie"];
```

Удаление COOKIES производится отправкой новой cookie с именем удаляемой без каких-либо дополнительных параметров. Например:

```
$data = $my_cookie;
```

```
setcookie("my_cookie");
```

```
echo "Следующие данные были удалены:<br>" . $data;
```

Создание настраиваемого интерфейса сайта с помощью механизма COOKIE

Пользователям нравится, когда они имеют возможность влиять на работу программы. В частности пользуется популярностью такая возможность, как настройка интерфейса пользователя по своему желанию. Конечно, полностью создавать новый дизайн считается неправильным, так как дизайн создан профессионалом и помечен его копирайтом. И если непрофессиональный пользователь начнет перерисовывать, может получиться непрофессиональный дизайн, на котором стоит копирайт. Однако некоторые параметры интерфейса пользователь может подобрать для себя. Функция настройки дизайна не является ключевой для работы приложения. Т. е. если даже COOKIES и не работает, то пользователь сможет работать с приложением, но в том дизайне, который предложит ему разработчик.

Необходимо реализовать возможность пользователю выбирать цвет текста и фона страницы. Для начала необходимо указать форму, с помощью которой пользователь будет изменять параметры интерфейса.

```
<html><head>
```

```
  <title>Параметры тестовой страницы </title>
```

```
</head><body>
```



```

&nbsp;<?echo $message;?><br>
<h3> Настройки</h3>
<form name='sform' action="" method="POST">
<b>Цвет фона</b>
<select name='bg_color'>
<option value='white' <?if( $_COOKIE['bg_color'] == "white" ) echo "selected"?> > white
<option value='black' <?if($_COOKIE['bg_color']=="black") echo "selected"?> >black
</select>   <br>
<b>Цвет текста</b>
<select name='text_color'>
<option value='red' <?if($_COOKIE['text_color']=="red") echo "selected"?> > red
<option value='blue'<?if($_COOKIE['text_color']=="blue") echo "selected"?> > blue
</select>   <br><br>
<input type="submit" name="bt" value="Сохранить">
</form>
</body></html>

```

Следующая функция запишет в COOKIE параметры из html формы.

```

function set_params(){
setcookie("bg_color",$_POST['bg_color'], time()+600);
setcookie("text_color",$_POST['text_color'], time()+600);
header("Location: $_SERVER[PHP_SELF]?bt=check_params");}

```

Хорошим тоном будет вывод пользователю сообщения о результате действия: либо параметры успешно изменены, либо пользователь не может воспользоваться данной функцией, так как у него не работает COOKIE. Это выполнит функция:

```

function check_params(){
if(isset($_COOKIE['bg_color'])) return "Настройки сохранены";
else return "Настройки не могут быть сохранены, так как Ваш браузер не поддерживает
COOKIE";}

```

Теперь вызываются эти функции в нужном порядке

```

if(isset($_POST['bt'])) {set_params();}
else if(isset($_GET['bt'])) {$message=check_params();}
else $message = "Ваши текущие настройки."

```

Далее создаем тестовую страницу.

```

<html><head>
<style>
body {background-color: <?if(isset($_COOKIE['bg_color']))echo $_COOKIE['bg_color']; else
echo "grey"?>;
color: <?if(isset($_COOKIE['text_color']))echo $_COOKIE['bg_color']; else echo "green"?>;}
</style>
<title>Тестовая страница</title>
</head><body >
Тест
</body></html>

```

Реализация механизма сессий в PHP. Сессия открывается с помощью функции `session_start()`, создающей специальный служебный файл с именем, соответствующим идентификатору сессии (сокращенно SID), в который впоследствии будут записаны все данные, связанные с текущей сессией. Место размещения этих файлов зависит от настроек PHP. Также эта функция используется для продолжения текущей сессии. То есть, если для данного пользователя файл создан, то функция `session_start()` не будет создавать новый файл, а просто откроет существующий. Таким образом, она должна быть вызвана на каждой странице, использующей данные текущей сессии. При использовании в скриптах сессий необходимо

иногда чистить директорию с данными временными файлами, так как там со временем может накопиться большое количество ненужных файлов.

Доступ к данным сессии можно получить через суперглобальный массив `$_SESSION`.

Регистрация данных в сессию осуществляется:

```
session_start();
```

```
$_SESSION['name'] = "Алесь Васильков";
```

Теперь на любой странице данного сайта можно обратиться к посетителю по имени:

```
session_start();
```

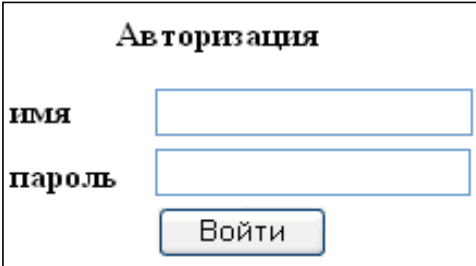
```
echo " You are " . $_SESSION['name'];
```

Если требуется уничтожить какой-то элемент в текущей сессии, то следует использовать функцию `unset($_SESSION['имя_элемента'])`:

```
unset($_SESSION['name']);
```

Завершает работу сессии функция `session_destroy()`. Она уничтожает файл, связанный с текущей сессией. Если необходимо обнулить все значения, хранящиеся в сессии, то следует вызвать функцию `session_unset()`, которая уничтожит все элементы в суперглобальном массиве `$_SESSION` текущей сессии.

Авторизация с помощью сессий. Форма для авторизации показана на рисунке 1.



Авторизация	
имя	<input type="text"/>
пароль	<input type="password"/>
<input type="submit" value="Войти"/>	

Рисунок 1 – Форма авторизации

Форма авторизации хранится в файле `admin.html`. Данные из формы обрабатываются скриптом `access.php`. В случае успешной авторизации происходит перенаправление на файл `index.php` (`index.php` работает только, если пользователь авторизовался).

HTML-код формы авторизации:

```
<form name="loginForm" action="access.php" method="post">
<table>
<tr><td> </td><td> Авторизация </td> </tr>
<tr><td> имя</td> <td><input type="text" name="login" ></td> </tr>
<tr><td>пароль</td><td><input type="password" name="psw"></td> </tr>
<tr><td></td><td><input type="submit" name='submit' value="Войти"></td></tr>
</table>
</form>
<script>
```

```
document.loginForm.login.focus();
```

```
</script>
```

Обработчик формы:

```
<?function authorize ($login,$psw){
if($login=="l"&&$psw=="p") return 1;
else return 0;}
```

```

session_start();
if($_POST['submit']){
if (authorize ($_POST['sotr'],$_POST['pas'])){
// если авторизовался, переходим на основной модуль
echo "<html><head><META HTTP-EQUIV='Refresh' CONTENT = '0;
URL=http://localhost/index.php'> </head> </html>" ;
} else { echo "Неверный пароль";}}?>

```

Чтобы не разрешить неавторизованному пользователю доступ на страницу, нужно в начало файла (в нашем случае *index.php*) добавить строки:

```

session_start();
if(!isset($_SESSION['user_id'])) {// если не зарегистрировался
echo "Доступ на данную страницу Вам запрещен"; exit; }

```

Порядок выполнения работы

1. Изучить принцип работы механизма cookies и средства его реализации в php.
2. Разобрать и выполнить пример по разработке настраиваемого пользовательского интерфейса.
3. Изучить принцип работы механизма сессий и средства его реализации в php.
4. Разобрать и выполнить пример по реализации авторизация с помощью сессий.
5. Создать отчет, включив в него описание работы по пунктам 2, 4, ответы на контрольные вопросы.
- 6.

Задание для самостоятельной работы

Вариант 1

1. Спросите у пользователя телефон с помощью формы. Затем сделайте так, чтобы в другой форме (поля: имя, фамилия, телефон) при ее открытии поле телефон было автоматически заполнено.
2. Спросите дату рождения пользователя. При следующем заходе на сайт напишите сколько дней осталось до его дня рождения. Если сегодня день рождения пользователя - поздравьте его!

Вариант 2

1. Запишите в сессию время захода пользователя на сайт. При обновлении страницы выводите сколько секунд назад пользователь зашел на сайт.
2. Сделайте счетчик посещения сайта посетителем. Каждый раз, заходя на сайт, он должен видеть надпись: 'Вы посетили наш сайт % раз!'.

Вариант 3

1. Спросите у пользователя email с помощью формы. Затем сделайте так, чтобы в другой форме (поля: имя, фамилия, пароль, email) при ее открытии поле email было автоматически заполнено.

Вариант 4

1. Сделайте счетчик обновления страницы пользователем. Данные храните в сессии. Скрипт должен выводить на экран количество обновлений. При первом заходе на страницу он должен вывести сообщение о том, что вы еще не обновляли страницу.

2. Запомните дату последнего посещения сайта пользователем. При заходе на сайт напишите ему, сколько дней он не был на вашем сайте.

Вариант 5

1. Сделайте счетчик обновления страницы пользователем. Данные храните в сессии. Скрипт должен выводить на экран количество обновлений. При первом заходе на страницу он должен вывести сообщение о том, что вы еще не обновляли страницу.

2. Реализуйте выбор цвета шапки сайта пользователем. Сделайте несколько цветов для шапки. Пользователь может выбрать один из цветов с помощью выпадающего списка. Этот выбор будет сохранен в куки и пользователь, заходя на сайт, всегда будет видеть один и тот же цвет шапки. Можете заменить цвет на полноценный дизайн.

Контрольные вопросы

1. Для чего используются механизмы сессий и cookies?
2. Какие средства для работы с cookies в языке php?
3. Как сохранить и получить доступ к переменным сессии?

Лабораторная работа №11

Средства языка PHP для работы с СУБД MySQL

Цель работы: научиться выбирать из базы данных и выводить на веб-страницу данные, а также реализовывать поиск на основе критериев, введенных через HTML-формы.

Теоретические сведения

В языке PHP имеется ряд модулей («extensions») для взаимодействия с различными СУБД. По умолчанию при установке интерпретатора они могут быть отключены. Для подключения необходимо в конфигурационном файле «*php.ini*» раскомментировать соответствующую строку и перезапустить сервер. На рисунке 1 приведен алгоритм взаимодействия с СУБД на примере MySQL.

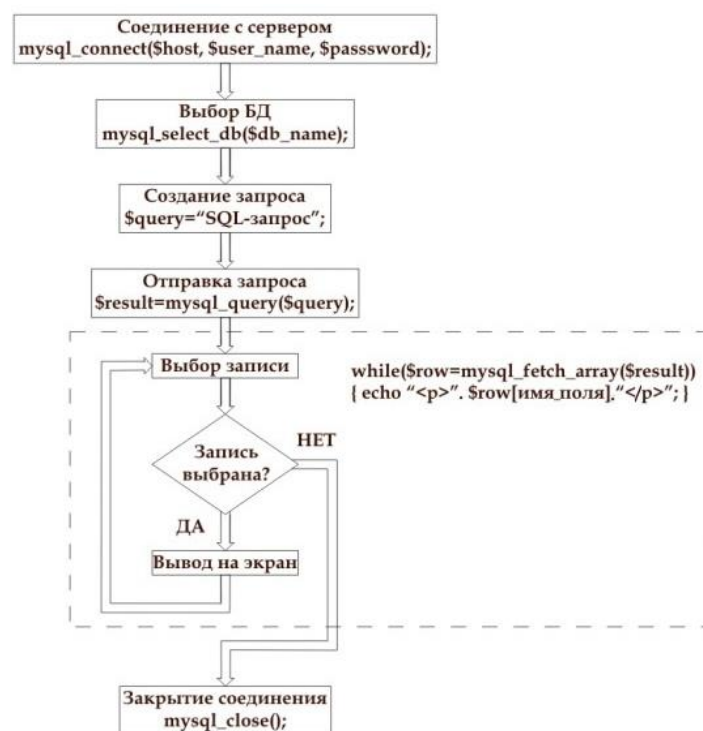


Рисунок 1 – Алгоритм взаимодействия с СУБД на примере MySQL

Пусть имеется база данных *list* таблица следующей структуры:

```
CREATE TABLE tbl(  
  id int(10) unsigned NOT NULL auto_increment primary key,  
  name varchar(45) NOT NULL, #название  
  ord int(10) unsigned NOT NULL # номер  
);
```

Для взаимодействия от имени веб-сервера в СУБД создан пользователь с именем *user_name* и паролем *user_psw*, имеющий привилегии для работы с базой данных *list*.

Требуется организовать возможность просмотра через веб-интерфейс записей из данной таблицы по страницам по 5 записей на каждой странице. Вид приведен на рисунке 2.

Всего : 8	
Номер	Название
1	Основы алгоритмизации и программирования
2	Модели данных и системы управления базами данных
3	Программирование для Интернет
4	Технология разработки программного обеспечения
5	Конструирование программ и языки программирования
Страницы: [1 2]	

Рисунок 2 – Интерфейс пользователя

Данная задача решается по этапам.

Установка соединения с СУБД:

```
<? // параметры соединения с базой данных
$host = "localhost";
$user = "user_name";
$pass = "user_psw";
$db = "list";
mysql_connect ( $host, $user, $pass ) or die ( "Нет соединения с MySQL сервером :
".mysql_error());
mysql_query("SET CHARACTER SET cp1251");
mysql_select_db( $db ) or die ( "Нет соединения с базой данных : ".mysql_error()); ?>
Вывод всех записей из таблицы (без постраничной навигации):
<?$query="SELECT name,ord FROM tbl ORDER BY ord"; //echo $query;
$result=mysql_query($query) or die(mysql_error());
$all =MYSQL_NUMROWS($result); // сколько всего записей
echo "<b>Всего :</b> $all";
if ($all>0){//если есть данные то выводим им
echo"<table border=1 width=100% cellpadding=0 cellspacing=0>
<tr><td width='40'>Номер</td><td>Название</td></tr>";
while($array = mysql_fetch_array($result, MYSQL_ASSOC)){
echo "<tr><td width='40'>$array[ord]</td>
<td> $array[name]</td></tr>";
}
?></table><?
}?>
```

Затем необходимо перейти непосредственно к постраничному выводу. Для извлечения данных определенной страницы из базы данных используется конструкция *LIMIT* оператора *SELECT*. Необходимо знать номер страницы и количество записей на странице. Номер страницы будем передавать в адресной строке, а для хранения количества записей на странице введем переменную *\$record_on_page*. Значение переменных будут определяться следующим образом:

```
if (!isset($_GET['page'])) $_GET['page']=0;
$record_on_page=5;
```

Добавим к запросу конструкцию *LIMIT* следующим образом:

```
$query=" LIMIT ".$_GET['page']*$record_on_page.", $record_on_page";
```

Далее будут выводиться записи соответствующей страницы. И тут появляется самая сложная проблема – нужно знать, сколько всего записей находится в таблице (либо сколько записей удовлетворяет условию поиска, если запрос содержит конструкцию *WHERE*). А запрос теперь выдает столько записей, сколько указано в переменной *\$record_on_page*. Для решения данной проблемы необходимо воспользоваться следующим приемом: добавить в sql-запрос опцию *SQL_CALC_FOUND_ROWS*, затем с помощью функции *FOUND_ROWS()* получить это количество записей. Таким образом, код скрипта, выполняющий вывод записей преобразуется:

```
$query="SELECT SQL_CALC_FOUND_ROWS name, ord FROM tbl
ORDER BY ord LIMIT ".$_GET['page']*$record_on_page.", $record_on_page";//echo $query;
$result=mysql_query($query) or die(mysql_error());
//считаем сколько всего данных
```

```

$res=mysql_query("SELECT FOUND_ROWS() as number;") or die(mysql_error());
$all=mysql_result($res,0,"number");
echo "<b>Bcero :</b> $all";
$all_in_page=MYSQL_NUMROWS($result);
if ($all_in_page>0){//если есть данные то выводим им
echo "<table border=1 width=100% cellpadding=0 cellspacing=0>
<tr><td width='40'>Homep </td><td>Название</td></tr>";
while($array = mysql_fetch_array($result, MYSQL_ASSOC)){
echo "<tr><td width='40'>$array[ord]</td>
      <td> $array[name]</td></tr>";
    }
?></table><?
}
Вывод на экран элемента перехода по страницам:
echo " <div>Страницы: [ ";
for($i=0; $i<ceil($all/$record_on_page);){
    if ($i+1==$_GET['page']+1) echo "<span>".++$i." </span>";
    else echo " <a href='\".$action.\"?page=$i'>\".++$i.\" </a>";
}
echo " ]</div>";

```

В переменной \$action хранится URL страницы.

Функциональный элемент поиск. Критерии поиска вводятся через форму на рисунке 3.

Рисунок 3 – Форма поиска

Далее следует создать функцию, которая будет возвращать SQL-запрос на поиск данных. Для хранения данных дополняется база данных list из предыдущего примера двумя таблицами следующей структуры:

```

CREATE TABLE `inf` (
  `id` int(10) unsigned NOT NULL auto_increment primary key,
  `fld1_name` varchar(45) NOT NULL default "",
  `fld2_name` varchar(45) NOT NULL default "",
  `id_category` int(10) unsigned NOT NULL default '0',
  `activity` tinyint(1) NOT NULL default '1',
  `ord` int(10) unsigned NOT NULL default '0',
  `date` datetime default NULL
) ENGINE= MyISAM DEFAULT CHARSET= cp1251;
CREATE TABLE `category` (
  `id` int(10) unsigned NOT NULL auto_increment primary key,
  `name` varchar(45) NOT NULL default "#название категории"
) ENGINE= MyISAM DEFAULT CHARSET=cp1251;

```

Html-код формы поиска выглядит так:

