# Программирование на стороне сервера. Часть 2. Yii2.

Оглавление

[Программирование на стороне сервера. Часть 2. Yii2. 1](#_Toc3384391)

[Оглавление 1](#_Toc3384392)

[Инструкция к выполнению: 3](#_Toc3384393)

[Лабораторная работа №1. Запуск приложения. Работа с базами данных. 3](#_Toc3384394)

[Установка приложения 3](#_Toc3384395)

[Запуск приложения 3](#_Toc3384396)

[Функциональность 3](#_Toc3384397)

[Структура приложения Yii 3](#_Toc3384398)

[Жизненный цикл пользовательского запроса 5](#_Toc3384399)

[Говорим «Привет» 5](#_Toc3384400)

[Создание действия 5](#_Toc3384401)

[Создание представления 6](#_Toc3384402)

[Работа с базами данных 8](#_Toc3384403)

[Подготавливаем базу данных 8](#_Toc3384404)

[Настраиваем подключение к БД 9](#_Toc3384405)

[Создаём потомка Active Record 9](#_Toc3384406)

[Создаём Action 10](#_Toc3384407)

[Создаём View 11](#_Toc3384408)

[Испытываем в действии 12](#_Toc3384409)

[Лабораторная работа №2. Работа с формами. Генерация кода при помощи Gii. 13](#_Toc3384410)

[Работа с формами 13](#_Toc3384411)

[Создание модели 13](#_Toc3384412)

[Создание действия 14](#_Toc3384413)

[Создание представления 15](#_Toc3384414)

[Как работает вся эта «магия» 17](#_Toc3384415)

[Генерация кода при помощи Gii 18](#_Toc3384416)

[Запускаем Gii 18](#_Toc3384417)

[Генерация класса Active Record 19](#_Toc3384418)

[Создание CRUD кода 21](#_Toc3384419)

[Испытываем в действии 22](#_Toc3384420)

[Лабораторная работа №3. Аутентификация и авторизация. 25](#_Toc3384421)

[Аутентификация 25](#_Toc3384422)

[Настройка yii\web\User 25](#_Toc3384423)

[Реализация yii\web\IdentityInterface 25](#_Toc3384424)

[Использование yii\web\User 27](#_Toc3384425)

[Авторизация 28](#_Toc3384426)

[Фильтры контроля доступа 28](#_Toc3384427)

[Лабораторная работа №4. Итоговая работа по Yii2. 32](#_Toc3384428)

[Общие требования: 32](#_Toc3384429)

[Полезные ресурсы: 32](#_Toc3384430)

[Варианты для разработки ИС: 32](#_Toc3384431)

# Инструкция к выполнению:

К каждой работе предоставляется отчет, состоящий:

1. Титульный лист
2. Лист с целью и задачами лабораторной работы
3. Ход выполнения
4. Листинг

# Лабораторная работа №1. Запуск приложения. Работа с базами данных.

## Установка приложения

1. Скачайте базовую версию Yii по ссылке (https://www.yiiframework.com/download)
2. Разархивируйте в корневую директорию веб-сервера

## Запуск приложения

После установки Yii базовое приложение будет доступно либо по URL http://hostname/basic/web/index.php, либо по http://hostname/index.php, в зависимости от настроек Web сервера.

**Информация:**далее в данном руководстве предполагается, что Yii установлен в директорию basic/web, которая, в свою очередь, установлена как корневой каталог в настройках Web сервера. В результате, обратившись по URL http://hostname/index.php, Вы получите доступ к приложению, расположенному в basic/web.

### Функциональность

Установленный шаблон простого приложения состоит из четырех страниц:

* домашняя страница, отображается при переходе по URL http://hostname/index.php
* страница "About" ("О нас")
* на странице "Contact" находится форма обратной связи, на которой пользователь может обратиться к разработчику по e-mail
* на странице "Login" отображается форма авторизации. Попытайтесь авторизоваться с логином/паролем "admin/admin". Обратите внимание на изменение раздела "Login" в главном меню на "Logout".

