Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Томский государственный университет систем управления и радиоэлектроники». (ТУСУР)

УТВЕРЖДАЮ

Заведующий кафедрой
«Управление инновациями»

_____ /*А.Ф.Уваров*
(подпись) *(ФИО)*
"_____" _____ 2012 г.

МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ К ЛАБОРАТОРНЫМ ЗАНЯТИЯМ

по дисциплине

Веб-программирование

Составлено кафедрой «Управление инновациями»

Для студентов, обучающихся
по направлениям подготовки
221000.62 «Мехатроника и робототехника»
221400.62 «Управление качеством»
222000.62 «Инноватика»

Форма обучения очная

Составитель
к.т.н., Титков Антон Вячеславович
" 20 " сентября 2012 г

Томск 2012 г.

СОДЕРЖАНИЕ

Введение

Изучение дисциплины «Веб-программирование» имеет важное значение в специальной подготовке студентов по направлениям «Мехатроника и робототехника», «Управление качеством», «Инноватика» и предоставляет им инструментарий презентации в интернет своих разработок, своей работы, деятельности инновационной фирмы путем размещения информации на веб-сайтах.

Цель данного пособия состоит в выработке навыков в программировании веб-сайтов и их разработке. Для полноценного понимания и усвоения материала необходимо предварительно изучить дисциплины «Информатика» и «Математика».

Для углубленного изучения и освоения материала целесообразно выполнение лабораторных работ, наряду с другими различными формами обучения студентов: тесты, задачи, упражнения, которые используются при проведении практических занятий в университете, выполнение контрольных и аудиторных работ, а также при самостоятельном изучении данной дисциплины.

Одним из наиболее интенсивных способов изучения дисциплины является самостоятельное выполнение лабораторных работ, на которых вырабатываются навыки построения веб-сайтов.

Предлагаемые лабораторные работы позволяют глубже освоить теоретические и практические вопросы, понять принципы программирования веб-сайтов и научиться создавать свои интернет приложения.

Лабораторная работа №1. Установка и настройка веб-сервера с PHP.

Цель занятия: научиться устанавливать на персональный компьютер программное обеспечение, необходимое для программирования и отладки веб-сценариев.

Задание: установить на flash-накопитель веб-сервер Apache, интерпретатор языка PHP, текстовый редактор Notepad++, веб-браузер Firefox с расширением Firebug.

Ход работы

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

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

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

Первый режим наиболее удобен, если комплекс устанавливается на жесткий диск компьютера, а не на flash-накопитель, поэтому необходимо выбрать второй вариант. На вопрос создавать ярлыки для запуска и остановки Denwer на рабочем столе отвечайте отрицательно. Для запуска комплекса будем использовать файлы Run.exe, Stop.exe и Restart.exe в каталоге X:\WebServers\denwer\, где X:\WebServers\ - диск и папка, в которую установлен комплекс.

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

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

```
subst_drive = Z:
```

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

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

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

Для остановки Denwer выполните файл X:\WebServers\denwer\Stop.exe.

Лабораторная работа №2. Создание HTML-документа.

Цель занятия: освоить язык гипертекстовой разметки HTML, научиться создавать html-документы.

Задание: необходимо создать валидный html-документ. Документ может содержать произвольный контент и должен содержать следующий набор элементов:

- изображение (тег img);
- список (теги ul, ol, dl);
- заголовок (теги h1, h2, …, h6);
- ссылка и якорь (тег a);
- таблица (тег table);
- параграф (тег p);
- блок (тег div);
- встроенный элемент (тег span);
- форматированный текст (теги pre, code);
- жирный текст (тег b);
- курсивный текст (тег i);
- подчеркивание (тег u);
- перевод строки (тег br).

Ход работы

Задание необходимо начать с изучения основ языка разметки гипертекста и понятия валидации документов. Самоучитель HTML - http://htmlbook.ru/samhtml. Валидация документов - http://htmlbook.ru/samhtml/validatsiya-dokumentov. В качестве справочника по веб-технологиям предлагается использовать сайт http://htmlbook.ru/, где представлено множество полезной информации по HTML и CSS.

Для проверки валидности документов необходимо использовать дополнение HTML Validator для Firefox. Инструкция по установке и использованию - http://htmlbook.ru/samhtml/validatsiya-dokumentov/proverka-dannykh-na-validnost. Кроме этого для работы с html-документом потребуется дополнение Firebug. Видео-инструкция по установке и использованию - http://zuzle.name/lessons/firebug/.