```

<form name="find_form" action="" method="POST">
<table><tr><td align="right"></td>
<td><input type="text" name="find_string" value="<?echo $_POST['find_string'];?>" /></td>
</tr><tr><td align="right">искать в </td>
<td><SELECT name="find_field">
    <OPTION value="fld1_name" <?if ($_POST['find_field']=="fld1_name") echo
"selected";?>> поле fld1
    <OPTION value="fld2_name" <?if ($_POST['find_field']=="fld2_name") echo
"selected";?>> поле fld2
</SELECT> </td>
</tr><tr><td align="right" valign="top"> <b>Категория </b></td>
<td>
<? $res = mysql_query("SELECT * FROM category"); $i=0;
while ($arr = mysql_fetch_array($res, MYSQL_ASSOC)){
echo "<input type=\"checkbox\" name=\"cat[\" value=\"\"$arr[id]\"";
if (isset($_POST['cat'])&&in_array($arr[id],$_POST['cat'])) echo "checked";
echo ">$arr[name]"; if(++$i%2==0) echo "<br>";
} ?> </td>
</tr><tr><td align="right"><b>Период</b></td><td></td></tr>
<tr><td align="right">c</td>
<td><input type="text" name="date_begin" value="2006-01-01"/></td>
</tr><tr><td align="right">no</td>
<td><input type="text" name="date_end" value="2009-01-01"/></td>
</tr><tr><td align="right"><b>Статус</b></td>
<td><SELECT name="activity">
    <OPTION value=0 <?if ($_POST['activity']=="все") echo selected";?> > все
    <OPTION value=1 <?if ($_POST['activity']=="1") echo "selected";?> > активные
    <OPTION value=-1 <?if ($_POST['activity']=="0") echo "selected";?> > неактивные
</SELECT></td>
</tr><tr><td align="right">сортировать по</td>
<td><SELECT name="sort_by">
    <OPTION value="id_category" <?if ($_POST['sort_by']=="id_category") echo
"selected";?>> категориям
    <OPTION value="activity" <?if ($_POST['sort_by']=="activity") echo "selected";?> >
статусу
    <OPTION value="date" <?if ($_POST['sort_by']=="date") echo "selected";?>> date
</SELECT></td>
</tr><tr>
<td align="right"></td>
<td><input type="submit" name="button" value="find"/></td>
</tr></table>
</form>

Функция, формирующая SQL-запрос выглядит:
function generate_find_query(){// формирует запрос на поиск данных
$query="SELECT * FROM inf WHERE $_POST[find_field]= \"$_POST[find_string]\" AND
date BETWEEN \"$_POST[date_begin]\" AND \"$_POST[date_end]\""; //добавили поля ввода
(элемент INPUT)
//добавляем статус (элемент SELECT)
if($_POST['activity']!="все") $query.=" AND activity=$_POST[activity]";
//добавляем категории (элемент checkbox)
$query.=" AND id_category in (" . implode(", ", $_POST['cat']) . ")";
//добавляем сортировку
$query.=" ORDER BY $_POST[sort_by]";
return $query;
}

```


Вызов функции:

```
if (isset($_POST['button'])) echo generate_find_query ();
```

Порядок выполнения работы

1. Изучить алгоритм работы с СУБД, приведенный в кратких теоретических сведениях.
2. Разобрать и выполнить пример по выводу данных из таблицы.
3. Разобрать и выполнить пример по разработке функционального элемента «Поиск».
4. Для функционального элемента «Поиск» разработать функцию, выводящую на веб-страницу результаты поиска в виде таблицы.
5. Разработать веб-страницу по следующему заданию

Дана БД, скрипт для создания БД находится в файле (shop.sql). Создать скрипт, для вывода статистики работы магазина за указанный пользователем месяц. Интерфейс пользователя приведен на рисунке 4.

месяц	Выбрать
Количество заказов	XX
Выручка по категориям	
Категория 1	XX
Категория 2	XX
Средняя стоимость заказа	XX

Рисунок 4 – Интерфейс пользователя скрипта статистики

Внести изменения в скрипт:

- а. Вместо средней стоимости выводить минимальную и максимальную стоимости, вид интерфейса приведен на рисунке 5.

Минимальная стоимость заказа	XX
Максимальная стоимость заказа	XX

Рисунок 5 – Изменения в интерфейсе скрипта статистики

б.* Реализовать показ статистики только по выбору месяца, без нажатия кнопки «Выбрать». Кнопку выбрать исключить из пользовательского интерфейса.

- б. Создать отчет, включив в него описание работы по пунктам 2, 4, ответы на контрольные вопросы.

Контрольные вопросы

- 1 Какие функции в языке PHP предназначены для работы с MySQL и каково их назначение?
- 2 Сформулируйте алгоритм построения SQL-запроса на основании данных HTML-формы.

Лабораторная работа №12

Работа с графикой в PHP

Цель работы: изучить возможности модуля **gd**, научиться обрабатывать и выводить в браузер средствами PHP графические изображения.

Теоретические сведения

Для работы с графикой в PHP имеется модуль **gd** (по умолчанию при установке интерпретатора отключен).

Далее рассмотрены принципы работы с графическими изображениями. Прописывается функция, выполняющая масштабирование изображений – выводить на экран изображение с заданными размерами с авторским знаком и выводить изображение в браузер. Параметры: имя файла с изображением, ширину и высоту изображения.

```
function resize_img($filename,$w,$h){
```

Для начала корректируются размеры нового изображения, чтобы не исказить пропорции в отмасштабированном.

```
$ratio=$w/$h;// отношение длины к ширине в новых размерах
```

```
$size_img=getimagesize($filename);// получаем размеры
```

```
$src_ratio=$size_img[0]/$size_img[1]; /* отношение длины к ширине в реальном изображении*/
```

Изменяются размеры с целью соблюдения отношения длины к ширине в реальном изображении после масштабирования:

```
if ($ratio<$src_ratio) $h=$w/$src_ratio; else $w=$h*$src_ratio;
```

Затем непосредственно масштабирование. Для начала нужно создать пустое изображение:

```
$dest_img=imagecreatetruecolor($w,$h);
```

Далее получаем в память изображение в зависимости от типа файла:

```
if($size_img[2]==2)$src_img=imagecreatefromjpeg($filename);
```

```
else if($size_img[2]==1) $src_img=imagecreatefromgif($filename);
```

Теперь выполняем непосредственно масштабирование.

```
//копируем в $dest_img отмасштабированное $src_img
```

```
if(!imagecopyresampled($dest_img,$src_img,0,0,0,0,$w,$h,$size_img[0],$size_img[1]))return false;
```

Далее проставить авторский знак:

```
$color=imagecolorallocate($dest_img,0,0,0);
```

```
imagestring($dest_img,1,$w/2,$h-15,"@Tanya",$color);
```

Изображение готово, в зависимости от типа оно выводится:

```
if($size_img[2]==2){
```

```
header("Content-type: image/jpeg");
```

```
imagejpeg($dest_img);}
```

```
else if($size_img[2]==1){
```

```
header("Content-type: image/gif");
```

```
imagegif($dest_img);}
```

И не забыть освободить память, занятую под изображения:

```
imagedestroy($dest_img); imagedestroy($src_img);}
```

Функция готова. Вставка масштабированного изображения в html-страницу:

```
$img=$_GET['img'];// принимаем имя файла
```

```
resize_img($_GET['img'],$_GET['width'],$_GET['height']); //"/../images/1.gif"/ вызываем функцию
```

При вставке изображения в html-страницу возникает следующая проблема: браузеру невозможно одновременно отправить и текстовые данные, и изображение. Поэтому следует поступить так: в том месте страницы, где необходимо поместить масштабированное изображение, вставить тег:

``, где *script_name* – имя файла, содержащего выше приведенный код; *img_file_name* – имя файла, в котором хранится картинка; *x* – ширина картинки, *y* – высота картинки.

Тег `` «зарезервирует» место под картинку. После загрузки страницы сработает скрипт, который в «зарезервированное» место вставит изображение.

Порядок выполнения работы

1. Изучить принцип работы с графикой PHP.
2. Разобрать и выполнить пример по обработке изображения.
3. Создать отчет, включив в него описание работы по пункту 2 ответы на контрольные вопросы.

Задание для самостоятельной работы

Вывести изображения, которые должны быть сжаты до 200 px по ширине. Также у картинок должен быть заполнен атрибут альтернативного имени.

Контрольные вопросы

1. Какая библиотека используется для работы с графикой?
- В чем особенность работы с графикой в языке PHP?

Лабораторная работа №13

Объектно-ориентированное программирование в PHP

Цель работы: научиться описывать и создавать экземпляры классов, переопределять оператор вывода, а также использовать объектно-ориентированное программирование для создания html-страниц.

Теоретические сведения

Описание классов в языке PHP:

```
class имя_класса{  
    свойства;  
    методы;  
};
```

Создание экземпляра класса:

```
$имя_переменной=new имя_класса;
```

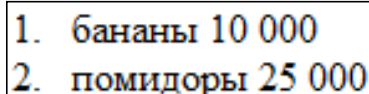
Имя функции-конструктора в языке PHP – `__construct()`.

Чтобы переопределить оператор `echo`, нужно создать функцию `__toString()`.

Наследование:

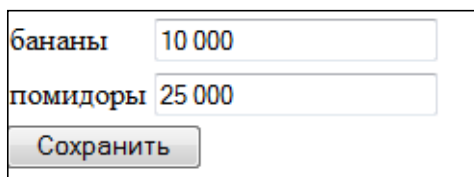
```
class имя_класса_предка extends имя_класса_потомка{  
}
```

Далее рассматривается задача. Имеется прайс лист, показанный на рисунке 1. Вывести веб-страницу с отображением прайс-листа в зависимости от привилегии пользователя (рисунок 2, 3).



1.	бананы	10 000
2.	помидоры	25 000

Рисунок 1 – Вид прайс-листа для продавца



бананы	<input type="text" value="10 000"/>
помидоры	<input type="text" value="25 000"/>
<input type="button" value="Сохранить"/>	

Рисунок 2 – Вид прайс-листа для администратора



ТЕБЕ НЕЛЬЗЯ СМОТРЕТЬ ПРАЙС-ЛИСТ!!!

Рисунок 3 – Вид веб-страницы для неавторизованного пользователя

В начале создается базовый класс `price_list`, который будет содержать свойство `$price_list`, для хранения массива, конструктор, инициализирующий массив и абстрактный метод для вывода массива (прайс-листа).

```
<?php  
abstract class price_list{//базовый класс  
protected $price_list;// прайс-лист  
function __construct($price_list_mess){//конструктор  
    $this->price_list = $price_list_mess;  
}
```

```

abstract function __toString();//функция вывода
};
Далее создается два класса, наследующих базовый класс, но функция вывода в каждом из
них будет определена по своему.
class show_price_list extends price_list{// для продавца
function __ toString {//описание функции вывода для продавца
Вывод будет осуществляться не на веб-страницу, а в буфер.
ob_start();
?><ol><?
foreach($this->price_list as $index=>$val)echo "<li>$index $val </li>";
?></ol><?
$a=ob_get_contents();// получаем содержимое буфера
ob_end_clean();//очищаем буфер
return $a;
}
}
class show_edit_price_list extends price_list{//для администратора
function __toString(){//описание функции вывода для администратора
ob_start();?>
<form action="" method="POST"><table><?
foreach($this->price_list as $index=>$val)
echo "<tr><td>$index</td><td><input type='text' name='$index' value='$val'></td></tr>";
?></table><input type="submit" value="Сохранить"/></form><?
$a=ob_get_contents();
ob_end_clean();
return $a;
}
}
};
session_start(); $_SESSION['us_g']=3;
$price_list= array("бананы"=>"1,5", "помидоры"=>"2");
Определяется тип переменной $price_block в зависимости от категории пользователя:
switch($_SESSION['us_g']){
case 1: $price_block=new show_price_list($price_list);
case 2: $price_block =new show_edit_price_list($price_list);
default: $price_block ="ТЕБЕ НЕЛЬЗЯ СМОТРЕТЬ ПРАЙС-ЛИСТ!!!";
};?>
<html><head></head><body><?echo $price_block;?></body></html>

```

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме «Объектно-ориентированное программирование в PHP».
2. Разобрать и выполнить приведенный пример.
3. Создать отчет, включив в него описание работы по пункту 2, ответы на контрольные вопросы.

Задание для самостоятельной работы

Вариант 1

Класс Дробное число со знаком (Fractions). Число должно быть представлено двумя полями: целая часть - длинное целое со знаком, дробная часть - беззнаковое короткое целое. Реализовать арифметические операции сложения, вычитания, умножения и операции сравнения. В функции main проверить эти методы.

Вариант 2

Класс Деньги для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа unsigned char - для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения. В функции main проверить эти методы.

Вариант 3

Класс Равнобочная трапеция, члены класса: координаты 4-х точек. Предусмотреть в классе конструктор и методы: проверка, является ли фигура равнобочной трапецией; вычисления и вывода сведений о фигуре: длины сторон, периметр, площадь. В функции main продемонстрировать работу с классом: дано N трапеций, найти количество трапеций, у которых площадь больше средней площади.

Контрольные вопросы

1. Как описать экземпляр класса?
2. Как описать наследование классов?
3. Как создать экземпляр класса?
4. Как переопределить оператор вывода echo?

Лабораторная работа №14

Взаимодействие сервлета и jsp-страниц

Цель работы: научиться разделять динамическую и статическую части web-страниц, путем создания сервлета и взаимодействующих с ним jsp-страниц.

Теоретические сведения

Технология Java Servlet предоставляет web-разработчикам простой последовательный механизм для увеличения функциональности web-сервера и для доступа к существующим коммерческим системам. Сервлеты Java расширяют возможности вебприложений.

Сервлеты обеспечивают компонентный, платформонезависимый метод для построения web-приложений без ограничений производительности программ. В отличие от собственных механизмов сервера (Netscape Server API или модулей Apache), сервлеты серверно- и платформонезависимы.

Сервлеты имеют доступ ко всему семейству прикладных программных интерфейсов Java APIs, включая JDBC API для доступа к базам данных. Сервлеты также могут иметь доступ к библиотеке специальных вызовов HTTP и получать все преимущества полноценного языка Java.

HTTP – это Hyper Text Transfer Protocol – протокол передачи гипертекста. По большому счету передается не гипер, а самый обычный текст. Гипертекстом он становится тогда, когда его отображает браузер. Именно браузер в соответствии с тэгами форматирует текст, делает ссылки (именно это делает текст гипертекстом), отображает рисунки и т. д. HTTP считается одним из самых простых протоколов. На сегодняшний день существует четыре версии протокола HTTP – 0.9, 1.0, 1.1 и 2. Наиболее распространенными командами являются GET и POST.

Технология JSP это расширение технологии Java Servlet, предназначенное для поддержки создания HTML и XML страниц. Она упрощает комбинирование фиксированных или статических данных с динамическим содержанием.

После настройки сервера Tomcat создайте свой первый сервлет, как показано в примере 1: Сначала в Eclipse выберите создать новый проект Dynamic Web project. Нажмите Далее. Введите название проекта (рисунок 1) и нажмите Finish. Для создания нового сервлета нажмите правой клавишей мыши на проекте, затем New-Servlet. Задайте Имя класса (рисунок 2) и обратите внимание на URL mapping.

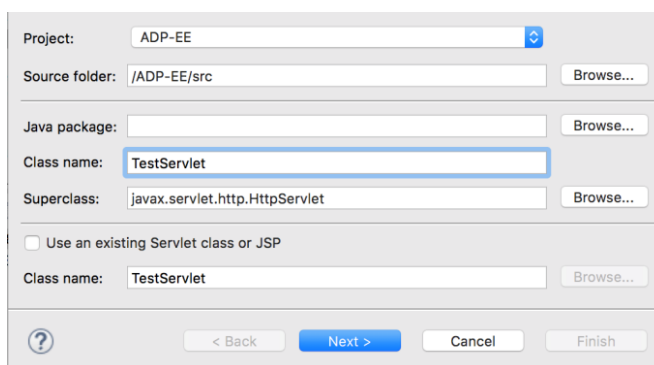


Рисунок 1 – Ввод названия проекта

Рисунок 2 – Ввод названия класса

После выполнения всех действий сервлет создан. Теперь измените содержимое метода doGet:

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType("text/html");
    // Actual logic goes here.
    PrintWriter out = response.getWriter();
    out.println("<h1>" + "hello from [ " + request.getRequestURI() + "]</h1>");
}
```

Запустите сервлет. Нажмите правой кнопкой мыши по Сервлету и выберите Run As – Run on Server. При успешном выполнении всех пунктов на экране увидите результат, показанный на рисунке 3.

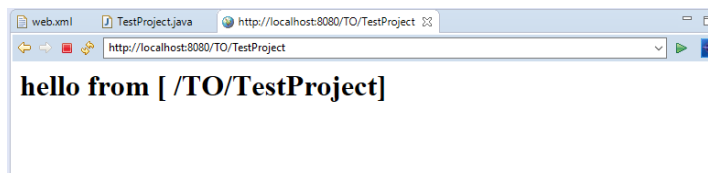


Рисунок 3 – Результат выполнения сервлета

Так же такой же результат можно увидеть, если загрузить браузер и ввести адрес <http://localhost:8080/TO/TestProject>. (TO - проект, TestProject - сервлет)

Web.xml – дескриптор развертывания приложения располагается в каталоге /WEB-INF. Можно создать этот файл вручную или использовать возможности Eclipse. На названии проекта ADP-EE нажмите правой кнопкой мыши и вызовите свойства, где выберите Java EE Tools-Generate Deployment Descriptor Stub (рисунок 4). В результате создастся web.xml.

