Используем базу данных, созданную в лабораторных работач 1-3.

Создадим папку Classes, содержащую все файлы с классами и реализацией их методов. Также она должна сожержать папку библиотеки PHPExcel и класс PHPExcel. Пропишем основные классы.

* **DB.class.php**

<?php

class DB {

protected $db\_host = '127.0.0.1:3307'; // адрес сервера

protected $db\_user = 'root'; //Имя пользователя

protected $db\_pass = ''; //Пароль

protected $db\_name = 'bib'; //Имя БД

*//подключение к базе данных*

*//убедитесь, что данная функция вызывается на каждой странице, где необходима работа с бд*

public function connect() {

$connection = mysql\_connect($this->db\_host, $this->db\_user, $this->db\_pass) or die('no connection');

mysql\_select\_db($this->db\_name, $connection);

return true;

}

*//принимает набор строк mysql и возвращает ассоциативный массив,*

*//где ключами массива являются имена столбцов в наборе строк.*

*//Если для singleRow установлено значение true, он возвращает одну строку вместо массива строк.*

public function processRowSet($rowSet, $singleRow=false)

{

$resultArray = array();

while($row = mysql\_fetch\_assoc($rowSet)) //ассоциативный массив

{

array\_push($resultArray, $row);

}

if($singleRow === true)

return $resultArray[0];

return $resultArray;

}

*//Производит выборку строк из базы данных.*

*//возвращает полную строку или строки из таблицы $table, используя $where в качестве предложения where.*

*//возвращаемое значение - ассоциативный массив с именами столбцов в качестве ключей.*

public function select($table, $where)

{

$sql = "SELECT \* FROM $table WHERE $where";

$result = mysql\_query($sql);

if(mysql\_num\_rows($result) == 1)

return $this->processRowSet($result, true);

return $this->processRowSet($result);

}

*//Производит выборку всех строк из базы данных из таблицы $table.*

*//возвращаемое значение - ассоциативный массив с именами столбцов в качестве ключей.*

public function selectAll($table)

{

$sql = "SELECT \* FROM $table";

$result = mysql\_query($sql);

if(mysql\_num\_rows($result) > 0)

return $this->processRowSet($result);

return $this->processRowSet($result);

}

*// Обновляет текущую строку в базе данных.*

*// принимает массив данных, где ключами в массиве являются имена столбцов,*

*// а значения - это данные, которые будут вставлены в эти столбцы.*

*// $table - это имя таблицы, а $where - предложение sql where.*

public function update($data, $table, $where) {

foreach ($data as $column => $value) {

$sql = "UPDATE $table SET $column = $value WHERE $where";

mysql\_query($sql) or die(mysql\_error());

}

return true;

}

*// Вставляет новую строку в базу данных.*

*// принимает массив данных, где ключами в массиве являются имена столбцов,*

*// а значения - это данные, которые будут вставлены в эти столбцы.*

*// $table - это имя таблицы.*

public function insert($data, $table) {

$columns = "";

$values = "";

foreach ($data as $column => $value) {

$columns .= ($columns == "") ? "" : ", ";

$columns .= $column;

$values .= ($values == "") ? "" : ", ";

$values .= $value;

}

$sql = "INSERT into $table ($columns) values ($values)";

mysql\_query($sql) or die(mysql\_error());

//возвращает идентификатор пользователя в базе данных.

return mysql\_insert\_id();

}

}

?>

**класс Пользователей (User.class.php)**

Этот класс будет определять каждого пользователя.

В этом классе мы будем использовать конструктор - это функция, которая автоматически вызывается при создании очередной копии класса. Это позволяет нам автоматически публиковать некоторые атрибуты после создания проекта. В этом классе конструктор будет брать единственный аргумент: ассоциативный массив, который содержит один ряд из таблицы users нашей БД.

<?php

require\_once 'UserTools.class.php';

require\_once 'DB.class.php';

class User {

public $idu; // id пользователя

public $mail; // его почта

public $name; // его логин

public $ind; // его пароль

//Конструктор вызывается всякий раз, когда создается новый объект.

//Принимает ассоциативный массив со строкой БД в качестве аргумента.

function \_\_construct($data) {

$this->idu = (isset($data['idu'])) ? $data['idu'] : "";

$this->mail = (isset($data['mail'])) ? $data['mail'] : "";

$this->name = (isset($data['name'])) ? $data['name'] : "";

$this->ind = (isset($data['ind'])) ? $data['ind']: "";

}

public function save($isNewUser = false) {

//создается новый объект базы данных.

