Алгоритмизация и программирование

Лекция 5.3

Авторизация и аутентификация

На данном этапе созданный нами и описанный в лекции 5.2 бэкенд сайта доступен всем, и любой может создавать, редактировать и удалять разделы сайта.

Нам необходимо защитить эту страницу – наш файл cp.php. Сделать так, чтобы в него могли заходить только авторизованные администраторы.

Для этого нам надо создать сервис, позволяющий регистрировать администраторов и сервис, позволяющий проверять валидность администраторов, путём проверки логина и пароля администратора.

Данные администраторов в нашем проекте будут храниться в базе данных backend в таблице admins.

В нашем проекте на данном этапе проектирования предполагается, что сайтом будет управлять один администратор со статусом supervisor.

У такого администратора полные права на управление сайтом и только он может принимать решение создавать ему помощников – других администраторов или нет.

При такой постановке задачи нам потребуется создание одной записи в таблице admins, где будут указаны логин, пароль и статус администратора - супервайзера.

Создание таблицы в базе данных

Начнём, как всегда, с создания таблицы в базе данных.

127.0.0.1 » backend » admins

admins

Поле	Тип	Null	По умолчанию
id	int(10)	Нет	
login	varchar(255)	Нет	
password	varchar(255)	Нет	
status	tinyint(1)	Нет	1

Индексы

Имя индекса	Тип	Уникальный	Упакован	Поле	Уникальных элементов	Сравнение	Null	Комментарий
PRIMARY	BTREE	Да	Нет	id	0	A	Нет	

Здесь в поле status по умолчанию будет храниться единица, которая обозначает статус «супервайзер».

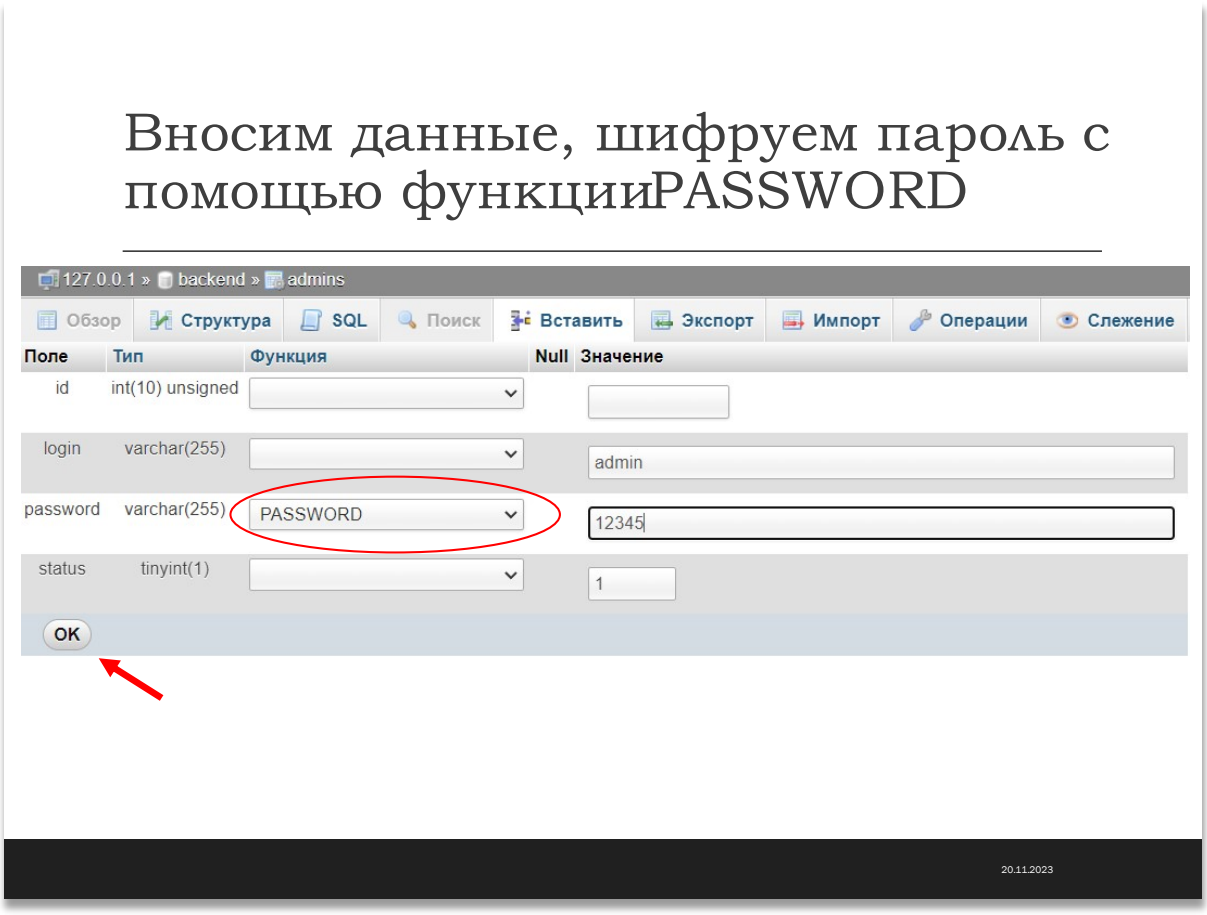
Пароль в базе данных должен храниться в зашифрованном виде. Это необходимо для защиты паролей от посторонних лиц.