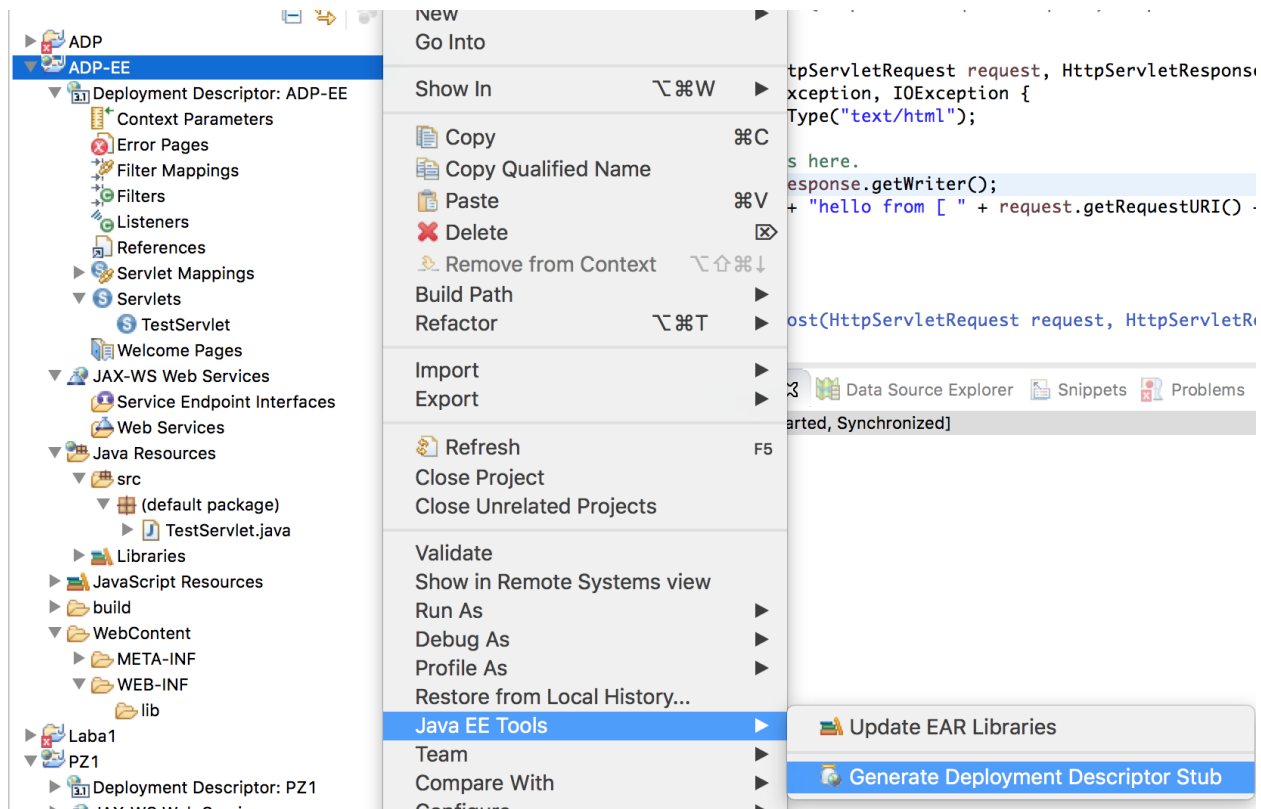


Рисунок 4 – Создание Web.xml

В содержимое web.xml внесите дополнения:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
    version="3.1">
    <display-name>ADP-EE</display-name>
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
        <welcome-file>default.html</welcome-file>
        <welcome-file>default.htm</welcome-file>
        <welcome-file>default.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>TestServlet</servlet-name>
        <servlet-class>by.bsac.adp.pz1.TestServlet</servlet-class>
    </servlet>
</web-app>
```

В дескрипторном файле можно определять параметры инициализации, MIME-типы, mapping сервлетов и JSP, стартовые страницы и страницы с сообщениями об ошибках, а также параметры для безопасной авторизации и аутентификации. Этот файл можно сконфигурировать так, что путь к сервлету в браузере не будет совпадать с истинным именем класса сервлета. Для версии сервлетов 3.0 теги мэппинга не нужны, так как эта информация определяется аннотацией.

Пример 2: Необходимо написать сервлет, который выдает пустую HTML-страницу с фоном зеленого цвета.

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
```

```
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class TestServlet
 */
```

```
@WebServlet("/TestServlet")
public class TestServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
```

Серийный номер версии класса. Значение сгенерировано eclipse автоматически. Используется для проверки. Константа serialVersionUID может быть использована для синхронизации версий объектов класса, если приложение будет расположено на *кластере серверов.

```
private static final String CONTENT_TYPE = "text/html; charset=windows-1251";
public void init() throws ServletException { }
```

Метод doGet вызывается в случае, если в сервлет поступает GET запрос. Метод принимает два параметра request и response. Request содержит в себе информацию, пришедшую со стороны клиента. В Response вносится информацию, которую отправляется на клиента.

```
public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    response.setContentType(CONTENT_TYPE);
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Servlet_2</title></head>");
    out.println("<body bgcolor='#00ff00'>");
    out.println("</body>");
    out.println("</html>");
    out.close();}
```

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    doGet(request, response);}
```

```
public void destroy() { }}
```

Основная идея JSP очень проста – сама страница представляет из себя шаблон с уже заготовленными HTML-тэгами, между которыми надо вставить нужные данные. Например:

```
<HTML> <HEAD>
    <TITLE>Hello World Sample<TITLE>
</HEAD>
<BODY>
    <H1>
        [данные]
    </H1>
</BODY>
</HTML>
```

Страницы JSP и сервлеты никогда не используются в информационных системах друг без друга. Причиной являются принципиально различные роли, которые играют данные компоненты в приложении. Страница JSP ответственна за формирование пользовательского интерфейса и отображение информации, переданной с сервера. Сервлет выполняет роль контроллера запросов и ответов, то есть принимает запросы от всех связанных с ним JSP-страниц, вызывает

соответствующую бизнес-логику для их (запросов) обработки и в зависимости от результата выполнения решает, какую JSP поставить этому результату в соответствие.

JSP при первом обращении преобразуется в сервлет и работает уже как сервлет. JSP НЕ ЯВЛЯЕТСЯ страницей наподобие HTML-страницы – это еще один сервлет – просто его вывод не надо программировать. Его можно просто нарисовать. И в нужные места подставить данные. Но т.к. JSP-страница хоть как-то напоминает HTML, то дизайнеру проще с ней работать. Ее подготовка со всеми данными происходит на сервере. Именно там подставляются все данные. А пользователю в браузер приходит уже готовая HTML-страница, на которой никаких признаков JAVA нет.

JSP имеет в своем составе несколько predefined объектов (их надо воспринимать, как уже готовые объекты, которые могут быть использованы). Это request, response, out, session, application, config, pageContext и page.

request – Это объект HttpServletRequest, связанный с запросом, который позволяет обращаться к параметрам запроса (через метод getParameter), типу запроса (GET, POST, HEAD, и т.д.), и входящим HTTP заголовкам.

response – Это объект типа HttpServletResponse, связанный с ответом на запрос клиента. Поскольку поток вывода буферизован, можно изменять коды состояния HTTP и заголовки ответов, даже если это недопустимо в обычном сервлете, но лишь в том случае если какие-то данные вывода уже были отправлены клиенту.

out – Это объект типа PrintWriter, используемый для отправки вывода клиенту. Используется практически исключительно скриптами, поскольку выражения JSP автоматически помещаются в поток вывода, что избавляет от необходимости явного обращения к out.

session – Это объект типа HttpSession, связанный с запросом. Сессии создаются автоматически, и эта переменная существует даже если нет ссылок на входящие сессии.

application – Это объект типа ServletContext полученный через использование метода getServletConfig().getContext().

config – Это объект типа ServletConfig для текущей страницы.

pageContext – В JSP представлен новый класс PageContext для изолированного использования специфических особенностей сервера.

page – является синонимом для this, и не нужен при работе с Java. Эта переменная создавалась с расчетом на перспективу, когда возможно появятся другие языки программирования скриптов, отличные от Java.

Нет необходимости регистрировать jsp-страницы в дескрипторе развертывания. Достаточно разместить их в области, доступной из структуры каталогов web-приложения, и они станут доступны. Но можно зарегистрировать jsp-страницу подобно сервлету:

```
<servlet>
```

```
<servlet-name>JSP_name1</servlet-name>  
<jsp-file>index.jsp</jsp-file>  
</servlet>  
<servlet-mapping>  
<servlet-name> JSP_name1</servlet-name>  
<url-pattern>jsp_name2</ url-pattern >  
</servlet-mapping >
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме.
2. Разобрать и выполнить примеры.
3. Создать отчет, включив в него описание работы по пункту 2, результаты выполнения заданий для самостоятельной работы согласно варианту и ответы на контрольные вопросы.

Задание для самостоятельной работы

Вариант 1

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P1" и кнопкой "Submit". После заполнения пользователем поля для ввода и нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, в поле P1 которой должно содержаться введенное значение, повторенное два раза.

Вариант 2

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P2" и кнопкой "Submit". После заполнения пользователем поля для ввода и нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, поле P2 которой должно быть пустым, а кнопка должна содержать введенное значение.

Вариант 3

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P3" и кнопкой "Submit". После нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, в поле P3 которой должно содержаться сообщение о том, были ли введены данные.

Вариант 4

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P4" и кнопкой "Submit". После заполнения пользователем поля для ввода и нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, в поле P4 которой должно содержаться введенное значение в обратном порядке.

Вариант 5

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P5" и кнопкой "Submit". После нажатия кнопки "Submit" сервлет должен выдать HTML-страницу, содержащую ещё одно поле для ввода.

Вариант 6

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P6" и кнопкой "Submit". После заполнения пользователем поля для ввода и нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, в поле P6 которой должно содержаться фразу «Введенное значение:» и далее само введенное значение.

Вариант 7

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P7" и кнопкой "Submit". После нажатия кнопки "Submit" сервлет должен выдать HTML-страницу, содержащую ещё одну кнопку.

Вариант 8

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P8" и кнопкой "Submit". Только после заполнения пользователем поля для ввода и нажатия кнопки "Submit" сервлет должен выдать HTML-страницу без поля для ввода.

Вариант 9

Написать сервлет, который выдает HTML-страницу с кнопкой "Submit". После нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, содержащую также и поле для ввода с именем «P9».

Вариант 10

Написать сервлет, который выдает HTML-страницу с полем для ввода с именем "P10" и кнопкой "Submit". После заполнения пользователем поля для ввода и нажатия кнопки "Submit" сервлет должен выдать такую же HTML-страницу, в поле P10 которой отображалась бы длина введенного значения.

Контрольные вопросы

1. Что такое WEB-приложение?
2. Охарактеризуйте методы запроса GET и POST.
3. Каково назначение сервлетов?
4. От какого класса обычно наследуется сервлет?
5. Назовите и опишите методы, которые переопределяют сервлеты?
6. Что такое дескриптор развертывания?
7. Перечислены коды состояния HTTP и сообщения, связанные могут быть возвращены с сервера.
8. Для чего используются JSP (Java Server Pages).
9. Назовите преимущества Java Servlet.
10. Из каких этапов состоит жизненный цикл сервлета?

Лабораторная работа №15

Создание web-страниц с использованием сессий

Цель работы: закрепить на практике умение сохранять и изменять информацию о действиях клиента с использованием механизма сессий и cookie.

Теоретические сведения

Сеанс есть сессия между клиентом и сервером, устанавливаемая на определенное время, за которое клиент может отправить на сервер сколько угодно запросов. Сеанс устанавливается непосредственно между клиентом и веб-сервером в момент получения первого запроса к веб-приложению. Каждый клиент устанавливает с сервером свой собственный сеанс, который сохраняется до окончания работы с приложением.

Сеанс используется для обеспечения хранения данных при последовательном выполнении нескольких запросов различных веб-страниц или на обработку информации, введенной в пользовательскую форму в результате нескольких HTTP-соединений (например, клиент совершает несколько покупок в Интернет магазине; студент отвечает на несколько тестов в системе дистанционного обучения). Как правило, при работе с сессией возникают следующие проблемы:

- поддержка распределенной сессии (синхронизация/репликация данных, уникальность идентификаторов и т. д.);
- обеспечение безопасности;
- проблема инвалидации сессии (expiration), предупреждение пользователя об уничтожении сессии и возможность ее продления (watchdog).

Чтобы открыть явный доступ к экземпляру сеанса/сессии пользователя веб-приложения, используются методы `getSession(boolean create)` или `getSession()` интерфейса `HttpServletRequest`. Экземпляр запроса возвращает ссылку на объект сессии. Метод не создает сессию, а только дает доступ с помощью запроса к экземпляру сессии `HttpSession`, соответствующему данному пользователю.

Если для метода `getSession(boolean create)` входной параметр равен `true`, то сервлет-контейнер проверяет наличие активного сеанса, установленного с данным клиентом. В случае успеха метод возвращает дескриптор этого сеанса. В противном случае метод устанавливает/создает новый сеанс:

```
HttpSession session = request.getSession(true);
```

после чего начинается сбор информации о клиенте.

Если метод `getSession(boolean param)` вызывать с параметром `false`, то ссылка на активный сеанс будет возвращена, если он существует, если же сеанс уничтожен или не был активирован, то метод возвратит `null`. Метод `getSession()` без параметров работает так же, как и `getSession(true)`.

Сессия содержит информацию о дате и времени создания последнего обращения к сессии, которая может быть извлечена с помощью методов `getCreationTime()` и `getLastAccessedTime()`.

Метод `String getId()` возвращает уникальный идентификатор, который получает каждый сеанс при создании. Метод `isNew()` возвращает `false` для уже существующего сеанса и `true` – для нового сеанса. Задать время (в секундах) бездействия сессии, по истечении которого экземпляр сессии будет уничтожен, можно методом `setMaxInactiveInterval(long sec)`.

Чтобы сохранить значения переменной в текущем сеансе, используется метод `setAttribute(String name, Object value)` класса `HttpSession`, прочесть – `getAttribute(String name)`, удалить – `removeAttribute(String name)`. Список имен всех переменных, сохраненных в текущем сеансе, можно получить, используя метод Enumeration `getAttributeNames()`, работающий так же, как и соответствующий метод интерфейса `HttpServletRequest`.

Принудительно завершить сеанс можно методом `invalidate()`. В результате для сеанса будут уничтожены все связи с используемыми объектами, и данные, сохраненные в старом сеансе, будут потеряны для клиента и всех приложений.

Пример 1. Добавление информации в сессию с помощью сервлета.

```
public class SessionControl extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        HttpSession session = request.getSession(true);
        if (session.getAttribute("role") == null)
        { session.setAttribute("role", "moderator"); }
        /* количество запросов, которые были сделаны к данному сервлету
        * текущим пользователем в рамках текущей пользовательской сессии */
        Integer counter = (Integer) session.getAttribute("counter");
        if (counter == null) { session.setAttribute("counter", 1); }
        else { /* увеличивает счетчик обращений к текущему сервлету и кладет его в сессию
        */
            counter++; session.setAttribute("counter", counter); }
        request.setAttribute("lifecycle", "CONTROL request LIFECYCLE");
        request.getRequestDispatcher("/jsp/sessionattr.jsp").forward(request, response); }
}
```

Далее необходимо создать jsp-страницы `sessionattr.jsp` для того, чтобы отразить информацию о роли пользователя и счетчике обращений.

```
<% @ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<html><head><title>Session Attribute</title></head>
<body> Role: ${role}
<br /><hr /> Counter: ${counter}
<br /><hr /> MaxInactiveInterval: ${pageContext.session.maxInactiveInterval}
<br/> ID session: ${pageContext.session.id}
<br/> Lifecycle: ${lifecycle}
<br/> <a href="index.jsp">Back to index.jsp</a> </body></html>
```

Содержимое стартовой jsp-страницы:

```
<% @ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<html> <head><title>Index Page</title></head>
<body> Lifecycle: ${lifecycle}
<a href= "sessionServlet">Session Servlet</a> </body></html>
```

Время жизни сессии при отсутствии активности пользователя задано с помощью тега `session-config` в `web.xml` в виде:

```
<session-config>
<session-timeout>20</session-timeout>
</session-config>
```

где время ожидания задается в минутах.

В результате в браузер будет выведено в качестве данных сеанса счетчик обращений – объект типа `Integer` и роль пользователя. В ответ на пользовательский запрос сервлет `SessionServlet` возвращает страницу `sessionattr.jsp`, на которой отображаются все атрибуты сессии, а также идентификационный номер сессии и время инвалидации сессии.

Для хранения информации на компьютере клиента используются возможности класса `javax.servlet.http.Cookie`. `Cookie` – это небольшие блоки текстовой информации, которые сервер посылает клиенту для сохранения в файлах `cookies`. Клиент может запретить браузеру прием файлов `cookies`. Браузер возвращает информацию обратно на сервер как часть заголовка HTTP, когда клиент повторно заходит на тот же веб-ресурс.

Чтобы послать cookie клиенту, сервлет должен создать объект класса Cookie, указав конструктору имя и значение блока, и добавить их в объект-response. Конструктор использует имя блока в качестве первого параметра, а его значение – в качестве второго.

```
Cookie cookie = new Cookie("BSAC", "The Belarusian State Academy of Telecommunications");  
response.addCookie(cookie);
```

Извлечь информацию cookie из запроса можно с помощью метода getCookies() объекта HttpServletRequest, который возвращает массив объектов, составляющих этот файл.

```
Cookie[] cookies = request.getCookies();
```

После этого для каждого объекта класса Cookie можно вызвать метод getValue(), который возвращает строку String с содержимым блока cookie. В данном случае этот метод вернет значение "The Belarusian State Academy of Telecommunications".

Экземпляр Cookie имеет целый ряд параметров: путь, домен, номер версии, время жизни, комментарий. Одним из ключевых является срок жизни в секундах от момента первой отправки клиенту, определить который следует методом setMaxAge(long sec). Если параметр не указан, то cookie существует только до момента прерывания контакта клиента с приложением.

Пример 2. Добавление cookie в сервлет, где иницируется процесс их извлечения и передачи информации.

Jsp-страница maincookie.jsp: информация из cookie.

```
<% @ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<html><head><title>from Cookie </title></head>  
<body> ${messages}  
<a href="cookiecontroller">Messages from Cookie</a> </body></html>
```

При старте приложения \${messages} ничего не выведет, так как атрибут messages не существует. Сервлет CookieController добавляет cookie к объекту-ответу и пытается извлечь cookie из объекта-запроса. При первом запуске извлекать нечего, так как запрос пришел без cookie.

