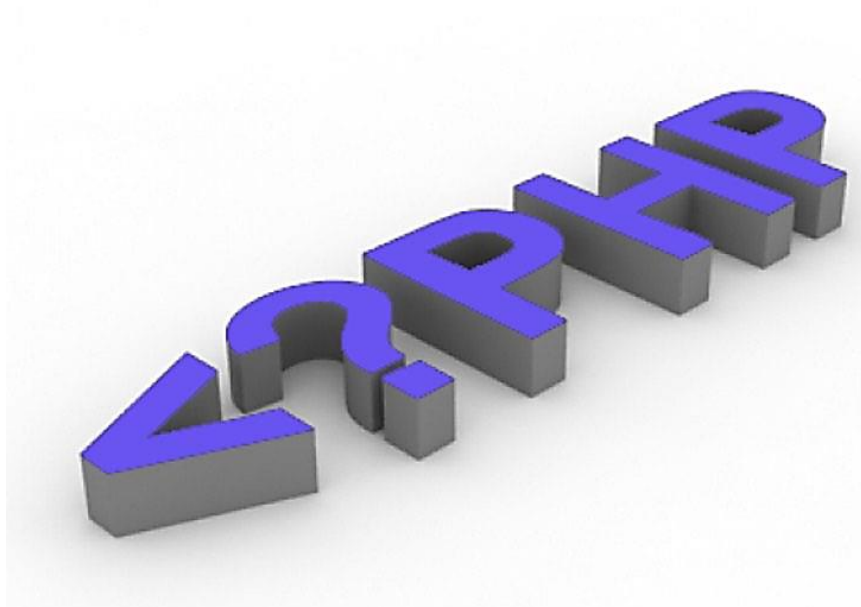


С.В. Одиночкина

Основы разработки приложений на PHP

Практикум



Санкт-Петербург

2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
УНИВЕРСИТЕТ ИТМО

С.В. Одиночкина

Основы разработки приложений на PHP

Практикум

 **УНИВЕРСИТЕТ ИТМО**

Санкт-Петербург

2016

Одиночкина С.В., Основы разработки приложений на PHP. Практикум. – СПб: Университет ИТМО, 2016. – 66 с.

Основной целью пособия является изучение базовых подходов к реализации серверной части web-приложения. Данный курс позволит изучить основные возможности использования языка PHP, обеспечить взаимодействие с данными, а также реализовать формирование динамического контента web-приложения.

Предназначено для подготовки бакалавров по направлению «11.03.02 Информационные технологии и системы связи» по бакалаврским программам: «Инфокоммуникационные системы» и «Облачные технологии».

Рекомендовано к печати Ученым советом факультета инфокоммуникационных технологий, протокол № 09/16 от 24.11.2016г.



Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

© Университет ИТМО, 2016

© С.В. Одиночкина, 2016

Оглавление

Введение.....	5
1. Настройка инструментальной среды.....	7
1.1 Первая программа на PHP.....	7
1.2 Простейшие программы на PHP.....	8
1.3 Создание статической основы web-страниц.....	9
2 Создание базы данных MySQL.....	10
2.1 Создание БД «MySiteDB».....	10
2.2 Создание пользователя admin.....	12
2.3 Создание таблицы notes.....	13
2.4 Создание таблицы comments.....	14
2.5 Создание таблицы authors.....	16
2.6 Создание межтабличных связей.....	16
2.7 Заполнение таблиц.....	18
2.8 Файл подключения базы данных.....	18
3 Простой вывод данных. Страницы default.php и comments.php.....	20
3.1 Вывод данных из базы на страницу.....	20
3.2 Обмен данными между серверными страницами.....	22
4 Ввод и обработка данных (элементы HTML-форм).....	25
4.1 Отправка почты.....	25
4.2 Страница для добавления заметок.....	25
4.3 Обработка HTML-переключателей.....	29
4.4 Обработка HTML-флагов.....	30
5 Модификация данных.....	31
5.1 Страница для редактирования заметок.....	31
5.2 Создание страницы удаления заметок.....	35
6 Работа с заметками.....	36
6.1 Работа со страницей default.php.....	36
6.2 Работа с комментариями к заметкам.....	37
7 Страница статистики inform.php.....	37
7.1 Общее количество заметок и общее количество комментариев.....	37
7.2 Подсчет количества заметок и комментариев за последний месяц.....	38
7.3 Последняя добавленная заметка.....	39
7.4 Самая комментируемая заметка.....	40
7.5 Размещение данных на странице.....	40
8 Реализация поиска по сайту.....	43
8.1 Реализация поиска по сайту.....	43
8.2 Обработка строки поиска.....	46
9 Работа с файлами.....	48
9.1 Вывод списка файлов.....	48

9.2 Загрузка файлов на сервер	51
9.3 Удаление файла с сервера	53
9.4 Работа с содержимым файла	54
10 Основы разграничения прав доступа	56
Литература	60
Приложение 1. Схема сайта «MyTravelNotes»	61
Приложение 2. Схема базы данных «MySiteDB»	62
Приложение 3. Структура таблиц базы данных	63

Введение

В результате изучения курса, проводимого под руководством преподавателя, студенты овладеют базовыми теоретическим знаниями и практическими навыками, необходимыми для разработки веб-приложений на языке программирования PHP, а также навыками проектирования баз данных и работы с системой управления базами данных MySQL.

Цель курса

Целью курса является изучение основных возможностей языка программирования PHP и принципов взаимодействия с базами данных на примере MySQL.

После изучения курса слушатели смогут:

- разрабатывать базовые веб-приложения;
- реализовывать подключение веб-приложения к базе данных с целью хранения и обмена информацией между базой данных и приложением;
- работать с веб-интерфейсом MySQL PhpMyAdmin;
- использовать методы GET и POST для передачи и обмена данными;
- использовать HTML-формы для обеспечения ввода, вывода и обработки данных веб-приложения;
- реализовывать работу с файлами и каталогами;
- использовать основные принципы администрирования веб-приложения.

В ходе работы необходимо разработать сайт под названием «MyTravelNotes», содержащий записи автора сайта о его путешествиях, а также базу данных (БД) под названием «MySiteDB», содержащую контент сайта. Данная задача включает в себя реализацию следующих функций веб-проекта:

1. Возможность добавления записей автора и комментариев к ним (при этом все заметки и комментарии должны передаваться и храниться на сервере в БД);
2. Возможность модификации и удаления заметок и комментариев в БД посредством форм сайта;

3. Обеспечение защиты данных с помощью логина и пароля, разграничение доступа к данным с учетом установленного уровня доступа (администратор и пользователь);
4. Данные о пользователях и их уровне доступа должны храниться на сервере в БД;
5. Обеспечение корректного входа и выхода с сайта;
6. Реализацию возможности обратной связи посетителей сайта с автором блога;
7. Обеспечение дружелюбного пользовательского интерфейса и корректной организации навигации по разделам сайта.

Кроме того, при разработке данного программного проекта должны учитываться основные принципы web-дизайна (юзабилити) для удобства работы с сайтом конечного пользователя.

