

Оглавление

Введение	4
Лабораторная работа №1: Установка набора дистрибутивов Denwer	6
Лабораторная работа №2: Создание статического каркаса сайта. Работа с инструментарием среды разработки Adobe Dreamweaver	10
Лабораторная работа №3: Создание базы данных MySQL	11
Лабораторная работа №4: Простой вывод данных. Страницы blog.php и comments.php.	20
Лабораторная работа №5: Ввод и правка данных с помощью формы.....	24
Лабораторная работа №6: Работа с заметками	33
Лабораторная работа №7: Страница статистики inform.php	35
Лабораторная работа №8: Реализация поиска по сайту	42
Лабораторная работа №9: Передача файлов на сервер	47
Лабораторная работа №10: Автоматизация работы средствами инструментальной среды Adobe Dreamweaver. Разграничение доступа к разделам сайта	53
Литература	67
Приложение 1. Схема сайта «MyTravelNotes»	68
Приложение 2. Схема базы данных «MySiteDB»	69
Приложение 3. Структура таблиц базы данных	70
Приложение 4. Основные сведения о работе с базой данных	72

Лабораторная работа №1: Установка набора дистрибутивов Denwer

В данной лабораторной работе рассматриваются установка набора дистрибутивов, необходимых для разработки серверных приложений с помощью языка программирования PHP и настройка инструментальной среды Adobe Dreamweaver на работу с виртуальным сервером.

Упражнение 1: Установка набора дистрибутивов Denwer

В данном упражнении будет продемонстрирована установка набора дистрибутивов, в состав которого входят дистрибутивы Apache, PHP, MySQL, phpMyAdmin и других систем и приложений, необходимых для организации разработки серверных приложений с использованием средств языка программирования PHP.

1. Скачайте дистрибутив базового комплекта Denwer - Denwer_base (около 2Мб), самую последнюю версию комплекта Denwer можно взять с сайта - www.denwer.ru/dis/Base.
2. Запустите инсталлятор. Вначале архив будет автоматически распакован во временную директорию, а затем автоматически запустится инсталлятор.
3. По умолчанию для установки комплекса используется директория C:\WebServers, нажмите Enter, чтобы согласиться с этим выбором. В указанном каталоге будут расположены абсолютно все компоненты системы, и вне его никакие файлы в дальнейшем не создаются (исключая ярлыки на Рабочем столе).
4. Далее вам предложат ввести имя виртуального диска, который будет связан с только что указанной директорией. Рекомендуется согласиться со значением по умолчанию (Z:). Важно, что диска с этим именем еще не должно содержаться в системе — чаще всего так и происходит с диском Z:.
5. После этого начнется копирование файлов дистрибутива. В ходе установки вам будет предложена установка ярлыков на Рабочий стол – это необходимо сделать для дальнейшего удобства работы.
6. В конце установки будет задан вопрос, как именно вы собираетесь запускать и останавливать комплекс (2 варианта):

- Автоматически создавать виртуальный диск при загрузке машины (а при остановке серверов этот диск не отключать);
 - Создавать виртуальный диск только по явной команде старта комплекса (при щелчке по ярлыку запуска на Рабочем столе). И, соответственно, отключать диск от системы — при остановке серверов.
- Рекомендуемый вариант – второй.**

7. Установка завершена. На Рабочем столе Windows дважды щелкните на ярлычке Start Denwer (если вы не создавали ярлыки, то можно запустить Денвер по команде C:\WebServers\denwer\Run.exe).
8. Дождавшись, когда все консольные окна исчезнут, откройте браузер и наберите в нем адрес: <http://localhost>. В случае успешной установки откроется страница, оповещающая о этом (см. Рис. 1.1).

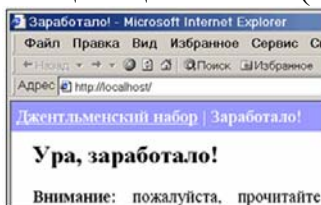


Рис.1.1. Окно страницы <http://localhost>

9. Изучите информацию страницы <http://localhost/>. Обратите особое внимание на разделы посвященные тестированию и работе с имеющимися утилитами.

Упражнение 2: Настройка Adobe Dreamweaver на работу с сервером Apache

В данном упражнении будет продемонстрирована настройка инструментальной среды разработки Adobe Dreamweaver для работы с виртуальным сервером Apache.

1. Запустите программу *Adobe Dreamweaver*.
2. Выберите пункты меню **Веб-сайт - Управление веб-сайтами**. Откроется диалоговое окно «Управление сайтами». В нем нажмите кнопку «Создать». Откроется окно настройки сайта. На вкладке **Веб-сайт** окна настройки сайта введите:
 - имя сайта: «*MyTravelNotes*»;
 - укажите путь к локальной папке его расположения: `C:\WebServers\home\localhost\www\`

3. В том же окне перейдите на вкладку **Серверы**, где необходимо задать параметры виртуального сервера. Для этого нажмите «+». Откроется окно настроек сервера.
4. На вкладке **Базовый** укажите:
 - Имя сервера: *localhost*
 - Подключение с помощью: *Локальный / Сетевой*
 - Папка сервера: *C:\WebServers\home\localhost\www*
 - URL-адрес: *http://localhost*
5. В том же окне перейдите на вкладку **Дополнительно**:
 - В разделе **Удаленный сервер** установите галочку напротив пункта «*Сохранить сведения о синхронизации*».
 - В разделе **Тестовый сервер** из выпадающего меню **Серверная модель** выберите *PHP MySQL*.
6. Сохраните параметры. Обратите внимание, что в окне «Настройка сайта» на вкладке **Серверы** должны стоять два флага для localhost: **Удаленный** и **Тестовый** (если их нет - поставьте их самостоятельно).
7. В окне **Управление сайтами** появится имя созданного проекта сайта «*MyTravelNotes*». Нажмите кнопку **Готово**. Для корректировки, удаления и других операций с проектом сайта используйте соответствующие кнопки управления в данном окне (рис. 1.2).

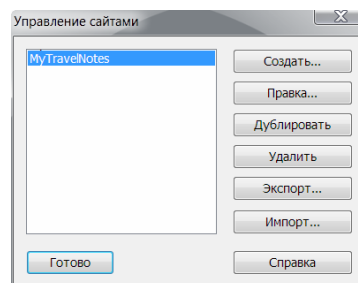


Рис. 1.2. Окно управления проектом сайта.

Упражнение 3: Первая программа на PHP

В этом упражнении Вы напишите программу “Hello, world!” на языке PHP, используя среду разработки Adobe Dreamweaver.

1. Создайте в программе Adobe Dreamweaver новую страницу (**Файл – Создать**. Выберите тип страницы PHP).
2. Введите следующий PHP-код:

```
<BODY>
<?PHP
echo "Hello, world!";
?>
</BODY>
```

3. Сохраните сценарий в папку сайта под именем hello.php.
4. Проверьте результат в браузере (F12).

При просмотре результата выполнения файла hello.php в браузере просмотрите html-код (например, в IE меню **Вид – Просмотр HTML-кода**). Обратите внимание на то, что php-кода на странице нет – это значит, что php-сценарий был обработан сервером, после чего сервер передал в браузер результат обработанного php-сценария.

Упражнение 4: Простейшие программы на PHP

Далее представлены дополнительные упражнения для закрепления навыков создания простейших программ в среде Adobe Dreamweaver на языке программирования PHP. Для их выполнения в Adobe Dreamweaver создайте новый .php файл (**Файл – Создать – Выбрать тип файла .php**) **examples.php** и выполните предложенные далее задания.

1. Переменной \$a необходимо присвоить значение 10, переменной \$b присвоить значение 20. Выведите значения переменных на экран.
2. Затем переменной \$c присвойте значение суммы этих переменных (переменной \$a и переменной \$b). Выведите значение переменной \$c на экран.
3. Далее увеличьте значение переменной \$c в три раза и выведите полученный результат на экран.
4. Разделите переменную \$c на разность переменных \$b и \$a, выведите результат на экран.
5. Введите новые переменные \$p и \$b. Присвойте переменной \$p значение «Программа», а переменной \$b значение «работает».
6. Затем сложите переменные, содержащие эти слова («Программа» и «работает»), при этом слова должны быть разделены пробелом (' '). Результат необходимо присвоить переменной \$result.
7. Далее с помощью оператора «.=» необходимо к строке «Программа работает» добавит слово «хорошо». Результат необходимо присвоить переменной \$result.
8. Есть две переменные: \$q = 5 и \$w = 7. Создайте скрипт, в результате выполнения которого эти две переменные «обмениваются» значениями –

переменная \$q получает значение 7, переменная \$w получает значение 5, при этом не создавая новых переменных (вариант \$q = 7 и \$w = 5 не рассматривается).

Лабораторная работа №2: Создание статического каркаса сайта. Работа с инструментарием среды разработки Adobe Dreamweaver

В данной лабораторной работе иллюстрируется создание двух первых страниц сайта – **blog.html** и **inform.html** в статическом виде с помощью средств разработки инструментальной среды Adobe Dreamweaver. Страница **blog.html** является первой страницей сайта, должна загружаться в браузере и содержать собственно заметки автора блога. Страница статистики **inform.html** будет вспомогательной страницей, содержащей статистическую информацию о размещенных на сайте заметках и комментариях.

Упражнение 1: Настройка Adobe Dreamweaver

В данном упражнении продемонстрирован процесс создания web-проекта в инструментальной среде Adobe Dreamweaver.

1. Запустите Denwer.
2. Запустите Adobe Dreamweaver, откройте многофункциональное окно **Настройки (Правка – Настройки)**, категория **Создать документ**. Установите тип документа по умолчанию (DTD) – HTML5 и кодировку по умолчанию – кириллица (Windows).

Упражнение 2: Создание статической основы web-страниц

В данном упражнении необходимо создать две первые статические страницы проекта, которые станут основой для дальнейшей разработки. Страница **blog.html** является первой страницей сайта, должна загружаться в браузере и содержать собственно заметки автора блога. Страница статистики **inform.html** будет вспомогательной страницей, содержащей статистическую информацию о размещенных на сайте заметках и комментариях.

1. В Dreamweaver перейдите в меню **Файл – Создать**.
2. Создайте статичный документ в формате HTML для будущей страницы с названием **blog.html**. На странице должно располагаться меню переходов

между страницами и место размещения основного контента сайта (рис.2.1):

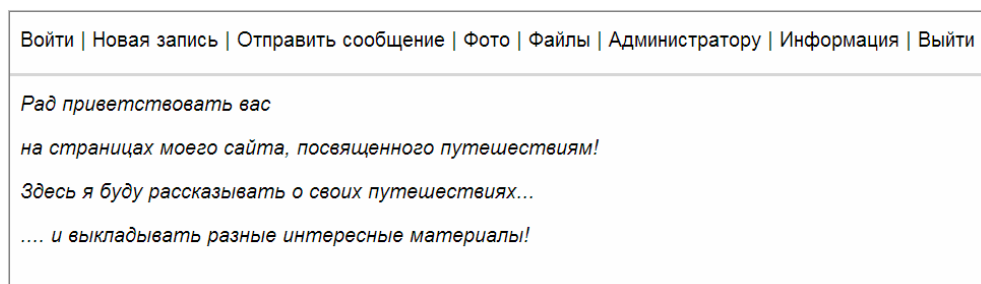


Рис. 2.1. Пример страницы blog.html

3. Сохраните страницу под именем blog.html.
4. Создайте страницу статистики inform.html. Схема страницы (рис. 2.2):

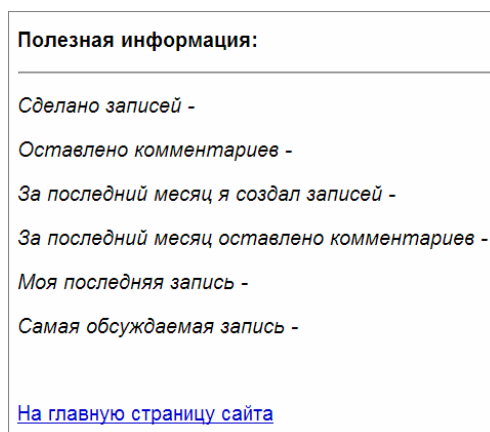


Рис. 2.2. Пример страницы inform.html

5. Сохраните таблицу под именем inform.html.
6. Свяжите гиперссылками созданные страницы (меню **Изменить** – **Создать ссылку** или вручную средствами языка разметки HTML).

Лабораторная работа №3: Создание базы данных MySQL

В ходе выполнения данной лабораторной работы необходимо создать в MySQL новую базу данных с названием «**MySiteDB**» и добавить в нее две таблицы: **notes** и **comments**. **Notes** содержит заметки блога; **comments** – комментарии к этим заметкам. Схема данных (рис.3.1):

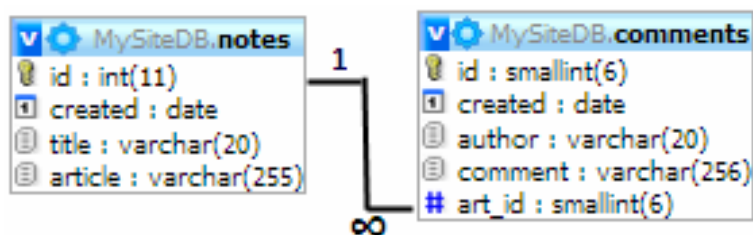


Рис.3.1. Схема базы данных “MySiteDB”

В Приложении 4 представлена информация об основных понятиях, необходимых для работы с базой данных.

Упражнение 1: Создание БД «MySiteDB»

В этом упражнении реализуется запрос на создание новой базы данных.

1. Создайте новый php документ, который будет называться **create_db.php**.
2. Создайте соединение с сервером localhost. Имя сервера **localhost**, пользователь **root**, пароля нет.
3. Создайте базу данных:
 - 3.1. Сформируйте запрос на создание базы **MySiteDB** с использованием SQL;
 - 3.2. Реализуйте запрос на создание БД с помощью функции `mysqli_query()`.
4. Сохранить документ, выполнить запрос.
5. С помощью утилиты **PhpMyAdmin** убедитесь, что создана новая база данных. Для этого запустите утилиту: `http://localhost/tools/phpmyadmin` (или `http://localhost` и выберите PhpMyAdmin из списка утилит).
6. Вторично выполните запрос, чтобы убедиться, что соединение есть, а база не создается (т.к. она была уже создана ранее, в ходе предыдущего выполнения скрипта).
7. Желательно добавить цикл `if` для обнаружения неполадок в работе.

