МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

С.В. Одиночкина

Web-программирование: PHP

практикум

Санкт-Петербург

УДК 004.655, 004.657, 004.62

С.В.Одиночкина

Web-программирование PHP - СПб: НИУ ИТМО, 2012. – 79 с.

В пособии излагаются методические рекомендации к выполнению лабораторных работ по дисциплине «Web-программирование PHP-технологии».

Предназначено для студентов, обучающихся по всем профилям подготовки бакалавров направления: 210700 Инфокоммуникационные технологии и системы связи.

Рекомендовано к печати Ученым советом факультета Инфокоммуникационных технологий, протокол №4 от 13 декабря 2011г.

В 2009 году Университет стал победителем многоэтапного конкурса, в результате которого определены 12 ведущих университетов России, которым присвоена категория «Национальный исследовательский университет». Министерством образования и науки Российской Федерации была утверждена программа его развития на 2009–2018 годы. В 2011 году Университет получил наименование «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

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 обработкку 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 черех

фрму 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 *tblNamt* – 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 исколючите 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 передачи идентификатора заметки со страницы **blog.php** на страницу **comments.php**), *см. рис. 5.1*:

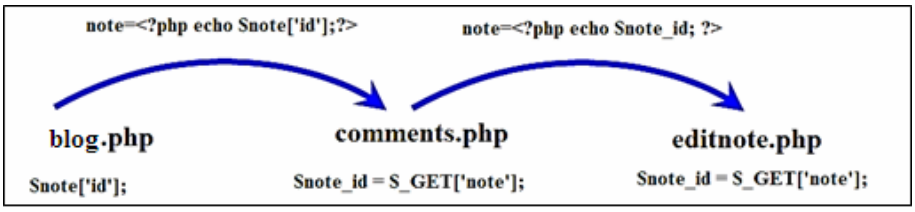

Рис.5.1. Схема обмена данными методом GET

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

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

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
выполения 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);
    $edit_note = mysqli_fetch_array ($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 $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>

<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 забудтье реализвать удаление комментариев к удаляемым записям.