Эти страницы используют смежный хедер (шапка сайта) и футер (подвал). В "шапке" находится главное меню, при помощи которого пользователь перемещается по сайту. В "подвале" - копирайт и общая информация.

В самой нижней части окна Вы будете видеть системные сообщения Yii - журнал, отладочную информацию, сообщения об ошибках, запросы к базе данных и т.п. Выводом данной информации руководит [встроенный отладчик](https://github.com/yiisoft/yii2-debug/blob/master/docs/guide/README.md), он записывает и отображает информацию о ходе выполнения приложения.

В дополнение к веб приложению имеется консольный скрипт с названием yii, который находится в базовой директории приложения. Этот скрипт может быть использован для выполнения фоновых задач и обслуживания приложения.

### Структура приложения Yii

Ниже приведен список основных директорий и файлов вашего приложения (считаем, что приложение установлено в директорию basic):

basic/ корневой каталог приложения

composer.json используется Composer'ом, содержит описание приложения

config/ конфигурационные файлы

console.php конфигурация консольного приложения

web.php конфигурация Web приложения

commands/ содержит классы консольных команд

controllers/ контроллеры

models/ модели

runtime/ файлы, которые генерирует Yii во время выполнения приложения (логи, кэш и т.п.)

vendor/ содержит пакеты Composer'а и, собственно, сам фреймворк Yii

views/ виды приложения

web/ корневая директория Web приложения. Содержит файлы, доступные через Web

assets/ скрипты, используемые приложением (js, css)

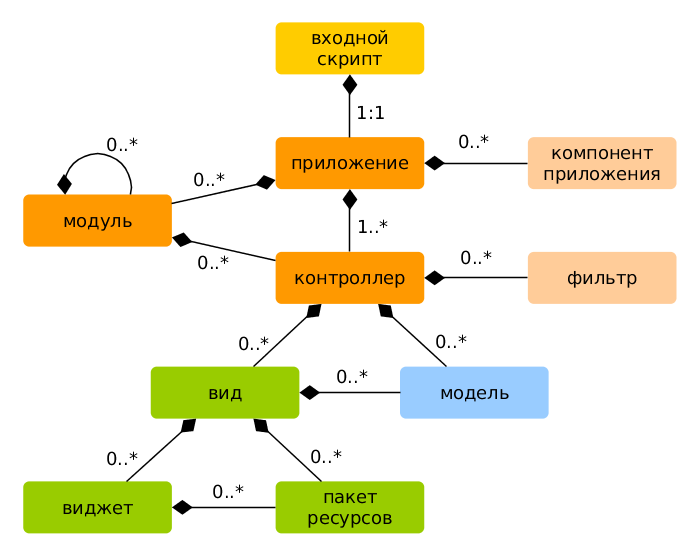
index.php точка входа в приложение Yii. С него начинается выполнение приложения

yii скрипт выполнения консольного приложения Yii

В целом, приложение Yii можно разделить на две категории файлов: расположенные в basic/web и расположенные в других директориях. Первая категория доступна через Web (например, браузером), вторая не доступна извне и не должна быть доступной т.к. содержит служебную информацию.

В Yii реализован [архитектурный паттерн MVC](http://ru.wikipedia.org/wiki/Model-View-Controller), которая соответствует структуре директорий приложения. В директории models находятся [Модели](http://stuff.cebe.cc/yii2docs-ru/guide-structure-models.html), в views расположены [Виды](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html), а в каталоге controllers все [Контроллеры](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html) приложения.

