

Лабораторная работа №3

Разработка административного модуля

Разработка административной части может быть осуществлена двумя способами:

- Создание нового контроллера с необходимыми действиями
- Использование модулей

Предпочтительнее использование модулей - отдельной части приложения (приложение в приложении).

Разработка модуля может быть автоматизирована за счет генератора кода Gii.

Создание модуля в Gii

Gii представлен в Yii как модуль. Вы можете активировать Gii, настроив его в свойстве modules. В зависимости от того, каким образом вы создали приложение, вы можете удостовериться в наличии следующего кода в конфигурационном файле config/web.php,

```
$config = [ ... ];  
if (YII_ENV_DEV) {  
    $config['bootstrap'][] = 'gii';  
    $config['modules']['gii'] = [  
        'class' => 'yii\gii\Module',  
    ];  
}
```

Приведенная выше конфигурация показывает, что находясь в режиме разработки, приложение должно включать в себя модуль с именем gii, который реализует класс yii\gii\Module.

Если вы посмотрите входной скрипт web/index.php вашего приложения, вы увидите следующую строку, устанавливающую константу YII_ENV_DEV в значение true.

```
defined('YII_ENV') or define('YII_ENV', 'dev');
```

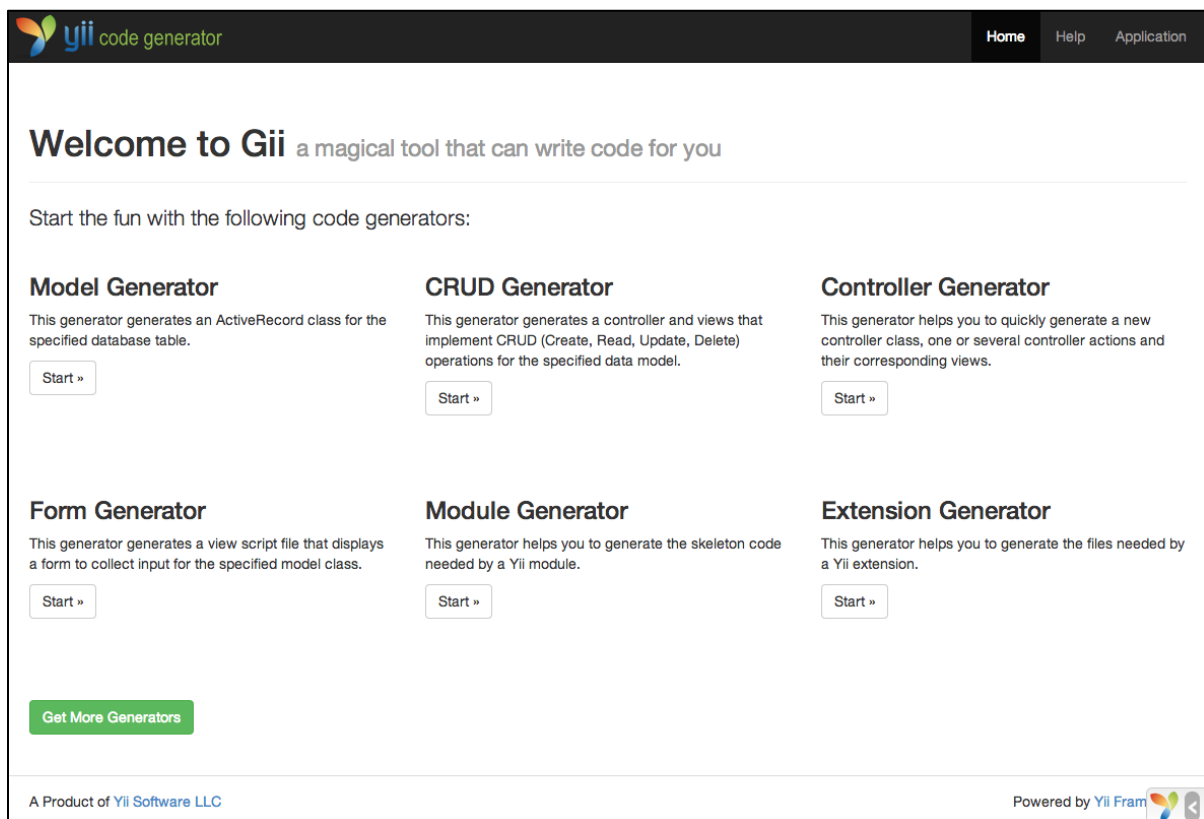
Благодаря этой строке ваше приложение находится в режиме разработки, и Gii уже активирован в соответствии с описанной выше конфигурацией. Теперь вы можете получить доступ к Gii по следующему адресу:

<http://hostname/index.php?r=gii>

Примечание: Если вы пытаетесь получить доступ к Gii не с локального хоста, по умолчанию, в целях обеспечения безопасности, доступ будет запрещён. Вы можете изменить настройки Gii, чтобы добавить разрешённые IP адреса, как указано ниже

```
'gii' => [  
    'class' => 'yii\gii\Module',  
    'allowedIPs' => ['127.0.0.1', '::1', '192.168.0.*', '192.168.178.20']  
],
```

Выполнить запуск Gii можно обратившись по ссылке:
<http://yii2/web/index.php?r=gii>



Перейдите в раздел Module Generator для создания нового модуля.

Для создания нового модуля укажите имя создаваемого класса с учетом пространства имен и его идентификатор.

Module Generator

This generator helps you to generate the skeleton code needed by a Yii module.

Module Class

Module ID

Code Template

default (D:\OSPanel\domains\yii2\vendor\yiisoft\yii2-gii\generators\module\default)

Preview

В результате будет созданы следующие файлы:

| Code File | Action | <input checked="" type="checkbox"/> |
|---|--------|-------------------------------------|
| modules\admin\Module.php | create | <input checked="" type="checkbox"/> |
| modules\admin\controllers\DefaultController.php | create | <input checked="" type="checkbox"/> |
| modules\admin\views\default\index.php | create | <input checked="" type="checkbox"/> |

Для их генерации необходимо нажать кнопку Generate.

Для подключения модуля необходимо подключить его в файле конфигурации.

The module has been generated successfully.
To access the module, you need to add this to your application configuration:

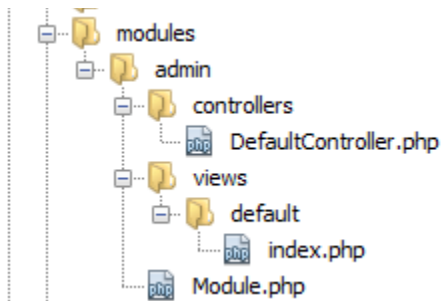
```
<?php
.....
'modules' => [
    'admin' => [
        'class' => 'app\modules\admin\Module',
    ],
],
.....
```

Для этого сгенерированный код необходимо скопировать и перенести в файл config/web.php

```
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    //'layout' => 'basic',
    'language' => 'ru',
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm'    => '@vendor/npm-asset',
    ],

    'modules' => [
        'admin' => [
            'class' => 'app\modules\admin\Module',
        ],
    ],
],
```

В данной папке дублируется структура приложения Yii, т.е. уже есть папка controllers с контроллерами, папка views с представлениями. Также мы можем создать папку models, в которой будут храниться модели модуля.



Добавьте в файле представления index.php заголовок «Административная часть».

Для доступа к модулю необходимо обратиться по следующему пути (пример)
<http://yii2/web/index.php?r=admin>

Аутентификация и авторизация

В базовом приложении содержится класс User (папка моделей), реализующий интерфейс для авторизации пользователя и определяющий необходимые методы. Реализация интерфейса `yii\web\IdentityInterface`, включает следующие методы:

- **findIdentity():** Этот метод находит экземпляр identity class, используя ID пользователя. Этот метод используется, когда необходимо поддерживать состояние аутентификации через сессии.

- findIdentityByAccessToken(): Этот метод находит экземпляр identity class, используя токен доступа. Метод используется, когда требуется аутентифицировать пользователя только по секретному токenu (например в RESTful приложениях, не сохраняющих состояние между запросами).
- getId(): Этот метод возвращает ID пользователя, представленного данным экземпляром identity.
- getAuthKey(): Этот метод возвращает ключ, используемый для основанной на cookie аутентификации. Ключ сохраняется в аутентификационной cookie и позже сравнивается с версией, находящейся на сервере, чтобы удостовериться, что аутентификационная cookie верная.
- validateAuthKey(): Этот метод реализует логику проверки ключа для основанной на cookie аутентификации.