В папке X:\WebServers\home\localhost\www\, где X:\WebServers\ - диск и папка, в которую установлен Denwer, создайте папку lab1. В ней создайте файл index.html со следующим содержимым:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Моя первая веб-страница</title>
  </head>
  <body>
    <h1>Заголовок страницы</h1>
    <p>Основной текст.</p>
  </body>
</html>
```

Для просмотра в браузере созданного документа введите в адресной строке X:\WebServers\home\localhost\www\lab1\index.html. В таком случае браузер загружает файл index.html непосредственно с диска компьютера. В нашем случае необходимо, чтобы браузеру данный файл передавал веб-сервер. Для этого введите в адресной строке браузера адрес http://localhost/lab1/index.html. В этом случае браузер обращается по адресу http://localhost, где localhost означает текущий компьютер, вместо localhost можно ввести IP-адрес 127.0.0.1. Веб-сервер по данному адресу (т.е. Ваш Denwer) вернет Вашему браузеру содержимое файла index.html, расположенного в директории (папке) lab1. Если все сделано правильно, то в браузере Вы увидите результат, как показано на рисунке 1.

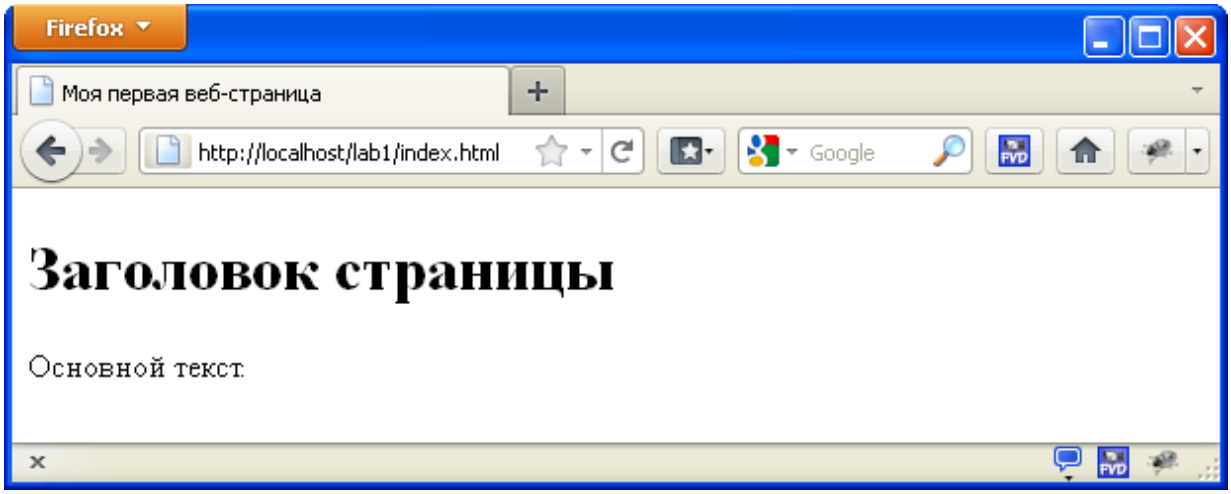

Рисунок 1 – Вид веб-страницы в окне браузера

Если браузер выдаст сообщение, что файл не найден, значит, произошла ошибка, возможно, не запущен веб-сервер.

Если в адресной строке вводить адрес без указания конкретного файла, например, http://localhost/lab1/, то веб-сервер вернет содержимое файла index.html или результат выполнения файла index.php (если конечно в директории такие файлы существуют).

Заполните созданный документ произвольным содержимым. Используйте все теги, указанные в задании. Описание структуры html-документа - http://htmlbook.ru/samhtml/struktura-html-koda. Справочник тегов HTML - http://htmlbook.ru/html. Заголовок документа head должен содержать теги meta с различными атрибутами (http://htmlbook.ru/html/meta).

При помощи Firebug изучите DOM (Document Object Model) созданного Вами документа (окно Firebug открывается клавишей F12). Что такое DOM - http://javascript.ru/tutorial/dom/intro.

Лабораторная работа №3. Построение системы HTML-документов и их оформление при помощи CSS.

Цель занятия: научиться создавать систему html-документов, освоить оформление html-документов при помощи каскадных таблиц стилей.

Задание: создайте систему html-документов, соединенных друг с другом ссылками. Разработанные документы оформите при помощи стилей CSS.

Ход работы

Создайте 5 html-документов. Изучите основы стилей CSS (http://htmlbook.ru/samcss) и оформите созданные html-документы при помощи CSS. Существует 3 способа подключения стилей CSS к html-документу:

1. описание в атрибуте style тегов:

```
<div style="border: 1px solid gray;">…</div>;
```

2. описание в теге style, расположенном в заголовке документа:

```
<head>
  …
  <style>
    div {
      border: 1px solid gray;
      float: left;
    }
  </style>
  …
</head>
```

3. подключение из внешнего css-файла при помощи тега link:

```
<link rel="stylesheet" type="text/css" href="style.css">
```

при этом style.css содержит описание правил css, например:

```
div {
  border: 1px solid gray;
  float: left;
}
#menu div {
  margin: 10px;
  background: #CCCCCC url(./path/to/image.jpg) repeat-x top left;
}
```

Используйте все 3 способа подключения CSS.

Для оформления документов используйте следующие свойства CSS:

– font-family, font-size, font-weight, color, text-decoration;
– margin, padding;
– background-color, background: #FFFFFF url(./path/to/image.jpg) repeat-x top left;
– border: 1px solid gray; для таблиц border-collapse (separate, collapse);
– для div: display (block, inline), float, position (absolute, relative).

Справочник свойств CSS - http://htmlbook.ru/css.

В верхней части каждого документа должно быть расположено меню для перехода между документами. Переходы между документами сделайте при помощи относительных и абсолютных ссылок (http://htmlbook.ru/samhtml/ssylki/absolyutnye-i-otnositelnye-ssylki).

При помощи Firebug изучите элементы Ваших страниц (в правой части окна Firebug вкладки «Стиль», «Скомпилированный стиль», «Макет», «DOM»). Найдите соответствие между атрибутами тегов во вкладке HTML основного окна Firebug и их свойствами во вкладке DOM (пояснения - http://javascript.ru/tutorial/dom/attributes).

Лабораторная работа №4. JavaScript. Динамическое изменение HTML-документа в браузере.

Цель занятия: изучить основы javascript и научиться динамически изменять html-документ.

Задание: создать html-документ, сделать динамическое создание списка группы.

Ход работы

Создайте html-документ и вставьте в него нумерованный список с атрибутом id, равным list, затем добавьте две строчки.

```
<p onclick=”add_to_list()”> Добавить в список</p>
<ul id=”list”></ul>
```

onclick=”add_to_list()” означает, что для данного элемента <p> задаем javascript событие, которое будет срабатывать при клике на элемент, а в тегах <ul> будет находиться будущий список.

Добавьте в <head> тег <script> с атрибутом type равным "text/javascript". Здесь можно уже вставлять javascript код. Создайте массив, содержащий в себе список группы.

```
var arr = ["Ivanov", "Petrov", "Sidorov"];
```

Так же необходимо создать переменную для счета элементов в массиве.

```
var counter = 0;
```

Теперь настало время для определения функции add_to_list(). Алгоритм работы этой функции следующий:

1. Проверить, есть ли в массиве запись, если нет, то выйти из функции. Это делается с помощью условия if (typeof(arr[counter]) != "undefined"). Функция typeof() возвращает тип данных объекта, находящегося в скобках. В данном случае тип данных элемента массива.

2. Получить список, куда мы будем добавлять созданный элемент, и присвоить его переменной list. Для выполнения этого пункта необходимо дать понятие о DOM (document object model) - объектная модель, используемая для XML/HTML-документов. Согласно DOM-модели, документ является иерархией. Каждый HTML-тег образует отдельный элемент-узел, каждый фрагмент текста - текстовый элемент, и т.п. Проще говоря, DOM - это представление документа в виде дерева тегов. Это дерево образуется за счет вложенной структуры тегов плюс текстовые фрагменты страницы, каждый из которых образует отдельный узел.

Для манипуляций с DOM используется объект document и функции:

getElementById('el_id') – возвращает элемент дерева с id равным el_id.

getElementsByTagName('li') – возвращает массив элементов дерева, тэгом которых является тэг li .

getElementsByName(‘el_name’) – возвращается массив элементов дерева, у которых атрибут name равен el_name.

createElement(‘element’) – функция для создания нового элемента, где element – это тот элемент, который необходимо создать.

appendChild(‘element’) – добавляет в DOM новый элемент.

var list = document.getElementById('list') – получает элемент с id равным list и присваивает объект в переменную.

3. Создать элемент li и присвоить его переменной li. Чтобы создать элемент необходимо воспользоваться функцией createElement('element'), где element – это тот элемент, который необходимо создать.

```
var li = document.createElement('LI');
```

4. Добавить в него текст. Для этого используется свойство innerHTML, которое позволяет вставить код внутрь элемента.

```
li.innerHTML = arr[counter];
```

5. Вставить элемент в конец списка при помощи метода appendChild.

```
list.appendChild(li);
```

6. Увеличить счетчик.

```
counter++;
```

Лабораторная работа №5. Фреймворк jQuery для JavaScript.

Цель занятия: изучить работу с jQuery.

Задание: при помощи jQuery сформировать часть html-документа.

Ход работы

jQuery – это библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML. Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими.

Скачайте библиотеку jQuery с http://jquery.com/ и подключите ее в новом html файле. Затем создайте переменные для счетчика (counter) и массив, хранящий список группы (arr). Добавьте в body следующий код:

```
<p onclick="add_to_list()"> Добавить в список</p>
<ul id="list">
</ul>
```

Определите функцию add_to_list() и сделайте проверку массива на существование в массиве записи с номером, хранящемся в переменной counter.

Вся работа с jQuery ведется с помощью функции $. Получить элемент в jQuery можно по его любому параметру. Для этого используются селекторы.

$("div") – получим все элементы div.

$("#my_id") – получим все элементы с атрибутом id равным my_id.

$(".my_class") – получим все элементы с атрибутом class равным my_class.

Это основные типы селекторов. Так же можно указывать связи между ними. Подробно об этом можно прочитать на сайте http://jquery.com/.

Следующая строчка получает список и добавляет в конец новую запись из массива.

