МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
"Московский технологический университет"
МИРЭА

Институт информационных технологий

Подлежит возврату
№

ИНТЕРНЕТ-ТЕХНОЛОГИИ И СИСТЕМЫ

Методические указания
по выполнению лабораторных и практических работ
Часть 3
(Работы № 5, 6)

для студентов
Направление подготовки
09.03.04 Программная инженерия

Профиль подготовки
Интеллектуальные программные системы и комплексы

МОСКВА 2017

Составитель И.И. Холкин
Редактор В.М. Панченко

Методические указания к лабораторным и практическим работам по дисциплине «Интернет-технологии и системы», Часть 3, предназначены для студентов 4-го курса очной формы обучения (квалификация Бакалавр. Направление подготовки 09.03.04 Программная инженерия. Профиль подготовки Интеллектуальные программные системы и комплексы).

Предполагается, что студенты изучили языки HTML, JavaScript, визуальный WYSIWYG редактор для разработки и управления Web-сайтом и умеют применять основные приемы Интернет - программирования и Web- дизайна на практике, выполнив 4 лабораторные работы [1]. Данные лабораторные и практические работы (№5-6) закрепляют и расширяют полученные знания в области Интернет-технологий. Печатается по решению редакционно-издательского совета университета.

Рецензенты: Б.Б. Чумак,
А.Н. Райков

©МИРЭА, 2017

Лабораторная работа № 5
Практическое занятие № 5

Серверные скрипты и их использование

Цель работы

Ознакомление с принципами, построения **серверных скриптов**. Формирование навыков создания и использования **серверных скриптов**.

Методические указания

1. Серверные скрипты и их использование

Серверные скрипты представляют собой приложения, выполняемые на **Web-сервере.**

1.1. Создание приложений Web-сервера

Web-сервер обеспечивает работу Web-узлов Интернета (или локальной сети на базе протокола TCP/IP), позволяя множеству пользователей одновременно обращаться к одним и тем же страницам HTML. Web-сервер рассылает копии этих страниц клиентским программам - браузерам. В то же время, современные подходы к созданию приложений Интернета, ориентированных на бурно развивающую электронную коммерцию, требуют активного взаимодействия с пользователем и обеспечения тесной обратной связи с ним, по аналогии с обычными автономными офисными приложениями.

Для этого необходимо, чтобы содержание страницы HTML, отправляемой пользователю, не было простой копией файла на сервере, а создавалось динамически, программным путем, в зависимости от действий пользователя. Например, пользователь может заполнять формы и отправлять их серверу или работать через сервер с таблицей удаленной базы данных, в которой хранятся списки товаров. Такая схема работы напоминает клиент - серверную и распределенную архитектуры. Отличие заключается в том, что, во-первых, передавать клиентской программе надо не массив значений, который она обработает сама, а уже готовую страницу HTML. Ее надо полностью подготовить на Web-сервере, вследствие чего нагрузка на сервер увеличивается. С другой стороны, нагрузка на компьютеры пользователей снижается, так как там требуется только отобразить страницу в окне браузера. Во-вторых, информация, особенно коммерческая, передаваемая по глобальной сети, должна быть хорошо защищена. Она может шифроваться промежуточными программными модулями или, по крайней мере, при ее передаче должна использоваться более надежная и безопасная версия протокола HTTP.

Сам Web-сервер (программа) не занимается генерацией страниц. Ему и так хватает работы по рассылке этих страниц тысячам посетителей Web-узлов. При поступлении запроса на выполнение нестандартной операции Web-сервер загружает в виде отдельного процесса один из специальных модулей, который выполняет создание нужной страницы HTML и затем передает ее обратно Web-серверу, который отсылает ее соответствующему клиенту (см. рис. 1).

Такие модули пишутся на самых разных языках программирования и могут работать под управлением самых разных операционных систем.

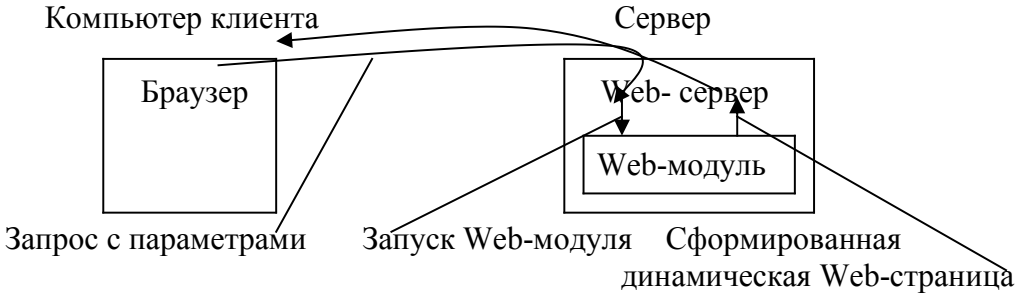


Рис. 1. Формирование динамической Web-страницы

1.1.1. Создание заготовки Web-модуля в системе Delphi (Факультатив)

В системе Delphi 5 есть Мастер, позволяющий путем нескольких уточняющих вопросов подготовить пустое Web-приложение (модуль, загружаемый Web-сервером в ответ на специфический запрос пользователя) [4]. Этот Мастер, вызывается командой File > New (Файл > Создать) с последующим выбором значка Web Server Application (Приложение Web-сервера). Мастер предложит выбрать один трёх типов будущего серверного Web-модуля (рис. 2).

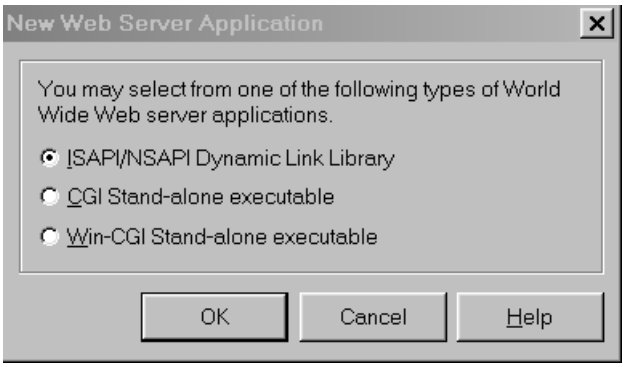


Рис. 2 Выбор типа приложения Web-сервера

- ISAPI/NSAPI Dynamic Link Library. Создание библиотеки .DLL, которая автоматически загружается Web-сервером по мере необходимости. Данные для обработки передаются в библиотеку и возвращаются Web-серверу в виде специальной структуры с помощью программного интерфейса ISAPI фирмы Microsoft или NSAPI фирмы Netscape.
- CGI Stand-alone executable. Создание серверного консольного приложения. Оно по мере необходимости запускается Web-сервером. Приложение получает параметры и передает результаты работы напрямую.
- Win-CGI Stand-alone executable. Создание серверного приложения в формате программы Windows. Оно вызывается так же, как консольное приложение, но информация передается и получается Web-сервером не напрямую, а через специальный промежуточный файл .INI. Этот вариант удобен тем, что позволяет упростить отладку модуля путем ручного формирования файла .INI с нужным запросом.

Каждый из этих модулей запускается на сервере. При обращении к нему из браузера пользователя Web-сервер загружает соответствующую программу, при необходимости передает ей параметры и ожидает окончания ее работы, после чего отсылает Результа-

ты обратно браузеру пользователя.и к Web-серверу обращается множество пользователей (тысячи), то для каждого из них запускается копия программного модуля, что может вызвать значительную перегрузку компьютера. В таких случаях лучше использовать модули в формате ISAPI/NSAPI, которые позволяют ускорить запуск благодаря тому, что выполнены в формате .DLL и позволяют Web-серверу использовать их функции во внутреннем рабочем пространстве. Форматы CGI требуют запуска полноценной консольной программы или программы Windows, что предъявляет повышенные требования к компьютеру.
Недостаток подобного подхода в том, что при каждом изменении модуля в формате .DLL надо перезапускать сам Web-сервер, чтобы он выполнил перезагрузку этого модуля в свое адресное пространство.

1.1.2. Параметры и результаты

Web-модуль получает от Web-сервера набор параметров *(запрос),* указанных пользователем вручную или сгенерированных браузером автоматически. В зависимости от этих параметров модуль выполняет те или иные действия. Запрос передается в виде объекта, имеющего тип TWebRequest.
Допустим, создан Web-модуль MyTest.exe. Он должен быть размещен в подходящий виртуальный каталог Web-сервера (условный Web-узел www.my-site.ru), для которого установлены права на запуск программ CGI на сервере. Как правило, такой каталог называется cgi-bin. Данный модуль будет вызван, если пользователь наберет в адресной строке своего браузера следующий текст.

http://www.my-site.ru/cgi-bin/MyTest.exe/PARAM1?f7

Здесь:
http://www.my-site.ru — адрес Web-узла;
cgi-bin — каталог;-
MyTestexe — запускаемый Web-модуль;
/PARAM1 — название параметра;
? — символ, отделяющий название параметра от его значения;
f7 — значение параметра PARAM1.

Модуль можно запускать и без параметров:

http://www.my-site.ru/cgi-bin/MyTest.exe

а также со значением, указываемым модулю без названия параметра:

http: //www.my-site.ru/cgi-bin/MyTest.exe?c5

После того как на сервере выполнены нужные действия, зависящие от переданной информации, модуль возвращает серверу результат своей работы. Он записывает в объект, имеющий тип TWebResponse. Этот результат представляет собой строку, содержащую текст *HTML,* который и отображается браузером пользователя в качестве текущей страницы.

1.1.3. Пример создания Web-модуля в системе Delphi

Для выполнения рассмотренного ниже примера на локальном компьютере должен быть установлен Web- сервер PWS или IIS

1.1.3.1. Содержание примера

При обращении к Web-модулю без параметров отображается пустая страница с коротким приветствием. Если указан параметр /DAY со значением name, то сообщается название текущего дня недели, если значение равно full, отображается дата. Если задан параметр /TIME, то показывается текущее время.

1.1.3.2. Создание пустого Web-модуля

Web-модуль создается с помощью Мастера. Выберем консольный тип приложения - CGI Stand-alone executable. На экране откроется окно WebModule1 (Рис. 3).

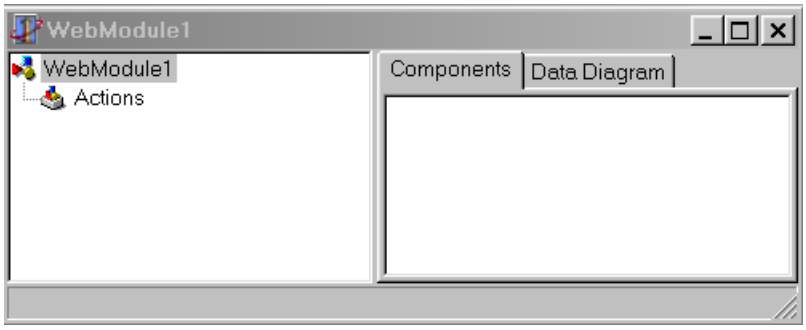
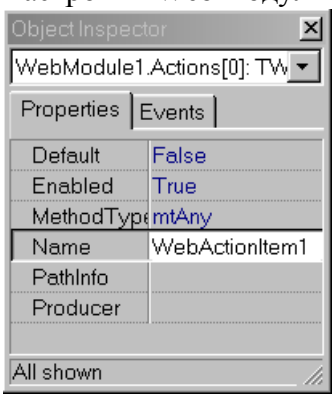


Рис. 3 Окно настройки Web-модуля



Пока что в списке действий (Actions) нет ни одного ответного действия модуля на поступление запроса от Web-сервера. Чтобы добавить новое действие, надо выбрать строку Actions, щелкнуть на ней правой кнопкой мыши и выбрать в контекстном меню пункт Add item (Добавить элемент). В список добавляется новая строка, соответствующая новому созданному объекту класса TWebActionItem, а Инспектор объектов покажет свойства этого объекта, представленные на рис. 4 и в табл. 1.

Рис. 4Настройка свойств действия, выполняемого Web-модулем

Таблица 1. Свойства объектов класса TWebActionItem

Свойство	Назначение
Default	Имеет значение true, если данное действие выполняется по умолчанию, когда в ответ на запрос не выполнено ни одно из других действий
Enabled	Имеет значение true, если Web-сервер может использовать данное действие
MethodType	Тип обрабатываемого запроса. По умолчанию имеет значение mtAny (все запросы)
Name	Имя объекта
PathInfo	Имя параметра, задание которого активизирует вызов данного объекта
Producer	Имя поставщика Web, формирующего результат работы модуля

Основное свойство — это свойство PathInfo, в котором надо задать одно из обрабатываемых названий параметра (/DAY, /TIME) или не указывать ничего. Таким образом,

для двух параметров и значения, принятого по умолчанию, надо подготовить три объекта в списке Actions.
У первого (пусть он называется DefaultAction) в свойстве PathInfo не указывается ничего, у второго (DayAction) — задается строка/DAY, у третьего (TimeAction) — строка /TIME. Свойство Producer сохраняется пустым (рис. 5).

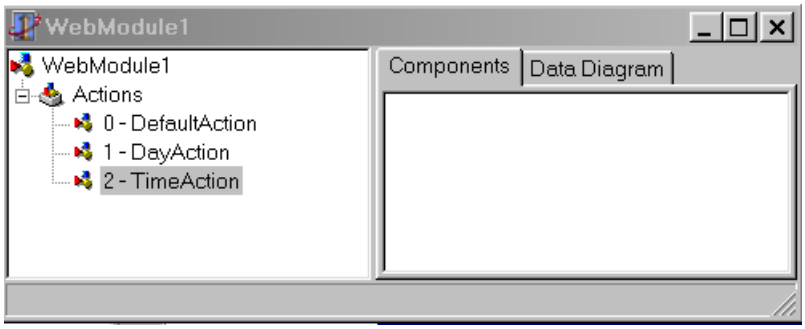


Рис. 5 Готовый набор параметров, обрабатываемых Web-модулем

1.1.3.3. Реакция на запрос

Чтобы объект DefaultAction мог реагировать на запрос Web-сервера, на вкладке Events (События) для этого объекта надо выбрать свойство OnAction (оно единственное) и создать обработчик этого события двойным щелчком.

```
procedure TWebModule1.WebModule1DefaultActionAction(Sender: TObject;
    Request: TWebRequest; Response: TWebResponse;
  var Handled: Boolean);

begin

end;
```

Данный обработчик будет вызываться при обращении из браузера к Web-модулю без параметров, например с помощью следующей строки.

```
http://www.my-site.ru/cgi-bin/MyTest.exe
```

Параметр Request процедуры содержит всю информацию о запросе. В частности, в свойстве Query этого параметра хранится значение запроса, если оно было указано пользователем после знака? Например, пусть обращение выполнялось следующим образом.

```
http://www.my-site.ru/cgi-bin/MyTest.exe?проверка
```

В этом случае значение переменной Request.Query равно строке 'проверка'.
Результат работы модуля записывается в виде строки в свойство Content параметра Response обработчика. Так как при обращении к модулю без специальных параметров должна отображаться пустая страница, вся логика работы данного обработчик уместится в одном операторе присваивания.

```
Response.Content  :=  '<body bgcolor=white><center>'  +
'                              <h3>привет! </h3></center>
                       </body>' ;
```

Обработка запросов с параметрами чуть сложнее. Чтобы сформировать название дня недели, надо узнать его номер с помощью стандартной функции DayOfWeek, которая

принимает значения от 1 (суббота) до 7 (пятница). В качестве ее параметра надо указать структуру типа TDateTime, содержащую информацию о текущем дне. А такой день можно получить с помощью другой функции — Now.
Полное текстовое представление даты формируется обращением к стандартной функции DateToStr.
Процедура, вызываемая при указании параметра/DAY (реакция на событие OnAction объекта DayAction), выглядит следующим образом.

```
procedure TWebModule1.WebModule1DayActionAction(Sender: TObject;
    Request: TWebRequest; Response: TWebResponse;
  var Handled: Boolean);
const DAY_NAMES:  array[1..7]  of string = ('воскресенье', 'понедельник',
'вторник', 'среда', 'четверг', 'пятница', 'суббота');
begin
if Request.Query = 'name'
  then  Response.Content  :=  '<center>сегодня  '  +  DAY_NAMES  [DayOf-
Week(Now)] + '</center>'
else if Request.Query =  'full'
  then Response.Content :=  '<center>' +
        DateToStr(Now) + '</center>'
end;
```

А вот обработчик запроса с параметром /TIME. В нем используется стандартная функция TimeToStr, преобразующая текущее время в текстовое представление.

```
procedure TWebModule1.WebModule1TimeActionAction(Sender: TObject;
    Request: TWebRequest; Response: TWebResponse;
  var Handled: Boolean);
begin
  Response.Content  :=  '<center>  сейчас  '  +  TimeToStr(Now)  +
  '</center>'
end;
```

Заметим, что пользователю необходимо написать только код, выделенный в рамки. Остальной код система Delphi 5 генерирует автоматически.