В нашем примере для шифрования пароля применим встроенную функцию базы данных MySQL – password(), параметром которой укажем наш секретный пароль. Функция password() зашифрует пароль необратимым шифром.

Внесём в таблицу admins с помощью сервиса phpMyAdmin одну единственную запись:

login = admin;

password=password(‘12345’).



phpMyAdmin сформировал запрос и исполнил его:

```
INSERT INTO `backend`.`admins` (`id`, `login`, `password`, `status`)
VALUES (NULL, 'admin', PASSWORD('12345'), '1');
```

В результате у нас появилась в таблице запись:

Результат SQL-запроса

Хост: 127.0.0.1
База данных: backend
Время создания: Ноя 21 2023 г., 01:01
Создан: phpMyAdmin 3.5.1 / MySQL 5.5.25
SQL-запрос: SELECT * FROM `admins` LIMIT 0, 30 ;
Строки: 1

id	login	password	status
1	admin	*00A51F3F48415C7D4E8908980D443C29C69B60C9	1

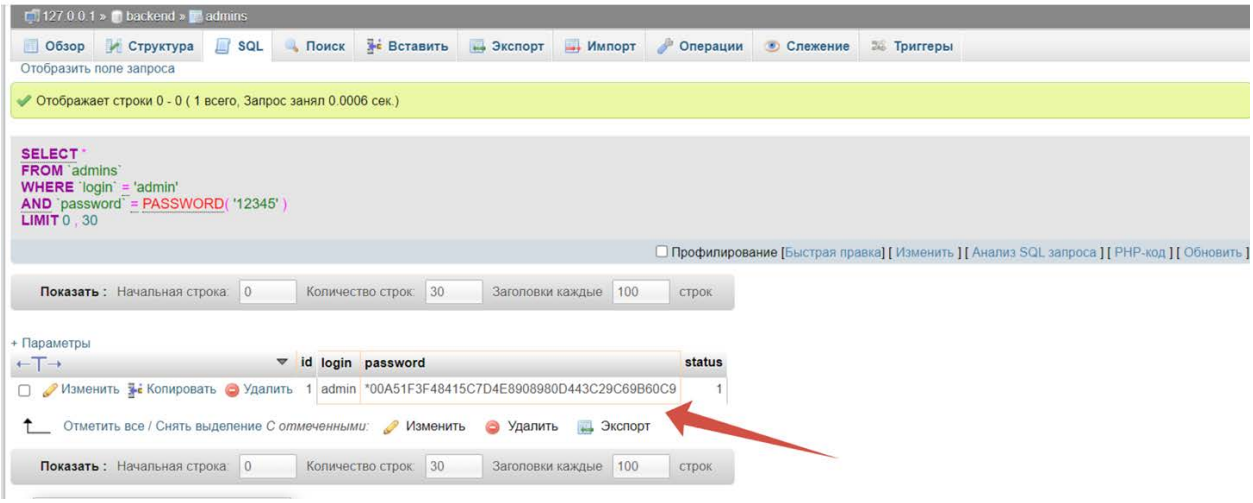
Как видите пароль зашифровался длинной строкой, расшифровать которую практически невозможно.

