

СЕВАСОПОЛЬСКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ



Создание динамического сайта на PHP и MySQL

Методические указания
к выполнению лабораторных работ
по дисциплине «Технологии Web - приложений»
для студентов очной формы обучения
по направлению 09.03.01 «Информатика и вычислительная
техника»
Часть 2

Севастополь

2014

УДК 621.396.6

МЕТОДИЧЕСКИЕ УКАЗАНИЯ к выполнению лабораторных работ по дисциплине «Технологии Web - приложений» для студентов очной формы обучения по направлению 09.03.01 «Информатика и вычислительная техника» / Сост. доцент кафедры ИТКС Лелеков С.Г. – Севастополь: Изд-во СГУ, 2015. – 36с.

Целью методических указаний является оказание помощи студентам в выполнении лабораторных работ по дисциплине «Технологии Web - приложений».

Приведены варианты заданий, рекомендации по выполнению, требования к оформлению работ.

Методические указания рассмотрены и утверждены на заседании кафедры информационных технологий и компьютерных систем, протокол №11 от 30 июня 2014г.

Допущено учебно-методическим центром СГУ в качестве методических указаний.

Рецензент: Шаталова Ю.Г. доцент кафедры «Информационных технологий и компьютерных систем».

Содержание

1. ЦЕЛИ И ЗАДАЧИ ЛАБОРАТОРНОГО ПРАКТИКУМА	4
2. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ	4
3.1. Лабораторная работа №6. «Введение в основы разработки динамических Web - сайтов с использованием языка PHP и Web – сервера Apache»	4
3.2. Лабораторная работа №7. Разработка и подключение базы данных Web - сайта на основе MySQL	8
3.2. Лабораторная работа №8. Создание динамических страниц сайта средствами PHP	12
3.3. Лабораторная работа №9. «Создание динамических страниц сайта средствами PHP. Реализация гостевой книги»	17
3.4. Лабораторная работа №10. «Создание динамических страниц сайта средствами PHP. Регистрация и авторизация пользователей»	21
3.5. Лабораторная работа №11. «Создание динамических страниц сайта средствами PHP. Организация поиска на сайте»	26
3.6. Лабораторная работа №12. «Создание динамических страниц сайта средствами PHP. Доступ администратора на сайт. Часть 1»	29
3.7. Лабораторная работа №12. «Создание динамических страниц сайта средствами PHP. Доступ администратора на сайт. Часть 2.»	32
Приложение 1. Установка Denwer.....	34

1. ЦЕЛИ И ЗАДАЧИ ЛАБОРАТОРНОГО ПРАКТИКУМА

Выполнение лабораторных работ предусмотрено рабочим планом подготовки бакалавров по направлению «Информатика и вычислительная техника» для студентов очной формы обучения.

Выполнение цикла лабораторных работ преследует следующие цели:

В первой части (лабораторные работы 1-5) - общее знакомство с методами и средствами разработки статических Web - сайтов, языком разметки HTML, стилевому оформлению страниц с помощью CSS, основам программирования на Javascript.

Во второй части (лабораторные работы 6-12) - общее знакомство с методами и средствами разработки динамических Web – сайтов с использованием средств языка сценариев PHP и СУБД MySQL.

2. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Методические указания по выполнению лабораторной работы содержат теоретическое введение, пример выполнения заданий лабораторной работы, индивидуальное задание, требования к оформлению отчета. Все варианты индивидуальных заданий имеют примерно одинаковую трудоемкость. Способ выбора варианта определен в заданиях.

Перед выполнением лабораторной работы необходимо внимательно прочесть текст методических указаний, ознакомиться с рекомендуемой литературой. Предлагаемый пример выполнения задания желательно повторить, чтобы уже на его основе выполнять индивидуальное задание. При этом допускается использование других средств и способов выполнения заданий, не рассмотренных в данных методических указаниях.

По результатам выполнения работы оформляется печатный отчет, который обязательно должен содержать:

- титульный лист;
- описание выполнения индивидуального задания;
- список использованной литературы.

Требования к описанию выполнения индивидуального задания в каждой работе свои и изложены в методических указаниях к ее выполнению.

3.1. Лабораторная работа №6. «Введение в основы разработки динамических Web - сайтов с использованием языка PHP и Web – сервера Apache»

Введение

Все сайты в сети могут быть двух типов: статические и динамические сайты. С точки зрения посетителя сайта порой не важно, на какой странице он находится, статической или динамической, иногда даже трудно точно это определить.

Статической называется страница, которая целиком хранится на сервере и показывается посетителю в своем неизменном виде (следует учесть, что статическая страница может содержать некоторые изменяемые элементы, например баннеры, однако она все равно остается статической).

Динамической называется страница, которая формируется сервером из нескольких частей или путем внесения(изменения) данных.

Рассмотрим процесс генерации динамической страницы подробнее. Если статическая страница не претерпевает практически никаких изменений и попадает на экран к посетителю в своем "первозданном" виде, то динамической страницы, такой, как ее видит

человек, на сервере не существует. Она собирается различными способами из данных, хранящихся на сервере.

С появлением средств программирования, выполняемых на стороне сервера, появилась возможность вносить изменения в данные отправляемые посетителю. Это позволяет изменять контент, реагируя на действия посетителя. Добиться этого можно только разделив информацию и дизайн сайта.

Использование динамических страниц позволяет хранить некий шаблон дизайна, в который, в зависимости от страницы, на которую зашел посетитель, помещается необходимое наполнение. Такой вариант очень удобен, ведь для всего сайта создается один или несколько шаблонов, и все изменения дизайна, которые требуется сделать на сайте, производятся только с ними.

Данный подход также позволяет разграничить полномочия людей, занимающихся наполнением и модификацией сайта. Т.е. одни сотрудники могут заниматься дизайном, другие наполнением страниц. В идеале они даже не будут пересекаться, только посетитель сайта будет видеть результат их совместной работы на экране своего монитора.

Ход работы

Продолжим работу над сайтом об автомобилях, начатом в третьем задании.

Далее требуется создать еще ряд страниц, например, с перечнем автомобилей, их конкретным описанием и др. Количество их может меняться, они могут претерпевать изменения. Все это требует внесения некоторой динамики в проектируемый сайт.

Начнем с того, что большинство страниц будет содержать повторяющиеся фрагменты, например заголовок и верхнее меню с формой.

Выделим эту часть в самостоятельный блок и сохраним в отдельном файле. Для этого: Создадим папку blocks, в ней файл top.php.

Вырежем из файла index.php все, что касается заголовка и верхнего меню с формой авторизации и поместим во вновь созданный файл.

На место вырезанного фрагмента вставляем его запрос на php.

```
<?php
    require_once "blocks/top.php";
?>
```

При формировании динамической страницы Web – сервер найдет файл top.php и вставит его вместо запроса.

Дальнейшая работа над заданием требует установки Web – сервера Apache, например, в составе Denwer(см. приложение 3). Папка www, созданная при выполнении заданий лабораторной работы 2.2 должна находиться в папке с именем Вашего host, например, proba_control.local.

После старта Denwer, если все сделано правильно при вводе в браузер адреса www.proba_control.local должен отображаться Ваш сайт об автомобилях. Если этого не происходит, нужна консультация у преподавателя!

Такие же действия проделаем:

1. С главной статьей. Сохраним все теги в папке blocks в файле main_article.php.
2. С баннерами. Сохраним все теги в папке blocks в файле banners_240.php.
3. С подвалом. Сохраним все теги в папке blocks в файле footer.php.

Следующий шаг – создание новых страниц. Создадим страницу со статьями articles.php, на основе страницы index.php. Вместо заголовка «Сайт об автомобилях» напишем «Статьи».

В основном блоке заменим main_article.php на list_articles.php.

Получим следующий код:

```

<head>
  <title> Статьи </title>
  <meta http-equiv="Content-type" content="text/html; charset=1251" />
  <link rel="stylesheet" type="text/css" href="styles/main.css" />
</head>
<body>
  <table id="main">
    <?php
      require_once "blocks/top.php";
    ?>

    <tr> <!-- начало контента -->
      <td colspan="2">
        <table id="content">
          <tr>
            <td> <!--1-я ячейка -->
              <?php
                require_once "blocks/list_articles.php";
              ?>
            </td>
            <td id="banners_240"> <!--2-я ячейка -->
              <?php
                require_once "blocks/banners_240.php";
              ?>
            </td>
          </tr>
        </table>
      </td>
    </tr> <!-- конец контента -->
    <tr>
      <td colspan="2">
        <hr />
      </td>
    </tr>
    <tr>
      <?php
        require_once "blocks/footer.php";
      ?>
    </tr>
  </table>
</body>

```

В папке blocks создадим новый файл list_articles.php. Разместим в нем следующий текст:

```

<div class="article">
  <h2>Статья 1</h2>
  <p class="article_img">
    
  </p>
  <p>Превью...</p>
  <p>
    <a href="article.php?id=1"> Читать дальше </a>
  </p>
</div>
<hr />

```

В файле main.css добавим выравнивание по центру.
 .article h1, .article h2, .article_img {
 text-align: center;
}

Для надо добавить класс с атрибутом width:200px!!!

Все готово для добавления контента. Заменяем заголовок «Статья1» и абзац «Превью» на тексты из приложения 2. В качестве абзаца «Превью» возьмем первые абзацы соответствующих текстов из приложения 2. Так для Polo Sedan:

```
<div class="article">
    <h2>Polo Sedan</h2>
    <p class="article_img">
        
    </p>
    <p>Новый Volkswagen Polo Sedan появился в продаже летом 2010 г. Автомобиль спроектирован специально для российского рынка на базе бюджетного бразильского VW Pointer (Gol) Sedan. Предшественником Фольксваген Поло седан был Volkswagen Pointer, однако появление этой модели не нашло широкого отклика у российских автолюбителей.</p>
    <p>
        <a href="article.php?id=1"> Читать дальше </a>
    </p>
</div>
<hr />
```

Эту операцию следует проделать для всех автомобилей.

Реализация ссылки `<a href="article.php?id=1"> Читать дальше </a>` будет рассмотрена во второй части курса. А пока можно подготовить файлы для ее реализации.

Создадим страницу с полным текстом статьи. Для этого в главной папке создадим файл article.php.

Основу для него копируем из файла www/articles.php.