1.1.3.4. Запуск Web-модуля

Написав всего два условных оператора и четыре оператора присваивания, нам удалось создать с помощью системы Delphi 5 законченный Web-модуль, пригодный для размещения на Web-сервере.
Приложение надо откомпилировать, переименовать как MyTest.exe и разместить в каталоге Web-сервера, соответствующем виртуальному каталогу cgi-bin с правами на запуск на сервере исполняемых EXE-модулей.
Если отсутствует полноценная возможность отладки Web-приложений (нет доступа к Web-серверу, ограничено время в Интернете и так далее), то можно организовать работу Web-сервера в локальном режиме. Например, установить на компьютер Web-сервер Microsoft Personal Web Server.
Тогда можно обратиться из браузера к этому серверу (он должен быть запущен), набрав в адресной строке такой текст.

```
http://127.0.0.1
```

Это IP-адрес локального компьютера. Соединение браузера с Web-сервером произойдет автоматически.
Есть и более привычный способ организации связи с локальным Web-сервером. В настройках протокола TCP/IP (значок Сеть на Панели управления) можно указать имя компьютера (вкладка Конфигурация DNS, поле Имя компьютера), например TestBed. Это название служит виртуальным синонимом текущего IP-адреса (127.0.0.1) компьютера, и теперь обращаться к запущенному Web-серверу можно, набрав в строке браузера следующий адрес.

http://TestBed

Для проверки работы Web-модуля в адресной строке браузера задаются различные параметры запуска (рис. 6)

http://TestBed/cgi-bin/MyTest.exe
http://TestBed/cgi-bin/MyTest.exe/DAY?name
http://TestBed/cgi-bin/MyTest.exe/DAY?full
http://TestBed/cgi-bin/MyTest.exe/TIME

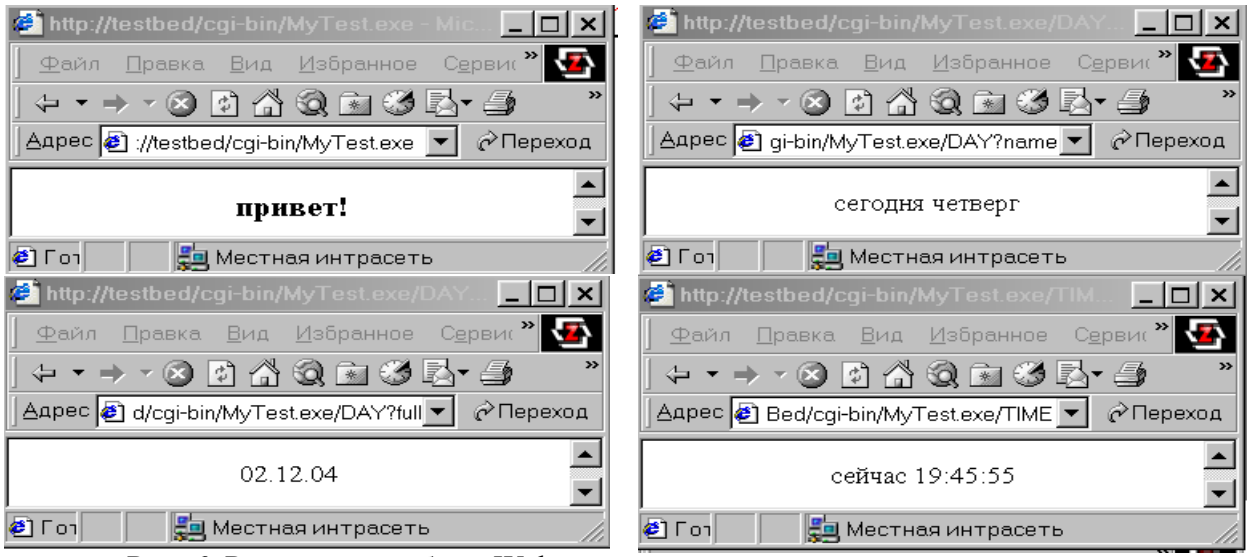


Рис. 6. Результаты работы Web-модуля с различными параметрами

1.2. Разработка серверных скриптов на PHP

Сегодня PHP занимает лидирующие позиции среди средств разработки скриптов для Web. В настоящий момент PHP обладает огромным количеством функций, которые позволяют решать множество проблем, возникающих при разработке Web-скриптов. Следует отметить, что для PHP не нужна специальная среда, он является кроссплатформенным, т. е. может работать и под Windows, и под UNIX, и под Linux. Эта особенность также сыграла не последнюю роль в популяризации PHP.

Еще одним большим плюсом является то, что в Интернете для PHP существует большая библиотека готовых скриптов. Как небольших в виде счетчика посещений Web-страницы, так и целых систем, которые позволяют создать сайт с нуля при минимальном написании кода.

Среди значительного количества книг и пособий по PHP следует отметить книги [6,7], в которых дано множество интересных примеров с подробным описанием и комментариями сложных строк кода. Здесь вы найдете счетчик посещений, реализацию

голосования и гостевой книги, генератор надёжных паролей, загрузку файлов на сервер, почтовую рассылку и многое другое. (Часть из этих примеров в сокращенном варианте приведена в данных методических указаниях.) Рассмотрены практические вопросы программирования на языке PHP и создания полноценных интерактивных Web-сайтов. На реальных примерах показаны особенности работы с APACHE, MySQL, phpMyAdmin и с популярными готовыми решениями — Mambo, phpBB, FCKEditor и CPanel. Материал сопровождается множеством иллюстраций, схем и полезных советов: начиная с использования общедоступных интернет-сервисов, таких как курс валют, и заканчивая настройкой Web-сервера APACHE. Для более глубокого изучения PHP можно рекомендовать книги [8-11].

1.2.1. Кратко о PHP

В 1995 году Р. Лердорф разработал PHP/FI (Personal Home Page/Forms Interpreter, Персональная домашняя страница/Интерпретатор Форм).

В 1997 году Э.Гутманс и З. Сураски в поисках удобного средства для создания динамических Web-страниц обнаружили PHP/FI. Они оценили идею Расмуса, но их не устроила скорость работы и функциональные возможности данного инструмента. Поэтому они переписали его с нуля, это было возможно, так как исходный код PHP/FI был доступен абсолютно всем. Так появился PHP 3. Язык получился удобным, обладал лучшей функциональностью и скоростью работы, по сравнению с его предком. Изменения коснулись и названия языка, теперь он стал — PHP: Hypertext Preprocessor (PHP: Препроцессор гипертекста), что было очень удачным решением со стороны разработчиков.

PHP был и остается языком с открытым кодом, и в его усовершенствовании может принять участие каждый. Благодаря этому, PHP смог быстро развиваться и совершенствоваться, что и произошло в результате.

После выхода 3-й версии PHP стал завоевывать внимание многих разработчиков в области Web-программирования, и с каждым годом эта тенденция все увеличивалась, к тому же нашлось достаточное количество людей, которые выразили желание принять участие в совершенствовании PHP. В настоящее время PHP используется сотнями тысяч разработчиков. Согласно рейтингу корпорации TIOBE, базирующемся на данных поисковых систем, в мае 2016 года PHP находился на 6 месте среди языков программирования. К крупнейшим сайтам, использующим PHP, относятся Facebook, Wikipedia и др. 3 декабря 2015 года было объявлено о выходе PHP версии 7.0.0.

PHP-сценарии можно встраивать в HTML-код или создавать отдельные скрипты, **которые будут полностью обрабатываться интерпретатором PHP.**

Для того чтобы создать PHP-сценарий, следует использовать следующую конструкцию:

```
<?php...
// ... код PHP ...
?>
```

То есть код на PHP оформляется специальными тегами:

```
<?php...?>
```

Таким же образом PHP-код можно встраивать и в web-страницу. Следующий пример демонстрирует, как это делается.

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

```
<title>
<?php
echo "Заголовок";
?>
</title>
</head>
<body>
<?php
echo "<b>тело документа</b>";
?>
</body>
</html>
```

Этот сценарий выведет в заголовке окна слово Заголовок, а в теле документа полужирным начертанием текст **тело документа**. Здесь функция echo служит для вывода текста на страницу. Обратите внимание, что с ее помощью можно сразу форматировать текст, используя различные тэги (в данном примере это тег <b>, применяющий полужирное начертание шрифта).

Рассмотрим работу препроцессора PHP более подробно. После того как на вход препроцессора поступает файл, он начинает "просматривать" (или обрабатывать) его содержимое.

Как только препроцессору встречается открывающий тег, обозначающий код PHP, тут он переключается в активный режим, т. е. начинается выполнение команд, которые заключены в эти специальные теги. Некоторые команды после обработки будут заменены на HTML-теги. Некоторые (например, осуществляющие подключение к базе данных) будут просто выполнены. Как только встречается закрывающий тег, обозначающий окончание кода на PHP, препроцессор опять переключается в режим просмотра кода, пока ему не встретится очередной участок кода на PHP или пока не будет достигнут конец файла.

Как частный случай, можно рассмотреть пример, когда файл полностью состоит из команд PHP, тогда препроцессор от начала до конца обрабатывает файл. Но такой подход применяется очень редко, чаще всего PHP-команды совместно используются с HTML-тегами, а можно сказать наоборот, HTML-теги используются с PHP-командами.

Положительным моментом обработки PHP-программы препроцессором является то, что пользователь не видит исходный код программы, он только получает результат ее выполнения.

Комментарии

Есть несколько способов комментировать строки, чтобы они правильно обрабатывались интерпретатором:

```
#       строка комментария
// строка комментария
/* многострочный комментарий */
```

Следует помнить, что эти строки будут являться комментариями в PHP, но никак не в HTML. Напомним, для того чтобы комментировать строки в HTML, следует использовать конструкцию:

```
<!-- многострочный комментарий в HTML -->
```

Типы данных и переменные

В языке PHP используется шесть основных типов данных:

- integer — целое число;

- double — число с плавающей точкой;
- boolean — логический тип данных; может возвращать true (истина) или false (ложь);
- string — строковый тип данных;
- array — массив;
- object — объект; с его помощью можно создавать собственные объекты.

Для объявления переменных нет отдельного блока — их можно объявлять в любой части сценария, но перед именем переменной всегда следует ставить знак доллара ($). В языке PHP учитывается регистр символов, например, переменные $abc и $Abc — это две разные переменные. Для того чтобы объявить переменную, используется следующая конструкция: $chislo=10;

Здесь объявляется переменная chislo, которая имеет тип integer и значение 10.

Если же использовать конструкцию

```
$stroka="10";
```

переменная stroka будет иметь значение 10, но ее тип уже будет string. Это произошло из-за того, что в этом случае число занесено в кавычки. В PHP преобразование типов выполняется автоматически, но тип переменной можно задавать и вручную, например:

```
settype($chislo,integer);
```

Функция settype преобразует переменную, имя которой указано в первом параметре, в тип, который указан во втором параметре. В данном примере переменная $chislo будет преобразована в тип integer.

Для того чтобы узнать тип переменной, следует использовать функцию gettype. У нее всего один параметр — имя переменной, тип которой нужно вернуть. Например, gettype ($chislo).

Для определения типа можно использовать и другие функции. Например, можно узнать, является ли переменная определенным типом. |

- is_integer — возвращает true, если тип переменной integer.
- is_double — возвращает true, если тип переменной double.
- is_boolean — возвращает true, если тип переменной boolean. '
- is_string — возвращает true, если переменная строкового типа.
- is_array — возвращает true, если переменная является массивом.
- is_object — возвращает true, если переменная — объект.

Все эти функции принимают как аргумент имя переменной, тип которой следует проверить.

Преобразования типов можно выполнять и следующим образом:

```
$chislo="10"; //$chislo сейчас имеет тип string
$chislo=(int)$chislo; //переводим в тип integer
```

Для того чтобы проверить, объявлена ли переменная, используется функция isset(). В качестве параметра она принимает имя переменной, а затем возвращает значение true, если переменная объявлена. Для удаления переменной можно использовать функцию unset(). Например, команда unset ($perem) удалит переменную $perem и освободит память, которую эта переменная занимала.

Массивы

Есть несколько способов объявления массива. Рассмотрим первый.

```
$mass = array("элемент1","элемент2","элемент3");
```

В этой строке мы объявили массив с именем mass на три элемента:

```
$mass[0]= "элемент1"; $mass[1]= "элемент2"; $mass[2] = "элемент3";
```

Заметим, что индексация в массиве начинается с нуля. Массивы можно объявлять и более простым способом.

```
$mass[]= "элемент1"; $mass[]= "элемент2"; $mass[]= "элемент3";
```

Это будет аналогично созданию массива с указанием индексов элементов. Не указывая индексы, мы избавляем себя от банальных ошибок, которые могут возникнуть при затирании какого-то элемента массива. Если мы напишем $mass[]=2; то по умолчанию занесем значение 2 в следующую по очереди ячейку массива.

Пример:

```
$mass[0]="элемент1"; $mass[l]= "элемент2"; $mass[6]= "элемент3"; $mass[]= "элемент4";
```

В этом случае строковое значение элемент4 будет занесено в седьмую ячейку.

Константы

Константа содержит значение, которое нельзя изменять на протяжении всего сценария. Для ее объявления следует использовать функцию define, например

```
define(const, 20, true)
```

Это означает, что объявлена константа с именем const, которая имеет значение 20. Третий параметр указывает, будет ли константа чувствительна к регистру. Если указано true, то регистр символов учитывается.

Аналогично функции isset() для переменной, существует функция defined(), которая проверяет существование константы. Функция возвращает true, если константа объявлена.

В PHP также определены несколько стандартных констант (с ними можно работать как с обычными пользовательскими константами).

Более подробно с описанием PHP можно ознакомится, например, в [7,9,11].

Отметим основные преимущества PHP [6]**.**

- Фрагменты PHP-кода можно внедрять прямо в HTML-документ и наоборот. Главное при этом не забыть установить для файла расширение PHP.
- Так как PHP специально разработан для Web, многие вещи реализуются намного проще, чем если бы то же самое выполнялось с помощью других языков.
- PHP очень прост в освоении и позволяет достаточно быстро решать поставленные перед ним задачи.
- PHP бесплатен.
- PHP — кроссплатформенный язык, что не привязывает его к определенной операционной системе или определенному Web-серверу. В результате получается свобода в выборе программного обеспечения при работе с PHP.
- Совместимость программ, разработанных в ранних версиях PHP, с его более поздними версиями.
- PHP очень гибок. Он не зависит от браузера, т. к. программа выполняется на сервере, а результат возвращается в виде HTML- документа.
- Простота работы с базами данных: PHP может работать с различными СУБД, например, MySQL, MS SQL Server, ORACLE, InterBase, Firebird и др. При-

чем работа может быть организована с использованием различных технологий доступа к данным.

- PHP является объектно-ориентированным языком, что позволяет вам воспользоваться всеми преимуществами этой технологии.
- PHP располагает огромным количеством стандартных функций, которые позволяют решить множество задач за минимальное количество времени.
- PHP многогранен, вам предоставляется множество готовых инструментов для вашего творчества, например, вы можете работать с файлами формата PDF, электронной почтой или даже разработать интернет-магазин, который будет использовать средства электронной коммерции, такие как WebMoney или PayPal.
- PHP может работать с XML.
- В настоящее время в Интернете есть огромное количество готовых решений, написанных на PHP, начиная от простых счетчиков на вашу страницу и заканчивая системами управления и построения сайтов, которые позволяют до максимума упростить процесс создания и сопровождения сайта.
- Наконец, вы можете принять участие в совершенствовании PHP, т. к. его код открыт для всех.

1.2.2. Среда разработки PHP-программ

Любую программу удобно разрабатывать в специальной среде, конечно, можно писать и в Блокноте. Однако недостатки этого выбора сразу налицо: во-первых, синтаксис языка не подсвечивается, во-вторых, нет ни удобной справки, ни подсказок, ни возможностей автоматически вставлять наиболее распространенные команды или фрагменты кода, ни многих других, которые не только помогают сократить время разработки программы, но и сделать этот процесс максимально удобным и комфортным.

В настоящее время существует большое количество редакторов, в которых можно создавать и редактировать файлы PHP.

Информацию по самым разнообразным редакторам можно найти на сайтах:

http://www.thelinuxconsultancy.co.uk/phpeditors.php
http://www.php-editors.com/review/

Ниже рассматривается Сборник программ (редакторы кода) для веб-разработчиков **WeBuilder 2015.**

1.2.2.1. WeBuilder 2015 v13.3.0.165 Final

Сборник программ (редакторы кода) для веб-разработчиков доступны в четырех изданиях:

HTMLPad | Rapid CSS Editor | Rapid PHP Editor | WeBuilder

WeBuilder - редактор веб-страниц с подсветкой синтаксиса HTML, CSS, javascript, VBScript, PHP, ASP, SSI и Perl. Кроме редактора веб-страниц включает в себя встроенный веб-браузер, файловый менеджер и FTP клиент. Программа оснащена многочисленными подсказками по различным языкам программирования, позволяет производить замену кода по многим параметрам, проверять правильность тегов и т.д. Вы можете быстро и легко вставлять таблицы, любые элементы web-форм, скрипты, просматривать страницы во внутреннем браузере. Имеется настраиваемая цветная подсветка программного кода, браузер папок, браузер элементов кода (переменные, классы, библиотека элементов, библиотека тэгов), встроенный FTP-клиент, валидатор кода. Также вы сможете создавать и редактировать свои собственные сниппеты, отображать,

настраивать и располагать панели по своему вкусу, создавать документы из шаблонов, которые можно создавать и править самостоятельно.

Возможности программы:

- Редактирование HTML, CSS, javascript, PHP, ASP и Ruby кода.
- Быстрое начало работы с помощью встроенные мастеров HTML и CSS.
- Полностью настраиваемый редактор, меню, панели, горячие клавиши.
- Эффективное использование многократно встречающихся фрагментов кода.
- Встроенный FTP-менеджер.
- Создание и редактирование шаблонов страниц и сниппетов (фрагментов исходного текста или кода программы, применяемых в поисковых системах, текстовых редакторах и средах разработки).
- Легкий в изучении и использовании интерфейс.
- Подсветка синтаксиса HTML, CSS, javascript, VBScript, PHP, ASP, WML, XML.
- Подсветка синтаксиса ASP.Net, C#.Net, Ruby, eRuby, Perl, SQL.
- Поддержка Юникода (UTF-8).
- **Отладка PHP-кода с помощью встроенного PHP-дебаггера xDebug.**
- Сохранение и открытие файлов по ftp-протоколу.
- Управление проектами.
- Работа с базами данных MySQL, PostgreSQL, Firebird, Interbase.
- Поиск и замена с использованием регулярных выражений.
- Проверка орфографии в html-коде и txt-файлах.
- Функция "сворачивания кода".
- Готовые к использованию шаблоны страниц

1.2.2.2. Денвер

Как известно, для выполнения скриптов и их отладки необходим Web-сервер, который смог бы принимать запросы пользователя и отправлять ему соответствующие ответы. Для того чтобы Web-сервер мог выполнять скрипты, написанные именно на PHP, необходимо установить PHP. Если вы захотите работать с базой данных, а выполнение последующих лабораторных работ по дисциплине «Интернет-технологии и системы» предусматривает обязательное ее использование, то вам придется установить СУБД.

