Федеральное государственное образовательное учреждение высшего профессионального образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

Кафедра вычислительных машин, комплексов, систем и сетей

Н.И. РОМАНЧЕВА

назад

ИНТЕРНЕТ-ТЕХНОЛОГИИ в ГА

ПОСОБИЕ

к выполнению лабораторных работ №3,4

для студентов 5 курса
специальности 220100
дневного обучения

Москва- 2005

Рецензент доктор техн. наук, доцент А.А.Егорова

Романчева Н.И.

ИНТЕРНЕТ-ТЕХНОЛОГИИ В ГА: Пособие к выполнению лабораторных работ № 3,4. - М.: МГТУ ГА, 2005.- 40 с.

Данное пособие издается в соответствии с учебным планом для студентов специальности 220100 дневного обучения.

Рассмотрены и одобрены на заседаниях кафедры 11.05.2005г. и методического совета 11.05.2005 г.

С О Д Е Р Ж А Н И Е

1 ОСНОВНЫЕ ТРЕБОВАНИЯ И ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Настоящее пособие предназначено для студентов специальности 220100, выполняющих лабораторные работы по дисциплине "Интернет-технологии в ГА". В пособие включены материалы по лабораторным работам № 3,4.

Продолжительность каждой лабораторной работы - 4 часа.

Целью проведения лабораторных работ является закрепление основных теоретических положений, изложенных в лекциях на примере широко используемых в ГА технологий PHP и ASP.NET.

В процессе выполнения лабораторных работ студенты должны освоить приемы и методы:

- работы с PHP-машиной,
- сервисами и продуктами на платформе .Net .

Лабораторная работа состоит из следующих этапов:

1) домашняя подготовка;

2) выполнение работы на компьютере в соответствии с заданием;

3) сдача выполненной работы преподавателю на персональном компьютере;

4) распечатка результатов работы на принтере;

5) оформление отчета;

5) защита лабораторной работы.

В процессе домашней подготовки студент:

- изучает лекционный материал, материалы по темам данного пособия и дополнительной литературы,
- знакомится с заданием на выполнение лабораторной работы;
- готовит отчет по выполнению лабораторной работы (пункты, отмеченные знаком *).

Выполнение лабораторной работы производится во время занятий в классе ИВЦ МГТУГА в присутствии преподавателя. В процессе выполнения лабораторной работы студент последовательно выполняет задание. По завершению работы - демонстрирует преподавателю результаты.

Сдача работы преподавателю на персональном компьютере заключается в демонстрации выполненной работы и выполнении непосредственно при преподавателе индивидуального дополнительного задания.

После приема преподавателем лабораторной работы на ПК студент:

- сохраняет результаты лабораторной работы на дискете, выданной преподавателем, в каталоге со своей фамилией;
- распечатывает результаты на принтере на подготовленных листах формата А4.

Отчет по каждой лабораторной работе должен содержать:

- название работы*;
- цель лабораторной работы*;
- задание на выполнение лабораторной работы*;
- краткие комментарии по выполнению лабораторной работы*;
- распечатки файлов результатов, подписанные преподавателем.

Защита лабораторной работы преподавателю проводится по контрольным вопросам и при наличии оформленного отчета (распечатки должны быть приклеены). После защиты лабораторной работы делается соответствующая запись на отчете студента.

В соответствии с Положением МГТУ ГА о зачетах и курсовых экзаменах студент, не защитивший 2-х работ, не допускается к выполнению следующей лабораторной работы.

2 ЛАБОРАТОРНАЯ РАБОТА №3
СОЗДАНИЕ СЦЕНАРИЯ ФОРУМА И ЭЛЕМЕНТОВ СИСТЕМЫ ФОРМИРОВАНИЯ ЗАКАЗОВ ЧЕРЕЗ INTERNET

2.1 Цель лабораторной работы

Получение навыков работы с языком серверных скриптов –PHP, создание сценария форума и элементов системы формирования заказов через Internet.

2.2 Задание на выполнение лабораторной работы

1) Разработать программу и сценарий форума с использованием PHP-кода, встроенного в HTML-код, позволяющую любому пользователю:

- открыть тему для обсуждения,
- высказать свое мнение по уже открытой теме.

Предусмотреть проверку вводимой пользователем информации, если название темы или сообщение не введены, то выполнение сценария завершается, и пользователю выдаются сообщения об ошибке с возможностью вернуться с помощью кнопки «Назад».

База данных должна создаваться динамически, на основе простых текстовых файлов: в файл tems.txt записывать название тем, в файл nN.txt записывать количество сообщений по теме с номером N, в файл mN.txt записывать сообщения по теме с номером N.

Предусмотреть:

- переменную, определяющую высоту окна компонента выбора тем. Если количество тем не превышает заданной величины, например, 5, то высота определяется количеством тем, если превышает значение 5, высота остается равной 5. При этом должна появляться полоса вертикальной прокрутки;

- HTML-форму с окном Select и кнопкой Submit для передачи содержимого формы в файл look.php, управляющий просмотром содержания дискуссий по темам,

- кнопку Submit «Отправить в форум» для передачи содержимого формы в файл add.php

Управляющие файлы index.php, look.php, add.php разместить в папке forum.

2) Разработать программу, содержащую элементы системы формирования заказов через Internet:

- включить проверку корректности адреса электронной почты,
- заказ отправляется в виде текста, с возможностью проверки и дополнения;
- заказ передается на сервер в базу данных последовательно;
- если по каким-либо причинам заказ оказался незавершенным, по истечении определенного промежутка времени заказ аннулируется;
- оперативные данные о покупателях хранить в текстовом файле list.txt, завершенные заказы заносить в файл order.txt;
- каждый клиент связан со своей сессией, каждая сессия, соответственно, содержит информацию о поведении своего клиента.

Управляющие файлы index.php, look.php, send.php разместить в папке zakaz.

2.3 Основные теоретические сведения

Под форумом понимается программа, позволяющая любому желающему открыть тему для обсуждения, а также высказать свое мнение по уже открытой теме. Разрабатываемый в данной лабораторной работе форум сделан простым, но достаточно функциональным. Он состоит из 4-х файлов управления. База данных создается динамически, на основе простых текстовых файлов.

В общем виде синтаксис определения функции PHP следующий:

function имя_функции(арг1[=зн1], арг2[=зн2], ... аргN[=знN])

{ операторы_тела_функции; }

Имя функции должно быть уникальным с точностью до регистра букв. Это означает, что, во-первых, имена MyFunction, myfunction и даже MyFuNcTiOn будут считаться одинаковыми, и, во-вторых, мы не можем переопределить уже определенную функцию (стандартную или нет — не важно), но зато можем давать функциям такие же имена, как и переменным в программе (конечно, без знака $ в начале). Список аргументов, состоит из нескольких перечисленных через запятую переменных, каждую из которых необходимо задать при вызове функции (впрочем, когда для этой переменной присвоено через знак равенства значение по умолчанию (обозначенное =знМ), ее можно будет опустить. Конечно, если у функции не должно быть аргументов (как это сделано у функции time()), то следует оставить пустые скобки после ее имени, например:

function SimpleFunction() { ... }

В фигурные скобки заключается тело функции. В нем могут быть любые операторы, включая даже операторы определения других функций (правда, эти "другие функции" не будут локальными, а станут далее "видны" для всей программы, но только с того момента, как до их описания дойдет управление). Если функция должна возвращать какое-то значение, что среди них должен встретиться оператор return. Если же она должна отработать без возврата значений, то оператор return можно и не указывать (или указывать без задания возвращаемого значения).

