Лабораторная работа №№ 20-24

Создание проекта «Форум»

Цель: сформировать навыки и умения создания проектов регистрации пользователей.

Ход работы:

1. Выполнить задания в соответствии с указаниями.
2. Оформить отчет по лабораторной работе в соответствии с требованиями (содержание отчета см. ниже).
3. Подготовить ответы на контрольные вопросы (устно)

Указания по выполнению

Шаг 1: Создание таблиц базы данных

Всегда полезно начинать с создания хорошей модели данных при создании приложения. Давайте опишем наше приложение одним предложением: мы собираемся создать форум, на котором есть **пользователи,** которые создают **темы** в различных **категориях** . Другие пользователи могут **оставлять** ответы. .

- категории
- темы
- Сообщений

Эти три объекта связаны друг с другом, поэтому мы будем обрабатывать их в нашей таблице. Посмотрите на схему ниже.

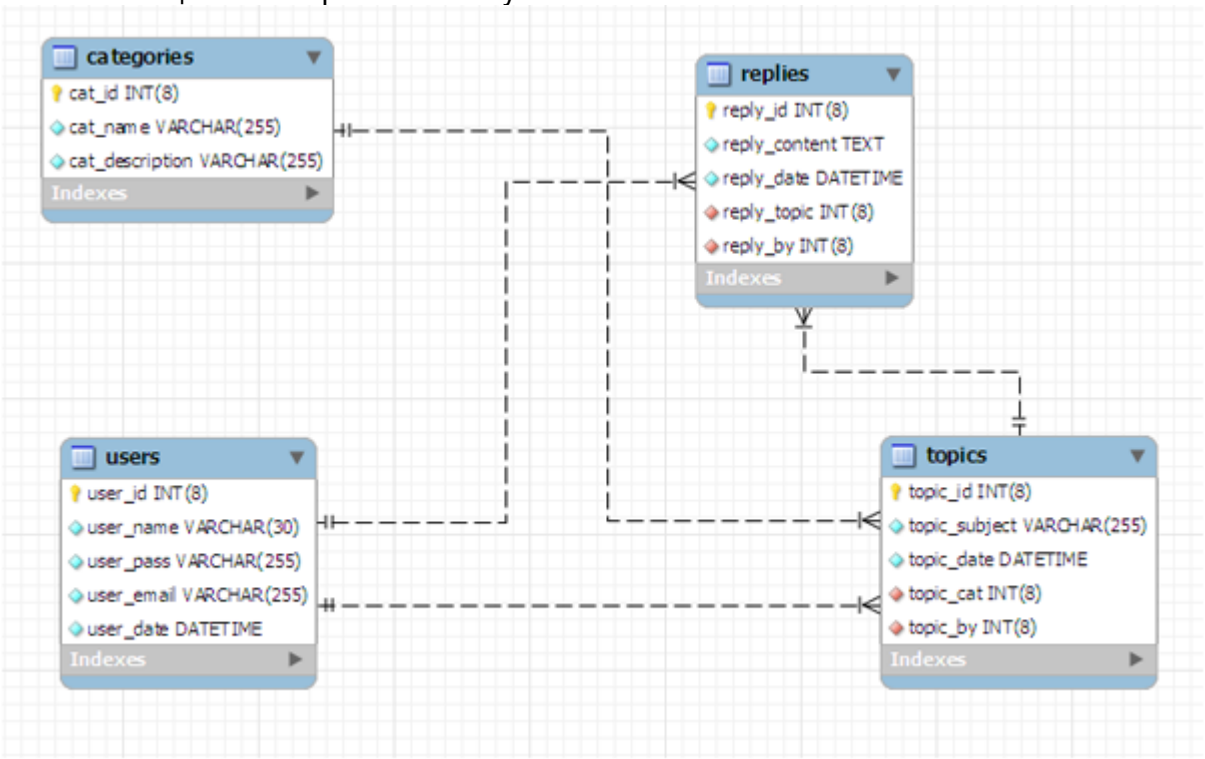


Таблица пользователей

```
CREATE TABLE users (
user_id INT(8) NOT NULL AUTO_INCREMENT,
user_name VARCHAR(30) NOT NULL,
user_pass VARCHAR(255) NOT NULL,
user_email VARCHAR(255) NOT NULL,
user_date DATETIME NOT NULL,
user_level INT(8) NOT NULL,
UNIQUE INDEX user_name_unique (user_name),
PRIMARY KEY (user_id)
) TYPE=INNODB;
```

Таблица категорий

```
CREATE TABLE categories (
cat_id INT(8) NOT NULL AUTO_INCREMENT,
```

```
cat_name VARCHAR(255) NOT NULL,
cat_description VARCHAR(255) NOT NULL,
UNIQUE INDEX cat_name_unique (cat_name),
PRIMARY KEY (cat_id)
) TYPE=INNODB;
```

Вмя категории должно быть уникальным.

Таблица тем

```
CREATE TABLE topics (
topic_id INT(8) NOT NULL AUTO_INCREMENT,
topic_subject VARCHAR(255) NOT NULL,
topic_date DATETIME NOT NULL,
topic_cat INT(8) NOT NULL,
topic_by INT(8) NOT NULL,
PRIMARY KEY (topic_id)
) TYPE=INNODB;
```

Эта таблица почти такая же, как и другие таблицы, за исключением поля topic_by. Это поле относится к пользователю, который создал тему. Topic_cat относится к категории, к которой относится тема. Мы не можем форсировать эти отношения, просто объявив поле. Мы должны сообщить базе данных, что это поле должно содержать существующий user_id из таблицы users или действительный cat_id из таблицы категорий. Мы добавим некоторые отношения после того, как я обсудил таблицу сообщений.

Таблица сообщений

```
CREATE TABLE posts (
post_id INT(8) NOT NULL AUTO_INCREMENT,
post_content TEXT NOT NULL,
post_date DATETIME NOT NULL,
post_topic INT(8) NOT NULL,
post_by INT(8) NOT NULL,
PRIMARY KEY (post_id)
) TYPE=INNODB;
```

Это так же, как остальные таблицы; здесь также есть поле, которое ссылается на user_id: поле post_by. Поле post_topic относится к теме, к которой принадлежит сообщение.

Внешний ключ является ссылочным ограничением между двумя таблицами. Внешний ключ идентифицирует столбец или набор столбцов в одной (ссылающейся) таблице, которая ссылается на столбец или набор столбцов в другой (ссылающейся) таблице. Некоторые условия:

- Столбец в таблице ссылок, на которую ссылается внешний ключ, должен быть первичным ключом
- Указанные значения должны существовать в ссылочной таблице.

При добавлении внешних ключей информация связывается воедино, что очень важно для нормализации базы данных.

Пришло время добавить их в таблицы, которые мы уже создали, с помощью оператора ALTER, который можно использовать для изменения уже существующей таблицы.

Сначала мы свяжем темы с категориями:

```
ALTER TABLE topics ADD FOREIGN KEY(topic_cat) REFERENCES categories(cat_id) ON
DELETE CASCADE ON UPDATE CASCADE;
```

Последняя часть запроса уже говорит, что происходит. Когда категория удаляется из базы данных, все темы также будут удалены. Если cat_id категории изменится, каждая тема также будет обновлена. Вот для чего нужна часть ОБНОВЛЕНИЯ КАСКАДА. Конечно, вы можете отменить это, чтобы защитить свои данные, так что вы не можете удалить категорию, если у нее все еще есть связанные темы. Если вы хотите сделать это, вы можете заменить часть «ON DELETE CASCADE» на «ON DELETE RESTRICT». Также есть SET NULL и NO ACTION, которые говорят сами за себя.

Теперь каждая тема связана с категорией. Давайте свяжем темы с пользователем, который его создаст.