```
$("#list").append("<li>"+arr[counter]+"</li>");
```

Инкрементируйте счетчик.

Все готово, теперь можно увидеть, на сколько jQuery облегчает работу web-разработчикам. Так же в jQuery входят функции для создания различных визуальных эффектов, такие как анимация, появление и затухание, скольжение и прочие. О них так же можно почитать на официальном сайте jQuery.

Добавьте в body две строчки:

```
<p id="fade" onclick="fade()">Скрыть список</p>
<p id="show" onclick="show()" style="display:none">Показать список</p>
```

Причем вторая строка будет по умолчанию скрыта. Затем создайте функцию fade() с кодом:

```
$("#list").fadeOut('slow', function(){
  $("#fade").css('display', 'none');
  $("#show").css('display', '');
});
```

Здесь мы применяем к списку “list” функцию fadeOut() – это стандартная функция из библиотеки jQuery для затухания объекта. Она принимает два значения:

1. Скорость затухания (fast, slow).
2. Функция, которая будет вызвана по завершению.

Дальше скрываем элемент с id равным fade и показываем элемент с id равным show.

Создайте функцию show по аналогии с fade, только сделайте появление быстрым и при ее вызове будет скрыта строка «Показать список» и, наоборот, показана строка «Скрыть список».

Лабораторная работа №6. PHP. Создание страницы авторизации. POST и GET запросы.

Цель занятия: изучить основы php, создать страницу авторизации на основе GET и POST запросов.

Задание: при помощи php создать страницу авторизации.

Ход работы

Для разработки страницы авторизации для гостевой книги, создайте папку в каталоге «C:\WebServers\home\localhost\www» с именем вашего сайта. Далее поместите в вашу папку файл с именем index.php – файлы index.php и index.html запускаются автоматически при переходе на ваш сайт.

index.php будет содержать в себе текст с приветствием, где имя пользователя будет передаваться GET запросом. GET запрос – это самый распространенный вид HTTP запроса. С его помощью происходит запрос любого файла на сервере и передача параметров. Параметры GET запроса передаются в адресной строке «resource?param1=value1&param2=value2», здесь resource – адрес сайта, затем идет знак вопроса, далее параметр и его значение. Несколько параметров разделяются знаком “&”.

Код index.php содержит в себе html с php вставками. Php код начинается с «<?php» и заканчивается «?>». Внутри будет находиться проверка GET запроса. Если параметр username существует и равен какому-либо значению, то выведем на экран «username», иначе «Гость». Работа с параметрами запросов осуществляется через массивы $_GET и $_POST.

```
if ($_GET['username'])
 echo $_GET['username'];
else
  echo 'Гость';
```

Затем разместите в index.php ссылку на файл login.php, который будет содержать html форму для авторизации и проверки логина и пароля, вшитого в php. Форма создается при помощи тега <form>. В него поместите три тега input с типами text, password и submit. Для тега form задайте атрибут method с параметром POST. То есть отправка формы произойдет при помощи метода POST. Применяется для передачи пользовательских данных заданному ресурсу. Например, в блогах посетители обычно могут вводить свои комментарии к записям в HTML-форму, после чего они передаются серверу методом POST и он помещает их на страницу. При этом передаваемые данные (в примере с блогами — текст комментария) включаются в тело запроса. Аналогично с помощью метода POST обычно загружаются файлы на сервер. Атрибут action оставим пустым, это означает, что после отправки формы, будет происходить перенаправление на эту же страницу.

Как правило, при разработке приложения требуется отладка, т.е. процесс нахождения, локализации и устранения ошибок. Для этого в php существует функция var_dump(), которая возвращается строку - структуру объекта, который был передан ей. Но все данные в этой строке не отформатированы, поэтому для правильного отображения необходимо разместить это строку внутри тегов pre. Поэтому нужно создать новый файл debug.php и создать там функцию для отладки.

```
function debug($obj){
  echo '<pre>';
  var_dump($obj);
  echo '</pre>';
}
```

В файле login.php для подключения файла debug.php используем функцию require_once(path_to_file), где path_to_file – путь к файлу, который нужно подключить. Теперь написав debug($_POST), на экране появятся все параметры POST запроса.

Для проверки логина и пароля зададим две переменные:

```
$username = 'user';
$password = '1234';
```

И проверка, если человек вошел, то будет отображаться надпись «Привет, username», иначе форма для входа:

```
<?php
if ($_POST['username'] == $username && $_POST['password'] == $password){
  echo 'Привет '.$_POST['username'];
} else {
?>
<h3>Войти</h3>
<form method="POST" action="">
  <p>Имя пользователя: <input type="text" name="username"></p>
  <p>Пароль: <input type="password" name="password"></p>
  <p><input type="submit"></p>
</form>
<?php
}
?>
```

Лабораторная работа №7. Работа с сессиями. Реальная авторизация и регистрация.

Цель занятия: научиться работать с сессиями.

Задание: создать авторизацию и регистрацию.

Ход работы

Веб-сервер не поддерживает постоянного соединения с клиентом, и каждый запрос обрабатывается, как новый, безо всякой связи с предыдущими. То есть, нельзя ни отследить запросы от одного и того же посетителя, ни сохранить для него переменные между просмотрами отдельных страниц. Для решения этих двух задач и были изобретены сессии.

Собственно, сессии, если в двух словах - это механизм, позволяющий однозначно идентифицировать браузер и создающий для этого браузера файл на сервере, в котором хранятся переменные сеанса.

Теперь создайте файл registration.php, который будет содержать форму регистрации пользователя.