2.3.1 Базовые функции

1) int strlen(string $st)

Одна из наиболее полезных функций. Возвращает просто длину строки, т. е., сколько символов содержится в $st. Строка может содержать любые символы, в том числе и с нулевым кодом (что запрещено в Си). Функция strlen() будет правильно работать и с такими строками.

2) int strpos(string $where, string $what, int $fromwhere=0)

Данная функция выполняет поиск в строке $where подстроки (то есть последовательности символов) $what и в случае успеха возвращает позицию (индекс) этой подстроки в строке. Первый символ строки, как и в Си, имеет индекс 0. Необязательный параметр $fromwhere можно задавать, если поиск нужно вести не с начала строки $from, а с какой-то другой позиции. В этом случае следует эту позицию передать в $fromwhere. Если подстроку найти не удалось, функция возвращает false. Однако, проверяя результат вызова strpos() на false - используйте для этого только оператор =.

3) string substr(string $str, int $from [,int $length])

Назначение данной функции — возвращать участок строки $str, начиная с позиции $start и длиной $length. Если $length не задана, то подразумевается подстрока от $start до конца строки $str. Если $start больше, чем длина строки, или же значение $length равно нулю, то возвращается пустая подстрока. Однако эта функция может делать и еще довольно полезные вещи. К примеру, если мы передадим в $start отрицательное число, то будет считаться, что это число является индексом подстроки, но только отсчитываемым от конца $str (например, -1 означает "начиная с последнего символа строки"). Параметр $length, если он задан, тоже может быть отрицательным. В этом случае последним символом возвращенной подстроки будет символ из $str с индексом $length, определяемым от конца строки.

4) int strcmp(string $str1, string $str2)

Сравнивает две строки посимвольно (точнее, побайтово) и возвращает: 0, если строки полностью совпадают; -1, если строка $str1 лексикографически меньше $str2; и 1, если, наоборот, $str1 "больше" $str2. Так как сравнение идет побайтово, то регистр символов влияет на результаты сравнений.

5) int strcasecmp(string $str1, string $str2)

То же самое, что и strcmp(), только при работе не учитывается регистр букв. Например, с точки зрения этой функции "ab" и "AB" равны.

2.3.2 Открытие файла

Как и в Си, работа с файлами в PHP разделяется на три этапа. Сначала файл открывается в нужном режиме, при этом возвращается некое целое число, служащее идентификатором открытого файла (дескриптор файла). Затем настает очередь команд аботы с файлом (чтение или запись, или и то и другое), причем они "привязаны" уже к дескриптору файла, а не к его имени. После этого файл лучше всего закрыть (хотя это можно и не делать, поскольку PHP автоматически закрывает все файлы по завершении сценария). Например,

int fopen(string $filename, string $mode, bool $use_include_path=false)

В приведенном примере функция открывает файл с именем $filename в режиме $mode и возвращает дескриптор открытого файла. Если файл не был открыт, то, как это принято, fopen() возвращает false.

Необязательный параметр $use_include_path говорит PHP о том, что, если задано относительное имя файла, его следует искать также и в списке путей, используемом инструкциями include и require. Обычно этот параметр не используют.

Параметр $mode может принимать следующие значения:

- r r — файл открывается только для чтения. Если файла не существует, вызов регистрирует ошибку. После удачного открытия указатель файла устанавливается на его первый байт, т. е. на начало;

- r r+ — файл открывается одновременно на чтение и запись. Указатель текущей позиции устанавливается на его первый байт. Как и для режима r, если файла не существует, возвращается false. Следует отметить, что если в момент записи указатель файла установлен где-то в середине файла, то данные запишутся прямо поверх уже имеющихся, а не "раздвинут" их, при необходимости увеличив размер файла;

- r w — создает новый пустой файл. Если на момент вызова уже был файл с таким именем, то он предварительно уничтожается. В случае неверно заданного имени файла вызов не выполняется;

.- r w+ — аналогичен r+, но если файла изначально не существовало, создает его. После этого с файлом можно работать как в режиме чтения, так и записи. Если файл существовал до момента вызова, его содержимое удаляется;

- r a — открывает существующий файл в режиме записи, и при этом сдвигает указатель текущей позиции за последний байт файла. Этот режим полезен, если требуется что-то дописать в конец уже имеющегося файла. вызов неуспешен в случае отсутствия файла;

- r a+ — открывает файл в режиме чтения и записи, указатель файла устанавливается на конец файла, при этом содержимое файла не уничтожается. Отличается от значения *a* тем, что если файла изначально не существовало, то он создается. Этот режим полезен, если необходимо дописать в файл (например, в журнал), но не известно, создан ли уже такой файл.

Кроме этого, в конце любой из строк r, w, a, r+, w+ и a+ может находиться еще один необязательный символ — b или t. Если указан b (или не указан вообще никакой), то файл открывается в режиме бинарного чтения/ записи. Если же это t, то для файла устанавливается режим трансляции символа перевода строки, т. е. он воспринимается как текстовый.

2.3.3 Конструкция foreach

Данный оператор цикла является новым, и используется только начиная с PHP 4. Аналогичный оператор можно найти в Perl. Он призван упростить последовательную обработку всех элементов массива. Существуют две системы синтаксиса для этого оператора:

Foreach (array_expression as $value) выражения

Foreach (array_expression as $key => $value) выражения

В первом случае значения переменных массива по очереди присваиваются переменной $value, над которой производятся действия. В итоге получается своеобразный указатель на то или иное значение переменной из необходимого массива.

Второй способ аналогичен первому, только в этом случае помимо значения переменной будет еще использоваться и ключ, т. е. обозначение этой переменной.

2.3.4 Классы объектов в PHP

Более сложным структурным компонентом программы (по сравнению с функцией) является объект. Для примера будем рассматривать класс объектов как расширение понятия функции. У функции в качестве аргументов были переменные, и то необязательно. У класса объектов в качестве аргументов могут быть тоже переменные (их обычно называют свойствами объекта класса). Но у класса объектов в качестве аргументов имеются и функции (их обычно называют методами). Упрощенно можно сказать, что класс объектов — это связанная совокупность переменных и функций,

В web-документах часто приходится применять различные формы для ввода пользователем данных. После щелчка на кнопке отправки данных данные отправляются в файл-сценарий и там обрабатываются. Рассмотрим простейшую форму с названием поля ввода, самим полем ввода и кнопкой для отправки данных. Класс таких объектов представлен ниже:

```
<?php
class input
{
        var $nam,$act;
        function form_param($nam,$act)
        {
   $this-> nam = $nam;
   $this-> act = $act;
        }
   function form_input()
      {
      echo "<form action=$this->act>
      Input <input type='text' name=$this->nam>
```

```
                <input type='submit'value='Send'>
                </form>";
        }
          }
$myinput = new input;
$myinput->form_param('user','script.php');
$myinput->form_input();
```

После ключевого слова class во второй строке стоит имя, которое дается этому классу, например, input. Далее в операторных скобках (третья и предпоследняя строки) пишется собственно класс объектов простой формы ввода данных. Наш класс состоит из двух переменных и двух функций. Переменными обозначим имя поля ввода — $nam и имя сценария, куда введенные данные будут отправляться для обработки — $act. Имена переменных введены через запятую, после ключевого слова var. Первая функция form_param() служит для приема значений переменных $nam и $act извне. Синтаксическое построение $this->nam = $nam упрощенно можно перевести так: в этом классе переменной пат присваивается введенное значение переменной $nam. Аналогично и для второй строки.

Следующая функция formjnputQ генерирует в веб-документе простую форму с именем поля ввода, равным значению nam. Action для формы будет равно значению act. Синтаксическое построение $this->act означает, что в качестве имени action берется значение переменной act этого же класса.