Данный класс может быть взят за шаблон-образец. В данном классе авторизация не предусматривает взаимодействие с базой данной, лишь определяя статическое свойство \$users.

```
namespace app\models;


class User extends \yii\base\BaseObject implements \yii\web\IdentityInterface
{
    public $id;
    public $username;
    public $password;
    public $authKey;
    public $accessToken;

    private static $users = [
        '100' => [
            'id' => '100',
            'username' => 'admin',
            'password' => 'admin',
            'authKey' => 'test100key',
            'accessToken' => '100-token',
        ],
        '101' => [
            'id' => '101',
            'username' => 'demo',
            'password' => 'demo',
            'authKey' => 'test101key',
            'accessToken' => '101-token',
        ],
    ];
}
```

В файле web.php определяется компонент user, где указывается необходимый класс авторизации пользователей. Свойство enableAutoLogin отвечает за автоматическую авторизацию пользователя на основе его cookies.

```
'components' => [
    'request' => [
        // !!! insert a secret key in the following (
        'cookieValidationKey' => 'dsgfsg3414feaefq',
    ],
    'cache' => [
        'class' => 'yii\caching\FileCache',
    ],
    'user' => [
        'identityClass' => 'app\models\User',
        'enableAutoLogin' => true,
    ],
],
```

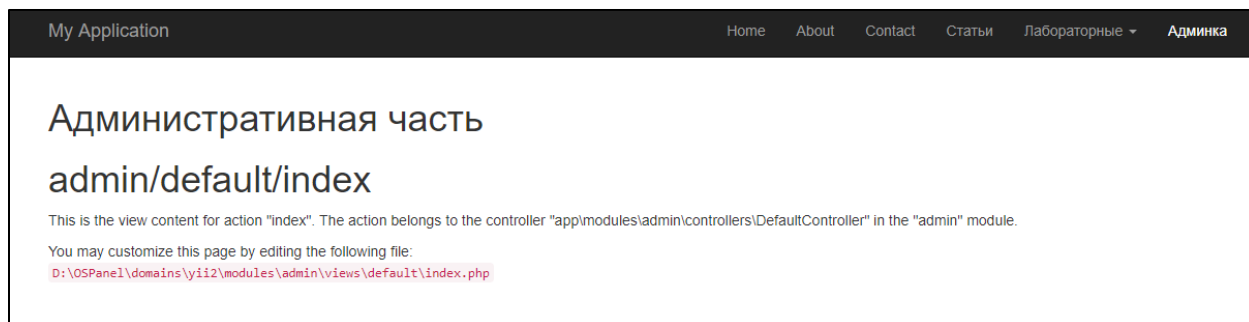
Для авторизации пользователей на основе информации, хранящейся в базе данных, необходимо наличие соответствующей таблицы, которую следует создать дополнительно. Поле auth_key будет использоваться в дальнейшем для авторизации на основе данных куки.

| # | Имя | Тип данных | Длина/Знач... | Беззна... | Разреш... | Zerofill | По умолчанию |
|---|----------|------------|---------------|-------------------------------------|-------------------------------------|-------------------------------------|---------------------|
|  1 | id | INT | 10 | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | AUTO_INCREMENT |
| 2 | username | VARCHAR | 255 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | Нет значения по ... |
| 3 | password | VARCHAR | 255 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | Нет значения по ... |
| 4 | auth_key | VARCHAR | 255 | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | Нет значения по ... |

В Yii2 предусмотрены две «роли» - авторизованного пользователя и просто гостя, для которых созданы соответствующие алиасы (@ и ? соответственно).

Разграничим права доступа: гость может работать со всеми страницами, а доступ в административную часть будет иметь только авторизованный пользователь – администратор.

Для начала следует изменить путь в файле main.php, прописав путь к модулю admin.



Следующее, что необходимо сделать – ограничить доступ к созданному административному модулю по логину и паролю.

Создайте новый контроллер AppAdminController в модуле admin. Данный класс будет наследовать базовый класс Controller и в нем будут прописаны все настройки.

Измените класс DefaultController. Теперь он наследует класс AppAdminController.

Авторизация доступна за счет фильтра контроля доступа yii\filters\AccessControl (ACF). ACF проверяет набор правил доступа – behaviors.

Внесите изменения в контроллер AppAdminController.