1 Настройка инструментальной среды

В данной лабораторной работе рассматриваются основные настройки по умолчанию набора дистрибутивов, необходимых для разработки серверных приложений с помощью языка программирования PHP, в также выполняются простейшие программы на языке PHP.

Имя создаваемого в ходе выполнения лабораторных работ проекта - «*MyTravelNotes*». Расположение данных по умолчанию (без учета особенностей установки в конкретных случаях):

Сборка	Путь к проекту	Путь к БД
Denwer	C:\WebServer\home\localhost\www	C:\WebServers\usr\local\mysql-5.5\data\ИмяВашейБД
Open Server	C:\OpenServer\domains\localhost\	C:\OpenServer\userdata\MySQL-5.6\ИмяВашейБД

URL-адрес для просмотра и работы в браузере: *http://localhost/ПутьКФайлу*

1.1 Первая программа на PHP

1. Запустите виртуальный сервер, если он еще не запущен.
2. При работе с **Open Server** создайте папку *MyTravelNotes* по адресу *C:\OpenServer\domains\localhost*.
Все файлы проекта будут сохраняться именно в ней, т.е. по адресу *C:\OpenServer\domains\localhost\MyTravelNotes*.
3. При работе с **Denwer** файлы проекта необходимо сохранять по адресу *C:\WebServer\home\localhost\www*.
4. Создайте в программе Notepad++ новую страницу.
5. Введите следующий код:

```
<html>
  <head>
    <title>Test page</title>
  </head>
  <body>
    <?php
      echo "Hello, world!";
    ?>
  </body>
</html>
```


6. Сохраните сценарий в папку сайта под именем **hello.php**. Путь:
C:\OpenServer\localhost\MyTravelNotes
или
C:\WebServer\home\localhost\www
для Open Server и Denwer соответственно.
7. Проверьте результат в браузере, введя в адресную строку браузера
localhost/MyTravelNotes/hello.php
или
localhost/hello.php
для Open Server и Denwer соответственно.
8. В дальнейшем все файлы проекта будут сохраняться и выполняться соответствующим образом (как указано в п.п.6 и 7), если не указано иное.

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.

1.2 Простейшие программы на PHP

Далее представлены дополнительные упражнения для закрепления навыков создания простейших программ на языке программирования PHP. Для их выполнения создайте новый `.php` файл **examples.php** и выполните предложенные ниже задания.

1. Переменной `$a` необходимо присвоить значение 10, переменной `$b` присвоить значение 20. Выведите значения переменных на экран.

```
<?php
    $a = 10;
    $b = 20;
    echo $a, $b;
?>
```

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.

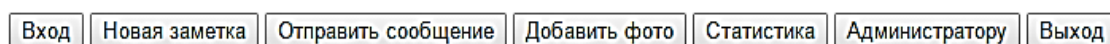
5. Введите новые переменные \$p и \$b. Присвойте переменной \$p значение «Программа», а переменной \$b значение «работает».
6. Затем сложите переменные, содержащие эти слова («Программа» и «работает»), при этом слова должны быть разделены пробелом (‘ ‘). Результат необходимо присвоить переменной \$result.
7. Далее с помощью оператора «.=» необходимо к строке «Программа работает» добавить слово «хорошо». Результат необходимо присвоить переменной \$result.

1.3 Создание статической основы web-страниц

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

На страницы необходимо добавить код HTML, который является основой для дальнейшей работы других технологий.

1. Создайте новую страницу **default.php**. На странице должно располагаться меню переходов между страницами и место размещения основного контента сайта (рисунок 1.1):



*Рад приветствовать вас
на страницах моего сайта, посвященного путешествиям.*

Рисунок 1.1 - Пример страницы default.php

2. Визуальная разметка страницы (оформление) может быть произвольная.
3. Сохраните страницу в папке проекта.
4. Создайте страницу статистики **inform.php**. Схема страницы (рисунок 1.2):

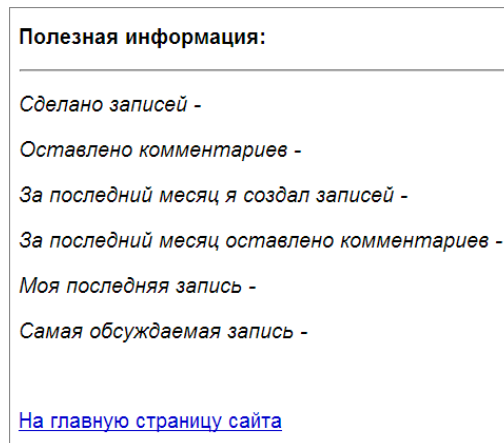


Рисунок 1.2 - Пример страницы *inform.php*

5. Сохраните страницу в папке проекта.
6. Свяжите гиперссылками созданные страницы.

2 Создание базы данных MySQL

В ходе выполнения данной лабораторной работы необходимо создать в MySQL новую базу данных с названием «*MySiteDB*» и добавить в нее три таблицы: **notes**, **comments** и **authors**. **Notes** содержит заметки блога; **comments** – комментарии к этим заметкам; **authors** – информацию о зарегистрированных пользователях. Схема данных (рисунок 2.1):



Рисунок 2.1 - Схема базы данных «*MySiteDB*»

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

2.1 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.
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 аргу-
ментов 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";
}
?>

```

2.2 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 прав доступа MySQL *Пользователи (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 *Пользователи (Privileges)*. Изучите список пользователей.

Способ 2: создание нового пользователя с помощью утилиты **PhpMyAdmin**

1. Запустите утилиту **PhpMyAdmin** и перейдите на вкладку *Пользователи (Privileges)*. Нажмите кнопку «Добавить пользователя (Add a new user)».
2. Введите имя пользователя (**admin**), хост - локальный (**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.

2.3 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 SMALLINT 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 SMALLINT 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";
}
?>

```

2.4 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 «Создать таблицу (*Create new table*)», присвойте имя таблице – **comments**; количество полей - **5**, нажмите кнопку «Go» (*рисунок 2.2*).

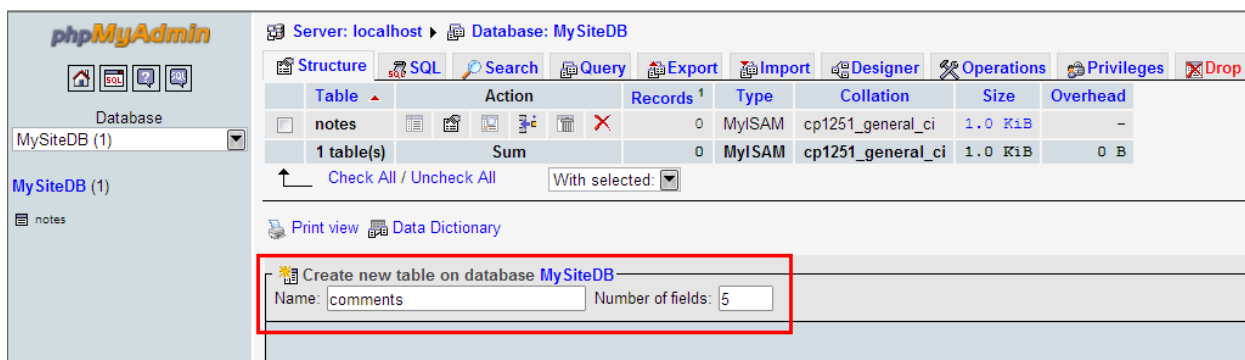


Рисунок 2.2 - Создание новой таблицы с помощью утилиты phpMyAdmin

4. Создание полей таблицы comments:

4.1. В открывшемся окне заполните необходимые поля таблицы (рисунок 2.3) и нажмите кнопку «Сохранить (Save)».