В блоке контента заменяем

```
require_once "blocks/list_articles.php";
на
require_once "blocks/full_article.php";
```

Создадим теперь блок full_article.php. Основу берем из блока blocks/list_articles.php

```
<div class="article">
<h2>Polo Sedan</h2>
    <p class="article_img">
        
    </p>
    <p>Новый Volkswagen Polo Sedan появился в продаже летом 2010 г. Автомобиль спроектирован специально для российского рынка на базе бюджетного бразильского VW Pointer (Gol) Sedan. Предшественником Фольксваген Поло седан был Volkswagen Pointer, однако появление этой модели не нашло широкого отклика у российских автолюбителей.</p>
    <p>
        <a href="article.php?id=1"> Читать дальше </a>
    </p>
</div>
<hr />
```

Убираем нижнюю ссылку «Читать далее» и добавляем вместо нее полный текст из файла text.txt.

Эту операцию следует проделать для всех автомобилей.

Сайт готов.

3.2. Лабораторная работа №7. Разработка и подключение базы данных Web - сайта на основе MySQL

Цель: Создать базу данных сайта.

База данных набор взаимосвязанных таблиц, содержащих всю необходимую для функционирования объекта информацию.

Для работы с базой данных (создания таблиц, их заполнения, извлечения из них информации) требуется СУБД – система управления базами данных. Мы будем использовать, входящую в состав Denwer, СУБД PhpMyAdmin - это программа написанная на [PHP](#) и предназначенная для управления сервером [MySQL](#). PhpMyAdmin поддерживает широкий набор операций над MySQL. Наиболее часто используемые операции поддерживаются с помощью пользовательского интерфейса (управление базами данных, таблицами, полями, связями, индексами, пользователями, правами, и т. д.), одновременно вы можете напрямую выполнить любой SQL запрос.

Чтобы подключиться к PhpMyAdmin:

- 1) убедитесь, что у Вас запущен Denwer (см. Приложение 3 в части1 методических указаний)
- 2) в строке заголовка браузера наберите localhost/tools/phpmyadmin(рис.1).

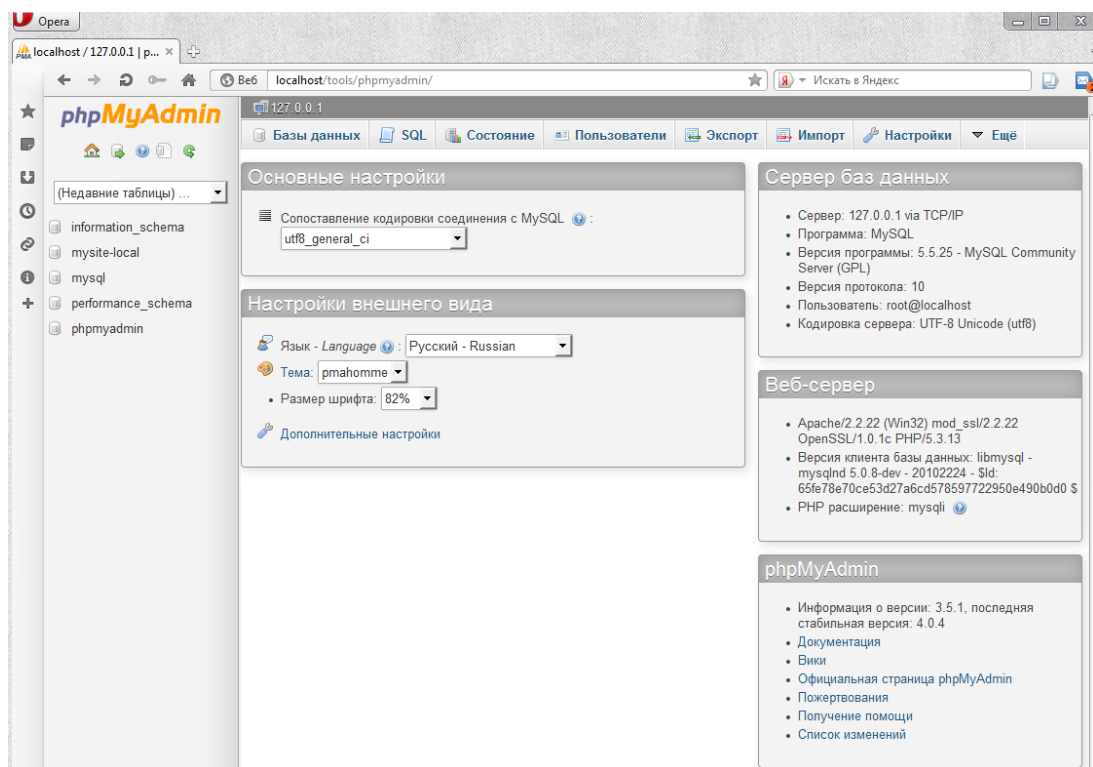


Рис.1 Стартовое окно PhpMyAdmin

Создание новой базы данных.

Чтобы создать новую базу данных, надо перейти на вкладку «Базы данных», дать название, например mydatabase и выбрать кодировку БД, используя список «Сравнение». Так как предполагается использование только символов русского и английского алфавита, то можно воспользоваться кодировкой cp1251_general_ci. Нажать кнопку «Создать» (рис.2).

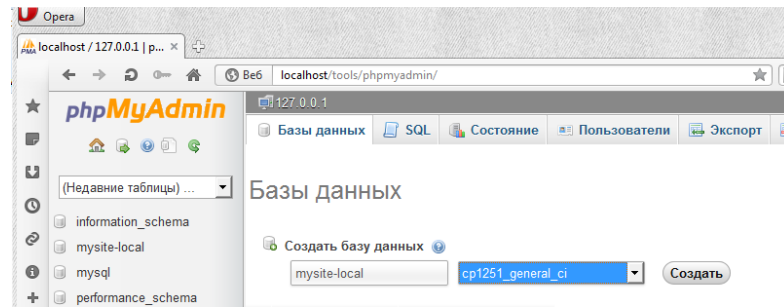


Рис.2 Создание базы данных

База данных создана. Приступим к созданию таблиц, необходимых для сайта об автомобилях, проектирование которого было начато в предыдущих лабораторных работах.

Первой создадим таблицу со статьями “articles”, в которую требуется перенести все имеющиеся в нашем распоряжении статьи.

Столбцами таблицы будут: id – номер строки (записи); title – название статьи; intro_text – краткое содержание статьи; full_text – полное содержание статьи. Всего 4 поля. Нажимаем создать (рис.3).

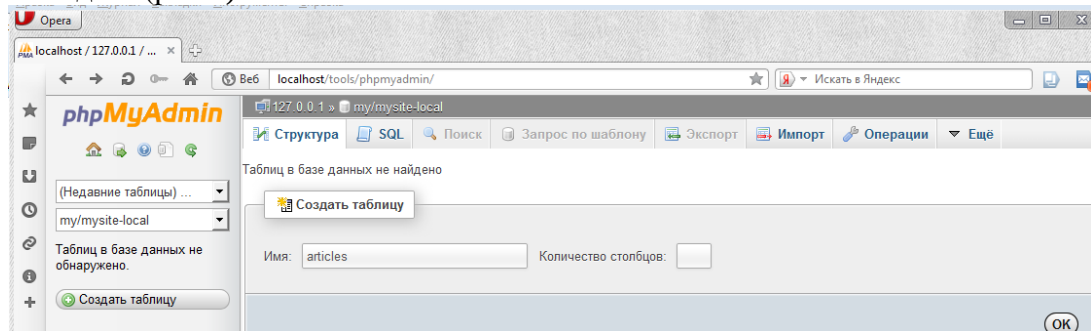


Рис.3. Создание таблицы articles.

Появится конструктор создания полей, где всю эту информацию и изложим (рис 4).

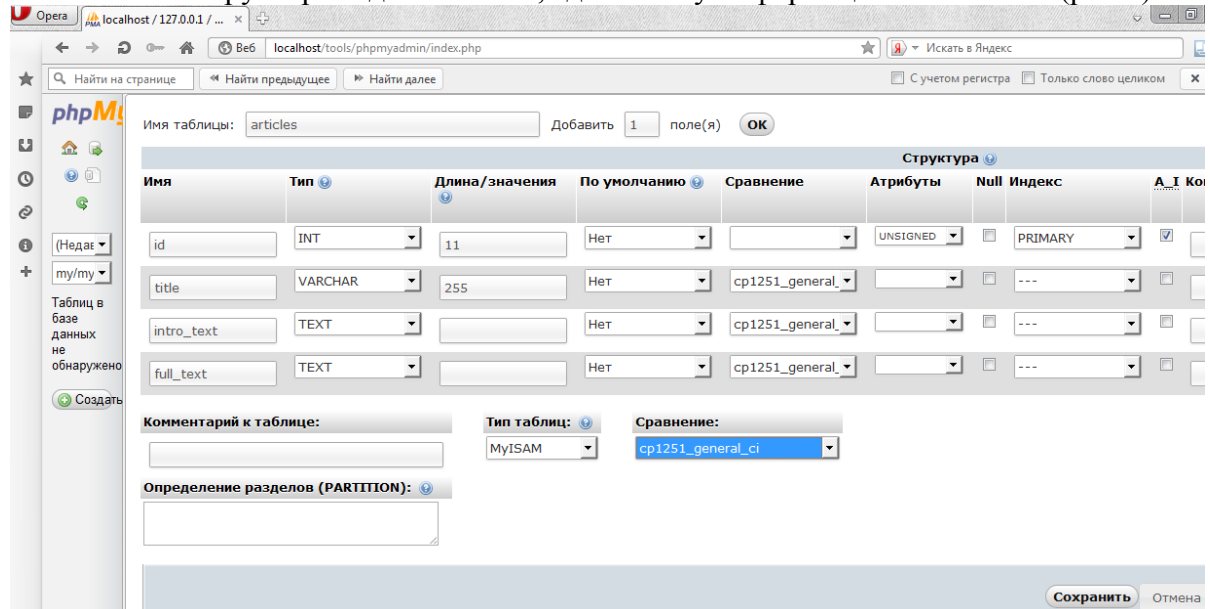


Рис. 4. Структура таблицы articles/

Здесь тип поля выбран: INT для номера строки, VARCHAR для названия статьи, TEXT для краткого и полного содержания статьи. Длины полей – максимальные значения для этих типов. Сравнение: для текстовых кодировка cp1251_general_ci. Атрибут целого без знаковый UNSIGNED. Id – ключевое поле (PRIMARY) с АВТОИНКРЕМЕНТОМ.

Тип таблиц можно выбрать MyISAM, кодировка cp1251_general_ci.
Остальные поля можно не изменять.
После нажатия кнопки «Сохранить» таблица будет создана.

Добавим записи в таблицу articles. Для этого открываем таблицу и используем вкладку «Вставить» (Рис.5).