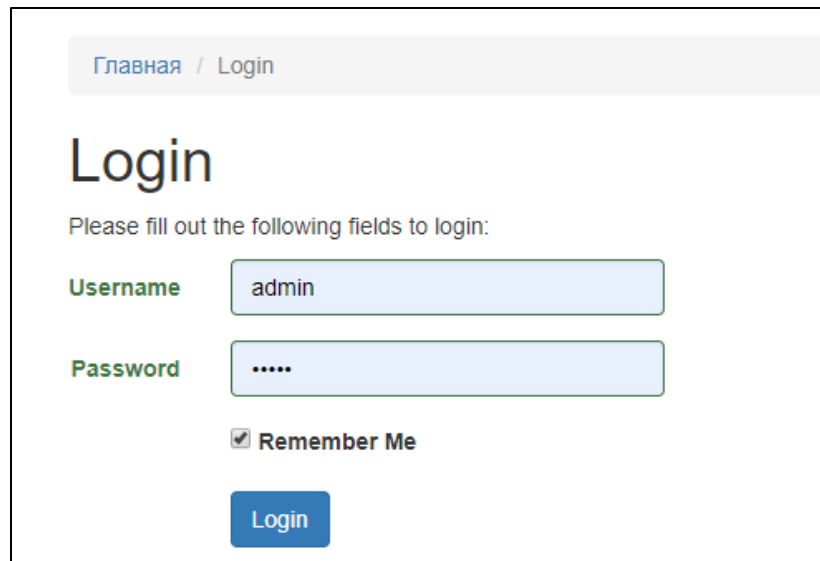
Теперь доступ к административному разделу требует авторизации. Происходит обращение к контроллеру SiteController и действию actionLogin.

```
<?php
/*...5 строк */

namespace app\modules\admin\controllers;
use yii\web\Controller;
use yii\filters\AccessControl;

/** Description of AppAdminController ...5 строк */
class AppAdminController extends Controller{

    public function behaviors() {
        return
        [
            'access' => [
                'class' => AccessControl::className(),
                'rules' => [
                    [
                        'allow' => true,
                        'roles' => ['@']
                    ]
                ]
            ]
        ];
    }
}
```



В `actionLogin` происходит ряд проверок. Если пользователь является просто гостем – его необходимо перенаправить на главную страницу. Далее создается экземпляр модели `LoginForm`, в которую загружаем данные и вызываем метод `login`, который позволяет авторизовать пользователя. При этом, если данные загружены и метод `login` вернул истину, то выполняет редирект на предыдущую страницу. Иначе будет сгенерировано представление `login`, в которое передается модель.

```
public function actionLogin()
{
    if (!Yii::$app->user->isGuest) {
        return $this->goHome();
    }

    $model = new LoginForm();
    if ($model->load(Yii::$app->request->post()) && $model->login()) {
        return $this->goBack();
    }

    $model->password = '';
    return $this->render('login', [
        'model' => $model,
    ]);
}
```

Рассмотрим `LoginForm`, в которой есть метод `login`.


```

public function login()
{
    if ($this->validate()) {
        return Yii::$app->user->login($this->getUser(), $this->rememberMe ? 3600*24*30 : 0);
    }
    return false;
}

```

В данном методе происходит валидация данных в соответствии с описанными в модели правилами. Одно из правил validatePassword:

```

public function validatePassword($attribute, $params)
{
    if (!$this->hasErrors()) {
        $user = $this->getUser();

        if (!$user || !$user->validatePassword($this->password)) {
            $this->addError($attribute, 'Incorrect username or password.');
        }
    }
}

```

Данный метод проверяет наличие ошибок (введены ли имя пользователя, пароль). Если ошибок не было – создается объект user путем вызова метода getUser. В противном случае выводится сообщение об ошибке.

Метод getUser проверяет, не авторизован ли пользователь. Если пользователь не авторизован (свойство user принимает значение false по умолчанию), тогда пользователя необходимо найти и вернуть. Для его поиска вызывается статический метод findByUsername, в которое передается введенное в форму имя пользователя username класса User.

```

public function getUser()
{
    if ($this->_user === false) {
        $this->_user = User::findByUsername($this->username);
    }

    return $this->_user;
}

```

Поиск осуществляется среди элементов массива users модели User.

Если пользователь найден – возвращается его имя, иначе – null.

В случае успеха операций происходит авторизация пользователя. Также проверяется, необходимо ли сохранять данные пользователя в куках `$this->rememberMe` ? $3600*24*30 : 0$) на 30 дней.

Внесем изменения, позволяющие осуществлять поиск пользователя в базе данных.