Содержимое сервлета CookieController:

```
public class CookieController extends HttpServlet {  
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws  
ServletException, IOException {  
        CookieAction.setCookie(response); // добавление cookie  
        // извлечение cookie и добавление информации к request  
        request.setAttribute("messages", CookieAction.addToRequest(request));  
        request.getRequestDispatcher("/jsp/maincookie.jsp").forward(request, response); }  
}
```

Сервлет CookieAction формирует и извлекает cookie:

```
public class CookieAction {  
    private static int number = 1;  
    public static void setCookie(HttpServletResponse resp) {  
        String name = "TO";  
        String role = "00" + number++;  
        Cookie c = new Cookie(name, role);  
        c.setMaxAge(60 * 60); // время жизни файла cookie  
        resp.addCookie(c); // добавление cookie к объекту-ответу  
        value = resp.getLocale().toString();  
        Cookie loc = new Cookie("locale", value);  
        resp.addCookie(loc);  
    }  
    public static ArrayList<String> addToRequest(HttpServletRequest request) {  
        ArrayList<String> messages = new ArrayList<>();  
        Cookie[] cookies = request.getCookies();  
        if (cookies != null) {  
            messages.add("Number cookies : " + cookies.length);  
            for (int i = 0; i < cookies.length; i++) {
```



```

Cookie c = cookies[i];
messages.add(c.getName() + " = " + c.getValue()); } }
return messages;
}
}

```

При первом запуске приложения cookie, естественно, переданы не будут, так как они еще не созданы, и браузер еще не получал ни одного ответа. При повторном обращении к сервлету cookie уже будет передан с запросом

Файл cookie, как это видно из указанного примера, может быть перезаписан при генерации каждого нового объекта-ответа.

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме.
2. Разобрать и выполнить примеры 1-2.
3. Создать отчет, включив в него описание работы по пункту 2, результаты выполнения заданий для самостоятельной работы согласно варианту и ответы на контрольные вопросы.

Задание для самостоятельной работы

Для всех заданий использовать авторизованный вход в приложение. Параметры авторизации, дату входа в приложение и время работы сохранять в сессии.

Вариант 1

В файле хранится текст. Для каждого из слов, которые вводятся в текстовые поля HTML-документа, вывести в файл cookie, сколько раз они встречаются в тексте.

Вариант 2

Записать в файл cookie все вопросительные предложения текста, которые хранятся в текстовом файле.

Вариант 3

Код программы хранится в файле. Подсчитать количество операторов этой программы и записать результаты поиска в файл cookie, перечислив при этом все найденные операторы.

Вариант 4

Сохранить в cookie информацию, введенную пользователем, и восстановить ее при следующем обращении к странице.

Вариант 5

Выбрать из текстового файла все числа-полидромы и их количество. Результат сохранить в файле cookie.

Вариант 6

В файле хранится текст. Найти три предложения, содержащие наибольшее количество знаков препинания, и сохранить их в файле cookie.

Вариант 7

В файле хранится код программы. Удалить из текста все комментарии и записать измененный файл в файл cookie.

Вариант 8

Выбрать из файла все адреса электронной почты и сохранить их в файле cookie.

Вариант 9

В файле хранится HTML-документ. Найти и вывести все незакрытые теги с указанием строки и позиции начала в файл cookie.

Вариант 10

Выбрать из файла имена зон (*.by, *.kz и т. д.), вводимые пользователем, и сохранить их в файле cookie.

Контрольные вопросы

1. Что такое сессия?
2. Для чего используется механизм сессий?
3. Охарактеризуйте методы сессии.
4. Что такое cookie?
5. Для чего используется cookie?
6. Охарактеризуйте методы cookie.
7. Охарактеризуйте интерфейсы и методы, которые позволяют следить за событиями, связанными с сеансом.

Лабораторная работа №16

Взаимодействие с библиотекой JSTL

Цель работы: научиться разделять динамическую и статическую части web-страниц, путем создания сервлета и взаимодействующих с ним jsp-страниц.

Теоретические сведения

Создание страниц с применением JSTL позволяет упростить разработку и отказаться от вживления java-кода в JSP. Страницы со скриптами трудно читаемы, что вызывает проблемы как у программиста, так и у веб-дизайнера, не обладающего глубокими познаниями в Java.

Библиотека тегов JSTL состоит из пяти групп тегов: основные теги – core, теги форматирования – formatting, теги для работы с SQL – sql, теги для обработки XML – xml, функции-теги для обработки строк – functions.

Библиотеки следует разместить в каталоге /lib проекта или в соответствующем каталоге контейнера сервлетов. При указании значения параметра xmlns элемента root (для JSP-страницы значение параметра директивы taglib uri=("")) необходимо быть внимательным, так как при неправильном указании адреса JSP-страница не сможет получить доступ к тегам JSTL.

Подключение библиотеки core:

- `<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>` – для обычной JSP;
- `<jsp:root version="1.2" xmlns:c="http://java.sun.com/jsp/jstl/core">...</jsp:root>` – для XML формата JSP.

Подключение библиотеки fmt:

- `<% @ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>`

Подключение библиотеки sql:

- `<% @ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql" %>`

Подключение библиотеки xml:

- `<% @ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>`

Подключение библиотеки functions:

- `<% @ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>`

При использовании библиотеки **JSTL** необходимо указывать контекст, где сервер приложений должен искать значение/переменную или куда он должен обратиться. Список predefined контекстов представлен в следующей таблице 1.

Таблица 1 – Список predefined контекстов

Контекст	Описание
pageScope	Контекст страницы. Переменные доступны только для текущей страницы.
requestScope	Контекст запроса. Переменные доступны на всех страницах, а также сервлетам, обслуживающим текущий запрос пользователя.
sessionScope	Контекст сессии. Переменные доступны в течение всей сессии пользователя, т.е. пока не будет закрыт браузер или не закончится предельное время бездействия.

Контекст	Описание
applicationScope	Контекст приложения. Доступ к переменным возможен на всех страницах веб-приложения. Это самый глобальный контекст.
param	В этом контексте располагаются параметры, отправленные пользователем/браузером либо как параметры адресной строки, либо как поля html-формы.
paramValues	Список значений параметров, которые были переданы серверу приложений. Формат отличается от предыдущего контекста. Если там <i>param</i> имел тип <code>HashMap<String, String></code> , то здесь <code>HashMap<String, String []></code> .
header	В этом объекте хранится информация об http-заголовках, которые были переданы от браузера/клиента веб-серверу.
headerValues	Список значений http-заголовков.
initParam	Конфигурационные параметры, указанные для страницы или сервлета в дескрипторе приложений <code>web.xml</code>
cookie	Список переменных, помещенных внутрь cookie.
pageContext	Ссылка на объект <code>pageContext</code> , автоматически создаваемых внутри jsp-страницы.

Библиотека **core** содержит в себе наиболее часто используемые решения. Теги общего назначения:

`<c:out />` – вычисляет и выводит значение выражения (таблица 2);

Таблица 2 – Синтаксис тега `c:out`

Attribute	Description	Required	Default
value	Выводимая информация	Yes	None
default	значение, выводимое по умолчанию	No	body
escapeXml	True – если тэг должен опускать специальные символы xml	No	true

`<c:set />` – создает и устанавливает переменную в указанную область видимости (таблица 3);

Таблица 3 – Синтаксис тега c:set

Attribute	Description	Required	Default
value	Значение для установки	No	body
target	Имя переменной, свойство которой будет модифицировано	No	None
property	Модифицируемое свойство	No	None
var	Имя переменной, чтобы хранить информацию	No	None
scope	Область видимости	No	Page

<c:remove /> – удаляет переменную из указанной области видимости (таблица 4);

Таблица 4 – Синтаксис тега c:remove

Attribute	Description	Required	Default
value	Значение для установки	No	body
target	Имя переменной, свойство которой будет модифицировано	No	None
property	Модифицируемое свойство	No	None
var	Имя переменной, чтобы хранить информацию	No	None
scope	Область видимости	No	Page

<c:catch /> – перехватывает обработку исключения (таблица 5).

Таблица 5 – Синтаксис тега c:catch

Attribute	Description	Required	Default
var	Имя переменной, которая перехватит <code>java.lang.Throwable</code> , если оно будет выброшено каким-нибудь элементом в теле	No	None

Теги условного перехода:

<c:if /> – тело тега выполняется, если значение выражения true (таблица 6);

Таблица 6 – Синтаксис тега c:if

Attribute	Description	Required	Default
test	Условие	Yes	None
var	Переменная для хранения результатов сравнения	No	None
scope	Область видимости для переменной, которая хранит результат сравнения	No	page

<c:choose /> (<c:when />, <c:otherwise />) – то же, что и <c:if /> с поддержкой нескольких условий и действия, производимого по умолчанию (таблица 7).

Таблица 7 – Синтаксис тега c:choose

Attribute	Description	Required	Default
test	условие	Yes	None

Итераторы:

<c:forEach /> – выполняет тело тега для каждого элемента коллекции, массива, итерируемого объекта (таблица 8);

Таблица 8 – Синтаксис тега c:forEach

Attribute	Description	Required	Default
items	источник(коллекция) для организации цикла	No	None
begin	Начальное значение счетчика цикла	No	0
end	Конечное значение счетчика цикла	No	Last element
step	Шаг цикла	No	1
var	Переменная для хранения текущего значения счетчика цикла	No	None
varStatus	Имя переменной, хранящей статус (количество пройденных циклов) счетчика цикла	No	None

<c:forEachTokens /> – выполняет тело тега для каждой лексемы в строке (таблица 9).

Таблица 9 – Синтаксис тега c:forEachTokens

Attribute	Description	Required	Default
delims	Символы-разделители	Yes	None

Теги обработки URL:

<c:redirect/> – перенаправляет запрос на указанный адрес URL (таблица 10);

Таблица 10 – Синтаксис тега c:redirect

Attribute	Description	Required	Default
url	URL для перенаправления с помощью пользовательского браузера	Yes	None
context	/ расположение web-приложения	No	Current application

<c:import/> – добавляет на JSP содержимое указанного веб-ресурса (таблица 11);

Таблица 11 – Синтаксис тега c:import

Attribute	Description	Required	Default
url	URL для импортирования	Yes	None
context	/ указание расположения web-приложения	No	Current application
charEncoding	Кодировка импортируемых данных	No	ISO-8859-1
var	Переменная для хранения импортированного текста	No	Print to page
scope	Область видимости переменной, что хранит импортируемый текст	No	Page
varReader	Имя альтернативной переменной для java.io.Reader	No	None

<c:url/> – формирует адрес с учетом контекста приложения request. getContextPath() (таблица 12);

Таблица 12 – Синтаксис тега c:url

Attribute	Description	Required	Default
value	Основной URL	Yes	None
context	/ расположение web-приложения	No	Current application
var	Имя переменной, для сохранения URL	No	Print to page
scope	Область видимости сохраненного URL	No	Page

<c:param/> – добавляет параметр к запросу, сформированному при помощи <c:url/> (таблица 13).

Таблица 13 – Синтаксис тега c:param

Attribute	Description	Required	Default
name	Имя параметра запроса для установления в URL	Yes	None
value	Значение параметра запроса для установления в URL	No	Body

Пример 1. Множественный условный переход.

```
<% @ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<html> <head><title>Core: choose</title></head>
```

```
<body> <c:set var="number" value="50"/>
```

```
<c:choose>
```

```
<c:when test="{ number > 10 }" >
```

```
<c:out value="число { number } больше 10"/>
```

```
</c:when>
```

```
<c:when test="{ number > 40 }" >
```

```
<c:out value="число { number } больше 40"/>
```

```
</c:when> <c:when test="{ number > 60 }" >
```

```
<c:out value="число { number } больше 60"/>
```

```
</c:when>
```

```
<c:otherwise> <c:out value="число { number } меньше 10"/>
```

```
</c:otherwise>
```

```
</c:choose>
```

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

Операции по управлению условным переходом существенно расширяются благодаря тегу `<c:choose>`. Выполнение производится только одного из всего набора действий тега после определения истинности условия. Все остальные выражения, даже истинные, игнорируются, после чего управление передается тегу, следующему после тега `</c:choose>`.

Если ни одно из действий `<c:when>` не выполнено, то есть `test` не принял значение `true`, то управление переходит к единственному оператору `<c:otherwise>`, который может и отсутствовать.

Библиотека **formatting** содержит теги форматирования и интернационализации для разработки локализованных приложений. При грамотном использовании страницы вообще не будут содержать текст, а вся информация (надписи на кнопках и ссылках, заголовки, сообщения и пр.) будет извлекаться из файлов свойств.

Теги интернационализации:

`<fmt:setLocale/>` – устанавливает региональные установки для страницы на основе объекта класса `Locale`;

`<fmt:setBundle/>`, `<fmt:bundle/>` – устанавливают объект `ResourceBundle`, используемый на странице. В зависимости от установленного значения локали выбирается файл ресурсов (как правило, `properties`), соответствующий указанному языку, стране или региону;

`<fmt:message/>` – извлекает локализованное сообщение из ресурса, определенного тегами `<fmt:setBundle/>` или `<fmt:bundle/>`;

`<fmt:requestEncoding />` – с атрибутом `value="utf-8"` устанавливает кодировку входящего запроса.

Теги форматирования дат и чисел:

`<fmt:timeZone/>`, `<fmt:setTimeZone/>` – устанавливает часовой пояс, используемый для форматирования;

`<fmt:formatNumber/>`, `<fmt:formatDate/>` – форматирует числа/даты с учетом установленной локали (региональных установок) либо указанного шаблона;

<fmt:parseNumber/>, <fmt:parseDate/> – переводит строковое представление числа/даты в объекты подклассов Number / Date.

Пример 2. Выводит числа в соответствии с региональными установками.

```
<% @ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
```

```
<% @ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<% @ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html><head><title>Формат числа</title></head>
<body>
<c:set var="currentNumber" value="118000"/>
<c:out value="Вывод формата числа : ${currentNumber}"/>
<br/> Формат (по умолчанию) : <fmt:formatNumber value="${currentNumber}"/>
<br/> Процентный формат : <fmt:formatNumber value="${currentNumber}"
type="percent"/>
<br/> <fmt:setLocale value="be-BY"/>
Белорусские рубли : <fmt:formatNumber value="${currentNumber}" type="currency"/>
<br/> <fmt:setLocale value="pl-PL"/> Польская валюта :
<fmt:formatNumber value="${currentNumber}" type="currency"/>
<br/> Французская валюта : <fmt:setLocale value="fr-FR"/>
<fmt:formatNumber value="${currentNumber}" type="currency"/><br/>
</body></html>
```

Библиотека **sql** используется для выполнения запросов SQL непосредственно из JSP и обработки результатов запроса в JSP.

Теги:

- <sql:dateParam> – определяет параметры даты для <sql:query> либо <sql:update>;
- <sql:param> – определяет параметры <sql:query> либо <sql:update>;
- <sql:query> – выполняет запрос к БД;
- <sql:setDataSource> – устанавливает data source (пула соединений) для <sql:query>, <sql:update>, и <sql:transaction> тегов;
- <sql:transaction> – объединяет внутренние теги <sql:query> и <sql:update> в одну транзакцию;
- <sql:update> – выполняет запрос на вставку/удаление/изменение БД.

В промышленном программировании данная библиотека не используется из-за прямого доступа из JSP в СУБД, что является явным нарушением шаблона MVC.

Библиотека **xml** используется для импорта, парсинга и обработки данных из XML-документов в документе JSPX.

Список тегов:

- <x:set> – XML-версия тега <c:set>;
- <x:out> – XML-версия тега <c:out>;
- <x:forEach> – XML-версия тега <c:forEach>;
- <x:if> – XML-версия тега <c:if>;
- <x:choose> – XML-версия тега <c:choose>;
- <x:when> – XML-версия тега <c:when>;
- <x:otherwise> – XML-версия тега <c:otherwise>;
- <x:parse> – разбор XML-документа;
- <x:transform> – трансформация XML-документа с применением XSLT-преобразования;
- <x:param> – XML-версия тега <c:param>, определяющая параметры для тега <x:transform>.

XML-документ может быть импортирован из внешнего файла или может быть размещен непосредственно в коде JSP, что практикуется крайне редко.

Пример 3. В XML-документе представлена информация о нескольких студентах, причем данные об одном студенте находятся в теге <student>. Вывод информации о всех студентах производится тегом <x:forEach select="\$doc/students/student" var="stud"> где элемент <student> со всем его содержимым ассоциируется с переменной stud после выбора из объекта doc с помощью простого оператора \$doc/students/student. Далее на каждой итерации цикла из объекта stud

выбираются элементы `<x:out select="$stud/name"/>` и атрибуты `<x:out select="$stud/@login"/>`, которые выводятся в браузер тегом `<x:out>`.