4.2. Для поля **id** добавьте следующие атрибуты: обозначьте автоинкремент **A_I** и первичный ключ **PRIMARY** в поле со списком **INDEX**.

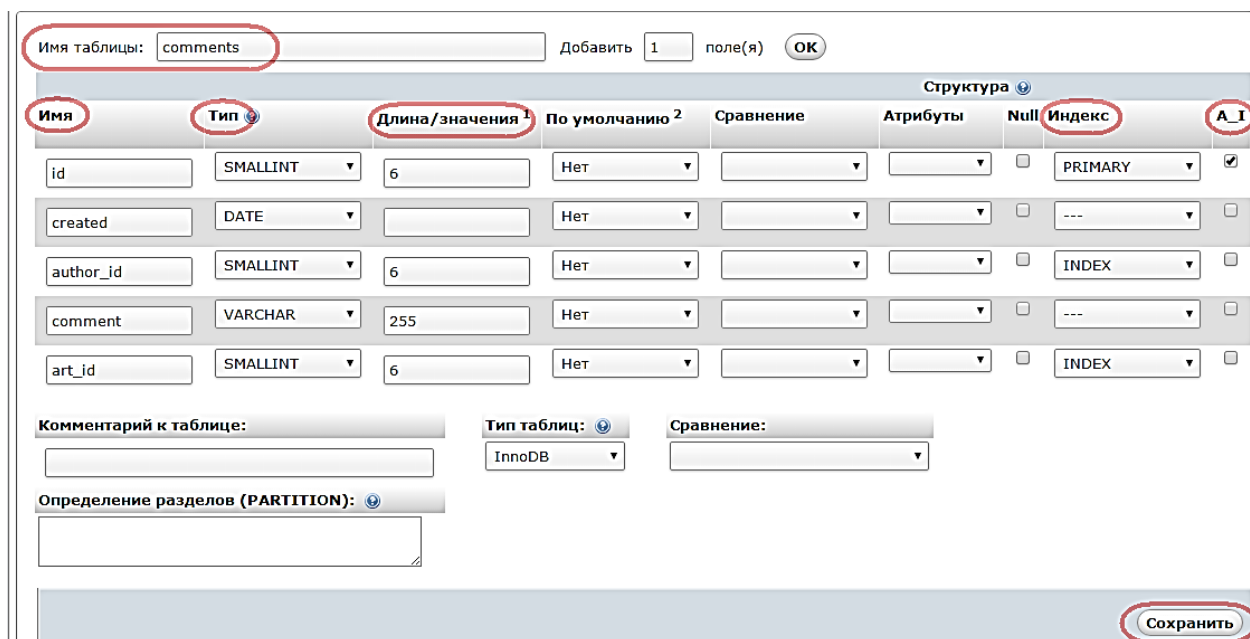


Рисунок 2.3 - Заполнение полей таблицы comments

5. Полученный результат должен выглядеть следующим образом (рисунок 2.4):

#	Имя	Тип	Сравнение	Атрибуты	Null	По умолчанию	Дополнительно	Действие
<input type="checkbox"/> 1	id	smallint(6)			Нет	Нет	AUTO_INCREMENT	Изменить Удалить Ещё
<input type="checkbox"/> 2	created	date			Нет	Нет		Изменить Удалить Ещё
<input type="checkbox"/> 3	author_id	smallint(6)			Нет	Нет		Изменить Удалить Ещё
<input type="checkbox"/> 4	comment	varchar(255)	utf8_general_ci		Нет	Нет		Изменить Удалить Ещё
<input type="checkbox"/> 5	art_id	smallint(6)			Нет	Нет		Изменить Удалить Ещё

Рисунок 2.4 - Результат создания таблицы

2.5 Создание таблицы authors

Самостоятельно средствами PhpMyAdmin создайте таблицу authors (см. приложение 3). В ходе создания таблицы поля для заполнения должны выглядеть следующим образом (рисунок 2.5):

Имя	Тип	Длина/значения	По умолчанию	Сравнение	Атрибуты	Null	Индекс	A_I
id	SMALLINT	6	Нет			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>
login	VARCHAR	10	Нет			<input type="checkbox"/>	—	<input type="checkbox"/>
password	VARCHAR	10	Нет			<input type="checkbox"/>	—	<input type="checkbox"/>
rights	ENUM	'u', 'a'	Как определено: u			<input type="checkbox"/>	—	<input type="checkbox"/>

Тип данных ENUM позволяет выбрать одно значение из списка указанных


Значение по умолчанию

Рисунок 2.5 – создание таблицы authors

2.6 Создание межтабличных связей

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

Внимание! При работе со сборкой **Open Server** инструмент **Designer** для утилиты **PHPMyAdmin** необходимо устанавливать и настраивать отдельно. См. дополнительную информацию к курсу и информацию на ресурсе <http://open-server.ru/>. Без данного инструмента связи между таблицами создаются через разделы таблиц **Связи**.

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» (рисунок 2.6) сначала между таблицами **notes** и **comments**. При этом используйте стратегию каскадирования для поддержания ссылочной целостности (каскадное удаление, каскадное обновление). Для этого выберите команду «Создать связь»  - поле первичного ключа в главной таблице (**Notes**) – поле внешнего ключа в подчиненной таблице (**Comments**). В открывшемся окне выберите «On delete CASCADE», «On update CASCADE».