Данные для заполнения берем из Приложения 2 первой части методических указаний.
Информацию в полях intro_text и full_text оформляем тегами HTML – абзаца (<p></p>)(это важно!). Поле ID заполнять не нужно, так как для него был указан автоинкремент.

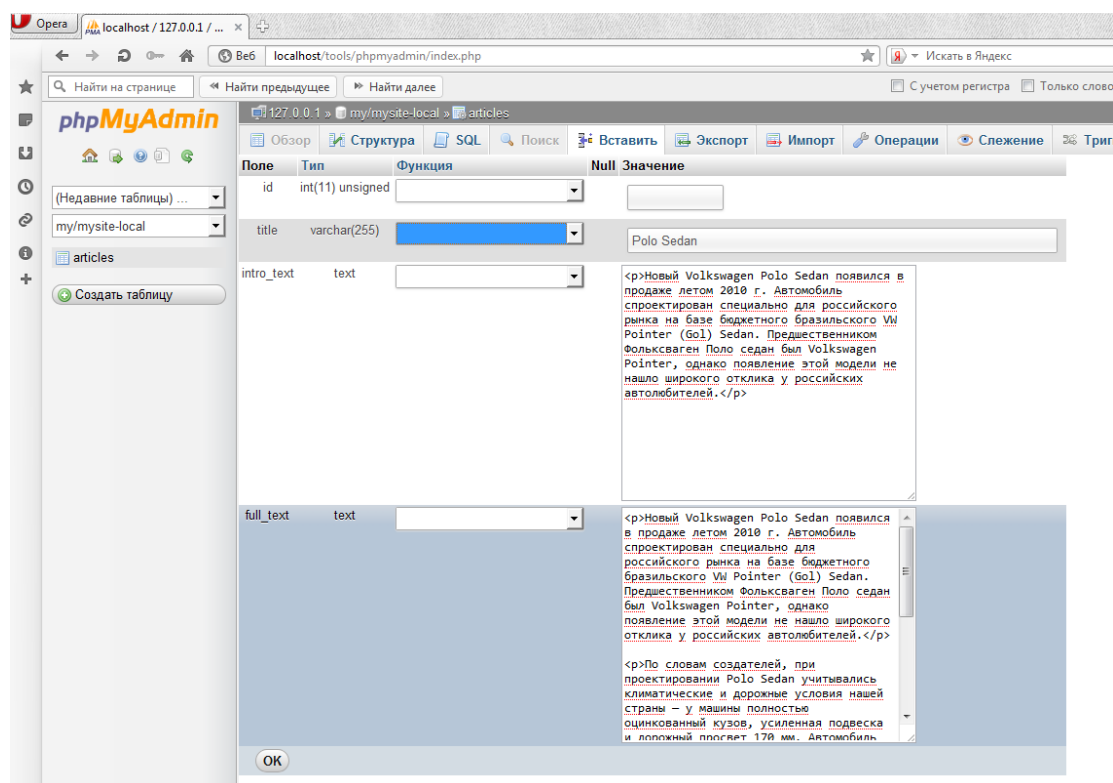


Рис.5. Добавление записи в таблицу articles.

После нажатия “ОК” запись будет добавлена.
Самостоятельно добавьте еще 3 записи (о Ford Focus, о Ford Mondeo, о Лада Калина) (рис.6).

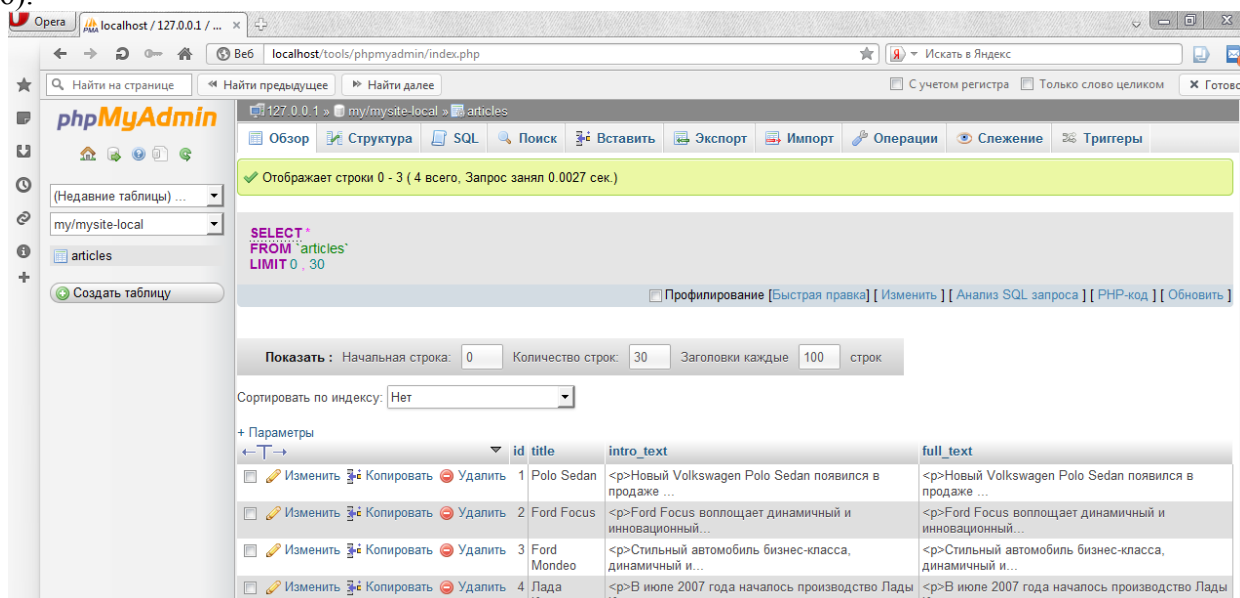


Рис.6. Заполненная таблица Articles

Все статьи хранятся в базе данных, что позволяет их изменять, добавлять новые, не затрагивая страниц сайта.

Теперь соединим наш сайт с базой данных.

Чтобы подключить БД надо открыть с ней соединение, а по окончании работы закрыть его.

Для этого в папке www создадим папку lib, а в ней файл functions.php.

Добавим в этот файл следующий код:

```
<?php
    $mysqli=false;
]   function connectDB(){
        global $mysqli;
        $mysqli =new mysqli("localhost", "root", "", "mydatabase");
        $mysqli->query("SET NAMES 'cp1251'");
        }

]   function closeDB() {
        global $mysqli;
        $mysqli->close();
        }
?>
```

Здесь для связи PHP и MYSQL используется объект(переменная) \$mysqli (i сокращение от improved). Эта переменная глобальная и служит основой работы практически для всех функций, которые будут созданы позже.

В строке \$mysqli =new mysqli("localhost", "root", "", "mydatabase") четыре параметра: имя Host, имя пользователя (root), пароль – отсутствует, имя БД.

В следующей строке указывается используемая кодировка.

Функция closeDB() пояснений не требует.

Задание на лабораторную работу

1. Создать базу данных для сайта об автомобилях, используя выше приведенный пример;
2. Создать базу данных для сайта, разрабатываемого самостоятельно из лабораторной работы №2.2
3. Подготовить отчет (титульный лист, структура базы данных, скриншоты).

3.2. Лабораторная работа №8. Создание динамических страниц сайта средствами PHP

Цель работы: ознакомиться со способами динамической генерации страниц пользователя с использованием PHP.

Продолжим рассмотрение примера по созданию сайта об автомобилях

Связь сайта с базой данных

Требуется, чтобы все статьи из базы данных подключались к сайту. Изменения коснутся в основном двух страниц папки: статьи, отдельная статья.

Займемся первой задачей.

Требуется вывести все статьи с краткой информацией об автомобилях из БД. Сейчас эту задачу решает файл blocks/articles.php

Разделим эту задачу на две:

1. Собственно вывод всех статей;
2. Шаблон вывода каждой статьи.

Начнем со второй. В папке blocks сохраним файл list_articles.php под именем intro_article.php. Он будет служить шаблоном для создания страницы с любой краткой информацией о конкретном автомобиле, полученной из базы данных.

В этом файле необходимо заменить конкретные сведения об автомобиле информацией о произвольном автомобиле из базы данных.

Получим следующий текст:

```
<div class="article">
    <h2><?php echo $title; ?> </h2>
    <p class="article_img">
        " />
    </p>
    <?php echo $intro_text; ?>
    <p>
        <a href="article.php?id=<?php echo $id; ?>"> Читать дальше </a>
    </p>
</div>
<hr />
```

Все конкретная информация об автомобиле заменена на переменные PHP. Подставляя разные значения переменных, будем получать разные страницы!

Оператор echo служит для вывода информации в браузер.

Теперь решим первую задачу: необходимо просмотреть и выбрать из БД все резюме об автомобилях и выводить их в браузер, используя шаблон intro_article.php.

Для этого заменим содержимое файла list_articles.php на следующий текст:

```
<?php
    $articles = getAllArticles();
    for ($i=0; $i<count($articles); $i++) {
        $id=$articles[$i]["id"];
        $title=$articles[$i]["title"];
        $intro_text=$articles[$i]["intro_text"];
        include "intro_article.php";
    }
?>
```

Здесь функция getAllArticles(); служит для получения таблицы(двумерного массива) со статьями из базы данных. Последующий цикл выбирает из этого массива значения переменных \$id, \$title, \$intro_text и вызывает шаблон intro_article.php

Создадим функцию `getAllArticles()` в файле `functions.php`. Она осуществит выборку всех записей из таблицы `articles` в виде массива указателей.

```
function getAllArticles() {
    global $mysqli;
    connectDB();
    $result_set=$mysqli->query("SELECT * FROM `articles`");
    closeDB();
    return resultSetToArray($result_set);
}
```

Обратите внимание на использование обратных апострофов.

Так как `$result_set` это массив указателей, то приведем его к виду двумерного массива с помощью функции `resultSetToArray()`.

```
function resultSetToArray($result_set) {
    $array=array();
    while (($row=$result_set->fetch_assoc()) != false)
        $array[]=$row;
    return $array;
}
```

Метод `fetch_assoc()` возвращает очередной указатель на следующую запись в массиве `$result_set`.

Чтобы все функции заработали надо подключить библиотеку ко всем файлам сайта. Для этого создадим новый файл в папке `www/start.php`.

```
<?php
    require_once "lib/functions.php";
?>
```

Его подключим к файлу `index.php` (рис.1).