$db = new DB();

// если пользователь уже зарегистрирован,

// просто обновляется информация о нем.

if(!$isNewUser) {

$data = array(

"mail" => "'$this->mail'",

"name" => "'$this->name'",

"ind" => "'$this->ind'"

);

//обновляет строку в базе данных

$db->update($data, 'users', 'idu = '.$this->idu);

}else {

// если пользователь заходит впервые

$data = array(

"mail" => "'$this->mail'",

"name" => "'$this->name'",

"ind" => "'$this->ind'"

);

$this->id = $db->insert($data, 'users'); // его данные добавляются в таблицу Users

}

return true;

}

}

?>

* **класс UserTools.class.php**

Этот класс будет содержать функции, которые имеют отношение к пользователям: login(), logout(), checkUsernameExists() и get(). Но с расширением данного приложения, Вы можете добавить еще множество других.

<?php

require\_once 'User.class.php'; //Подключение используемых классов

require\_once 'DB.class.php';

class UserTools {

*// Вход пользователя в систему. Сначала проверяется, соответствуют ли*

*// имя пользователя и пароль строке в базе данных.*

*// Если это успешно, устанавливаются переменные сеанса*

*// и сохраняется объект пользователя.*

public function login($name, $ind)

{

$hashedPassword = md5($ind); //хэшируем введенный пароль, т.к. в таблице БД он захэширован

$result = mysql\_query("SELECT \* FROM users WHERE name = '$name' AND ind = '$hashedPassword'"); // Ищем пользователя в БД

if(mysql\_num\_rows($result) == 1) // если в результате запроса была возвращена одна строка, то создается сессия с данным пользователем

{

$\_SESSION["users"] = serialize(new User(mysql\_fetch\_assoc($result)));

$\_SESSION["logged\_in"] = 1;

return true;

}else{

return false;

}

}

// Выход пользователя из системы. Уничтожает переменные сеанса.

public function logout() {

unset($\_SESSION["users"]);

unset($\_SESSION["login\_time"]);

unset($\_SESSION["logged\_in"]);

session\_destroy();

}

// Проверяет, существует ли данное имя пользователя в бд.

// Вызывается при регистрации, чтобы убедиться, что все имена пользователей уникальны.

public function checkUsernameExists($name) {

$result = mysql\_query("SELECT idu from users where name='$name'");

if(mysql\_num\_rows($result) == 0)

{

return false;

}else{

return true;

}

}

// получает пользователя

// возвращает объект User. Принимает идентификатор пользователя в качестве входа

public function get($idu)

{

$db = new DB();

$result = $db->select('users', "idu = $idu");

return new User($result);

}

}

?>

**класс Книги (Item.class.php)(соответствует таблице books)**

Этот класс будет определять книги.

В этом классе мы будем использовать конструктор, логика которого такая же, как и в классе User. В этом классе конструктор будет так же брать единственный аргумент: ассоциативный массив, который содержит один ряд из таблицы items нашей БД.

<?php

require\_once 'ItemTools.class.php';

require\_once 'DB.class.php';

class Item {

public $idb;

public $nameb;

public $isbm;

public $esbm;

public $link;

public $limt;

public $pag;

public $price;

//Конструктор вызывается всякий раз, когда создается новый объект.

//Принимает ассоциативный массив со строкой БД в качестве аргумента.

function \_\_construct($data) {

$this->idb = (isset($data['idb'])) ? $data['idb'] : "";

$this->nameb = (isset($data['nameb'])) ? $data['nameb'] : "";

$this->isbm = (isset($data['isbm'])) ? $data['isbm'] : "";

$this->esbm = (isset($data['esbm'])) ? $data['esbm'] : "";

$this->link = (isset($data['link'])) ? $data['link'] : "";

$this->limt = (isset($data['limt'])) ? $data['limt'] : "";

$this->pag = (isset($data['pag'])) ? $data['pag'] : "";

$this->price = (isset($data['price'])) ? $data['price'] : "";

}

}

?>

**класс ItemTools.class.php**

Данный класс представляет собой набор методов для работы с книгами, данные о которых получаются из базы данных.