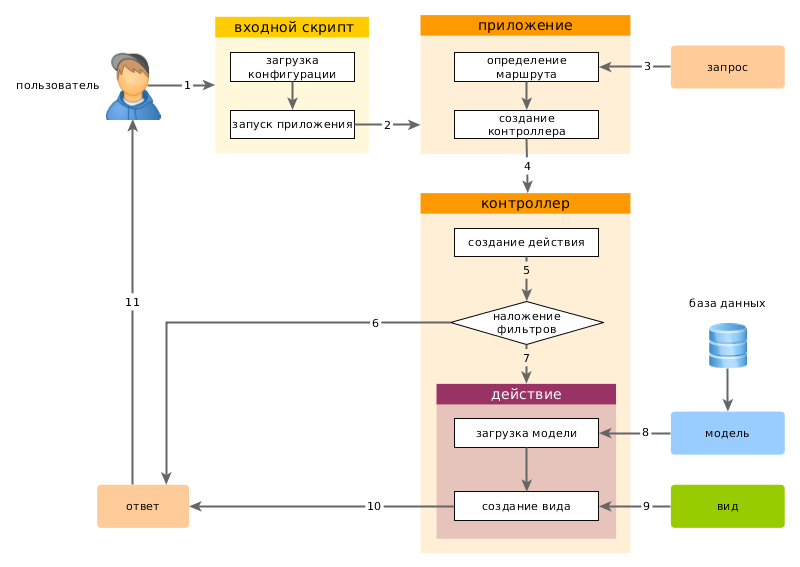
Диаграмма ниже демонстрирует внутреннее устройство приложения.



В каждом приложении Yii есть точка входа в приложение, web/index.php это единственный PHP-скрипт доступный для выполнения из Web. Он принимает входящий запрос и создает экземпляр [приложения](http://stuff.cebe.cc/yii2docs-ru/guide-structure-applications.html). [Приложение](http://stuff.cebe.cc/yii2docs-ru/guide-structure-applications.html) обрабатывает входящие запросы при помощи [компонентов](http://stuff.cebe.cc/yii2docs-ru/guide-concept-components.html) и отправляет запрос контроллеру. [Виджеты](http://stuff.cebe.cc/yii2docs-ru/guide-structure-widgets.html) используются в [Видах](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html) для построения динамических интерфейсов сайта.

### Жизненный цикл пользовательского запроса

На диаграмме показано как приложение обрабатывает запрос.



1. Пользователь обращается к [точке входа](http://stuff.cebe.cc/yii2docs-ru/guide-structure-entry-scripts.html) web/index.php.
2. Скрипт загружает конфигурацию [configuration](http://stuff.cebe.cc/yii2docs-ru/guide-concept-configurations.html) и создает экземпляр [приложения](http://stuff.cebe.cc/yii2docs-ru/guide-structure-applications.html) для дальнейшей обработки запроса.
3. Приложение определяет [маршрут](http://stuff.cebe.cc/yii2docs-ru/guide-runtime-routing.html) запроса при помощи компонента приложения [запрос](http://stuff.cebe.cc/yii2docs-ru/guide-runtime-requests.html).
4. Приложение создает экземпляр [контроллера](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html) для выполнения запроса.
5. Контроллер, в свою очередь, создает [действие](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html) и накладывает на него фильтры.
6. Если хотя бы один фильтр дает сбой, выполнение приложения останавливается.
7. Если все фильтры пройдены - приложение выполняется.
8. Действие загружает модель данных. Вероятнее всего из базы данных.
9. Действие генерирует вид, отображая в нем данные (в т.ч. и полученные из модели).
10. Сгенерированный вид приложения передается как компонент [ответ](http://stuff.cebe.cc/yii2docs-ru/guide-runtime-responses.html).
11. Компонент "ответ" отправляет готовый результат работы приложения браузеру пользователя.

### Говорим «Привет»

Рассмотрим, как создать новую страницу с надписью «Привет». В процессе решения задачи вы создадите [действие контроллера](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html) и [представление](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html):

* Приложение обработает запрос и передаст управление соответствующему действию;
* Действие, в свою очередь, отобразит представление с надписью "Привет" конечному пользователю.

### Создание действия

Для нашей задачи потребуется [действие](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html) say, которое читает параметр message из запроса и отображает его значение пользователю. Если в запросе не содержится параметра message, то действие будет выводить «Привет».