Одним из самых распространенных Web-серверов является Apache, а одной из самых распространенных СУБД, предназначенных для реализации Web-задач, является MySQL. Именно эти два программных продукта наиболее часто используют в связке с PHP. Дело в том, что они достаточно давно развиваются в очень тесном сотрудничестве друг с другом, направленном на улучшение положительных сторон каждого из них. Таким образом, при соответствующей настройке стабильность и скорость работы вашего сайта будет на должном уровне.

Для того чтобы приступить к установке, нужно сначала скачать (только не торопитесь пока это делать) соответствующие дистрибутивы. В табл. 1.1 приводится необходимая информация [6].

Таблица 1.1. Дистрибутивы для установки
Web-сервера с поддержкой PHP и MySQL

Название программы	Официальный сайт (откуда можно скачать дистрибутив)	Размер дистрибутива
Apache (Web-сервер)	www.apache.org	4,9 Мбайт
PHP	www.php.net	8,9 Мбайт
СУБД MySQL	www.mysql.com (справка на русском языке доступна на www.mysql.com/doc/ru/index.html)	16,9 Мбайт
phpMyAdmin	www.phpmyadmin.net	3,4 Мбайт
Итого:		34,1 Мбайт

phpMyAdmin — популярное средство, позволяющее сделать процесс работы с базой данных очень удобным, без него вам бы пришлось осуществлять большинство действий (например, создание таблиц) в командной строке, что очень неудобно.

Не удивительно, если некоторых испугает итоговая цифра. Не всем захочется качать такой объем данных через Интернет, к тому же все это придется настраивать вручную. То есть не в лучших традициях Windows-приложений, когда запустил установочный файл, пару раз нажал кнопку Next, и все готово. Придется открывать конфигурационные файлы и прописывать настройки вручную.

Заметим, что конфигурационные файлы настроек составляют [7]:

php.ini (PHP) более 20 стр.;

httpd.conf (сервер apache) —— около 20 стр.

В данном случае есть более простой выход, под названием Денвер. С помощью него вы можете установить Apache, PHP, MySQL, phpAdmin буквально за несколько минут, при этом размер дистрибутива, который вам придется скачать, будет равен (3-5.4) Мбайт.

На официальном сайте www.denwer.ru он характеризуется как:

"Джентльменский набор Web-разработчика ("Д.н.w.р" читается "Денвер") — набор дистрибутивов, используемый Web-разработчиками (программистами и дизайнерами) для отладки сайтов на "домашней" (локальной) Windows-машине без необходимости выхода в Интернет".

- Денвер устанавливается в один-единственный каталог и вне его ничего не изменяет. Он не пишет файлы в Windows-директорию и не прописывает никакой информации в реестре. При желании вы можете даже поставить себе сразу два Денвера, и они не будут конфликтовать.
- Никакие "сервисы" NT/2000 не "прописываются". Если вы запустили Денвер, то он работает. Если завершили — то перестает работать, не оставляя после себя следов.
- Системе не нужен деинсталлятор — достаточно просто удалить каталог.
- Все конфигурирование и настройка под конкретную машину происходит автоматически.

- Денвер работает в ОС Windows XP/Vista/7/8.
- Денвер автономен: он может располагаться в любой директории на диске (или даже на флэш-накопителе). Он также не изменяет системных файлов Windows, так что может быть деинсталлирован путем простого удаления своей папки.
- Модульность, расширяемость, компактность. Нет необходимости выкачивать многомегабайтные дистрибутивы отдельных компонентов. Базовая версия Денвера, включающая Apache+SSL+PHP5+MySQL5 +phpMyAdmin, имеет размер всего около 5.4МБ и при этом полностью функциональна.
- Централизованная система запуска и остановки всех компонентов Денвера. Благодаря своей автономности, после остановки Денвер полностью "изчезает" из системы и может быть скопирован в другую директорию или даже удален.
- Эмулятор sendmail: возможность отладки скриптов, отправляющих почту. Все письма, созданные в PHP-скриптах, не отправляются наружу, а складываются в специальную директорию /tmp/!sendmail.

Установка Денвера

Для начала необходимо скачать последнюю версию дистрибутива, для этого достаточно зайти по ссылке http://www.denwer.ru/dis/latest и скачать предложенный файл.

После того как файл будет закачан, можно приступить к установке Денвера. Файл представляет собой самораспаковывающийся архив, поэтому выполните на нем двойной щелчок левой кнопкой мыши, после чего появится окно Инсталлятор.

Нажимаем Да, архив начнет распаковываться. По окончании распаковки вы увидите окно DOS-программы. Денвер написан на Perl, это можно заметить по заголовку окна (в конце заголовка можно увидеть perl.exe). Нажимаем Enter, как нам и предлагают. Далее видим следующее DOS- окно.

Здесь нам предлагается ввести путь и имя директории, в которую будет установлен Денвер, а вместе с ним и PHP, Apache, MySQL, phpMyAdmin. По умолчанию предлагается путь C:\WebServers, рекомендуется оставить его. Далее нажимаем <Enter> После чего нам будет задан вопрос, действительно ли мы уверенны в том, что хотим установить Денвер в указанную на предыдущем шаге папку. Нажмите клавишу <Y> и затем <Enter>.

Теперь нас информируют о следующем шаге установки — выбор буквы виртуального диска, который будет появляться при запуске Денвера. Нажмите <Enter>. После чего вы можете ввести любую букву или согласиться с предложенной — Z, рекомендуется сделать выбор в пользу последнего варианта. Нажмите <Enter>. Все готово к копированию файлов Денвера в указанный вами путь, необходимо еще раз нажать <Enter>, чтобы приступить к этому процессу.

После того как копирование завершится, вам будет предложено выбрать режим запуска Денвера, возможны следующие варианты:

- автоматический — комплекс будет запускаться автоматически при входе в ОС;
- ручной — чтобы начать работу с комплексом, необходимо будет запустить его с помощью специального ярлыка, по завершении работы производится остановка комплекса также с помощью специального ярлыка.

Предлагается выбрать второй вариант, как более удобный, т. к. в этом случае вы будете запускать комплекс только тогда, когда он вам нужен. Нажмите <2>. Далее вас спросят:

"Хотите ли вы, чтобы на рабочем столе были размещены ярлыки, предназначенные для работы с комплексом?"

Рекомендуется согласиться и нажать <Y>. После этого процедура установки будет закончена. Нажмите <Enter>. Окно установки закроется, и вы увидите на рабочем столе три новых ярлыка:

- Start servers — запуск комплекса
- Stop servers — остановка работы комплекса;
- Restart servers — перезапуск комплекса (т. е. сначала будет осуществлена остановка комплекса, а потом сразу его запуск).

После установки Денвера вам будут доступны: Apache, PHP, MySQL, phpMyAdmin, Perl.

Вообще PHP может быть установлен двумя разными способами:

- как внешняя программа — при этом каждый раз, когда пользователь запрашивает PHP-скрипт, на сервере запускается новый процесс в лице программы php-cgi.exe;
- как модуль Web-сервера — в этом случае PHP будет работать как одно целое вместе с Web-сервером.

У каждого из вариантов есть свои плюсы и минусы. Первый вариант позволяет добиться большей безопасности для Web-сервера, второй обеспечивает лучшее быстродействие при выполнении скриптов, а также позволяет получить доступ к дополнительным возможностям PHP, но более сложен в настройке по сравнению с первым. *Установка Денвера предполагает, что PHP будет работать как модуль Web-сервера Apache.*

Стоит также отметить, что если на вашем компьютере установлен Firewall, то для корректной работы Web-сервера необходимо прописать в нем соответствующие правила (или установить разрешения). Если вы используете Windows XP и у вас установлен Microsoft IIS, то рекомендуется остановить его работу, перед тем как запускать Денвер.

Последним советом, перед тем как приступить к работе с Денвером, будет такой: когда вы тестируете или отлаживаете ваши программы, т. е. у вас запущен Денвер, не рекомендуется находиться в Интернете или быть подключенным к локальной сети, так как иначе ваша система может быть объектом атаки. Если для вас это не приемлемо, то есть вам нужна связь с внешним миром, тогда настройте соответствующим образом ваш Firewall.

Работаем с Денвером

Для того чтобы запустить Денвер (запустить ваш персональный Web-сервер), необходимо выполнить двойной щелчок левой кнопкой мыши на ярлыке Start servers. После чего вы увидите окно запуска, оно будет отображаться несколько секунд, потом исчезнет, а в трее появится пиктограмма в виде пера, обозначающая работу Web-сервера Apache на вашей системе.

После запуска Денвера на вашем компьютере будет создан специальный виртуальный диск, если вы не сменили его имя при установке Денвера, то это будет диск Z:

Когда вы хотите закончить работу с Web-сервером, например, вам нужно перегрузить компьютер, рекомендуется воспользоваться ярлыком Stop servers, чтобы остановить Apache.

Ярлык Restart servers осуществляет перезапуск Web-сервера. Это действие необходимо, когда вы совершили некоторые изменения и хотите, чтобы они вступили в силу. Под изменениями могут подразумеваться создание нового виртуального хоста (или другими словами: добавление нового сайта, например, primer.ru), внесение изменений в конфигурацию Apache, PHP или MySQL.

Ярлыки Start Servers, Stop Servers и Restart Servers ссылаются на файлы Start.exe, Stop.exe, Restart.exe, находящиеся в папке WebServers\etc\.

Эти ярлыки представляют собой систему управления запуском и завершением работы комплекса Денвер или, другими словами, вашего персонального Web-сервера.

Итак, у вас запущен Web-сервер, предлагается открыть браузер и набрать в нем http://localhost, после этого нажмите <Enter>. Вы увидите тестовую страницу Денвера (Рис. 7).

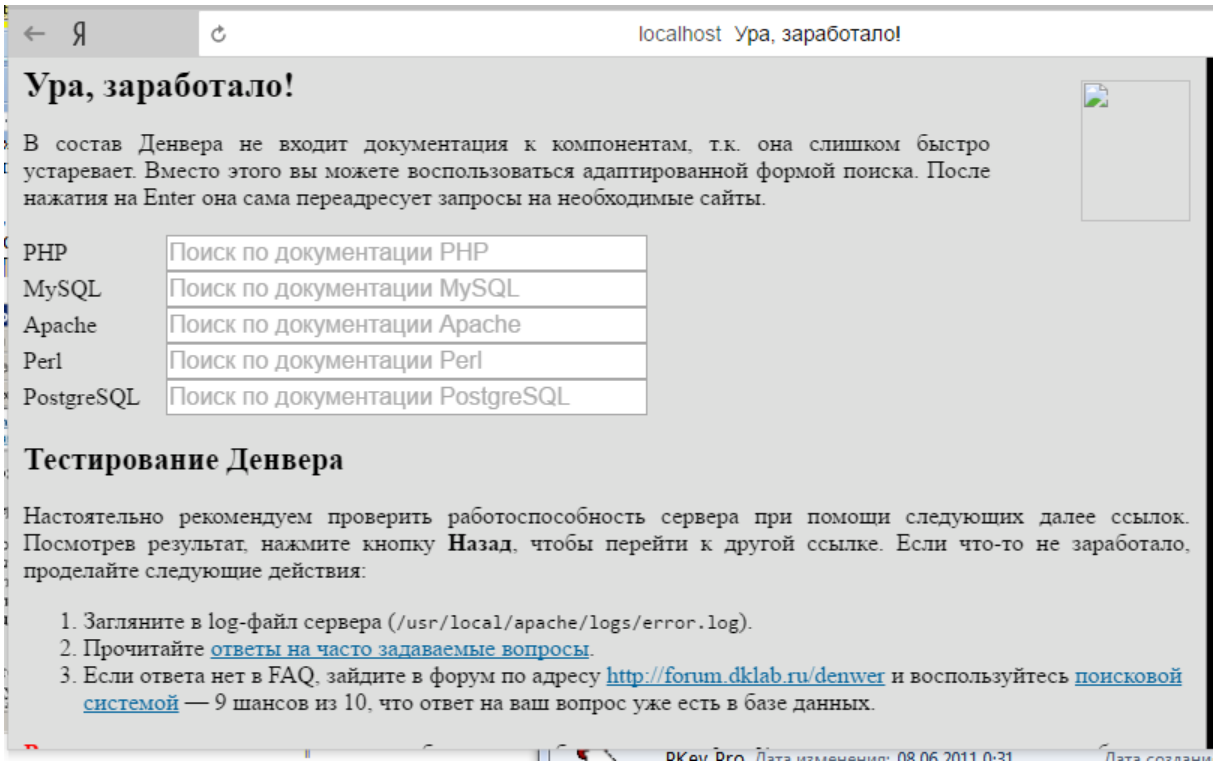


Рис. 7. Тестовая страница Денвера

Выберите пункт Утилиты, вы увидите утилиты и инструменты, которые включает в себя Денвер (рис. 8).

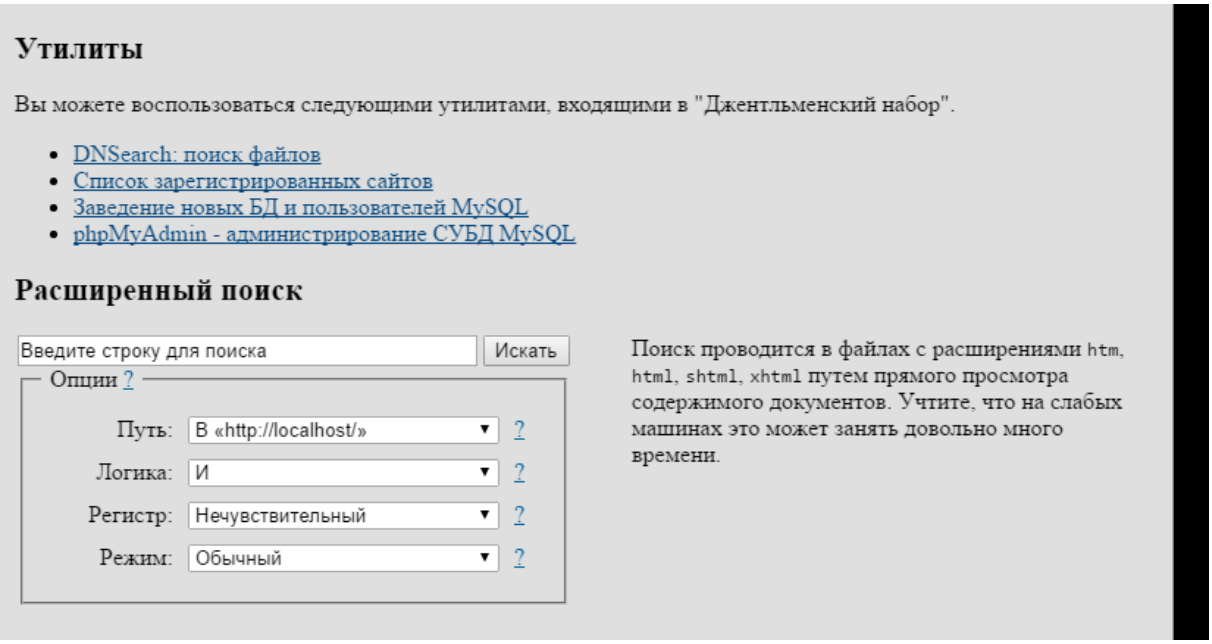


Рис. 8. Утилиты и инструменты Денвера

Рассмотрим их более подробно:

- DNSearch: поиск файлов — аналог поисковых систем в Интернете, например, таких как Rambler или Google, т. е. вы вводите искомую фразу и устанавливаете условия поиска, после чего нажимаете кнопку Искать;
- Список зарегистрированных сайтов — данный инструмент позволяет отобразить в виде списка все зарегистрированные сайты на вашем Web-сервере;
- Заведение новых БД и пользователей MySQL — название данного инструмента говорит само за себя.
- phpMyAdmin - администрирование СУБД MySQL — это система администрирования MySQL через Web-интерфейс. Позволяет создавать базы данных, таблицы, выполнять к ним запросы и т. д.;

Выберите пункт Список зарегистрированных сайтов, появится Список зарегистрированных сайтов на вашем Web-сервере.

Выберите testl.ru, вы должны увидеть следующее — рис. 9.

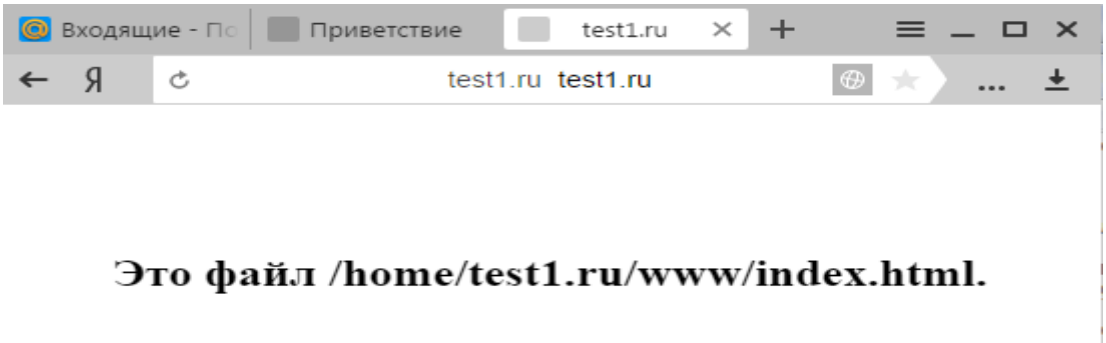


Рис. 9. Тестовая страница Денвера

Таким образом, Вы только что протестировали работоспособность установленного комплекса Денвер.

Денвер изнутри

В официальной документации к Денверу сказано, что он "похож на небольшой UNIX". И это на самом деле верно, т. к. организация каталогов создана по подобию этой операционной системы. Как говорилось ранее, после запуска Web-сервера в системе появляется виртуальный диск (как правило, с именем Z:). Откройте его в проводнике и посмотрите, что на нем располагается (см. рис. 10).