Для проверки такого пароля потребуется запрос, который будет сравнивать зашифрованный той же функцией password () введённый пароль с записью, которая хранится в поле password.

Проверим это задав следующий запрос в окне SQL запросов phpMyAdmin:

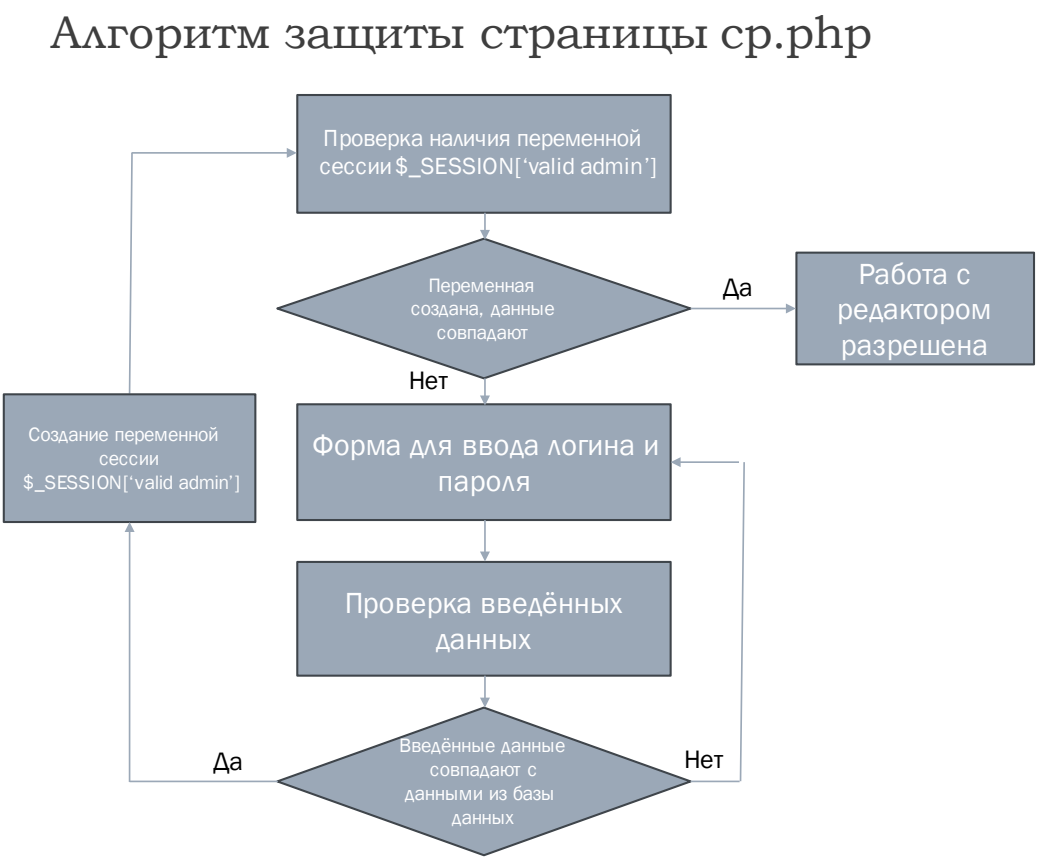
```
SELECT * FROM `admins` WHERE
`login`='admin'
and`password`=PASSWORD('12345')
```

Результат удался, phpMyAdmin нашел нашу запись с нашим логином и паролем!