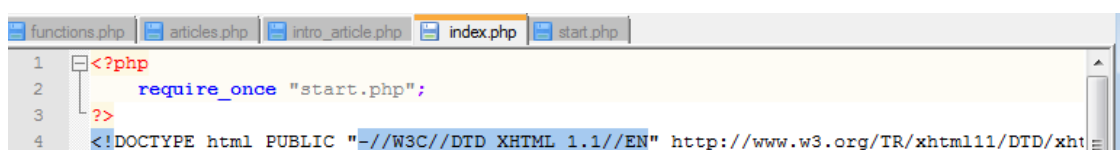


Рис.1. Подключение файла `start.php` к файлу `index.php`

Такое же подключение сделаем и на основных страницах папки `www` `articles.php` и `article.php`.

После обновления вид экрана браузера должен выглядеть как и до переделки (рис.2)

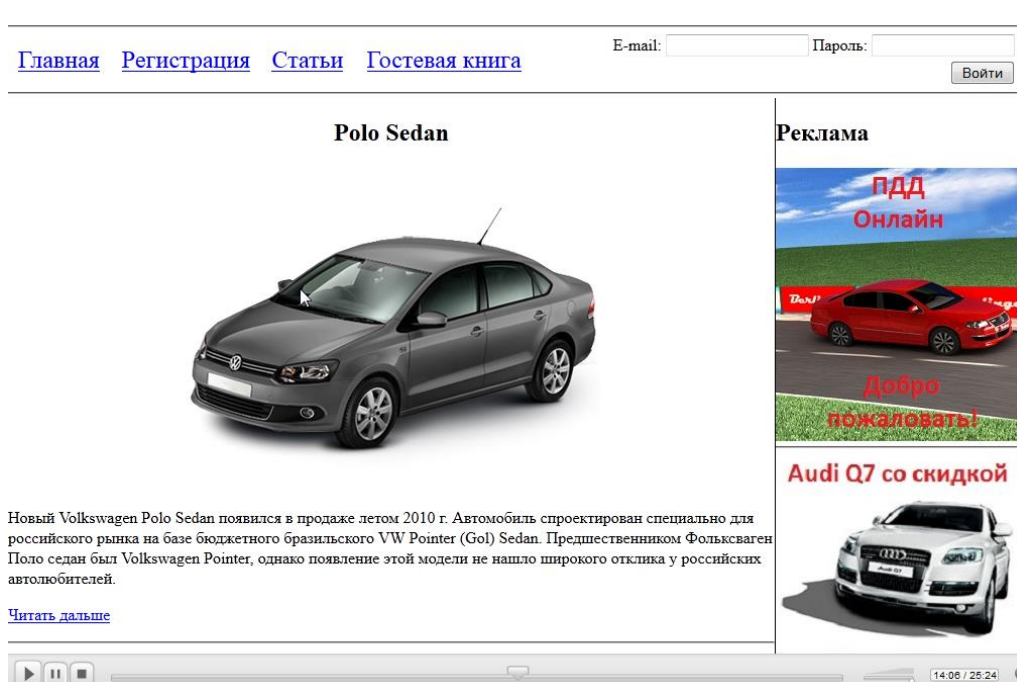


Рис.2. Динамически сформированная страница сайта об автомобилях

Доработка страницы с полным текстом статьи.

Теперь доработаем страницу с полным текстом статьи.

Откроем файл `www/article.php`. И исправим его в соответствии с рисунком 3.

```

1 <?php
2     require_once "start.php";
3     $article = getArticle($_GET["id"]);
4     $id = $article["id"];
5     $title = $article["title"];
6     $full_text = $article["full_text"];
7
8 >?
9 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/
10 <html xmlns="http://www.w3.org/1999/xhtml">
11 <head>
12     <title><?php echo $title;?></title>
13     <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
14     <link rel="stylesheet" type="text/css" href="styles/main.css" />
15 </head>

```

Рис. 3. Доработка файла `article.php`

Массив `$_GET` является глобальным и через него передаются значения переменных, запрошенных в `intro_article.php`.

Теперь в папке `blocks` откроем файл `full_article.php` и исправим его. Он практически повторяет файл `intro_article.php` (Рис.4).

```

1 <div class="article">
2     <h1><?php echo $title; ?></h1>
3     <p class="article_img">
4         " />
5     </p>
6     <?php echo $full_text; ?>
7
8 </div>

```

Рис.4. Исправление файла `full_article.php`

Займемся теперь функцией `getArticle()`.

```
function getArticle($id){
    global $mysqli;
    connectDB();
    $result_set=$mysqli->query("SELECT * FROM `articles` WHERE `id`='".$id'");
    closeDB();
    return $result_set->fetch_assoc();
}
```

Обратите внимание на использование апострофов!

Обновите окно браузера и Вы сможете получать полные тексты статей из базы данных.

Работа с баннерами.

Информация о баннерах также должна храниться в базе данных.

Создадим таблицу `banners` с двумя столбцами: `id` (с теми же значениями полей, что и поле `id` в таблице `articles`), `code` (с теми же значениями полей, что и поле `intro_text` в таблице `articles`).

Добавим в нее новые записи, тексты которых берем из файла `blocks/banners_240.php` (рисунок 5).

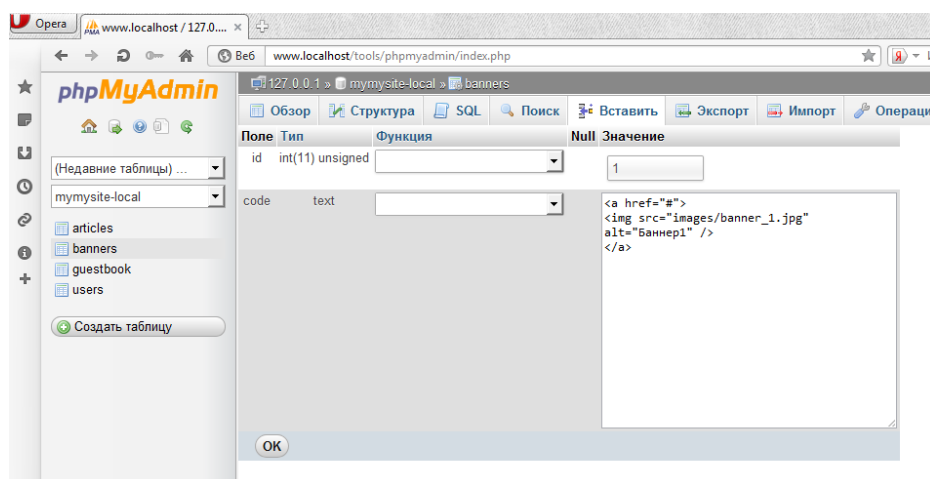


Рисунок 5. Добавление записей в таблицу `banners`.

В папке `blocks` создадим шаблон для вывода баннеров файл `banner.php`.

```
<div class="banner"><?php echo $code; ?></div>
<hr />
```

Теперь в файле `banners_240.php` организуем вывод баннеров из базы данных.

```
<h2>Реклама</h2>
<?php
    $banners=getAllBanners();
    for ($i=0; $i<count($banners); $i++){
        $code=$banners[$i]["code"];
        include "banner.php";
    }
?>
```

Осталось создать функцию `getAllBanners()`. Очевидно, что она будет аналогична функции `getAllArticles()`.

```
function getAllBanners() {  
    global $mysqli;  
    connectDB();  
    $result_set=$mysqli->query("SELECT * FROM `banners`");  
    closeDB();  
    return resultSetToArray($result_set);  
}
```

Задание на лабораторную работу

1. Подключить базу данных для сайта об автомобилях, используя выше приведенный пример;
2. Подключить базу данных для сайта, разрабатываемого самостоятельно из лабораторной работы №2.2
3. Подготовить отчет (титальный лист, структура базы данных, скриншоты).

3.3. Лабораторная работа №9. «Создание динамических страниц сайта средствами PHP. Реализация гостевой книги»

Цель работы показать средства взаимодействия сайта и базы данных на примере реализации гостевой книги.

Гостевая книга – это набор некоторых записей, каждая из которых содержит имя, посетившего сайт и содержание его отзыва.

Все записи должны храниться в базе данных.

С помощью СУБД phpMyAdmin создадим новую таблицу. Для входа в базу данных воспользуемся ссылкой localhost/tools/phpmyadmin. В базе данных mydatabase создадим новую таблицу с названием guestbook с числом полей 3 (id, name, comment). Атрибуты полей соответствуют атрибутам полей таблицы articles, созданной в предыдущей лабораторной работе (Рис.1).

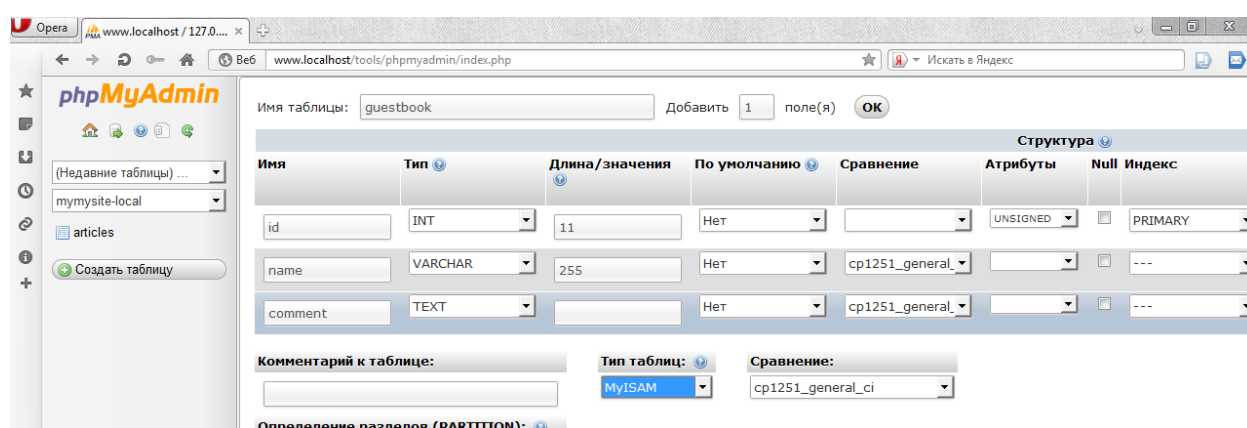


Рисунок 1. Структура таблицы guestbook

Теперь сделаем верстку страницы guestbook. Основой для нее послужит файл index.php.

Скопируем его и разместим в корневой папке нашего сайта под именем guestbook.php.

Поменяем тег <title> на Гостевая книга. Заменим вызов основной статьи на guestbook.php.



Рисунок 2. Подготовка файла гостевой книги.

Теперь в папке blocks создадим новый блок guestbook.php.