ALTER TABLE topics ADD FOREIGN KEY(topic_by) REFERENCES users(user_id) ON DELETE RESTRICT ON UPDATE CASCADE;
Этот внешний ключ такой же, как и предыдущий, но есть одно отличие: пользователь не может быть удален, пока есть темы с идентификатором пользователя. Мы не используем CASCADE здесь, потому что в наших темах может быть ценная информация. Мы не хотим, чтобы эта информация была удалена, если кто-то решит удалить свою учетную запись. Чтобы по-прежнему предоставлять пользователям возможность удалять свои учетные записи, вы можете создать функцию, которая анонимизирует все их темы, а затем удалить их. К сожалению, это выходит за рамки этого урока.
Связать сообщения с темами:
ALTER TABLE posts ADD FOREIGN KEY(post_topic) REFERENCES topics(topic_id) ON DELETE CASCADE ON UPDATE CASCADE;
И, наконец, свяжите каждое сообщение с пользователем, который сделал это:
ALTER TABLE posts ADD FOREIGN KEY(post_by) REFERENCES users(user_id) ON DELETE RESTRICT ON UPDATE CASCADE;
Это часть базы данных! Это было довольно много работы, но результат, отличная модель данных, определенно стоит того.
Шаг 2: Введение в систему верхнего / нижнего колонтитула
Каждая страница форума нуждается в нескольких основных вещах, таких как тип документа и некоторая разметка. Вот почему мы добавим файл header.php вверху каждой страницы и файл footer.php внизу. Header.php содержит тип документа, ссылку на таблицу стилей и некоторую важную информацию о форуме, такую как тег заголовка и метатеги.
header.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PHP-MySQL forum</title>
</head>
<body>
   <h1>My forum</h1>
    <div id="wrapper">
    <div id="menu">
        <a class="item" href="/forum/index.php">Home</a> –
        <a class="item" href="/forum/create_topic.php">Create a topic</a> –
        <a class="item" href="/forum/create_cat.php">Create a category</a>

        <div id="userbar">
        <div id="userbar">Hello Example.
    </div>
        <div id="content">
```

Div обертки будет использоваться, чтобы упростить стиль всей страницы. Меню div, очевидно, содержит меню со ссылками на страницы, которые нам еще предстоит создать, но помогает понять, куда мы движемся. Div пользовательской панели будет использоваться для небольшой верхней панели, которая содержит некоторую информацию, такую как имя пользователя и ссылку на страницу выхода из системы. Очевидно, что страница содержания содержит фактическое содержимое страницы.
Внимательный читатель, возможно, уже заметил, что мы упускаем некоторые вещи. Нет `</body>` или `</html>` . Они находятся на странице footer.php, как вы можете видеть ниже.

```
</div><!-- content -->
</div><!-- wrapper -->
<div id="footer">Created for Nettuts+</div>

</body>
</html>
```

Когда мы добавляем верхний и нижний колонтитулы на каждой странице, остальная часть страницы встраивается между верхним и нижним колонтитулом. Этот метод имеет некоторые преимущества. В первую очередь все будет оформлено правильно. Краткий пример:

```
<?php
$error = false;
if($error = false)
{
   //the beautifully styled content, everything looks good
   echo '<div id=»content»>some text</div>';
}
else
{
   //bad looking, unstyled error ☹
}
?>
```

Как видите, страница без ошибок приведет к хорошей странице с контентом. Но если есть ошибка, все выглядит действительно безобразно; поэтому лучше убедиться, что не только настоящий контент правильно стилизован, но и ошибки, которые мы можем получить.

Еще одним преимуществом является возможность внесения быстрых изменений. Вы можете убедиться сами, отредактировав текст в footer.php, когда закончите этот урок; Вы заметите, что нижний колонтитул меняется на каждой странице сразу. Наконец, мы добавляем таблицу стилей, которая предоставляет нам некоторую базовую разметку — ничего особенного.

```
body {
   background-color: #4E4E4E;
   text-align: center;
}

#wrapper {
   width: 900px;
   margin: 0 auto;
}

#content {
   background-color: #fff;
   border: 1px solid #000;
   float: left;
   font-family: Arial;
   padding: 20px 30px;
   text-align: left;
   width: 100%;
}

#menu {
```

```
    float: left;
    border: 1px solid #000;
    border-bottom: none;
    clear: both;
    width:100%;
    height:20px;
    padding: 0 30px;
    background-color: #FFF;
    text-align: left;
    font-size: 85%;
}

#menu a:hover {
    background-color: #009FC1;
}

#userbar {
    background-color: #fff;
    float: right;
    width: 250px;
}

#footer {
    clear: both;
}

/* begin table styles */
table {
    border-collapse: collapse;
    width: 100%;
}

table a {
    color: #000;
}

table a:hover {
    color:#373737;
    text-decoration: none;
}

th {
    background-color: #B40E1F;
    color: #F0F0F0;
}

td {
    padding: 5px;
}

/* Begin font styles */
h1, #footer {
```

```
    font-family: Arial;
    color: #F1F3F1;
}

h3 {margin: 0;

/* Menu styles */
.item {
    background-color: #00728B;
    border: 1px solid #032472;
    color: #FFF;
    font-family: Arial;
    padding: 3px;
    text-decoration: none;
}

.leftpart {
    width: 70%;
}

.rightpart {
    width: 30%;
}

.small {
    font-size: 75%;
    color: #373737;
}
#footer {
    font-size: 65%;
    padding: 3px 0 0 0;
}

.topic-post {
    height: 100px;
    overflow: auto;
}

.post-content {
    padding: 30px;
}

textarea {
    width: 500px;
    height: 200px;
}
```

Шаг 3: Готовимся к действию

Прежде чем мы сможем прочитать что-либо из нашей базы данных, нам нужно соединение. Вот для чего предназначен connect.php. Мы включим его в каждый файл, который собираемся создать.

```php
<?php
//connect.php
$server = 'localhost';
$username = 'root';
$password = '';
$database = 'databasename';

$link = mysqli_connect($server, $username, $password, $database);
if ($link == false){
    print("Ошибка: Невозможно подключиться к MySQL " . mysqli_connect_error());
}
?>
```

Просто замените значения переменных по умолчанию в верхней части страницы на собственные.

Шаг 4: Отображение обзора форума

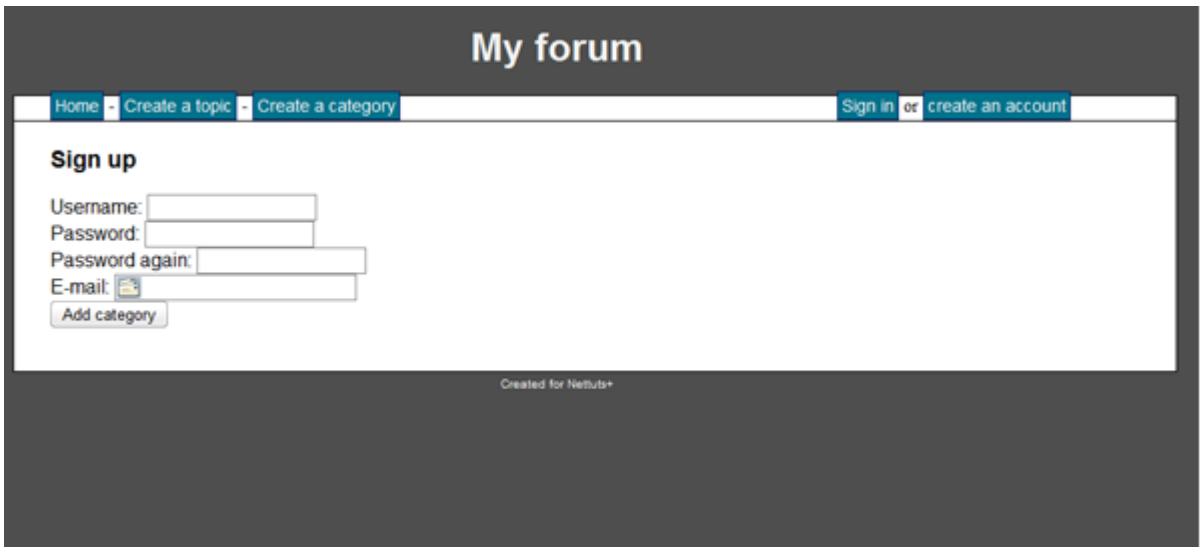
Так как мы только начали с некоторых базовых методов, мы сейчас сделаем упрощенную версию обзора форума.