```
<form method="POST" action="registration.php">
  <p>Имя пользователя: <input type="text" name="username"></p>
  <p>Email: <input type="text" name="email"></p>
  <p>Пароль: <input type="password" name="password"></p>
  <p>Пароль(повторить): <input type="password" name="password2"></p>
  <p><input type="submit"></p>
</form>
```

В самом начале файла необходимо вызвать функцию session_start(), она создает сессию или продолжает текущую на основе текущего идентификатора сессии, который передается через запросы, такие как GET, POST или cookie. В большинстве случаев используют сессии на cookie, поэтому перед функцией start_session() не должно быть функций, возвращающих сообщение в браузер. Затем делаем проверку, аутентифицирован ли пользователь.

```
if ($_SESSION['username']) {
  echo 'Вы уже зарегистрированы';
  return 0;
}
```

Далее проверяем, существует ли POST запрос, если да, то проверяем совпадают ли пароли и заносим в переменную сессии данные из POST запроса.

```
if ($_POST) {
  if ($_POST['password'] == $_POST['password2']) {
    $_SESSION['username'] = $_POST['username'];
    echo 'Пользователь '.$_SESSION['username'].' зарегистрирован';
    return 0;
   }
   else {
    echo 'Введенные пароли не совпадают';
    return 0;
   }
}
```

И в index.php заменяем все POST на SESSION.

Теперь необходимо создать файл logout.php. Этот файл будет содержать функцию, которая разрушит все данные, зарегистрированные в сессии – session_destroy().

```
<?php
session_start();
session_destroy();
header('Location: index.php');
?>
```

Перед тем как отправить форму, ее нужно проверить. Проверка формы будет осуществляться посредством javascript. Для этого у формы определим событие onsubmit="return checkForm(this)". Функция checkForm будет вызываться перед отправкой данных. Проверять будем имя пользователя, заполнено он или нет и email. Шаблон email`a будет [английские_буквы]@ [английские_буквы]. [английские_буквы]. Такую проверку можно сделать, используя регулярные выражения - это формальный язык поиска и осуществления манипуляций с подстроками в тексте. По сути это строка-образец, состоящая из символов и метасимволов и задающая правило поиска. Более подробно о регулярных выражениях можно почитать на Википедии (http://ru.wikipedia.org/wiki/Регулярные_выражения).

Вообще регулярное выражение для проверки электронной почты довольно громоздкое:

```
^[-a-z0-9!#$%&'*+/=?^_`{|}~]+(?:\.[-a-z0-9!#$%&'*+/=?^_`{|}~]+)*@(?:[a-z0-9]([-a-z0-9]{0,61}[a-z0-9])?\.)*(?:aero|arpa|asia|biz|cat|com|coop|edu|gov|info|int|jobs|mil|mobi|museum|name|net|org|pro|tel|travel|[a-z][a-z])$
```

Но оно слишком тяжело для понимания, поэтому воспользуемся простой формой `[a-zA-Z]*@[a-zA-Z]*\.[a-zA-Z].`

В начале функции определим все переменные, которые нам понадобятся, и массивы с ошибками.

```
var el, // Сам элемент
elName, // Имя элемента формы
value, // Значение
type; // Атрибут type для input-ов
reg = /[a-zA-Z]*@[a-zA-Z]*\.[a-zA-Z]/;
var errorList = [];
var errorText = {
   1 : "Не заполнено поле 'Имя'",
   2 : "Не заполнено поле 'E-mail'",
}
```

Далее проходимо по всем элементам формы и проверяем все теги input. Если поле для имени пустое или электронная почта не соответствует шаблону, записываем номера ошибок в массив.

```
for (var i = 0; i < form.elements.length; i++) {
  el = form.elements[i];
  elName = el.nodeName.toLowerCase();
  value = el.value;
  if (elName == "input") {
    type = el.type.toLowerCase();
    if (type == "text") {
        if (el.name == "name" && value == "") errorList.push(1);
        if (el.name == "email" && !value.match(reg)) errorList.push(2);
    }
}
```

Затем проверяем массив с ошибками. Если он пуст, то возвращаем true, иначе формируем текст для ошибки, выводим это на экран и возвращаем false.