Чтобы получить форму в веб-документ и на экран, нужно, во-первых, создать конкретный объект нашего класса и, во-вторых, вызвать этот объект. Дополним предудущий сценарий следующими тремя строками в конце:

```
Smylnput = new Input;
$myinput-
>fonn_param('user'.'script.php'):
$my1nput->forin_1nput():
```

$myinput — это имя конкретного объекта (экземпляр объекта класса) из класса input, оператор new — это специальный оператор-конструктор, создающий конкретный экземпляр объекта класса.

Вторую строку можно перевести так: значения переменных $nam и $act следует принять равными user и script.php соответственно. И наконец, третья строка — сгенерируй форму. Это указания PHP-интерпретатору. Изменяя значения переменных $nam и $act во второй строке, можно получить-формы ввода с разными параметрами. У всех форм название поля ввода будет Input

2.3.5 Примеры сценариев

В приведенных примерах сценариев пронумерованы строки, которые относятся к PHP-коду. Непронумерованные строки представляют собой простой HTML-код.

Листинг файла index.php представлен ниже.

```
<html><head><title>НАЗВАНИЕ ФОРУМА</title></head>
<body bgcolor="#c0c0c0"><center>
<table height="100%" width="100%"><tr><td align=center>
<table BGCOLOR="f0f8ff" width="90%"><tr>
<td width="20%"></td>
<td width="60%" align=center><b>
<font face="verdana" color="#ff0000">НАЗВАНИЕ ФОРУМА</td>
<td align=right><a href="#">К сайту</a>
</td></tr></table>
<table BGCOLOR="f0f8ff" width="90%"><tr>
<td align=center>
<H5><font face="verdana">
<?php
if(file_exists("tems.txt")){                    1-ая строка
$t=@file("tems.txt");                           2
echo "<br>Темы обсуждения, их ".count($t);      3
if(count($t)<5) $size=count($t);else $size=5;   4
```

```
?>
</h5></td></tr><tr><td align=center>
<form action="look.php" method="POST">                5
<select name="t[]" size="<?php echo $size;?>">        6
<?php
for($i=0;$i<count($t);$i++){                          7
echo "<option value=".$i.">".$t[$i];                  8
}
$kt = count($t);                                      9
echo"</select><br><br><input type='submit'            10
        value='Посмотреть'></form>";
}else{                                                11
$kt=0;                                                12
echo"<br>НЕТ ТЕМ!";                                   13
}
?>
</td></tr><tr><td align=center><br>
<h5><font face="verdana">Новая тема - давайте обсудим
следующее:</h5>
</td></tr>
<tr><td align=center>
<form action="add.php" method="POST">
<font>Тема (до 50 зн.) <input name="tema" type="text" SIZE=50
MAXLENGTH=55>
</td></tr></table>
<table BGCOLOR="f0f8ff" width="90%"><tr><td align=center>
Имя (до 30 зн.) <input name="nam" type="text" SIZE=25
MAXLENGTH=35></td>
<td align=center>Email <input name="mail" size=25 type="text"></td>
</tr></table>
```

```
<table BGCOLOR="f0f8ff" width="90%"><tr><td align=center>
Текст базового сообщения (до 900 зн.)<br>
<textarea name="mes" rows=10 cols=60 ></textarea>
</td><tr><td align=center><br>
<input name="N" type="hidden" value="<?php echo $kt;?>">          14
<input name="add" type="hidden" value="ts">                       15
<input type="reset" value="Очистить">
<input type="submit" value="Отправить в форум">
</form></td></tr></table></td></tr></table></body></html>
```

Файл index.php - если тем обсуждения нет, то условие строки 1 (file_exists(«tems.txt»)) не выполняется, и строки кода до 11-й пропускаются PHP-интерпретатором. В 12-й строке объявляется переменную — количество тем ($kt) и присваиваем ей значение 0. В 13-й строке выводится сообщение об отсутствии тем, которое вписывается в HTML-код. В 14-й строке скрытой переменной N присваивается, с помощью PHP-вкрапления в HTML-код, значение, равное количеству тем ($kt), то есть 0. Если условие в строке 1 выполняется, считываем названия тем в массив $t, и тогда в строке 3 выводим на экран сообщение о наличии тем и их количестве. count($t) - количество элементов массива $t в точности совпадает с количеством тем. В строке 6 указываем, что имя компонента Select является массивом. Это потребуется для простой и корректной передачи выбранного значения из Select в PHP-сценарий. Высоту окна устанавливаем равной значению $size, сформированного в 4-й строке. В 7-й и 8-й строках с помощью цикла создаем окно Select и заполняем его содержимым файла tems.txt. В качестве значения для составляющих Option устанавливаем номер шага цикла. В данном случае этот номер будет соответствовать индексу элемента массива. Напомним, что индексация элементов массива начинается с нуля.В 9-й строке устанавливаем значение переменной количества тем ($kt) равным размерности массива count($t). Кнопка Submit со значением «Отправить в форум» в нижней форме отправляет к файлу-сценарию add.php, передавая ему переменную- удостоверение с именем add.

Кнопка Submit со значением «Отправить в форум» отправляет к файлу-сценарию add1.php, передавая ему переменную с именем add.

Листинг файла add1.php представлен ниже.

```
<?php
    if ((isset($_POST['add']))&&($_POST['add']=="t")){
echo"<html><title>Добавление по теме</title><body>";
if (!empty($_POST['nam']))$tn=$_POST['nam'];else $tn='MisterX';
if (!empty($_POST['mail']))$te=($_POST['mail']);else $te='не введён';
if (!empty($_POST['mes']))$t2=($_POST['mes']);else {
exit("<table cellspacing='5' cellpadding='5' width='100%' height='100%'>
<tr><td><center><H3>Не введено сообщение!</h3><br>
<INPUT TYPE='button' VALUE='НАЗАД' onClick='history.go(-1)'>
</td></tr></table>");
}
if ((!empty($_POST['N1']))||($_POST['N1']==0))$N1=($_POST['N1']);else{
exit("<center>Неопознанная ошибка! Попробуйте ещё раз!<br>
<br><a href='index.php'><H4>Назад</h4></a>");
}
$N1++;
$dat = date("d m y H:i");
$km = file("n".$N1.".txt");
$km[0]++;
$fp = @fopen("n".$N1.".txt","w");
fputs($fp,$km[0]);
fclose($fp);
$fp = @fopen("m".$N1.".txt", "a");
fwrite ($fp, "   ".$dat." пишет ".$tn."  Email ".$te."\n   ".$t2."\n\n");
fclose ($fp);
exit("<table cellspacing='5' cellpadding='5' width='100%' height='100%'>
<tr><td><center><H3>Ваше сообщение успешно загружено!</h3>
```

```
<br><a href='index.php'><H4>НА ФОРУМ</h4></a></td></tr></table>
</body></html>");
}
?>
```

Файл add1.php - первой строке проверяется скрытая переменная-удостоверение, только при ее наличии и верном значении будет выполняться этот сценарий. После условия стоят операторные фигурные скобки, в которые заключен весь остальной код. Во второй строке выводится стандартный начальный HTML-код. С 3-й по 8-ю строки проверяется ввод пользователем информации. Если название темы или сообщение не введены, выполнение сценария завершается, и пользователю выдаются сообщения об ошибке с возможностью вернуться при помощи кнопки назад для повторного ввода данных. В строке 8 тоже проверяется скрытая переменная N, которая содержит информацию о количестве тем, и поскольку она, в принципе, может равняться нулю, условие выставлено составное. Оно состоит из двух условий, соединенных логическим оператором ИЛИ. В PHP он записывается символом «||». Составное условие звучит так: если переменная не пустая или равна нулю. Переменная N передается сценарию из файла index.php напрямую. Ее отсутствие либо связано с ошибкой web-сервера, либо обусловлено причинами нештатного характера. В любом случае дальнейшее выполнение сценария невозможно. Поэтому сценарий завершается, и пользователь получает соответствующее уведомление с гиперссылкой для возврата к исходной странице. В 9-й и 10-й строках увеличиваем на 1 значение количества тем ($N++) и вносим текущую дату и время в переменную $dat. В строках 11-13 открываем файл tems.txt и добавляем в него текст названия темы из переменной $tema.