Вариант реализации создания БД MySiteDB

```
<?php
//Создать соединение с сервером
$link = mysqli_connect ("localhost", "root", "");
if ($link) {
    echo "Соединение с сервером установлено", "<br>";
} else {
    echo "Нет соединения с сервером";
}
```



```

//Создать БД MySiteDB
//Сначала формирование запроса на создание
$db = "MySiteDB";
$query = "CREATE DATABASE $db";

//Затем реализация запроса на создание. Важна последовательность
аргументов функции: соединение с сервером, SQL-запрос.
$create_db = mysqli_query($link, $query);
if ($create_db) {
    echo "База данных $db успешно создана";
} else {
    echo "База не создана";
}
?>

```

Упражнение 2: Создание пользователя admin

В этом упражнении Необходимо создать нового пользователя базы данных с именем **admin** и паролем **admin** с правами администратора. Пользователей можно добавлять двумя способами:

- при помощи SQL-запроса GRANT
- в таблице назначения привилегий MySQL (Privileges) с помощью утилиты PhpMyAdmin.

Выберите один из двух приведенных далее способов.

Способ 1: создание нового пользователя с помощью SQL-запроса GRANT

1. Создайте новый php-документ, который будет называться create_user.php;
2. Создайте соединение с сервером;
3. Сформируйте SQL-запрос на создание нового пользователя базы данных:

```

$query = "GRANT ALL PRIVILEGES ON *.* TO 'admin'@'localhost'
IDENTIFIED BY 'admin'
WITH GRANT OPTION";

```

//*. * - глобальный уровень привилегий, применяется ко всем базам на сервере.

4. Реализуйте запрос.

- Проверка создания пользователя. С помощью утилиты PhpMyAdmin убедитесь, что создан новый пользователь. Для этого запустите утилиту PhpMyAdmin и перейдите на вкладку Privileges. Изучите список пользователей.

Способ 2: создание нового пользователя с помощью утилиты PhpMyAdmin

1. Запустите утилиту PhpMyAdmin и перейдите на вкладку Privileges. Нажмите кнопку «Add a new user».
2. Введите имя пользователя (**admin**), имя сервера (**localhost**), пароль с подтверждением (**admin**). Предоставьте новому пользователю все права (**global privileges – Check All**).
3. Убедитесь, что новый пользователь создан корректно.
4. Все дальнейшие действия с базой данных будут проводиться под пользователем **admin** с паролем **admin** и соответствующими правами, если иное не указано в задании.

Упражнение 3: Создание таблицы notes

В данном упражнении будет продемонстрирован один из способов создания таблиц в ранее созданной базе данных на примере создания таблицы **notes**. Таблица **notes** содержит заметки автора блога. Данная таблица будет создана средствами PHP. Информацию о полях таблицы см. в *Приложении 3*.

1. Создайте новый php-документ, который будет называться **create_tbl.php**;
2. Создайте соединение с сервером уже под созданным ранее пользователем **admin** с паролем **admin**.
3. Подключитесь к базе данных MySiteDB.
4. Сформируйте запрос на создание таблицы **notes** с полями, указанными в *Приложении 3*.

//Формирование запроса

```
$query = "CREATE TABLE notes  
        (id INT NOT NULL AUTO_INCREMENT,  
        PRIMARY KEY (id),  
        created DATE,  
        title VARCHAR (20),  
        article VARCHAR (255))";
```

5. Реализуйте запрос на создание таблицы.

6. С помощью утилиты PhpMyAdmin убедитесь, что создана новая таблица. Для этого запустите утилиту, перейдите к базе данных MySiteDB и просмотрите ее структуру. В ней должна появиться соответствующая таблица.

Вариант реализации создания таблицы notes

```
<?php
//Соединение с сервером
$link = mysqli_connect ('localhost', 'admin', 'admin');

//Выбор БД
$db = "mySiteDB";
$select = mysqli_select_db($link, $db);
if ($select){
    echo "База успешно выбрана", "<br>";
} else {
    echo "База не выбрана";
}

//Создание таблицы
//Формирование запроса
$query = "CREATE TABLE notes
        (id INT NOT NULL AUTO_INCREMENT,
        PRIMARY KEY (id),
        created DATE,
        title VARCHAR (20),
        article VARCHAR (255))";

//Реализация запроса
$create_tbl = mysqli_query ($link, $query);
if ($create_tbl){
    echo "Таблица успешно создана", "<br>";
} else {
    echo "Таблица не создана";
}
?>
```

Упражнение 4: Создание таблицы comments

В данном упражнении будет продемонстрирован другой способ создания таблиц в ранее созданной базе данных на примере создания таблицы **comments**. Таблица **comments** содержит комментарии пользователей к

заметкам автора блога. Таблица будет создана с помощью утилиты PhpMyAdmin. Информацию о полях таблицы см. в *Приложении 3*.

1. Запустите браузер.
2. Запустите утилиту *phpMyAdmin*. В главном окне *PHPMyAdmin* выберите БД *MySiteDB*.
3. В поле “Create new table”, присвойте имя таблице – *comments*; количество полей - 5, нажмите кнопку «Go» (рис. 3.2).

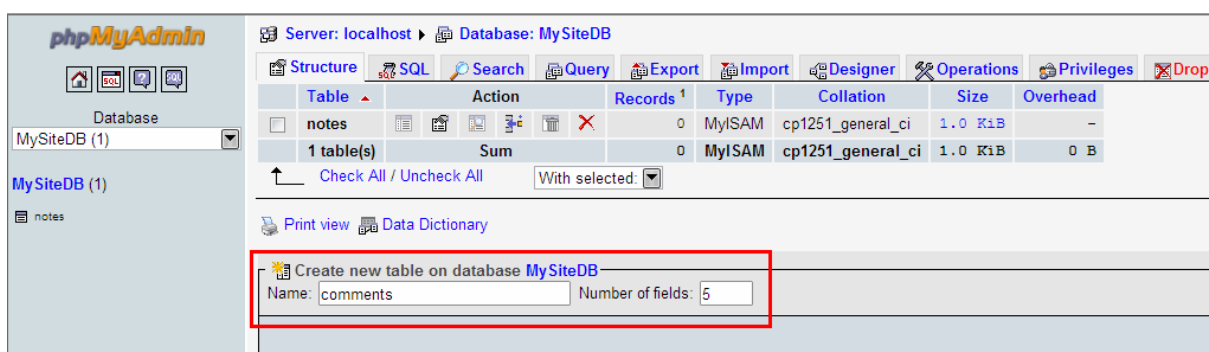


Рис. 3.2. Создание новой таблицы с помощью утилиты *phpMyAdmin*.

4. Создание полей таблицы *comments*:
 - 4.1. В открывшемся окне заполните необходимые поля таблицы (рис. 3.3) и нажмите кнопку «Save».
 - 4.2. Для поля *id* добавьте следующие атрибуты: обозначьте автоинкремент *A_I* и первичный ключ *PRIMARY* в поле со списком *INDEX*.

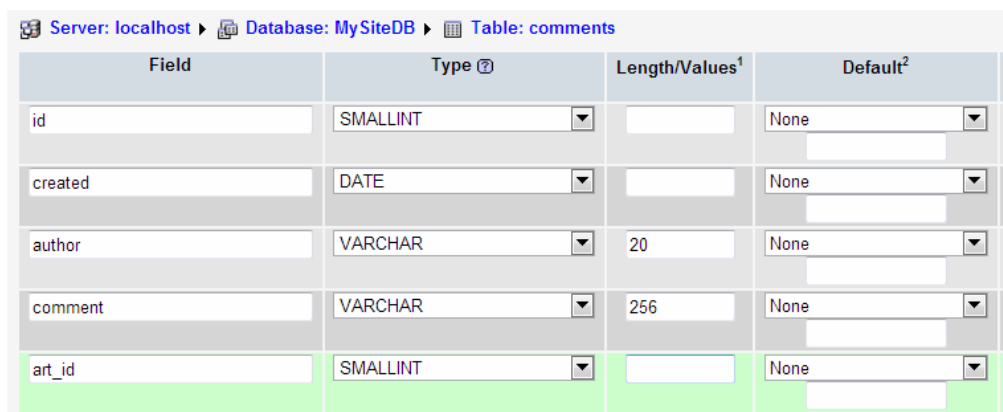


Рис. 3.3. Заполнение полей таблицы

5. Полученный результат должен выглядеть следующим образом (рис. 3.4):




































	Field	Type	Collation	Attributes	Null	Default	Extra	Action
<input type="checkbox"/>	id	smallint(6)			No	None	auto_increment	      
<input type="checkbox"/>	created	date			No	None		      
<input type="checkbox"/>	author	varchar(20)	cp1251_general_ci		No	None		      
<input type="checkbox"/>	comment	varchar(256)	cp1251_general_ci		No	None		      
<input type="checkbox"/>	art_id	smallint(6)			No	None		      

Рис. 3.4. Результат создания таблицы

Упражнение 5: Создание межтабличных связей

В данном упражнении необходимо создать связи между таблицами для поддержания целостности данных web-приложения.

1. Для организации межтабличных связей выберите БД MySiteDB, вкладку Designer. Откроется окно схемы данных.
2. С помощью инструментов окна Designer создайте связь «один ко многим» (рис. 3.5).

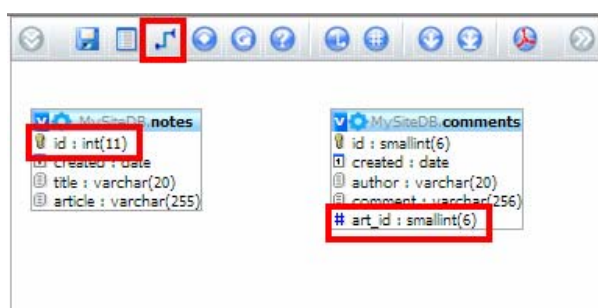


Рис. 3.5 Поле окна инструментов Designer

3. Введите в созданные таблицы несколько записей – для проверки их работы и для использования на будущих серверных страницах сайта. Для этого выберите нужную таблицу и нажмите кнопку **Insert** в группе **Action**. После заполнения соответствующих полей таблицы нажмите кнопку **Go**.

⇒ Помните, что поля **id** в таблицах заполнять не надо – они заполняются автоматически.

⇒ Поле **art_id** таблицы **comments** должно быть привязано к полю **id** таблицы **notes**, т.к. комментарии создаются только в привязке к конкретной заметке. MySQL автоматически в поле **art_id** таблицы **comments** подставляет выпадающий список **id** уже созданных заметок, вам необходимо лишь выбрать **id** заметки из выпадающего списка.

Упражнение 6: Регистрация базы данных в Adobe Dreamweaver для подключения к сайту

Регистрация включает задание имени и пароля пользователя, от имени которого ведется работа с базой (в нашем случае admin), а также адрес сервера данных (localhost) и имя базы (MySiteDB).

1. Запустите сервер и базу данных MySQL.
2. В Adobe Dreamweaver в меню **Базы данных** нажмите кнопку «плюс (+)». Появится единственный пункт – **Подключение MySQL**.
3. Далее появится диалоговое окно, которое необходимо заполнить (см. рис.3.6). Последний раздел «База данных» можно заполнить как самостоятельно, так и нажав **Выбрать**. Протестируйте соединение. Затем нажмите **ОК**.

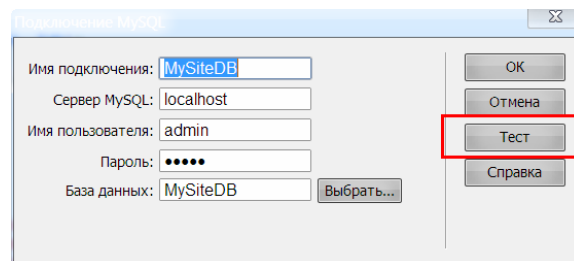


Рис. 3.6. Заполнение полей окна подключения базы данных к проекту

4. После подключения в меню Базы данных должна отобразиться подключенная нами БД MySiteDB с двумя таблицами (рис. 3.7).

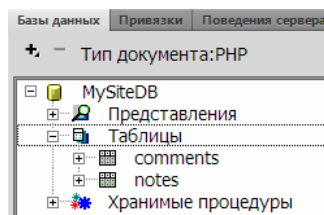


Рис. 3.7. Отображение подключенной к проекту базы данных

5. Убедитесь, что в меню **Файлы** появилась новая папка **Connections** и в ней файл **MySiteDB.php** (название файла совпадает с именем соединения).

Упражнение 7: Файл подключения базы данных

В ходе выполнения данного упражнения необходимо внести изменения в код автоматически созданного файла подключения базы данных для настройки корректной работы подключенной базы данных с кириллицей.

1. Откройте файл MySiteDB.php и внесите в него следующие изменения:
 - 1.1. Измените наименования переменных для удобства дальнейшей работы;
 - 1.2. Измените функцию *mysql_pconnect()* на *mysqli_connect()*;
 - 1.3. Внесите дополнения для корректной кодировки символов в базе данных:

```
<?php
# FileName="Connection_php_mysql.htm"
# Type="MYSQL"
# HTTP="true"
$localhost = "localhost";
$db = "MySiteDB";
$user = "admin";
$password = "admin";

$link = mysqli_connect($localhost, $user, $password) or
trigger_error(mysql_error(),E_USER_ERROR);
//trigger_error выводит на страницу сообщение об ошибке. Первый параметр
- сообщение об ошибке
//в строковом виде, в данном случае возвращается функция mysql_error(),
второй - числовой код //ошибки(почти всегда используется значение
константы E_USER_ERROR, равное 256)

//Следующие строки необходимы для того, чтобы MySQL воспринимал
кириллицу.
//Параметры функции mysqli_query(): идентификатор соединения с сервером
и запрос SQL

mysqli_query($link, "SET NAMES cp1251;") or die(mysql_error());
mysqli_query($link, "SET CHARACTER SET cp1251;") or die(mysql_error());
?>
```

Лабораторная работа №4: Простой вывод данных. Страницы `blog.php` и `comments.php`.