```
if (!errorList.length) return true;
var errorMsg = "При заполнении формы допущены следующие ошибки:\n\n";
for (i = 0; i < errorList.length; i++) {
  errorMsg += errorText[errorList[i]] + "\n";
}
alert(errorMsg);
return false;
```

Лабораторная работа №8. Чтение и запись в файл. Регистрация с записью в файл. Авторизация из файла.

Цель занятия: научиться работать с файлами.

Задание: создать авторизацию и регистрацию на файлах.

Ход работы

Для работы с файлами в php существует несколько функций.

fopen – функция для открытия файла.

```
$fp = fopen('filename, 'param');
```

filename и param это обязательные параметры. Первый отвечает за имя файла, который необходимо открыть, а второй определяет режим файла:

r – открытие файла только для чтения.

r+ - открытие файла одновременно на чтение и запись.

w – создание нового пустого файла. Если на момент вызова уже существует такой файл, то он уничтожается.

w+ - аналогичен r+, только если на момент вызова фай такой существует, его содержимое удаляется.

a – открывает существующий файл в режиме записи, при этом указатель сдвигается на последний байт файла (на конец файла).

a+ - открывает файл в режиме чтения и записи при этом указатель сдвигается на последний байт файла (на конец файла). Содержимое файла не удаляется.

Записывать данные в файл при помощи PHP можно при помощи функции fwrite(). Это функция принимает 2 обязательных параметра и 1 необязательный. В качестве обязательных параметров выступает дескриптор файла и режим файла:

```
$test = fwrite($fp, $mytext);
```

По завершению работы с файлом, его нужно закрыть, используя функцию fclose($fp).

Теперь в файле registration.php после строки if '($_POST) {' поставим проверку файла, если он существует, то открываем его и перемещаем указатель в конец строки, если нет, то создаем его.

```
$fp = fopen('users.txt', 'a+');
if (!$fp)
  $fp = fopen('users.txt', 'w+');
```

В условие проверки паролей сформируем строку с данными, которые будут разделены знаком &, запишем ее в файл и закроем его.

```
$mytext = 'username='.$_POST['username'].'&password='
  .$_POST['password'].'&email='.$_POST['email']. "\r\n";
$test = fwrite($fp, $mytext);
fclose($fp);
```

Следующим шагом будет преобразование файла login.php. Теперь мы будем получать данные о пользователях не в самом сценарии, а из отдельного файла. Для этого в самом начале файла снова начнем сессию, подключим файл debug.php для отладки и делаем проверку, если существует POST запрос, то открываем файл user.txt, построчно считываем его содержимое и сравниваем с данными POST запроса. В случае успеха, добавляем пользователя в сессию и переходим на index.php.

```
<?php
session_start();
require_once('debug.php');
if ($_POST) {
  $fp = fopen('users.txt', 'r');
  while (!feof ($fp)) {
    $buffer = fgets($fp);
    preg_match('/username=([^&]*)&/', $buffer, $user);
    preg_match('/password=([^&]*)&/', $buffer, $pass);
    if ($_POST['username'] == $user[1] &&
      $_POST['password'] == $pass[1]) {
      $_SESSION['username'] = $_POST['username'];
      header('Location: index.php');
    }
  }
  fclose ($fp);
} else {
?>
<html>
  <head>
  <title>Login page</title>
</head>
<body>
  <h3>Войти</h3>
  <form method="POST" action="">
    <p>Имя пользователя: <input type="text" name="username"></p>
    <p>Пароль: <input type="password" name="password"></p>
    <p><input type="submit"></p>
  </form>
  <?php
  }
  ?>
</body>
</html>
```

Лабораторная работа №9. Гостевая книга на файлах.

Цель занятия: создание гостевой книги на файлах.

Задание: создать гостевую книгу.

Ход работы

Первоначально нужно определить функционал гостевой книги, т.е. как она будет работать. Сообщения могут оставлять только зарегистрированные пользователи, а пользователи – гости будут видеть только сами комментарии и надпись, что нужно зарегистрироваться. Гостевая книга будет находиться в файле index.php. Все данные будут записываться в файл guestbook.txt.

Для создание гостевой книги необходима форма, которая будет добавлять сообщения, поэтому разместим ее в файле после поля для логина.

```
<form method="POST">
  <p> Тема поста: <input type="text" name="theme"></p>
  <p>Текст поста:
    <textarea rows="10" cols="45" name="text"></textarea></p>
  <p><input type="submit"></p>
</form>
```

Затем запрос этой формы необходимо обработать. Поместим в начало файла после запуска сессии проверку, если существует POST запрос с параметрами theme, text и пользователь находится в сессии, то открываем или создаем файл guestbook.txt, записываем туда данные, закрываем файл и создаем переменную с сообщением "Комментарий был успешно добавлен".