Запомним и сохраним этот запрос. Будем в дальнейшем использовать его в нашем сценарии.

Алгоритм защиты страницы cp.php



При входе на страницу cp.php сценарий проверки проверяет наличие глобальной переменной сессии $_SESSION[‘valid admin’]. Эта переменная – массив, в которой хранится логин и пароль:

$_SESSION[‘valid admin’][‘login’]

$_SESSION[‘valid admin’][‘password’]

Если такая переменная существует и хранящиеся в ней данные совпадают с теми. Что хранятся в базе данных, то сценарий допускает проверенного администратора к работе с редактором сайта.

В противном случае, сценарий предлагает пройти авторизацию и показывает форму, в которую нужно ввести правильный логин и пароль.

Данные из формы проверяются на соответствие с теми, что хранятся в базе данных и если это так, то сценарий формирует переменную сессии $_SESSION[‘valid admin’] и отправляет её на проверку к началу алгоритма.

Реализуем задуманный алгоритм

Создаём файл с функциями

Для управления аутентификацией (валидацией) администратора создадим несколько функций и разместим их в отдельном файле fns.php.

Функция проверки валидности админа check_admin()

```
////Функция проверки валидности админа/////
function check_admin()
{
	if(isset($_SESSION['valid_admin']))
		{
			$l=$_SESSION['valid_admin']['login'];
			$pa=$_SESSION['valid_admin']['password'];
			$r=mysql_query("select * from admins where
login='$l'and password=password('$pa')");
			if($r)
			{
				if(mysql_num_rows($r)==0)
				{

					return false;
				}
				else
				{
					return true;
				}
			}

		}
}
```

Функция check_admin() проверяет наличие переменной сессии $_SESSION['valid_admin']. В этой переменной – глобальном ассоциативном массиве по нашему замыслу хранятся логин и пароль авторизованного администратора. Функция извлекает из переменной сессии логин и пароль и формирует запрос к базе данных на проверку наличия логина и пароля в таблице admins. В случае отсутствия в базе данных совпадающих записей, функция возвращает false, в случае наличия – возвращает true.

Функция проверки правильности ввода логина и пароля check_valid_form

```
//////функция проверки правильности ввода логина и пароля/////////
function check_valid_form($login,$password)
{
	$q="select * from admins where login='$login'and
password=password('$password')";
	$r=mysql_query($q);
			if($r)
			{
				if(mysql_num_rows($r)==0)
				{

					return false;
				}
				else
				{
					session_start();
					$row = mysql_fetch_assoc($r);
```

```
     $_SESSION['valid_admin']['login']=$row['login'];

     $_SESSION['valid_admin']['password']=$password;
                          return true;
                    }
               }
}
```

Функция check_valid_form принимает два параметра – логин и пароль, формирует запрос к базе данных на проверку наличия логина и пароля в таблице admins. В случае отсутствия в базе данных совпадающих записей, функция возвращает false, в случае наличия – запускает сессию функцией session_start(), формирует переменную сессии $_SESSION['valid_admin'] и записывает в нее логин и пароль. Возвращает true.

Функция отображения формы авторизации display_auth_form

```
function display_auth_form($action)//быстрое формирование формы
авторизации
     {

               $html='<div align="center">
               <form action="'.$action.'" method="post"  id="auth"
name="auth">
            <br>Логин<br>
            <input name="login" type="text" class="auth">
            <br>
            Пароль<br>
            <input name="password" type="password" class="auth">
            <br>
            <input name="submit" type="submit" value="Вход">
            <br>';

           $html.= '</form>
              </div>';
                 return $html;
          }
```

Функция display_auth_form принимает в качестве параметра ссылку на сценарий-обработчик формы, генерирует html код формы авторизации и записывает его в переменную $html, которую и возвращает.