Атрибутом Action формы в файле index.php является файл look.php. .Листинг файла look.php представлен ниже.

```
<html><head><title>Тема</title></head><body bgcolor='#c0c0c0'>
```

```
<table height='100%' width='100%'><tr><td align=center>
<?php
if (isset($_POST['t']))$n = $_POST['t'];else exit("Тема не выбрана!<br>
<br><INPUT TYPE='button' VALUE='НАЗАД'
onClick='history.go(-1)'>");
$n1 = $n[0];
$N = $n[0]+1;
?>
<center><table BGCOLOR='f0f8ff' width='90%'><tr><td
width='20%'></td>
<td width='60%' align=center><b>
<font face='verdana' color='#ff0000'>НАЗВАНИЕ ФОРУМА</td>
<td align=right><a href='#'>К сайту</a></td></tr></table>
<table BGCOLOR='f0f8ff' width='90%'>
<tr><td align=center>
<font face="verdana">
<?php
$ts=file("tems.txt");
$tn = file("n".$N.".txt");
echo"<br><b>Тема: $ts[$n1]</b><br>"."</font>
<br>Есть такие мнения по этой теме, их $tn[0]";
?>
</td></tr></table>
<table BGCOLOR="f0f8ff" width="90%"><tr><td>
<center><TEXTAREA ROWS=15 COLS=60>
<?php
$t=file("m".$N.".txt");
foreach ($t as $line)
echo $line;
?>
```

```
</TEXTAREA>
</td></tr></table>
<table BGCOLOR="f0f8ff" width="90%"><tr><td align=center><br>
<form action="index.php" method="POST">
<H5><input type="submit" value="Вернуться к темам">
<font face="verdana" color="#0000ff">или ниже добавить к этой теме:</
h5>
</form>
</td></tr></table>
<table BGCOLOR="f0f8ff" width="90%"><tr><td align=center>
<form action="add1.php" method="POST">
Имя (до 30 зн.) <input name="nam" type="text" SIZE=25
MAXLENGTH=35></td></tr>
<tr><td align=center>Email <input name="mail" type="text"></td>
</tr></table>
<table BGCOLOR="f0f8ff" width="90%"><tr>
<td align=center>
<table width="100%"><tr><td align=center>
Добавление к этой теме (до 900 зн.)<br>
<textarea name="mes" rows=10 cols=50 ></textarea>
</td></tr></table>
</td><tr><td align=center><br>
<input name="N1" type="hidden" value="<?php echo $n[0] ;?>">
<input name="add" type="hidden" value="t">
<input type="reset" value="Очистить">
<input type="submit" value="Отправить в форум">
</form>
</td></tr></table>
</td></tr></table>
</body>
```

</html>

Результат выполнения файла представлен на рисунке 2.1

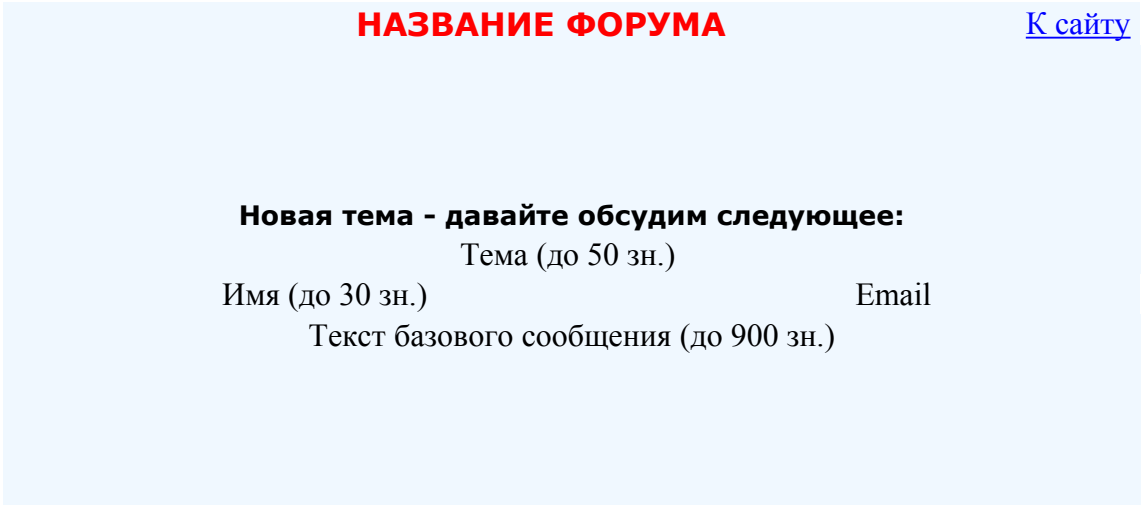


Рисунок 2.1 – Вид главной страницы форума при отсутствии тем

2.4 Вопросы к защите лабораторной работы

1) Что понимается под форумом?

2) Поясните алгоритм формирования значения переменной $size.

3) Как выполняется проверка вводимой пользователем информации?

4) Поясните назначение файла add1.php.

5) Поясните механизм изменения и дополнения уже существующего класса.

6) Перечислите базовые функции PHP и их синтаксис.

7)Поясните синтаксическое построение $this->act.

7) Назначение функции htmlspecialchars().

8) Какая функция создает из строки массив по заданному разделителю? Поясните ее синтаксис.

9) Перечислите основные элементы системы формирования заказов через Интернет.

10) Поясните основные алгоритмы, заложенные в программу системы заказов через Интернет.

11) Поясните, как сервер идентифицирует каждого посетителя.

2.5 Список рекомендуемой литературы

1) Мазуркевич А., Еловой Д. PHP: настольная книга программиста - Мн.: Новое знание, 2003. - 480 с.

2) Котеров Д.В. Самоучитель PHP 4.- СПб.: БХВ-Петербург, 2001. - 576 с.

3) http://microsoft.com/php

3 ЛАБОРАТОРНАЯ РАБОТА № 4
ОСНОВЫ ТЕХНОЛОГИИ ASP.NET. РАБОТА С БАЗАМИ ДАННЫХ

3.1 Цель лабораторной работы

Получить основные навыки работы с базами данных в среде ASP.NET

3.2 Задание на выполнение лабораторной работы

1) Ознакомиться с процессом компиляции в среде .NET Framework.
2) Ознакомиться с работой с базами данных ADO.NET.
3) Разработать и отладить страницы для чтения данных из файла MDB.
4) Создать серверные элементы управления и выполнить демонстрацию динамического изменения их свойств

3.3 Основные теоретические сведения

3.3.1 .NET Framework

Слова .NET и ASP.NET происходят от названия .NET Framework ("общая структура .NET") — набор объектов и планов (Blueprints), созданный компанией Microsoft для разработки приложений. ASP.NET работает благодаря .NET Framework.

У всех приложений, разработанных в .NET Framework, и в том числе у приложений ASP.NET, есть некоторые общие возможности, обеспечивающие им совместимость, безопасность и стабильность. Давайте рассмотрим эти возможности по одной.

3.3.2 Common Language Runtime

Common Language Runtime (CLR) означает "среда выполнения общего языка", среда, которая управляет выполнением кода. Другими словами, в ней запускается и поддерживается любой написанный вами код.