Для того, чтобы скрыть эту страницу авторизации от неавторизованного пользователя, можно при попытке обращения к Админке, перенаправить его по другому адресу, например, на главную страницу.

Для этого в файле `web.php` необходимо задать `loginUrl`:

```
'user' => [  
    'identityClass' => 'app\models\User',  
    'enableAutoLogin' => true,  
    'loginUrl' => 'index.php'  
],
```

Далее эту строку можно закомментировать.

Для того, чтобы отредактировать вид формы входа в админку, необходимо внести изменения в модель `LoginForm` и вид `login.php`

Например, так.

Далее зайдите в файл модели User.php. Стандартные настройки предполагают авторизацию двух пользователей, заданных статично. Для работы с базой данных, этих пользователей (admin, demo) можно удалить.

Также класс теперь будет наследовать базовый класс ActiveRecord, поэтому свойства класса также будут изменены.

Теперь код выглядит так:

```
<?php
```

```
namespace app\models;
use yii\db\ActiveRecord;
```

```
class User extends ActiveRecord implements \yii\web\IdentityInterface
{
    /**
     * {@inheritdoc}
     */
    public static function findIdentity($id)
    {
        return isset(self::$users[$id]) ? new static(self::$users[$id]) : null;
    }
}
```

Следуя правилам хорошего тона, переопределим метод, позволяющий связать класс с таблицей базы данных в явном виде:

```
public static function tableName() {
    return 'user';
}
```

Метод findIdentity будет возвращать найденного пользователя по его id.

```

public static function findIdentity($id)
{
    return static::findOne($id);
}

```

Метод findIdentityByAccessToken оставляем пустым. Его реализация сейчас не нужна.

```

public static function findIdentityByAccessToken($token, $type = null)
{
}

```

Метод findByUsername выполняет поиск пользователя по логину, введенному в поле формы.

```

public static function findByUsername($username)
{
    return static::findOne(['username' => $username]);
}

```

Метод getId позволяет получить id, оставляем его без изменений.

Метод getAuthKey и метод validateAuthKey меняем в соответствии с названием поля в базе данных.

```

public function getAuthKey()
{
    return $this->auth_key;
}

/**
 * {@inheritdoc}
 */
public function validateAuthKey($authKey)
{
    return $this->auth_key === $authKey;
}

```

Метод validatePassword будет сравнивать пароль, который хранится в БД с тем паролем, который ввел пользователь. Поскольку в явном виде пароль хранить небезопасно, следует его хешировать.

Для получения хэша пароля (для внесения в базу данных уже в хэшированном виде), напишите команду:

```
$hash = Yii::$app->getSecurity()->generatePasswordHash('12345');
echo $hash;
```

Полученный хэш скопируйте в таблицу user, поле пароль.

Таким образом, в базе данных хранится информация об администраторе.

| id | username | password | auth_key |
|----|----------|--|----------|
| 1 | admin | \$2y\$13\$JTHuLM3GcoAOwfccXh41zeK1aZhY3VaKu4TOGyGIETB... | |

При правильном вводе пары логин-пароль, будет осуществлен доступ в административную часть.

Административная часть

admin/default/index

This is the view content for action "index". The action belongs to the controller "app\modules\admin\controllers\DefaultController" in the "admin" module.

You may customize this page by editing the following file:

D:\OSPanel\domains\yii2\modules\admin\views\default\index.php

Однако, поле auth_key, отвечающее для сохранение данных в куках, остается пустым, если посмотреть состояние таблицы БД. В этом случае, создадим метод generateAuthKey в файле User.php, который будет генерировать случайную строку, которая в дальнейшем будет сохранена в таблице БД. Данный метод будет вызываться каждый раз при авторизации, заполняя новым значением поле auth_key, в случае, если была выбрана опция сохранения данных.

Меняем код в файле LoginForm.

```
public function login()
{
    if ($this->validate()) {
        if ($this->rememberMe)
        {
            $myuser = $this->getUser();
            $myuser->generateAuthKey();
            $myuser->save();
        }
        return Yii::$app->user->login($this->getUser(), $this->rememberMe ? 3600*24*30 : 0);
    }
    return false;
}
```

Теперь, если пользователь выбрал флажок — Сохранить, будет создаваться объект, для него будет генерировать поле auth_key и далее поле в базе данных будет обновляться.