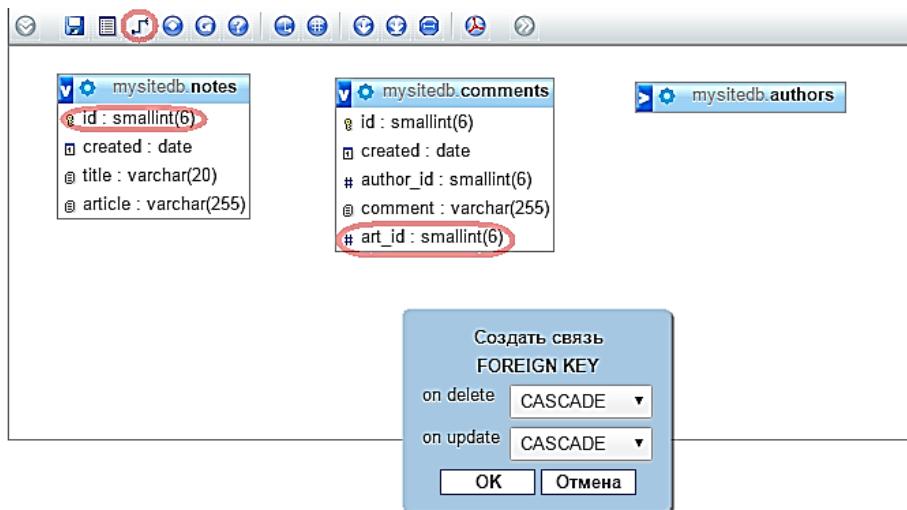


Рисунок 2.6 - Поле окна инструментов Designer

3. В результате успешного создания связи между таблицами **notes** и **comments** схема должна выглядеть как представлено на рисунке 2.7:

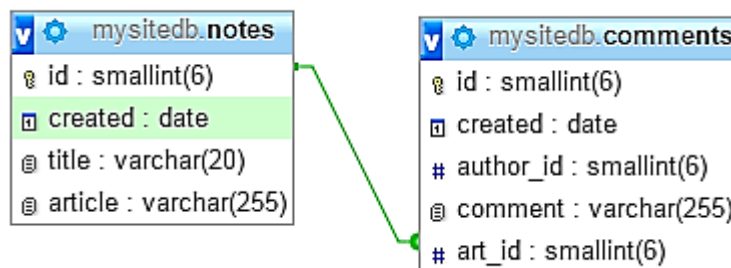


Рисунок 2.7 – Связь между таблицами Notes и Comments

4. Аналогичным образом создайте связь между таблицами **authors** и **comments**. Отличием от предыдущей связи будет являться выбранная стратегия поддержания ссылочной целостности (рисунок 2.8):

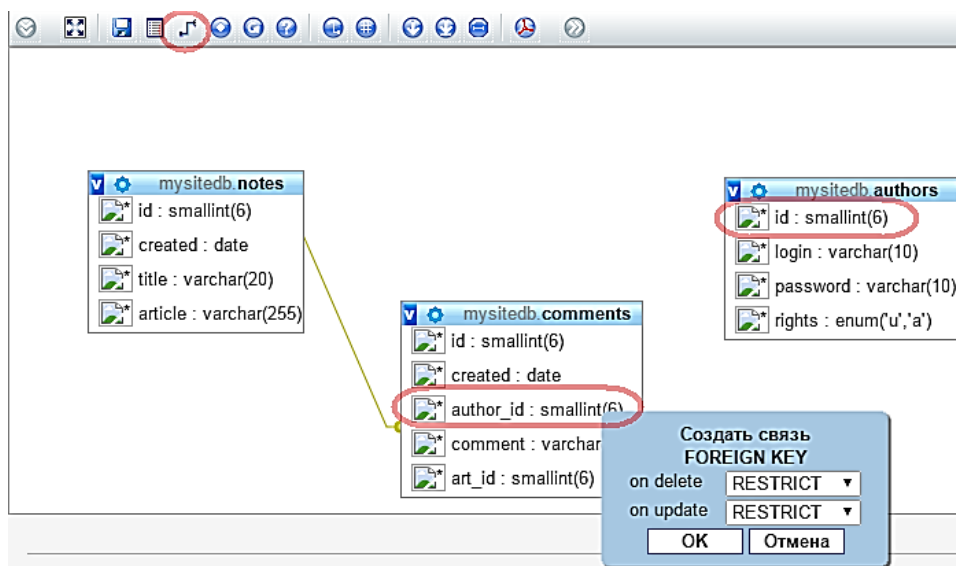


Рисунок 2.8 – Создание связи между таблицами authors и comments

5. В результате будет создана связь вида (см. рисунок 2.9):

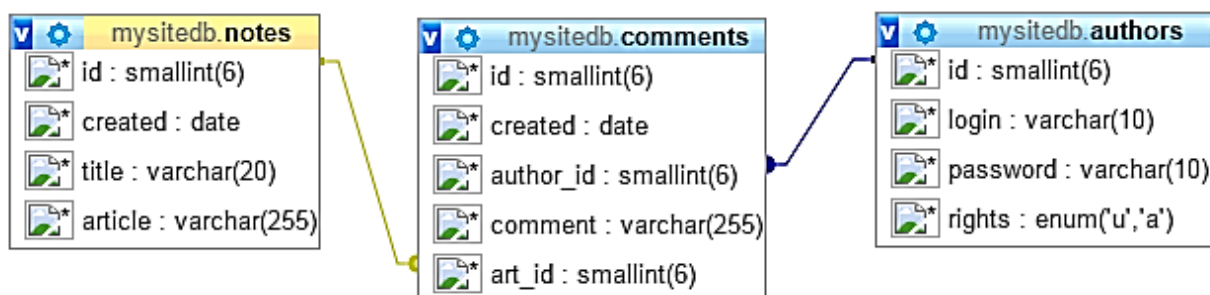


Рисунок 2.9 – связь между таблицами *authors* и *comments*

2.7 Заполнение таблиц

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 **Вставить (Insert)** на вкладках. После заполнения соответствующих полей таблицы нажмите кнопку **Ок**.

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

2.8 Файл подключения базы данных

В ходе выполнения данного упражнения необходимо создать файл, содержащий код подключения базы данных. В дальнейшем он будет использоваться на каждой странице, работающей с базой.

1. Создайте файл **MySiteDB.php** следующего вида:

```
<?php
    $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 utf8;") or die(mysql_error());
mysqli_query(\$link, "SET CHARACTER SET utf8;") or die(mysql_error());*

//Выбор базы данных на сервере localhost

mysqli_select_db(\$link, \$db);

?>

2. Сохраните файл в папке проекта.

3 Простой вывод данных. Страницы `default.php` и `comments.php`.

3.1 Вывод данных из базы на страницу

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

1. Откройте страницу `default.php`.
2. Создайте соединение с сервером. Оно реализовано в файле `MySiteDB.php` – данный файл надо включить с помощью функции `require_once()`, в качестве параметра передав ей путь к файлу («`MySiteBD.php`»):

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

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

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

```
$query = "SELECT * FROM notes";
```

⇒ 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.

- 3.2. Реализуйте запрос на выборку с помощью функции `mysqli_query`:

```
$select_note = mysqli_query($link, $query);
```

4. 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.

5. 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>";
```

6. 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()*.

Вариант полного кода страницы

```
<?php require_once("MySiteDB.php"); ?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Главная страница сайта</title>
    </head>
    <body>
        <menu>
            Вход
            Новая заметка
            Отправить сообщение
            Добавить фото
            Статистика
```

```
Администратору  
Выход  
</menu>
```

```
<p>Рад приветствовать вас на страницах моего сайта, посвященного путешествиям. </p>
```

```
<?php  
$query = "SELECT * FROM notes";  
$select_note = mysqli_query($link, $query);  
  
while ($note = mysqli_fetch_array($select_note))  
{  
    echo $note ['id'], "<br>";  
    echo $note ['created'], "<br>";  
    echo $note ['title'], "<br>";  
    echo $note ['article'], "<br>";  
}  
?>  
  
<body>  
</html>
```

3.2 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 на ней соединение с сервером и подключение к БД.