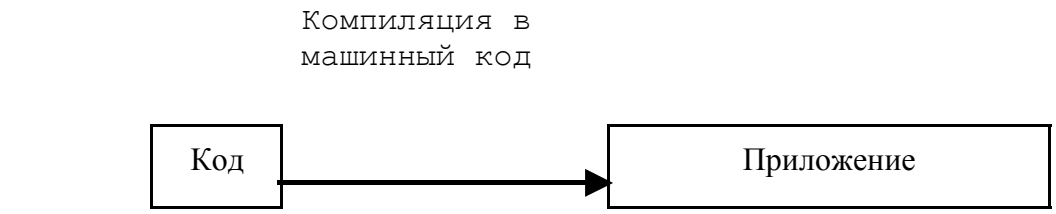
Традиционно при создании приложений пишется код на одном из языков программирования (таком, например, как Visual Basic), компилируете его в формат, понятный для компьютера (в единицы и нули), а затем отправляете откомпилированный код на выполнение. Обратите внимание, что компьютеры разных типов (например, PC и Macintosh) "говорят" на разных языках. Это значит, что для использования приложения на компьютере другого типа приложение необходимо перекомпилировать на язык этого компьютера. Но на .NET Framework дело обстоит несколько по-другому.

Конечно, при использовании .NET Framework и CLR все равно приходится писать код и заниматься его компиляцией. Однако вместо того чтобы компилировать его в то, что понимает компьютер, вы компилируете его в код языка, который называется Microsoft Intermediate Language (MS1L) (это название означает "промежуточный язык от Microsoft"). Такой язык является удобным средством представлять весь написанный вами код. Страницы ASP.NET также компилируются в код MSIL. При компиляции в этот код приложение создает так называемые метаданные. Они являются информацией, которая описывает само приложение. Метаданные рассказывают, что оно может делать, кому принадлежит и так далее.

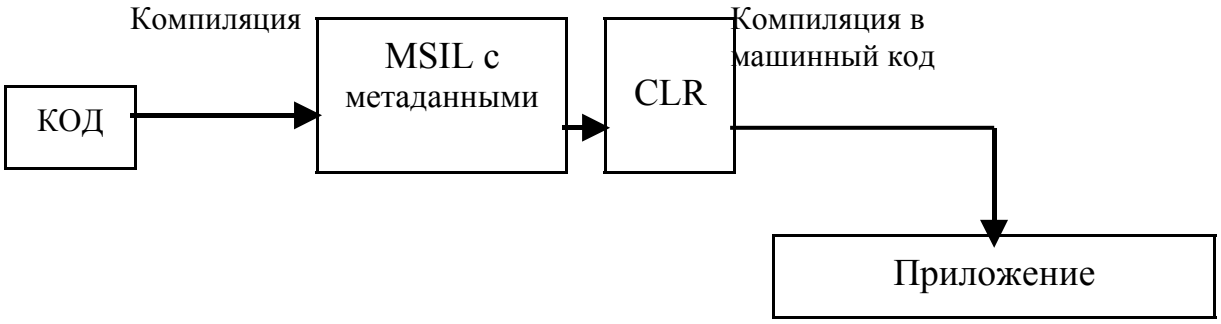
Вместе с MSIL и метаданными появился новый класс языков программирования: C#, Cobol, Perl и так далее. Эти языки похожи на те, что были до них, но их выводом может быть код, такой как MSIL или обычный компилированный.

Затем, когда нужно выполнять программу, за дело берется Common Language Runtime (CLR) и заново компилирует код — на этот раз на родной язык компьютера. Таким образом код MSIL может выполняться на компьютере любого типа. CLR говорит на языках многих компьютеров и делает за вас всю компиляцию на эти языки. Так что, откомпилировав свое приложение одни раз, вы можете передавать его на любой другой компьютер. На рисунке 3.1 показано, в чем состоит разница между

традиционным процессом компиляции и тем, что выполняется в .NET Framework..



а) Традиционный метод



б) В среде .NET Framework

Рисунок 3.1- Сравнительная схема компиляции

Используя метаданные, CLR ищет, каким образом следует запускать приложение. Благодаря этому установка программ становится очень легкой. Дело в том, что при традиционной установке информацию о приложении необходимо помещать в системный реестр или в центральное хранилище данных о приложениях. К сожалению, системный реестр можно вывести из строя любым изменением в приложении (перемещением его каталога, установкой нового компонента и так далее), и тогда приложение может и не работать так, как положено. Благодаря метаданным системный реестр становится ненужным. Вся необходимая информация хранится вместе с файлами приложения, так что все сделанные изменения вступают в силу автоматически

Код, который работает вместе с CLR, называется *управляемым кодом.* Такое название объясняется тем, что CLR управляет выполнением этого кода

и дает определенные преимущества (такие, например, как управление ресурсами), и при этом разработчику не нужно делать ручную настройку. В свою очередь, код, который выполняется вне CLR, называется *неуправляемым.*

Кроме того, CLR может выполнять обработку ошибок, поддерживать безопасность, поддерживать согласованность версий (versioning) и развертывание приложений, а также интеграцию с различными платформами. А это означает, что для написания приложений .NET (и в том числе приложений ASP.NET) можно выбрать любой язык программирования.

3. 3.3 Классы .NET Framework

.NET Framework всегда держит при себе планы (Blueprints), то есть описания объектов. В свою очередь, объектами считается все, что находится в .NET Framework: страницы ASP.NET, окна сообщений и так далее. Такие объекты размещаются внутри логических групп, которые называются *пространствами имен* (namespace). Например, все объекты, относящиеся к базам данных, можно расположить в пространстве имен System.Data (‘Система.Данные’), а все объекты XML — в System.Xml (‘Система.XML’) и так далее. Такое объединение в группы очень полезно при создании библиотек объектов. Пространства имен используются при создании приложений ASP.NET.

Каждое пространство имен из структуры .NET — это, в сущности, коллекция схем. ASP.NET поставляется вместе с собственными схемами, но иногда этого набора бывает недостаточно. Поэтому, чтобы создавать дополнительные типы объектов, приходится обращаться из ASP.NET к другим наборам. Получить доступ к этим дополнительным объектам и методам можно с помощью ключевого слова Import (импорт):

<%@ Import Namespace="System.Drawing" %>

Эта строка задает импорт всех классов из пространства имен System.Drawing (означает "Система.Чертеж"), например, font (шрифт) или image

(изображение). Теперь при создании своих объектов можно использовать эти схемы.

3. 3.4 Web-формы

Формы Web очень похожи на традиционные формы HTML. Разница между ними в том, что первые — это серверное средство, что означает возможность создания пользовательских элементов на сервере. В этом случае сервер полностью осведомлен о том, как выглядит интерфейс, какие функции он выполняет, какие данные следует ожидать и т.п.