Упражнение 1: Вывод данных из базы на страницу

В этом упражнении на главную страницу сайта необходимо вывести все заметки из таблицы БД `notes`.

1. Переименуйте `blog.html` в `blog.php` (старую `html`-страницу после переименования можно удалить).
2. Удалите записи на странице. Должны остаться только приветствие и навигационное меню.
3. Создайте соединение с сервером. Оно у нас уже реализовано в файле `MySiteDB.php` – файл надо просто включить с помощью функции `require_once()`, в качестве параметра передав ей путь к файлу («`Connections/MySiteDB.php`»):

```
<?php require_once ("connections/MySiteDB.php"); ?>
```

4. Далее необходимо вывести записи (строки) на страницу сайта из таблицы `notes`. Сначала надо реализовать запрос на выборку. Для этого:

4.1. выберите БД;

4.2. создайте SQL-запрос на выборку данных из таблицы (`SELECT fields FROM tableName`). Здесь `SELECT` – оператор выбора полей, `FROM` – оператор выбора таблицы-источника полей.

⇒ Если вам необходимо выбрать все поля таблицы (как в данном случае), то запрос можно построить так: `SELECT * FROM tablename`, где символ «`*`» обозначает все поля таблицы.

4.3. Реализуйте запрос на выборку.

5. Далее необходимо вывести запись на страницу сайта. Для этого используется функция `mysqli_fetch_array()`. Параметром функции является переменная, содержащая результат выполнения запроса к БД (в данном случае – реализации запроса на выборку); собственно функция получает по одной записи из таблицы за один раз. Каждая запись возвращается в виде массива.

6. Для вывода информации из массива по отдельным элементам необходимо придерживаться следующего синтаксиса:

```
//Вывод элементов массива
echo $note ['id'], "<br>";
echo $note ['created'], "<br>";
echo $note ['title'], "<br>";
echo $note ['article'], "<br>";
```

7. Сейчас из таблицы с помощью функции ***mysqli_fetch_array()*** выводится только одна запись. С помощью цикла необходимо сделать так, чтобы выводились все записи из таблицы. Для этого необходимо изменить часть кода следующим образом:

```
//Использование цикла while
while ($note = mysqli_fetch_array($select_note)){
    echo $note ['id'], "<br>";
    echo $note ['created'], "<br>";
    echo $note ['title'], "<br>";
    echo $note ['article'], "<br>";}
```

Здесь переменной с именем *\$select_note* присваивается результат выполнения запроса к БД *mysqli_query()*.

Упражнение 2: Обмен данными между серверными страницами

Каждая заметка на главной странице блога может быть прокомментирована. Для реализации этой функции необходимо сделать из заголовка каждой заметки гиперссылку, перейдя по которой посетитель попадет на страницу со списком комментариев к выбранной заметке. Кроме того, на этой же странице должна отображаться сама выбранная для комментирования заметка.

Следовательно, необходимо реализовать механизм обмена данными между страницами таким образом, чтобы при переходе по гиперссылке передавалась информация о том, какая именно заметка была выбрана.

Для этого необходимо ввести некий идентификатор, значение которого будет совпадать с *id* комментируемой заметки, и который будет передаваться при переходе по гиперссылке.

1. Создание гиперссылки

- 1.1. Создайте новую страницу **comments.php**, которая будет содержать комментарии к выбранной заметке.
- 1.2. Реализуйте соединение с сервером.
- 1.3. Реализуйте подключение к БД.
- 1.4. Для передачи идентификатора заметки введем аргумент **note**. В качестве значения он будет получать значение поля **id** таблицы **notes**.
- 1.5. На странице **blog.php** найдите фрагмент кода, передающего заголовок заметки **title** (`echo $note ['title'];`). Его необходимо отредактировать таким образом, чтобы он стал гиперссылкой на страницу комментариев **comments.php**, а также передавал **id** выбранной заметки:

```
while ($note = mysqli_fetch_array($select_note)){
    echo $note['id'], "<br>";
?>
<a href="comments.php?note=<?php echo $note['id']; ?>">
<?php echo $note ['title'], "<br>";?></a>

<?php
    echo $note ['created'], "<br>";
    echo $note ['content'], "<br>";
}
?>
```

Здесь мы создаем гиперссылку на страницу **comments.php** и в этой гиперссылке передаем идентификатор **note**, значение которого равно значению элемента массива **\$note['id']**, т.е. значению **id** заметки.

2. Страница **comments.php**

- 2.1. Перейдите на страницу **comments.php**. На данной странице должны отображаться комментарии к выбранной записи, а также сама комментируемая запись (для удобства посетителя сайта).
- 2.2. Данную задачу можно выполнить по аналогии с выводом заметок на странице **blog.php**. Основное отличие заключается в том, что вначале необходимо со станицы **blog.php** получить переданный с помощью идентификатора **note** id заметки. Это делается с помощью метода **\$_GET**:

//Переменной \$note_id необходимо присвоить id заметки, переданной с помощью метода \$_GET со страницы blog.php

```
$note_id = $_GET['note'];
```

- 2.3. Далее необходимо вывести значения полей **created**, **title**, **content** из таблицы **notes** для заметки с полученным **id**. Для этого используется SQL запрос

SELECT... FROM... WHERE...

В нем с помощью оператора **SELECT** выбираем необходимые поля таблицы; с помощью **FROM** определяем таблицу-источник выборки; **WHERE** задает условие отбора, по которому выбираем заметку с выбранным **id**:

```
//Формируем SQL-запрос на выборку с учетом переданного id заметки  
$query = "SELECT created, title, article FROM notes WHERE id = $note_id";
```

- 2.4. После формирования SQL-запроса его необходимо реализовать с помощью функции **mysqli_query()** и вывести данные на страницу с помощью функции **mysqli_fetch_array()**.
- 2.5. Затем аналогичным образом выведите комментарии к выбранной заметке. Обратите внимание, что SQL-запрос на выборку комментариев должен строиться следующим образом:

```
$query_comments = "SELECT * FROM comments WHERE art_id = $note_id";
```

В условии **WHERE** мы реализуем поддержку связи между таблицами, которые связаны по полям **id** (таблица **notes**) и **art_id** (таблица **comments**).

В переменной **\$note_id** содержится **id** выбранной заметки, следовательно, для выбора комментариев к этой заметке необходимо, чтобы значение поля **art_id** также было равно **\$note_id**.

3. Проверьте корректность данных между страницами **blog.php** и **comments.php**. При переходе по ссылке с **blog.php** на **comments.php** в адресной строке браузера должен отображаться **id** выбранной заметки, переданный с помощью идентификатора **note**.

4. Для того, чтобы выводились все комментарии, а не только первый – реализуйте цикл.
5. Если у заметки нет ни одного комментария – об этом надо сообщить.
 - 5.1. Под областью комментариев добавьте надпись *«Эту запись еще никто не комментировал»*.
 - 5.2. В коде программы создайте циклы с условием **if**: если хотя бы один комментарий существует – он должен быть выведен (т.е. элементы массива должны быть отображены); если количество комментариев равно нулю – должна выводиться надпись *«Эту запись еще никто не комментировал»*.

Лабораторная работа №5: Ввод и правка данных с помощью формы

В ходе выполнения данной лабораторной работы рассматриваются принципы ввода информации и модификации контента сайта с помощью обработки форм.

Упражнение 1: Отправка почты

Данное упражнение позволяет реализовать отправку сообщения через форму на сервер.

1. Создайте страницу **email.php**. Добавьте название страницы и пояснительный тест, форму с двумя текстовыми полями: **Тема сообщения** и **Текст сообщения**, кнопку **Отправить**, а также гиперссылку для возврата на главную страницу сайта.
2. Самостоятельно реализуйте обработку данных формы с помощью функции **mail()**. «Получить» отправленное сообщение вы можете по локальному адресу: C:\WebServers\tmp\!sendmail\
3. Проверьте корректность работы, создайте гиперссылки с главной страницы сайта на страницу email.php и со страницы email.php на страницу blog.php.
4. Самостоятельно реализуйте проверку заполнения всех полей формы для того, чтобы исключить отправку «пустого» письма.

Упражнение 2: Страница для добавления заметок

В этом упражнении будет проиллюстрировано создание страницы для добавления новых заметок – **newnote.php**.

1. Создайте новую страницу **newnote.php**, добавьте название и пояснительный текст.
2. Создать html-форму с именем «**newnote**», метод обработки данных – **POST**.
3. На форме разместите два поля: одно (типа *text*) для добавления заголовка заметки – с именем «**title**», другое (*textarea*) для добавления самой заметки – с именем «**article**». Добавьте параметры размера элементов формы.
4. Также поместите на поле кнопку отправки с именем «**submit**».

⇒ Не забывайте именовать html-форму и элементы html-формы (атрибут **name**). Эти имена важны при дальнейшей обработки данных, полученных через форму, в php-скриптах.

5. Добавление даты создания заметки

5.1. В таблице **notes**, заполняемой через создаваемую нами форму, осталось незаполненным поле **art_id** (поле с датой создания заметки) – для него мы не создавали элемент формы. PHP позволяет получать текущую дату автоматически, с помощью функции **date()**. Формат ее вызова: **date(<формат>)**. MySQL требует формат даты <год>-<месяц>-<число>, при этом год – 4 цифры, месяц – 2 цифры, число – 2 цифры. Согласно шаблону, вид вызова функции: **date("Y-m-d")**. Мы автоматизируем процесс получения текущей даты из формы.

5.2. Разместите на форме после второго текстового поля скрытое поле с именем «**created**».

5.3. Значение поле **created** будет получать через php- функцию **date()**. Результат добавления поля:

```
<input type="hidden" name = "created" id = "created"
      value = "<?php echo date("Y-m-d");?>" />
```

Вариант реализации html-формы

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

```

<p>Добавить новую заметку: </p>
<form id="newnote" name="newnote" method="post">
<input type="text" name="title" id="title" size="20" maxlength="20"/>
<textarea name="article" cols="55" rows="10" id="article"> </textarea>
<input type="hidden" name="created" id="created"
        value="<?php echo date("Y-m-d");?>"/>
<input type="submit" name="submit" id="submit" value="Отправить" />
</form>
<a href="blog.php">Возврат на главную страницу сайта</a>
</body>
</html>

```

6. Обработка html-формы. Вам необходимо создать php-скрипт, который выполнит два шага:

- Получит данные, введенные пользователем в поля созданной html-формы (т.е. новую заметку);
- Передаст эти данные в базу, где хранятся уже созданные ранее заметки.

6.1. Получение данных через форму. Для получения данных через форму необходимо:

6.1.1. Подключиться к серверу;

6.1.2. Выбрать базу данных;

6.1.3. Получить данные из полей формы. Данные мы получаем из элементов формы используя названия (атрибут **name**) этих элементов. Данные формы помещаются в массив `$_POST`, а затем присваиваются переменным php. Принцип получения:

```
$имя_переменной = $_POST [АтрибутNameЭлементаФормы];
```

Таким образом информация, введенная пользователем в форму, «присваивается» в качестве значения для переменной php.

```
//Получение данных из формы
```

```
$title = $_POST['title'];
```

```
$created = $_POST['created'];
```

```
$article = $_POST['article'];
```

6.2. Передача данных в базу

- 6.2.1. Данные в базу передаются по обычному принципу: формирование SQL-запроса – реализация SQL-запроса .
Формирование запроса: (в нем поле id получает свое значение автоматически):

```
//Формирование запроса
$query = "INSERT INTO notes (title, created, article)
VALUES ('$title', '$created', '$article)";
```

В запросе используется SQL-инструкция INSERT. Синтаксис инструкции:

```
INSERT INTO tblName (tblField 1, tblField 2, ... , tblField N)
VALUES (value 1, value 2, ... , value N);
```

В ней *tblName* – имя таблицы, *tblField* – имя поля таблицы (перечисляются в том порядке, в котором располагаются в таблице), *value* – вставляемое значение поля таблицы (порядок должен соответствовать порядку имен полей).

- 6.2.2. Реализуйте запрос с помощью функции `mysqli_query()`.

7. Проверьте корректность работы формы и обработки данных.
8. Самостоятельно программно исключите возможность передачи в базу данных пустой записи.
9. Добавьте гиперссылки между страницами **blog.php** и **newnote.php**.

Вариант реализации кода страницы newnote.php

```
<html>
<head>
    <title>Страница для добавления заметки</title>
</head>

<body>
    <p>Добавить новую заметку: </p>
    <form id="newnote" name="newnote" method="post" action="">
    <input type="text" name="title" id="title" size="20" maxlength="20"/>
    <textarea name="article" cols="55" rows="10" id="article"> </textarea>
    <input type="hidden" name="created" id="created"
        value="<?php echo date("Y-m-d");?>" />
```

```
<input type="submit" name="submit" id="submit" value="Отправить" />
</form>
```

```
<a href="blog.php">Возврат на главную страницу сайта</a>
</body>
</html>
```

```
<?php
//Подключение к серверу
require_once ("connections/MySiteDB.php");
//Выбор БД
$select_db = mysqli_select_db ($link, $db);

//Получение данных из формы
$title = $_POST['title'];
$created = $_POST['created'];
$article = $_POST['article'];

if (($title)&&($created)&&($article))
{
    //Формирование запроса
    $query = "INSERT INTO notes (title, created, article) VALUES ('$title',
'$created', '$article')";
    //Реализация запроса
    $result = mysqli_query ($link, $query);
}
?>
```

Упражнение 3: Страница для редактирования заметок

В этом упражнении необходимо создать страницу **editnote.php**, добавить название и пояснительный текст. Переход на эту страницу будет осуществляться со страницы **comments.php** (т.к. в начале этой страницы выводится текст комментируемой заметки).

1. Откройте страницу **comments.php**. Создайте между текстом комментируемой заметки и повторяющейся областью комментариев пустой абзац и введите текст «Изменить заметку». Сделайте ее гиперссылкой для перехода на страницу **editnote.php**.
2. Гиперссылка на **editnote.php**. Для передачи информации на страницу **editnote.php** о том, какая именно заметка модифицируется (заметка с

каким *id*), необходимо передать идентификатор заметки со страницы **comments.php** в строке URL-адреса через гиперссылку.