Создаём сценарий авторизации (файл auth.php)

```
<?php
session_start();
include'db_include.php';
include 'fns.php';

$p=$_GET['p'];
$form='';
///обработчик формы/////
if($p==1)
{
```

```
      $login=$_POST['login'];
      $password=$_POST['password'];
      $check=check_valid_form($login,$password);
      if($check)
      {
            header('location:cp.php');
      }
      else
      {
            $form.='<h3 align="center">Вы не авторизованы, попробуйте
еще раз</h3>';
      }
}
$form.=display_auth_form('?p=1');
?>
<!DOCTYPE html>
<html >
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1251" />
<title>Авторизация</title>
</head>

<body>
      <?=$form;?>
</body>
</html>
```

Исходя из алгоритма работы, данный сценарий запускается в следующих случаях:

1. Отсутствия переменной сессии;
2. Несовпадении логина и пароля, записанных в переменной сессии с записями в базе данных;
3. Несовпадения введённых в форму авторизации логина и пароля с записями в базе данных.

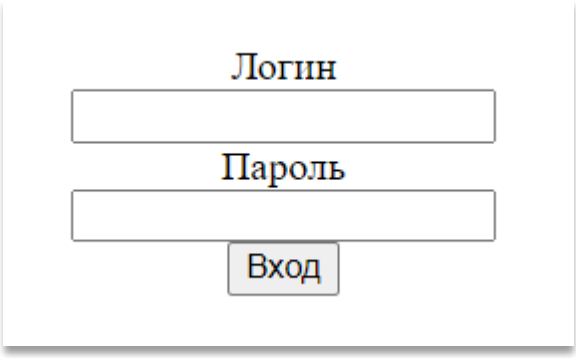
При первом обращении к сенарию, который обуславливается первым и вторым случаем, формируется html код формы авторизации, вызываемый пользовательской функцией display_auth_form, которой мы, в качестве параметра передаём ссылку на обработчик формы:

```
$form.=display_auth_form('?p=1');
```

Эта переменная вставлена в html шаблон, который размещён под сценарием php:

```
<body>
      <?=$form;?>
</body>
```

Форма выглядит примерно так:



Отправка формы вызывает сценарий обработчика формы:

```
///обработчик формы/////
if($p==1)
{
	$login=$_POST['login'];
	$password=$_POST['password'];
	$check=check_valid_form($login,$password);
	if($check)
	{
		header('location:cp.php');
	}
	else
	{
		$form.='<h3 align="center">Вы не авторизованы, попробуйте
еще раз</h3>';
	}
}
```

Сценарий принимает данные из глобального массива $_POST и вызывает пользовательскую функцию check_valid_form, передавая ей в качестве параметров логин и пароль, полученные из формы.

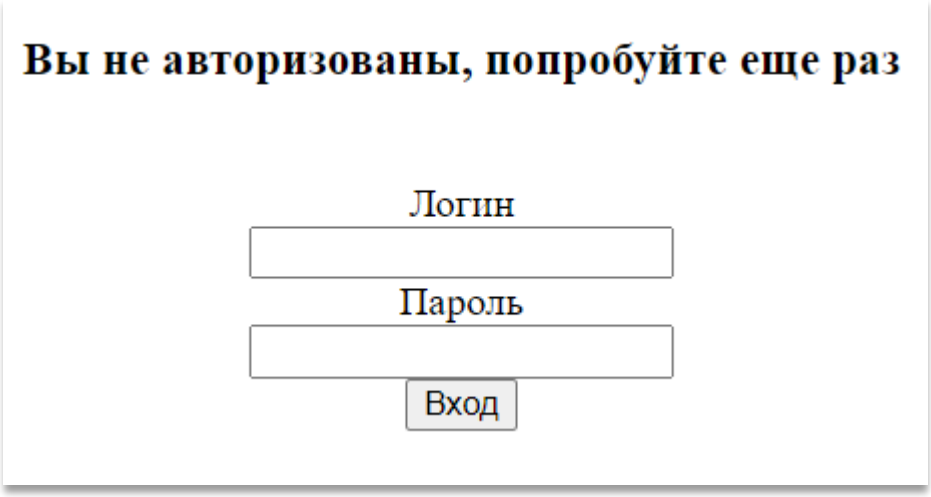
```
$check=check_valid_form($login,$password);
```

Если переменная $check содержит истину, значит введённые логин и пароль совпали с записью в базе данных и функция check_valid_form создала переменную сессии $_SESSION['valid_admin']. В этом случае заголовок header('location:cp.php') вызывает файл cp.php для нормальной работы по его назначению.