На сервере мы создаем объекты, называемые серверными элементами управления, которые представляют собой части интерфейса пользователя. В отличие от элементов формы HTML, эти объекты полностью контролируемы- они имеют свойства, события и методы, которыми можно манипулировать. Как только клиент запрашивает страницу, ASP.NET преобразовывает эти средства управления в HTML, который совершенно точно отображается в браузере (рисунок 3.2). Используя Web-формы, сервер знает о внешнем виде и функциях каждой конкретной формы

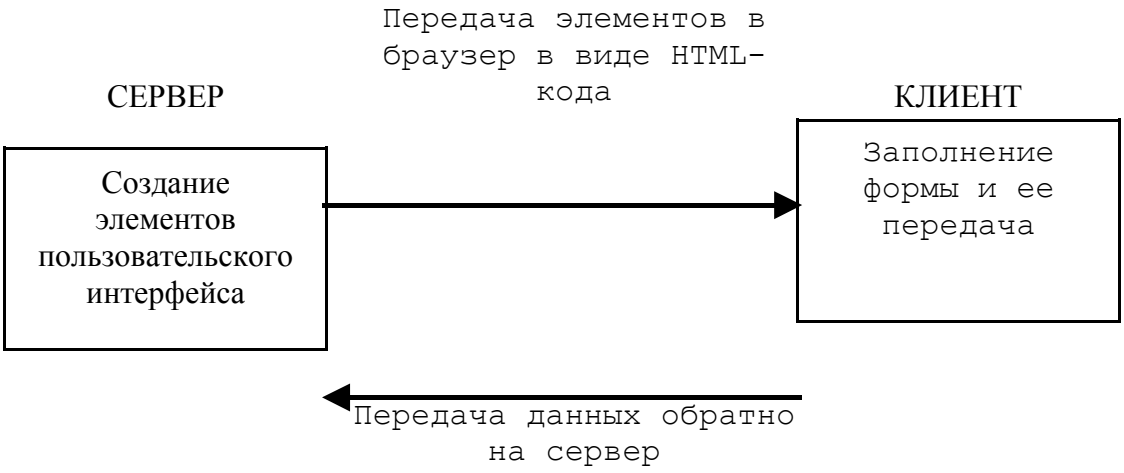


Рисунок 3.2

Посредством клиентского сценария, автоматически генерируемого ASP.NET, эти элементы управления предупреждают сервер всякий раз, когда что-либо происходит, например, нажатие на кнопку. Клиентский сценарий отсылает

информацию на сервер в момент возникновения события, часто не сообщая об этом пользователю. Таким образом, сервер непрерывно уведомляется о клиентских процессах, что объединяет их друг с другом в одно целое.
На практике это выглядит, как управляемая событиями модель.

Поскольку сервер сам создает элементы управления, он также может запомнить, что вводилось в каждый элемент.

3.3.5 Серверные элементы управления

Серверные элементы управления являются составляющими пользовательского интерфейса Web-формы. В ASP.NET выделяется четыре типа серверных элементов управления:

- серверные элементы управления HTML,
- элементы Web,
- средства подтверждения,
- пользовательские элементы.

Элементы управления HTML представляют собой обычные элементы формы HTML, такие как текстовые поля ввода и кнопки, но создаются они на сервере, где ими можно управлять.

Аналогичны им элементы Web, но они более функциональны и могут формировать сложный пользовательский интерфейс.

Средства подтверждения используются для проверки правильности пользовательского ввода данных.

Пользовательские элементы специально предназначены для реализации некоторой особой функциональности.

Все элементы управления, размещенные на сервере, имеют свойства, методы и события. Они намного функциональнее, чем традиционные элементы формы HTML, и существенно упрощают процесс формирования пользовательского интерфейса.

При создании серверного элемента не нужно заботиться о написании кода HTML. При поступлении запроса на страницу элемент управления

автоматически генерирует HTML, корректно отображаемый в браузере. Например, следующая строка создает сервере элемент управления Button:

<asp:Button text="Submit" runat="server"/>

В момент ответа на запрос клиенту выдается такой код HTML:

<input type="Submit" name="ctrll" value="Submit">

Эти две строки лишь похожи друг на друга. Первая выполняется только на сервере — клиент никогда не получит ее. (Даже если бы и получил, то не знал бы, что ней делать — браузер понимает только код HTML.) Вторая строка — это то, что принимает клиент.

Также ASP.NET знает о возможностях каждого браузера и, следовательно, отсылает соответствующий код каждому из них. Например, если браузер и поддерживает динамический HTML (DHTML), то ASP.NET не будет отправлять ему такой код. Этот подход называется *низкоуровневой поддержкой* связи с тем, что ASP.NET может сглаживать вывод кода HTML для браузеров, которые не поддерживают современный уровень функциональности.

В идеале, низкоуровневая поддержка должна отображать корректно i-ый элемент управления. Однако некоторые элементы будут отображаться неодинаково в различных браузерах, что связано с неидентичной трактовкой ими кода HTML. Тем не менее, при разработке под наиболее популярные версии браузеров низкоуровневая поддержка будет работать верно.

3.3.6 Работа с базами данных. ADO.NET

ADO.NET — новый этап в технологам ActiveX Data Objects (ADO), это модель доступа к данным, созданная специально для использования в Web. ADO.NET создает интерфейс ко всем совместимым с OLEDB источникам данных и обеспечивает соединеннее тми, извлечение, управление и обновление данных. ADO.NET можно применять на удаленном компьютере, в распределенном приложении и в случае распределенных данных.

ADO.NET создает структуру для доступа к любым типам данных в страницах, благодаря чему пользователи могут видеть или изменять

информацию, которая содержится в любом типе данных, включая базы, текстовые файлы и наборы данных XML.

ADO.NET — это версия программного интерфейса ADO, приспособленная для более эффективного представления данных на ASP-страницах. Например, она полностью воспринимает язык XML и приспособлена для общения с XML-приложениями. ADO.NET предлагает много возможностей, которые значительно облегчают работу разработчика. Страницы ASP.NET используют ADO.NET для связи с любым типом данных. Модель доступа к данным в ADO.NET и ASP.NETпредставлена на рисунке 3.3.

Интерфейс ADO.NET полностью совместим с источниками данных, поддерживающими объектные технологии OLE для баз данных, ими, как язык SQL или механизм баз данных Jet.

3.3.7 Объект DataSet

В центре интерфейса ADO.NET находится объект DataSet. Этот объект-концептуально новая идея, которая заменяет традиционный объект RecordSet в ADO.

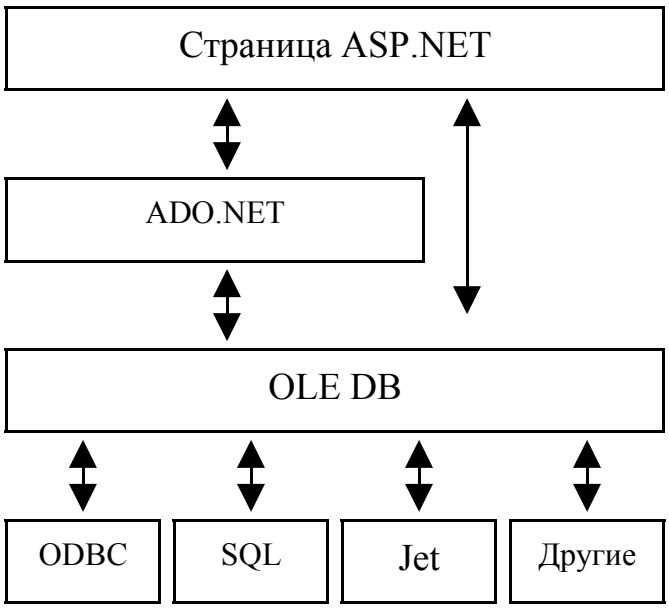


Рисунок 3.3- Модель доступа к данным в ADO.NET и ASP.NET.

DataSet — это простой, постоянно находящийся в оперативной памяти способ хранения данных, который обеспечивает последовательную программную модель для доступа к данным, независимо от их типа. В отличие от RecordSet, DataSet содержит иные наборы данных, включая ограничения, связи и даже несколько таблиц одноименно.

Данный объект содержит множество структур. В каждую структуру можно поместить несколько элементов. С каждым элементом в структуре можно производить различные операции — добавлять, изменять, просматривать и т.д. В этом заключаются основные принципы работы с DataSet.