3. При его получении на странице **editnote.php** используется метод GET (принцип работы аналогичен тому, что был использован при передаче идентификатора заметки со страницы **blog.php** на страницу **comments.php**), см. рис. 5.1:

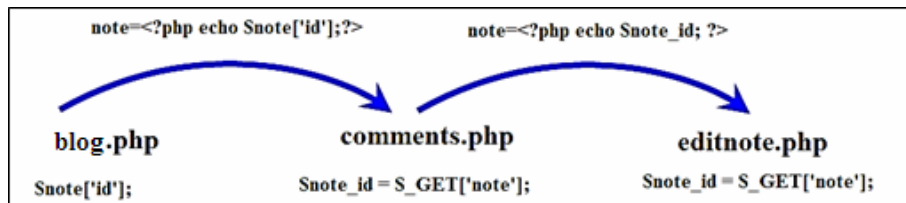


Рис.5.1. Схема обмена данными методом GET

Дополните гиперссылку со страницы **comments.php** на страницу **editnote.php**:

```
<a href="editnote.php?note=<?php echo $note_id; ?>">Исправить заметку</a>
```

4. Работа со страницей **editnote.php**

- 4.1. На странице **editnote.php** создайте html-форму с именем «**editnote**», метод обработки данных – POST.
- 4.2. На форме разместите два поля: одно (типа *text*) для изменения заголовка заметки – с именем «**title**», другое (*textarea*) для изменения самой заметки – с именем «**article**». Добавьте параметры размера элементов формы.
- 4.3. Также поместите на поле кнопку отправки с именем «**submit**».
- 4.4. Далее необходимо создать php-скрипт для обработки данных формы. Этот скрипт должен выполнять следующее:
 - Отображать редактируемую заметку в полях формы (т.е. помещать данные из базы в поля формы);
 - Получать измененные данные из формы;
 - Передавать изменение данные в таблицу.
5. Заполнение полей формы
 - 5.1. Введите переменную **\$note_id**, которая получит в качестве значения идентификатор обрабатываемой заметки. Это значение она должна получить через массив **\$_GET**.

- 5.2. Реализуйте соединение с сервером.
- 5.3. Выберите базу данных.
- 5.4. Далее необходимо сформировать запрос на получение заметки с выбранным **id** из базы данных, для размещения ее в полях формы. Запрос реализуется с помощью оператора SELECT, условием запроса должно быть **id** выбранной заметки.
- 5.5. Реализуйте сформированный запрос.
- 5.6. С помощью функции **mysqli_fetch_array()** поместите результат выполнения запроса (т.е. полученную строку) в массив.

Вариант реализации кода:

```
<?php
//получение идентификатора
$note_id = $_GET['note'];

//Соединение с сервером
require_once ("connections/MySiteDB.php");

//Выбор БД
$select_db = mysqli_select_db ($link, $db);

//Запрос к БД на получение строки, содержащей заметку с выбранным id
$query = "SELECT * FROM notes WHERE id = $note_id";

//Реализация запроса к БД
$result = mysqli_query ($link, $query);

//Помещение выбранной строки в массив
$edit_note = mysqli_fetch_array ($result);
?>
```

6. Необходимо, чтобы записи полученной заметки отображались в соответствующих полях формы. Для этого:
 - 6.1. Добавьте на html-форму скрытое поле с именем note (оно будет содержать id заметки).
 - 6.2. В html-форме задаем значение value для всех элементов из массива:

<!-- \$edit_note - это имя массива, в который помещается результат выполнения функции mysqli_fetch_array(); -->

```
<form id="editnote" name="editnote" method="post" action="">
<label for="title">Заголовок заметки</label>
<input type="text" name="title" id="title"
      value = "<?php echo $edit_note['title'];?>" />
<label for="article">Текст заметки </label>
<textarea name="article" id="article">
      <?php echo $edit_note['article'];?></textarea>
<input type="hidden" name="note" id="note"
      value="<?php echo $edit_note['id'];?>" />
<input type="submit" name="submit" id="submit" value="Изменить" />
</form>
```

7. Получение данных из формы после изменения. Принцип реализации похож на добавление новой заметки:

- 7.1. Получите из формы измененные данные с помощью метода `$_POST`;
- 7.2. Передайте данные в таблицу с помощью SQL-запроса. Разница заключается только в SQL-запросе - при добавлении используется `INSERT`, а при обновлении `UPDATE`.

⇒ Оператор `UPDATE` обновляет поля таблицы в соответствии с их новыми значениями в строках. Синтаксис запроса на обновление:

*UPDATE tblName SET fieldName1 = expr1, fieldName2 = expr2, ... ,
fieldName N = expr N WHERE ...*

где *tblName* – имя таблицы, *fieldName = expr* - указывается, какие именно поля надо изменить и какими должны быть их новые значения.

Вариант кода получения и передачи данных из формы

```
<?php
//Собственно обновление данных
//Получение обновленных значений из формы
$title = $_POST['title'];
$article = $_POST['article'];

//Создание запроса на обновление
```

```

$update_query = "UPDATE notes SET title = '$title', article = '$article'
                WHERE id = $note_id";
//Реализация запроса на обновление
$update_result = mysqli_query ($link, $update_query);
?>

```

8. Проверьте корректность работы скриптов.
9. Создайте гиперссылку для возврата на страницу комментариев.

Вариант полной реализации editnote.php кода

```

<?php
$note_id = $_GET['note'];
require_once ("connections/MySiteDB.php");
$select_db = mysqli_select_db ($link, $db);

$query = "SELECT * FROM notes WHERE id = $note_id";
$result = mysqli_query ($link, $query);
$select_note = mysqli_fetch_array ($result);
?>

<html>
<body>
<p>Страница редактирования заметки </p>

<form id="editnote" name="editnote" method="post" >
<label for="title">Заголовок заметки</label>
<input type="text" name="title" id="title"
        value = "<?php echo $select_note['title'];?>" />
<label for=" article">Текст заметки </label>
<textarea name=" article" id=" article">
        <?php echo $select_note['article'];?></textarea>
<input type="hidden" name = "note" id = "note"
        value="<?php echo $select_note['id'];?>" />
<input type="submit" name="submit" id="submit" value="Изменить" />
</form>

<a href="blog.php">Вернуться на главную страницу сайта</a>
</body>
</html>

```

```

<?php
$title = $_POST['title'];
$article = $_POST['article'];
$update_query = "UPDATE notes SET title = '$title', article = '$article'
                WHERE id = $note_id";
$update_result = mysqli_query ($link, $update_query);
?>

```

Упражнение 4: Создание страницы удаления заметок

Самостоятельно создайте страницу для удаления заметки `deletenote.php`. Переход на эту страницу также должен осуществляться со страницы `comments.php`.

Для реализации удаления записи из БД используется SQL- оператор DELETE. Синтаксис оператора DELETE:

DELETE FROM tblName WHERE ...

где `tblName` – имя таблицы.

Не забудьте реализовать удаление комментариев к удаляемым записям.

Лабораторная работа №6: Работа с заметками

Упражнение 1: Работа со страницей `blog.php`

1. Необходимо сделать так, чтобы последняя добавленная заметка отображалась в самом верху списка заметок. Для этого отредактируйте код SQL-запроса к БД таким образом, чтобы данные передавались в необходимом порядке. С этой целью используются следующий синтаксис:

SELECT *fieldName* FROM *tblName* ORDER BY *fieldName order*

fieldName – имя поля (полей) таблицы,

tblName – имя таблицы – источника,

order – порядок следования записей. Он может быть **ASC** – по возрастанию, **DESC** – по убыванию, **RAND** – в случайном порядке.

Например:

```
SELECT * FROM table ORDER BY name ASC
```

т.е. необходимо выбрать все поля из таблицы **table** и расположить их в порядке возрастания значений поля **name** (т.е. в алфавитном порядке, если поле строкового типа).

2. Необходимо на странице deletenote.php сделать ссылку возврата на страницу комментариев. При этом необходимо учитывать, что при переходе по простой ссылке (без дополнительных параметров) заметка все равно удалится, поэтому необходимо в гиперссылке «вернуть» странице комментариев принятые от нее аргументы GET, чтобы она снова вывела ту же самую заметку.

Вариант реализации кода

```
<?php
$note_id = $_GET['note'];
require_once ("connections/MySiteDB.php");
$select_db = mysqli_select_db ($link, $db);
$query = "SELECT * FROM notes WHERE id = $note_id";
$result = mysqli_query ($link, $query);
$delete_note = mysqli_fetch_array ($result);
?>

<html>
<body>
<p>Страница редактирования заметки </p>
<form id="editnote" name="editnote" method="post" >
<label for="title">Заголовок заметки</label>
<input type="text" name="title" id="title"
        value = "<?php echo $delete_note['title'];?>" />
<label for="article">Текст заметки </label>
<input type="text" name="article" id="article"
        value = "<?php echo $delete_note['article'];?>" />
<input type="hidden" name = "note" id = "note"
        value="<?php echo $delete_note['id'];?>" />
<input type="hidden" name = "MM_update" value="editnote" />
<input type="submit" name="submit" id="submit" value="Удалить" />
</form>
```

```

<?php
$submit = $_POST['submit'];
if ($submit)
{
    $delete_query = "DELETE FROM notes WHERE id = $note_id";
    $delete_result = mysqli_query ($link, $delete_query);
}
?>
</body>
</html>

```

Упражнение 2: Работа с комментариями к заметкам

Самостоятельно создайте страницу для добавления комментариев к заметкам, а также страницу для удаления комментариев.

Лабораторная работа №7: Страница статистики inform.php

В ходе выполнения лабораторной работы будет организована работа и выведены на страницу статистики следующие данные web-сайта:

1. Сколько всего было сделано записей в блоге;
2. Сколько комментариев было добавлено;
3. Сколько записей было сделано за последний месяц;
4. Сколько комментариев было оставлено за последний месяц;
5. Какая заметка была сделана последней;
6. Какую заметку больше всего комментировали.

Упражнение 1: Общее количество заметок и общее количество комментариев

1. Установите подключение к серверу и выберите базу данных.
2. С помощью SQL-запроса необходимо вычислить общее количество заметок в блоге. Для этого используется SQL-функция COUNT().
Данная функция возвращает количество строк, которые соответствуют определенным критериям. Синтаксис функции:

SELECT COUNT (fieldName) FROM tblName

Данная функция является **агрегатной**, т.е. позволяет выполнять различные действия сразу над многими записями.

Вариант реализации кода

```
//Вычисление количества заметок  
$query_allnotes = "SELECT COUNT(id) AS allnotes FROM notes";  
$allnotes = mysqli_query ($link, $query_allnotes) or die (mysqli_error());  
//mysqli_error()возвращает строку ошибки последней операции с MySQL  
$row_allnotes = mysqli_fetch_assoc ($allnotes);  
$allnotes_num = $row_allnotes['allnotes'];  
mysqli_free_result ($allnotes);  
//mysqli_free_result() освобождает память от результата запроса
```

3. Аналогичным образом реализуйте подсчет общего количества комментариев.

Упражнение 2: Подсчет количества заметок и комментариев за последний месяц

В ходе выполнения этого упражнения необходимо реализовать следующий алгоритм работы:

- Вычислить начальную и конечную даты текущего месяца;
- Подставить результаты этих вычислений в условие фильтрации SQL-запроса.

1. Работа с датой

//Функция getdate() возвращает массив, содержащий информацию о различных составляющих текущей даты (чтобы дальше работать с ними "по частям"). В массив помещаются: секунды, минуты, часы, порядковый номер дня, порядковый номер месяца, порядковый номер года, название дня недели, название месяца, количество секунд с начала эпохи Unix.

```
$date_array = getdate();
```

//Вычисление начальной даты текущего месяца

*//Функция mktime() возвращает объединенное значение времени. Аргументы:
кол-во часов, минут, секунд, № месяца, число, год.*

```
$begin_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'],1,  
$date_array['year']));
```

*//Т.к. время в данном случае не нужно - поставлены нули.
//Возвращенное функцией mktime() значение приведено к воспринимаемому
MySQL параметру даты "Y-m-d".*

//Вычисление конечной даты текущего месяца

```
$end_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'] + 1,0,  
$date_array['year']));
```

*//Здесь все аналогично, кроме того, что мы вводим число месяца, равное
нулю; на основании этого функция mktime(), встретив дату с нулевым
числом, вернет последнее число предыдущего месяца (28, 29, 30 или 31).*

2. Запрос на получение заметок за последний месяц

```
$query_lmnotes = "SELECT COUNT(id) AS lmnotes FROM notes  
WHERE created>='$begin_date' AND created<='$end_date'";  
$lmnotes = mysqli_query ($link, $query_lmnotes)or die (mysqli_error());  
$row_lmnotes = mysqli_fetch_assoc ($lmnotes);  
$lmnotes_num = $row_lmnotes['lmnotes'];  
mysqli_free_result ($lmnotes);  
  
//$begin_date – первое число месяца,  
//$end_date – последнее число месяца.
```

3. Аналогичным образом вычислите количество комментариев за последний месяц.

Упражнение 3: Последняя добавленная заметка

1. Для вывода последней добавленной заметки необходимо использовать оператор LIMIT в конструкции SELECT.

⇒ Выражение LIMIT используется для ограничения количества строк, возвращенных командой SELECT. LIMIT принимает один или два

числовых аргумента. Эти аргументы должны быть целочисленными константами. Если заданы два аргумента, то первый указывает на начало первой возвращаемой строки, а второй задает максимальное количество возвращаемых строк. При этом смещение начальной строки равно 0, а не 1 (т.к. первый элемент массива строк имеет индекс 0).

