**Практическая работа № 4.**

**Регистрация на PHP**

Давайте теперь реализуем регистрацию. Для этого сделаем отдельную страницу register.php. По заходу на эту страницу, пользователь должен увидеть форму, в которую он должен вбить желаемый логин и пароль:

<form action="" method="POST">

<input name="login">

<input name="password" type="password">

<input type="submit">

</form>

По нажатию на кнопку отправки форму логин и пароль пользователя должны занестись в базу данных с помощью INSERT запроса, вот так:

<?php

if (!empty(**$\_POST**['login']) and !empty(**$\_POST**['password'])) {

$login = **$\_POST**['login'];

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

$query = "INSERT INTO users SET login='**$login**', password='**$password**'";

mysqli\_query($link, $query);

}

?>

Затем пользователь может зайти на страницу авторизации, ввести логин и пароль, под которым он зарегистрировался и авторизоваться.

№1

Реализуйте описанную выше регистрацию. После этого зарегистрируйте нового пользователя и авторизуйтесь под ним. Убедитесь, что все работает, как надо.

№2

Модифицируйте ваш код так, чтобы кроме логина и пароля пользователю нужно было ввести еще и дату своего рождения и email. Сохраните эти данные в базу данных.

№3

Модифицируйте ваш код так, чтобы в базу автоматически сохранялась дата регистрации.

**Авторизуем сразу при регистрации**

Сейчас наша регистрация сделана таким образом, что пользователь первый раз вбивает логин-пароль регистрируясь, а потом сразу же вбивает их второй раз, желая зайти на сайт.

Это на самом деле не очень удобно и будет раздражать пользователей. Лучше сделать так, чтобы при успешной регистрации сразу же происходила автоматическая авторизация. Для этого сразу после успешной регистрации запишем в сессию пометку об успешной авторизации:

<?php

if (!empty(**$\_POST**['login']) and !empty(**$\_POST**['password'])) {

$login = **$\_POST**['login'];

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

$query = "INSERT INTO users SET login='**$login**', password='**$password**'";

mysqli\_query($link, $query);

**$\_SESSION**['auth'] = true; // пометка об авторизации

}

?>

№4

Модифицируйте ваш код так, чтобы после регистрации пользователь автоматически становился авторизованным.

**Добавляем id пользователя в сессию**

Пусть кроме пометки об авторизации мы бы хотели записать в сессию еще и его id.

В этом случае мы можем получить его с помощью функции mysqli\_insert\_id. Эта функция получает id последней вставленной в данном скрипте записи, то есть как раз то, что нам нужно.

Реализуем:

<?php

if (!empty(**$\_POST**['login']) and !empty(**$\_POST**['password'])) {

$login = **$\_POST**['login'];

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

$query = "INSERT INTO users SET login='**$login**', password='**$password**'";

mysqli\_query($link, $query);

**$\_SESSION**['auth'] = true;

$id = mysqli\_insert\_id($link);

**$\_SESSION**['id'] = $id; // пишем id в сессию

}

?>

№5

Запишите при регистрации в сессию еще и id пользователя.

**Скрываем пароль**

Поле ввода пароля обычно представляет собой инпут с типом password, в котором введенные символы скрываются под звездочками. Это сделано для того, чтобы злоумышленник не мог подсмотреть пароль пользователя через плечо в момент регистрации.

Сокрытие пароля таким образом конечно хорошо, но есть, однако, проблема - пользователь не видит, что вводит. Он может ошибиться при вводе какого-либо символа и зарегистрироваться не с тем паролем, с которым он хотел. Это будет печально:(, так как он затем не сможет авторизоваться на сайте.

Существует стандартное решение этой проблемы: пользователю показываются два инпута для ввода пароля - в первый инпут он вводит пароль, а во второй инпут - его подтверждение, то есть этот же пароль второй раз:

<form action="" method="POST">

<input name="login">

<input type="password" name="password">

<input type="password" name="confirm">

<input type="submit">

</form>

Задача нашего сайта - проверить, что пароль и его подтверждение совпадают, так как логично, что в этом случае пользователь ввел именно то, что хотел ввести:

<?php

if (!empty(**$\_POST**['login']) and !empty(**$\_POST**['password']) and !empty(**$\_POST**['confirm'])) {

if (**$\_POST**['password'] == **$\_POST**['confirm']) {

// регистрируем

} else {

// выведем сообщение о несовпадении

}

}

?>

№6

Модифицируйте ваш код так, чтобы при отправке формы пароль сравнивался с его подтверждением. Если они совпадают - то продолжаем регистрацию, а если не совпадают - то выводим сообщение об этом.

**Проверка логина на занятость**

Сейчас наша регистрация имеет одну проблему - новый пользователь нашего сайта может зарегистрироваться под уже существующим логином, что, конечно же, недопустимо.

Для решения проблемы необходимо перед запросом на добавление нового пользователя в базу данных, выполнить SELECT запрос, который проверит, занят желаемый логин или нет. Если не занят - регистрируем, если занят - не регистрируем, а выводим сообщение об этом.

Давайте напишем этот код:

<?php

if (!empty(**$\_POST**['login']) and !empty(**$\_POST**['password'])) {

$login = **$\_POST**['login'];

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

$query = "SELECT \* FROM users WHERE login='**$login**'";

$user = mysqli\_fetch\_assoc(mysqli\_query($link, $query));

if (empty($user)) {

$query = "INSERT INTO users SET login='**$login**', password='**$password**'";

mysqli\_query($link, $query);

**$\_SESSION**['auth'] = true;

} else {

// логин занят, выведем сообщение об этом

}

}

?>

№7

Модифицируйте ваш код так, чтобы при попытке регистрации выполнялась проверка на занятость логина и, если он занят, - выводите сообщение об этом и просите ввести другой логин.