```php
<?php
//create_cat.php
include 'connect.php';
include 'header.php';

echo '<tr>';
    echo '<td class="leftpart">';
        echo '<h3><a href="category.php?id=">
        Category name</a></h3> Category description goes here';
    echo '</td>';
    echo '<td class="rightpart">';
            echo '<a href="topic.php?id=">Topic subject</a> at 10-10';
    echo '</td>';
echo '</tr>';
include 'footer.php';
?>
```

будем обновлять эту страницу на протяжении всего урока, чтобы шаг за шагом она больше походила на конечный результат!

Шаг 5: Регистрация пользователя

Давайте начнем с создания простой HTML-формы, чтобы новый пользователь мог зарегистрироваться.



Страница PHP необходима для обработки формы. Мы собираемся использовать переменную $ _SERVER. Переменная $ _SERVER — это массив, значения которого автоматически устанавливаются при каждом запросе. Одним из значений массива $ _SERVER является REQUEST_METHOD. Когда страница запрашивается с помощью GET, эта переменная будет содержать значение «GET». Когда страница запрашивается через POST, она будет содержать значение «POST». Мы можем использовать это значение, чтобы проверить, была ли опубликована форма. Смотрите страницу signup.php ниже.

```php
<?php
//signup.php
include 'connect.php';
include 'header.php';

echo '<h3>Sign up</h3>';

if($_SERVER['REQUEST_METHOD'] != 'POST')
{
    /*the form hasn't been posted yet, display it
      note that the action="" will cause the form to post to the same page it is on */
    echo '<form method="post" action="">
        Username: <input type="text" name="user_name" />
        Password: <input type="password" name="user_pass">
        Password again: <input type="password" name="user_pass_check">
        E-mail: <input type="email" name="user_email">
        <input type="submit" value="Add category" />
     </form>';
}
else
{
    /* so, the form has been posted, we'll process the data in three steps:
        1. Check the data
        2. Let the user refill the wrong fields (if necessary)
        3. Save the data
    */
    $errors = array();

```

```php
if(isset($_POST['user_name']))
{
    //the user name exists
    if(!ctype_alnum($_POST['user_name']))
    {
        $errors[] = 'The username can only contain letters and digits.';
    }
    if(strlen($_POST['user_name']) > 30)
    {
        $errors[] = 'The username cannot be longer than 30 characters.'';
    }
}
else
{
    $errors[] = 'The username field must not be empty.';
}


if(isset($_POST['user_pass']))
{
    if($_POST['user_pass'] != $_POST['user_pass_check'])
    {
        $errors[] = 'The two passwords did not match.';
    }
}
else
{
    $errors[] = 'The password field cannot be empty.';
}

```

```php
    if(!empty($errors)) /*check for an empty array, if there are errors,
     they're in this array (note the ! operator)*/
    {
        echo 'Uh-oh.. a couple of fields are not filled in correctly..';
        echo '<ul>';
        foreach($errors as $key => $value) /* walk through the array
        so all the errors get displayed */
        {
            echo '<li>' .
        }
        echo '</ul>'';
    }
    else
    {
        //the form has been posted without, so save it
        //notice the use of mysql_real_escape_string, keep everything safe!
        //also notice the sha1 function which hashes the password
        $sql = "INSERT INTO
                    users(user_name, user_pass, user_email ,user_date, user_level)
                VALUES('" . mysql_real_escape_string($_POST['user_name']) . "',
                '" . sha1($_POST['user_pass']) . "',
                '" . mysql_real_escape_string($_POST['user_email']) . "',
                    NOW(),
                    0)";

        $result = mysqli_query($link, $sql);
        if(!$result)
        {
            //something went wrong, display the error
            echo 'Something went wrong while registering.'
            //echo mysql_error();
        }
        else
        {
            echo 'Successfully registered.'
        }
    }
}

include 'footer.php';
?>
```

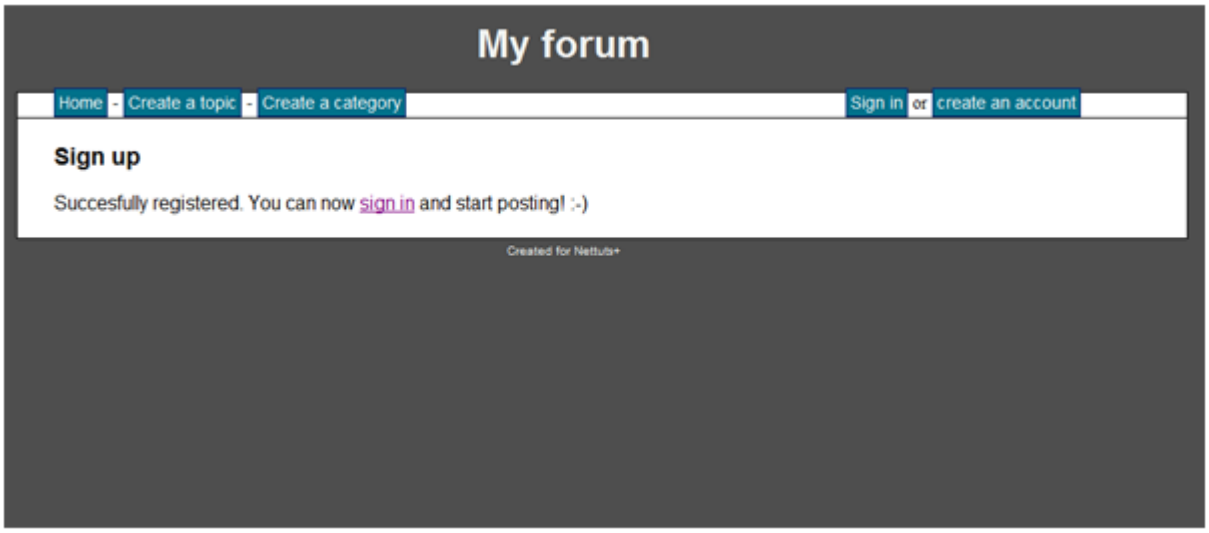
Многочисленные объяснения есть в комментариях, поэтому обязательно ознакомьтесь с ними. Обработка данных происходит в три этапа:

- Проверка данных
- Если данные неверны, покажите форму еще раз
- Если данные верны, сохраните запись в базе данных

mysqli_real_escape_string. Функция экранирует специальные символы в неэкранированной строке, поэтому ее можно безопасно разместить в запросе. Эта функция ДОЛЖНА использоваться всегда, за очень немногими исключениями. Слишком много скриптов, которые не используют его и могут быть взломаны очень легко. Не рискуйте, используйте mysqli_real_escape_string ().

Также вы можете видеть, что функция sha1() используется для шифрования пароля пользователя. Это тоже очень важная вещь для запоминания. Никогда не вставляйте простой пароль как есть. Вы ДОЛЖНЫ всегда шифровать это. Представьте себе хакера,

которому каким-то образом удается получить доступ к вашей базе данных. Если он видит все текстовые пароли, он может войти в любую учетную запись (администратора), которую он хочет. Если столбцы пароля содержат строки sha1, он должен сначала взломать их, что практически невозможно.
Примечание: также возможно использовать md5(), я всегда использую sha1(), потому что тесты показали, что он немного быстрее, хотя и немного. Вы можете заменить sha1 на md5, если хотите.
Если процесс регистрации прошел успешно, вы должны увидеть что-то вроде этого:



Попробуйте обновить экран phpMyAdmin, новая запись должна быть видна в таблице пользователей.

Шаг 6: Добавление аутентификации и пользовательских уровней

Важным аспектом форума является разница между обычными пользователями и администраторами / модераторами. Поскольку это небольшой форум, и добавление таких функций, как добавление новых модераторов и прочего, займет слишком много времени, мы сосредоточимся на процессе входа в систему и создадим некоторые функции администратора, такие как создание новых категорий и закрытие темы.
Теперь, когда вы выполнили предыдущий шаг, мы сделаем вашу вновь созданную учетную запись учетной записью администратора. В phpMyAdmin, нажмите на таблицу пользователей, а затем «Обзор». Ваша учетная запись, вероятно, появится сразу же. Нажмите на значок редактирования и измените значение поля user_level с 0 на 1. Вот и все. Вы не заметите никакой разницы в нашем приложении сразу, но когда мы добавим администратора, у него будет обычная учетная запись, и у вашей учетной записи будут другие возможности.
Процесс входа в систему работает следующим образом:

- Посетитель вводит данные пользователя и отправляет форму
- Если имя пользователя и пароль верны, мы можем начать сеанс
- Если имя пользователя и пароль неверны, мы снова показываем форму с сообщением

Файл signin.php находится ниже.