```
if ($_POST['theme'] && $_POST['text'] && $_SESSION['username']) {
  $fp = fopen('guestbook.txt', 'a+');
  if (!$fp) {
    $fp = fopen('guestbook.txt', 'w+');
  }
  $mytext = 'username='.$_SESSION['username'].'&text='
    .$_POST['text'].'&theme='.$_POST['theme']."&"."\r\n";
  $test = fwrite($fp, stripslashes($mytext));
  fclose($fp);
  $msg = "Комментарий был успешно добавлен";
}
$msg
```

Перед формой разместим условие, если существует, вывести его на экран, и если пользователь не в сессии то вместо формы выводим «Чтобы оставлять комментарии вы должны быть зарегистрированы».

```
<?php
if ($msg) {
  ?><p><?php echo $msg; ?></p><?php
}
if ($_SESSION['username']) {
?>
<form method="POST">
  <p> Тема поста: <input type="text" name="theme"></p>
  <p>Текст поста:
    <textarea rows="10" cols="45" name="text"></textarea></p>
  <p><input type="submit"></p>
</form>
<?php
} else {
?>
<p>Чтобы оставлять комментарии вы должны быть зарегистрированы</p>
<?php
}
?>
```

Далее следуют сами комментарии. Выводить их будем в таблице вида

имя пользователя	заголовок
текст сообщения	

Для этого открываем файл guestbook.txt с параметром “r”, затем при помощи функций feof() и fgets() считываем построчно файл. foef() – проверяет, достигнут ли конец файла, fgets() – возвращает строку из файла. Затем при помощи регулярных выражений вытаскиваем имя пользователя, заголовок, текст сообщения и заносим данные в таблицу.

```
<table class="table">
  <?php
    $fp = fopen('guestbook.txt', 'r');
    while (!feof ($fp)) {
  ?>
  <tr class="main">
    <?php
      $buffer = fgets($fp);
      preg_match('/username=([^&]*)&/', $buffer, $user);
      preg_match('/text=([^&]*)&/', $buffer, $text);
      preg_match('/theme=([^&]*)&/', $buffer, $theme);
    ?>
    <td><?php echo $user[1];?></td>
    <td><?php echo $theme[1];?></td>
 </tr>
  <tr class="text">
    <td colspan=2><?php echo $text[1]; ?></td>
  </tr>
  <?php
  }
  fclose ($fp);
  ?>
</table>
```

В принципе гостевая книга готова, но чтобы было удобнее работать, ее нужно оформить при помощи css. Объединим поля для регистрации и входа в один div с id=reg.

```
<div class="reg">
<a href="registration.php">Регистрация</0a>
<a href="login.php">Войти</a>

<p>Привет,
<?php
  if ($_SESSION['username']) {
    echo $_SESSION['username'];
?>
<a href="logout.php">Выйти</a>
<?php
} else {
  echo 'Гость';
}
?>
</p>
</div>
```

Создадим и подключим файл guestbook.css.

Пример файла guestbook.css:

```
table {
  width: 300px;
}
.main{
  font-size: 120%;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  color: #336;
   border: 10px solid #666;
}
.text{
  color: red;
  border: 1px solid #666;
  background: #eee;
  padding: 5px;
}
.reg{
  position: absolute;
  right: 10px;
  top: 10px;
  width: 225px;
  height: 180px;
  background: #f0f0f0;
}
```

Лабораторная работа №10. Перенос функционала с файлов на СУБД.

Цель занятия: научиться работать с базами данных.

Задание: перенести системы на файлах в базу данных.

Ход работы

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

В состав Denwer входит СУБД под названием phpmyadmin. Чтобы начать работать с ней, нужно запустить Denwer и в адресной строке набрать http://localhost/Tools/phpMyAdmin/. Изучите работу с phpmyadmin.

Для работы с базой данных используйте расширение MySQLi. MySQLi является улучшенной версией старого драйвера PHP MySQL.

В файле для регистрации сразу после строчки с началом сессии подключитесь к базе данных. Это делается при помощи функции mysqli_connect(). Она создает соединение с MySQL сервером.

```
$link = mysqli_connect(
  'localhost',  /* Хост, к которому мы подключаемся */
  'root',       /* Имя пользователя */
  '',           /* Используемый пароль */
  '');          /* База данных для запросов по умолчанию */
```

Создайте проверку на подключения к базе данных.

```
if (!$link) {
  printf("Невозможно подключиться к базе данных. Код ошибки: %s\n",
  mysqli_connect_error());
  exit;
}
```

Используя функцию mysqli_query($link, «sql запрос») можно отправлять различные запросы в базу данных. Для этого используется язык запросов SQL. С синтаксисом языка можно ознакомиться на википедии. Создайте базу данных, если она не существует.

```
mysqli_query($link, 'CREATE DATABASE IF NOT EXISTS my_db');
```

Теперь нужно подключиться к базе. Это можно сделать функцией mysqli_select_db().

```
mysqli_select_db($link,'my_db');
```

Удалите все функции для работы с файлами. После проверки паролей создайте таблицу с полями id, username, email, password с проверкой существования таблицы.