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

1.3. Для передачи идентификатора заметки введем аргумент **note**. В качестве значения он будет получать значение поля **id** таблицы **notes**.

1.4. На странице **default.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 ['article'], "<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 **default.php**. Основное отличие заключается в том, что вначале необходимо со станицы **default.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 default.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 `$note_id` содержится *id* выбранной заметки, следовательно, для выбора комментариев к этой заметке необходимо, чтобы значение поля *art_id* `created` также было равно `$note_id`.

3. Проверьте корректность данных между страницами **default.php** и **comments.php**. При переходе по ссылке с **default.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. Самостоятельно реализуйте вывод надписи «Эту запись еще никто не комментировал» для тех заметок, к которым нет комментариев (например, используя функцию `mysqli_num_rows()` или другие решения).

4 Ввод и обработка данных (элементы HTML-форм)

В ходе выполнения данной лабораторной работы рассматриваются принципы ввода и обработки информации при работе с формами HTML.

4.1 Отправка почты

Данное упражнение позволяет реализовать отправку сообщения через форму на сервер.

1. Создайте страницу **email.php**. Добавьте название страницы и пояснительный текст, 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

или

C:\OpenServer\userdata\temp\email

для **WebServer** и **Open Server** соответственно.

3. Проверьте корректность работы, создайте гиперссылки с главной страницы сайта на страницу **email.php** и со страницы **email.php** на страницу **default.php**.
4. Самостоятельно реализуйте проверку заполнения всех полей формы для того, чтобы исключить отправку «пустого» письма.

4.2 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 обработке 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

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Новая заметка</title>
  </head>
  <body>
    <p>Добавить новую заметку: </p>
    <form method="post">
      <input type="text" name="title" size="20" maxlength="20"/>
      <textarea name="article" cols="55" rows="10"></textarea>
      <input type="hidden" name = "created"
            value = "<?php echo date("Y-m-
d");?>" />
```

```

        <input type="submit" name="submit" value="Отправить"
/>
    </form>
    <a href="default.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 и выбрать базу данных;

6.1.2. 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
$query = "INSERT INTO notes (id, title, created, article)
        VALUES (NULL, '$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 **default.php** и **newnote.php**.

Вариант реализации кода страницы newnote.php

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Новая заметка</title>
  </head>
  <body>
    <p>Добавить новую заметку: </p>
    <form method="post">
      <input type="text" name="title" size="20" maxlength="20"/>
      <textarea name="article" cols="55" rows="10"></textarea>
      <input type="hidden" name="created"
        value="<?php echo date("Y-m-
d");?>" />
      <input type="submit" name="submit" value="Отправить"
    />
    </form>
    <a href="default.php">Возврат на главную страницу сайта</a>
  </body>
</html>
<?php
//Подключение к серверу
```

```

require_once ("MySiteDB.php");

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

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

```

4.3 Обработка HTML-переключателей

1. На любой странице сайта (можно создать новую и дать ей произвольное имя, указав тип данных .php) создайте HTML-форму.
2. В созданной форме создайте группу переключателей (radio) на произвольную тему.
3. В том же файле создайте php-скрипт, обрабатывающий информацию о том, какой переключатель был выбран пользователем. Обратите внимание, что radio позволяет выбрать только один вариант ответа.

Вариант кода

```

<html>
<form action="ИмяВашегоФайла.php" method = "GET" enctype =
"multipart/form-data">
    <p> Выберите Ваш любимый город: </p><br />
    <p><input type = "radio" name = "MyRadio" value = "Rome">Рим</p>
    <p><input type = "radio" name = "MyRadio" value = "Paris">Париж</p>
    <p><input type = "radio" name = "MyRadio" value =
    "Moscow">Москва</p>
    <p><input type = "submit" name = "submit" value = "Отправить" />
</form>
</html>
<?php
    $var = $_GET['MyRadio'];
    switch($var)

```

```

        {
            case " Rome":
                echo "You choose $var";
                break;

            case " Paris":
                echo "You choose $var";
                break;

            case " Moscow":
                echo "You choose $var";
                break;
        }
?>

```

4.4 Обработка HTML-флагов

1. Ниже на странице, созданной в предыдущем упражнении, создайте вторую HTML-форму.
2. В данной HTML-форме создайте группу флажков (checkbox) на произвольную тему.
3. В том же файле создайте php-скрипт, обрабатывающий информацию о том, какие варианты ответов были выбраны пользователем. Обратите внимание, что checkbox позволяет выбрать множество (от нуля до максимального количества) ответов.

Вариант кода

```

<html>
  <form action="ИмяВашегоФайла.php" method = "GET" enctype =
  "multipart/form-data">
    <p> Выберите Ваши любимые города: </p><br />
    <p><input type = "checkbox" name = "MyCheckBox[]"
      value = "Rome">Рим</p>
    <p><input type = "checkbox" name = "MyCheckBox[]"
  value = "Paris">Париж</p>
    <p><input type = "checkbox" name = "MyCheckBox[]"
      value = "Moscow">Москва</p>
    <p><input type = "submit" name = "submit" value = "Отправить"/>
  </form>
</html>

<?php

```

```

    $arr = $_GET['MyCheckBox'];
        if(empty($arr))
            {
                echo "Вы не выбрали ни один вариант";
            }
        else
            {
                $count = count($arr);
echo "Вы выбрали:."<br />";
for($i=0; $i<$count; $i++)
            {
                echo $arr[$i]."<br />";
            }
        } ?>

```

5 Модификация данных

В данной лабораторной работе рассматривается реализация редактирования уже существующих заметок, а также их удаление.

5.1 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 **default.php** на страницу **comments.php**), см. рисунок 5.1:

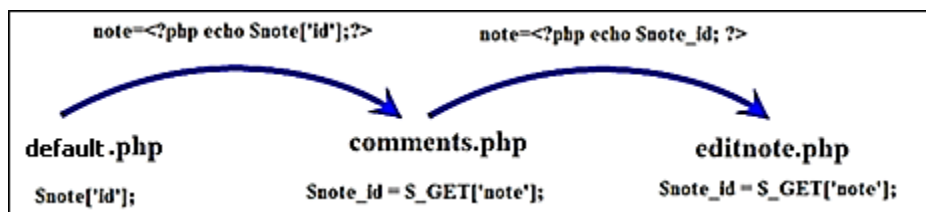


Рисунок 5.1 - Схема обмена данными методом GET

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

```
<a href="editnote.php?note=<?php echo $note_id; ?>">Изменить 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 измененные 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 и выбор базы данных
require_once ("MySiteDB.php");

//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
выполняя функции mysqli_fetch_array(); -->
```