```php
<?php
//signin.php
include 'connect.php';
include 'header.php';

echo '<h3>Sign in</h3>';

//first, check if the user is already signed in. If that is the case,
// there is no need to display this page
if(isset($_SESSION['signed_in']) && $_SESSION['signed_in'] == true)
{
    echo 'You are already signed in, you can <a href="signout.php">sign out</a> if you want.';
}
else
{
    if($_SERVER['REQUEST_METHOD'] != 'POST')
    {
        /*the form hasn't been posted yet, display it
          note that the action="" will cause the form to post to
          the same page it is on */
        echo '<form method="post" action="">
            Username: <input type="text" name="user_name" />
            Password: <input type="password" name="user_pass">
            <input type="submit" value="Sign in" />
         </form>';
    }
    else
    {
        /* so, the form has been posted, we'll process the data in three steps:
            1. Check the data
            2. Let the user refill the wrong fields (if necessary)
            3. Varify if the data is correct and return the correct response
        */
        $errors = array();

        if(!isset($_POST['user_name']))
        {
            $errors[] = 'The username field must not be empty.';
        }

        if(!isset($_POST['user_pass']))
        {
            $errors[] = 'The password field must not be empty.';
        }

        if(!empty($errors)) /*check for an empty array, if there are errors,
         they're in this array (note the ! operator)*/
        {
            echo 'Uh-oh.. a couple of fields are not filled in correctly..';
            echo '<ul>';
            foreach($errors as $key => $value) /* walk through the array so all
            the errors get displayed */
            {
                echo '<li>' .
            }
            echo '</ul>';
        }
        else
```

```php
    {
        //the form has been posted without errors, so save it
        //notice the use of mysql_real_escape_string, keep everything safe!
        //also notice the sha1 function which hashes the password
        $sql = "SELECT
                    user_id,
                    user_name,
                    user_level
                FROM
                    users
                WHERE
                    user_name = '" .mysqli_real_escape_string($_POST['user_name']) . "'
                AND
                    user_pass = '" .sha1($_POST['user_pass']) . "'";

        $result = mysqli_query($link, $sql);
        if(!$result)
        {
            //something went wrong, display the error
            echo 'Something went wrong while signing in. Please try again later.';
            //echo mysql_error();
        }
        else
        {
            //the query was successfully executed, there are 2 possibilities
            //1.
            //2.
            if(mysqli_num_rows($result) == 0)
            {
                echo 'You have supplied a wrong user/password combination.'
            }
            else
            {
                //set the $_SESSION['signed_in'] variable to TRUE
                $_SESSION['signed_in'] = true;

                //we also put the user_id and user_name values in the $_SESSION,
                //so we can use it at various pages
                while($row = mysql_fetch_assoc($result))
                {
                    $_SESSION['user_id'] = $row['user_id'];
                    $_SESSION['user_name'] = $row['user_name'];
                    $_SESSION['user_level'] = $row['user_level'];
                }

                echo 'Welcome, ' .
            }
        }
    }
}

include 'footer.php';
?>
```

Это запрос, который находится в файле signin.php:

SELECT
  user_id,
  user_name,
  user_level
FROM
  users
WHERE
  user_name = ‘» . mysql_real_escape_string($_POST[‘user_name’]) . «‘

AND
user_pass = '» . sha1($_POST['user_pass'])
Очевидно, нам нужна проверка, чтобы сказать, принадлежат ли предоставленные учетные данные существующему пользователю. Многие сценарии извлекают пароль из базы данных и сравнивают его с помощью PHP. Если мы сделаем это напрямую через SQL, пароль будет сохранен в базе данных один раз при регистрации и никогда не покинет его. Это безопаснее, потому что все реальные действия происходят на уровне базы данных, а не в нашем приложении.
Если пользователь вошел в систему успешно, мы делаем несколько вещей:

```
<?php
//set the $_SESSION['signed_in'] variable to TRUE
$_SESSION['signed_in'] = true;
//we also put the user_id and user_name values in the $_SESSION, so we can use it at various pages
while($row = mysql_fetch_assoc($result))
{
   $_SESSION['user_id'] = $row['user_id'];
   $_SESSION['user_name'] = $row['user_name'];
}
?>
```

Во-первых, мы установили для переменной $ sign_in '$ _SESSION значение true, чтобы мы могли использовать его на других страницах, чтобы убедиться, что пользователь вошел в систему. Мы также поместили имя пользователя и идентификатор пользователя в переменную $ _SESSION для использования на другой странице. , Наконец, мы показываем ссылку на обзор форума, чтобы пользователь мог сразу начать.
Конечно, для входа требуется другая функция: выход! Процесс выхода на самом деле намного проще, чем процесс входа. Поскольку вся информация о пользователе хранится в переменных $ _SESSION, все, что нам нужно сделать, это сбросить их и отобразить сообщение.
Теперь, когда мы установили переменные $ _SESSION, мы можем определить, вошел ли кто-то в систему. Давайте сделаем последнее простое изменение в header.php:
Заменить:

```
<div id=»userbar»>Hello Example.
```

С:

```
<div id="userbar">
<?php
if($_SESSION['signed_in'])
{
    echo 'Hello'
}
else{
    echo '<a href="signin.php">Sign in</a> or <a href="signup.php">create an account</a>.';
}
?>
</div>
```

Если пользователь вошел в систему, он увидит его имя на главной странице со ссылкой на страницу выхода. Наша аутентификация завершена! К настоящему времени наш форум должен выглядеть так:

Шаг 7: Создание категории

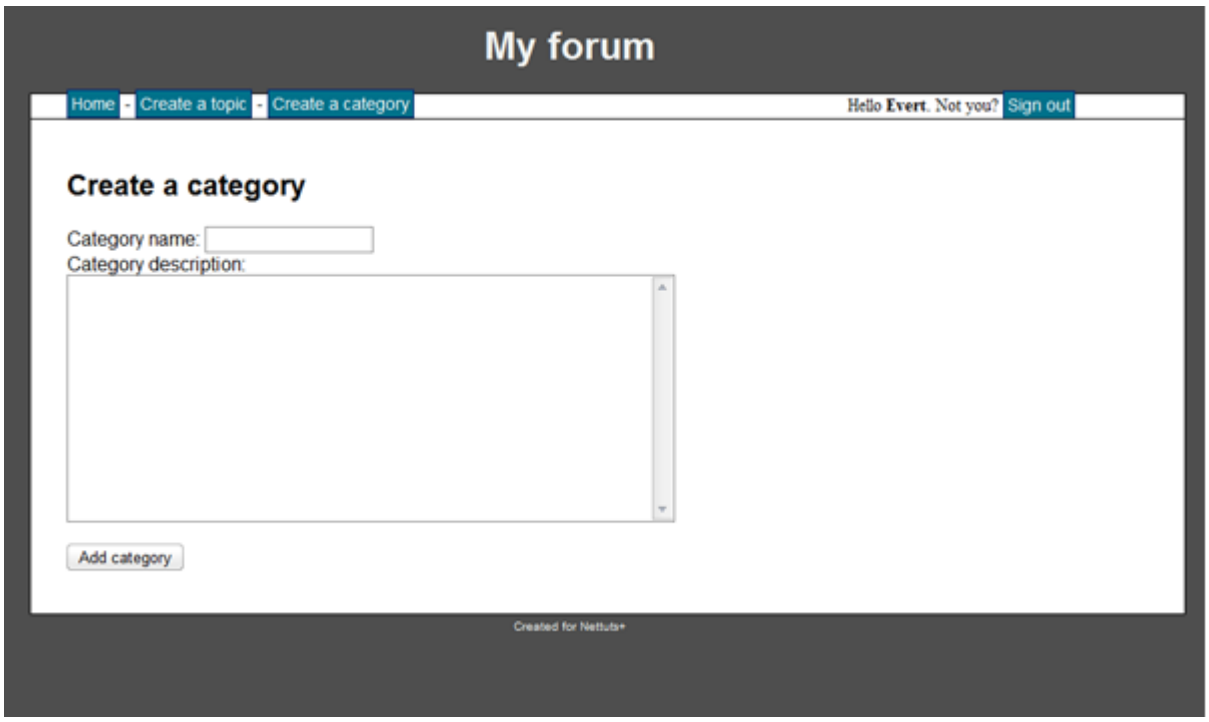
Мы хотим создавать категории, поэтому давайте начнем с создания формы.

```
<form method="post" action="">
    Category name: <input type="text" name="cat_name"/>
    Category description: <textarea name="cat_description"></textarea>
    <input type="submit" value="Add category"/>
</form>
```

Этот шаг очень похож на шаг 4 (регистрация пользователя), поэтому я не буду здесь подробно объяснять. Если вы выполнили все шаги, вы сможете понять это довольно быстро.

```php
<?php
//create_cat.php
include 'connect.php';

if($_SERVER['REQUEST_METHOD'] != 'POST')
{
    //the form hasn't been posted yet, display it
    echo '<form method='post' action=''>
        Category name: <input type='text' name='cat_name'/>
        Category description: <textarea name='cat_description' /></textarea>
        <input type='submit' value='Add category' />
     </form>';
}
else
{
    //the form has been posted, so save it
    $sql = "INSERT INTO categories(cat_name, cat_description)
       VALUES(" . mysql_real_escape_string($_POST['cat_name']) . "', " .
    $result = mysqli_query($link, $sql);
    if(!$result)
    {
        //something went wrong, display the error
        echo 'Error' .
    }
    else
    {
        echo 'New category successfully added.';
    }
}
?>
```

Как видите, мы запустили скрипт с проверкой $ _SERVER, после проверки, есть ли у пользователя права администратора, необходимые для создания категории. Форма отображается, если она еще не была отправлена. Если это так, значения сохраняются. Еще раз, SQL-запрос подготовлен и затем выполнен.



Шаг 8: Добавление категорий в index.php

Мы создали несколько категорий, поэтому теперь мы можем отображать их на первой странице. Давайте добавим следующий запрос в область содержимого index.php.

```
SELECT
    categories.cat_id,
    categories.cat_name,
    categories.cat_description,
FROM
    categories
```

Этот запрос выбирает все категории, их имена и описания из таблицы категорий. Нам нужно всего лишь немного PHP для отображения результатов. Если мы добавим эту часть точно так же, как и на предыдущих шагах, код будет выглядеть следующим образом.

```php
<?php
//create_cat.php
include 'connect.php';
include 'header.php';
$sql = "SELECT
            cat_id,
            cat_name,
            cat_description,
        FROM
            categories";

$result = mysqli_query($link, $sql);

if(!$result)
{
    echo 'The categories could not be displayed, please try again later.';
}
else
{
    if(mysqli_num_rows($result) == 0)
    {
        echo 'No categories defined yet.';
    }
    else
    {
        //prepare the table
        echo '<table border="1">
              <tr>
                <th>Category</th>
                <th>Last topic</th>
              </tr>';

          while($row = mysqli_fetch_assoc($result))
          {

echo '<tr>';
    echo '<td class="leftpart">';
        echo '<h3><a href="category.php?id=">
        Category name</a></h3> Category description goes here';
    echo '</td>';
    echo '<td class="rightpart">';
            echo '<a href="topic.php?id=">Topic subject</a> at 10-10';
    echo '</td>';
echo '</tr>';

include 'footer.php';
?>
```

Обратите внимание, как мы используем cat_id для создания ссылок на category.php. Все ссылки на эту страницу будут выглядеть следующим образом: category.php? Cat_id = x, где x может быть любым числовым значением. Это может быть новым для вас. Мы можем проверить URL с помощью PHP для значений $ _GET. Например, у нас есть эта ссылка:

category.php?cat_id=23

Оператор echo $ _GET [ёcat_id ‘];’ будет отображаться «23». В следующих нескольких шагах мы будем использовать это значение для получения тем при просмотре одной

категории, но темы нельзя просмотреть, если мы их еще не создали. Итак, давайте создадим несколько тем!

Шаг 9: Создание темы

На этом этапе мы объединяем методы, которые мы изучили на предыдущих этапах. Мы проверяем, вошел ли пользователь, мы будем использовать входной запрос для создания темы и создания некоторых основных HTML-форм.

Структура create_topic.php вряд ли может быть объяснена в виде списка или чего-то еще, поэтому она переписана в псевдокоде.

```
<?php
if (user is signed in)
{
    //the user is not signed in
}
else
{
    //the user is signed in
    if(form has not been posted)
    {
        //show form
    }
    else
    {
        //process form
    }
}
?>
```

Вот реальный код этой части нашего форума, посмотрите объяснения под кодом, чтобы увидеть, что он делает.