Например:

```
SELECT * FROM table LIMIT 5,10; // возвращает строки 6-15
```

Вариант реализации кода

//Последняя добавленная заметка

//Таблица notes сортируется по дате публикации заметки по убыванию, а затем из нее берется только самая первая запись (LIMIT 0,1) - "начиная с нулевой записи выбрать одну запись"

```
$query_last_note = "SELECT id, title FROM notes  
ORDER BY created DESC LIMIT 0,1";  
$lastnote = mysqli_query ($link, $query_last_note) or die (mysqli_error());  
$row_lastnote = mysqli_fetch_assoc ($lastnote);  
mysqli_free_result ($lastnote);
```

Упражнение 4: Самая комментируемая заметка

В ходе выполнения этого упражнения необходимо реализовать следующий алгоритм работы:

- Связать таблицы **notes** и **comments** по полям **id** и **art_id** соответственно, чтобы затем вычислить количество комментариев для каждой заметки;
- Выполнить группировку таблицы **comments** по идентификатору заметки.
- Вычислить количество комментариев для каждой заметки.
- Отсортировать результат по количеству комментариев для каждой заметки по убыванию.
- Вывести первую запись из получившегося набора записей.

1. Построение SQL-запроса

```
$query_mcnote = "SELECT notes.id, notes.title FROM comments, notes  
WHERE comments.art_id=notes.id
```

*GROUP BY notes.id
ORDER BY COUNT(comments.id) DESC LIMIT 0,1";*

В тексте запроса:

GROUP BY – оператор группировки. Группировка – это объединение записей в группы по какому-либо критерию (т.н. критерию группировки), который записывается сразу после оператора (в данном случае группировка по полю id таблицы notes).

ORDER BY COUNT (comments.id) DESC LIMIT 0,1 – осуществление сортировки по убыванию результатов выполнения агрегирующей функции COUNT() по id комментариев и вывод первой записи (т.е. записи с самым большим количеством комментариев).

Реализуйте данный запрос и поместите его результат в массив.

Упражнение 5: Размещение данных на странице

С помощью php-сценариев и оператора echo вывести результаты на страницу сайта. Ниже представлен вариант реализации кода вывода информации:

```
<html>
<body>
Сделано записей - <?php echo $allnotes_num; ?><br>
Оставлено комментариев - <?php echo $allcomments_num; ?><br>
За последний месяц я создал записей - <?php echo
                                $row_lmnotes['lmnotes'];?><br>
За последний месяц оставлено комментариев - <?php echo
                                $row_lmcomments['lmcomments'];?><br>
Моя последняя запись -
                                <a href="comments.php?note=<?php echo $row_lastnote['id'];?>">
                                <?php echo $row_lastnote['title'];?></a><br>
Самая обсуждаемая запись -
                                <a href="comments.php?note=<?php echo $row_mcnote['id'];?>">
                                <?php echo $row_mcnote['title'];?>
</a><br><br>

<p><a href="blog.php">Возврат на главную страницу сайта </a></p>
</body>
</html>
```

В представленном коде:

\$allnotes_num, *\$allcomments_num*, *\$row_imnotes*, *\$row_lmcomments*, *\$row_lastnote*, *\$row_mcnote* – массивы, в которые помещаются результаты выполнения функций *mysqli_fetch_array()*, вызываемых в ранее созданном коде для получения и хранения соответствующих данных.

Ниже представлен возможный вариант реализации всего кода страницы *inform.php* (без табличной html-структуры).

```
<?php require_once ("connections/MySiteDB.php");?>
<?php
mysqli_select_db ($link, $db);

//Вычисление количества заметок
$query_allnotes = "SELECT COUNT(id) AS allnotes FROM notes";
$allnotes = mysqli_query ($link, $query_allnotes) or die (mysqli_error());
$row_allnotes = mysqli_fetch_assoc ($allnotes);
$allnotes_num = $row_allnotes['allnotes'];
mysqli_free_result ($allnotes);

//Вычисление количества комментариев
$query_allcomments = "SELECT COUNT(id) AS allcomments FROM comments";
$allcomments = mysqli_query ($link, $query_allcomments) or die
(mysqli_error());
$row_allcomments = mysqli_fetch_assoc ($allcomments);
$allcomments_num = $row_allcomments['allcomments'];
mysqli_free_result ($allcomments);

//Работа с датой
$date_array = getdate();
$begin_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'],1,
                                                $date_array['year']));
$end_date = date ("Y-m-d", mktime(0,0,0, $date_array['mon'] + 1,0,
                                                $date_array['year']));

//Заметки за последний месяц
$query_lmnotes = "SELECT COUNT(id) AS lmnotes FROM notes
                  WHERE created>='$begin_date' AND created<='$end_date'";
$lmnotes = mysqli_query ($link, $query_lmnotes) or die (mysqli_error());
$row_lmnotes = mysqli_fetch_assoc ($lmnotes);
$lmnotes_num = $row_lmnotes['lmnotes'];
```

```

mysqli_free_result ($lmnotes);

//Комментарии за последний месяц
$query_lmcomments = "SELECT COUNT(id) AS lmcomments FROM comments
                     WHERE created >= '$begin_date' AND created <=
'$end_date'";
$lmcomments = mysqli_query ($link, $query_lmcomments) or die (mysqli_error());
$row_lmcomments = mysqli_fetch_assoc ($lmcomments);
$lmcomments_num = $row_lmcomments['lmcomments'];
mysqli_free_result ($lmcomments);

//Последняя добавленная заметка
$query_last_note = "SELECT id, title FROM notes
                   ORDER BY created DESC LIMIT 0,1";
$lastnote = mysqli_query ($link, $query_last_note) or die (mysqli_error());
$row_lastnote = mysqli_fetch_assoc ($lastnote);
mysqli_free_result ($lastnote);

//Самая комментируемая заметка
$query_mcnote = "SELECT notes.id, notes.title FROM comments, notes
                WHERE comments.art_id=notes.id
                GROUP BY notes.id
                ORDER BY COUNT(comments.id) DESC LIMIT 0,1";
$mcnote = mysqli_query($link, $query_mcnote) or die (mysqli_error());
$row_mcnote = mysqli_fetch_assoc($mcnote);
mysqli_free_result ($mcnote);
?>

<html>
<body>
Сделано записей - <?php echo $allnotes_num; ?><br>
Оставлено комментариев - <?php echo $allcomments_num; ?><br>
За последний месяц я создал записей - <?php echo
                                     $row_lmnotes['lmnotes'];?><br>
За последний месяц оставлено комментариев - <?php echo
                                     $row_lmcomments['lmcomments'];?><br>
Моя последняя запись -
    <a href="comments.php?note=<?php echo $row_lastnote['id'];?>">
        <?php echo $row_lastnote['title'];?></a><br>
Самая обсуждаемая запись -
    <a href="comments.php?note=<?php echo $row_mcnote['id'];?>">

```

```

<?php echo $row_mcnote['title'];?>
</a><br><br>

<p><a href="blog.php">Возврат на главную страницу сайта </a></p>
</body>
</html>

```

Лабораторная работа №8: Реализация поиска по сайту

В ходе выполнения данной лабораторной работы будут изучены основные функции работы со строками, а также поиск информации по web-сайту (по одному и нескольким словам поискового запроса).

Упражнение 1: Реализация поиска по сайту

В ходе выполнения данного упражнения с использованием функций работы со строками необходимо реализовать возможность поиска по ключевому слову заметки на главной странице сайта (возможны два варианта реализации: поиск по одному слову и поиск по фразе).

1. Поиск по одному ключевому слову: Для реализации поиска по одному слову можно использовать оператор LIKE и заменители символов %.

Вариант реализации кода:

```

//Поиск по одному слову
$user_search = $_GET['usersearch'];
if (!empty($user_search))
{
    $query_usersearch = "SELECT * FROM notes
        WHERE title LIKE '%$user_search%'
        OR article LIKE '%$user_search%'";
    $result_usersearch = mysqli_query($link, $query_usersearch);
    while ($array_usersearch = mysqli_fetch_array($result_usersearch))
    {
        echo $array_usersearch['id'];
        echo $array_usersearch['title'];
        echo $array_usersearch['article'];
    }
}

```

```
}
```

2. Реализация поиска по фразе:

- 2.1. Фразу надо разбить на отдельные слова (подстроки) и пометить в массив подстрок с помощью функции ***explode()***;

```
$search_query = "SELECT * FROM tableName"
$where_clause = ' '; // условие поиска
$user_search = $_GET['usersearch']; // получаем данные из поля
поиска
$search_words = explode(' ', $user_search);

foreach($search_words as $word)
{
    // Формируем условие поиска
    $where_clause .= " fieldName LIKE '%$word%' OR "
}
if (!empty($$where_clause))
{
    $search_query .= " WHERE $$where_clause ";
}
```

- 2.2. Для того, чтобы в конце строки запроса не была оператора OR, можно использовать функцию ***implode()***, создающую строку из массива подстрок, переданного ей в качестве аргумента. Представленный ранее код можно изменить следующим образом:

```
$search_query = "SELECT * FROM tableName"
$where_list = array();
$user_search = $_GET['usersearch'];
$search_words = explode(' ', $user_search);

foreach($search_words as $word)
{
    // В конец массива добавляется новый элемент
    $where_list[] = "article LIKE '%$word%'";
}
$where_clause = implode(' OR ', $where_list);
```

```

if (!empty($where_clause))
{
    $query_usersearch .= " WHERE $where_clause";
}

```

Вариант реализации кода:

```

//Поиск по фразе (по содержанию заметки)

$user_search = $_GET['usersearch'];
$where_list = array();
$query_usersearch = "SELECT * FROM notes";
$search_words = explode(' ', $user_search);

foreach($search_words as $word)
{
    $where_list[] = " article LIKE '%$word%'";
}
$where_clause = implode (' OR ', $where_list);
if (!empty($where_clause))
{
    $query_usersearch .= " WHERE $where_clause";
}

$res_query = mysqli_query($link, $query_usersearch);
while ($res_array = mysqli_fetch_array($res_query))
{
    echo $res_array['id'], "<br>";
    echo $res_array['article'], "<br>", "<hr>", "<br>";
}

```

Упражнение 2: Обработка строки поиска

Строка поиска должна содержать несколько слов, разделенных одним пробелом. Но надо учитывать, что пользователь может вводить слова поискового запроса через запятую (например, «заметка, моя, новая»). Такую строку необходимо перед передачей в запрос в базе данных обработать и привести к необходимому виду. Минимальная обработка осуществляется в два этапа:

- Замена запятых на пробелы;

- Удаление лишних пробелов между словами строки.

1. Замена запятых на пробелы осуществляется с помощью функции ***str_replace()***. Эта функция заменяет строку поиска на строку замены. Обязательными являются три аргумента: что заменить, чем заменить, где заменить. Следовательно, в данном случае вызов функции будет выглядеть следующим образом:

```
str_replace( ' , ' , ' ', $user_search );
```

2. Удаление лишних пробелов между словами строки поискового запроса: в том случае, когда запятые были заменены на пробелы, появились лишние пробелы (т.е. более одного) между словами строки запроса. Если их не удалить, то в запросе они будут рассматриваться как пустые элементы массива, на основании которого формируется запрос к базе данных. Следовательно, при таком запросе будут выдаваться все записи базы данных. Для удаления пустых элементов массива можно сделать следующее:

- Создать новый массив, в котором будут сохраняться только действительные (непустые) критерии поиска. На основании этого массива будет строиться запрос к базе данных.
- Для создания этого массива можно пройти в цикле *foreach* все элементы уже существующего созданного ранее массива, используя управляющую конструкцию *if* найти все непустые элементы и скопировать их в новый массив.

2.1. В новый массив ***\$final_search_words*** помещаются непустые элементы уже существующего массива ***\$search_words***.

```
//Извлечение критериев поиска в массив
```

```
//Замена запятых на пробелы
```

```
$clean_search = str_replace( ' , ' , ' ', $user_search );
```

```
$search_words = explode( ' ', $user_search );
```

```
//Создаем еще один массив с окончательными результатами
```

```
$final_search_words = array();
```

```
//Проходим в цикле по каждому элементу массива $search_words.
```

```
//Каждый непустой элемент добавляем в массив с названием
```

```
//$final_search_words
```

```
if (count($search_words) > 0)
```

```

    {
        foreach($search_words as $word)
        {
            if (!empty($word))
            {
                $final_search_words[] = $word;
            }
        }
    }

```

- 2.2. Далее реализация работы с массивом такая же, разница – в используемом массиве (работа происходит с новым массивом ***\$final_search_words***).

```

foreach ($final_search_words as $word)
{
    $where_list[] = " article LIKE '%$word%'";
}
$where_clause = implode (' OR ', $where_list);
if (!empty($where_clause))
{
    $query_usersearch .= " WHERE $where_clause";
}

```

Вариант реализации кода:

```

//Поиск по фразе (по содержанию заметки)
$user_search = $_GET['usersearch'];
$where_list = array();
$query_usersearch = "SELECT * FROM notes";
$clean_search = str_replace(',', ' ', $user_search);
$search_words = explode(' ', $user_search);

//Создаем еще один массив с окончательными результатами
$final_search_words = array();

//Проходим в цикле по каждому элементу массива $search_words.
//Каждый непустой элемент добавляем в массив $final_search_words
if (count($search_words) > 0)
{
    foreach($search_words as $word)

```

```

        {
            if (!empty($word))
            {
                $final_search_words[] = $word;
            }
        }
    }
    //работа с использованием массива $final_search_words
    foreach ($final_search_words as $word)
    {
        $where_list[] = " article LIKE '%$word%'";
    }
    $where_clause = implode (' OR ', $where_list);
    if (!empty($where_clause))
    {
        $query_usersearch .= " WHERE $where_clause";
    }
    $res_query = mysqli_query($link, $query_usersearch);
    while ($res_array = mysqli_fetch_array($res_query))
    {
        echo $res_array['id'], "<br>";
        echo $res_array['article'], "<br>", "<hr>", "<br>";
    }
?>

```

Лабораторная работа №9: Передача файлов на сервер

В данной лабораторной работе будет изучены основные возможности PHP для реализации передачи файлов на сервер.

Упражнение 1: Вывод списка файлов

1. Создайте на локальном узле (в нашем случае в папке C:\WebServers\home\localhost\www\)) папки **photo** и **files** для размещения изображений и файлов соответственно. Поместите в эти папки несколько изображений и документов.
2. Создайте страницу **photo.php**, разместите на ней поясняющий текст, ниже поясняющего текста — две горизонтальные линии (между этими

линиями будет выводиться список имеющихся на web-узле изображений и ссылками на эти изображениями), см. *рис. 9.1*.