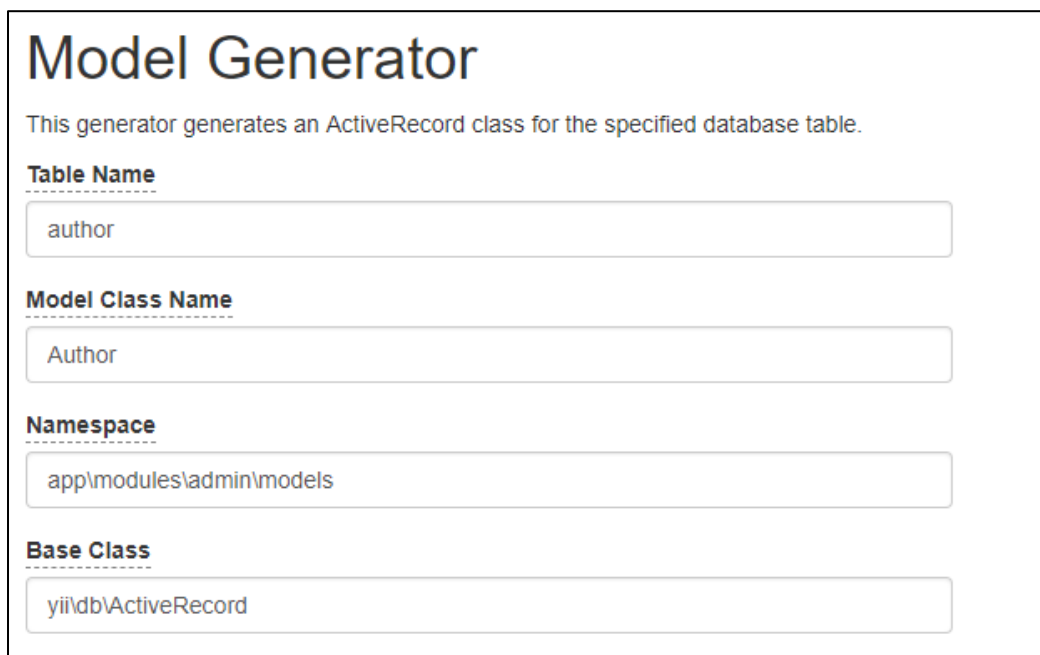
Реализация CRUD-функционала приложения на базе Gii

Создание класса Active Record на базе Gii

Чтобы использовать Gii для генерации класса Active Record, выберите "Генератор модели" (нажав на ссылку на главной странице Gii).

Заполните форму следующим образом:

- Имя таблицы – author
- Класс модели – author
- Пространство имен - app\modules\admin\models, поскольку модель должна быть создана в административном модуле.



Model Generator

This generator generates an ActiveRecord class for the specified database table.

Table Name
author

Model Class Name
Author

Namespace
app\modules\admin\models

Base Class
yii\db\ActiveRecord

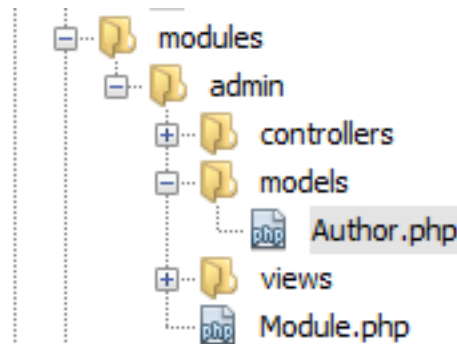
Затем нажмите на кнопку "Предварительный просмотр". Вы увидите, что models/Author.php перечислен в результатах создаваемых файлов классов. Вы можете нажать на имя файла класса для просмотра его содержимого.

Если вы уже создали такой же файл и хотите перезаписать его, нажмите кнопку diff рядом с именем файла, чтобы увидеть различия между генерируемым кодом и существующей версией.

Для перезаписи существующего файла установите флажок рядом с "overwrite" и нажмите кнопку "Generate". Для создания нового файла вы можете просто нажать "Generate".

После этого вы увидите страницу подтверждения, указывающую на то, что код был успешно сгенерирован. Если файл существовал до этого, вы также увидите сообщение о том, что он был перезаписан заново сгенерированным кодом.

Вернувшись в структуру проекта, можно увидеть, что была создана необходимая папка с классом модели.



Создание CRUD

CRUD расшифровывается как Create, Read, Update и Delete, предоставляющий четыре основные функции, выполняемые над данными на большинстве веб-сайтов.