Действия объявляются в [контроллерах](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html). Для простоты, вы можете объявить действие say в уже существующем контроллере SiteController, который определен в файле класса controllers/SiteController.php:

<?php

namespace app\controllers;

use yii\web\Controller;

class SiteController extends Controller

{

// ...существующий код...

public function actionSay($message = 'Привет')

{

return $this->render('say', ['message' => $message]);

}

}

В приведенном коде действие say объявлено как метод actionSay в классе SiteController. Yii использует префикс action чтобы различать методы-действия и обычные методы. Название после префикса action считается идентификатором соответствующего действия.

**Информация:**Идентификаторы действий задаются в нижнем регистре. Если идентификатор состоит из нескольких слов, они соединяются дефисами, то есть create-comment. Имена методов действий получаются путём удаления дефисов из идентификатора, преобразования первой буквы каждого слова в верхний регистр и добавления префикса action. Например, идентификатор действия create-comment соответствует методу actionCreateComment.

Метод действия принимает параметр $message, который по умолчанию равен "Привет". Когда приложение получает запрос и определяет, что действие say ответственно за его обработку, параметр заполняется одноимённым значением из запроса.

Внутри метода действия, для вывода отображения [представления](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html) с именем say, используется метод [render()](http://stuff.cebe.cc/yii2docs-ru/yii-base-controller.html" \l "render()-detail). Для того, чтобы вывести сообщение, в отображение передаётся параметр message. Результат отображения при помощи return передаётся приложению, которое отдаёт его пользователю.

### Создание представления

[Представления](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html) являются скриптами, которые используются для формирования тела ответа. Для нашего приложения вы создадите представление say, которое будет выводить параметр message, полученный из метода действия:

<?php

use yii\helpers\Html;

?>

<?= Html::encode($message) ?>

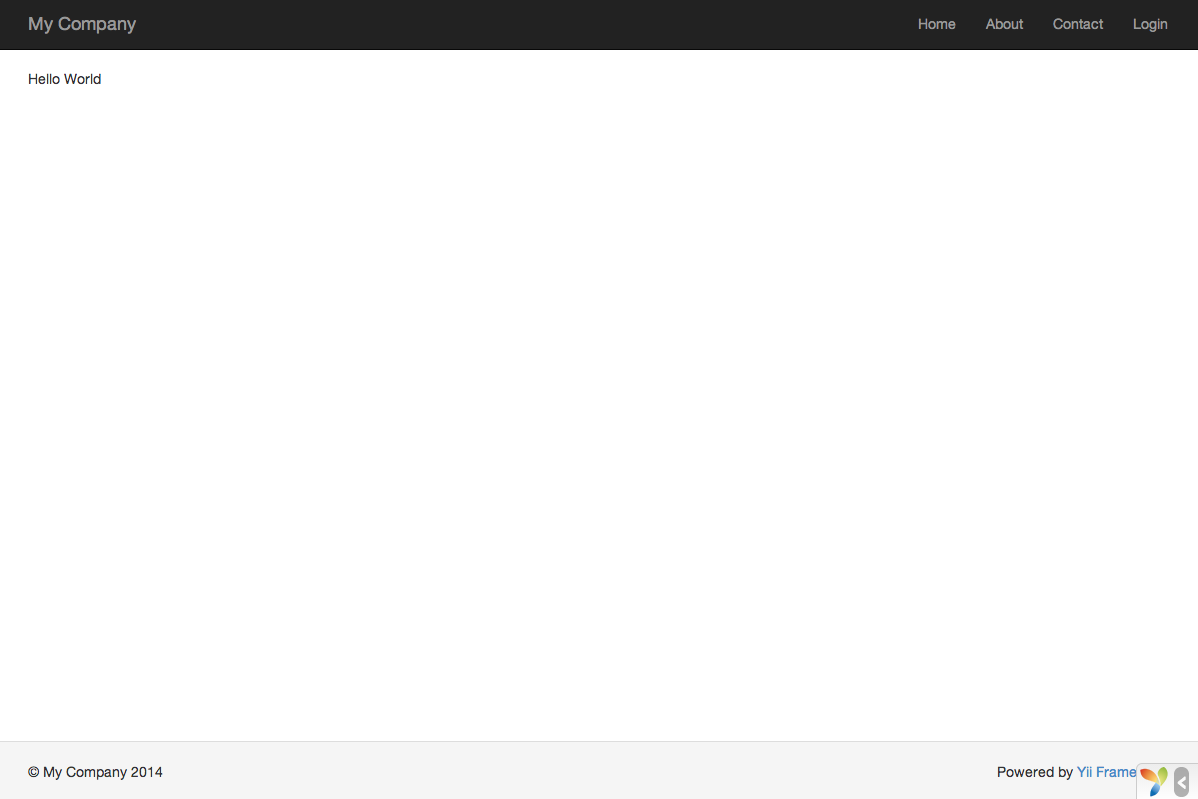
Представление say должно быть сохранено в файле views/site/say.php. Когда метод [render()](http://stuff.cebe.cc/yii2docs-ru/yii-base-controller.html" \l "render()-detail) вызывается в действии, он будет искать PHP файл с именем вида views/ControllerID/ViewName.php.