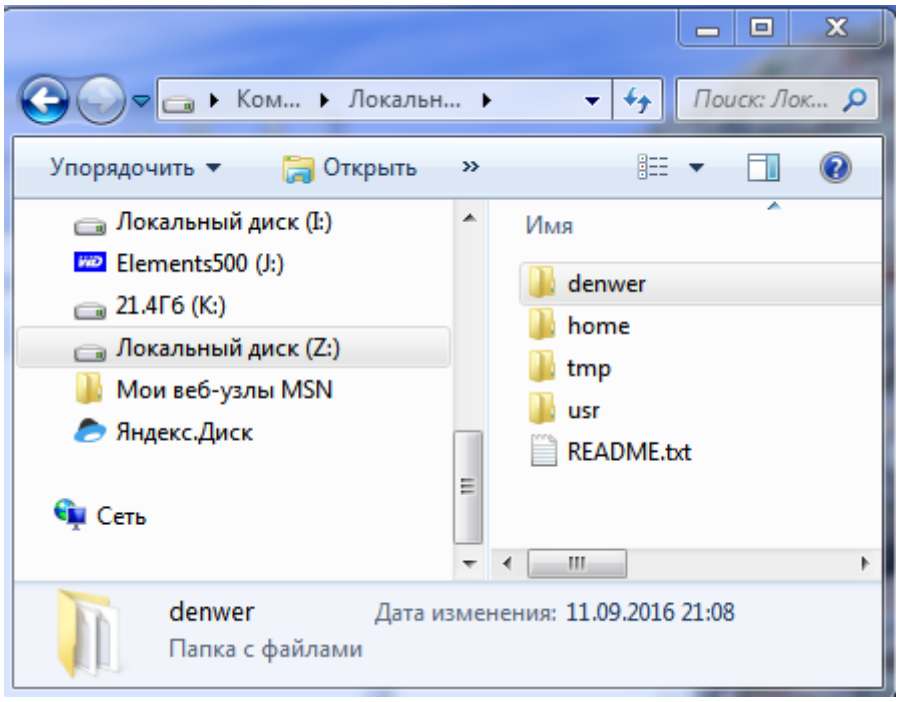


Рис. 10. Виртуальный диск Z, появившийся после запуска Денвера

Рассмотрим назначение каждой папки:

- denwer — хранит скрипты конфигурирования Денвера, здесь же находится его конфигурационный файл CONFIGURATION.txt, все его параметры обозначены русскими комментариями, в целях эксперимента можно посмотреть содержимое данного файла, но править его вручную не рекомендуется, т. к. это может негативно сказаться на работе всего комплекса;
- home — с данной папкой вы будете часто работать, так как в ней будут храниться ваши сайты (или виртуальные хосты);
- tmp — папка предназначена для хранения временной информации, такой как пользовательская сессия. В данной папке, вернее, в ее подкаталоге !sendmail будут храниться все письма, отправленные с помощью функции mail ().
- usr — системная папка, в которой хранятся Apache, PHP, MySQL и некоторые другие элементы.

1.2.3. Создаем свой сайт

По умолчанию, после установки Денвера, у вас будут доступны несколько виртуальных хостов, например, если вы наберете в браузере www.testl.ru, то у вас отобразится тестовая страничка (см. рис. 9).

Для того чтобы добавить новый сайт, достаточно создать в каталоге home папку с именем будущего сайта, только ее имя должно быть без www. Например, вы хотите создавать ваши программы, работая с виртуальным сайтом www.myprogram.ru, для этого вам необходимо создать папку myprogram.ru в каталоге home. Но это еще не все, внутри вновь созданной папки вам нужно создать еще одну с именем www (рис. 11).

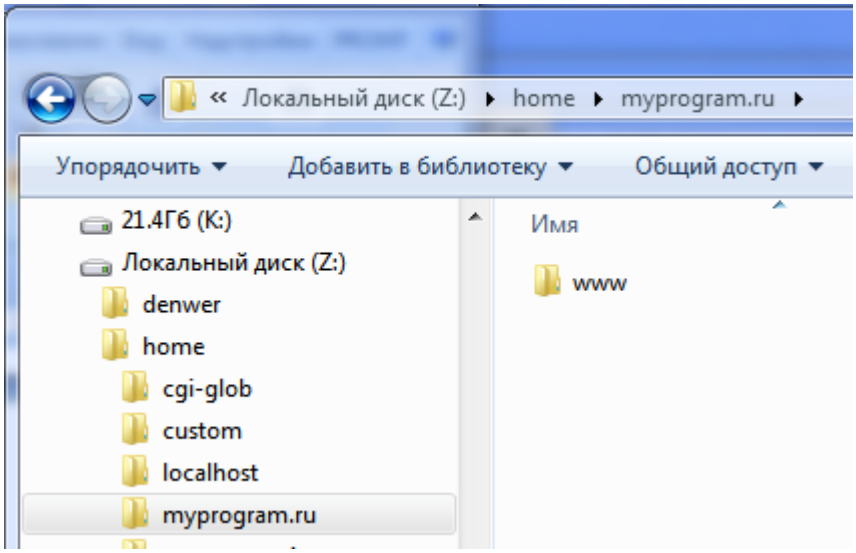


Рис. 11. Создание собственного виртуального сайта www.myprogram.ru

После этого останется только перезапустить Web-сервер с помощью ярлыка Restart servers, и ваш новый сайт будет готов к работе.

Конфигурационные файлы

Каким бы программным обеспечением вы не пользовались, если вы знаете, каким способом оно настраивается, то вам намного легче разобраться в нестандартных ситуациях, которые могут возникнуть при использовании данного ПО. Основное средство настройки Apache, PHP и MySQL — это специальные конфигурационные файлы. Все они открываются с помощью обычного блокнота. Конечно, никаких изменений в настройки комплекса мы вносить не будем, но эта информация может вам пригодиться в будущем.

- denwer\CONFIGURATION.txt — файл конфигурации Денвера (не рекомендуется вносить изменения в этот файл).
- usr\local\apache\conf\httpd.conf — файл конфигурации Apache.
- usr\local\php5\php.ini — файл конфигурации PHP.
- usr\local\ mysql-5.5 \my.cnf — файл конфигурации MySQL,

Удобство Денвера заключается еще и в том, что большинство параметров конфигурационных файлов переведено на русский язык, таким образом, процесс настройки упрощается.

1.2.4. Примеры серверных скриптов на PHP

Здесь мы рассмотрим создание нескольких серверных скриптов на PHP, а именно:

1. Получение системной информации об Apache, PHP, MySQL, ...
2. Просмотр переменных окружения сеанса выполнения Вашего скрипта.
3. Вывод ваших реквизитов: IP-адреса; Браузера; Адреса программы, которую вы запросили.
4. Счетчик посещений и просмотр его работы.

Выполнение этих серверных скриптов разместим на Вашем сайте

http://myprogram.ru/

При этом будут использованы Примеры серверных скриптов, рассмотренные в [6].

Создайте файл index.html следующего содержания:

```
<html>
<body>
<table width=100% height=100%>
<tr><td align=left>
<h2 align=center>Формирование серверных скриптов на <b>
PHP</b></h2>
<p>1. Информация о PHP. <a href="info.php">Получите системную информацию об Apache, PHP, MySQL, ...</a>
<p>2. Переменные окружения <a href="var_of_enviromentr.php">
Просмотрите переменные окружения сеанса выполнения Вашего скрипта</a>
<p>3. Ваши реквизиты: <a href= "info_user.php">IP-адрес; браузер; Адрес программы, которую вы запросили</a>
<p>4. Счётчик посещений. <a href= "counter\mypage.php">Просмотрите работу Счетчика посещений</a>
 </td></tr>
</table>
</body>
</html>
```

Сохраните файл index.html на вашем виртуальном сайте по адресу Z:\home\myprogram.ru\www.

Результат просмотра файла index.html в браузере представлен на рис. 12

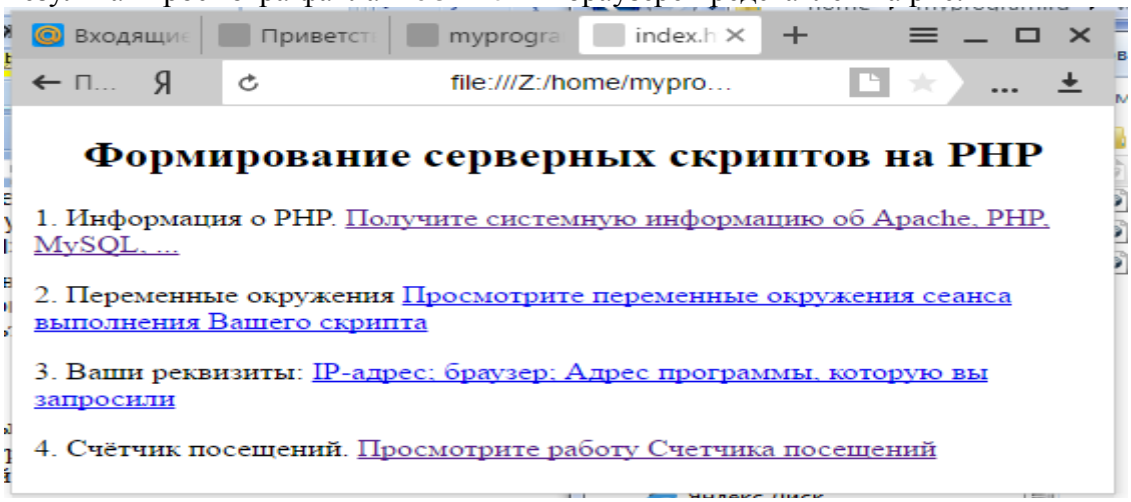


Рис. 12. Результат просмотра файла index.html в браузере

Как вы, вероятно уже догадались, эта главная страница вашего сайта предназначена для загрузки соответствующих серверных скриптов, обеспечивающих решение поставленной задачи.

1.2.4.1. Информация о PHP

Сейчас мы напишем небольшую программу. Создайте файл info.php следующего содержания:

```
<?php
phpinfo()
?>
```

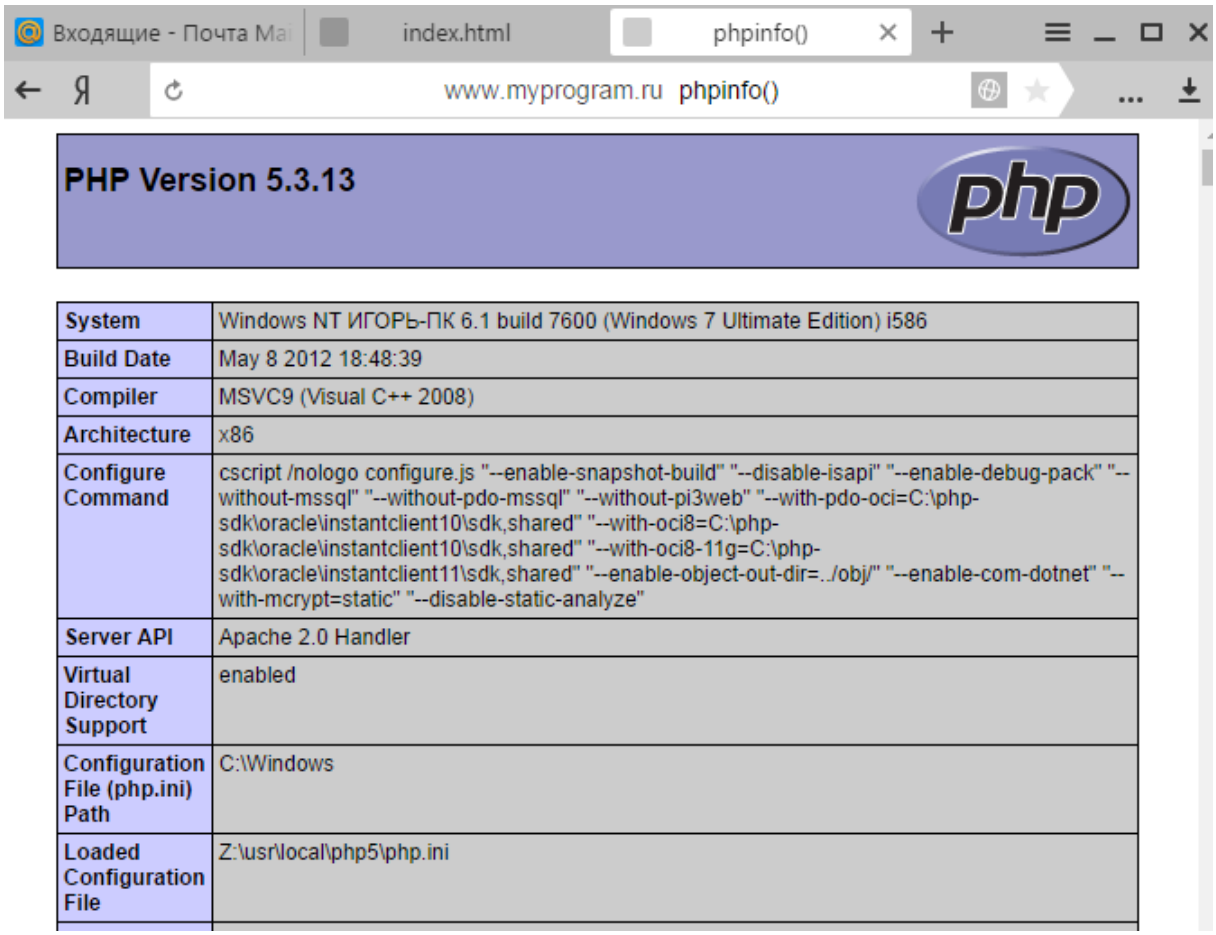
Теперь сохраните этот файл на вашем виртуальном сайте myprogram.ru в папке www.

Запустите с помощью ярлыка Start Servers сервер Apache. Наберите в браузере адрес http://www.myprogram.ru/, нажмите <Enter>. Вы увидите результат просмотра файла index.html, аналогичный рис. 12. Кликните на ссылку Получите системную информацию об Apache, PHP, MySQL, ... Вы увидите системную информацию о Apache, PHP и MySQL (рис. 13).

Как видите, с помощью специальной команды phpinfo() мы можем получить системную информацию о Apache, PHP, MySQL и других компонентах.

1.2.4.2. Просмотр переменных окружения

Как известно, к одному скрипту могут обращаться одновременно множество пользователей. Для каждого пользователя на сервере создается своеобразный сеанс выполнения скрипта. То есть в результате каждый пользователь работает в своем сеансе, не затрагивая других. Когда вы запрашиваете через браузер какую-то страничку, то это не вся информация, которая отправляется на сервер, кроме этого, туда уходит много дополнительных данных, как системных, необходимых для взаимодействия браузера с сервером, так и пользовательских. Эти данные называются переменными окружения и хранятся в специальном массиве с именем $_server.



System	Windows NT ИГОРЬ-ПК 6.1 build 7600 (Windows 7 Ultimate Edition) i586
Build Date	May 8 2012 18:48:39
Compiler	MSVC9 (Visual C++ 2008)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=C:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=C:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static" "--disable-static-analyze"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	Z:\usr\local\php5\php.ini

Рис. 13. Информация о Apache, PHP и MySQL

Создайте файл var_of_enviromentr.php, в котором мы выведем все переменные окружения на экран.

Листинг файла var_of_enviromentr.php

```
<pre>
//Выводим все переменные окружения на экран
<?php
print_r($_SERVER);
?>
<pre>
```

Сохраните файл var_of_enviromentr.php по адресу Z:\home\myprogram.ru\www Запустите сервер Apache. Наберите в браузере адрес http://www.myprogram.ru/, нажмите <Enter>. Вы увидите результат просмотра файла index.html, аналогичный рис. 12. Кликните на ссылку Просмотрите переменные окружения сеанса выполнения Вашего скрипта. Вы увидите переменные окружения сеанса выполнения Вашего скрипта (рис. 14).

1.2.4.3. Вывод реквизитов

Разработаем еще одну программу (листинг 2.1), которая будет выводить приветствие пользователю, а также:

- его IP-адрес, информация о котором хранится в $_server ['remote_addr'];

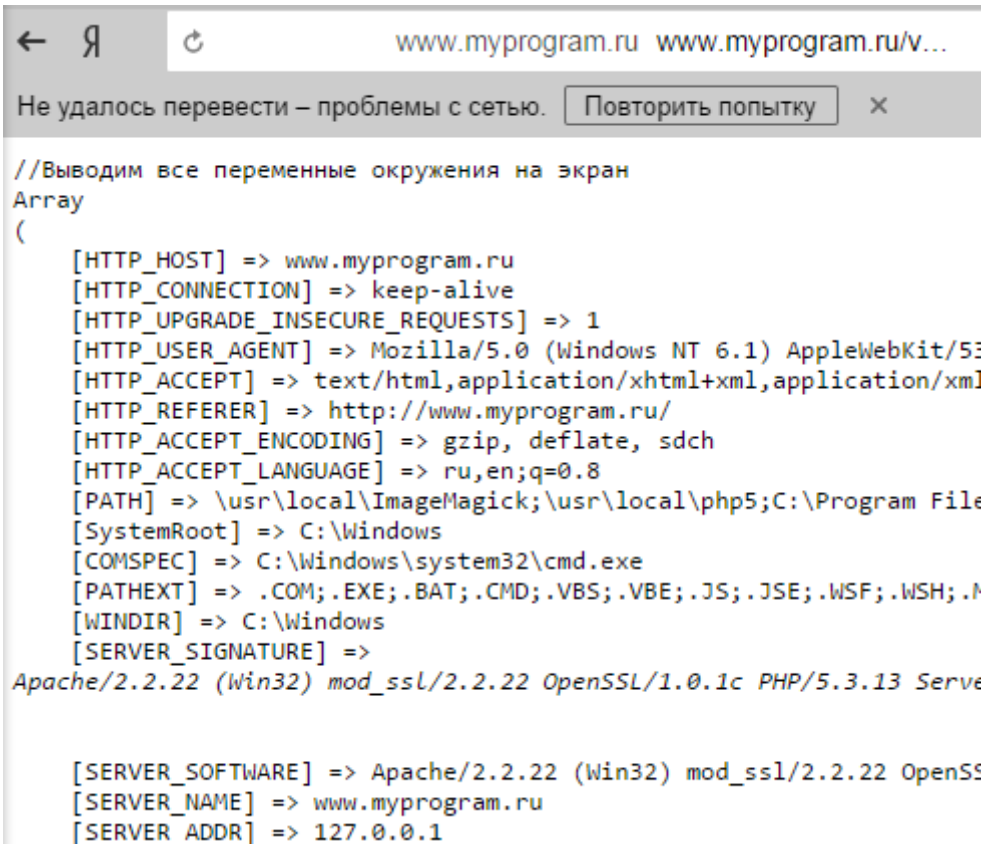


Рис. 14. Переменные окружения

- используемый им тип браузера, информация о котором хранится в $_SERVER['HTTP_USER_AGENT'];
- сведения о местонахождении на сервере запрошенного пользователем файла, информация о котором хранится в $ SERVER ['SCRIPT FILE-NAME'].