```
$query = "CREATE TABLE IF NOT EXISTS users (`id` INT( 11 ) NOT NULL AUTO_INCREMENT ,`username` VARCHAR( 150 ) NOT NULL ,`email` VARCHAR( 50 ) NOT NULL ,`password` VARCHAR( 50 ) NOT NULL ,PRIMARY KEY ( `id` ));";
mysqli_query($link, $query);
```

Вставьте в базу данные, которые пришли из POST запроса.

```
mysqli_query($link, 'INSERT INTO users(`username`, `email`, `password`) VALUES ("'.$_POST['username'].'","'.$_POST['email'].'", "'.$_POST['password'].'")');
```

Если какой-то запрос не выполняется, всегда можно посмотреть в чем ошибка, используя функцию mysqli_error($link). Она хранит последнюю ошибку.

В файле index.php после запуская сессии подключитесь к базе и попытайтесь создать базу данных как и в registration.php. Создаем таблицу с полями id, id_username, title, text. id_username хранит в себе id пользователя из таблицы users. После проверки POST запроса уберите все функции для работы с файлами и сделайте запрос в базу для получения id пользователя, присвойте его переменной $id. При выполнении запроса SELECT, mysqli_query возвращает массив, который нужно обработать функцией mysqli_fetch_assoc(), на выходе которой получается ассоциативный массив.

```
$result = mysqli_query($link, 'SELECT id FROM `users` WHERE username="'.$_SESSION['username'].'"');
while($row = mysqli_fetch_assoc($result) )
  $id = $row['id'];
```

Измените код для генерации столбцов для комментариев. Для этого получите все данные из таблицы messages и по id_username получите имя пользователя из таблицы users.

```
<?php
  $result = mysqli_query($link, 'SELECT * FROM `messages`');
  while($row = mysqli_fetch_assoc($result) ){
    $id = $row['id_username'];
    $text =$row['text'];
    $theme =$row['title'];
    $username_results = mysqli_query($link, 'SELECT username FROM `users` WHERE id="'.$id.'"');
    while($u_row = mysqli_fetch_assoc($username_results) )
      $username = $u_row['username'];
  ?>
```

```
<tr class="main">
  <td> <?php echo $username; ?> </td>
  <td> <?php echo $theme; ?> </td>
</tr>
<tr class="text">
  <td colspan=2><?php echo $text; ?> </td>
</tr>
<?php } ?>
```

Рекомендуемая литература

Основная литература:

1. Баранов Д.В. Построение эффективного взаимодействия с web-сайтом. HTML. CSS: Учебное пособие / Д. В. Баранов; Министерство образования Российской Федерации, Томский государственный университет систем управления и радиоэлектроники, Институт дополнительного образования. - Томск: ТУСУР, 2004. - 291[1] с.: ил. - Библиогр.: с. 292.

2. Дунаев В.В. Самоучитель JavaScript : самоучитель / В. В. Дунаев. - 2-е изд. - СПб.: Питер, 2005. - 394[6] с.: ил. - (Самоучитель). - Загл. на корешке :JavaScript. - Алф. указ.: с. 385-394. - ISBN 5-469-00804-5

3. Губин И.Г. Технология создания интернет-приложений : учебное пособие: в 4 разделах / И. Г. Губин; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники, Кафедра компьютерных систем в управлении и проектировании. - Томск: ТМЦДО, 2007. Раздел 3: Основы PHP и MySQL. - Томск: ТМЦДО, 2007. - 144 с.: ил.

4. Губин И.Г. Технология создания интернет-приложений : учебное пособие: в 4 разделах / И. Г. Губин; Федеральное агентство по образованию, Томский государственный университет систем управления и радиоэлектроники, Кафедра компьютерных систем в управлении и проектировании. - Томск: ТМЦДО, 2007. Раздел 4: Основы PHP и MySQL. - Томск: ТМЦДО, 2007. - 142 с. : ил., табл.

Дополнительная литература:

1. Практические занятия по PHP4: Краткий курс / В. А. Будилов; Ред. С. Л. Корякин-Черняк. - СПб.: Наука и Техника, 2001. - 352 с.

2. Знакомьтесь: World Wide Web: Учебное пособие: Пер. с нем. / Матиас Нольден. - Киев: BHV, 1996. - 336 с.

3. JavaScript, XML и объектная модель документа: учебное пособие / В. А. Будилов. - СПб.: Наука и Техника, 2001. - 348[4] с.

4. Д. Скляр, А. Трахтенберг PHP. «Рецепты программирования PHP». Издательства: Русская Редакция, БХВ-Петербург, 2007 г., 736 стр.

5. М.Браун, Д.Ханикатт «HTML в подлиннике», издательство: BHV, 2002, 1024 стр., перевод с английского

6. Муссиано, Кеннеди «HTML и XHTML. Подробное руководство», издательство: Символ-Плюс, 2008г., 752 стр, перевод с английского.

7. PHP: hypertext preprocessor. URL: http://php.net

8. Флэнаган Д. «JavaScript. Подробное руководство» Издательство: Символ-Плюс, 2008г., 992 стр, перевод с английского.