Если $check содержит ложь, значит в форму были введены некорректные значения логина и пароля, и обработчик добавляет в переменную $form сообщение о несостоявшейся авторизации:

```
$form.='<h3 align="center">Вы не авторизованы, попробуйте еще
раз</h3>';
```

После чего вновь вызывается функция display_auth_form и html шаблон отображает сообщение и форму авторизации.



Соединяем всё вместе

Дорабатываем файл cp.php

Добавляем функцию старта сессии и присоединяем файл fns.php в начало сценария:

```
<?
      session_start();
      include'db_include.php';
      include 'fns.php';
      /////Приём переменных//////////
```

Добавляем сценарий проверки валидности администратора, размещаем его над сценарием удаления раздела.

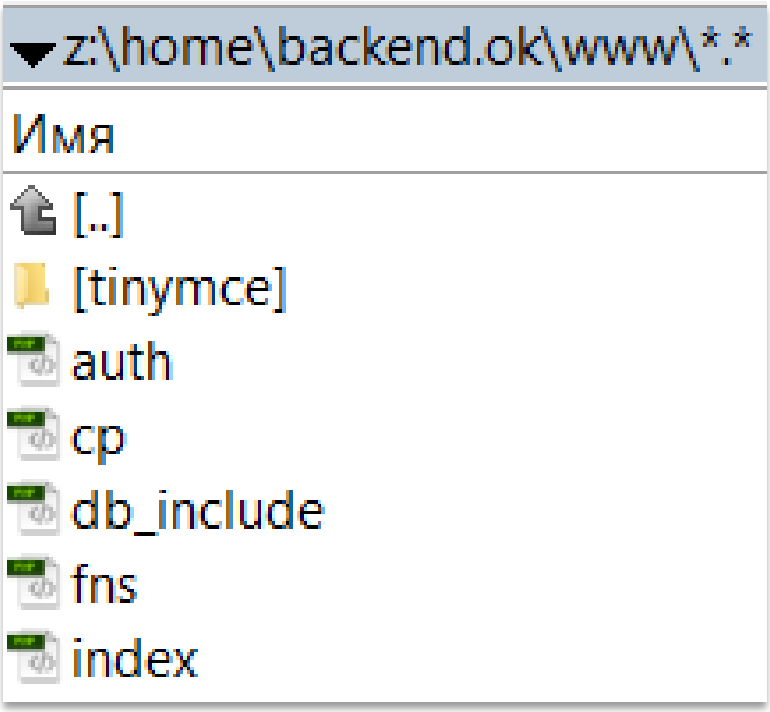
```
/////проверка валидности админа///////
      $valid_admin=check_admin();
      if(!$valid_admin) {
      header('location: auth.php');
      exit;}

      /////Удаление раздела/////////
```

Сценарий проверки валидности админа вызывает пользовательскую функцию check_admin(), которая возвращает в переменную $valid_admin ложь или истину. Если $valid_admin содержит ложь, заголовок header ('location: auth.php') вызывает файл auth.php и сценарий прекращает выполнение дальнейших сценариев с помощью оператора exit.

Если $valid_admin содержит истину, то ничего не происходит и сценарии файла cp.php работают в обычном режиме по их назначению.

Теперь файловая структура нашего проекта выглядит вот так:


z:\home\backend.ok\www*.*
Имя
[..]
[tinymce]
auth
cp
db_include
fns
index