<?php

require\_once 'Item.class.php';

require\_once 'DB.class.php';

class ItemTools {

// получает список книг

// возвращает список объектов Item. Принимает идентификатор ткниги в качестве входа

public function getItems()

{

$db = new DB();

$nam = unserialize($\_SESSION['users']); //Извлекаем из переменной сессии объект Users

$idu = $nam->idu; //Определяем id пользователя сессии

$result =$db->select('books inner join bubuy on books.idb=bubuy.idb',"idu = ".$idu); //Запрос в БД. Ищем купленные этим пользователем книги

for ($ind = 0; $ind < count($result); $ind++){

$items[$ind] = new Item($result[$ind]); //Заполняем массив объектов строками результата запроса

}

$\_SESSION["books"] = serialize($items); //Создаем переменную сессии и записываем туда массив объектов

return $items;

}

}

?>

Папка classes готова. Теперь выйдем из нее и создадим еще одну папку includes. В ней пропишем один файл.

global.inc.php необходим для каждой страницы сайта. Таким образом мы разместим все обычные операции, которые нам понадобятся на странице. К примеру, мы начнем session\_start(). Соединение с БД также откроется

<?php

//global.inc.php

require\_once 'classes/User.class.php';

require\_once 'classes/UserTools.class.php';

require\_once 'classes/Item.class.php';

require\_once 'classes/ItemTools.class.php';

require\_once 'classes/DB.class.php';

require\_once 'classes/PHPExcel.php';

//connect to the database

$db = new DB();

$db->connect();

//initialize UserTools object

$userTools = new UserTools();

$itemTools = new ItemTools();

//start the session

session\_start();

//refresh session variables if logged in

if(isset($\_SESSION['logged\_in'])) {

$user = unserialize($\_SESSION['users']);

$items = unserialize($\_SESSION['books']);

$\_SESSION['users'] = serialize($userTools->get($user->idu));

$\_SESSION['books'] = serialize($itemTools->getItems());

}

?>

Тут происходит несколько вещей. Прежде всего, мы открываем соединение с базой.

После соединения, мы начинаем функцию session\_start(). Функция создает сессию или продолжает текущую, если пользователь уже залогинен. Поскольку наше приложение рассчитано на то, чтобы пользователи входили\выходили, эта функция обязательна на каждой странице.

Далее мы проверяем залогинен ли юзер. Если да - мы обновим $\_SESSION['user'], чтобы отображать самую последнюю информацию о юзере. К примеру, если пользователь меняет свой e-mail, в сессии будет храниться еще старый. Но с помощью авто обновления такого не случится.

* **Регистрация пользователей (register.php)**

В этой странице у нас две части. Блок кода PHP в верхней части, и HTML - в нижней. PHP код загружает global.inc.php. Он также содержит информацию о проверке формы. Код HTML помогает нам создать простую форму. Более детальные объяснения в коде:

<?php

//register.php

require\_once 'includes/global.inc.php';

//инициализация php переменных, которые будут использваться в форме

$username = "";

$password = "";

$password\_confirm = "";

$email = "";

$error = "";

//проверка того, что форма была отправлена

if(isset($\_POST['submit-form'])) {

//извлечение $\_POST переменных из формы

$username = $\_POST['username'];

$password = $\_POST['password'];

$password\_confirm = $\_POST['password-confirm'];

$email = $\_POST['email'];

//инициализация переменных для валидации входных данных

$success = true;

$userTools = new UserTools();

//проверка того, что форма была заполнена корректно

//проверка, существует ли пользоватеем с таким логином в базе данных

if($userTools->checkUsernameExists($username))

{

$error .= "That username is already taken.<br/> \n\r";

$success = false;

}

//проверка пароля на совпадение

if($password != $password\_confirm) {

$error .= "Passwords do not match.<br/> \n\r";

$success = false;

}

if($success)

{

//подготовка данных для сохранения в объекте пользователя

$data['name'] = $username;

$data['ind'] = md5($password); //шифрование пароля

$data['mail'] = $email;

//создание нового объекта пользователя

$newUser = new User($data);

//сохранение нового пользователя в базе данных

$newUser->save(true);

//осуществление входа пользователя

$userTools->login($username, $password);

//переадресация пользователя на главную страницу

header("Location: index.php");

}

}

//если форма не была отправлена или были введены некоректные данные, то снова появляется форма регистрации