Листинг 2.1. Файл info_user.php

```
<?php
echo "Уважаемый пользователь!"."<BR>";
echo "ВАШ IP-адрес: ".$_SERVER['REMOTE_ADDR']."<BR>";
echo "ВАШ браузер: ".$_SERVER['HTTP_USER_AGENT']."<BR>";
echo "Адрес программы, которую вы запросили: ";
echo $_SERVER['SCRIPT_FILENAME']."<BR>";
?>
```

Обратите внимание, здесь точки служат для объединения частей строк в одну строку.

Результат выполнения скрипта представлен на рис. 15.

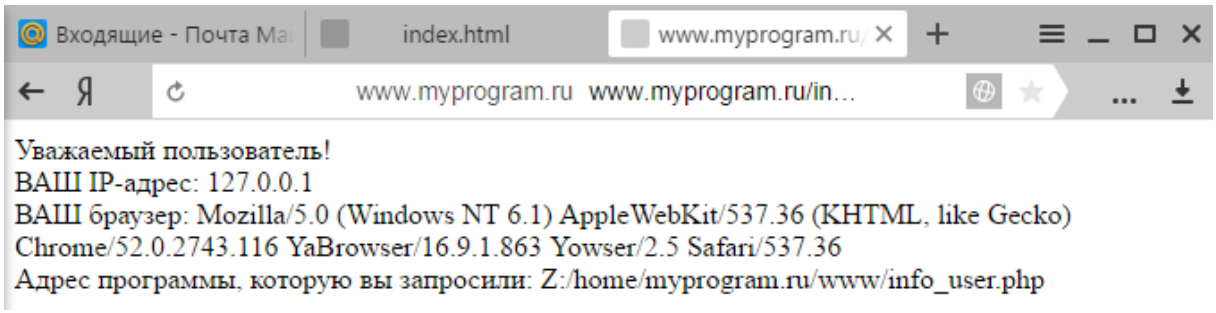


Рис. 15. Результат выполнения скрипта info_user.php

Следует отметить, что переменные окружения очень удобно использовать для реализации лог-журнала, в который заносится информация о каждом посетителе вашего сайта. Таким образом, вы можете узнавать их общее количество и отслеживать, растет ваша аудитория или нет.

1.2.4.4. Счетчик посещений

У большинства сайтов, размещенных в Интернете, вы можете видеть счетчики, которые отображают, какое число людей посетили данную страничку. Здесь мы рассмотрим реализацию собственного счетчика посещений сайта.

Предположим, вы создали HTML-страничку следующего вида:

Листинг 2.2. HTML-страничка

```
1  <html>
2
3  <head>
4  <title>Домашняя страничка Иванова Пети</title>
5  </head>
6  <body>
7
8  <H1>Домашняя страничка Иванова Пети</H1>
9  <BR>
10  <BR>
11   Привет!
12   Меня зовут Петя.
13   Я изучаю PHP.
14   </body>
15   </html>
```

Сохраните ее под именем mypage.html и запустите посредством браузера (рис. 16).

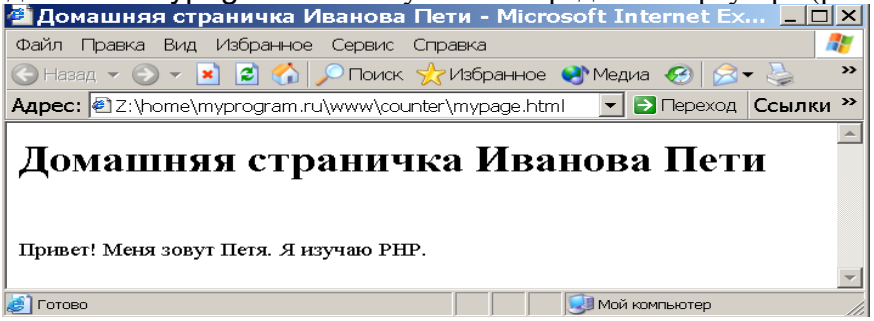


Рис. 16. Простая HTML-страница

Необходимо, чтобы на этой страничке размещался счетчик, который показывал бы количество посещений данной страницы. Для этого разработаем отдельную программу с именем counter.php (листинг 2.3).

Листинг 2.3. Файл counter.php

```
1  <?php
2  //Открываем файл counter.txt
3  $f=fopen("counter.txt","a+t") or die("Не могу открыть файл");

5  //Блокируем файл, чтобы никто не мог к нему обратиться
6  //пока мы с ним не закончим работу
7  flock($f,2);
8
9  //Читаем в переменную $s значение счетчика
10 $s=fgets($f);
11 //увеличиваем значение $s на 1
12 $s=$s+1;
13 //Удаляем все содержимое файла counter.txt
14 ftruncate($f,0);
15 //записываем новое число
16 fputs($f,$s);
17
18 //снимаем блокировку
19 flock($f,3);
20 //закрываем файл, теперь с ним могут работать другие
21 fclose ($f) ;
22 //выводим показания счетчика на экран
23 echo $s;
24 ?>
```

Логика работы данной программы построена следующим образом: число посещений хранится в обычном текстовом файле с именем counter.txt, при каждом новом обращении к описанной ранее программе это число увеличивается на единицу.

В PHP с файлом можно работать в двух режимах:

- текстовом — означает, что вы работаете со всей информацией, содержащейся в файле, как с текстом, состоящим из строк. Данный вариант можно сравнить, как будто работа ведется в обычном блокноте;
- бинарном — подразумевает, что файл состоит из байтов данных. Этот режим в основном предназначен для работы со всеми типами файлов отличных от текстового, например, графических.

Прежде чем работать с файлом, его нужно открыть, что мы и делаем в строке 3, с помощью функции fopen (), она имеет следующий синтаксис:

fopen(имя файла, режим работы);

Данная функция возвращает специальное число, которое называется дескриптор открытого файла, его нужно запомнить (сохранить в переменной), чтобы в дальнейшем можно было обращаться к файлу, потому что вся работа с ним осуществляется через этот дескриптор. Именно это мы и делаем, присваивая результат работы fopen() обычной переменной $f. То есть теперь мы можем работать с файлом через обычную пере-

менную, и нам не нужно постоянно указывать ни имя файла, ни режим работы с ним (к режимам работы мы еще вернемся чуть позже).

Дескриптор файла относится к типу данных resource, поэтому переменная $f будет тоже этого типа. Тип данных resource предназначен для хранения ссылки на внешний ресурс. В данном случае в роли ссылки выступает специальное число, которое указывает на открытый нами файл с определенным режимом работы.

Как уже было сказано ранее, одним из параметров функции fopen() является режим работы. На самом деле этот параметр имеет две составляющие, одну мы уже рассмотрели (вид — текстовый и бинарный), а другую, назовем ее условно способ работы с файлом, опишем сейчас — табл. 1.2..

Таблица 1.2. Способы работы с файлом

Способ работы с файлом	Описание
r	Файл открывается для чтения, указатель текущей позиции помещается в начало файла. Если файла не существует, то возникнет ошибка
r+	Файл открывается для чтения и записи, указатель текущей позиции помещается в начало файла. Если файл не существует, то возникнет ошибка
W	Создается пустой файл и открывается только для записи. Если файл с таким именем существует, то он перезаписывается
W+	Создается пустой файл и открывается для записи и для чтения. Если файл с таким именем существует, то он перезаписывается
a	Файл открывается для записи, указатель текущей позиции помещается в конец файла. Если файла не существует, то он создается
a+	Файл открывается для записи и чтения, указатель текущей позиции помещается в конец файла. Если файла не существует, то он создается

Именно в зависимости от указания способа работы будет зависеть то, что мы сможем делать с открытым файлом. Например, если он открыт для чтения, то при попытке записать в него какую-то информацию, возникнет ошибка. Итак, полная запись второго параметра функции fopen () выглядит следующим образом:

Способ_работы Режим_работы

где Способ_работы — это одно из значений, представленных в табл. 1.2.

Режим_работы — это один из двух приведенных далее вариантов.

- t — текстовый.

Например, как в программе counter.php: a+t.

a+ будет означать, что файл открывается для записи и чтения, указатель помещается в конец файла, если файла не существует, то он создается. А t будет означать, что работа с открытым файлом предполагается как с набором строк, т. е. как с обычным текстом.

- b — бинарный.

Например: a+b.

Здесь опять используем a+, но теперь работа с открытым файлом предполагается как с набором байтов.

В табл. 1.2 используется словосочетание "указатель текущей позиции", это понятие будет рассмотрено позже.

Весь процесс работы с файлом counter.txt, после его открытия, упрощенно можно описать следующим образом:

- Чтение значения счетчика в переменную,
- Увеличение этого значения на 1.
- Запись нового значение в файл.

В строке 7 используется функция блокировки файла flock(). Благодаря этому, пока один пользователь работает с файлом, никто другой, кроме него, не сможет вносить изменения в данный файл. Рассмотрим случай, когда блокировка файла не используется, т. е. когда существует вероятность, что два пользователя считывают значение файла counter.txt примерно в одинаковый промежуток времени. После этого первый пользователь записывает новое значение, увеличенное на 1, второй это делает чуть позже, например, из-за небольших задержек в сети. В результате значение счетчика увеличивается не на 2, как должно быть, а всего лишь на 1, что неправильно. Благодаря блокировке, после того как кто-то начал работу с файлом, никто другой не сможет работать с ним, пока блокировка не будет снята. Другие пользователи в этом случае будут выстроены в некоторую очередь, т. е. они смогут обратиться к файлу только после того, как он будет разблокирован.

Далее представлен синтаксис функции flock ():

flock (дескриптор файла, числовое значение);

где дескриптор файла — это имя переменной $f, которой мы присвоили значение дескриптора, возвращенного функцией fopen (), а числовое значение предназначено для указания варианта блокировки. Возможны следующие варианты (причем можно указывать как числовое значение, так и его альтернативное значение в виде константы):

- 2 или lock_ex — исключительный вариант блокировки, потому что пока один пользователь работает с файлом, никто другой не может к нему обратиться (используется в программе counter.php);
- 3 или lock_un — снятие установленной ранее блокировки.

Есть еще вариант мягкой (разделяемой) блокировки. Он устанавливается указанием 1 или константы LOCK_SH. В этом случае работа с файлом более лояльна — нет жесткого ограничения на то, что с файлом может работать только один пользователь. Не рекомендуется использовать данный вариант, т. к. нельзя дать 100% гарантии, что при работе с файлом не произойдет сбой, к тому же этот вариант не работает в операционной системе Windows.

Теперь еще немного углубимся в процесс работы с файлом и рассмотрим такое понятие, как указатель текущей позиции. Его назначение аналогично курсору при работе в текстовом редакторе — обозначение текущей позиции в открытом файле. Если необходимо что-то записать, то запись данных начнется именно с места, где установлен указатель. Так же, если необходимо прочитать данные из файла, процесс чтения будет

осуществлен, начиная с текущего положения указателя. Таким образом, можно сказать, что вся работа с файлом строится в соответствии с позицией указателя.

В табл. 1.2 вы можете найти описание того, где будет находиться указатель после открытия файла при различных режимах работы с ним.

Изначально, после открытия файла в режиме a+, указатель будет установлен в нулевую позицию, т. е. в начало файла.

В строке 10 мы получаем данные из файла с помощью функции fgets() , она имеет следующий синтаксис:

fgets(файловая переменная);

Данная функция читает очередную строку из файла. Единственный параметр, который необходимо передать в функцию fgets (), — это дескриптор файла. В его качестве мы опять указываем переменную $f, в которой этот дескриптор был сохранен ранее.

В результате выполнения в программе строки 10 функция fgets() возвратит первую и единственную строку файла counter.txt, где как раз и содержится значение счетчика. Эту строку мы сохраняем в переменной $s.

Переходим к строке 12, где происходит увеличение значения переменной $s (т. е. увеличение значения счетчика) на единицу. Для этого действия используется следующая конструкция:

$s=$s+1;

Стоит отметить, что ее можно заменить на альтернативный более короткий вариант:

$s+=$s;

который расшифровывается как "увеличить (знак плюс) на единицу значение переменной, находящейся с правой стороны от знака равно, и присвоить результат переменной $s (находящейся слева от знака операции)".

Теперь необходимо записать новое значение счетчика. Но логично перед этим удалить предыдущее. Как раз для этого используется функция ftruncate() в строке 14 программы, ее синтаксис выглядит следующим образом:

ftruncate(файловый дескриптор, размер);

Данная функция обрезает файл, файловый дескриптор которого передан в качестве первого параметра, до заданного размера, переданного в качестве второго параметра и указанного в байтах. Если процесс завершился успешно, то функция, вернет true, в противном случае — false.

Мы указываем в качестве второго параметра ноль (строка 14), что означает полностью очистить файл — урезать его до нулевого размера.

Теперь ничто не мешает осуществить запись нового значения счетчика в файл counter.txt, что и происходи в строке 16 с помощью функции fputs(). Ее полный синтаксис выглядит следующим образом:

fputs(файловый дескриптор, данные);

Эта функция осуществляет запись данных в файл. В строке 16 она как раз используется для записи значения переменной $s в файл counter.txt.

В строке 19 происходит снятие блокировки с файла counter.txt уже знакомой вам функцией flock (), для этого в качестве ее второго параметра указывается цифра 3. Далее работа с файлом заканчивается, т. к. он закрывается использованием функции

fclose (), после выполнения которой связь программа-дескриптор-файл будет разорвана.

Синтаксис функции fclose ():

fclose(файловый дескриптор);

В строке 23 мы выводим значение счетчика (переменной $s) на экран.

Можно сказать, что весь код программы counter.php практически разобран, конечно, за исключением нескольких моментов: когда происходило открытие файла (строка 3) с помощью функции fopen (), также была использована еще и функция die(), которая имеет следующий синтаксис:

die(текст сообщения об ошибке);

Данная функция выводит на экран текст, переданный в качестве параметра, и осуществляет немедленный выход из программы. Поэтому die() не используется сама по себе, а, как правило, в сочетании с другими (в данном случае с функцией fopen()), как вариант защитного механизма. Таким образом, если функции fopen() не удалось открыть файл, то в этом случае она вернет отрицательный результат (false), и сработает защитный механизм в лице функции die(), которая выведет текст "Не могу открыть файл", после чего выполнение программы будет прекращено. Если бы защитный механизм не использовался, то при дальнейшем выполнении программы она пыталась бы читать и записывать данные в никуда, т. к. файл не был открыт.

Когда две функции используются рядом, строится логическое выражение, которое нужно разделить с помощью специально предназначенного для этого логического оператора. В данном случае в качестве логического оператора используется or (в переводе с англ. "или").

Строка 3 будет обрабатываться препроцессором PHP следующим образом:

Выполнение программы на PHP начинается слева направо (в данном случае выполнение строки 3 не будет исключением).

Сначала будет выполнена функция fopen ().

Если fopen() выполнилась удачно, т. е. произошло открытие файла, и был возвращен файловый дескриптор, то далее будет выполняться строка 7.

Если функции fopen() не удалось открыть файл, то она вернет false в качестве результата, и в этом случае будет выполнена функция die() , после чего выполнение программы прекратится.

Логический оператор or позволяет строить логическую конструкцию (логическое выражение), состоящую из нескольких функций (в данном случае двух), в которой будет соблюдаться условие успешного выполнения одной из этих Функций. Так как выполнение всегда начинается слева направо, то первой в логической конструкции всегда будет исполняться функция, находящаяся слева от or, если ее выполнение успешно, то условие логической конструкции соблюдено и, значит, вторую функцию выполнять не надо. В противном случае, если слева от оператора or функция вернула false, произойдет выполнение функции справа, в результате чего будет опять соблюден принцип логической конструкции, который гласит, что одна из функций обязательно должна быть успешно выполнена. Запустите программу counter.php с помощью браузера, после этого нажмите несколько раз кнопку Обновить, и вы сможете наблюдать, как значение счетчика увеличивается.

Основная работа сделана, осталось только подключить счетчик к разработанной ранее HTML-страничке (см. листинг 2.2). Для этого необходимо изменить файл mypage.html следующим образом (см. строки с 14 по 22) — листинг 2.4.

Листинг 2.4. Файл mypage.php (измененный файл mypage. html)

```
1   <html>
2
3   <head>
4   <title>Домашняя страничка Иванова Пети</title>
5   </head>
6   <body>
7
8   <H1>Домашняя страничка Иванова Пети</H1>
9   <BR>
10  <BR>
11  Привет!
12  Меня зовут Петя.
13  Я изучаю PHP.
14  <BR><BR><BR>
15  <?php
16  //Выводим поясняющую надпись
17  echo "Данную страницу посетило уже:   ";
18  //в середине поясняющей надписи выводим значение счетчика
19  require_once("counter.php");
20  //окончание поясняющей надписи
21  echo " человек";
22  ?>
23  </body>
24  </html>
```

Как видите, появилось несколько новых строчек кода. В строке 14 был три раза добавлен стандартный HTML-тег перевода строки
. В строке 15 объявляется фрагмент кода на PHP, затем в строке 17 выводится текст с помощью echo. А вот в строке 19 используется новая функция языка require_once(), которая предназначена для подключения модуля, имя которого передается в качестве параметра и должно быть заключено в кавычки. В качестве модуля может выступать как программа на PHP, так и обычный HTML-документ. Таким образом, в строке 19 мы подключаем разработанный ранее модуль counter.php.