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" xmlns:x="http://java.sun.com/jsp/jstl/xml"
version="2.0">
  <jsp:directive.page contentType="text/html; charset=UTF-8"/>
  <html> <head><title>XML inside</title></head>
  <body>
    <x:parse var="doc">
      <students>
        <student login="iv" faculty="it">
          <name>Ivanov Ivan</name>
          <telephone>1112223</telephone>
          <address>
            <country>Belarus</country>
            <city>Minsk</city>
            <street>Brovki 12</street>
          </address>
        </student>
        <student login="petr" faculty="ec">
          <name>Petrov Petr</name>
          <telephone>4445556</telephone>
          <address>
            <country>Belarus</country>
            <city>Naroch</city>
            <street>Lenina 9</street>
          </address>
        </student>
      </students>
    </x:parse>
    <x:forEach select="$doc/students/student" var="stud">
      Name:    <x:out select="$stud/name"/><br/>
      Login:   <x:out select="$stud/@login" /><br/>
      Faculty: <x:out select="$stud/@faculty" /><br/>
      Country: <x:out select="$stud/address/country" /><br/>
      City:    <x:out select="$stud/address/city" /><br/>
      Street:  <x:out select="$stud/address/street" /><br/>
      Telephone: <x:out select="$stud/telephone" /><br/>
      <hr/><br/>
    </x:forEach>
  </body></html>
</jsp:root>
```

Теги из библиотеки **functions** выполняют роль, подобную смыслу библиотеки `fmt`, только не для чисел и дат, а для строк. Теги библиотеки используют префикс `fn` и во многом копируют известные методы класса `String` и не составляют особой сложности в применении.

Список тегов:

`${fn:length(аргумент)}` – подсчитывает число элементов в коллекции или длину строки;

`${fn:toUpperCase(String str)}` и `${fn:toLowerCase(String str)}` – изменяет регистр строки;

`${fn:substring(String str, int from, int to)}` – извлекает подстроку;

`${fn:substringAfter(String str, String after)}`

`${fn:substringBefore(String str, String before)}` – извлекает подстроку до или после указанной во втором аргументе;

`${fn:trim(String str)}` – обрезает все пробелы по краям строки;

`${fn:replace(String str, String str1, String str2)}` – заменяет все вхождения строки `str1` на строку `str2`;

`{fn:split(String str, String delim)}` – разбивает строку на подстроки, используя `delim`, как разделитель;

`{fn:join(массив, String delim)}` – соединяет массив в строку, вставляя между элементами подстроку `delim`;

`{fn:escapeXml(String xmlString)}` – сохраняет в обрабатываемой строке теги;

`{fn:indexOf(String str, String searchString)}` – возвращает индекс первого вхождения строки `searchString`;

`<c:if test="{fn:startsWith(String str, String part)}">` – возвращает истину, если строка начинается с подстроки `part`;

`<c:if test="{fn:endsWith(String str, String part)}">` – возвращает истину, если строка заканчивается на подстроку `part`.

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме.
2. Разобрать и выполнить примеры 1-3.
3. Создать отчет, включив в него описание работы по пункту 2, результаты выполнения заданий для самостоятельной работы согласно варианту и ответы на контрольные вопросы.

Задание для самостоятельной работы

Задание 1

Вариант 1: добавить на JSP содержимое указанного веб-ресурса.

Вариант 2: обработать исключение «деление на ноль».

Вариант 3: определить, является ли значение переменной отрицательным, положительным либо равным 0.

Вариант 4: по названию дня недели вывести сообщение о количестве занятий.

Вариант 5: с помощью цикла организовать вывод букв алфавита.

Вариант 6: разбить строку на подстроки и организовать вывод на экран.

Вариант 7: осуществить перенаправление между jsp-страницами.

Вариант 8: вывести дату по форматам 5 стран (выбрать различные форматы).

Вариант 9: вывести дату и время при задании 3 часовых поясов.

Вариант 10: удалить переменную из заданной области видимости и вывести соответствующее сообщение.

Задание 2

Организовать вывод всех данных и поиск в XML-документе, содержащем не менее 5 записей по 3 элемента на заданную тематику.

Вариант 1: XML-документ типа кулинарная книга.

Вариант 2: XML-документ типа расписание занятий.

Вариант 3: XML-документ типа календарь.

Вариант 4: XML-документ типа страна.

Вариант 5: XML-документ типа семья.

Вариант 6: XML-документ типа банковский вклад.

Вариант 7: XML-документ типа книжный магазин.

Вариант 8: XML-документ типа аптека.

Вариант 9: XML-документ типа туристическая путевка.

Вариант 10: XML-документ типа зоопарк.

Контрольные вопросы

1. Для чего используется библиотека тегов JSTL?

2. Охарактеризуйте каждую группу тегов. Приведите примеры.
3. Как подключить библиотеку для разных групп тегов?
4. Как организована работа с XML-документами?

Лабораторная работа №17

Создание web-страниц

Цель работы: научиться создавать web-проекты на языке C# в среде MS Visual Studio, добавлять функциональные элементы на форму, программировать события и изменять свойства элементов.

Теоретические сведения

Приложения типа Web могут вызываться из любого компьютера, подключенного к сети Internet. Для открытия такого приложения используется Web-браузер (например, Opera, Google Chrome, Internet Explorer и другие).

Создать web-проект можно следующими способами:

1) нужно сначала вызвать команду

File -> New -> Web Site...

В открывшемся окне, нужно выбрать шаблон «C#» и вложение «ASP .NET Empty Web Site». Можно также указать путь, где будут созданы рабочие файлы приложения (кнопка «Browse...»).

Место расположения файлов указывается в поле «Web-location». Доступны три способа размещения файлов:

- файловая система (File system);
- http-соединение;
- ftp-соединение.

Если выбрать «File system», то файлы приложения будут размещаться на локальном сервере (localhost), который создается системой. Это означает, что программа-клиент (наше приложение) и программа-сервер (условно отдаленный компьютер в сети) размещены на одном и том же (домашнем) компьютере. Фактически localhost – это IP-адрес, с помощью которого компьютер может обратиться в сети к самому себе, независимо от наличия или вида компьютерной сети.

В Visual Studio поддерживается несколько типов приложений ASP.NET, но все они компилируются и выполняются одинаково. Единственное, чем они отличаются, так это тем, какие файлы для них создаются в Visual Studio по умолчанию.

Ниже приведено краткое описание всех доступных шаблонов:

– ASP.NET Web Site (Веб-узел ASP.NET): создается полнофункциональный веб-сайт ASP.NET с готовой базовой инфраструктурой. Этот веб-сайт включает главную страницу с общей компоновкой (верхним колонтитулом, нижними колонтитулом и строкой меню) и готовые страницы default.aspx и about.aspx.

Также он содержит папку Accounts (Учетные записи) со страницами для регистрации, входа в систему и смены пароля и папку Scripts (Сценарии) с библиотекой jQuery для клиентских сценариев JavaScript.

– ASP.NET Empty Web Site (Пустой веб-сайт ASP.NET): создается почти пустой веб-сайт. Этот веб-сайт включает только сокращенную версию конфигурационного файла web.config, и ничего больше. Разумеется, с началом кодирования в него можно легко добавлять необходимые фрагменты.

– ASP.NET Dynamic Data Entities Web Site (Веб-сайт сущностей динамических данных ASP.NET): создается веб-сайт ASP.NET, использующий компонент ASP.NET Dynamic Data. Этот веб-сайт ориентирован на применение сущностной модели (Entity Model) для доступа к серверной базе данных, в то время как шаблон со схожим именем ASP.NET Dynamic Data LINQ to SQL Web Site предусматривает использование для этого более старого подхода LINQ to SQL.

– WCF Service (Служба WCF): создается служба WCF – библиотека серверных методов, которые удаленные клиенты (например, Windows-приложения) могут вызывать.

– ASP.NET Reports Web Site (Веб-узел отчетов ASP.NET): создается веб-сайт ASP.NET с элементом управления ReportView и компонентом SQL Server Reporting Services - инструмент для генерации отчетов по базам данных, которые можно просматривать и обрабатывать через Интернет. Шаблон ASP.NET Crystal Reports Web Site (Веб-сайт ASP.NET с Crystal Reports) обеспечивает похожие возможности, но с использованием программного обеспечения Crystal Reports.

Доступно также множество специализированных шаблонов, предназначенных для создания веб-приложений конкретных типов. Чтобы просмотреть их, необходимо щелкнуть на заголовке Online Templates (Шаблоны в Интернете) в левом крайнем углу диалогового окна New Web Site. Через некоторое время, необходимое для установления связи с веб-серверами Microsoft, добавляется список различных подкатегорий, каждая с собственной группой готовых шаблонов.

2) другой способ создания Web-приложения с помощью команды File -> New -> Project...

После этого откроется окно, в котором нужно выбрать шаблон «Visual C#» и тип проекта «ASP .NET Web Application».

После выполненных операций создается решение (Solution), в котором есть один проект типа веб-сайт. Если запустить на выполнение данный проект, то внизу в правой части экрана отобразится окно загруженного локального сервера. Для завершения работы приложения, нужно в MS Visual Studio вызвать команду «Stop Debugging» из меню «Debug».

Для добавления новой формы нужно выделить название приложения в Solution Explorer, сделать клик правой кнопкой «мыши» и в контекстном меню выбрать команду «Add New Item...». В результате откроется окно «Add New Item». В этом окне нужно выбрать шаблон «Visual C#» и элемент «Web Form». Имя формы (Name) оставляем сохраниться с расширением «file_name.aspx».

После добавления, в Solution Explorer можно увидеть дополнительные два файла:

- «file_name.aspx» – файл формы в терминологии HTML языка разметки гипертекста;
- «file_name.aspx.cs» – файл формы, который отвечает программному коду на языке C#.

С помощью этих двух файлов можно изменять внешний вид формы и организовывать работу в ней.

С помощью кнопок Design и Source можно переключаться между режимом проектирования и режимом кода страницы Default.aspx.

Пример 1. Дано три стороны треугольника: a, b, c. Используя формулу Герона, найти площадь треугольника.

Формула Герона имеет вид: $S = \sqrt{p \times (p - a) \times (p - b) \times (p - c)}$, где p – полупериметр, который находится по формуле: $p = \frac{a+b+c}{2}$.

Согласно с условием задачи, форма должна содержать следующие элементы управления:

- три поля ввода для ввода значений a, b, c;
- текстовые строки для вывода сообщений;
- кнопку для задания начала вычисления;
- текстовую строку и поле ввода, для вывода результат вычисления.

Перейти в режим проектирования можно с помощью кнопки «Design». С помощью «мышки» можно изменять размер формы.

При вынесении элемента управления в правой нижней части экрана можно изменять свойства и программировать события элементов управления.

Так для кнопки можно задать следующие свойства: цвет фона, шрифт, высота, ширина, отображаемый текст и другие. Событиями для кнопки являются: по нажатию, при загрузке, при инициализации и другие.

Вид формы может быть похожим на рисунок 1:

a=

b=

c=

S=

Далее необходимо запрограммировать события, которое будет генерироваться при клике на кнопке «Рассчитать». Программный код обработки события будет сформирован в файле «file_name.aspx.cs».

Необходимо выделить элемент управления Button1 и в списке свойств Properties перейти к вкладке Events. В вкладке Events сделать двойной клик «мышкой» напротив названия события «Click». Система откроет файл «file_name.aspx.cs» со следующим кодом:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
    }
}
```

В обработчик события Button1_Click(...) необходимо добавить код расчета площади треугольника по трем сторонам.

```
protected void Button1_Click(object sender, EventArgs e)
{
    double a, b, c, p, s;
    a = Double.Parse(TextBox1.Text);
    b = Double.Parse(TextBox2.Text);
    c = Double.Parse(TextBox3.Text);
    p = (a + b + c) / 2;
    s = Math.Sqrt(p * (p - a) * (p - b) * (p - c));
    TextBox4.Text = s.ToString();
}
```

Как известно, страницы ASP.NET содержат элементы управления ASP.NET вперемешку с обычными дескрипторами HTML. Добавляются HTML-дескрипторы либо путем ввода, либо перетаскиванием из соответствующей вкладки в панели Toolbox.

В состав Visual Studio входит очень удобный конструктор стилей, который позволяет форматировать любой статический элемент HTML с помощью свойств CSS.

Чтобы испробовать его, необходимо перетащить на страницу из раздела HTML в панели Toolbox элемент <div>. Этот элемент появится на странице в виде не имеющей границ панели. Далее щелкните на этой панели, чтобы выделить ее, и затем щелкнуть внутри поля Style в окне Properties (Свойства). После этого в поле Style появится кнопка с изображением троеточия (...). Щелчок на ней приводит к открытию диалогового окна Modify Style (Изменение стиля) с опциями для настройки цветов, шрифта, компоновки и рамки элемента, как показано на рисунке 17.2:

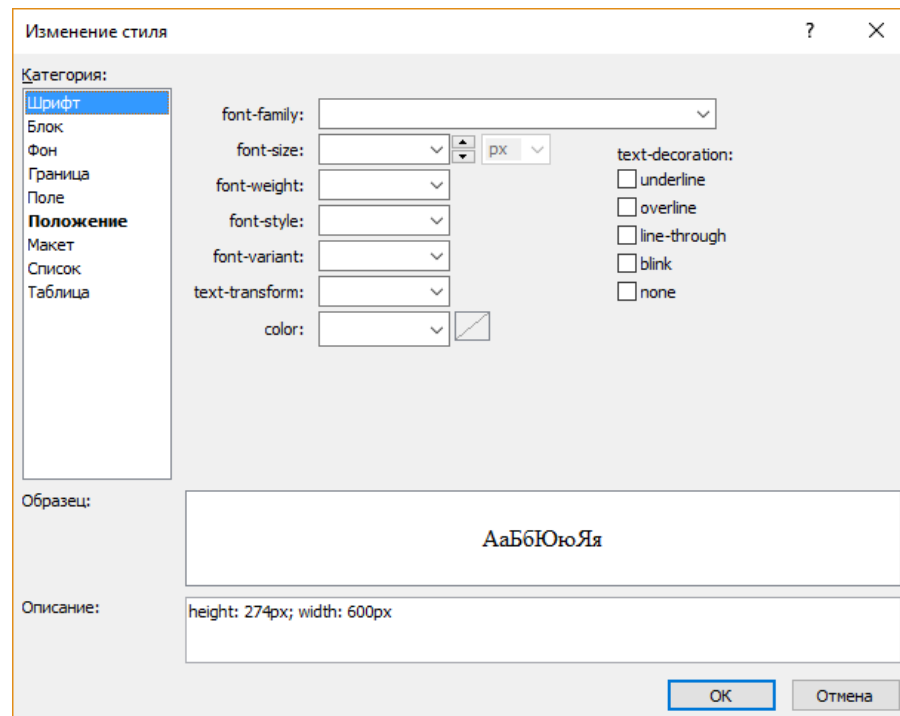


Рисунок 2 – Диалогового окна Modify Style

Созданный подобным образом новый стиль будет сохраняться как внутрискрочный и записываться в отвечающий за стиль атрибут изменяемого элемента. В качестве альтернативы можно определить именованный стиль в текущей странице (как предлагается по умолчанию) или в отдельной таблице стилей.

При желании сконфигурировать элемент HTML как серверный элемент управления для того, чтобы иметь возможность обрабатывать события и взаимодействовать с ним в коде, нужно переключиться в режим Source и добавить в дескриптор элемента управления требуемый для этого атрибут `runat="server"`.

Большая часть кода страницы ASP.NET помещается внутрь обработчиков событий, реагирующих на события веб-элементов управления. С помощью Visual Studio добавить в код обработчик событий можно одним из трех способов:

- ввести его вручную. В этом случае метод добавляется непосредственно в класс страницы. Необходимо указать соответствующие параметры, чтобы сигнатура обработчика событий в точности соответствовала сигнатуре события, которое необходимо обрабатывать. Также понадобится отредактировать дескриптор элемента управления, чтобы он связывал элемент управления с соответствующим обработчиком событий, добавив атрибут `OnИмя_события`;

- дважды щелкнуть на элементе управления в представлении визуального конструктора. В этом случае Visual Studio создаст обработчик события по умолчанию для этого элемента управления (и соответствующим образом настроит дескриптор элемента управления). Например, двойной щелчок на странице приводит к созданию обработчика события `Page.Load`, а двойной щелчок на кнопке – обработчика события `Click`;

- выбрать событие в окне Properties. Выделите элемент управления и щелкните на кнопке с изображением молнии в окне Properties. Отобразится список всех событий, предоставляемых этим элементом управления. После двойного щелчка на поле рядом с событием, которое необходимо обработать, Visual Studio автоматически сгенерирует обработчик события в классе страницы и настроит дескриптор элемента управления.

Visual Studio поддерживает две модели для кодирования веб-страниц:

- внутритекстовый код. Эта модель наиболее близка к традиционной модели ASP. Весь код и HTML-разметка сохраняются в одном файле с расширением `.aspx`. Код вставляется в один или более блоков сценария. Однако, хотя код и находится внутри блока сценария, поддержку функции IntelliSense и возможностей отладки он из-за этого не теряет, да и выполнять его линейным образом сверху вниз (подобно классическому ASP-коду) вовсе не обязательно. Вместо

этого в нем по-прежнему можно реагировать на события элементов управления и использовать подпрограммы. Эта модель удобна, поскольку позволяет хранить все в одном месте безо всяких излишеств и потому часто применяется для кодирования простых веб-страниц;

– отделенный код (code-behind). Эта модель подразумевает создание для каждой веб-страницы ASP.NET двух файлов: файла разметки (.aspx) с дескрипторами HTML и дескрипторами элементов управления, и файла кода (.cs) с исходным кодом страницы (при условии, что для программирования веб-страницы применяется язык C#). Такая модель обеспечивает более удобную схему организации, позволяя отделять пользовательский интерфейс от программной логики, что очень важно при создании сложных страниц.

Чтобы лучше понять, в чем состоит разница между моделями внутритекстового и отделенного кода, рассмотрим пример2.

Пример 2. Отображение текущего времени в текстовой метке и обновление страницы всякий раз, когда выполняется щелчок на кнопке.

В случае использования модели внутритекстового кода эта страница будет выглядеть следующим образом:

```
<% @ Page Language="C#" %>
<!DOCTYPE html>
<script runat="server">
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Текущее время: " + DateTime.Now.ToLongTimeString();
    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Test Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <asp:Button ID="Button1" runat="server" Text="Click Me!" OnClick="Button1_Click" />
        </div>
    </form>
</body>
</html>
```

Приведенные ниже коды файлов WebForm1.aspx и WebForm1.aspx.cs демонстрируют, как эта же страница будет поделена на две части в случае использования модели отделенного кода:

WebForm1.aspx.cs:

```
<% @ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm1.aspx.cs"
Inherits="WebApplication1.WebForm1" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Test Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <asp:Button ID="Button1" runat="server" Text="Click Me!" OnClick="Button1_Click" />
        </div>
    </form>
</body>
</html>
```

```

WebForm1.aspx:
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace WebApplication1
{
    public partial class WebForm1 : System.Web.UI.Page
    {
        protected void Button1_Click(object sender, EventArgs e)
        {
            Label1.Text = "Текущее время: " + DateTime.Now.ToLongTimeString();
        }
    }
}

```

Единственное реальное отличие между примером с внутритекстовым кодом и примером с отделенным кодом состоит в том, что в последнем случае класс страницы больше не является неявным, а наоборот – объявляется как класс, содержащий все методы страницы.