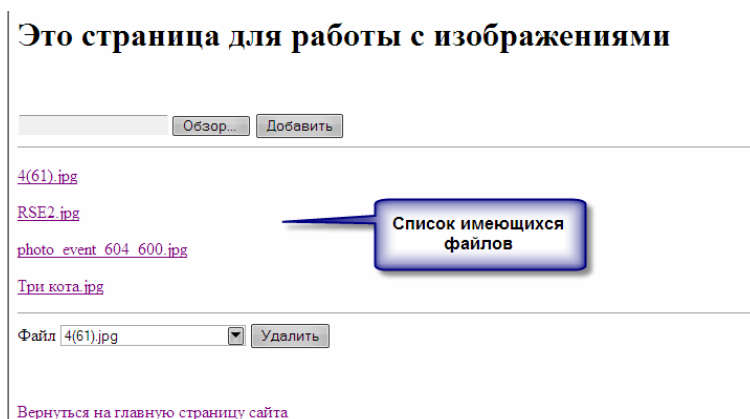


Рис. 9.1. Схема страницы photo.php

3. Для вывода списка файлов, уже имеющихся на сервере, необходимо создать два сценария:
 - первый будет формировать список файлов с изображениями,
 - второй - выводить этот список на страницу.

- 3.1. Формирование списка файлов. Для формирования списка файлов необходимо Получить путь к целевой папке, где хранятся необходимые файлы, а затем создать массив, в который будет помещаться необходимая информация о файлах (имена, пути к ним);

Вариант реализации кода

```
<?php
//Получаем полный путь к папке, где хранятся графические файлы
$image_dir_path = $_SERVER['DOCUMENT_ROOT'] . "/photo";

//Запускаем просмотр папки. Функция opendir() возвращает идентификатор
//папки
$image_dir_id = opendir($image_dir_path);

//$array_files - массив, в который будут помещаться все найденные файлы
$array_files = null;

//Служебная переменная, используемая для вычисления индекса следующего
//элемента массива $array_files
```

```

$i = 0;

//Запускаем цикл просмотра
while(($path_to_file = readdir($image_dir_id)) !== false)

//Функция readdir() возвращает полный путь к очередному файлу,
хранищемуся //в папке, идентификатор которой был возвращен функцией
opendir() и передан //в качестве параметра.
//$path_to_file получает полный путь к файлу для дальнейшей обработки.
Если в папке нет непросмотренных файлов - возвращается логическое
значение false
{
    if(($path_to_file != ".") && ($path_to_file != ".."))
//Точки обозначают вложенные файлы: одна точка - текущая папка, две
точки // - папка, в которую вложена текущая папка.
    {
        $array_files[$i] = basename($path_to_file);
        $i++;
//Помещаем имя найденного файла в массив $array_files. Функция basename()
//позволяет получить имя файла из полного пути к нему.
    }
}
closedir($image_dir_id);
//closedir() удаляет из памяти переданный ей идентификатор папки, таким
//образом завершая просмотр.
?>

```

3.2. Вывод списка файлов на страницу.

3.1.1. Найдите код, создающий две горизонтальные линии. Создайте следующий сценарий вывода списка файлов:

```

<?php
//Получаем количество элементов массива $array_files, т.е. количество
//найденных файлов.
$array_files_count = count($array_files);
if ($array_files_count)
{
    ?>
    <hr />
<?php

```

```

        sort($array_files);
        for ($i=0; $i<$array_files_count; $i++)
        {
//Выводим мена хранящихся в массиве файлов на страницу
            ?>
            <p><a href="/photo/<?php echo $array_files[$i]; ?>"
                        target="_blank">
                <?php echo $array_files[$i]; ?></a></p>
            <?php
        }
        ?>
        <hr />
    <?php
    }
?>

```

3.1.2. Создайте гиперссылку возврата на главную страницу и гиперссылку с главной страницы на страницу **photo.php** («Фото» в наборе гиперссылок).

4. Проверьте работу сценариев.

Упражнение 2: Отправка файлов на сервер

Отправка файлов на web-сайт состоит из двух этапов:

- создание необходимой формы для отправки файлов на странице сайта;
- написание сценария PHP для получения отправленного посетителем сайта файла.

1. Создание формы. Стандарт HTML предусматривает т.н. **поле выбора файла**, в которое посетитель должен будет ввести имя отправляемого файла. От обычного поля ввода оно отличается тем, что позволяет работать с окном открытия файлов Windows, а также оно отправляет серверной программе не введенное в него имя файла, а сам файл. Поле выбора файла создается с помощью тэга INPUT, атрибут type которого имеет значение "file". Также в форму с данным полем необходимо добавить скрытое поле, задающее максимальный размер отправляемого файла, с именем MAX_FILE_SIZE. Значение этого поля (т.е. максимально возможный размер файла) определяется в байтах.

Форма, из которой будет отправляться файл, должна кодировать данные по методу multipart/form-data и передавать данные только методом POST.

```

<!-- Форма для отправки файла на сервер -->
<form name = "file_upload" action="photo.php"
        enctype="multipart/form-data" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="65536" />
<input type="file" name="file_upload" />
<input type="submit" name="submit" value="Добавить" />
</form>

```

В данном случае максимальный размер файла равен 65536 байта, т.е. 64 Кбайта. Данный размер при необходимости может быть увеличен.

2. Получение отправленного файла. Все принятые файлы помещаются интерпретатором PHP в особую служебную папку, которая не является частью сайта (т.н. «буферная» папка). Сведения обо всех принятых и помещенных в буферную папку файлах хранятся во встроенном массиве PHP \$_FILES. Каждый элемент массива соответствует принятому файлу и представляет собой вложенный массив, содержащий различные сведения о файле. В начало страницы поместите следующий код:

```

<?php
//Сценарий отправки файла на сервер
//Проверяем, была ли выполнена отправка файла. Далее реализуем
сценарий.
if (isset($_POST["MAX_FILE_SIZE"]))
{
    $tmp_file_name = $_FILES["file_upload"]["tmp_name"];
    $dest_file_name = $_SERVER['DOCUMENT_ROOT'].
        "/photo/".$_FILES["file_upload"]["name"];
    move_uploaded_file($tmp_file_name, $dest_file_name);
}
?>

```

Функция ***move_uploaded_file (string filename, string destination)*** проверяет, является ли файл *filename* загруженным на сервер (переданным по протоколу HTTP POST). Если файл действительно загружен на сервер, он будет перемещён в место, указанное в аргументе *destination*.

Упражнение 3: Удаление файла с сервера

Для удаления файла надо выбрать необходимый файл. Это делается с помощью формы, в которую помещается список для выбора удаляемого

файла из имеющихся на сервере файлов. Для заполнения этого списка необходимо использовать уже созданный массив `$array_files`.

Реализация удаления происходит в два этапа:

- Создание формы для удаления файла;
- Создание сценария удаления файла.

1. Создание формы для выбора удаляемого файла

```
<!-- Форма для удаления файла с сервера -->
<form name="file_delete" action="photo.php" method="post"
      enctype="application/x-www-form-urlencoded">
Файл <select name = "file_delete" size="1">
      <option><option></select>
<input type="submit" name="submit" value="Удалить" />
</form>
```

Парные тэги SELECT создает список. Пункты списка создаются тэгами OPTION. Сам список имеющихся на сервере файлов необходимо получить из созданного ранее массива `$array_files`.

```
<!-- Форма для удаления файла с сервера -->
<form name="file_delete" action="photo.php" method="post"
      enctype="application/x-www-form-urlencoded">
Файл <select name = "file_delete" size="1">
<?php for ($i=0; $i<$array_files_count; $i++) { ?>
<option><?php echo $array_files[$i]; ?></option>
<?php } ?></select>
<input type="submit" name="submit" value="Удалить" />
</form>
```

Во включенном в форму цикле создается столько пунктов (тэгов `<option>`), сколько элементов присутствует в массиве `$array_files`.

2. Создание сценария удаления файла.

```
<?php
//Сценарий удаления файла
//Сначала проверяем, было ли запущено удаление файла
if (isset($_POST["file_delete"]))
{
```



```

        //Формируем полное имя файла
        $file_name = $_SERVER['DOCUMENT_ROOT'] . "/photo/" .
            $_POST["file_delete"];

        //Функция unlink() удаляет файл
        unlink($file_name);
    }
?>

```

3. Сохраните изменения, проверьте корректность работы.

Упражнение 4: Создание страницы работы с файлами

Самостоятельно создайте страницу для управления прочими файлами **files.php**. Она во всем аналогична странице **photo.php**, кроме:

1. Она должна обрабатывать файлы, находящиеся в папке **files**.
2. Просмотр файлов делать необязательно (это не изображения), следовательно не надо делать гиперссылки из имени каждого файла.
3. Необходимо увеличить максимальный размер файла хотя бы до 1 Мбайта.
4. Сделайте гиперссылки, связывающие данную страницу с главной страницей.

Лабораторная работа №10: Автоматизация работы средствами инструментальной среды Adobe Dreamweaver. Разграничение доступа к разделам сайта

В ходе выполнения данной лабораторной работы будут продемонстрированы возможности Adobe Dreamweaver для автоматизации создания часто используемых при создании web-приложения php-скриптов.

Упражнение 1: Автоматизация размещения данных на странице

1. Создайте на локальном узле (в нашем случае в папке C:\WebServers\home\localhost\www\) папку Dreamweaver. Создайте файл main.php – на нем разместите только элементы, указанные на *рис. 10.1*. Этот файл будет уменьшенной «копией» файла blog.php.

Все файлы, создаваемые в этой практической работе необходимо также сохранять в папке Dreamweaver.

Привет!

Это мини-вариант моего сайта о путешествиях!

[Вход](#)

[Добавить заметку](#)

[Администратору](#)

[Выход](#)

Рис. 10.1. Схема файла *main.php*

2. Автоматически создайте еще одно соединение с сервером (**Базы данных** – «+» - **Соединение MySQL**), сохраните все настройки **Dreamweaver**, назовите соединение **Dreamweaver** (рис. 10.2).

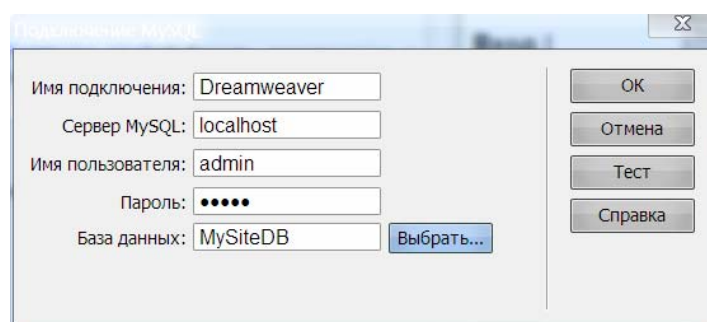


Рис.10.2. Окно создания соединения с сервером

3. Откройте файл с кодом соединения (**Connections/Dreamweaver.php**), добавьте в конец скрипта строки с кодировкой кириллицы:

```
mysql_query("SET NAMES cp1251;", $Dreamweaver) or die(mysql_error());  
mysql_query("SET CHARACTER SET cp1251;", $Dreamweaver) or  
die(mysql_error());
```

4. В результате у вас появится соединение с БД “*MySiteDB*” под названием **Dreamweaver**. Все соединения Dreamweaver автоматически размещает в одной папке с соответствующим названием.
5. Страница **main.php** является аналогом страницы **blog.php** , но создавать мы ее будем целиком средствами автоматизации **Dreamweaver**. Она также

будет содержать список записей блога, кроме того, мы разместим на ней некоторые дополнительные элементы. Этапы работы:

- Автоматическое создание набора записей (рекордсета), т.е. массива, в который помещаются записи из БД.
- Автоматическое размещение записей на странице сайта.

⇒ Помните: большую часть действий необходимо совершать в режиме дизайна.

6. Создание набора записей (рекордсета) страницы **main.php**. Для вывода записей из БД на страницу блога надо создать рекордсет (набор записей). Таким образом, записи из БД помещаются в автоматический созданный массив.

6.1. Выберите вкладку **Привязки** – «+» - **Набор записей (Запрос)**.

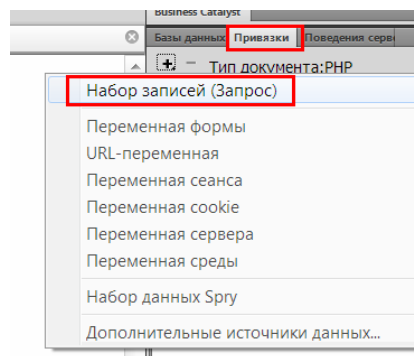


Рис. 10.3. Меню Привязки

6.2. Откроется окно автоматического создания набора записей. Заполните поля окна (рис. 10.4):

A screenshot of the 'Набор записей' (Recordset) dialog box. The 'Имя' (Name) field contains 'notes'. The 'Подключение' (Connection) dropdown is set to 'Dreamweaver'. The 'Таблица' (Table) dropdown is set to 'notes'. The 'Столбцы' (Columns) section has radio buttons for 'Все' (All) and 'Выбрано' (Selected), with 'Все' selected. Below this, a list of columns is shown: 'id', 'created', 'title', and 'article'. The 'Фильтр' (Filter) section has a dropdown set to 'Нет' (None) and an equals sign. The 'Сортировка' (Sorting) section has a dropdown set to 'created' and another set to 'по возрастанию' (ascending). On the right side, there are buttons for 'ОК', 'Отмена', 'Тест', 'Дополнительно...', and 'Справка'.

Рис. 10.4. Окно автоматического создания набора записей

7. Далее необходимо разместить заметки на странице сайта. **Помните, что автоматическое размещение элементов на странице ВСЕГДА проводится в режиме Дизайна.** Для этого мышкой перетащите составляющие рекордсета с вкладки Привязки на страницу сайта. Разместите их так, как считаете нужным (рис. 10.5).