```
<form method="post">
<p>Заголовок заметки: <input type="text" name="title"
value = "<?php echo $edit_note['title'];?>" /></p>
<p>Текст заметки: <textarea name="article">
<?php echo $edit_note['article'];?></textarea></p>
<input type="hidden" name = "note" value="<?php echo $edit_note['id'];?>" />
<input type="submit" name="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 ("MySiteDB.php");  
$query = "SELECT * FROM notes WHERE id = $note_id";  
$result = mysqli_query ($link, $query);  
$edit_note = mysqli_fetch_array ($result);  
?>
```

```
<!DOCTYPE html>
```

```

<html>
  <head>
    <meta charset="utf-8">
  </head>
  <body>
    <form method="post">
      <p>Заголовок заметки: <input type="text" name="title"
        value = "<?php echo $edit_note['title'];?>" /></p>
      <p>Текст заметки: <textarea name="article">
        <?php echo $edit_note['article'];?></textarea></p>
      <input type="submit" name="submit" value="Изменить" />
    </form>

    <a href = "default.php"> На главную </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);
?>

```

5.2 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.

Т.к. при создании связи между таблицами **notes** и **comments** было реализовано каскадное удаление, то при удалении заметки из таблицы **notes** автоматически должны удаляться все комментарии к ней из таблицы **comments**. Убедитесь в этом при проверке работы оператора DELETE.

Вариант кода

```
<?php
require_once("mysitedb.php");
$note_id = $_GET['note'];
mysqli_select_db($link, $db);

$query = "DELETE FROM notes WHERE id = $note_id";
$res = mysqli_query($link, $query);

//Работа с заголовками (см. документацию php)
//header("Location: default.php");
header( "refresh:5;url = default.php" );
    echo 'Your note was deleted. You\'ll be redirected in about 5 secs.
    If not, click <a href=" default.php">here</a>.';
?>
```

6 Работа с заметками

6.1 Работа со страницей default.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 используется 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).

6.2 Работа с комментариями к заметкам

Самостоятельно создайте страницу для добавления комментариев к заметкам.

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.

7.1 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.

7.2 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.

7.3 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);
```

7.4 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.

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

```

В представленном коде:

\$allnotes_num, *\$allcomments_num*, *\$row_lmnotes*, *\$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`.

```

<?php require_once ("MySiteDB.php");

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

```

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).

8.1 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 поместить 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>";
}

```

8.2 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>";
    }
?>

```

9 Работа с файлами

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

9.1 Вывод списка файлов

1. Создайте папку **photo** для размещения изображений. Поместите в эту папку несколько изображений.
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. рисунок 9.1.

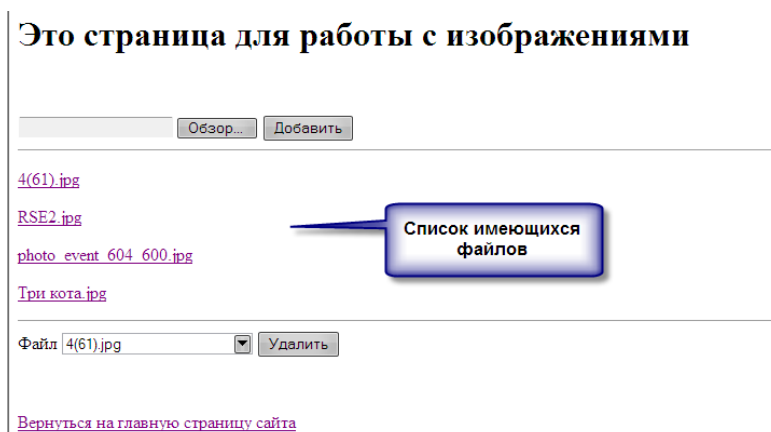


Рисунок 9.1 - Схема страницы *photo.php*

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);

Внимание! Представленные в листинге пути к файлам написаны для работы в папке *localhost\photo* (для Open Server) или *localhost\www\photo* (для Denwer). Внимательно отнеситесь к прописыванию путей к файлам. Например, если Ваша папка находится по следующему адресу *C:\OpenServer\domains\localhost\MyTravelNotes\photo*, то Вам необходимо это учитывать в пути к файлам и, например, вместо

```
$_SERVER['DOCUMENT_ROOT'] . "/photo";
```

писать

```
$_SERVER['DOCUMENT_ROOT'] . "/mytravelnotes/photo";
```

Фрагменты, содержащие пути, на которые необходимо обратить внимание, выделены в листинге.

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,
хранящемуся //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.

9.2 Загрузка 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="file" name="file_upload" />
<input type="hidden" name="MAX_FILE_SIZE" value="65536" />
<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*.

9.3 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=" multipart/form-data ">
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=" multipart/form-data ">
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.

9.4 Работа с содержимым файла

1. С помощью **PhpMyAdmin** откройте базу данных **MySiteDb** и импортируйте данные из таблицы **authors** в файл **authors.xml**. Для этого выберите таблицу **authors** – Вкладка **Экспорт** – Способ экспорта «*Быстрый – отображать минимум настроек*» - Вывод «*Сохранить на сервере в каталоге c:/openserver/userdata/temp/*» - Формат **XML** – **ОК** (см. рисунок 9.2)

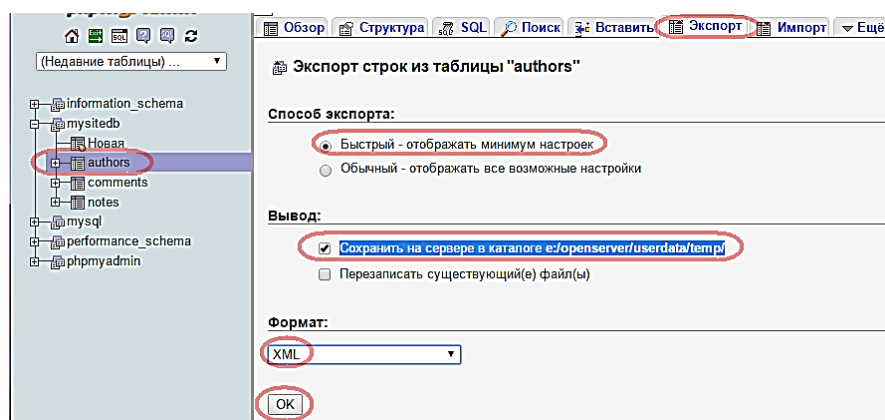


Рисунок 9.2 – Экспорт данных из таблицы

2. Скопируйте созданный в результате операции экспорта файл **authors.xml** в рабочую папку проекта. Изучите его.
3. В редакторе создайте новый файл **files.php**. В нем создайте код открытия и чтения данных из файла:

```
<?php
$f = fopen("authors.xml", "a+");
echo fread($f, filesize("authors.xml"));
?>
```

4. Убедитесь в работоспособности кода, запустив данный файл на выполнение (через адресную строку браузера).
5. Измените код на построчное чтение данных из файла:

```
<?php
$f = fopen("authors.xml", "a+");
echo fgets($f);
?>
```

6. Убедитесь в работоспособности кода. Т.к. в начале созданного xml файла много строк служебной информации, возможно вызов функции **fgets()** необходимо будет сделать несколько раз.
7. Закомментируйте весь только что созданный код. Напишите код, считывающий данные построчно из файла и помещающий его в массив. Выведите информацию из массива на страницы сайта:

```
$array = file("authors.xml");
print_r($array);
```

8. Закомментируйте код выше. Считайте данные из файла одной строкой:

```
$get_cont = file_get_contents("authors.xml");
echo $get_cont;
```

9. Измените данные в файле **authors.xml**, дополнив их произвольной информацией, например:

```
$put_cont = file_put_contents("authors.xml", "My new contents",
FILE_APPEND);
echo $put_cont;
```


10 Основы разграничения прав доступа

В данной лабораторной работе будут реализованы основы реализации разграничения прав доступа к контенту сайта с использованием механизма сессий.

10.1 Создание страницы входа на сайт

1. Создайте новый файл **login.html**, содержащий форму, передающую методом **POST** файлу **login.php** (он будет создан на следующем шаге) на обработку логин и пароль посетителя сайта.
2. Создайте файл **login.php**, принимающий данные, поступившие из html-формы файла **login.html**. Далее необходимо сравнить полученные данные с теми данными, которые имеются в базе данных, и если они совпадают (т.е. пользователь с таким логином и паролем существует), то для него необходимо запустить сессию.

Вариант реализации кода

```
<?php
require_once("MySiteDB.php");

$login = $_POST['login'];
$password = $_POST['password'];
//echo $login, $password, "<br>";

if(($login) &&($password))
{
    $query = "SELECT * FROM authors WHERE login = '$login' AND
password = '$password'";
    $send_query = mysqli_query($link, $query);
    $user_array = mysqli_fetch_array($send_query);
    $login = $user_array['login'];
    $rights = $user_array['rights'];

    $count = mysqli_num_rows($send_query);
    if ($count >0)
```

```

    {
        session_start();
        $_SESSION['login'] = $login;
        $_SESSION['rights'] = $rights;

        header( "refresh:3;url = default.php" );
        echo "Вход на сайт автоматически осуществится
        через 3 секунды или нажмите <a href='default.php'>сюда</a>.";
    }
    else
    {
        echo "Извините, Вы не зарегистрированы.";
    }
}
?>

```

3. Организуйте в проекте соответствующие гиперссылки для перехода между страницами.

10.2 Организация доступа к страницами сайта

В данном упражнении необходимо разграничить права доступа ко всем страницам сайта, согласно следующей информации:

Страница	Права доступа
default.php	Все пользователи
newnote.php	a
deletenote.php	a
editnote.php	a
email.php	u, a
inform.php	a
newcomment.php	u, a
photo.php	u, a

1. Возможный вариант реализации представлен на примере страницы **newnote.php**:

```

<?php
session_start();
if ($_SESSION['rights']=='a')

```

```

{
    echo "hello, ".$_SESSION['login'];
    ?>

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Новая заметка</title>
    </head>
    <body>
        <!--Здесь код формы для добавления новой заметки -->
    </body>
</html>

<?php
require_once ("MySiteDB.php");

$title = $_POST['title'];
$created = $_POST['created'];
$article = $_POST['article'];

if (($title)&&($created)&&($article))
    {
        $query = "INSERT INTO notes (id, title, created, article) VALUES (NULL,
'$title', '$created', '$article)";
        $result = mysqli_query ($link, $query);
    }
}
else
    {
        echo "Извините, у Вас нет доступа";
        echo "<a href = \"default.php\">На главную</a>";
    }
?>

```

2. Аналогичным образом организуйте разграничение прав доступа к другим страницам сайта. При желании можно необходимый код вынести в отдельный файл и подключать его к страницам, чтобы не дублировать информацию.

10.3 Создание страницы выхода с сайта

Самостоятельно создайте страницу **loguot.php**, позволяющую завершать сессию и таким образом осуществлять выход пользователя с сайта. Используйте функции *session_destroy()* и *session_unset()*.

10.4 Страница администрирования

В данном упражнении необходимо создать страницу с информацией для администратора сайта (**admin.php**), на которую будет выводиться информация о зарегистрированных пользователях, а также с помощью которой можно будет добавлять новых пользователей (для этого также необходимо будет создать страницу **newUser.php**).

1. Создайте новый файл **admin.php**. В нем (можно в табличной форме) выведите информацию на страницу сайта о зарегистрированных пользователях (т.е. записи из таблицы **authors**).
2. Проверьте корректность вывода информации.
3. В том же файле создайте форму для добавления нового пользователя. Данные должны передаваться методом **POST** на обработку в файл **newUser.php**, который будет создан на следующем шаге.
4. Создайте новый файл **newUser.php**. В нем реализуйте возможность добавления нового пользователя в таблицу зарегистрированных пользователей. Вариант реализации файла **newUser.php**:

```
<?php
require_once('mysitedb.php');

$login = $_POST['login'];
$password = $_POST['password'];
$rights = $_POST['rights'];

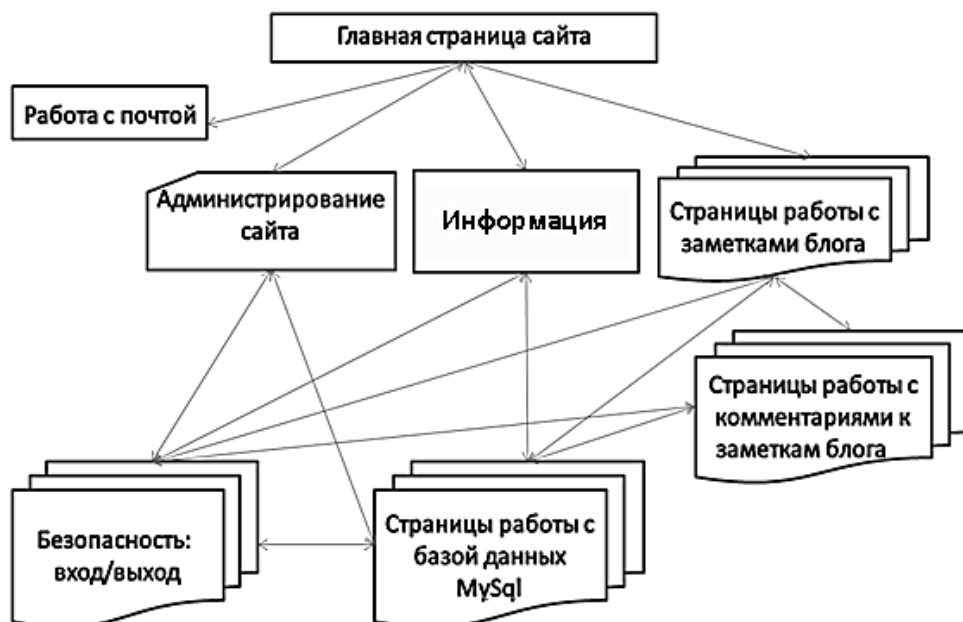
if(($login)&&($password)&&($rights))
{
    $query = "INSERT INTO authors VALUES (NULL, '$login',
'$password', '$rights)";
    mysqli_query($link, $query);