Подключение модуля аналогично тому, как если бы мы просто скопировали содержимое подключаемого файла и вставили в программу, откуда происходит его вызов.

Вообще понятие модуля достаточно условно, как правило, так называется обычная PHP-программа или ее фрагмент, которая специально была разработана для работы в составе другой программы. То есть модули изначально ориентированны на работу в составе чего-то.

Идем далее. В строке 21 опять происходит вывод текста с помощью echo. Таким образом, значение счетчика будет находиться в следующем тексте:

Данную страницу посетило уже ЗНАЧЕНИЕ СЧЕТЧИКА человек

В строке 22 указано окончание фрагмента кода на PHP. Теперь самое главное — нужно поменять расширение файла mypage с html на php, потому что теперь он содержит полноценную PHP-программу.

Сохраните программы mypage.php, counter.php по адресу Z:\home\myprogram.ru\www\counter. Кликните на ссылке Просмотрите работу Счётчика посещений. Вы сможете увидеть следующий результат — рис. 17.

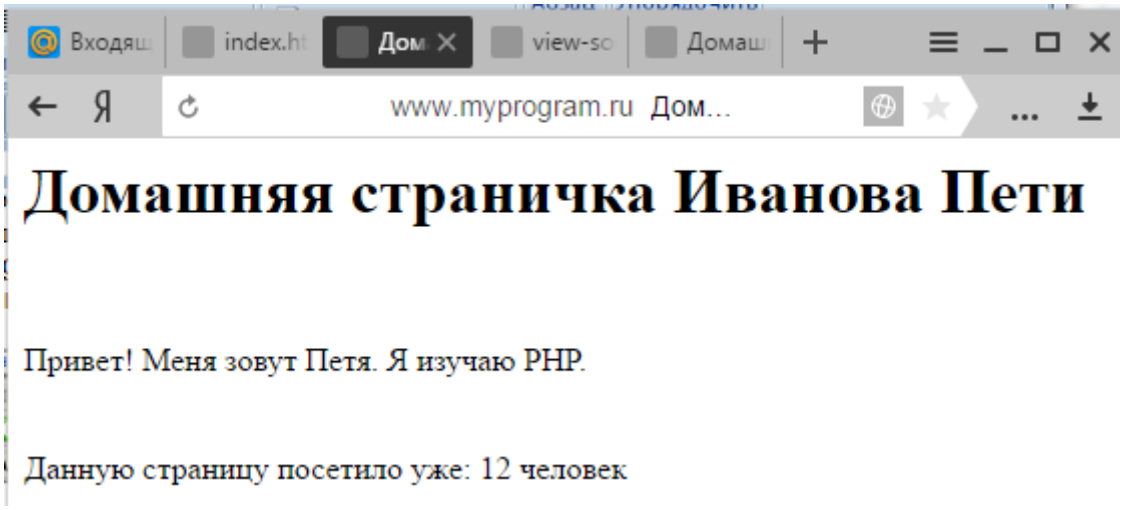


Рис 17. Окончательный вариантWeb-странички с размещенным на ней счетчиком посещений

Закройте сайт http://myprogram.ru/. Откройте его вновь, набрав в адресной строке браузера http://myprogram.ru/ и вновь проверьте работу всех ссылок.

Откройте браузер и наберите в нем http://localhost, после этого нажмите <Enter>. Вы увидите тестовую страницу Денвера (Рис. 7).

Выберите пункт Утилиты, вы увидите утилиты и инструменты, которые включает в себя Денвер. Нажмите ссылку Список зарегистрированных сайтов. Убедитесь, что сайт myprogram.ru присутствует в нём. Вы можете выйти на этот сайт, нажав на ссылку myprogram.ru. Заметим, что прежде, чем отключать компьютер следует остановить Denwer, нажав Stop Denwer.

Файлы, разобранные в п.1.2.4 Примеры серверных скриптов на PHP, 1.2.4.1.Информация о PHP, 1.2.4.2.Просмотр переменных окружения, 1.2.4.3. Вывод реквизитов, 1.2.4.4. Счетчик посещений, соответствующие им скриншоты и пояснения, включите в отчет по лабораторной и практической работе.

Задание

1. Изучите Методические указания или соответствующие разделы пособий по указанной тематике [4,5,6,7,9].
2. Выполните примеры, разобранные в п. 1.2 методических указаний.

3. По аналогии с указанными примерами сформируйте не менее 4-5 серверных скриптов, и используйте их в Вашем сайте (**Лабораторная работа № 1**).

Содержание отчета

1. Задание к работе
2. Листинги, скриншоты и пояснения примеров, разобранных в п. 1.2 методических указаний.
3. Распечатки страниц Web-узла (.htm-тексты и скриншоты на диске CD), в которые внесены изменения в соответствии с пунктом п.3 Задания.
4. URL вашего зарегистрированного Web-узла (http://www….)

Лабораторная работа № 6
Практическое занятие № 6

Динамические страницы и их формирование

Цель работы

Ознакомление с принципами создания динамических страниц. Формирование навыков создания и использования динамических страниц.

Методические указания

1. Динамические страницы и их формирование

1.1. Прием данных от Web-формы (Delphi) (Факультатив)

В лабораторной работе№7 (П. 1.1.) мы разобрали, как создаются и запускаются Web-модули, имеющие различные параметры. Однако чтобы организовать более привычный пользовательский интерфейс со стандартными элементами управления, надо вызов этих модулей спрятать «внутрь» страницы HTML.

Подготовим страницу HTML, на которой разместим следующий код.

```
<HTML>
<HEAD>
<TITLE>Тестовый пример</TITLE >
</HEAD>
<BODY>
```

```
<B>Введите Ваше имя и электронный адрес для подписки на наш список рассылки:</B>
<P><CENTER><form action="http: //TestBed/cgi-bin/getmail.exe" method="GET">
<table>
<tr><td>Имя:</td><td> <input type="text" size="20" name="Name"></td></tr>
<tr><td>Адрес почты:</td><td> <input type="text" size="20" name="Email"></td></tr>
<tr><td>  </td><td><input type="submit" name="Button" value="Отправить">
    </td></tr>
</table>
</form>
</CENTER>
</BODY>
</HTML>
```

По щелчку на кнопке Отправить выполнится Web-модуль, указанный как значение атрибута ACTION в теге FORM.

http://TestBed/cgi-bin/getmail.exe

Информацию от формы он получит, как обычно в виде параметров.
Если сейчас загрузить эту страничку в браузер, то она будет выглядеть так, как показано на рис. 1 (кроме заполненных полей).

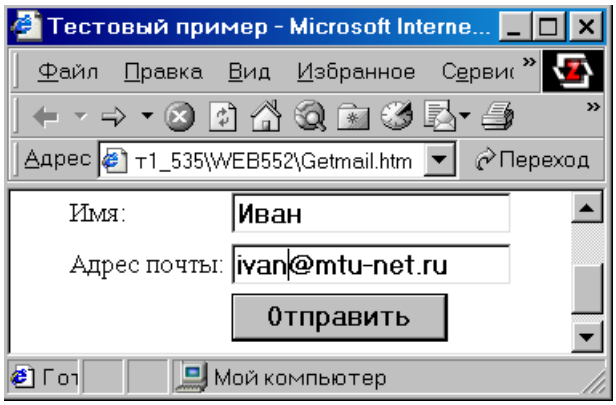


Рис. 1. Страница HTML, содержащая форму, при просмотре в браузере

С ее помощью, например, можно организовать регистрацию подписчиков на список рассылки электронных писем.
По щелчку на кнопке Отправить на сервере вызывается модуль getmail.exe, которому в качестве параметров передаются значения, введенные в два поля. Эти значения надо занести в базу данных по подписчикам, а на Web-сервере организовать специальную службу ведения этой базы и выполнения автоматической рассылки. Существует немало свободно распространяемых программ и бесплатных служб, позволяющих организовать такую работу без программирования. Поэтому в данном примере рассмотрим только технологию получения и обработки информации, получаемой через форму, которая имеется на странице HTML. Создадим Web-модуль и сформируем в нем един-

ственный обработчик события, у которого свойство PathInfo будет пустым. Этот обработчик запишется, например, так.

```
procedure TwebModule1.WebModule1WebActionItemlAction(
   Sender: TObject; Request: TWebRequest;
   Response: TWebResponse; var Handled: Boolean);
   begin
   with Request.QueryFields do
      Response.Content := '<html><body> Спасибо, ' + Values [Names [0] ] +
      ', Ваш адрес ' + Values [Names [1] ] + ' включен в список рассылки. </
      body></html>'
   end;
```

Содержимое возвращаемой страницы (Response.Content) представляет собой строку, состоящую из комментария и введенных пользователем значений. Теперь необходимо выполнить разбор не значения Request.Query, а значения Request.QueryFields, которое содержит длинную строку, полученную от формы.

```
Name=%C8%E2%E0%ED&Email=ivan@mtu-net.ru&
Button=0%F2%EF%FO%EO
```

Параметры отделены друг от друга символом &, названия элементов и соответствующие им значения — символом =. Если в качестве значения выступает символ, не относящийся к латинскому алфавиту, цифрам или знакам препинания (например, русская буква), то он кодируется двумя шестнадцатеричными цифрами, перед которыми ставится символ %. Так, в данном примере строка Иван, введенная в поле имени, представляется следующим образом.

```
Name=%C8%E2%E0%ED
```

Разбирать такую строку программным способом неудобно, поэтому вместо свойства Query лучше использовать свойство QueryFields (тип TStrings). Оно представляет собой массив строк, выделенных из исходной строки по символам &. То есть, в нашем случае в этом свойстве будут три строки.

```
Name=%C8%E2%E0%ED
Email=ivan@mtu-net.ru
Button=0%F2%EF%FO%EO
```

Однако и в таком виде требуется программно определять позицию символа = и выделять остаток строки, что относится к чисто рутинной работе. Как рассказывалось при описании класса TStrings, он позволяет автоматически анализировать и разбивать на пары строки, выполненные по схеме

название=значение

Свойство Values класса TStrings позволяет получить значение (заключительную часть) строки, начинающейся с подстроки, указанной в свойстве Value в качестве параметра, что и выполняет вышеприведенный код, выделяя значения первой и второй строк.

Если теперь поместить данный скомпилированный модуль (назвав его getmail.exe) в каталог /cgi-bin Web-сервера, то при вводе в поля значений Иван и ivan@mtu-net.ru и щелчке на кнопке Отправить на экране браузера появится изображение, показанное на рис. 2.

В нашем примере никакой регистрации, конечно, не произошло, но, зная, как работать с базами данных, можно развить этот пример. Вопросы создания Web-модулей, работающих с базами данных рассмотрены в [4,5].

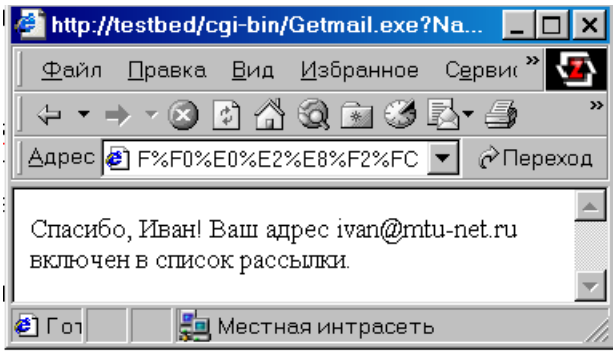


Рис. 2. Результат обработки Web-модулем данных,
полученных в результате отправки формы

1.2. Примеры создания динамических страниц на PHP

Используя результаты лабораторной работы № 5 и практического занятия № 5, рассмотрим примеры создания нескольких динамических страниц на PHP, а именно:

1. Динамическая страница приветствия.
2. Форма обратной связи.
3. Динамическая страница приветствия (защищённый вариант).
4. Отправка SMS

При этом будут использованы примеры, рассмотренные в [6,7].

Создайте файл index.html следующего содержания:

```
<html>
<body>

<table width=100% height=100%>
<tr>
<td align=left>
<h2 align=center>Формирование динамических страниц на <b>
PHP</b></h2>
<p>1.Динамическая страница приветствия.
<a href="privet\form.html">Войдите в систему</a>
<p>2.Форма обратной связи <a href="backform\backform.php">
Просмотрите Форму обратной связи</a>
<p>3.Динамическая страница приветствия (защищённый вариант).
<a href="privet\form1.html">Войдите в систему</a>
<p>4.Отправка SMS <a href= "sms\sms.php">Отправьте SMS</a>
</td>
</tr>
</table>

</body>
```

```
</html>
```

Сохраните файл index.html на вашем виртуальном сайте по адресу Z:\home\myprogram1.ru\www.

Результат просмотра файла index.html в браузере, путём обращения к сайту http://myprogram1.ru, представлен на рис. 3.

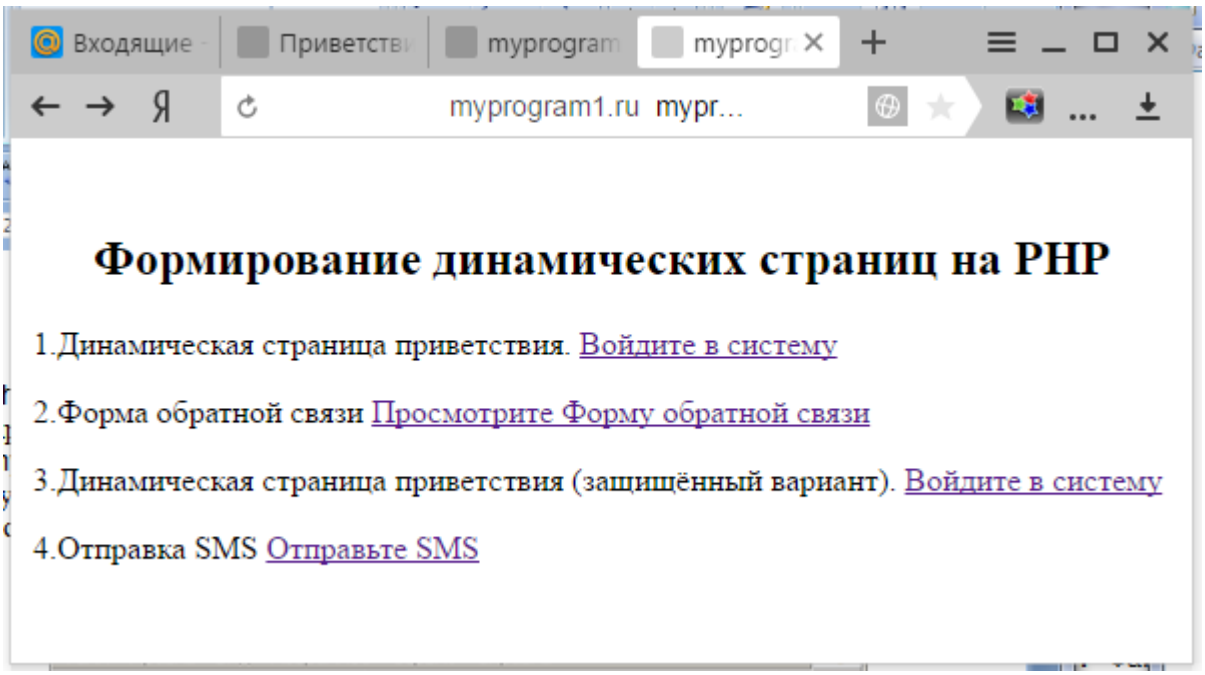


Рис. 3. Результат просмотра файла index.html в браузере

Как вы, уже догадались, эта главная страница вашего сайта предназначена для загрузки серверных скриптов, обеспечивающих формирование соответствующих динамических страниц.

1.2.1. Динамическая страница приветствия

Давайте напишем небольшую программу, предназначенную для работы с формами. Она будет состоять из двух файлов. Первый — это обычный HTML- документ, который будет предназначен для отображения формы. Второй — это программа на PHP, которая будет вызываться после того, как пользователь нажал на форме кнопку ВОЙТИ В СИСТЕМУ. Ее предназначение заключается в выводе надписи "ПРИВЕТ!".

Создайте файл с именем form.html (листинг 1) и сохраните его по адресу:

Z:\home\myprogram1.ru\www\privet.

Листинг 1. Файл form.html

```
<html>
<head>
<title>Приветствие</title>
</head>
<body>

<! -- Объявляем форму с обработчиком program for_form.php -- >
<FORM
name="my form" action="program_for_form.php">
```

```
Представьтесь :
<! -- Объявляем поле для ввода имени -- >
<input
type="text"
name="name_of_user"
value=""
title="Поле для ввода имени">

<! -- Переходим дважды на следующую строку -- >
<BR>
<BR>

<! -- Объявляем кнопку -- >
<input
type="submit" value="ВОЙТИ В СИСТЕМУ"
name="enter_system"
title="Нажмите,   если вы ввели имя,   чтобы увидеть результат">

</FORM>

</body>
</html>
```

Теперь необходимо создать второй файл с именем program_for_form.php (листинг 2) и сохранить его по адресу:

Z:\home\myprogram1.ru\www\privet.

Листинг 2. Файл program_for_form.php

```
<html>
<head>
<title>Обработчик program_for_form.php</title>
</head>
<body>

<?php
echo  " ПРИВЕТ!";
?>

</body>
</html>
```

Чтобы протестировать разработанную программу, откройте файл form.html через браузер, введите имя и нажмите кнопку ВОЙТИ В СИСТЕМУ.

Немного модифицируем пример — сделаем так, чтобы после слова "ПРИВЕТ! выводилось имя, которое ввел пользователь. Для того чтобы получить данные формы, в PHP-программе можно использовать 1 из 3 массивов:

- $_get — применяется, когда данные формы передаются методом get;
- $_post — используется, когда данные формы передаются методом post;

- $_request — применяется как альтернативный вариант двум предыдущим, то есть заменяет их, не важно каким методом передавались данные, они все равно будут доступны в этом массиве.