В целом, модель отделенного кода предпочтительнее использовать для сложных страниц. Хотя модель внутритекстового кода является немного более компактной для небольших страниц, с увеличением объема кода и HTML-разметки с ними становится гораздо проще иметь дело по отдельности. Вдобавок модель отделенного кода является более чистой с концептуальной точки зрения, поскольку явно отображает созданный класс и импортированные пространства имен. И, наконец, она также еще позволяет веб-дизайнеру улучшать разметку страниц, не затрагивая код, написанный разработчиком.

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме.
2. Разобрать и выполнить примеры 1-2.
3. Создать отчет, включив в него описание работы по пункту 2, результаты выполнения заданий для самостоятельной работы согласно варианту, код файлов «file_name.aspx» и «file_name.aspx.cs» и ответы на контрольные вопросы.

Задание для самостоятельной работы

Задание 1

Для всех вариантов модифицировать пример 1:

- добавить обработку исключения ввода данных: нечисловых символов, отрицательного числа...;
- добавить обработку исключения существования треугольника (сумма каждой двух сторон должна быть больше длины третьей);
- сделать привлекательным пользовательский интерфейс.

Задание 2

При программировании решения поставленной задачи реализовать требования согласно заданию 1. Предусмотреть создание функциональных элементов для ввода и вывода.

Вариант 1

Написать программу, позволяющую по последней цифре числа определить последнюю цифру его квадрата.

Вариант 2

Написать программу, которая по номеру дня недели (целому числу от 1 до 7) выдает в качестве результата количество уроков в вашей группе в этот день.

Вариант 3

Вводится номер месяца 2018 года. Определить сколько дней в этом месяце.

Вариант 4

Вводится номер месяца. Дать этому месяцу наименование.

Вариант 5

Дан рост трех человек: первого – x см, второго – y см, третьего – z см. Определить самого высокого.

Вариант 6

Вводится номер месяца 2018 года. Определить время года, которому соответствует этот месяц.

Вариант 7

Дан рост трех человек: первого – x см, второго – y см, третьего – z см. Определить самого маленького.

Вариант 8

В трех магазинах один и тот же товар имеет разные цены: в первом магазине, a руб., во втором – b руб., в третьем – c руб. Определить в каком магазине товар самый дорогой.

Вариант 9

В трех магазинах один и тот же товар имеет разные цены: в первом магазине, a руб., во втором – b руб., в третьем – c руб. Определить в каком магазине товар самый дешевый.

Вариант 10

На трех бензоколонках имеется a , b , c литров бензина. На какую бензоколонку вести новую партию бензина?

Контрольные вопросы

1. Из каких частей состоит рабочая область MS Visual Studio?
2. Какие существуют модели для кодирования веб-страниц?
3. Как добавить в код обработчик событий?
4. Как определить свойство или событие функциональных элементов?

Лабораторная работа №18

Взаимодействие с xml-документом

Цель работы: получить теоретические знания и практические навыки при работе с XML-документами при создании web-приложений, используя ЯП - C#.

Теоретические сведения

На сегодняшний день XML является одним из наиболее распространенных стандартов документов, который позволяет в удобной форме сохранять сложные по структуре данные. Поэтому разработчики платформы .NET включили в фреймворк широкие возможности для работы с XML.

«XML документация» или «Документирующие комментарии XML» – это специальные теги XML, которые содержатся в комментариях и описывают свойства или методы в конкретном файле.

Многие web-сайты предоставляют доступ к своему содержимому только вошедшим пользователям (прошедшим проверку подлинности). По умолчанию ASP.NET предоставляет шаблоны проекта веб-сайта, включающие страницы, с помощью которых можно выполнить проверку подлинности.

Для работы с web нужно посмотреть в отдельных компонентах включена ли «LINQ to SQL»

Пример 1: инструкция по созданию web-приложения, взаимодействующая с XML.

Шаг 1 (рисунок 1): Файл – Создать – Проект – Создать – Во вкладке C# «Веб» – «Веб-приложение ASP.NET (.NET Framework)» – Указать расположение и имя.

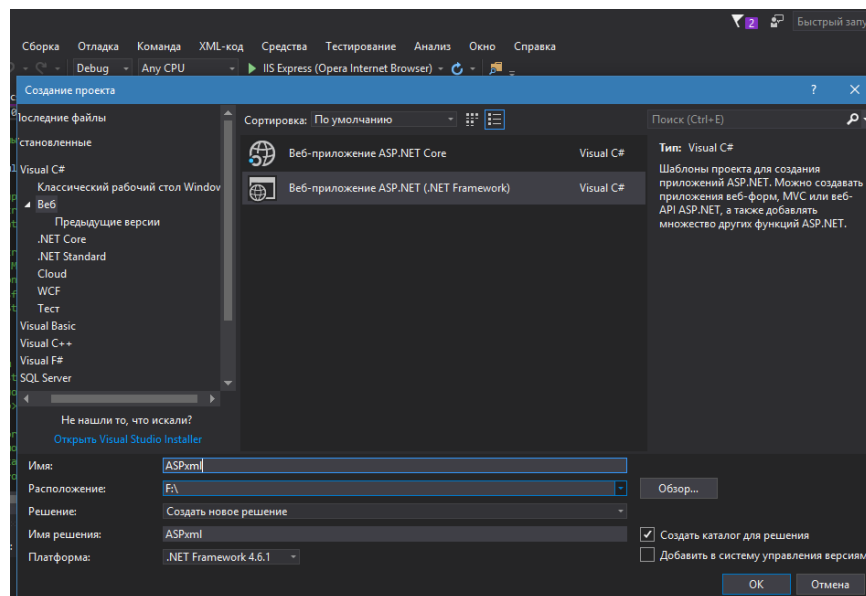


Рисунок 1 – Создание Веб-приложения

После этих действий создастся веб-приложение с содержимым, которые можно увидеть в «Обозревателе решений»

Файл Default.aspx: начальная страница

```
<%@ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="ASPxml._Default" %>
<asp:Content ID="FeaturedContent" ContentPlaceHolderID="FeaturedContent"
runat="server">
</asp:Content>
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
<h3> Загрузка \ Выгрузка БД в\из XML. </h3>
<div>
</div>
</asp:Content>
```

Шаг 2: добавление БД в «Обозревателе решений» ПКМ на App_Data – Добавить – Создать элемент. Во вкладке Данные – База данных SQL Server – название БД «MyDatabase» (рисунок 2).

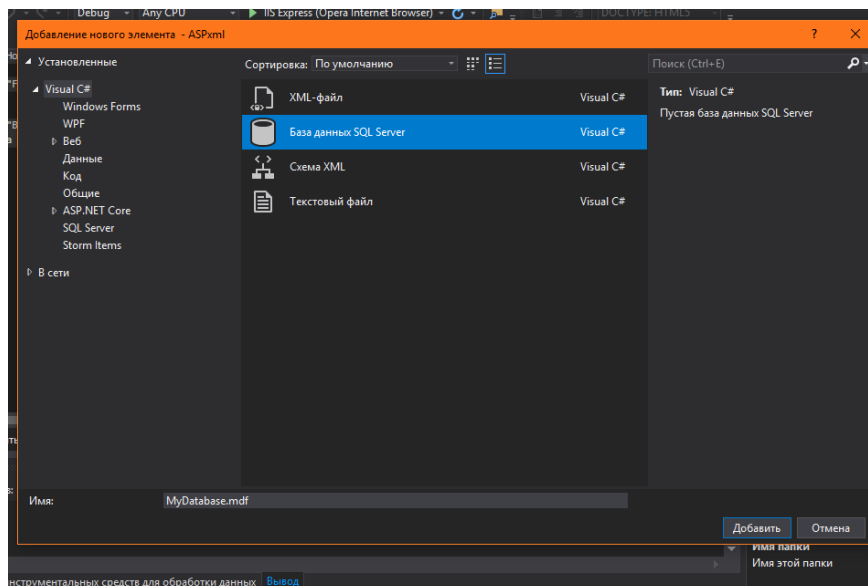


Рисунок 2 – Создание БД

После создания БД можно найти ее в App_Data. По БД левой кнопкой мыши и слева отобразится Обозреватель серверов с созданным сервером.

Шаг 3: создание таблицы для загрузки\выгрузки данных. Папка Таблицы – Добавить новую таблицу (рисунок 3).

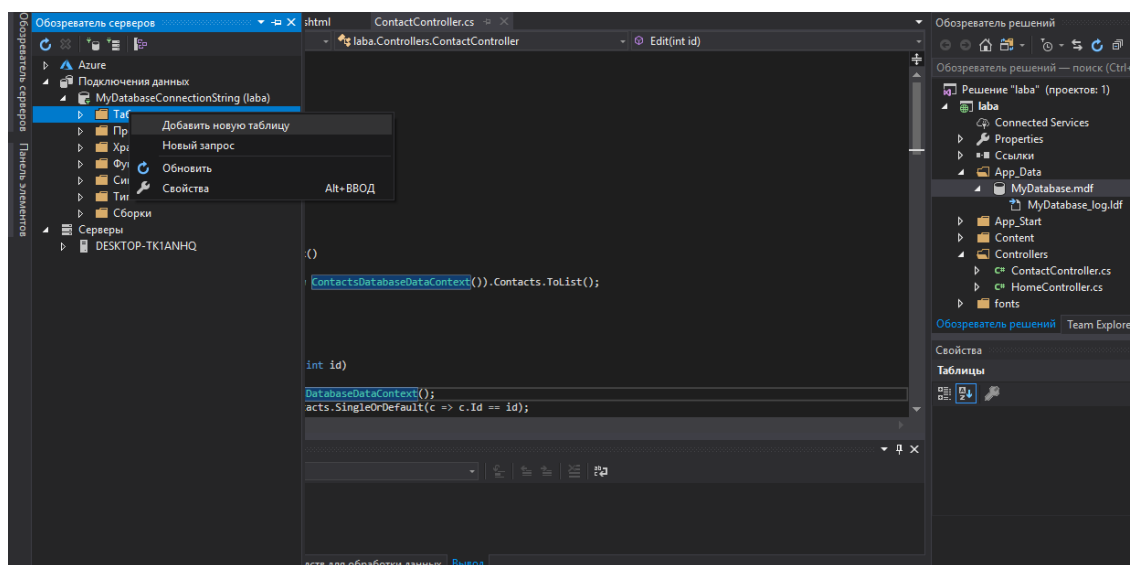


Рисунок 3 – Добавление таблицы

Автоматически имеется уже одно поле Id, нужно поменять его спецификацию индикатора в свойствах. На поле с Id и справа, под обозревателем, появятся его свойства, спецификация индикатора и меняем False на True (рисунок 4).

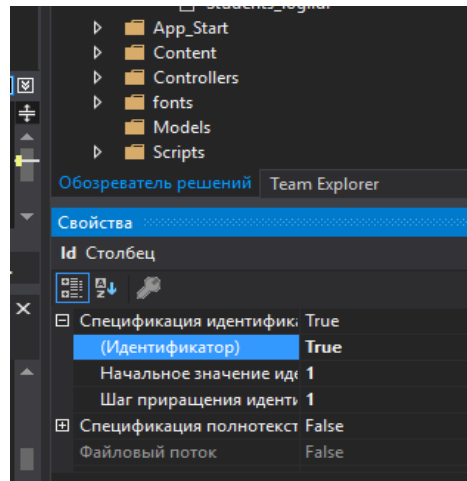


Рисунок 4 – Изменение идентификатора для Id

Необходимо создать другие поля для таблицы, как на рисунке 5. Имя таблицы в T-SQL CREATE TABLE [dbo].[EmployeeMaster] и нажать на кнопку «Обновить» (чуть выше таблицы).

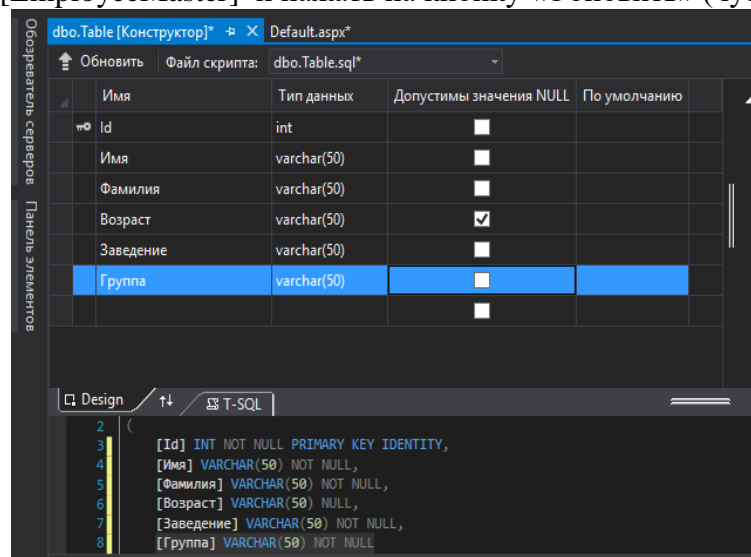


Рисунок 5 – Создание полей для таблицы БД

Шаг 4: Добавление сущности модели данных. ASPxml – добавить – создать элемент... – данные – модель ADO.NET EDM (рисунок 6).

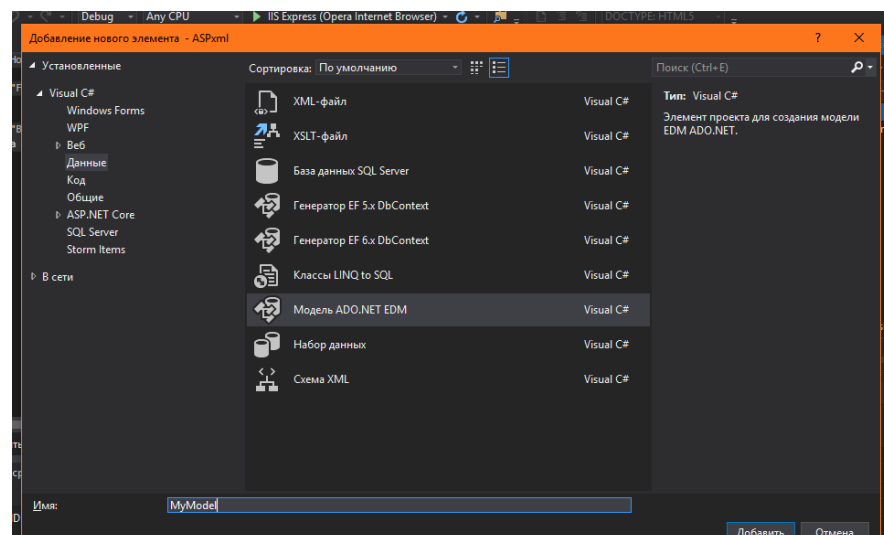


Рисунок 6 – Добавление сущности модели данных

Добавить – кнопка Далее – выбор MyDatabase – отметка галочками (рисунок 7) – Готово.

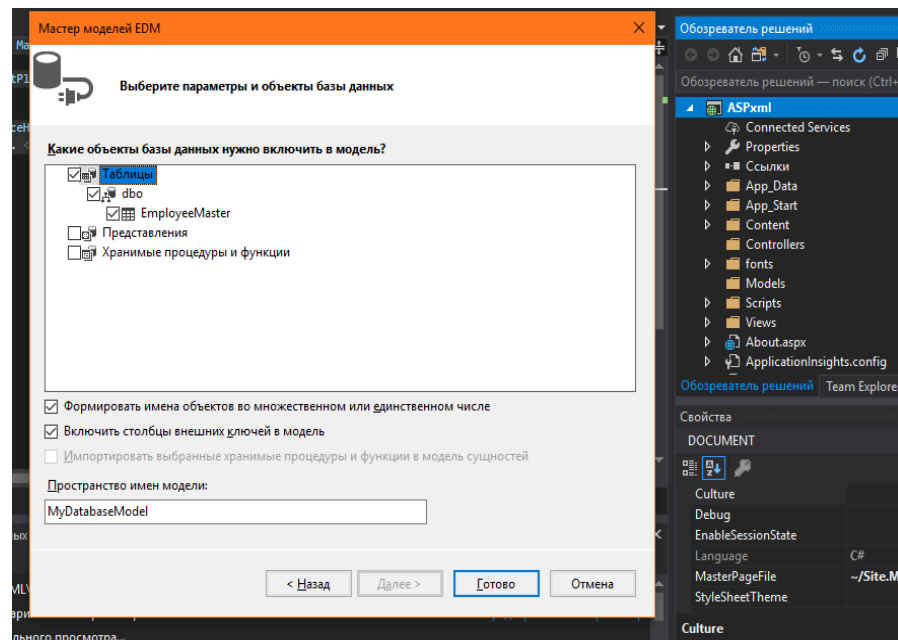


Рисунок 7 – Выборка таблицы

Шаг 5: добавление веб – страницы с кнопками и полями.