    header( "refresh:1;url=admin.php" );
}
?>
```

Литература

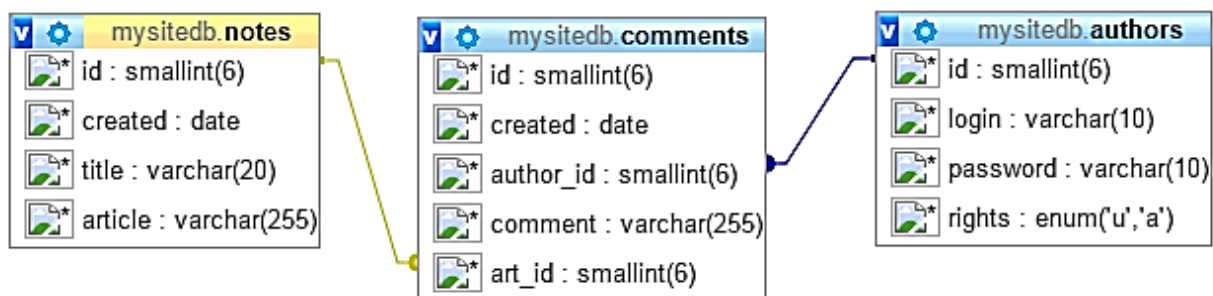
1. Php.ru. Справочник языка. Режим доступа <http://php.ru/manual/> свободный. Яз. русский (дата обращения 15.10.2016).
2. W3schools.com. PHP 5 Tutorial. Режим доступа <http://www.w3schools.com/php/default.asp> свободный. Яз. английский (дата обращения 12.10.2016).
3. Welling L., Thomson L. PHP and MySQL Web Development (5th Edition). Addison-Wesley Professional, 2016 – 1008 p.
4. Дронов В. PHP, MySQL, HTML5 и CSS 3. Разработка современных динамических Web-сайтов. СПб.: БХВ – Петербург, 2016 - 688 с.
5. Зандстра М. PHP. Объекты, шаблоны и методики программирования. – М.: Вильямс, 2015. – 576 с.
6. Колисниченко Д. PHP и MySQL. Разработка веб-приложений. СПб.: БХВ – Петербург, 2015 - 592 с.
7. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. СПб.: ПИТЕР, 2015. – 688 с.
8. Пауэрс Д. PHP. Создание динамических страниц. М.: Рид Групп, 2012 – 640 с.
9. Руководство по PHP. Режим доступа <http://php.net/manual/ru/> свободный. Яз. русский (дата обращения 12.10.2016).
10. Скляр Д., Трахтенберг А. PHP. Рецепты программирования. СПб.: БХВ – Петербург, 2015 - 784 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 «authors» содержит информацию о зарегистрированных 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(6)	Ключевое поле (AUTO_INCREMENT), INDEX – PRIMARY.
created	Дата создания заметки	DATE	
title	Заголовок заметки	VARCHAR (20)	Строка фиксированной длины 20 символов
article	Содержимое заметки	VARCHAR (255)	Строка фиксированной длины 255 символов

Таблица 2. Структура таблицы comments

Имя поля	Описание поля	Тип данных	Другое
id	Идентификатор записи	SMALLINT(6)	Ключевое поле (AUTO_INCREMENT), INDEX – PRIMARY.
created	Дата публикации комментария	DATE	
author_id	внешний ключ к таблице Authors	SMALLINT (6)	INDEX
comment	Содержимое комментария	VARCHAR (255)	Строка фиксированной длины 255 символов
art_id	Внешний ключ к таблице Notes	SMALLINT(6)	INDEX

Таблица 3. Структура таблицы authors

Имя поля	Описание поля	Тип данных	Другое
id	Идентификатор записи	SMALLINT(6)	Ключевое поле (AUTO_INCREMENT), INDEX – PRIMARY.
login		VARCHAR (10)	Логин пользователя
password		VARCHAR (10)	Пароль пользователя
rights	Права доступа: u – user, a - administrator	ENUM('u', 'a')	Права доступа пользователя к страницам сайта. Значение по умолчанию – u (пользователь)

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ПРОГРАММНЫХ СИСТЕМ

<http://ps.ifmo.ru>

Кафедра Программных систем входит в состав факультета Инфокоммуникационных технологий. На кафедре обеспечена возможность обучения студентов по современным образовательным стандартам в области программного обеспечения ИКТ. Для этого на кафедре работает высококвалифицированный преподавательский состав, имеется современная техническая база и специализированные лаборатории, оснащенные необходимым оборудованием и программным обеспечением; качественная методическая поддержка образовательных программ. Наши студенты принимают активное участие в российских и зарубежных исследованиях и конференциях, имеют возможность публикации результатов своих исследований в ведущих российских и зарубежных реферируемых изданиях. Существенным преимуществом является возможность выпускников продолжить научную деятельность в аспирантуре Университета ИТМО и в других передовых российских и зарубежных научных Центрах.

Кафедра обеспечивает подготовку бакалавров и магистров по образовательным программам:

- Интеллектуальные инфокоммуникационные системы - бакалавры;
- Программное обеспечение в инфокоммуникациях – магистры.

На кафедре реализуется международная образовательная программа DD Master Program, в рамках которой выпускники имеют возможность получить два диплома: Диплом Университета ИТМО с присвоением магистерской степени по направлению «Программное обеспечение в инфокоммуникациях» и Международный диплом - Master of Science in Technology Lappeenranta University of Technology in the field of Computer Science majoring in Software Engineering.

Выпускники кафедры обладают компетенциями:

- проектирования и создания рациональных структур ИКС;
- разработки алгоритмов решения задач и их программной реализации на основе современных платформ;
- моделирования процессов функционирования сложных систем; •обеспечения безопасности работы ИКС;
- реализации сетевых услуг и сервисов в ИКС;
- проектирования и разработки баз данных;
- разработки клиент-серверных приложений ИКС;
- проектирования, создания и поддержки Web-приложений;
- управления проектами перспективных направлений развития ИКС.

Трудоустройство выпускников кафедры возможно на предприятиях: ООО «Digital Design»; ООО «Аркадия»; ОАО «Ростелеком»; ООО «ЭПАМ Системз»; ООО «Т-Системс СиАйЭс» и многие другие.

Мы готовим квалифицированных инженеров в области инфокоммуникационных технологий с новыми знаниями, образом мышления и способностями быстрой адаптации к современным условиям труда.

Одиночкина Светлана Валерьевна

**Основы разработки приложений на PHP
Практикум**

В авторской редакции

Редакционно-издательский отдел Университета ИТМО

Зав. РИО

Н.Ф. Гусарова

Подписано к печати

Заказ №

Тираж

Отпечатано на ризографе

**Редакционно-издательский отдел
Университета ИТМО**
197101, Санкт-Петербург, Кронверкский пр., 49