Имя элемента массива, в котором хранятся данные об имени пользователя, такое же, как имя элемента формы, в которое было введено это самое значение. В силу того, что у поля ввода в предыдущем примере было имя name_of_user, при обращении к

$_REQUEST['name_of_user'] или

$_GET [' name_of_user']

(если метод не указан, то данные отправляются методом get, а метод указан не был) можно получить значение, которое ввел пользователь. Чтобы добавить новую возможность в программу, необходимо заменить строку

```
echo  " ПРИВЕТ!"
```

на следующий код:

```
echo  "ПРИВЕТ   ".$_GET['name_of_user']. "!";
```

Результаты выполнения измененной программы представлены на рис. 3, 4.

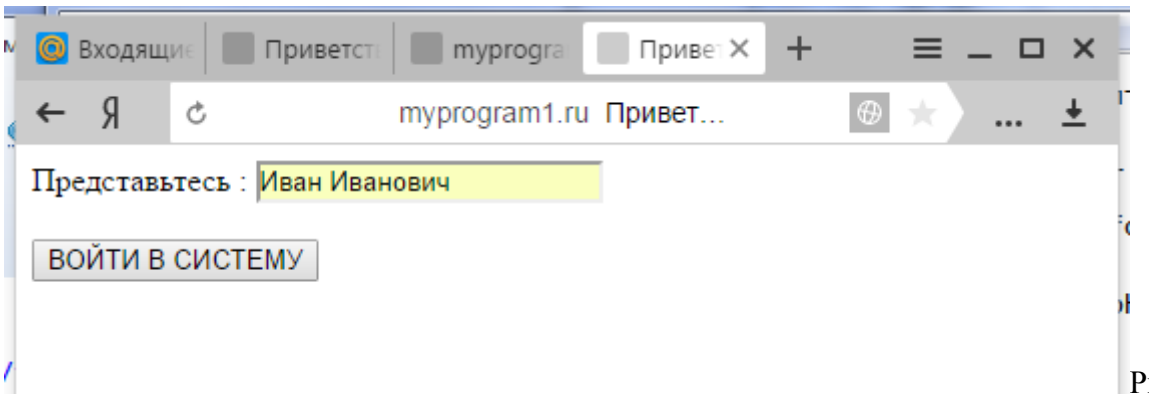


Рис. 3. Результат выполнения измененной программы до нажатия на кнопку

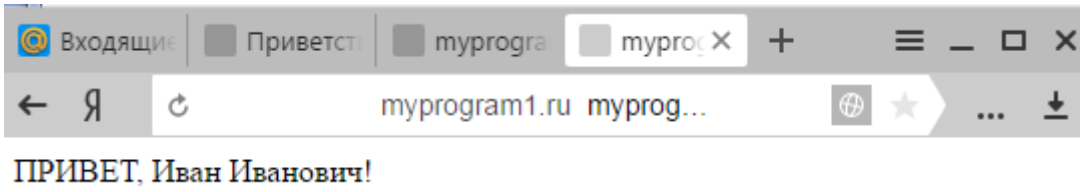


Рис. 4. Результат выполнения программы после нажатия на кнопку

1.2.1.1. Методы отправки данных формы

Как было сказано ранее, форма может быть отправлена на сервер двумя способами (методами): get и post. Рассмотрим их особенности и отличия.

1.2.1.1.1. Метод get

Метод get означает, что данные будут передаваться присоединенными к имени программы обработчика, указанного в параметре action формы, то есть:

скрипт, указанный в action + данные формы

Чтобы сервер смог их отличить, они присоединяются по специальным правилам.

Сначала после имени программы-обработчика ставится знак вопроса (?), таким образом, указывается, что будут передаваться данные. Например, если в параметре action формы указано program_for_form.php, то результат будет следующим: program_for_form.php?

Потом указывается имя первого объявленного на форме элемента, например, если первым указано поле для ввода с именем for_name, то результат будет следующим:

program_for_form.php?for_name

Далее идет знак равно, который разделяет имя элемента и его значение:

program_for_form.php?for_name=

После знака равно идет само значение элемента, например:

program_for_form.php?for_name=Vasya

Если элемент на форме один, то формирование запроса считается оконченным, и он отправляется на сервер.

Если же на форме несколько элементов, то передача следующего значения отделяется от первого знаком амперсанда (&), а далее все происходит по вышеописанной схеме:

имя элемента, знак равно, значение элемента.

Предположим, что после поля ввода с именем for_name располагается такое же поле ввода, только с именем password, тогда строка, сформированная методом get для отправки на сервер, будет следующей:

program_for_form.php?for_name=Vasya&password=123

Таким же образом с помощью знака & отделяются все последующие элементы — третий, четвертый и т. д., если конечно они есть.

Метод get удобен тем, что вы можете видеть сформированную браузером строку запроса для передачи на сервер и соответственно напрямую передавать обработчику формы данные, минуя саму форму, для этого достаточно сформировать вручную в браузере строку по описанному ранее правилу. Следует отметить, что русские буквы передаются в закодированном виде, так что вы не увидите for_name=Вася в строке запроса при передаче данных методом get (для кодирования используется шестнадцатеричное представление каждой русской буквы). Например, строка запроса для формы, состоящей из одного поля с именем for_name, содержащем значение Вася, будет следующей:

program_for_form.php?for_name=%C2%EO%FI%FF

Именно метод get используется, когда вы ищете что-то с помощью поисковых систем. После того как в строку поиска введено какое-то слово или фраза (а это не что иное, как поле ввода, размещенное на форме), и нажата кнопка ПОИСК (после этого запрос будет отправлен на сервер), вы можете наблюдать, какой запрос формируется в строке браузера.

1.2.1.1.2. Метод post

При использовании метода post данные отправляются незаметно для пользователя в теле запроса. Сам запрос при этом будет состоять из трех частей:

- запрос к программе-обработчику, указанной в параметре action формы;
- сообщение о том, какой объем данных будет передаваться;
- передача самих данных.

Давайте рассмотрим интересный пример объявления формы:

```
<FORM      action="program_for_form.php?razdel= 1&punkt=10"
 method=POST>
<input  type="text" name="for_name"> <input type="submit"  value="ВОЙТИ">
</FORM>
```

Как видите, несмотря на то, что объявлена post -форма (это краткое название формы, данные которой отправляются методом post), в параметре action вместе с программой-обработчиком указывается дополнительная информация, сформированная по всем правилам метода get. Это не ошибка, а очень удобная возможность, которая позволяет использовать сразу два способа отправки данных на сервер. На самом деле, весь секрет достаточно прост. Когда рассматривались составные части post -запроса на сервер, под первым пунктом было выделено "обращение к скрипту-обработчику", а это не что иное, как обычный GET-запрос. Таким образом, post-форму можно использовать для передачи данных методом get, но не наоборот.

1.2.1.1.3. Сравнение методов post и get [6]

Метод get рекомендуется использовать, когда происходит запрос некоторой информации (например, в поисковом сервере) или вы создаете что-то, содержащее иерархическую структуру, скажем, форум или сайт с множеством подразделов (например, там может быть раздел Скачать, подразделы Программы, Документация, которые в свою очередь могут содержать несколько подразделов). Он предпочтительнее, потому что тогда можно поставить закладку средствами браузера на нужную информацию, либо отправить ссылку, сформированную в строке запроса браузера, или коллеге по работе.

Недостатком метода get является ограничение на объем передаваемых данных. У post это ограничение тоже имеется, но оно слабее — так что с помощью post -метода можно передавать даже целые файлы.

Метод post лучше использовать, когда не следует отображать пересылаемые данные, например, при авторизации на форуме, когда вы вводите свой логин и пароль (хотя на самом деле и в этом случае можно просмотреть, что именно отправляется на сервер).

1.2.2. Форма обратной связи

Форма обратной связи представляет простейший механизм общения посетителя сайта с его администратором. Создайте файл backform.html (листинг 3).

Листинг 3. Файл backform.html

```
<FORM action="" method="post">
Ваше имя:<br>
<input type="text" name="person">
<br>
Ваш e-mail:<br>
```

```
<input type="text" name="email">
<br>
Вопрос администратору:<br>
<textarea name="question" cols=40 rows=5></textarea>
<br>
<input type="submit" name="okbutton" value="OK">
</FORM>
```

Как видите, форма состоит из четырех элементов (рис. 5):

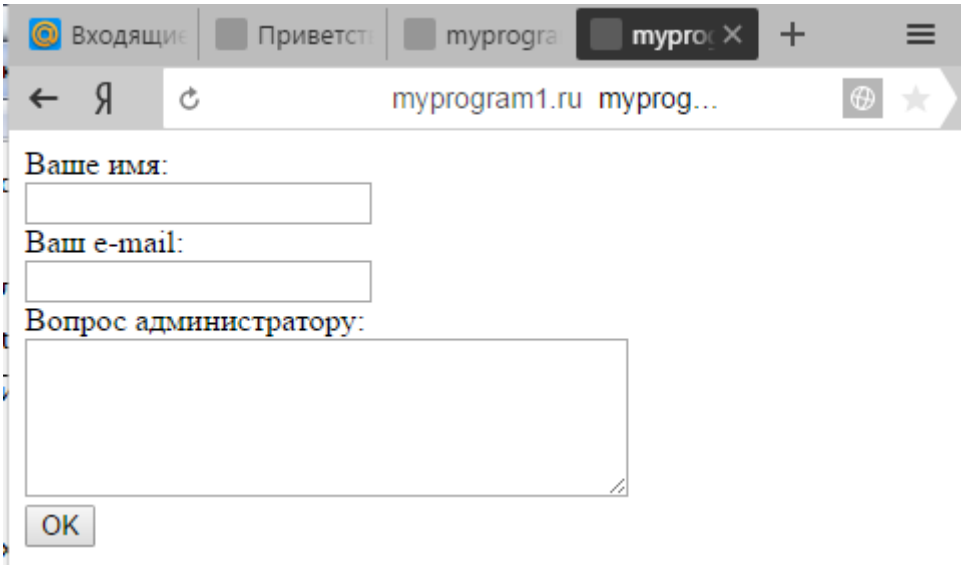


Рис. 5. Форма обратной связи

- поля для ввода имени пользователя;
- поля для ввода e-mail пользователя;
- поля для ввода многострочного текста — сюда пользователь будет писать свой вопрос к администратору сайта;
- кнопки, после нажатия которой форма будет отправлена на сервер.

Теперь создайте скрипт obrabotka.php (листинг 4)., который будет обрабатывать разработанную ранее форму.

Листинг 4. Файл obrabotka.php

```
1   <?php
2   //Проверяем, был ли скрипт вызван как обработчик формы
3   if (isset($_POST['okbutton']))
4   {
5   //Куда будем отправлять письмо
6   $komu="LittleBuddan@vr-online.ru";
7   //Тема письма
8   $tema="Вопрос от ".$_POST['person']." ".$_POST['email'];
9   //Само письмо
10  $text_pisma=$_POST['question'];
11
12  //Отправляем письмо
13  mail($komu,$tema,$text_pisma);
```

```
14
15    echo "Уважаемый ".$_POST['person']."! Ваш вопрос был отправлен
16    администратору";
17    echo "<br><a href=backform.php>Назад</a>";
18    }
19    ?>
```

В строке 3 используется новая конструкция, она называется условной и предназначена для проверки некоторого условия (логического выражения), при выполнении которого реализуется код, заключенный в фигурные скобки. Синтаксис условной конструкции:

```
If (условие)
{код, который будет выполнен при реализации условия  }
```

Следует отметить, что условие обязательно должно быть заключено в круглые скобки. В программе obrabotka.php было использовано следующее условие:

```
(isset ($_post ['okbutton' ]))
```

Таким образом, проверяется наличие элемента с именем okbutton в массиве $_post с помощью уже знакомой функции isset() — если он существует, то условие будет соблюдено (равно true), в противном случае условие выполнено не будет (равно false).

Зачем нужна такая проверка? Просто ничто не мешает кому-то запустить скрипт obrabotka.php напрямую,— не важно, по ошибке или по злому умыслу. Тогда программа будет работать неправильно, т. к. у нее не будет необходимых данных (ни имени пользователя, ни его e-mail, ни текста письма). Поэтому добавление условия позволяет добиться результата, при котором скрипт будет выполнен только в случае, когда пользователь заполнил форму и нажал кнопку ОК — именно тогда в массиве $_post будет создан элемент okbutton.

В строке 13 происходит отправка электронного письма с данными, которые ввел пользователь. Для этого используется стандартная функция mail(), ее синтаксис выглядит следующим образом:

```
mail(адрес получателя, тема, сообщение);
```

Если отправка письма успешна, то функция вернет true, в противном случае — false.

Описание ее параметров:

- адрес получателя — куда будет отправлено письмо, в программе мы задаем его в переменной $komu;
- тема — тема письма, мы формируем его в переменной $tema как объединение Фразы "Вопрос от", имени пользователя и его электронного адреса;
- сообщение — тело письма, мы формируем его в переменной $text_pisma равным значению элемента question формы.

Запустите HTML-документ backform.php, в котором хранится форма обратной связи, введите данные и нажмите кнопку ОК. Вы увидите результаты, изображенные на рис. 6, 7.

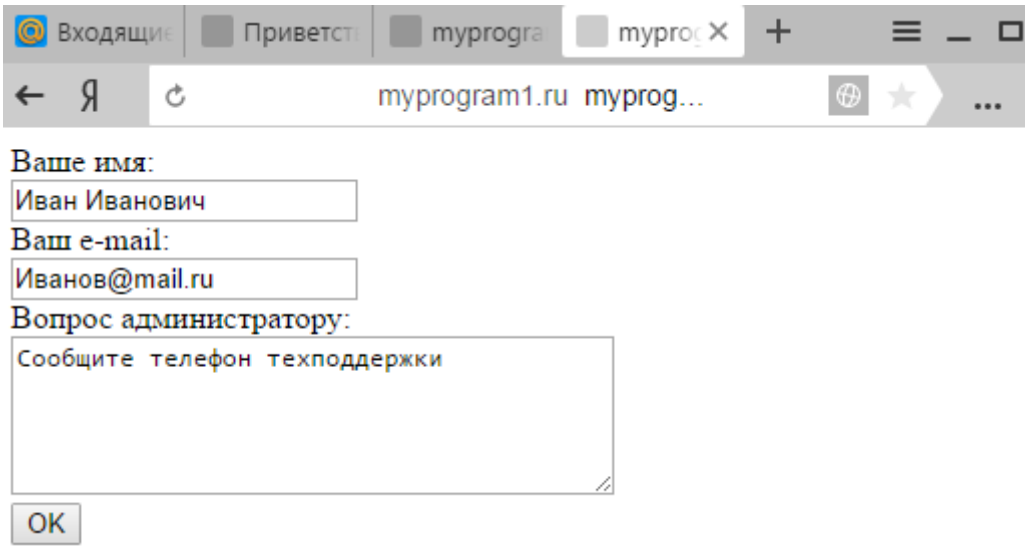


Рис. 6. Результат заполнения и отправки на сервер формы обратной связи до нажатия кнопки

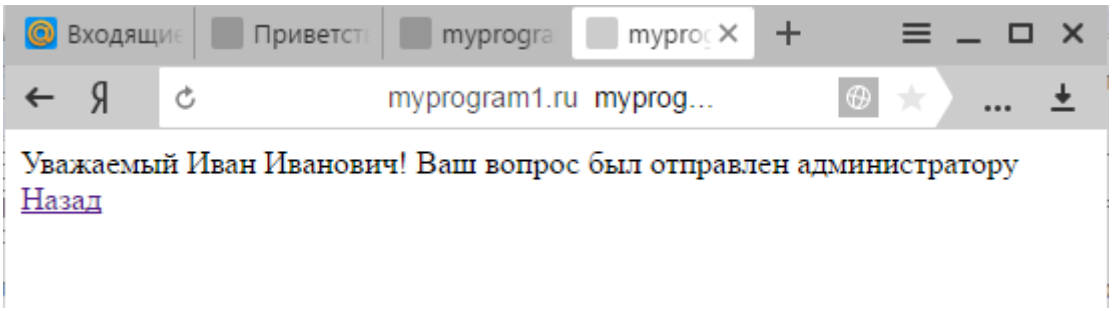


Рис. 7. Результат заполнения и отправки на сервер формы обратной связи после нажатия кнопки

Когда разрабатываются небольшие по объему и несложные в реализации скрипты, то часто используют подход, при котором форма и ее обработчик хранятся в одном файле. В листинге 5 представлен код программы, реализующей тот же самый механизм обратной связи, рассмотренный ранее, только теперь он реализован с помощью одного файла, сохраните его под именем backform.php по адресу

Z:\home\myprogram1.ru\www\backform

Листинг 5 Файл backform.php

```
<?php
//Если была нажата кнопка OK,
//то отправляем письмо и информируем об этом пользователя
if (isset($_POST['okbutton']))
{
if ($_POST['person']== '' or $_POST['email']== '' or
$_POST['question']== '')
{
echo "<font color='red'>Вы ввели не все данные</font>";
echo "<br><a href=backform.php>Назад</a>";
```

```
10    exit;
11    }
12
13    //Куда будем отправлять письмо
14    $komu="LittleBuddan@vr-online.ru";
15    //Тема письма
16    $tema="Вопрос от ".$_POST['person']." ".$_POST['email'];
17    //Само письмо
18    $text_pisma=$_POST['question'];
19
20    //Отправляем письмо
21    mail($komu,$tema,$text_pisma);
22
23    echo "Уважаемый ".$_POST['person']."! Ваш вопрос был отправлен
      администратору";
24    echo "<br><a href=backform.php>Назад</a>";
25    //Выполнять больше нечего, выходим из программы
26    exit;
27    }
28    ?>
29
30    <FORM action="" method="post">
31    Ваше имя:<br>
32    <input type="text" name="person">
33    <br>
34    Ваш e-mail:<br>
35    <input type="text" name="email">
36    <br>
37    Вопрос администратору:<br>
38    <textarea name="question" cols=40 rows=5></textarea>
39    <br>
40    <input type="submit" name="okbutton" value="OK">
41    </FORM>
```

Теперь первая часть файла отвечает за обработку формы, а вторая часть — за ее вывод. В строке 4 осуществляется проверка, был ли скрипт вызван как обработчик post-формы. Если нет (значит, условие (isset($_post ['okbutton'])) ложно), то будет выполнен фрагмент кода, начинающийся со строки 30, который отвечает за вывод пользователю формы обратной связи. Иначе, следующей будет выполнена строка 6, где проверяется еще одно условие — все ли поля формы были заполнены пользователем, перед тем как он нажал OK.