Чтобы создать функциональность CRUD используя Gii, выберите "CRUD Генератор" (нажав на ссылку на главной странице Gii). Будут созданы необходимые контроллеры и представления.

Заполните форму следующим образом, указывая полные пути.

CRUD Generator

This generator generates a controller and views that implement CRUD (Create, Read, Update, Delete) operations for the specified data model.

Model Class

Search Model Class

Controller Class

View Path

Base Controller Class

Widget Used in Index Page

☐ Enable I18N

☐ Enable Pjax

Code Template



















[Preview](#)

В результате будут созданы следующие файлы:

| Code File | Action | |
|--|--------|---|
| modules\admin\controllers\AuthorController.php | create | ✓ |
| modules\admin\models\AuthorSearch.php | create | ✓ |
| modules\admin\views\author_form.php | create | ✓ |
| modules\admin\views\author_search.php | create | ✓ |
| modules\admin\views\author\create.php | create | ✓ |
| modules\admin\views\author\index.php | create | ✓ |
| modules\admin\views\author\update.php | create | ✓ |
| modules\admin\views\author\view.php | create | ✓ |

Если все прописано верно, можно их сгенерировать.

Проверить работоспособность кода можно обратившись по адресу:
<http://yii2/web/index.php?r=admin/author>

| Главная / Authors | | | | |
|-------------------------------|----------------------|----------------------|----------------------|---|
| Authors | | | | |
| Create Author | | | | |
| Показаны записи 1-13 из 13. | | | | |
| # | Author ID | Name | Birth | |
| | <input type="text"/> | <input type="text"/> | <input type="text"/> | |
| 1 | 1 | Стивен Кинг | 1947 |    |
| 2 | 2 | Л.Н. Толстой | 1828 |    |
| 3 | 3 | Ф.М. Достоевский | 1821 |    |
| 4 | 4 | Б. Акунин | 1956 |    |
| 5 | 5 | А. Кристи | 1890 |    |
| 6 | 6 | Джоан Роулинг | 1965 |    |

Для того, чтобы ограничить доступ к этой странице для неавторизованных пользователей, переместите скрипт поведений в файл Module.php. Не забудьте импортировать класс `use yii\filters\AccessControl;`

```

public function behaviors() {
    return
    [
        'access' => [
            'class' => AccessControl::className(),
            'rules' => [
                [
                    'allow' => true,
                    'roles' => ['@']
                ]
            ]
        ]
    ]
};
}

```

Также класс Author должен быть классом-наследником AppAdminController.

Задания к лабораторной работе

Максимальный балл за лабораторную работу – 6 баллов (4 балла выставляются при выполнении заданий 1 и 2; задание 3 является необязательным, за него могут быть выставлены дополнительные 2 балла)

Итоговый балл может быть понижен в следующих случаях: 1) задания выполнены некорректно, не в полном объеме, содержат ошибки, реализованы недостаточно оптимально и т.п.; 2) студент не может обосновать и аргументированно ответить на вопросы по заданиям, не владеет теоретическим материалом и т.п.

Задание 1

Создайте административный модуль в соответствии с методическими указаниями, представленными в лабораторной работе:

- Доступ в административную часть доступен только авторизованным пользователям.
- Возможность доступа осуществляется на основании данных, представленных в таблице базы данных.
- Страница авторизации доступна из верхнего меню. Также на эту страницу должна вести ссылка из файла «Лабораторная работа №3».
- Данные сохраняются в куках.

Задание 2

Реализуйте CRUD-приложение для работы со всеми таблицами базы данных библиотеки (автор, жанр, книга).

- На главной странице сделайте навигацию (любой компонент Bootstrap) для перехода к работе с таблицами.
- Измените сгенерированный код, чтобы кнопки, подписи формы и заголовки таблицы были на русском языке.
- На странице просмотра информации о книге/авторе/жанре добавьте кнопку *Назад* к Главной странице Административного модуля.
- Добавьте пагинацию.

Задание 3 (дополнительное)

Добавьте регистрацию на сайте.

Контрольные вопросы

1. Как активировать Gii в Yii-приложении?
2. Как выполнить генерацию класса Active Record?
3. Какой метод авторизации используется в приложениях с простым контролем доступа?
4. Какие роли предусмотрены в Yii-приложении?
5. Как получить хэш пароля?