DataTable – объект, который представляет отдельную таблицу базы данных. (DataSet содержит коллекцию таблиц в объекте TablesCollection). Объект DataTable полностью отображает соответствующую таблицу, включая связи между данными и ключевые ограничения. Он содержит две другие коллекции, Rows и Columns, которые отображают данные и логическую структуру таблиц.

Объект *RelationsCollection* позволяет двигаться между таблицами согласно определенным для них связям. Больше не нужно использовать сложные связи и объединения. Фактические связи между таблицами представлены объектами *DataRelation,* содержащими информацию о двух соединенных таблицах, связи первичного и внешнего ключа и названия связей.

Связи можно также добавить с помощью объекта DataRelation. ADO.NET автоматически вводит ключевые ограничения: не разрешается изменение одной таблицы, которое может повредить ее связь с другой таблицей.

Объект *ExtendedProperties* содержит дополнительную информацию, например, имя пользователя и пароль.

3. 3.8 Модель объекта ADO.NET

ADO.NET состоит из двух ключевых частей: набора данных *DataSet* и средств управления *Managed Providers*. В наборе представлены данные, передаваемые между компонентами ADO.NET, а именно из памяти в страницы ASP.NET. Это механизм обработки данных вне их хранилища.

Managed Providers служат в качестве промежуточного звена между набором DataSet и памятью и обеспечивают механизм для соединения, доступа, управления и извлечения информации из любой совместимой с OLE DB базы данных, например Microsoft Access.

Microsoft поставляет два типа Managed Providers в комплекте ADO.NET: SQL и OLE DB. Первый используется исключительно для взаимодействия с Microsoft SQL Server и обеспечивает все методы коммуникации между SQL-сервером и набором данных. Второй, OLE DB, служит связующим звеном между набором и совместимым с OLE DB источником данных.

Для соединения с SQL-сервером одноименный Managed Provider применяет протокол, называемый *tabular data stream* (табличный поток данных). Это наиболее эффективный и не зависящий от OLE DB, ADO или ODBC метод для соединения с SQL-сервером. Данный протокол управляется CLR.

OLE DB-провайдер обеспечивает эффективное взаимодействие со всеми другими типами данных и, если есть необходимость, применяется для доступа к серверу SQL. Каждый провайдер имеет три компонента:

- интерфейс для соединения и управления базой данных, а также для взаимодействия с набором DataSet;
- поток данных для быстрого и эффективного доступа (похож на набор данных DataSet, но быстрее и с меньшим набором функций);
- объекты для соединения с базой данных и для исполнения ее специфических команд на нижнем уровне.

3. 3.9 Доступ к данным в среде ASP.NET

Можно выделить пять основных этапов получения данных, используя ASP.NET Web-страницы:

1) Установить соединение с базой данных.
2) Открыть созданное соединение.
3) Заполнить компоненту DataSet требуемыми данными.
4) Установить компоненту DataView для отображения данных.
5) Связать серверный элемент управления с компонентой DataView, используя привязку данных.

Перед тем как получить доступ к данным необходимо импортировать два пространства имен, содержащие компоненты работы с данными (если используемая база данных работает под управлением СУБД Microsoft SQL Server, то используется пространство имен System.Data.SQLClient.)

Затем необходимо организовать соединение с базой данных. Источником данных Data Source является созданный ранее файл Access.

Далее происходит открытие соединения с базой данных и исполняется SQL команда, которая возвращает необходимые данные из таблицы. Это соответствует первому и второму шагам в процессе доступа к данным.

Затем создается и заполняется возвращаемыми запросом данными объект типа DataSet. Это соответствует третьему шагу процесса.

Потом данные привязываются к элементу управления типа DataList, который автоматически отображает данные.

Далее остается просто создать элемент управления типа DataList. Элемент управления типа DataList использует шаблоны для форматирования данных и автоматически обрабатывает полученные записи. Он также получает размер окна браузера и подстраивает ширину колонок так, чтобы все колонки отображались в одном окне.

Ниже приведен листинг программы для чтения данных из файла MDB.

```
<%@Import Namespace="System.Data " %>
<%@Import Namespace="System.Data.OleDb" %>
```

```
<script language="VB" runat="server">
  sub Page_Load(obj as Object, e as EventArgs)
    dim objConn as new OleDbConnection("Provider=Microsoft.Jet.OLEDB.
4.0;Data Source=D:\inetpub\wwwroot\Mstuca\Web_Base.mdb")
    dim objCmd as new OleDbDataAdapter("SELECT * FROM (Humans
INNER JOIN Posit ON [Humans].[Posit]=[Posit].[Posit_ID]) INNER JOIN
Dep ON [Humans].[Dep]=[Dep].[Dep_Id] ORDER BY Last_Name,
First_Name, Middle_Name;",objConn)
    dim ds as DataSet = new Dataset()
    objCmd.Fill(ds,"Humans")
    MyDataList.DataSource=ds.Tables("Humans").DefaultView
    MyDataList.DataBind()
  end sub
</script>
<html>
<body>
<ASP:DataList id="MyDataList" RepeatColumns="1"
RepeatDirection="Vertical" runat="server">
  <ItemTemplate>
    <br><b>ФИО: </b>
    <%#DataBinder.Eval(Container.DataItem,"Last_Name")%> 
    <%#DataBinder.Eval(Container.DataItem,"First_Name")%> 
    <%#DataBinder.Eval(Container.DataItem,"Middle_Name")%> 
    <br><b>Телефон: </b>
     <%#DataBinder.Eval(Container.DataItem,"Office_Phone")%> 
    <br><b>Должность: </b>
     <%#DataBinder.Eval(Container.DataItem,"Posit_Name")%> 
    <br><b>Отдел: </b>
     <%#DataBinder.Eval(Container.DataItem,"Dep_Name")%> 
    <br><br>
```

```
    </ItemTemplate>
  </ASP:DataList>
  </body>
  </html>
```