?>

<html>

<head>

<title>Registration</title>

<link rel='stylesheet' href='login.css' />// подключение css файла

</head>

<body>

<?php echo ($error != "") ? $error : ""; ?>

<form action="register.php" method="post">

<div class="login-page">

<div class="form">

<form class="register-form">

Username <input type="text" value="<?php echo $username; ?>" name="username"/>

Password <input type="password" value="<?php echo $password; ?>" name="password" />

Password (confirm) <input type="password" value="<?php echo $password\_confirm; ?>" name="password-confirm" />

E-Mail <input type="text" value="<?php echo $email; ?>" name="email" />

<button type="submit" value="Register" name="submit-form">create</button>

<p class="message">Already registered? <a href="login.php">Sign In</a></p>

</form>

</div>

</div>

</form>

<a href="index.php">Return to Homepage</a>

</body>

</html>

 Когда форма отправляется пользователем загружается register.php и все значения отправляются в переменную $\_POST.

Значение каждого поля ввода превращается в переменную ($name, $ind...). Если форма еще не отправлена, все переменные будут содержать пустую строку. Если форма отправлена, но не прошла проверку, эти переменные будут содержать значения из прошлого отправления, чтобы пользователю не пришлось заново вводить данные.

Также если форма не прошла проверку, будет показана переменная $error. В этой переменной будут содержаться причины того, почему форма не прошла проверку.

Переменная $\_POST в ассоциативном массиве переменных передается текущему скрипту с помощью метода HTTP POST. Значение каждого поля доступно в $\_POST используя названия поля в качестве ключа. Прежде всего мы проверяем была ли нажата кнопка "Отправить". Мы проверяем $\_POST['submit-form'] с помощью функции isset(), поскольку “submit-form” было название кнопки формы.

Как только мы знаем, что кнопка нажата, можем начинать проверку формы. Мы будем использовать класс UserTools для проверки доступности имени пользователя. Если такое имя уже существует, мы тогда ставим $success на false и выводим эту ошибку в $error. Следующий шаг - это проверить совпадают ли пароли, с помощью простого сравнения.

Если форма прошла проверку, мы размещаем информацию в массив $data и используем его для создания нового объекта User. Мы вызовем функцию save() того объекта User с параметрами $isNewUser - true. Далее мы войдем в систему с помощью функции login() и сделаем редирект на index.php.

**Страницы входа и выхода**

**login.php**

Также как и в register.php, login.php будет иметь проверку формы вверху и форму для входа снизу.

<?php

//login.php

require\_once 'includes/global.inc.php';

$error = "";

$username = "";

$password = "";

//проверка того, что форма была отправлена

if(isset($\_POST['submit-login'])) {

$username = $\_POST['username'];

$password = $\_POST['password'];

$userTools = new UserTools();

if($userTools->login($username, $password)){

//при успешном входе происходит переадресация на главную страницу

header("Location: index.php");

}else{

//если данный пользователь не был найден, то выводится сообщение и снова появлется форма входа

$error = "Incorrect username or password. Please try again.";

}

}

?>

<html>

<head>

<title>Login</title>

<link rel='stylesheet' href='login.css' />

</head>

<body>

<?php

if($error != "")

{

echo $error."<br/>";

}

?>

<div class="login-page">

<div class="form">

<form class="login-form" action="login.php" method="post">

Username<input type="text" name="username" value="<?php echo $username; ?>"/>

Password<input type="password" name="password" value="<?php echo $password; ?>"/>

<button type="submit" value="Login" name="submit-login">Login</button>

<p class="message">Not registered? <a href="register.php">Create an account</a></p>

</form>

</div>

</div>

</body>

</html>

**logout.php**

Страница выхода еще проще. Эта страница не нуждается в HTML. Функция этой страницы выйти из приложения с помощью функции logout(). После выхода пользователь редиректится на главную.