В нем разместим форму для ввода комментария гостя:

```

<n2> Добавить запись</n2>
<form name="guestbook" action="" method="post">
  <table>
    <tr>
      <td>Имя: </td>
      <td>
        <input type="text" name="name" />
      </td>
    </tr>
    <tr>
      <td>Комментарий: </td>
      <td>
        <input type="text" name="comment" />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="submit" name="button_guestbook" value="Добавить" />
      </td>
    </tr>
  </table>
</form>

```

Атрибут action имеет пустое значение, чтобы форма обрабатывалась в данном скрипте.

После обновления файлов в браузере подготовленная форма примет вид, как на рисунке 3.

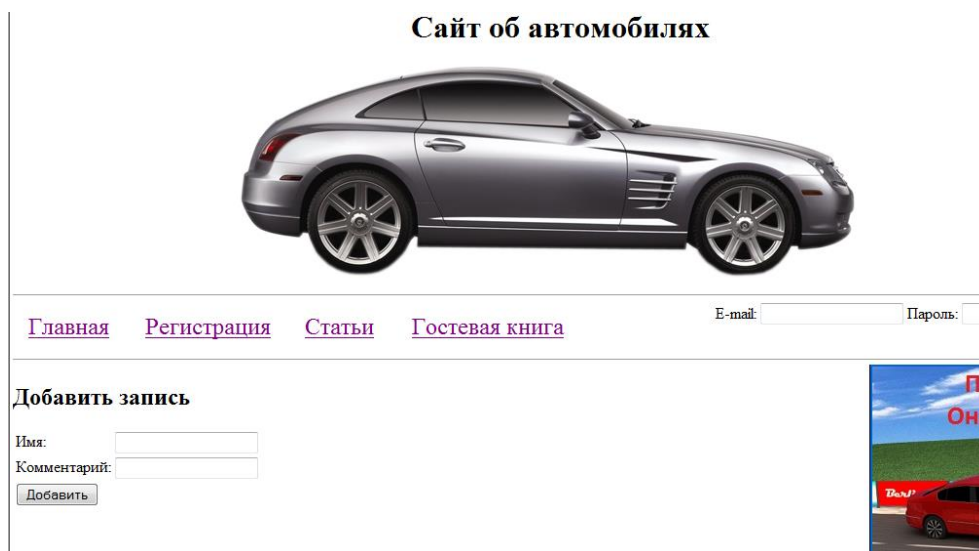


Рисунок 3. Форма для добавления записей в гостевую книгу

Теперь добавим к форме вывод уже имеющихся записей в Гостевой книге.

Для этого нам понадобится функция getAllGuestBookComments(), с помощью которой и можно выбрать все записи из базы данных.

Создадим ее.

Она во многом схожа с функцией getAllArticles(), меняется только название целевой таблицы.

```

    }
    function getAllArticles(){
        global $mysqli;
        connectDB();
        $result_set=$mysqli->query("SELECT * FROM `articles`");
        closeDB();
        return resultSetToArray($result_set);
    }

    function getAllGuestBookComments(){
        global $mysqli;
        connectDB();
        $result_set=$mysqli->query("SELECT * FROM `guestbook`");
        closeDB();
        return resultSetToArray($result_set);
    }

```

Вернемся к блоку guestbook.php. Добавим после вывода формы вывод всех уже имеющихся отзывов с помощью шаблона guestbook_comment.php.

```

</form>
<h2> Записи в гостевой книге </h2>
<div>
    <?php
        $comments=getAllGuestBookComments();
        for ($i=0;$i<count($comments);$i++){
            $name=$comments[$i]["name"];
            $comment=$comments[$i]["comment"];
            include "blocks/guestbook_comment.php";
        }
    ?>
</div>

```

Создадим теперь этот шаблон(Рисунок 4).

```

functions.php | articles.php | intro_article.php | index.php | guestbook.php | guestbook.php | guestbook_comment.php
1 <p><b><?php echo $name;?></b> <?php echo $comment;?></p>

```

Рисунок 4. Шаблон для вывода комментариев из гостевой книги.

Если обновить сайт, то увидим (Рисунок 5):

Добавить запись Имя: Комментарий: Добавить Записи в гостевой книге	Реклама 
--	---

Рисунок 5. Страница гостевой книги, пока без записей.

Теперь разберемся с процессом добавления записей в гостевую книгу.

Записи создаются и отправляются в базу данных с помощью формы файла guestbook.php. Поэтому добавим в существующий скрипт на PHP перед выводом записей из гостевой книги следующий фрагмент:

```

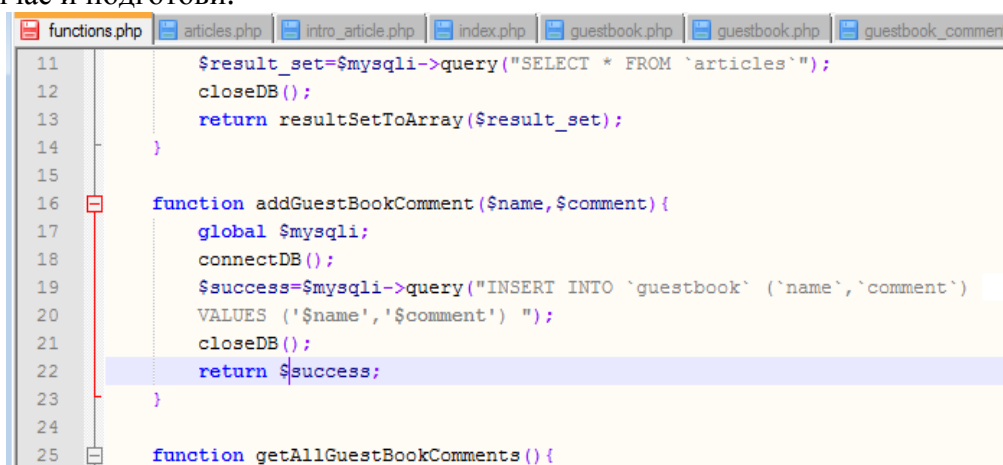
if (!empty($_POST["button_guestbook"])) {
    $name=htmlspecialchars($_POST["name"]);
    $comment=htmlspecialchars($_POST["comment"]);
    if ((strlen($name)<3) || (strlen($comment)<3)) $success=false;
    else $success=addGuestBookComment($name,$comment);
    if (!$success){
        $alert="Ошибка при добавлении новой записи!";
        include "alert.php";
    }
}

```

Этот фрагмент работает, когда будет послано сообщение с помощью кнопки «Добавить».

Функция htmlspecialchars() служит для ликвидации попыток ввести вредоносный код в базу данных. Здесь же проверяется правильность заполнения формы.

Собственно добавление записей осуществляет функция addGuestBookComment(), которую сейчас и подготови.



```

functions.php | articles.php | intro_article.php | index.php | guestbook.php | guestbook.php | guestbook_comment.php
11      $result_set=$mysqli->query("SELECT * FROM `articles`");
12      closeDB();
13      return resultSetToArray($result_set);
14  }
15
16  function addGuestBookComment($name,$comment){
17      global $mysqli;
18      connectDB();
19      $success=$mysqli->query("INSERT INTO `guestbook` (`name`,`comment`)
20      VALUES ('$name','$comment') ");
21      closeDB();
22      return $success;
23  }
24
25  function getAllGuestBookComments(){

```

Обратите внимание на правильное использование кавычек!

Функция alert.php выдаст сообщение, в случае возникновения ошибки. Создадим ее в папке blocks, используя javascript:

```

<script type="text/javascript">
    alert("<?php echo $alert; ?>");
</script>

```

После обновления сайта должна срабатывать проверка на наличие записей, и добавление правильных комментариев.

Задание на лабораторную работу

1. Подготовить гостевую книгу для сайта об автомобилях, используя выше приведенный пример;
2. Подготовить гостевую книгу для сайта, разрабатываемого самостоятельно из лабораторной работы №2.2
3. Подготовить отчет (титальный лист, структура базы данных, скриншоты).

3.4. Лабораторная работа №10. «Создание динамических страниц сайта средствами PHP. Регистрация и авторизация пользователей»

Цель работы: показать средства взаимодействия сайта и базы данных на примере регистрации и авторизации пользователей.

Информация о пользователях хранится в базе данных.

Поэтому регистрация пользователя это всего лишь добавление записи в базу данных.

Последовательность действий для добавления записей в базу данных была уже рассмотрена на примере создания записи в Гостевой книге(см. ЛР№7).

Итак, в корневой папке www создадим страницу reg.php.

В качестве шаблона возьмем страницу index.php и заменим основное содержание на reg.php(Рисунок 1).

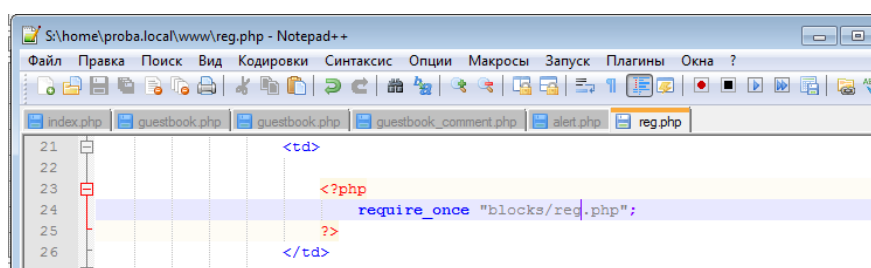


Рисунок 8. Создание файла для регистрации reg.php.

Теперь создадим новый блок blocks/reg.php

В нем разместим необходимую форму (Рисунок 2).

```

<h2> Регистрация </h2>
<form name="reg" action="" method="POST">
  <table>
    <tr>
      <td>E-mail:</td>
      <td>
        <input type="text" name="email" />
      </td>
    </tr>
    <tr>
      <td>Пароль:</td>
      <td>
        <input type="password" name="password_1" />
      </td>
    </tr>
    <tr>
      <td>Подтвердите пароль:</td>
      <td>
        <input type="password" name="password_2" />
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <input type="submit" name="button_reg" value="Зарегистрироваться" />
      </td>
    </tr>
  </table>
</form>

```

Рисунок 2. Форма регистрации.

Перед отправкой форму требуется обработать. Поместим в ее начало код:

```

<?php
if (!empty($_POST["button_reg"])) {

```

```

$email=htmlspecialchars($_POST["email"]);
$password_1=htmlspecialchars($_POST["password_1"]);
$password_2=htmlspecialchars($_POST["password_2"]);
if (strlen($email)<3) $success=false;
    elseif (strlen($password_1) <3) $success=false;
    elseif ($password_1 != $password_2) $success=false;
    else $success=addUser($email,md5($password_1));
if (!$success) $alert="Ошибка при регистрации пользователя!";
    else $alert="Вы успешно зарегистрировались!";
include "alert.php";

}

?>

```

Метод md5() позволяет хешировать пароль перед записью в базу данных.