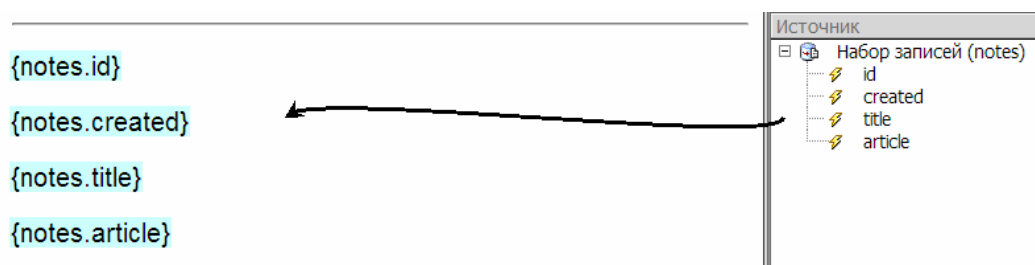


Рис. 10.5. Размещение составляющий рекордсета на странице в режиме дизайна

8. Создание повторяющейся области. На данный момент выводится только первая заметка сайта. Необходимо сделать так, чтобы выводились все заметки, имеющиеся в базе. Для этого надо организовать повторяющуюся область:

- 8.1. В режиме дизайна выделите те элементы, которые необходимо повторять (в нашем случае – все элементы рекорсета).
- 8.2. Выберите **Поведение сервера** – «+» - **Повторить область**.
- 8.3. Установите вывод 10 записей на странице за один раз.
- 8.4. В результате на странице обозначится повторяющаяся область.

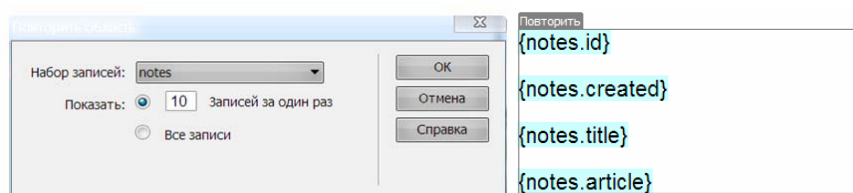


Рис. 10.6. Представление повторяющейся области

Упражнение 2: Создание постраничного навигатора

Ранее было установлено условие – вывод не больше 10 заметок на страницу. Если заметок в базе больше, то надо создать возможность переходов между страницами вывод заметок.

1. Добавьте в базу больше 10 заметок.

2. Создайте на странице **main.php** абзац с набором навигационных гиперссылок (рис. 10.7):

Привет!

Это мини-вариант моего сайта о путешествиях!

На главную | Вперед | Назад | На последнюю

[Вход](#)

[Добавить заметку](#)

[Администратору](#)

[Выход](#)

Рис. 10.7. Абзац с набором навигационных гиперссылок

3. Для создания гиперссылок навигатора необходимо выделить слово/фразу (например, «на главную»), выбрать на панели **Поведение Сервера** знак «+», подменю **Разбиение набора записей по страницам** и выбрать команду «Разбиение набора записей по страницам».
4. Создайте все необходимые ссылки навигатора. Проверьте корректность их работы.
5. Далее необходимо создать на странице **main.php** информацию об общем количестве заметок в наборе записей и о количестве заметок, отображающихся в данный момент на конкретной странице. Принцип реализации аналогичен принципу разбиения на страницы:
- 5.1. Разместите внизу страницы запись «Заметки с... по ... из...»
- 5.2. Далее: вкладка **Поведение сервера – Отобразить счетчик записей**.
- 5.3. Вставьте соответствующие записи.

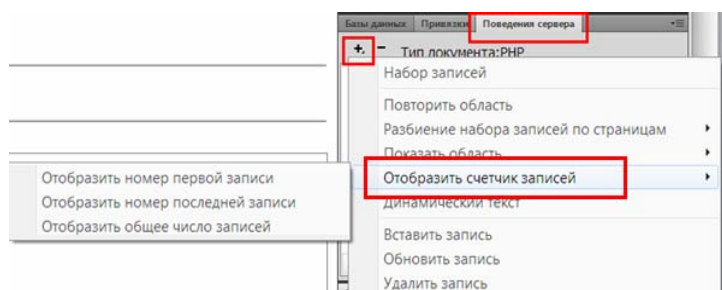


Рис. 10.8. Отображение счетчика записей

Упражнение 3: Обмен данными между страницами

1. Создайте страницу для вывода комментариев **comm.php**.
2. Создайте ссылки между страницами (помните, что мы работает с новыми страницами: **main.php** и **comm.php**). Для перехода с главной страницы на страницу комментариев ссылка – это заголовок заметки.
3. Вручную организуйте передачу номера заметки между страницами, при этом для корректной работы обращайте внимание, какие переменные ввел автоматически Dreamweaver и используйте их. Для передачи номера заметки в гиперссылку на странице **main.php** добавьте идентификатор **note** с соответствующим значением. На странице комментариев он будет получен автоматически.
4. На странице **comm.php** создайте два рекордсета (*рис. 10.9*): один (**notes**) для вывода комментируемой заметки, другой с именем **comments** для вывода комментариев к заметке.
5. Обратите внимание на Фильтр «Параметр URL» - в нем мы автоматически получаем идентификатор заметки **note**, переданный в ссылке.

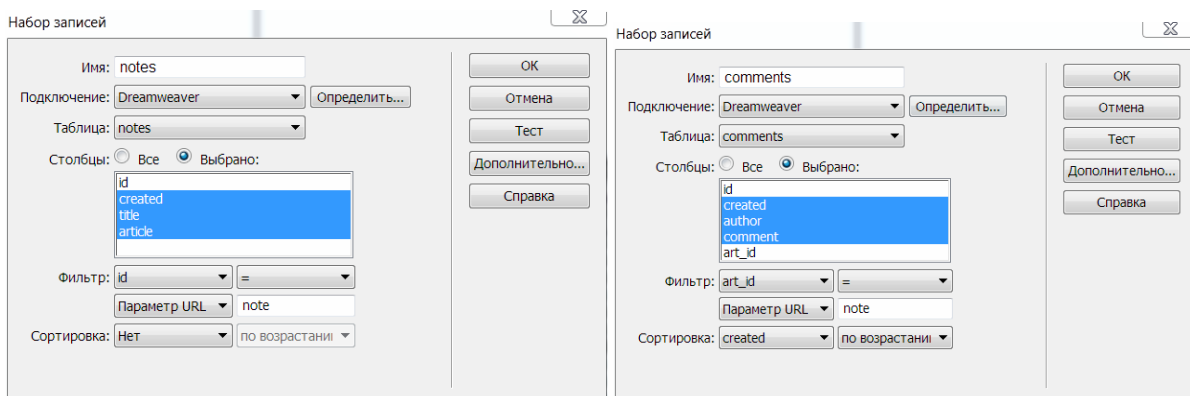


Рис. 10.9. Создание рекордсетов notes и comments

6. Перетащите на страницу все необходимые поля.
7. Организуйте для комментариев повторяющуюся область.

Упражнение 4: Страница добавления заметки

1. Создайте страницу **addnew.php** для автоматического добавления пользователем новой заметки.
2. Создайте на странице форму (меню **Вставка – Форма**). Имя формы – **addnote**.

3. Поставьте в форму текстовый курсор. Внизу страницы появится редактор форм (если не откроется автоматически, то выберите меню **Окно – Свойства**). Заполните форму:

Рис. 10.10. Окно свойств формы

4. Добавьте на форму текстовое поле (**Вставка – Форма – текстовое поле**). **Id** и **name** поля - *title*; надпись – *Заголовок*.
5. Установите свойства текстового поля:

Рис. 10.11. Окно свойств текстового поля

6. Аналогичным образом создайте в форме область текста *Содержание* с названием *article*, со следующими свойствами:

Рис. 10.12. Окно свойств области текста

7. Далее вручную на форме необходимо добавить скрытое поле с именем *created*, получающее автоматически дату создания заметки (функция *date()*).
8. В завершении на форму добавьте кнопку **Submit** (**Вставка – Форма - Кнопка**).

Добавить новую заметку:

Заголовок

Содержание

Добавить

Рис. 10.13. Форма добавления новой заметки

9. Для автоматического добавления в базу данных новой заметки выберите вкладку **Поведение сервера** – “+” – **Вставить запись**. Заполните открывшуюся форму:

Рис. 10.14. Форма автоматизации добавления записей в базу

10. Проверьте работу формы (отображение введенных через форму данных в БД “MySiteDB”). Убедитесь, что данные отображаются корректно, текущая дата принимается автоматически.
11. Создайте гиперссылку со страницы **main.php** на **addnew.php**.
12. Обновление и удаление записей также автоматически реализуется с помощью соответствующих команд меню. Вручную надо только передать идентификатор заметки и значение полей, передающихся в форму.

Упражнение 5: Разграничение доступа к данным

Для создания разграничения доступа к страницам сайта необходимо решить следующие задачи:

- Создать в БД MySiteDB новую таблицу со списком пользователей «privileges». Эта таблица будет включать поля, содержащие имена пользователей, пароли и сведения о правах доступа к различным страницам сайта.
- При попытке пользователя просмотреть закрытые страницы сайта спросить у него имя пользователя и пароль.
- Если имя и пароль, введенные посетителем, есть в БД, то перенаправить пользователя на главную страницу сайта и сделать доступными имеющиеся ссылки.
- Если введенные имя пользователя и пароль в БД отсутствуют, сделать доступной для пользователя только главную страницу сайта; гиперссылки для него будут недоступны.
- Предоставить зарегистрированным пользователям возможность корректного выхода с сайта.
- Отнести всех зарегистрированных пользователей к двум категориям: users(u), administrators(a). Для администраторов должны быть доступны все страницы сайта, для простых пользователей (users) часть страниц закрыты.

1. Создание таблицы списка пользователей. Создайте в БД «MySiteDB» таблицу «privileges», см. *Приложение 3*.
2. Добавьте в таблицу единственную запись, задающую администратора (например, admin, admin, a).
3. Создайте страницу входа на сайт **login.php**. На этой странице необходимо создать форму для ввода имени пользователя и пароля. Для этого:
 - 3.1. Создайте форму с именем **login**.
 - 3.2. В созданную форму поместите два текстовых поля ввода и кнопку. Параметры первого поля: имя – **name**, ширина **20**, максимальное количество символов **20**. Параметры второго поля: имя – **password**, ширина и максимальное количество символов – **20**, тип – **поле ввода пароля**. Кнопка: имя – **submit**, надпись – **«Войти»**.
 - 3.3. Далее выберите **Поведение сервера – «+» - Проверка подлинности пользователя – Вход пользователя в систему**. Заполните появившееся окно:

- Получить данные из формы: **login**;
 - Поле имени пользователя: **name**;
 - Поле пароля: **password**;
 - Проверка с помощью подключения: **Dreamweaver**;
 - Таблица: **privileges**;
 - Столбец имени пользователей: **name**;
 - Столбец паролей: **password**;
 - Если войти удалось, перейдите к: **main.php**
 - Если не удалось войти, перейдите к: **main.php** (если пользователь не прошел проверку, он имеет право доступа только к главной странице. Вариант: создать страницу-«заглушку», на которую в этом случае будет перенаправлен пользователь).
 - Ограничение доступа на основе: **Имя пользователя, пароль и уровень доступа.**
 - Получить уровень доступа из: **rights**.
4. Создайте на странице **main.php** гиперссылку на страницу **login.php** (в меню всех гиперссылок слово «Вход»).
 5. Необходимо ограничить вход на страницу **addnew.php** незарегистрированным пользователям.
 - 5.1. Откройте страницу **addnew.php**.
 - 5.2. **Поведение сервера — «+» - Проверка подлинности пользователя — Ограничение доступа к странице.**
 - 5.3. Заполните появившееся диалоговое окно:

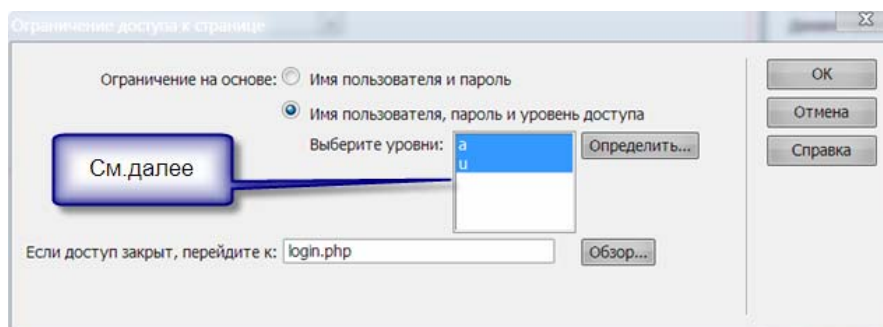


Рис. 10.15. Форма ограничения доступа к странице

6. Определение уровней доступа. Права доступа «**u**» и «**a**» добавьте самостоятельно вручную, нажав кнопку на форме «*Определить...*». Откроется диалоговое окно определения уровней доступа (рис. 10.16). Введите информацию по правам доступа, используя «+» ядл добавления обозначений.

После создания уровней доступа не забудьте в окне «Ограничение доступа к странице» (предыдущее окно) выбрать оба пункта - «**u**» и «**a**», чтобы разрешить доступ к данной странице как зарегистрированным пользователям, так и администраторам.

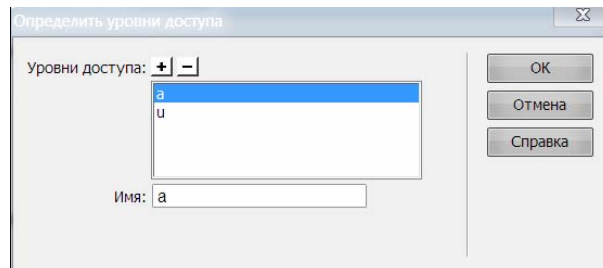


Рис. 10.16. Диалоговое окно определения уровней доступа

7. Создайте страницу **logout.php** – страницу выхода с сайта. Создайте на ней пояснительный текст. Для выхода с сайта пользователь должен будет щелкнуть расположенную на этой странице гиперссылку (это также дает пользователю возможность передумать и вернуться к работе с сайтом под своим логином и паролем).
8. Создайте на странице **logout.php** гиперссылку «Выход» и выделите ее. Далее **Поведение сервера** – «+» - **Проверка подлинности пользователя** – **Выход пользователя из системы**. Появится окно **Log Out User**, в нем необходимо задать настройки момента выхода с сайта:



Рис. 10.17. Окно организации выхода из системы

На этой же странице задайте гиперссылку возврата на главную страницу (для того, чтобы пользователь мог вернуться не выходя из-под своей учетной записи).