Одно из главных правил при работе с формами — это проверка данных, которые ввел пользователь, и в первую очередь факта, ввел ли он их вообще. Делается это простой проверкой значений тех элементов формы, с данными которых будет осуществляться работа в программе. Реализуется это очень легко, т. к. независимо от того, какой метод передачи используется (post или get), для каждого элемента формы, при создании которого был указан параметр name, на сервер придет строка в виде имя=значение. Все что нам остается после этого — проверить, не содержит ли нуж-

ный нам элемент пустое значение, что мы и делаем с помощью следующего логического выражения:

```
($_POST['person']=='' or $_POST['email1]=='' or $_POST['question']=='')
```

То есть условие в строке 6 будет верно, если поле person или поле email или поле для ввода многострочного текста question имеют пустое значение. В этом случае высветится предупреждение "Вы ввели не все данные" и работа программы завершится (рис. 8).

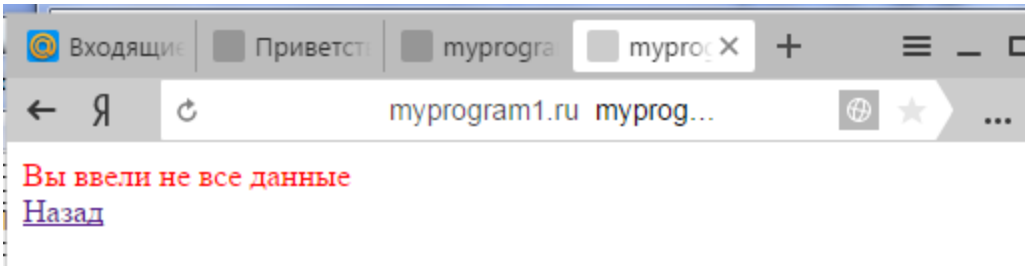


Рис. 8. Предупреждение, возникающее вследствие наличия на форме незаполненных данными элементов

Таким образом, в программу встроена защита от отправки пустых сообщений, а заодно создан более дружественный интерфейс, который выводит для пользователя поясняющее сообщение.

Особого внимания заслуживает строка 8:

```
echo "<font color='red'>Вы ввели не все данные</font>";
```

Обратите внимание, в ней используется два вида кавычек. Это очень важный момент, т. к. если его не соблюсти, программа работать не будет. Первые двойные кавычки необходимы для строки, которая выводится с помощью команды echo, вторые — одинарные, для HTML-тега, находящегося внутри строки. Рассмотрим случай, при котором использовался только один вариант кавычек — двойных, при этом строка 8 приобрела бы следующий вид:

```
echo "<font color=”red”'>Вы ввели не все данные</font>";
```

и была бы обработана препроцессором гипертекста по ошибочной схеме.

То есть выполнение программы backform.php закончилось бы ошибкой из-за Незнакомой для PHP команды

```
red”'>Вы ввели не все данные</font>"
```

В Денвере в качестве функции mail() используется отладочная заглушка, которая не отправляет письма, а складывает их в директорию

```
C:\WebServers\tmp\!sendmail.
```

Вы можете открыть любое такое письмо с помощью двойного щелчка мыши, конечно, если у вас установлен почтовый клиент, например, Outlook (рис. 9) , The Bat или другой.

Данное письмо можно также просмотреть в обычном Блокноте .

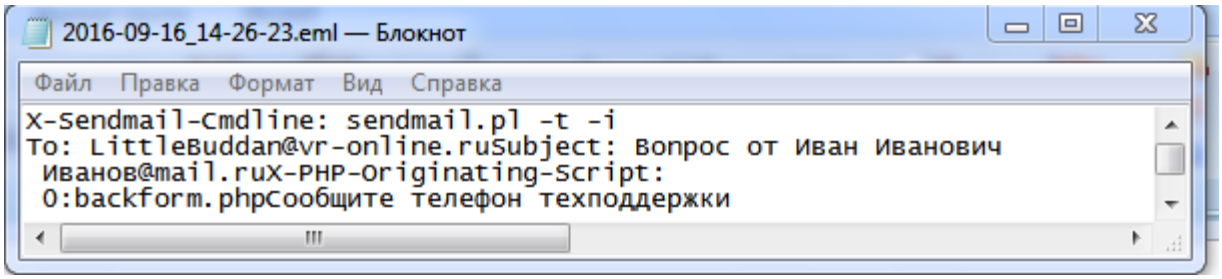


Рис. 9. Электронное письмо, сформированное функцией mail() и открытое с помощью Блокнота

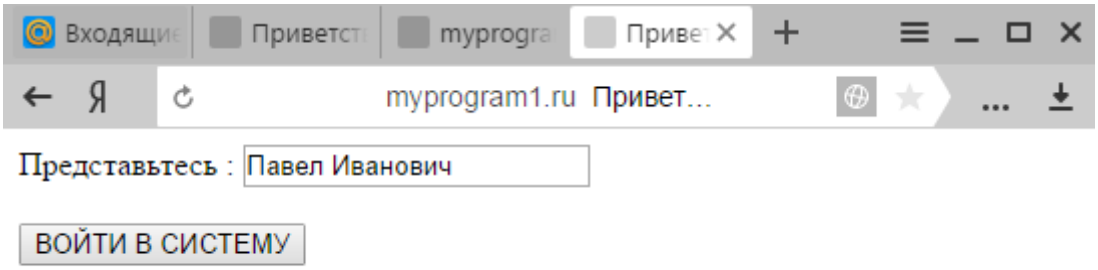
1.2.3. Динамическая страница приветствия (защищённый вариант)

По аналогии с примером 1.2.2. (Форма обратной связи) создайте динамическую страницу приветствия (пример 1.2.1.), в которой скрипт **program_for_form1.php** будет выполнен только в случае, когда пользователь заполнит форму и нажмёт кнопку **ВОЙТИ В СИСТЕМУ** — именно тогда в массиве **$_post** будет создан элемент **enter_system.** Такая проверка исключает запуск скрипта **program_for_form1.php** напрямую,— не важно, по ошибке или по злому умыслу.

Ясно, что при объявлении формы с обработчиком **program for_form1.**php следует использовать **method="post",** а в сам обработчик включить фразу

```
if (isset($_POST["enter_system"]))
{echo  "ПРИВЕТ,  ".$_POST['name_of_user']. "!";
```

Результаты выполнения измененной программы представлены на рис. 10, 11



. Рис. 10. Результат выполнения измененной программы до нажатия на кнопку

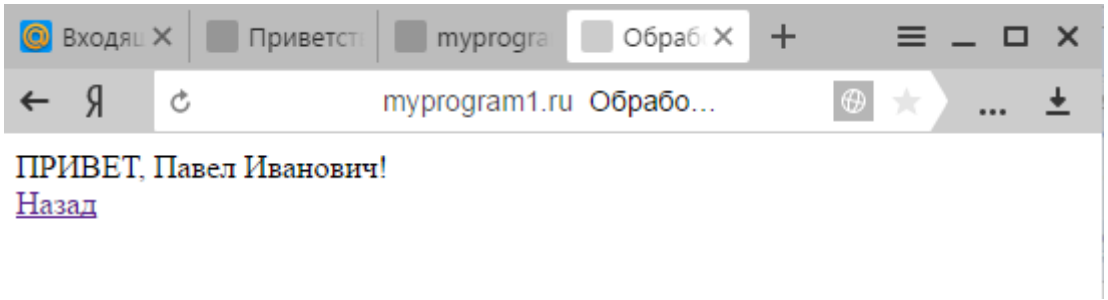


Рис. 11. Результат выполнения измененной программы после нажатия на кнопку

1.2.4. Отправка SMS

Рассмотрим механизм организации на сайте такого интересного сервиса, как отправка бесплатных SMS [7]. Один из способов отправки сообщений через Интернет осуществляется с помощью отсылки письма на e-mail следующего вида:

<номер>@<адрес сайта оператора>

Эту услугу предоставляют не все операторы. Более детально эту информацию можно узнать на сайте конкретного оператора. Например, при отправке SMS на номер оператора Киевстар указывается не адрес сайта kyivstar.net, а sms.kyivstar.net. Например, дадим пользователю возможность отправлять сообщения на номера операторов: Киевстар, UMC и Джинс. На первом этапе разработаем форму, через которую пользователь будет вводить необходимые данные: имя оператора и номер телефона для дозвона на сервер оператора, а также собственно текст SMS-сообщения.

Создайте сценарий sms.php (Файл sms.php сохраните по адресу Z:\home\myprogram1.ru\www\sms), после чего введите в него следующий HTML-код (Листинг 6):

Листинг 6 Файл sms.php

```
<html>
<head>
<title>Отправка SMS</title>
</head>
<body>
<form method=POST action=sms.php>
 <table border=0 align=center>
  <tr>
   <td>
    Оператор
   </td>
   <td>
    <select name='operator'>
     <option value='KSTAR'>Киевстар</option>
     <option value='JEANS'>Джинс</option>
     <option value='UMC'>UMC</option>
    </select>
   </td>
  </tr>
  <tr>
  <td>
  Номер
  </td>
  <td>
  <input type=text name=nomer maxlength=7>
  </td>
   </tr>
   <tr>
   <td>
  Сообщение
   </td>
    <td>
     <textarea name=sms cols=40 rows=6></textarea>
     </td>
     </tr>
  <tr>
```

```
      <td>
       .
      </td>
      <td>
        <center> <input type=reset value='Очистить форму'>
        <input type=hidden name=action value=true>
         <input type=submit value='Отправить SMS'>   </center>
      </td>
    </tr>
  </table>
</form>
<p> <font color="red">
<b>
```

Форма находится в таблице. Внешний вид уже заполненной формы представлен на рис. 12.Рис. 12.. Форма для ввода адреса и текста SMS-сообщения

Далее напишем сценарий, который будет обрабатывать полученные данные. Чтобы он ложно не срабатывал, введена невидимая переменная $action, которая будет определять, когда пользователь просто загрузил страницу, а когда он отправляет данные формы.

Листинг 6. Файл sms.php (продолжение).Вначале сценария проводятся всевозможные проверки, и лишь после того как проверяется, что все данные введены корректно, сценарий обрабатывает их и отправляет письмо указанному оператору. Заметьте, что в данном примере сама отправка письма занимает лишь одну строчку:

```
mail($email,"",$sms);
```

Второй параметр оставлен пустым, хотя там обычно указывается тема письма. При отправке SMS тема игнорируется, так что нет смысла ее указывать.

Если данные введены корректно, то форма отправки сообщений принимает вид рис. 13. Если данные введены некорректно, то форма отправки сообщений принимает вид рис. 14 (Неверно введен номер), рис. 15 (Отсутствует текст сообщения).

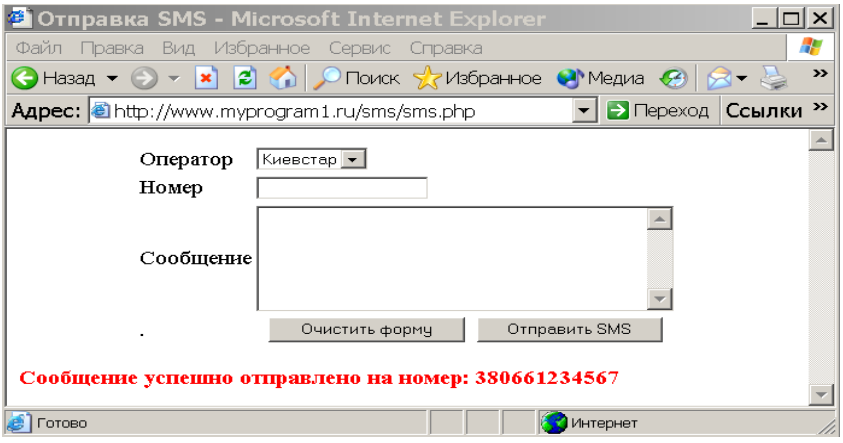


Рис. 13. Форма отправки сообщений при корректном вводе данных

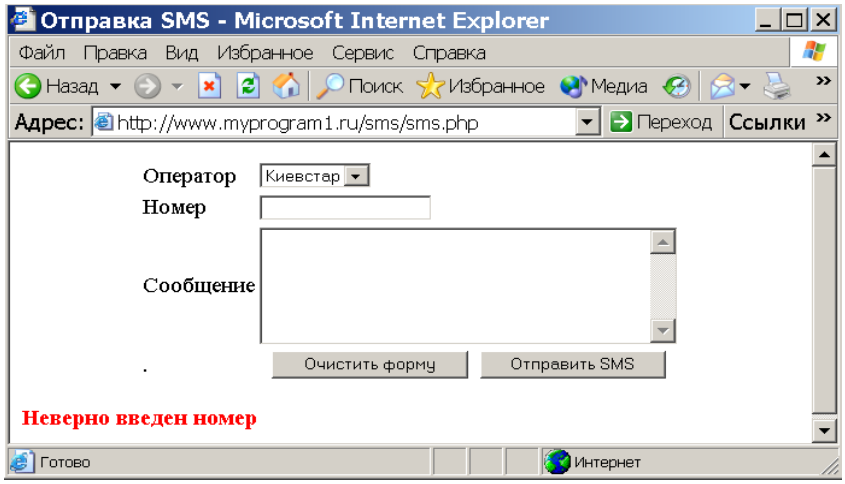


Рис. 14 Форма отправки сообщений при неверном вводе номера

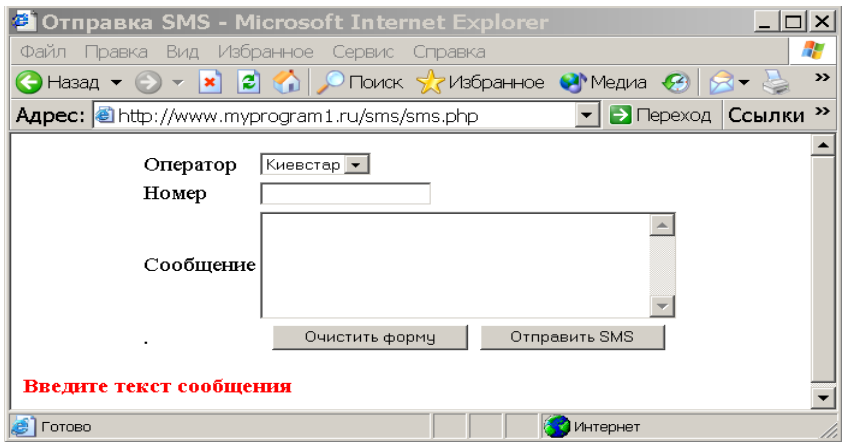


Рис. 15 Форма отправки при отсутствии сообщения

Файлы, разобранные в п.1.2, Примеры создания динамических страниц на PHP, 1.2.1. Динамическая страница приветствия, 1.2.2. Форма обратной связи, 1.2.3. Динамическая страница приветствия (защищённый вариант), 1.2.4. Отправка SMS, соответствующие им скриншоты и пояснения, включите в отчет по лабораторной и практической работе

Задание

1. Изучите Методические указания или соответствующие разделы пособий по указанной тематике [4,5,6,7,9].
2. Выполните примеры, разобранные в 1.2 методических указаний
3. По аналогии с указанными примерами сформируйте не менее 3-4 серверных скриптов, и используйте их для генерации ди-

намических страниц в вашем сайте (**Лабораторная работа № 4**). Рекомендуется задействовать при этом базу данных (См., например, [6,7,8,10]).

Содержание отчета

1. Задание к работе
2. Листинги, скриншоты и пояснения примеров, разобранных в п. 1.2 методических указаний.
3. Распечатки страниц Web-узла (.htm-тексты и скриншоты на CD), в которые внесены изменения в соответствии с пунктом п.3 Задания.
4. URL вашего зарегистрированного Web-узла (http://www….)

Библиографический список

1. Холкин И. И. Интернет- технологии и системы. Методические указания по выполнению лабораторных работ. № 0448, Москва, МИРЭА, 2005 , — 24 с.
2. Петюшкин А. В. HTML в Web-дизайне. — СПб.: БХВ-Петербург, 2005. — 400 с
3. Николенко Д. В. Практические занятия no JavaScript — СПб: Издательство "НАУКА И ТЕХНИКА", 2000. 128 с
4. Бобровский С. Delphi 5: учебный курс - СПб: Издательство «Питер», 2000. — 640 с.
5. Подольский С. В., Скиба С. А., Кожедуб О. А. Разработка интернет-приложений в Delphi. — СПб.: БХВ-Петербург, 2002. - 432 с.
6. Шкрыль А. А. PHP — это просто. Программируем для Web-сайта. —.СПб.: БХВ-Петербург, 2006. — 368 с.
7. Парижский С.М., Литвиненко Н.А. PHP. Теория и практика - К.: "МК-Пресс", 2006. — 384 с.
8. Харрис Э. PHP/MySQL для начинающих. / Пер. с англ. —М.: КУДИЦ-ОБРАЗ, 2005.—384 с.
9. Мазуркевич А.М., Еловой Д.С. PHP: настольная книга программиста. — М.: Новое знание, 2004. — 479 с.: ил.
10. Ульман Л. MySQL / Пер. с англ. — М.: ДМК Пресс; СПб.:

Питер, 2004.-352 с.: ил.
11. Колисниченко Д.Н. Самоучитель PHP 5.—СПб.: Наука и Техника, 2004, —576 с: ил.
12. Холкин И. И. Интернет-технологии и системы. Методические указания по выполнению лабораторных работ. Часть 2, № 0939, Москва, МИРЭА, 2010, — 24 с.
13. Холкин И. И. Интернет-технологии и системы. Методические указания по выполнению лабораторных работ. Часть 3, № 1173, Москва, МИРЭА, 2012, — 16 с.