Файл Default.aspx: страница с кнопками и полями, благодаря которым можно будет загружать и сохранять XML-документ, а также выгружать с XML-документа данные в таблицу.

```
<% @ Page Title="Home Page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="ASPxml._Default" %>
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <h3> Загрузка \ Выгрузка БД в\из XML. </h3>
    <div>
        <table>
            <tr>
                <td> Выберите файл : </td>
                <td>
                    <asp:FileUpload ID="FileUpload1" runat="server" />
                </td>
                <td>
                    <asp:Button ID="btnImport" runat="server" Text="Загрузить данные"
                    OnClick="btnImport_Click" />
                </td>
            </tr>
        </table>
        <div>
            <br />
            <asp:Label ID="lblMessage" runat="server" Font-Bold="true" />
            <br />
            <asp:GridView ID="gvData" runat="server" AutoGenerateColumns="False"
            AutoGenerateDeleteButton="True" AutoGenerateEditButton="True"
            OnSelectedIndexChanged="gvData_SelectedIndexChanged" DataKeyNames="Id"
            DataSourceID="SqlDataSource1">
                <EmptyDataTemplate>
                    <div style="padding:10px">
                        Данные не найдены!
                    </div>
                </EmptyDataTemplate>
                <Columns>
                    <asp:BoundField HeaderText="Имя" DataField="Имя" SortExpression="Имя" />
                    <asp:BoundField HeaderText="Фамилия" DataField="Фамилия" SortExpression="Фамилия" />
                </Columns>
            </asp:GridView>
        </div>
    </div>
</asp:Content>
```

```

        <asp:BoundField HeaderText="Возраст" DataField="Возраст" SortExpression="Возраст" />
        <asp:BoundField HeaderText="Заведение" DataField="Заведение"
SortExpression="Заведение" />
        <asp:BoundField HeaderText="Группа" DataField="Группа" SortExpression="Группа" />
    </Columns>
</asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%%$
ConnectionStrings:ConnectionString %>" SelectCommand="SELECT * FROM [EmployeeMaster]"
UpdateCommand="UPDATE [EmployeeMaster] Set [Имя]=@Имя, [Фамилия]=@Фамилия,
[Возраст]=@Возраст, [Заведение]=@Заведение, [Группа]=@Группа where [id]=@id"
DeleteCommand="DELETE from [EmployeeMaster] where Id=@Id"></asp:SqlDataSource>
    <br />
    <asp:Button ID="btnExport" runat="server" Text="Выгрузить данные"
OnClick="btnExport_Click" />
</div>
</div>
</asp:Content>

```

Файл Default.aspx.cs: функционал для кнопок.

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Xml.Linq;
namespace ASPxml{
    public partial class _Default : Page{
        protected void Page_Load(object sender, EventArgs e){
            if (!IsPostBack){
                populateDatabaseData();
                lblMessage.Text = "Данные БД действительны "; }
        }
        private void populateDatabaseData(){
            using (MyDatabaseEntities dc = new MyDatabaseEntities()){
                gvData.DataBind();
            }
        }
        protected void btnImport_Click(object sender, EventArgs e){
            if (FileUpload1.PostedFile.ContentType == "application/xml" ||
FileUpload1.PostedFile.ContentType == "text/xml"){
                try{
                    string fileName = Path.Combine(Server.MapPath("~/UploadDocuments"),
Guid.NewGuid().ToString() + ".xml");
FileUpload1.PostedFile.SaveAs(fileName);
                    XDocument xDoc = XDocument.Load(fileName);
                    List<EmployeeMaster> emList = xDoc.Descendants("Employee").Select(d => new
EmployeeMaster{
                        Имя = d.Element("Имя").Value,
                        Фамилия = d.Element("Фамилия").Value,
                        Возраст = d.Element("Возраст").Value,
                        Заведение = d.Element("Заведение").Value,
                        Группа = d.Element("Группа").Value
                    }).ToList();
                }
            }
        }
    }
}

```

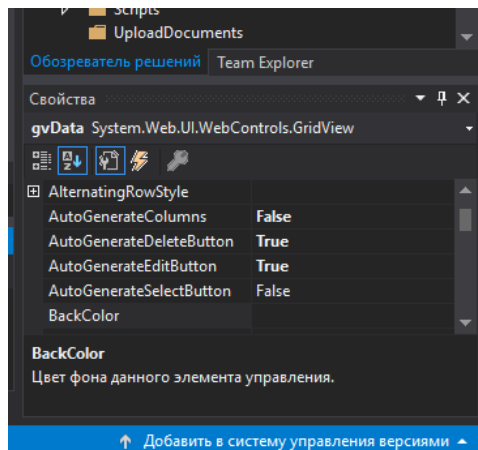



Рисунок 8 – Включение кнопок удалить\ правка

Далее нажатие на стрелочку (справа таблицы) – новый источник данных (рисунок 9).

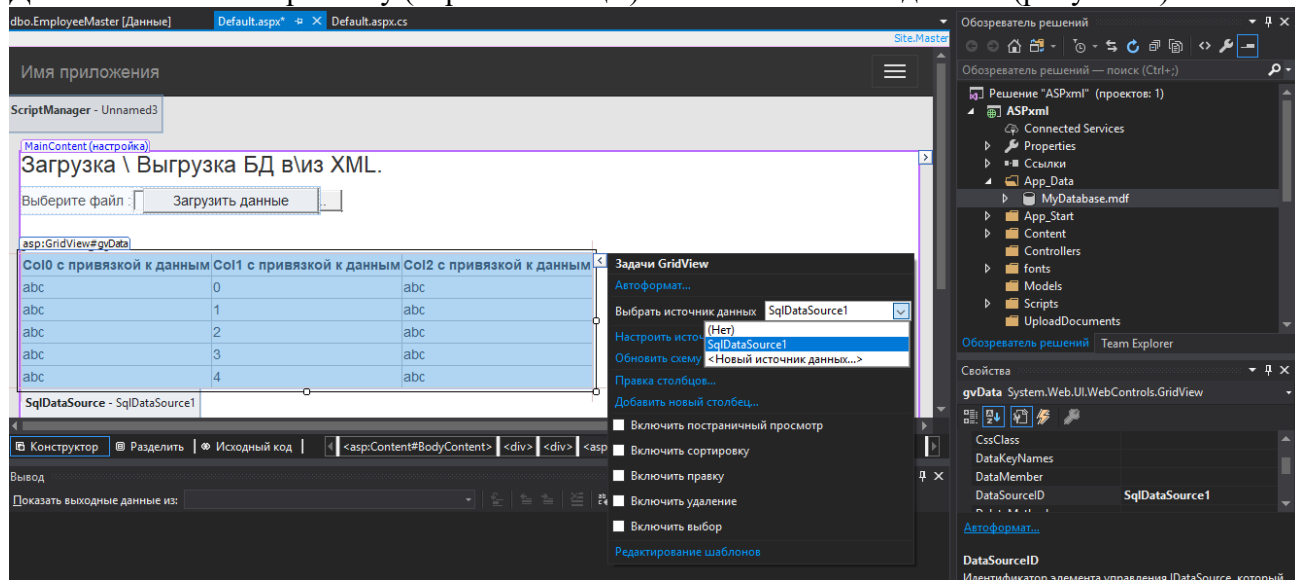


Рисунок 9 – Создание нового источника данных

После база данных – выбрать соединение – далее – выделяем « * » (рисунок 10) – далее – Готово.

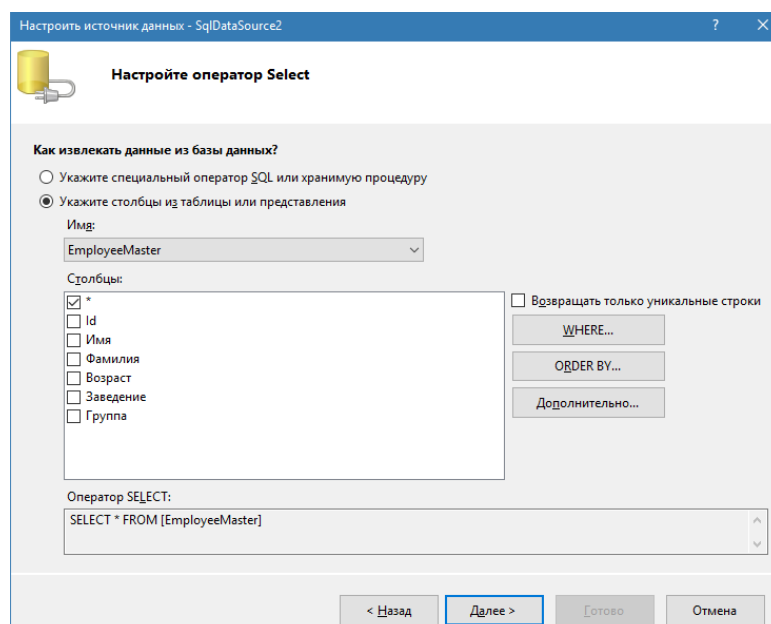


Рисунок 10 – Настройка таблицы

В исходном коде под `</Columns>` добавить:

```
// удаление, редактирование таблиц в бд
</asp:GridView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server" ConnectionString="<%"$
ConnectionString:ConnectionString %">
    SelectCommand="SELECT * FROM [EmployeeMaster]"
UpdateCommand="UPDATE [EmployeeMaster] Set [Имя]=@Имя, [Фамилия]=@Фамилия,
[Возраст]=@Возраст, [Заведение]=@Заведение, [Группа]=@Группа where [id]=@id"
DeleteCommand="DELETE from [EmployeeMaster] where Id=@Id"></asp:SqlDataSource>
<br />
```

Шаг 7: Запуск проекта (рисунок 11).

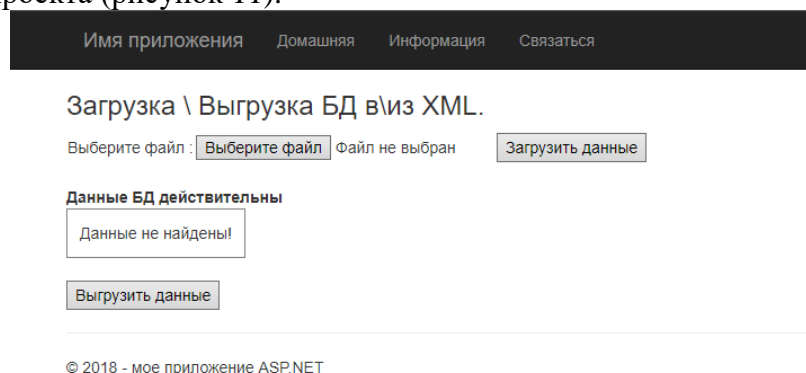


Рисунок 11 – Стартовая Веб-страница

Через таблицу БД создаются данные (рисунок 12): App_Data – MyDatabase – Обозреватель серверов – Таблицы – EmployeeMaster – ПКМ – показать таблицу данных – вводим данные – Enter – запуск проекта (рисунок 13).

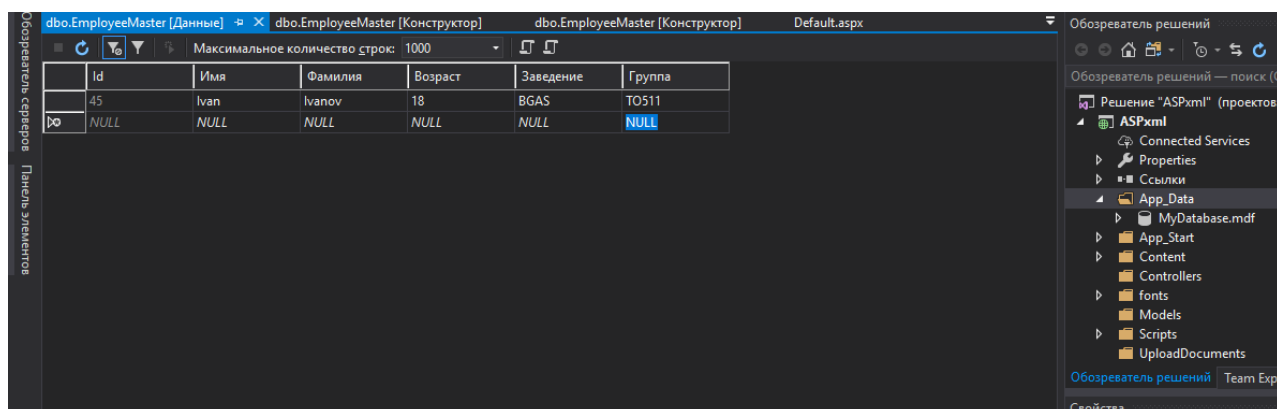


Рисунок 12 – Заполнение полей таблицы

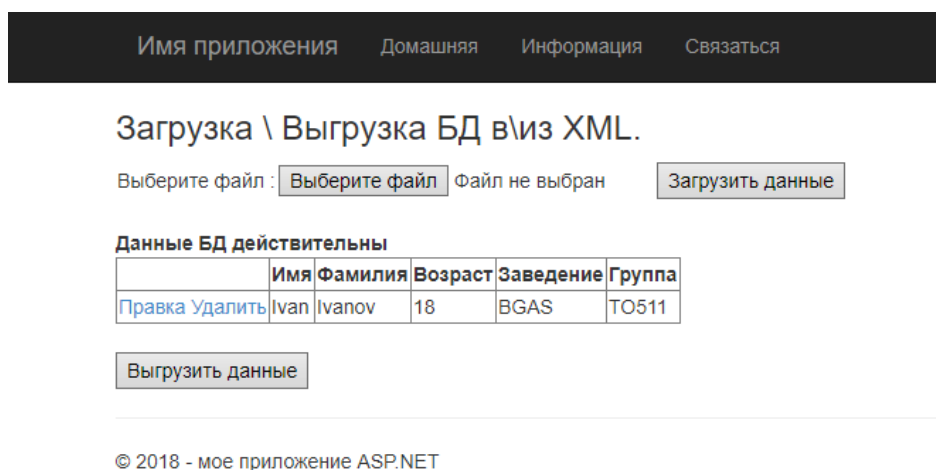


Рисунок 13 – Запущенный проект с созданной таблице

Выгрузить данные – запуск загруженного xml-документа (рисунок 14).