Теперь в базе данных mydatabase (вызвать которую можно localhost/tools/phpmyadmin) создадим таблицу users, где и будет храниться информация о пользователях.

В таблице будет три поля: id, email, password (Рисунок 3).

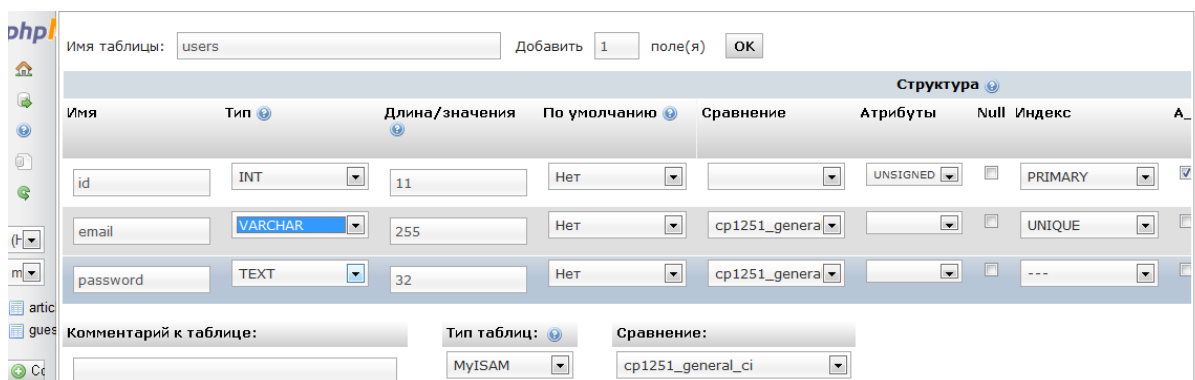


Рисунок 3. Структура таблицы users.

Осталось добавить функцию addUser().

Она во многом похожа на функцию addGuestBookComment(). Добавим ее в папку lib в файл functions.php.

```

function addUser($email,$password){
    global $mysqli;
    connectDB();
    $success=$mysqli->query("INSERT INTO `users` (`email`,`password`)
VALUES ('$email','$password') ");
    closeDB();
    return $success;
}

```

Если сделать обновление файлов браузера, то теперь можно выполнять регистрацию (рисунок 4).

[Главная](#) [Регистрация](#) [Статьи](#) [Гостевая книга](#)

Регистрация

E-mail:
 Пароль:
 Подтвердите пароль:

Рисунок4. Регистрация нового пользователя.

Теперь займемся авторизацией.

Сложность задачи заключается в том, чтобы пользователь после регистрации на одной странице оставался известным сайту и на других страницах.

Достигается это с помощью, так называемых сессий. В них хранится, например, Email и пароль после авторизации.

Модернизируем файл start.php. Это позволит на каждой странице начинать сессию.

```

1  <?php
2      session_start();
3      require_once "lib/functions.php";
4  ?>

```

Теперь создадим функцию, которая проверит, зарегистрирован пользователь или нет.

```

function checkUser($email,$password){
    global $mysqli;
    connectDB();
    $result_set=$mysqli->query("SELECT * FROM `users` WHERE `email`='$email' AND `password`='$password'");
    closeDB();
    if ($result_set->fetch_assoc()) return true;
    else return false;
}

```

Теперь создадим в корне сайта файл с именем auth.php. Он будет вызван после регистрации пользователя (см. ЛР№3).

```

<?php
    require_once "start.php";
    $email=htmlspecialchars($_POST["email"]);
    $password=htmlspecialchars($_POST["password"]);
    $password=md5($password);
    if (checkUser($email, $password)){
        $_SESSION["email"]=$email;
        $_SESSION["password"]=$password;
    }
    else $_SESSION["error_auth"]=1;
    header("Location: ".$_SERVER["HTTP_REFERER"]);
    exit;
?>

```

Переменная `$_SESSION` является глобальной для всех страниц сайта. Редирект с помощью функции `header()` позволяет вернуться на страницу, с которой выполнялась авторизация.

После авторизации пользователя можно убрать форму авторизации с экрана и заменить ее некоторым приветствием.

Откроем файл `blocks/top.php`. Сейчас его основным содержимым является форма авторизации (Рисунок 5).



Рисунок 5. Файл `top.php` перед внесением изменений.

Форма нужна, если пользователь не авторизован, и не нужна, если пользователь авторизовался. Поэтому извлечем форму авторизации в новый файл `auth_form.php`, а на ее месте разместим проверку – авторизован ли пользователь?

```

<?php
    if (checkUser($_SESSION["email"],$_SESSION["password"]))
        require_once "blocks/user_panel.php";
    else require_once "blocks/auth_form.php";
?>

```

В файле `auth_form.php` сделаем обработку ошибки, в случае неправильной авторизации пользователя (Рисунок 6).



Рисунок 6. Форма авторизации пользователя с предварительной проверкой авторизации пользователя.

Функция `unset()` сбрасывает признак ошибки, чтобы он не выводился повторно.

Если пользователь зарегистрирован, то в работу вступит файл user_panel.php. Добавим файл user_panel.php в папку blocks.

```
<p>Пользователь: <b><?php echo $_SESSION["email"];?> </b>
<br />
    <a href="logout.php">Выход</a>
</p>
```

Для выхода из сессии подготовим файл logout.php.

```
<?php
    session_start();
    unset($_SESSION["email"]);
    unset($_SESSION["password"]);
    header("Location: ".$_SERVER["HTTP_REFERER"]);
    exit;
?>
```

Задание на лабораторную работу

1. Подготовить средства для регистрации и авторизации пользователя на сайте об автомобилях, используя выше приведенный пример;
2. Подготовить средства для регистрации и авторизации пользователя на сайте, разрабатываемого самостоятельно из лабораторной работы №2.2
3. Подготовить отчет (титальный лист, структура базы данных, скриншоты).

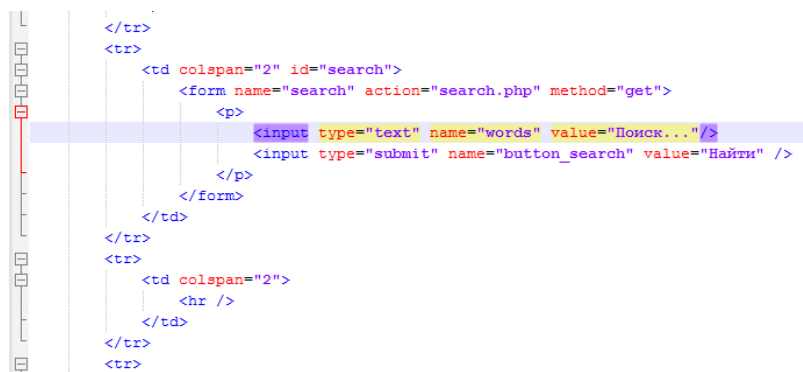
3.5. Лабораторная работа №11. «Создание динамических страниц сайта средствами РНР. Организация поиска на сайте»

Цель работы: ознакомиться с методами организации поиска информации на сайте.

Попробуем организовать поиск информации по статьям, занесенных в базу данных.

Для начала создадим форму поиска. Поместить ее можно, например, над строкой главного меню и формы авторизации.

Откройте в блокноте файл blocks/top.php и внесите в него следующие изменения (рисунок 1):



Риснок 1. Добавление формы поиска в файл top.php.

Как видно из рисунка 1, в файл top.php добавлено 2 строки: первая содержит форму, вторая – горизонтальную линию.

В форме использован метод get – он более предпочтителен, чем метод Post для организации поиска.

После обновления файлов (Не забудьте добавить идентификатор #search в файл стилей main.css для выравнивания текста в ячейке вправо!) в окне браузера получим следующий результат(рисунок 2):

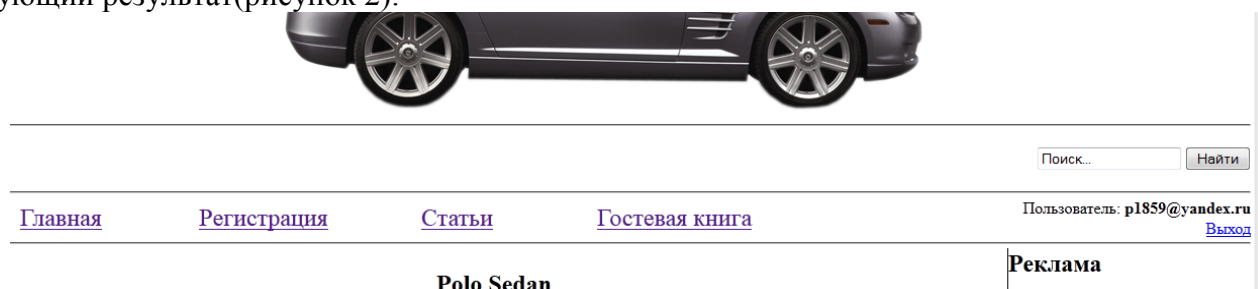


Рисунок 2. Вид окна браузера после обновления top.php.

При вводе слов для поиска содержимое поля по умолчанию «Поиск...» придется стирать. Чтобы это происходило автоматически, добавим в форму небольшой скрипт на Javascript.

```

<input type="text" name="words" value="Поиск..."
  onfocus="if(this.value=='Поиск...') this.value='';"
  onblur="if(this.value=='') this.value='Поиск...';"/>

```

Ну и последнее, что можно сделать с внешним видом формы – убрать отступы. Для этого в файл стилей включите:

```

#search form, #search p {
  margin:0;
}

```

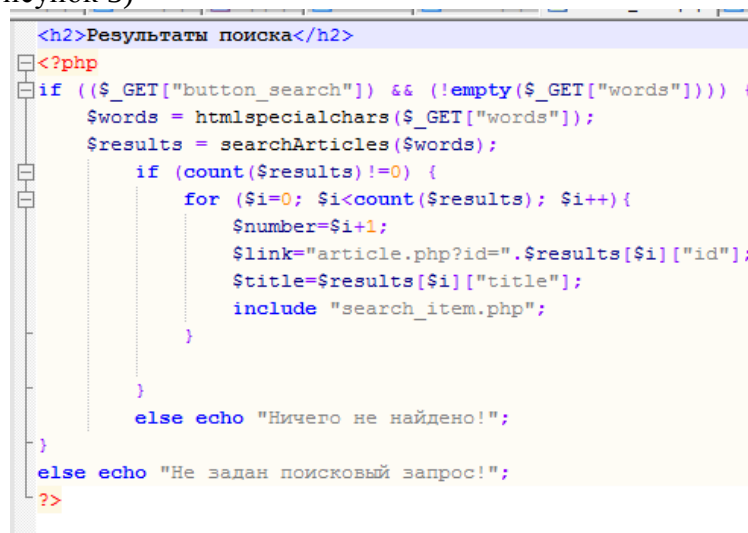
Разработка страницы с результатами поиска.