```php
<?php
//create_cat.php
include 'connect.php';
include 'header.php';

echo '<h2>Create a topic</h2>';
if($_SESSION['signed_in'] == false)
{
    //the user is not signed in
    echo 'Sorry, you have to be <a href=»/forum/signin.php»>signed in</a> to create a topic.';
}
else
{
    //the user is signed in
    if($_SERVER['REQUEST_METHOD'] != 'POST')
    {
        //the form hasn't been posted yet, display it
        //retrieve the categories from the database for use in the dropdown
        $sql = "SELECT
                    cat_id,
                    cat_name,
                    cat_description
                FROM
                    categories";

        $result = mysqli_query($link, $sql);

        if(!$result)
        {
            //the query failed, uh-oh 😕
            echo 'Error while selecting from database.'
        }
        else
        {
            if(mysqli_num_rows($result) == 0)
            {
                //there are no categories, so a topic can't be posted
                if($_SESSION['user_level'] == 1)
                {
                    echo 'You have not created categories yet.';
                }
                else
                {
                    echo 'Before you can post a topic, you must wait for an admin to create some categories.';
                }
            }
            else
            {

                echo '<form method="post" action="">
                    Subject: <input type="text" name="topic_subject" />
                    Category:';

                echo '<select name="topic_cat">';
                    while($row = mysqli_fetch_assoc($result))
                    {
                        echo '<option value="' . $row['cat_id'] . '">' .
                    }
                echo '</select>';

```

```
                echo 'Message: <textarea name="post_content" /></textarea>
                    <input type="submit" value="Create topic" />
                 </form>';
            }
        }
    }
    else
    {
        //start the transaction
        $query = "BEGIN WORK";
        $result = mysqli_query($link, $query);

        if(!$result)
        {
            //Damn!
            echo 'An error occured while creating your topic.'
        }
        else
        {

            //the form has been posted, so save it
            //insert the topic into the topics table first, then we'll save the post into the posts table
            $sql = "INSERT INTO
                        topics(topic_subject,
                               topic_date,
                               topic_cat,
                               topic_by)
                    VALUES('" . mysqli_real_escape_string($_POST['topic_subject']) . "',
                               NOW(),
                               " . mysqli_real_escape_string($_POST['topic_cat']) . ",
                               " . $_SESSION['user_id'] . "
                               )";

            $result = mysqli_query($link, $sql);
            if(!$result)
            {
                //something went wrong, display the error
                echo 'An error occured while inserting your data.'
                $sql = "ROLLBACK;";
                $result = mysqli_query($link, $sql);
            }
            else
            {
                //the first query worked, now start the second, posts query
                //retrieve the id of the freshly created topic for usage in the posts query
                $topicid = mysqli_insert_id($link);

                $sql = "INSERT INTO
                            posts(post_content,
                                  post_date,
                                  post_topic,
                                  post_by)
                        VALUES
                            ('" . mysql_real_escape_string($_POST['post_content']) . "',
                                  NOW(),
```

```
                                    " . $topicid . ",
                                    " . $_SESSION['user_id'] . "
                                )";
                    $result = mysqli_query($link, $sql);

                    if(!$result)
                    {
                        //something went wrong, display the error
                        echo 'An error occured while inserting your post.'
                        $sql = "ROLLBACK;";
                        $result = mysql_query($sql);
                    }
                    else
                    {
                        $sql = "COMMIT;";
                        $result = mysqli_query($link, $sql);

                        //after a lot of work, the query succeeded!
                        echo 'You have successfully created <a href="topic.php?id='. $topicid . '">your new topic</a>.';
                    }
                }
            }
        }
    }

    include 'footer.php';
    ?>
```

Будем обсуждать эту страницу в двух частях, показывая форму и обрабатывая форму.

Отображение формы

Мы начинаем с простой формы HTML. Здесь есть что-то особенное, потому что мы используем выпадающий список. Этот раскрывающийся список заполнен данными из базы данных, используя этот запрос:

```
SELECT
    cat_id,
    cat_name,
    cat_description
FROM
    categories
```

Это единственная потенциально запутанная часть здесь; это довольно кусок кода, как вы можете увидеть, посмотрев на файл create_topic.php в нижней части этого шага.

Обработка формы

Процесс сохранения темы состоит из двух частей: сохранение темы в таблице тем и сохранение первого сообщения в таблице постов. Это требует чего-то достаточно продвинутого, что выходит за рамки этого урока. Это называется транзакцией, что в основном означает, что мы начинаем с выполнения команды start, а затем откатываем при возникновении ошибок базы данных и фиксируем, когда все прошло хорошо. Подробнее о транзакциях .

```
<?php
//start the transaction
$query = «BEGIN WORK;»;
$result = mysql_query($query);
//stop the transaction
$sql = «ROLLBACK;»;
$result = mysql_query($sql);
//commit the transaction
$sql = «COMMIT;»;
$result = mysql_query($sql);
?>
```

Первый запрос, используемый для сохранения данных, — это запрос на создание темы, который выглядит следующим образом:

```
INSERT INTO
    topics(topic_subject,
            topic_date,
```

```
            topic_cat,
            topic_by)
VALUES('» . mysql_real_escape_string($_POST['topic_subject']) . «',
      NOW(),
      » . mysql_real_escape_string($_POST['topic_cat']) . «,
      » . $_SESSION['user_id'] . «)
```

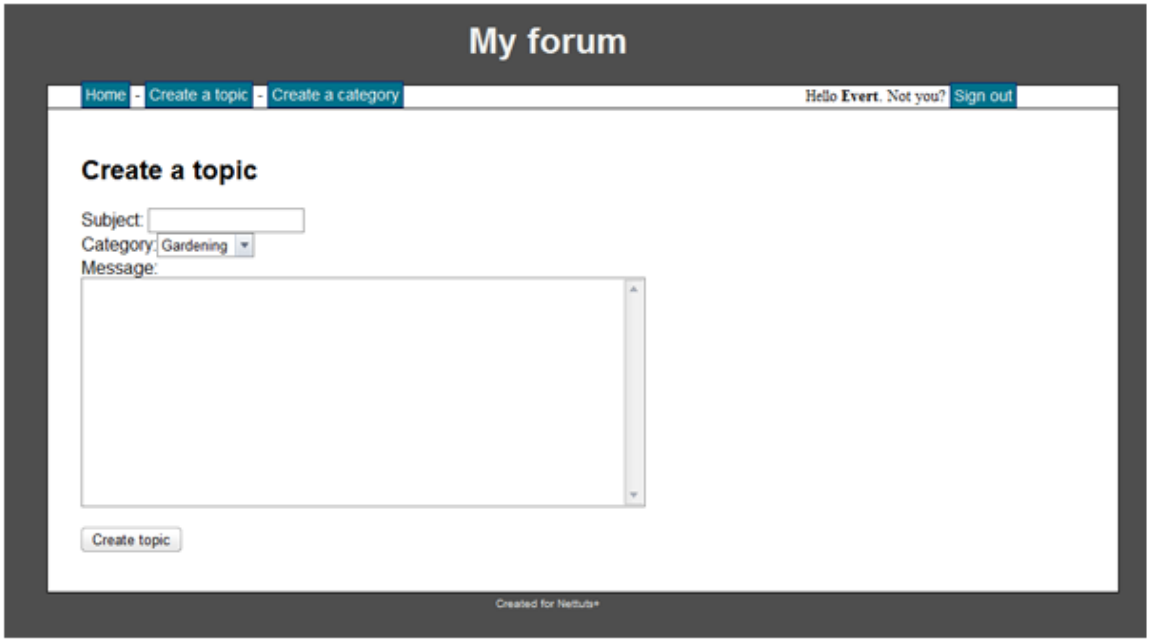
Сначала определяются поля, затем значения для вставки. Первый из них мы видели раньше, это просто строка, которая становится безопасной с помощью mysql_real_escape_string (). Второе значение, NOW (), является функцией SQL для текущего времени. Третье значение, однако, это значение, которое мы не видели раньше. Это относится к (действительному) идентификатору категории. Последнее значение относится к (существующему) user_id, который в данном случае является значением $ _SESSION [«user_id»]. Эта переменная была объявлена во время входа в систему.

Если запрос выполнен без ошибок, мы переходим ко второму запросу. Помните, что мы все еще делаем транзакцию здесь. Если бы у нас были ошибки, мы бы использовали команду ROLLBACK.