```
<?xml version="1.0"?>
- <Employees>
  - <Employee>
    <Id>45</Id>
    <Имя>Ivan</Имя>
    <Фамилия>Ivanov</Фамилия>
    <Возраст>18</Возраст>
    <Заведение>BGAS</Заведение>
    <Группа>T0511</Группа>
  </Employee>
</Employees>
```

Рисунок 14 – Созданный XML-документ с данными из таблиц

XML-документ можно загрузить в таблицу через кнопки Выберите файл – Загрузить данные. Его можно редактировать как и в таблице на веб-странице, так и в самом документе. При этом данные будут сохраняться в БД.

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме.
2. Разобрать и выполнить пример.
3. Создать отчет, включив в него описание работы по пункту 2, результаты выполнения заданий для самостоятельной работы согласно варианту, код файлов «file_name.aspx» и «file_name.aspx.cs» и ответы на контрольные вопросы.

Задание для самостоятельной работы

Задание 1

Для всех вариантов модифицировать пример 1:

- добавить обработку исключений (3 различных случая);
- сделать привлекательным пользовательский интерфейс.

Задание 2

Добавить в пример 1 возможность:

Вариант 1: сортировки возраста по убыванию.

Вариант 2: добавления человека.

Вариант 3: сортировки фамилий по возрастанию.

Вариант 4: сортировки по группе.

Вариант 5: сортировки по заведению.

Вариант 6: подсчет среднего возраста.

Вариант 7: сортировка имен по возрастанию.

Вариант 8: сортировки по гендерному признаку.

Вариант 9: группировка по возрасту: молодежь, средний возраст, пожилые.

Вариант 10: сортировка по идентификатору по убыванию.

Контрольные вопросы

1. Что собой представляет xml-документ?
2. Перечислите шаги, для создания web-проекта, использующего xml-документ.

Лабораторная работа №19

Средства для работы с БД

Цель работы: получить теоретические знания и практические навыки по созданию и работе с базой данных для web-приложений на ЯП С#.

Теоретические сведения

Шаблон архитектуры Model-View-Controller (MVC) разделяет приложение на три основных компонента: модель, представление и контроллер. Платформа ASP.NET MVC представляет собой альтернативу схеме веб-форм ASP.NET при создании веб-приложений. Платформа ASP.NET MVC является легковесной платформой отображения с широкими возможностями тестирования и, подобно приложениям на основе веб-форм, интегрирована с существующими функциями ASP.NET, например, с главными страницами и проверкой подлинности на основе членства. Платформа MVC определяется в сборке System.Web.Mvc.

MVC представляет собой стандартный шаблон разработки, знакомый многим специалистам. Некоторые типы веб-приложений имеют преимущества при создании на платформе MVC. Для других может быть целесообразно использование традиционной схемы приложения ASP.NET, основанной на веб-формах и обратной передаче. В некоторых случаях возможно сочетание двух подходов: применение одной схемы не исключает использования другой.

Многие веб-сайты предоставляют доступ к своему содержимому только вошедшим пользователям (прошедшим проверку подлинности). По умолчанию ASP.NET предоставляет шаблоны проекта веб-сайта, включающие страницы, с помощью которых можно выполнить проверку подлинности.

Пример 1 – Инструкция по созданию Web-приложения, взаимодействующая с БД.

Файл – Создать – Проект – Создать – «Веб-приложение ASP.NET(.NET Framework)».

Далее появится окно, где нужно выбрать шаблон MVC (Model-View-Controller) и нажать «ОК». После этих действий создастся веб-приложение с содержимым, которые можно просмотреть в «Обозревателе решений».

Перед началом работы необходимо создать БД (рисунок 1). «Обозреватель решений» – App_Data – Добавить – Создать элемент. Во вкладке Веб – Данные – База данных SQL Server – Дать имя БД «MyDatabase».

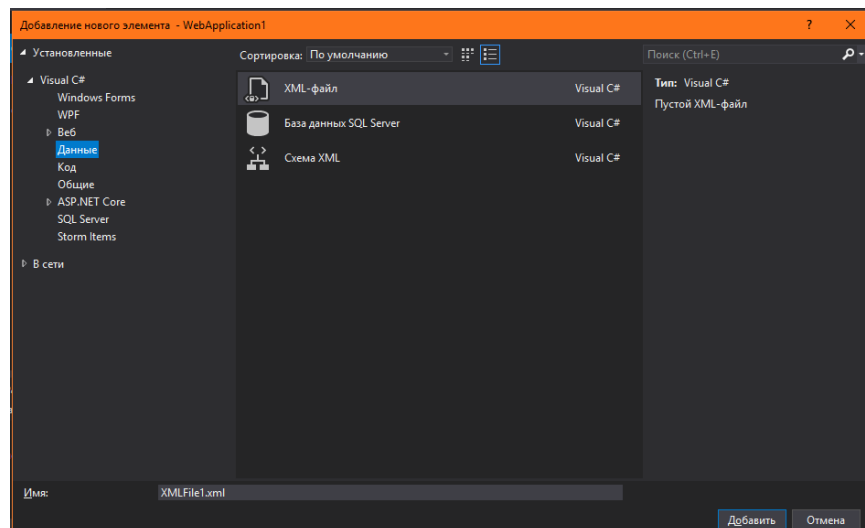


Рисунок 1 – Создание БД

После создания БД можно найти ее в App_Data. Щелкнуть по нашей БД левой кнопкой мыши и отобразится Обзор серверов с созданным сервером. Необходимо добавить таблицу: папка Таблицы – Добавить новую таблицу (рисунок 2).

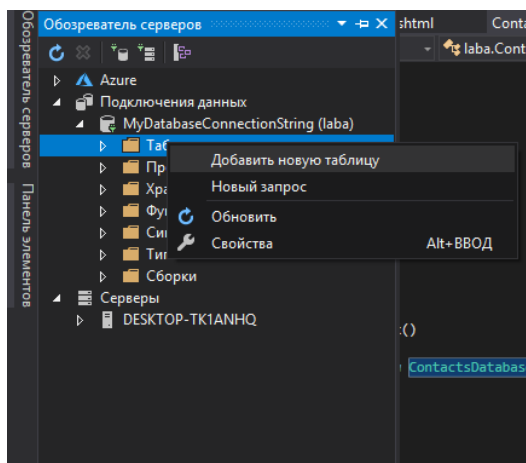


Рисунок 2 – Добавление таблицы

Автоматически имеется уже одно поле Id, нужно поменять его спецификацию индикатора в свойствах. Нажать на поле с Id и справа, под обзором, появятся его свойства. Найти спецификацию индикатора и изменить False на True.

Далее необходимо создать другие поля для таблицы, как на рисунке 3. Изменить имя таблицы в T-SQL на Contacts и чтобы таблица сохранилась, нажать на кнопку «Обновить» (чуть выше таблицы).

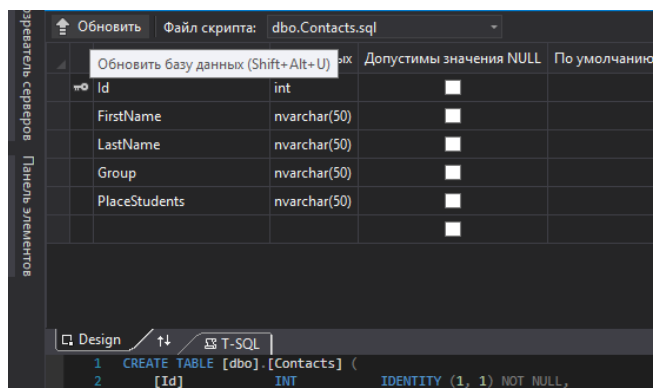


Рисунок 3 – Создание полей для таблицы БД

Открыть Обзоратель серверов – нажать по созданному серверу правой кнопкой мыши – Обновить – перейти к папке Таблицы – таблицу Contacts – Показать таблицу данных. Заполнить таблицу данными.

В обзорателе решений выбрать папку Models – Добавить – Создать элемент – Данные – Классы LINQ to SQL – назвать ContactsDatabase (рисунок 4).

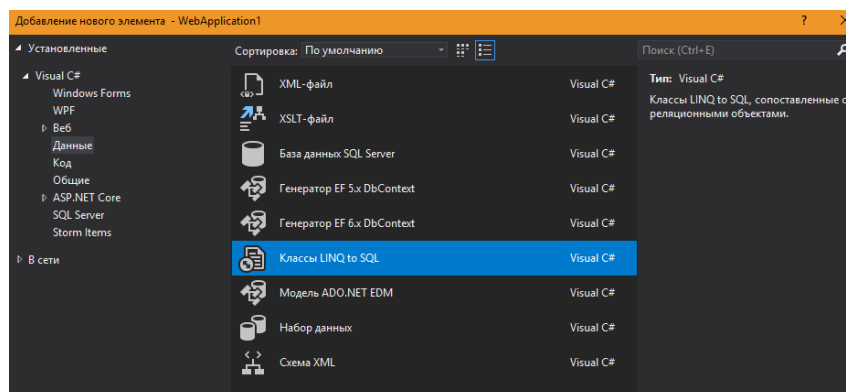


Рисунок 4 – Выборка классов LINQ

С блока «Обзоратель серверов» перетянуть с БД таблицу на открывшееся окно и должен получиться результат, как показано на рисунке 5.

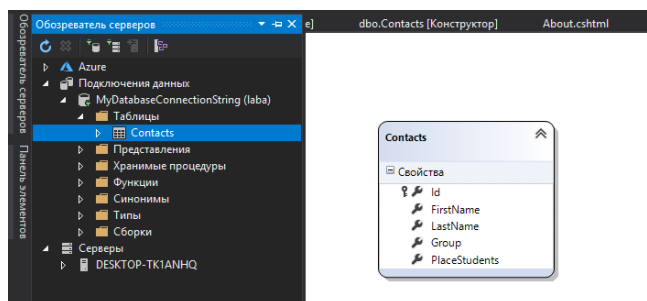


Рисунок 5 – Модель

Далее в обзорателе решений выбрать папку Controllers – Добавить – Контроллер – Контроллер MVC 5 с действиями чтения записи – назвать ContactController. Создать форму таблицы на Web-сайте: папка Controllers ->ContactController.cs:

```
using laba.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
```

```
namespace laba.Controllers{
    public class ContactController : Controller {
        private int id;
        // GET: Contact
        public ActionResult Index() {
            ViewData.Model = (new ContactsDatabaseDataContext()).Contacts.ToList();
            return View(); // Добавить представление, как на рисунке 19.6
        }
        // GET: Contact/Edit/5
        public ActionResult Edit(int id){
```

```
            var db = new ContactsDatabaseDataContext();//Подключение к БД, при появлении
            ошибки перейти Быстрые действия и выбрать using ... .Models
            var contact = db.Contacts.SingleOrDefault(c => c.Id == id);
            ViewData.Model = contact;
            return View();//Добавить представление, в шаблоне выбрать Edit
```

```

    }

    // POST: Contact/Edit/5
    [HttpPost]
    // программирование изменения данных
    public ActionResult Edit(int id, FormCollection collection){
        try{
            // TODO: Add update logic here
            var db = new ContactsDatabaseDataContext();
            var contact = db.Contacts.SingleOrDefault(c => c.Id == id);
            contact.FirstName = collection["Firstname"]; //Обращение к полям
            contact.LastName = collection["Lastname"];
            contact.Group = collection["Group"];
            contact.PlaceStudents = collection["PlaceStudents"];

            db.SubmitChanges();
            return RedirectToAction("Index");
        }
        catch {
            return View();
        }
    }

    // GET: Contact/Delete/5
    public ActionResult Delete(int id){
        var db = new ContactsDatabaseDataContext();
        var contact = db.Contacts.SingleOrDefault(c => c.Id == id);

        ViewData.Model = contact;
        return View(); //Добавить представление, в шаблоне выбираем Delete
    }

    // POST: Contact/Delete/5
    [HttpPost]
    public ActionResult Delete(int id, FormCollection collection) {
        try{
            // TODO: Add delete logic here
            var db = new ContactsDatabaseDataContext();
            var contact = db.Contacts.SingleOrDefault(c => c.Id == id);
            contact.FirstName = collection["Firstname"];
            contact.LastName = collection["Lastname"];
            contact.Group = collection["Group"];
            contact.PlaceStudents = collection["PlaceStudents"];

            db.Contacts.DeleteOnSubmit(contact);
            db.SubmitChanges();
            return RedirectToAction("Index");
        }
        catch {
            return View();
        }
    }
}

```


Рисунок 6 – Добавление представления

Запустить проект, чтобы проверить работает ли форма. В браузере, который установлен по умолчанию, должен открыться сайт с первоначальным шаблоном (рисунок 7).

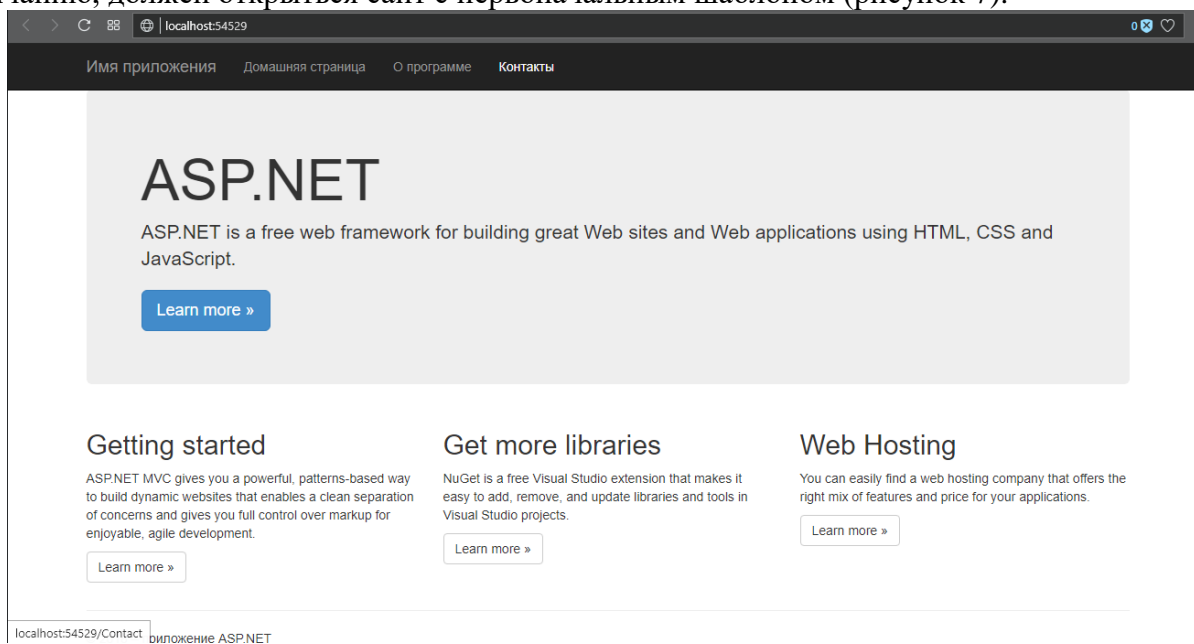


Рисунок 7 – Стартовая страница

Далее необходимо установить ссылку на форму при щелчке по “Контакт”. Перейти в обозреватель решений – папка Views – Shared - _Layout и изменить код:

Папка Views Shared - _Layout: редактирование страницы с формой

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>@ ViewBag.Title – приложение ASP.NET</title>
@Styles.Render("~/Content/css")
@Scripts.Render("~/bundles/modernizr")
</head>
<body>
<div class="navbar navbar-inverse navbar-fixed-top">
<div class="container">
<div class="navbar-header">
```

```

        <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
        </button>
        @Html.ActionLink("Имя приложения", "Index", "Home", new { area = "" }, new {
@class = "navbar-brand" })
    </div>
    <div class="navbar-collapse collapse">
        <ul class="nav navbar-nav">
            <li>@Html.ActionLink("Домашняя страница", "Index", "Home")</li>
            <li>@Html.ActionLink("О программе", "About", "Home")</li>
            <li>@Html.ActionLink("Контакты", "Index", "Contact")</li> //создание ссылки
        </ul>
    </div>
</div>
</div>
<div class="container body-content">
    @RenderBody()
    <hr />
    <footer>
        <p>&copy; @DateTime.Now.Year – приложение ASP.NET</p>
    </footer>
</div>

@Scripts.Render("~/bundles/jquery")
@Scripts.Render("~/bundles/bootstrap")
@RenderSection("scripts", required: false)
</body>
</html>

```

Опять запустить проект и перейти по «Контакты». Отобразится таблица как на рисунке 8.

FirstName	LastName	Group	PlaceStudents
Иван	Иванов	ТО	БГАС

Рисунок 8 – Страница «Контакты» с таблицей

Задание для самостоятельной работы:

Задание 1

Для всех вариантов модифицировать пример 1:

- добавить информация для оставшихся страниц;
- сделать привлекательным пользовательский интерфейс.

Задание 2

Создать web-проект на заданную тематику, состоящий как минимум из 2 страниц и содержащий таблицу данных.

Вариант 1: кулинарная книга.

Вариант 2: список достопримечательностей.

Вариант 3: телефонная книга.

Вариант 4: книжный магазин.

Вариант 5: аптека.

Вариант 6: детский сад.

Вариант 7: ботанический сад.

Вариант 8: таблица Менделеева.

Вариант 9: нобелевские лауреаты.

Вариант 10: специи стран мира.

Контрольные вопросы

1. Опишите шаблон MVC.
2. Перечислите шаги, для создания web-проекта, использующего БД.

Лабораторная работа №20

Современные технологии продвижения web-проекта

Цель работы: применить на практике теоретические навыки раскрутки сайта и продвижения web-ресурса.

Теоретические сведения

Оптимизация сайта – работа с кодом и текстом web-страниц, имеющая целью использование совокупности внутренних факторов, применяемых на странице web-сайта, оказывающих влияние на результат выдачи по данному поисковому запросу.

Начинается оптимизация сайта с выяснения потребностей Клиента, направления его бизнеса, сленговых терминов. После чего с помощью информации, находящейся в открытом доступе, выявляется семантическое ядро запросов, которое затем будет использовано в тексте страниц сайта. Другими словами, подбираются наиболее часто спрашиваемые пользователями ключевые слова и фразы, относящиеся к бизнесу Клиента, и включаются в контент страницы. Анализ сайтов конкурентов также обычно включен в оптимизацию сайтов.

Далее разрабатывается структура сайта, происходит работа с кодом страниц, в том числе и с HTML тегами. Оптимизация сайтов настраивает «внутренности» сайта на восприятие его поисковыми машинами как наиболее подходящего в качестве ответа на данный запрос пользователя.

Продвижение сайтов – комплекс мероприятий, направленных на изменение внешних факторов, влияющих на результат выдачи по данному поисковому запросу или опосредованно связанных с поисковой выдачей. При продвижении сайта выполняются работа с внешним ссылочным ранжированием (получение тематических ссылок с других web-ресурсов, отвечающих выбранной стратегии продвижения сайта, на нужные страницы ресурса Клиента). В результате через некоторое время, после индексации поисковой машиной страниц с такими ссылками, сайт постепенно начинает занимать в поисковой выдаче лидирующие позиции.

Продвижение сайтов возможно и при отсутствии оптимизации сайта, однако в этом случае процесс продвижения сайта не всегда стабилен и надежен, и становится, по крайней мере, более трудоемким, а результат не таким устойчивым – любое изменение поискового алгоритма, меняющего «вес» ссылок, может сместить сайт с первых позиций вниз.

Из опосредованно связанных с поисковой выдачей способов продвижения сайтов можно выделить размещение контекстной рекламы в поисковой машине. При ответе на определенный запрос пользователя поисковая машина вместе с результатами поисковой выдачи покажет и рекламное объявление Клиента, заказанное им под этот запрос. Данный метод продвижения сайтов обычно применяют в случае необходимости получить целевых посетителей на сайт без временных затрат, характерных для продвижения сайта и до того, как сайт займет надлежащее место в выдаче по этому запросу. Наибольший минус этого метода в том, что доверие пользователей (и количество заходов на сайт) на порядок больше к поисковой выдаче, чем к результатам контекстной рекламы.

6. Раскрутка сайтов - привлечение посетителей на сайт Заказчика способами, не связанными с результатами выдачи поисковых машин по данному запросу. В категорию «раскрутка сайта» (другое название – непоисковое продвижение сайта) относят все способы рекламы сайта, которые не связаны с получением проиндексированных поисковой машиной текстовых ссылок на сайт или поисковой выдачей.
7. Чтобы привлечь больше посетителей, надо увеличить посещаемость сайта. Поскольку основная масса посетителей ищет информацию через поисковые системы, то успех привлечения посетителей во многом зависит от того, насколько страницы сайта успешно оптимизированы для попадания на первые строчки поисковых систем по наиболее востребованным ключевым словам.
8. Понятие «раскрутка сайта» предполагает:
 - анализ поведения посетителей;
 - SEO-продвижение (Search Engine Optimization) или поисковая оптимизация, то есть приведение ресурса в состояние, которое обеспечит его заметность для поисковых систем.

9. – размещение графических баннеров на сайтах, ведение тематических рассылок, размещение пресс-релизов, рекламирование ресурса вне Интернета;
– повышение юзабилити (удобства использования ресурсом);
10. – развитие разделов: статьи и классификаторы, глоссарий, новостной раздел, каталог сайтов, доска объявлений, форум;
11. – создание и индексация страниц магазина (товарного раздела, раздела бренда, страниц товара, страниц заказа и других вспомогательных страниц);
12. – разделение контента по доменам и субдоменам;
– использование e-mail маркетинга;
– использование маркетинга в социальных сетях (SMM).

Баннер – рекламный графический блок, связанный гиперссылкой с рекламируемым web-сайтом или страницей (переход по гиперссылке называется «переход по баннеру» или «клик»). Форма рекламного обращения в Интернете – наиболее распространённая на сегодняшний день. Выглядит как прямоугольная картинка или текст. Важен размер баннера, от которого зависит скорость его загрузки и, значит, вероятность попадания его в поле зрения потребителя.

Есть два основных пути размещения баннерной рекламы:

- индивидуальные договорённости с конкретными сайтами (платные или на основе взаимного обмена баннерами);
- обращение к услугам агентства интернет-рекламы, которое предложит размещение на целом ряде сайтов.

Порядок выполнения работы

1. Изучить краткие теоретические сведения по теме.
2. Создать отчет, включив в него результаты выполнения заданий для самостоятельной работы и ответы на контрольные вопросы.

Задание для самостоятельной работы:

Создать макет web-сайта с учетом юзабилити. Провести комплекс мероприятий по продвижению созданного ресурса:

- аналитика;
- SEO-продвижение;
- создать E-mail-рассылку;
- другие инструменты.

Предложить варианты для дальнейшей раскрутки web-сайта.

Контрольные вопросы

1. Что такое оптимизация?
2. Какие мероприятия входят в раскрутку web-сайта? Охарактеризуйте их.