Результат выполнения программы приведен на рисунке 3.4

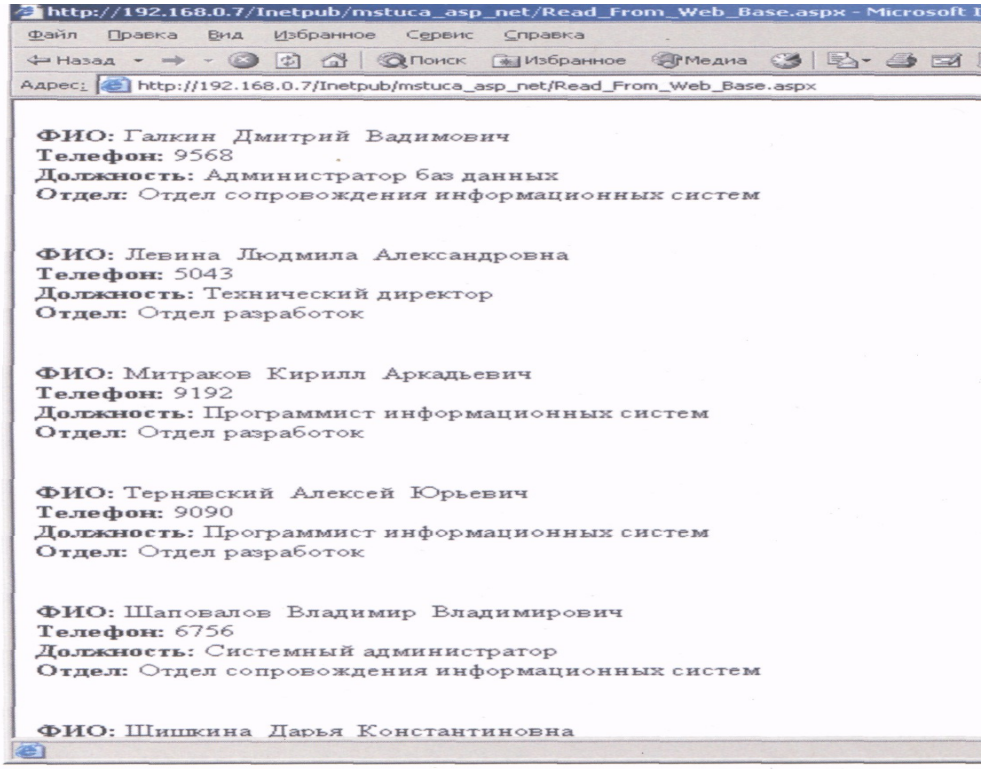


Рисунок 3.4 – Окно с результатами выполнения программы

Листинг программы для создание серверных элементов управления и демонстрации динамического изменения их свойств приведен ниже.

```
<%@Import Namespace="System.Web.UI.WebControls " %>

<html>
  <head>
    <title>Dynamic Table</title>

<script language="VB" runat="server">
  Sub SubmitButton_Click(Source As Object, e As EventArgs)
    Table1.Bgcolor = Select1.Value
```

```
    Table1.Border = Select2.Value
    Table1.Cellpadding = Select3.Value
    Table1.Cellspacing = Select4.Value

    If Select5.Value = "1" Then
      Tr1.Bgcolor = Select6.Value
    Else If Select5.Value = "2" Then
      Tr2.Bgcolor = Select6.Value
    Else If Select5.Value = "3" Then
      Tr3.Bgcolor = Select6.Value
    End If

   If radio1.Checked = True Then
     Labet1.Text=Text1.Value
   Else If radio2.Checked = True Then
     Labet1.Text= " "
   End If
    End Sub
</script>
</head>

<body>
<form id="Form1" method="post" runat="server">
  <center>
<h3><asp:Label runat="server" id="Labet1">Table Header</asp:label></h3>
<br>
<table runat="server" id="table1" cellspacing=3 cellpadding=3 border=1>
    <tr runat="server" id="Tr1">
      <td>One</td>
      <td>NO_One</td>
```

```
    </tr>
    <tr runat="server" id="Tr2">
      <td>Two</td>
      <td>NO_Two</td>
    </tr>
    <tr runat="server" id="Tr3">
      <td>Three</td>
      <td>NO_Three</td>
    </tr>
  </table>
<p>
</center>
<br>
<br>
<br>
Table Header:
<Input type="text" runat="server" id="Text1" value="Table Header"></Input>
<br>
Table Background Color:
<select id="Select1" runat="server">
  <option value="#ffffff">Белый</option>
  <option value="#ff0000">Красный</option>
  <option value="#ffff10">Жёлтый</option>
</select>
<br>
Table Border:
<select id="Select2" runat="server">
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
```

```
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
</select>
<br>
Table Cellpadding:
<select id="Select3" runat="server">
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
</select>
<br>
Table Cellspacing:
<select id="Select4" runat="server">
  <option value="0">0</option>
  <option value="1">1</option>
  <option value="2">2</option>
  <option value="3">3</option>
  <option value="4">4</option>
  <option value="5">5</option>
</select>
<br>
<p>
Select a Row:
<select id="Select5" runat="server">
  <option>1</option>
  <option>2</option>
```

```
  <option>3</option>
</select>
<br>
Row Color:
<select id="Select6" runat="server">
  <option>White</option>
  <option>Yellow</option>
  <option>Red</option>
  <option>Blue</option>
</select>
<br>
Show Table Header:
<input type ="radio" runat="server" id="radio1" name="radio1"
checked="true">Yes
<input type ="radio" runat="server" id="radio2" name="radio1">No
<br>
<center>
<input type="submit" runat="server" id="submit1"
value="Submit" onserverclick="SubmitButton_Click">
</center>
</form>
</body>
</html>
```

Результат выполнения программы приведен на рисунке 3.5

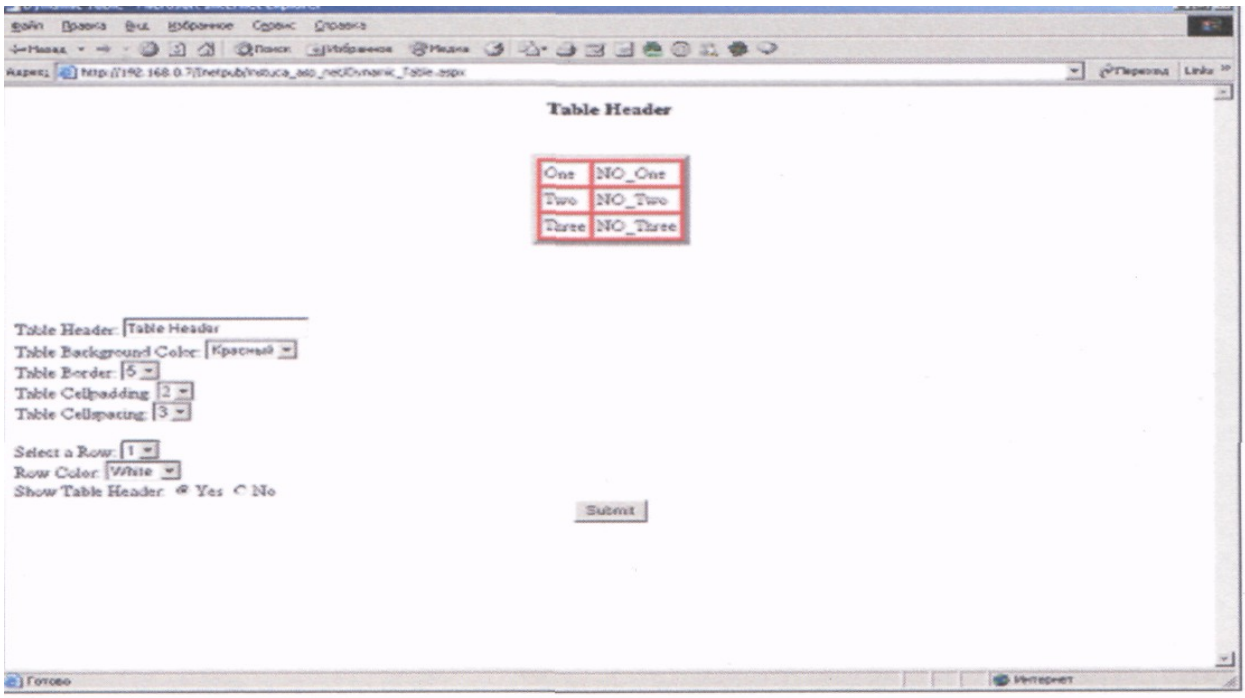


Рисунок 3.5 – Окно с результатами выполнения программы

3.4 Вопросы к защите лабораторной работы

1) Что понимается под технологией .NET?
2) Перечислите основные компоненты технологии .NET.
3) Что понимается под пространством имен?
4) Как осуществляется доступ к данным в среде ASP.NET?
5) Чем отличается процесс компиляции в среде .NET Framework?
6) Назовите основные функции Managed Providers.
7) Основное значение объекта DataSet.
8) Какие структуры содержит объект DataSet?
9) Перечислите основные типы серверных элементов управления в ASP.NET.

3.5Список рекомендуемой литературы

1) http:// **www.microsoft.com/net**
2) Бучас Г. ASP.NET. Учебный курс.- Спб.: Питер, 2002 – 512с.
3) Пол Гиббонз. Платформа .NET для Java- программирования. Библиотека программиста.- Спб.: Питер, 2003- 336с.