```
INSERT INTO
      posts(post_content,
      post_date,
      post_topic,
      post_by)
VALUES
      ('» . mysql_real_escape_string($_POST['post_content']) . «',
       NOW(),
       » . $topicid . «,
       » . $_SESSION['user_id'] . «)
```

Первое, что мы делаем в этом коде, это используем mysqli_insert_id (), чтобы получить последний сгенерированный идентификатор из поля topic_id в таблице тем. Как вы помните из первых шагов этого урока, идентификатор генерируется в базе данных с помощью auto_increment.

Затем сообщение вставляется в таблицу сообщений. Этот запрос очень похож на запрос по темам. Разница лишь в том, что этот пост относится к теме, а тема относится к категории. С самого начала мы решили создать хорошую модель данных, и вот результат: хорошая иерархическая структура.



Шаг 10: просмотр категории
Мы собираемся сделать обзорную страницу для отдельной категории. Мы только что создали категорию, было бы удобно просматривать все темы в ней. Сначала создайте страницу под названием category.php.
Краткий список того, что нам нужно:
Необходим для отображения категории

- cat_name
- cat_description

Необходим для отображения всех тем

- topic_id
- topic_subject
- topic_date
- topic_cat

Давайте создадим два SQL-запроса, которые извлекают именно эти данные из базы данных.

```
SELECT
   cat_id,
   cat_name,
   cat_description
FROM
   categories
WHERE
   cat_id = » . mysql_real_escape_string($_GET[‘id’])
```

Приведенный выше запрос выбирает все категории из базы данных.

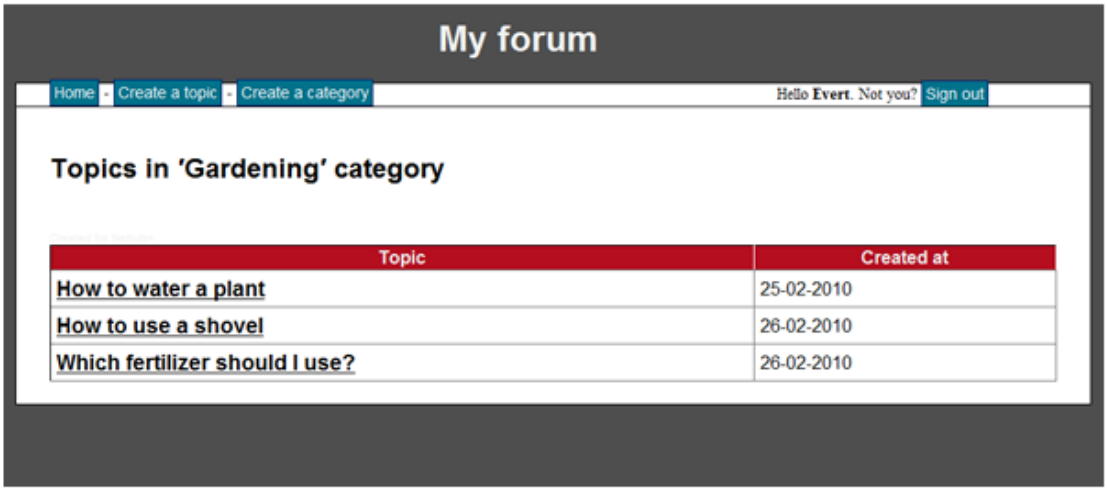
```
SELECT
   topic_id,
   topic_subject,
   topic_date,
   topic_cat
FROM
   topics
WHERE
   topic_cat = » . mysql_real_escape_string($_GET[‘id’])
```

Вышеуказанный запрос выполняется в цикле while, в котором мы повторяем категории. Делая это таким образом, мы увидим все категории и последние темы для каждой из них.
Полный код category.php будет следующим:

```php
<?php
//create_cat.php
include 'connect.php';
include 'header.php';

//first select the category based on $_GET['cat_id']
$sql = "SELECT
            cat_id,
            cat_name,
            cat_description
        FROM
            categories
        WHERE
            cat_id = " . mysqli_real_escape_string($_GET['id']);

$result = mysqli_query($link, $sql);

if (!$result)
{
    echo 'The category could not be displayed, please try again later.'
}
else
{
    if(mysqli_num_rows($result) == 0)
    {
        echo 'This category does not exist.';
    }
    else
    {
        //display category data
```

```php
        while($row = mysqli_fetch_assoc($result))
        {
            echo '<h2>Topics in '
        }

        //do a query for the topics
        $sql = "SELECT
                    topic_id,
                    topic_subject,
                    topic_date,
                    topic_cat
                FROM
                    topics
                WHERE
                    topic_cat = " . mysqli_real_escape_string($_GET['id']);

        $result = mysqli_query($link, $sql);

        if(!$result)
        {
            echo 'The topics could not be displayed, please try again later.';
        }
        else
        {
            if(mysqli_num_rows($result) == 0)
            {
                echo 'There are no topics in this category yet.';
            }
            else
            {
                //prepare the table
                echo '<table border="1">
                      <tr>
                        <th>Topic</th>
                        <th>Created at</th>
                      </tr>';

                while($row = mysqli_fetch_assoc($result))
                {
                    echo '<tr>';
                        echo '<td class="leftpart">';
                            echo '<h3><a href="topic.php?id=' . $row['topic_id'] . '">' .
                        echo '</td>';
                        echo '<td class=»rightpart»>'’;
                            echo date('dm-Y', strtotime($row['topic_date']));
                        echo '</td>';
                    echo '</tr>'’;
                }
            }
        }
    }
}

include 'footer.php';
?>
```

И вот окончательный результат нашей страницы категорий:



Шаг 11: Просмотр темы

SQL-запросы на этом этапе являются сложными. PHP-часть — это все то, что вы видели раньше. Давайте посмотрим на запросы. Первый получает основную информацию о теме:

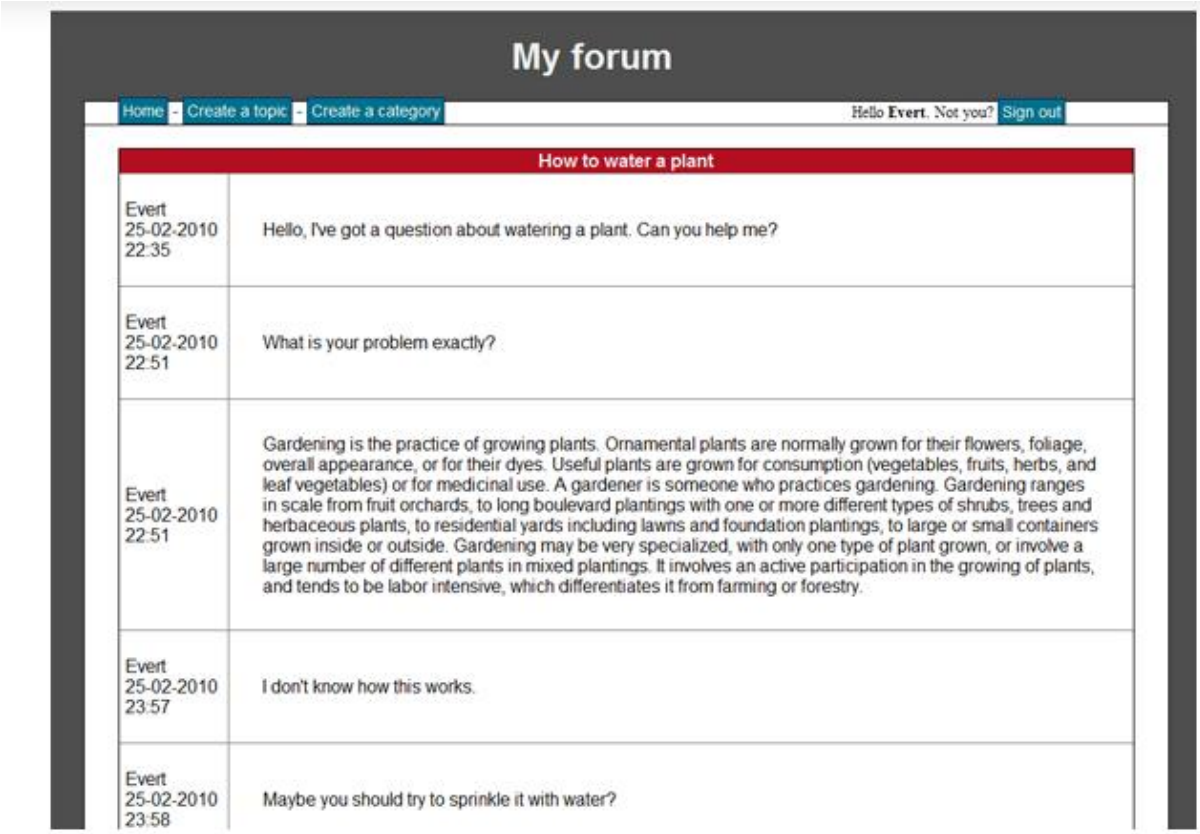
```
SELECT
   topic_id,
   topic_subject
FROM
   topics
WHERE
   topics.topic_id = » . mysql_real_escape_string($_GET[‘id’])
```

Эта информация отображается в заголовке таблицы, которую мы будем использовать для отображения всех данных. Далее мы извлекаем все сообщения в этой теме из базы данных. Следующий запрос дает нам именно то, что нам нужно:

```
SELECT
   posts.post_topic,
   posts.post_content,
   posts.post_date,
   posts.post_by,
   users.user_id,
   users.user_name
FROM
   posts
LEFT JOIN
   users
ON
   posts.post_by = users.user_id
WHERE
   posts.post_topic = » . mysql_real_escape_string($_GET[‘id’])
```

На этот раз нам нужна информация от пользователей и таблицы сообщений — поэтому мы снова используем LEFT JOIN. Условие таково: идентификатор пользователя должен совпадать с полем post_by. Таким образом, мы можем показать имя пользователя, который ответил на каждый пост.

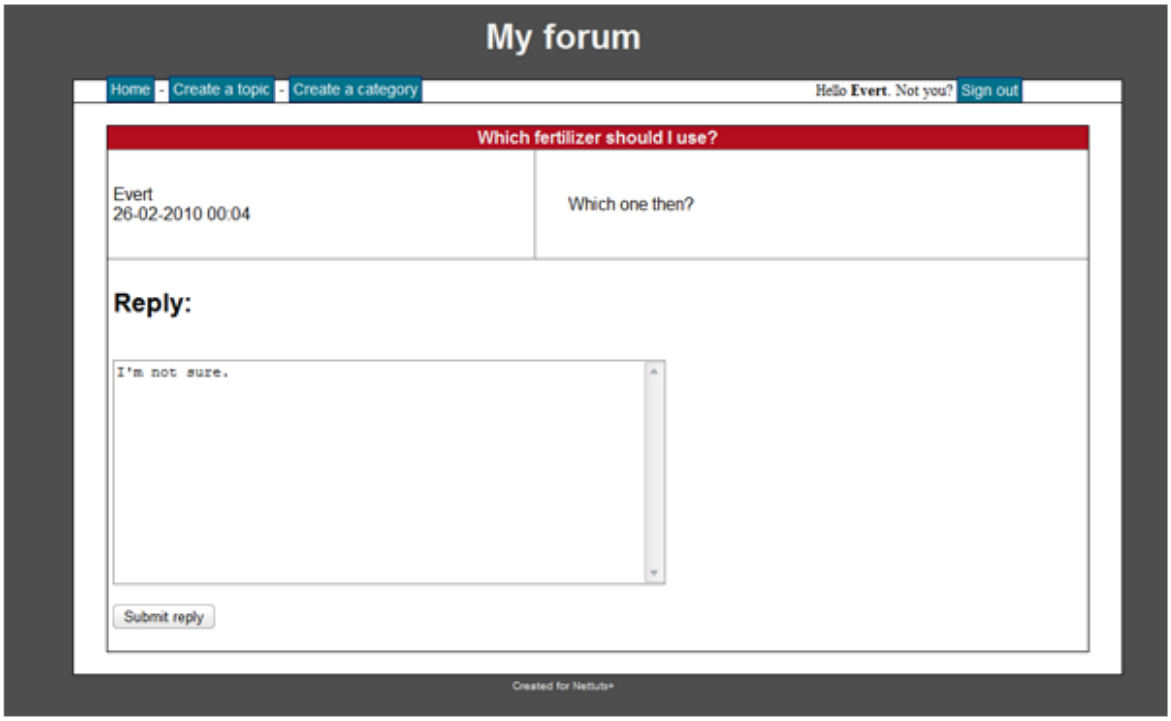
Окончательный вид темы выглядит так:



Шаг 12: Добавление ответа

Давайте создадим последнюю недостающую часть этого форума, возможность добавить ответ. Начнем с создания формы:

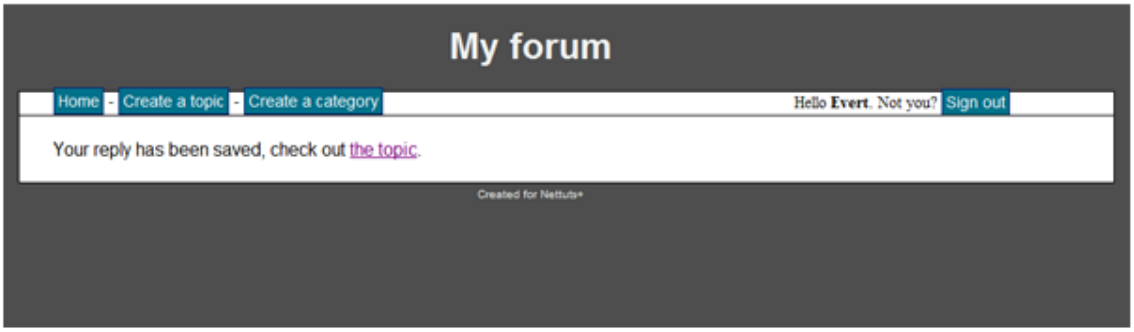
```
<form method="post" action="reply.php?id=5">
    <textarea name="reply-content"></textarea>
    <input type="submit" value="Submit reply" />
</form>
```



Полный код reply.php выглядит следующим образом.

```php
<?php
//create_cat.php
include 'connect.php';
include 'header.php';

if($_SERVER['REQUEST_METHOD'] != 'POST')
{
    //someone is calling the file directly, which we don't want
    echo 'This file cannot be called directly.';
}
else
{
    //check for sign in status
    if(!$_SESSION['signed_in'])
    {
        echo 'You must be signed in to post a reply.';
    }
    else
    {
        //a real user posted a real reply
        $sql = "INSERT INTO
                    posts(post_content,
                          post_date,
                          post_topic,
                          post_by)
                VALUES ('" . $_POST['reply-content'] . "',
                        NOW(),
                        " . mysqli_real_escape_string($_GET['id']) . ",
                        " . $_SESSION['user_id'] . ")";

        $result = mysqli_query($link, $sql);

        if(!$result)
        {
            echo 'Your reply has not been saved, please try again later.';
        }
        else
        {
            echo 'Your reply has been saved, check out <a href=»topic.php?id='
            .htmlentities($_GET['id']) . '">the topic</a>.';
        }
    }
}

include 'footer.php';
?>
```

Комментарии в коде довольно подробно описывают, что происходит. Мы проверяем реального пользователя и затем вставляем сообщение в базу данных.


My forum
Home - Create a topic - Create a category
Hello Evert. Not you? Sign out
Your reply has been saved, check out the topic.
Created for Nettuts+