Теперь создадим в корне сайта страницу с результатами поиска search.php. Основой для нее послужит страница index.php.

В ней заменяем заголовок «Сайт об автомобилях» на «Результаты поиска», а также

Вместо блока main_article подключим блок blocks/search_result.php, который пока не создан.

Создадим его(рисунок 3)



```
<h2>Результаты поиска</h2>
<?php
if (($GET["button_search"]) && (!empty($GET["words"]))) {
    $words = htmlspecialchars($GET["words"]);
    $results = searchArticles($words);
    if (count($results)!=0) {
        for ($i=0; $i<count($results); $i++){
            $number=$i+1;
            $link="article.php?id=".$results[$i]["id"];
            $title=$results[$i]["title"];
            include "search_item.php";
        }
    }
    else echo "Ничего не найдено!";
}
else echo "Не задан поисковый запрос!";
?>
```

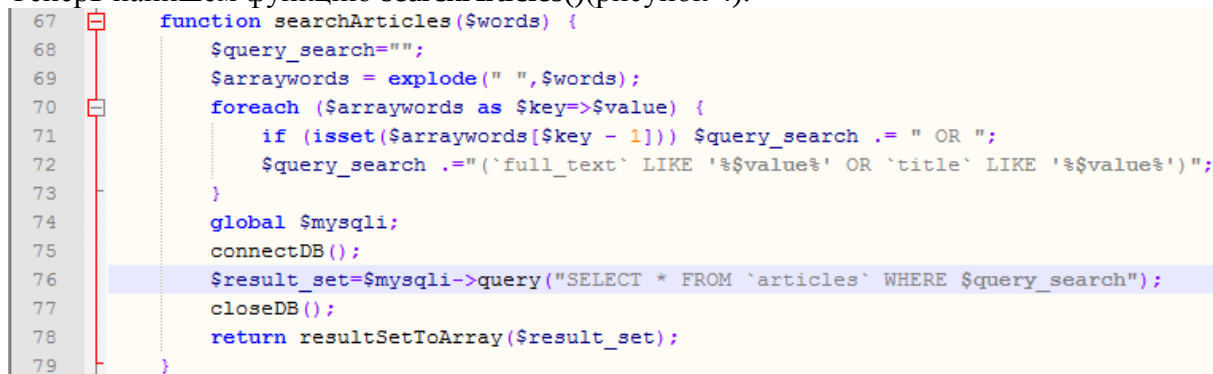
Рисунок 3. Файл search_result.php

Работа этого блока сводится к следующему: Если нажата кнопка поиска и список слов для поиска непустой, то функция searchArticles(\$words) массив статей, в которых обнаружены искомые слова. Они будут выведены на экран с помощью шаблона search_item.php.

Этот шаблон может иметь, например, следующий вид:

```
<p><?php echo $number;?>. <a href="<?php echo $link;?>"><?php echo $title;?></a></p>
```

Теперь напомним функцию searchArticles()(рисунок 4).



```
67 function searchArticles($words) {
68     $query_search="";
69     $arraywords = explode(" ", $words);
70     foreach ($arraywords as $key=>$value) {
71         if (isset($arraywords[$key - 1])) $query_search .= " OR ";
72         $query_search .="('full_text' LIKE '%$value%' OR 'title' LIKE '%$value%')";
73     }
74     global $mysqli;
75     connectDB();
76     $result_set=$mysqli->query("SELECT * FROM `articles` WHERE $query_search");
77     closeDB();
78     return resultSetToArray($result_set);
79 }
```

Рисунок 4. Функция searchArticles(\$words)

Пояснения к рисунку 4:

- В строке 68 подготавливается переменная, в которой будет сформирована основная часть запроса на MySQL;
- В строке 69 использована функция explode(), позволяющая разбить исходную строку на отдельные слова (аналогична методу split в Java);
- В строке 71 обеспечивается добавление союза OR только между конструкциями запроса;

- В строке 72 LIKE обеспечивает поиск нужного слова в таблице. Символы процента допускают существование любых символов до и после искомого слова.

Задание на лабораторную работу

1. Организовать поиск информации по базе данных сайта об автомобилях, используя выше приведенный пример;
2. Организовать поиск информации по базе данных сайта, разрабатываемого самостоятельно из лабораторной работы №2.2.
3. Подготовить отчет (титальный лист, структура базы данных, скриншоты).

3.6. Лабораторная работа №12. «Создание динамических страниц сайта средствами PHP. Доступ администратора на сайт. Часть 1»

Цель работы: ознакомиться с методами управления сайтом.

Управление сайтом предполагает возможность редактирования его содержимого лицами, ответственными за его состояние и развитие. Назовем их «Администраторами».

Для того, чтобы администратор мог попасть на сайт и выполнить все необходимые работы, он должен иметь особый статус по отношению к другим пользователям.

Поэтому первое, что можно сделать, разнести зарегистрированных пользователей на две категории: обычные пользователи и пользователи с правами администратора. Это потребует изменения таблицы «USERS» базы данных mydatabase.

Добавим поле admin в таблицу users.

Добавить столбцы

Структура

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты
admin	TINYINT	1	Как определен 0		UNSIGNED

Сохранить

После нажатия кнопки сохранить структура базы данных будет выглядеть как на рисунке

	id	email	password	admin
☐ Изменить ☐ Копировать ☐ Удалить	3	ttt@yandex.ru	81dc9bdb52d04dc20036dbd8313ed055	0
☐ Изменить ☐ Копировать ☐ Удалить	4	qqqq	3bad6af0fa4b8b330d162e19938ee981	0
☐ Изменить ☐ Копировать ☐ Удалить	5	1111	b59c67bf196a4758191e42f76670ceba	0

Сейчас можно зарегистрировать нового пользователя или просто исправить значение поля admin существующих записей на единицу.

Теперь для опознавания пользователя как администратора необходимо сделать запрос к базе данных. Оформим его в виде функции isAdmin().

```
function isAdmin($email) {
    global $mysqli;
    connectDB();
    $result_set = $mysqli->query("SELECT * FROM `users` WHERE `email`='$email'");
    $row = $result_set->fetch_assoc();
    closeDB();
    return $row["admin"];
}
```

Можно создать отдельную папку, например, с именем "admin" и в ней расположить отдельный файл index.php.

```

<?php
    require_once "start.php";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
<html>
<head>
    <title> Admin - панель </title>
    <meta http-equiv="Content-type" content="text/html; charset=1251" />
    <link rel="stylesheet" type="text/css" href="styles/main.css" />
</head>
<body>

</body>
</html>

```

Изменим файл start.php для проверки возможности доступа пользователя к средствам администратора.

```

<?php
session_start();
require_once "../lib/functions.php";
if (!((checkUser($_SESSION["email"],$_SESSION["password"]) && isAdmin($_SESSION["email"])))){
    header("Location: /admin/auth.php");
    exit;
}

?>
<?php error_reporting(E_ALL ^ E_NOTICE); ?>

```

Если пользователь еще не авторизовался как администратор, то создадим файл авторизации auth.php.

За основу возьмем файл index.php.

В его начале осуществляется проверка на повторную авторизацию пользователя.

Также добавлена форма для авторизации.

```

<?php
session_start();
require_once "../lib/functions.php";

if ((checkUser($_SESSION["email"],$_SESSION["password"]) && isAdmin($_SESSION["email"]))) {
    header("Location: /admin/index.php");
    exit;}

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
<html>
<head>
    <title> Вход в Admin - панель </title>
    <meta http-equiv="Content-type" content="text/html; charset=1251" />
    <link rel="stylesheet" type="text/css" href="styles/main.css" />
</head>
<body>
    <div>
        <!-- Вставка -->
        <form name="auth" action="../auth.php" method="post">
            <p>
                E-mail:<br/>
                <input type="text" name="email" />
            </p>
            <p>
                Пароль:<br/>
                <input type="password" name="password" />
            </p>
            <p>
                <input type="submit" name="button_auth" value="Войти" />
            </p>
        </form>
    </div>
</body>
</html>

```

Чтобы при вводе неправильных данных об администраторе сайт сообщал об этом добавим в позицию <Вставка1> следующий код:

```
<?php
    if ($_SESSION["error_auth"]){
        echo "Неверное имя или пароль!";
        unset($_SESSION["error_auth"]);
    }
    else if ($_SESSION["email"]!=""){
        echo "У Вас нет прав администратора!";
        unset($_SESSION["email"]);
    }
?>
```

Теперь добавим в Admin – панель средства, например, для добавления новой статьи. В файл index.php в его тело вставим, например, следующий текст:

```
<body>
    <h1 id="header"> Admin -панель</h1>
    <h2> Меню</h2>
    <p>
        <a href="#">Добавить новую статью</a>
    </p>
</body>
```

После возможности добавления новой статьи необходимо предоставить возможность выхода из режима администратора. Для этого воспользуемся функцией logout.php.

```
<p>
    <a href="../logout.php">Выход</a>
</p>
/body>
```

Задание на лабораторную работу

1. Организовать доступ к сайту с правами администратора, используя выше приведенный пример;
2. Организовать доступ к сайту с правами администратора, разрабатываемого самостоятельно из лабораторной работы №2.2
3. Подготовить отчет (титальный лист, структура базы данных, скриншоты).

3.7. Лабораторная работа №12. «Создание динамических страниц сайта средствами PHP. Доступ администратора на сайт. Часть 2.»

Цель работы: продолжить знакомство с методами управления сайтом администратором.

В предыдущей работе была создана страница администратора с возможностью добавления новой статьи в базу данных. Реализуем эту возможность.