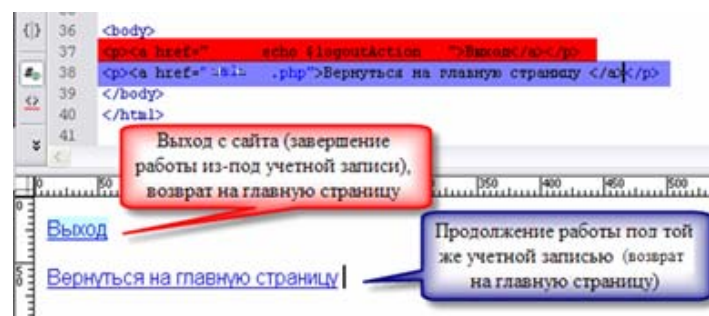


Рис. 10.18. Схема расположения гиперссылок

9. Создайте на странице **main.php** ссылку «Выход» (надпись уже есть с списке ссылок) на страницу **logout.php**.
10. Проверьте корректность работы сценариев.

Упражнение 6: Создание административных страниц для управления пользователями

Необходимо создать две административные страницы:

- страницу списка пользователей (**users.php**),
 - страницу для добавления нового пользователя (**adduser.php**).
1. Чтобы отобразить список пользователей на странице **users.php**, можно создать таблицу, состоящую из двух столбцов (имя пользователя, права доступа). Пароль не выводится, т.к. является конфиденциальной информацией. В таблицу поместите самостоятельно созданный набор записей (Привязки – Набор записей). Не забудьте организовать повторяющуюся область (Поведение сервера – Повторить область). Добавьте еще несколько пользователей в базу данных. Проверьте работу сценария.

Список пользователей:

Повторить	Имя	Права доступа
	{users.name}	{users.rights}

Список пользователей:

Имя	Права доступа
admin	a
Вася	u
Петя	u

Рис. 10.19. Организация списка пользователей.

2. Также добавьте ссылку на страницу **adduser.php** для добавления нового пользователя (страницу создадим на следующем шаге) и ссылку возврата на главную страницу сайта.
3. Для добавления пользователя на странице **adduser.php** создайте форму с необходимыми полями для ввода имени, логина и прав доступа (a/u) и модель серверного поведения **Вставить запись**.

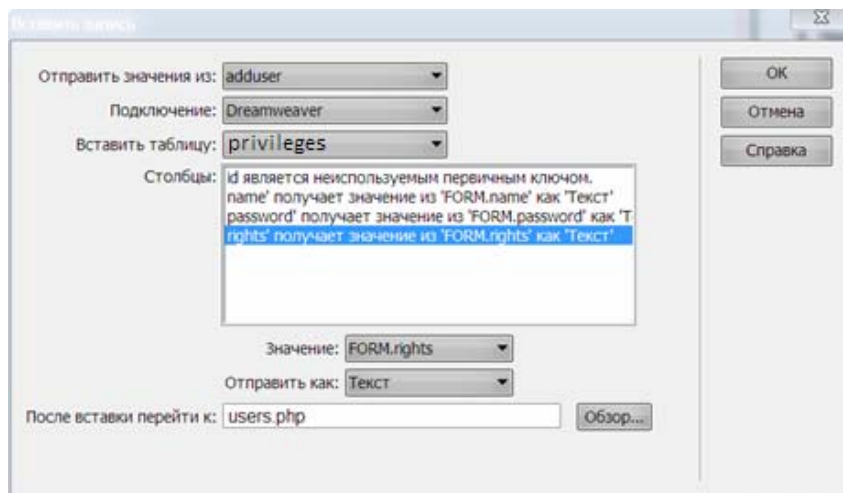


Рис. 10.20. Окно работы с записями

4. Для проверки работы сценария добавьте нового пользователя (одного или нескольких) с правами пользователя (u).

Упражнение 7: Административная часть сайта

1. На странице **main.php** сделайте гиперссылку на страницу **users.php** на надписи «Администратору».
2. Далее создайте .html страницу (например, rights.html) с тестом: «Извините, у вас нет прав доступа для просмотра данной страницы» и с гиперссылкой возврата на главную страницу сайта.
3. Организуйте доступ на страницу **users.php** только для тех, у кого административные права (т.е. права доступа «a»):

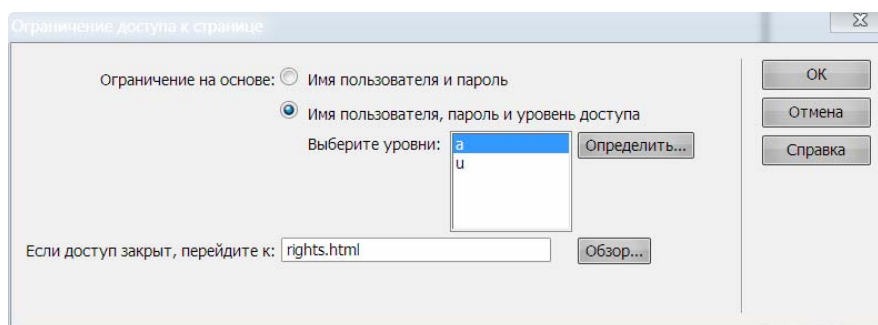


Рис. 10.21. Форма ограничения доступа к странице

4. Проверьте корректность выполнения сценариев. Зайдите на сайт под пользователем с пользовательскими (не администраторскими) правами доступа и попробуйте перейти по ссылке «Администратору». Убедитесь, что для пользователя вход на страницу закрыт. Зайдите на сайт с правами администратора и убедитесь, что имеете право доступа к данной странице.

Литература

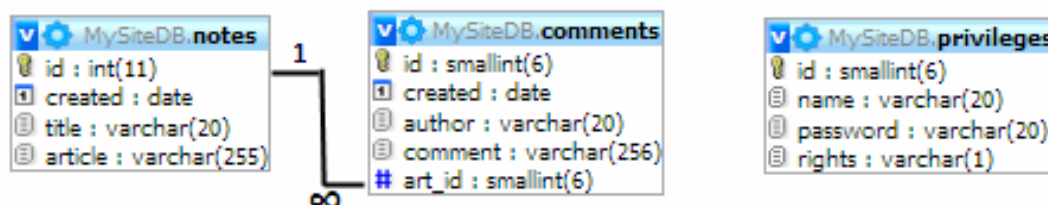
1. Вэллинг Л., Томсон Л. Разработка веб-приложений с помощью PHP и MySQL. М.: Вильямс, 2010. – 848 с.
2. Дамашке Г. PHP и MySQL. М.: ИТ-Пресс, 2012 – 320 с.
3. Дронов В. PHP, MySQL и Dreamweaver. Разработка интерактивных Web-сайтов. СПб.: БХВ – Петербург, 2007. – 478 с.
4. Зандстра М. PHP. Объекты, шаблоны и методики программирования. – М.: Вильямс, 2011. – 560 с.
5. Зервас К. Web 2.0. Создание приложений на PHP. М.: Вильямс, 2009. – 544 с.
6. Колисниченко Д.Н. PHP 5/6 и MySQL 6. Разработка Web-приложений. СПб.: БХВ-Петербург, 2010. – 560 с.
7. Кузнецов М., Симдянов И. PHP 5/6. СПб.: БХВ-Петербург, 2009. – 1024 с.
8. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. СПб.: ПИТЕР, 2011. – 496 с.
9. Пауэрс Д. Adobe Dreamweaver, CSS, Ajax и PHP. .: БХВ-Петербург, 2009. – 1056 с.
10. Пауэрс Д. PHP. Создание динамических страниц. М.: Рид Групп, 2012 – 640 с.

Приложение 1. Схема сайта «MyTravelNotes»



В ходе последовательного выполнения лабораторных работ создаются несколько статических страниц, служащих основой сайта, затем разрабатываются динамические страницы для обмена данными с базой данных MySQL, далее статические страницы преобразуются в динамические, добавляются динамические страницы для работы с заметками, комментариями, работы с почтой, обмена файлами с сервером и т.п. Конечным этапом работы является создание страниц и скриптов для организации разграничения доступа пользователей к сайту в целях обеспечения безопасности.

Приложение 2. Схема базы данных «MySiteDB»



1. Таблица «notes» содержит информацию о заметках на сайте;
2. Таблица «comments» содержит информацию о комментариях к заметкам;
3. Таблица «privileges» содержит информацию о пользователях и их правах доступа к информации на сайте.

Приложение 3. Структура таблиц базы данных

Таблица 1. Структура таблицы notes

Имя поля	Описание поля	Тип данных	Другое
id	Идентификатор записи	SMALLINT	Ключевое поле (AUTO_INCREMENT), INDEX – PRIMARY.
created	Дата создания заметки	DATE	
title	Заголовок заметки	VARCHAR (50)	Строка фиксированной длины 50 символов
article	Содержимое заметки	VARCHAR (255)	Строка фиксированной длины 255 символов

Основные атрибуты полей

PRIMARY – определяет поле, являющееся первичным ключом.

UNSIGNED – запрещает числовым полям принимать отрицательное значение.

AUTO_INCREMENT (A_I) – поле счетчика (отдельный тип данных «счетчик» отсутствует).

BLOG – позволяет задать значение поля по умолчанию.

NOT NULL – определяет обязательное для заполнения поле.

Таблица 2. Структура таблицы comments

Имя поля	Описание поля	Тип данных	Другое
id	Идентификатор записи	SMALLINT	Ключевое поле (AUTO_INCREMENT), INDEX – PRIMARY.
created	Дата публикации заметки	DATE	
author	Автор комментария	VARCHAR (20)	
comment	Содержимое комментария	VARCHAR (255)	Строка фиксированной длины 255 символов
art_id	Внешний ключ для «привязки» комментария к заметке в таблице Notes	SMALLINT	

Таблица 3. Структура таблицы privileges

Имя поля	Описание поля	Тип данных
id	Id записи	SMALLINT, ключевое поле, AUTO_INCREMENT
name	VARCHAR (20)	Имя пользователя
password	VARCHAR (20)	Пароль пользователя
rights	VARCHAR(1)	Права доступа пользователя к страницам сайта

Последнее поле rights может содержать только один символ, обозначающий права доступа:

a – права доступа администратора;

u – права доступа пользователя.

Приложение 4. Основные сведения о работе с базой данных

База данных - структурированное хранилище некоторого набора данных определенной предметной области в именованной области жесткого диска.

СУБД (Система Управления Базой данных) - это программное обеспечение, имеющее средства обработки на языке БД и управляющее доступом к БД. Оно хранит данные и предоставляет пользователям возможность извлечения и модификации этих данных.

Реляционные БД хранят информацию в виде связанных друг с другом таблиц. Таблица состоит из полей (столбцов), записей (строк) и ячеек.

Обязательным элементом таблицы является ключ. Ключ – это поле, которое однозначно определяет значения всех остальных полей в таблице. Ключ всегда уникален, т.е. в любой момент времени таблица БД не может содержать никакие две различные записи, имеющие одинаковые значения ключевых полей.

В реляционной базе данных таблицы связаны друг с другом. Связи бывают трех типов:

- Один-к-одному;
- Один-ко-многим;
- Многие-ко-многим.

Основные понятия языка SQL

SQL —(Structured Query Language) это язык структурированных запросов. SQL был специально разработан для взаимодействия с базами данных.

Основные операторы определения *структуры данных*:

- CREATE (создание);
- ALTER (изменение);
- DROP (удаление).

Основные операторы работы с *данными*:

- SELECT (выборка);
- INSERT (вставка);
- UPDATE (обновление);
- DELETE (удаление).

Основные функции PHP для работы с MySQL

При работе с БД **в первую очередь** устанавливается соединение с сервером.

Функция соединения с сервером (параметры функции: имя сервера, имя пользователя, пароль):

//Присваивание значений переменным, необходимым для подключения к серверу: имя сервера, имя пользователя, пароль.

\$hostname = 'localhost';

\$user = 'administrator';

\$password = 'admin_password';

*//Функция установки соединения с серверу **mysqli_connect()**. Параметры функции: имя сервера, имя пользователя, пароль.*

*\$link = **mysqli_connect** (\$hostname, \$user, \$password);*

Вариант с литеральными строками:

*\$link = **mysqli_connect** ('localhost', 'administrator', 'admin_password');*

Важно: пароль является **обязательным** параметром функции **mysqli_connect()**. Если пароля нет, то ему присваивается пустое значение:

\$hostname = 'localhost';

\$user = 'administrator';

\$password = '';

*\$link = **mysqli_connect** (\$hostname, \$user, \$password);*

Вариант с литеральными строками:

*\$link = **mysqli_connect** ('localhost', 'administrator', '');*

Выбор базы данных

После установки соединения с сервером необходимо выбрать БД для дальнейшей работы.

Функция выбора БД для дальнейшей работы (параметры функции: имя соединения с сервером, имя БД):

***mysqli_select_db**(\$link, \$database);*

Вариант с литеральными строками:

***mysqli_select_db**(\$link, 'MyDatabase');*

Выполнение запросов MySQL

Функция PHP **mysqli_query()** позволяет использовать основные операторы языка SQL (SELECT, INSERT, UPDATE, DELETE) для обращения к базе данных.

Параметрами функции **mysqli_query()** являются соединение с сервером и SQL-запрос.

Например:

```
//соединение с сервером  
$link = mysqli_connect ($hostname, $user, $password);  
//формирование SQL-запроса  
$query = "CREATE DATABASE MyDatabase";  
//Выполнение функции mysqli_query() для реализации запроса  
$result = mysqli_query($link, $query);
```

Или

```
$link = mysqli_connect ($hostname, $user, $password);  
$result = mysqli_query($link, "CREATE DATABASE MyDatabase");  
//Здесь сам запрос передается непосредственно в функцию, без  
«промежуточной» переменной $query (как в предыдущем примере). Первый  
вариант предпочтительней, т.к. облегчает читаемость кода (SQL-запросы  
могут быть длинными).
```