<?php

require\_once 'includes/global.inc.php';

$userTools = new UserTools();

$userTools->logout();

header("Location: index.php");

?>

**Страница настроек пользователя Settings.php**

<?php

require\_once 'includes/global.inc.php';

//проверка авторизации пользователя

if(!isset($\_SESSION['logged\_in'])) {

header("Location: login.php");

}

//получение объекта пользователя из сессии

$user = unserialize($\_SESSION['users']);

//инициализация php переменных для формы

$mail = $user->mail;

$message = "";

//проверка отправления формы

if(isset($\_POST['mail'])) {

//retrieve the $\_POST variables

$mail = $\_POST['mail'];

$user->mail = $mail;

$user->save();

$message = "Settings Saved<br/>";

}

//если форма не была отправлена или были введены некорректные данные, то форма появляется снова

?>

<html>

<head>

<title>Change Settings</title>

<link rel='stylesheet' href='settings.css' />

</head>

<body>

<div class="settings-page">

<div class="form">

<form class="settings" action="settings.php" method="post">

E-Mail <input type="text" value="<?php echo $mail; ?>" name="mail" />

<button type="submit" value="Update" name="submit-settings">Update</button>

<?php echo $message; ?>

</div>

</div>

<p><a href="index.php">Return to Homepage</a></p>

</body>

</html>

**index.php**

<?php

//welcome.php

require\_once 'includes/global.inc.php';

//проверка авторизации пользователя

if(!isset($\_SESSION['logged\_in'])) {

header("Location: login.php");

}

//получение объекта пользователя из сессии

$users = unserialize($\_SESSION['users']);

$books = unserialize($\_SESSION['books']); // Переменная сессии содержит информацию о книгах пользователя

?>

<html>

<head>

<title>Welcome <?php echo $users->name; ?></title>

<link rel='stylesheet' href='index.css' /> // Подключение css

</head>

<body>

<h1>

Welcome to our site <?php echo $users->name; ?>

</br>

You've been registered and logged in</h1>

<form action="logout.php">

<button type="submit">Log Out</button>

</form>

<form action="settings.php">

<button type="submit">Change my setting</button>

</form>

<form action="json.php">

<button type="submit">json-file</button>

</form>

<form action="excel.php">

<button type="submit">excel</button>

</form>

<?php

echo "<br/>Number of books = ", count($books), "<br/><br/>"; // Выводит количество купленных книг

// Построчно выводит информацию о книгах

for($i = 0; $i < count($books); $i++){

$b=(string)$books[$i]->link;

echo "Name : ", $books[$i]->nameb, "<br/>";

echo "<img src=$b width='100' hight='200' />", "<br/>"; // Отображение фото по ссылке из таблицы books

echo "Price : ", $books[$i]->price, "<br/><br/>";

}

?>

</body>

</html>

**Вывод в json**

Создадим файл json.php

<?php

require\_once 'includes/global.inc.php';

$js = unserialize($\_SESSION['books']);

echo json\_encode($js); // Вывод переменной сессии в формате json

?>

**Вывод в excel**

**Создание файла excel.php**

<?php

require\_once 'includes/global.inc.php';

$objPHPExcel = new PHPExcel(); // создание объекта PHPExcel

$objPHPExcel->getProperties()->setCreator("Maarten Balliauw")

->setLastModifiedBy("Maarten Balliauw")

->setTitle("Office 2007 XLSX Test Document")

->setSubject("Office 2007 XLSX Test Document")

->setDescription("Test document for Office 2007 XLSX, generated using PHP classes.")

->setKeywords("office 2007 openxml php")

->setCategory("Test result file");

$nam = unserialize($\_SESSION['books']);

$objPHPExcel->setActiveSheetIndex(0)->setCellValue("A1", "Название");

$objPHPExcel->setActiveSheetIndex(0)->setCellValue("B1", "Цена");

for($i = 0; $i < count($nam); $i++){

$n=$i+2;

$objPHPExcel->setActiveSheetIndex(0)->setCellValue("A$n", $nam[$i]->nameb);

$objPHPExcel->setActiveSheetIndex(0)->setCellValue("B$n", $nam[$i]->price);

} // Заполнение таблицы в Excel

$objPHPExcel->setActiveSheetIndex(0);

header('Content-Type: application/vnd.openxmlformats-officedocument.spreadsheetml.sheet');

header('Content-Disposition: attachment;filename="stat.xlsx"');

header('Cache-Control: max-age=0');

header('Cache-Control: max-age=1');

header ('Expires: Mon, 26 Jul 1997 05:00:00 GMT'); // Date in the past

header ('Last-Modified: '.gmdate('D, d M Y H:i:s').' GMT'); // always modified

header ('Cache-Control: cache, must-revalidate'); // HTTP/1.1

header ('Pragma: public'); // HTTP/1.0

$objWriter = PHPExcel\_IOFactory::createWriter($objPHPExcel, 'Excel2007');

$objWriter->save('php://output');

?>

Сайт готов.)