Для упрощения кода вынесем пункты меню Admin – панели в отдельный блок. Для этого в папке admin создадим папку blocks и в нее поместим файл menu.php:

```
<h1 id="header"> Admin -панель</h1>
<h2> Меню</h2>
<p>
    <a href="newarticle.php">Добавить новую статью</a>
</p>
<p>
    <a href=" ../logout.php">Выход</a>
</p>
```

Соответственно, файл index.php примет вид:

```
<?php
    require_once "start.php";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
<html>
<head>
    <title> Admin - панель </title>
    <meta http-equiv="Content-type" content="text/html; charset=1251" />
    <link rel="stylesheet" type="text/css" href="styles/main.css" />
</head>
<body>
    <?php require_once "blocks/menu.php"; ?>
</body>
</html>
```

На основе этого файла создадим файл newarticle.php.

```
<?php
    require_once "start.php";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd>
<html>
<head>
    <title> Добавление новой статьи </title>
    <meta http-equiv="Content-type" content="text/html; charset=1251" />
    <link rel="stylesheet" type="text/css" href="styles/main.css" />
</head>
<body>
    <?php require_once "blocks/menu.php"; ?>
    <!-- Вставка формы -->
</body>
</html>
```

Теперь создадим форму для ввода текста новой статьи и вставим ее в файл newarticle.php.

```

<div>
  <form name="newarticle" method="post" action="">
    <p>
      Название:<input type="text" name="title"/>
    </p>
    <div>
      Вступление:<br/>
      <textarea name="intro_text" cols="40" rows="10"></textarea>
    </div>
    <div>
      Полный текст:<br/>
      <textarea name="full_text" cols="40" rows="10"></textarea>
    </div>
    <p>
      <input type="submit" name="button_newarticle" value="Добавить статью"/>
    </p>
  </form>
</div>

```

После отправки формы она обрабатывается в этом же файле. Для этого добавим в заголовочную часть дополнительные операторы на PHP.

```

<?php
  require_once "start.php";
  if (isset($_POST["button_newarticle"]))
    $success=addArticle($_POST["title"],$_POST["intro_text"],$_POST["full_text"]);
?>

```

Функция addArticle и произведет добавление статьи в базу данных. Прежде чем ее написать добавим в файл newarticle.php реакцию на успешную или не успешную попытку добавления:

```

<?php require_once "blocks/menu.php";
if ($success) echo "Статья успешно добавлена.";
else if (isset($_POST["button_newarticle"])) echo "Ошибка!";
?>

```

Теперь следует создать функцию addArticle. За основу можно взять, например, функцию addUsers.

```

function addArticle($title,$intro_text,$full_text){
  global $mysqli;
  connectDB();
  $success=$mysqli->query("INSERT INTO `articles` (`title`,`intro_text`,`full_text`)
VALUES ('$title','$intro_text','$full_text') ");
  closeDB();
  return $success;
}

```

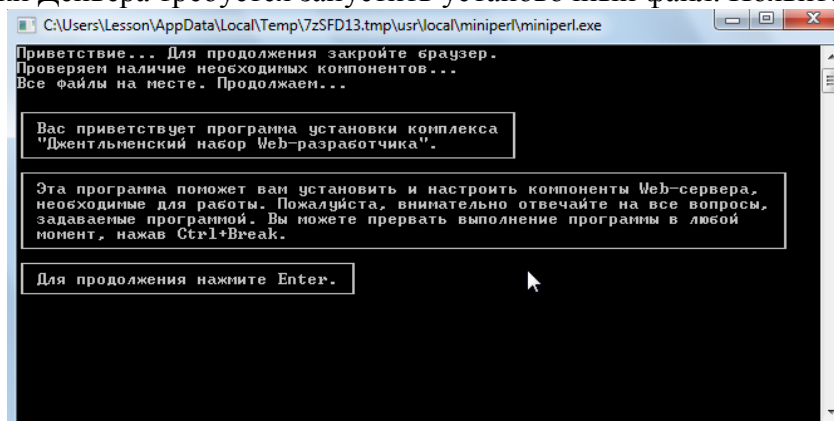
Проверка работы сайта показывает возможность добавления новых статей.

Задание на лабораторную работу

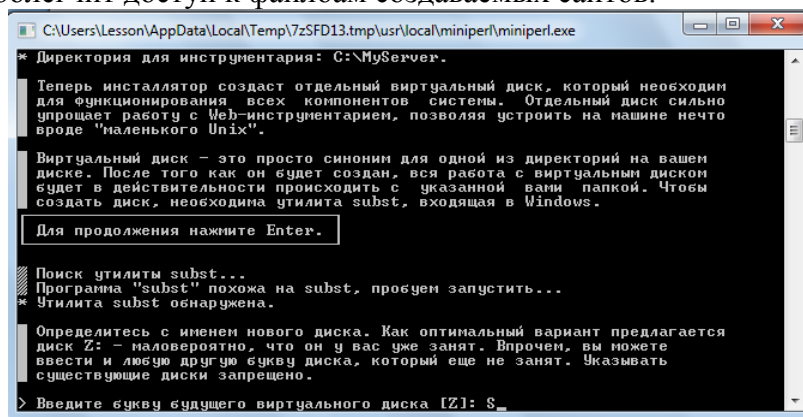
1. Организовать доступ к сайту с правами администратора, используя выше приведенный пример;
2. Организовать доступ к сайту с правами администратора, разрабатываемого самостоятельно из лабораторной работы №2.2
3. Подготовить отчет (титульный лист, структура базы данных, скриншоты).

Приложение 1. Установка Denwer

Denwer включает: сервер Apache; интерпретатор PHP; сервер доступа к MySQL. Для установки Денвера требуется запустить установочный файл. Появится окно:

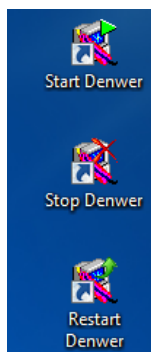


Далее после нажатия Enter и ввода директории для установки, Denwer запросит директорию для виртуального диска. Виртуальный диск – это синоном папки установки, что в дальнейшем облегчит доступ к файлоам создаваемых сайтов.



Далее после нажатия Enter, будет запрос о выборе одного из двух вариантов. Лучше выбрать второй.

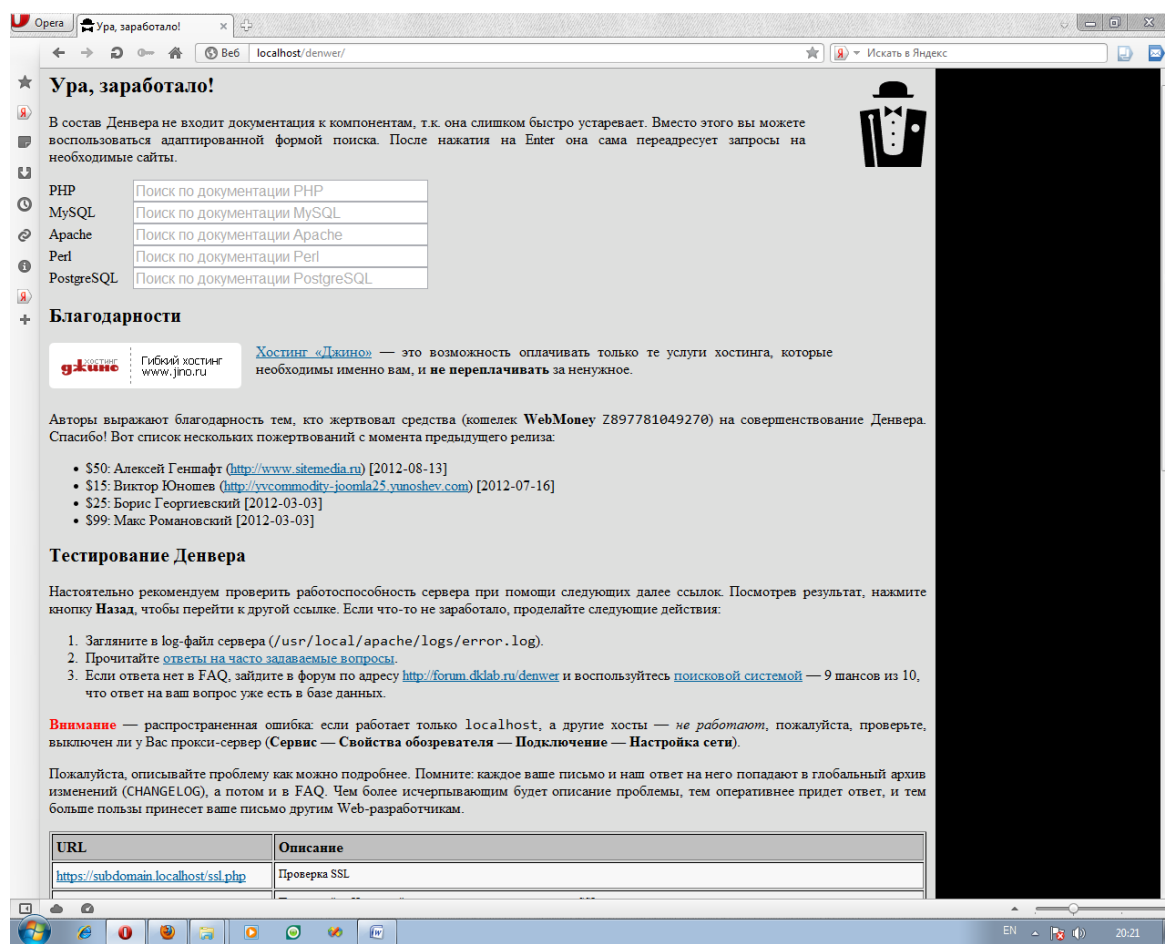
После окончания загрузки на рабочем столе появятся три иконки для запуска, остановки и перезапуска Denwer.



После запуска Denwer в папке Мой компьютер появится виртуальный диск для работы с файлами Denwer.

Для проверки работы сервера следует открыть браузер и в адресной строке ввести Localhost.

Появится визитное окно Denwer.



Основными причинами неудачного старта Denwer являются запущенные программы, занимающие порт 80, например Skype. Нужно их перенастроить или просто отключить.

Для продолжения работы требуется создать виртуальный host. Для этого надо зайти в папку home на виртуальном диске и в ней создать папку с именем hosta, например proba.local. Внутри этой папки требуется создать папку с именем www. Для подтверждения всех изменений перезапустите Denwer (Restart Denwer).

В адресной строке браузера введите *proba.local*. Если появилось такое окно в браузере, значит все сделано верно.



После подключения Denwer можно приступить к проектированию сайта. Темой пусть будет продажа автомобилей.

Для начала создайте в папке www файл «.htaccess». Он не имеет имени, а есть только расширение. Поэтому для его создания требуется воспользоваться, например Блокнотом.

Запишем в его начало требуемую кодировку страниц, например:

AddDefaultCharset Win1251

Данная кодировка подразумевает использование только символов английского и русского алфавитов.