Стоит отметить, что в коде выше параметр message [экранируется для HTML](http://stuff.cebe.cc/yii2docs-ru/yii-helpers-basehtml.html" \l "encode()-detail) перед выводом. Это обязательно так как параметр приходит от пользователя, который может попытаться провести [XSS атаку](http://ru.wikipedia.org/wiki/Межсайтовый_скриптинг) путём вставки зловредного JavaScript кода.

Вы можете дополнить представление say HTML тегами, текстом или кодом PHP. Фактически, представление say является простым PHP скриптом, который выполняется методом [render()](http://stuff.cebe.cc/yii2docs-ru/yii-base-controller.html" \l "render()-detail). Содержимое, выводимое скриптом представления, будет передано пользователю приложением.

После создания действия и представления вы можете перейти на новую страницу по следующему URL:

http://hostname/index.php?r=site%2Fsay&message=Привет+мир



Будет отображена страница с надписью «Привет мир». Она использует ту же шапку и футер, что и остальные страницы приложения. Если вы не укажете параметр message, то увидите на странице «Привет». Это происходит потому, как message передаётся в метод actionSay() и значение по умолчанию — «Привет».

**Информация:**Новая страница использует ту же шапку и футер, что и другие страницы, потому что метод [render()](http://stuff.cebe.cc/yii2docs-ru/yii-base-controller.html" \l "render()-detail) автоматически вставляет результат представления say в, так называемый, [макет](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html) views/layouts/main.php.

Параметр r требует дополнительных пояснений. Он связан с [маршрутом (route)](http://stuff.cebe.cc/yii2docs-ru/guide-runtime-routing.html), который представляет собой уникальный идентификатор, указывающий на действие. Его формат ControllerID/ActionID. Когда приложение получает запрос, оно проверяет параметр r и, используя ControllerID, определяет какой контроллер следует использовать для обработки запроса. Затем, контроллер использует часть ActionID, чтобы определить какое действие выполняет реальную работу. В нашем случае маршрут site/say будет соответствовать контроллеру SiteController и его действию say. В результате, для обработки запроса будет вызван метод SiteController::actionSay().

**Информация:**Как и действия, контроллеры также имеют идентификаторы, которые однозначно определяют их в приложении. Идентификаторы контроллеров используют те же правила именования, что и идентификаторы действий. Имена классов контроллеров получаются путём удаления дефисов из идентификатора, преобразования первой буквы каждого слова в верхний регистр и добавления в конец Controller. Например, идентификатор контроллера post-comment соответствует имени класса контроллера PostCommentController.

**Задание:**

1. Реализуйте все примеры, описанные выше.
2. Создайте новое действие, принимающее три параметра (имя, пол и возраст). Также задайте им значения по умолчанию. Передайте эти параметры в соответствующее представление. В представлении должна происходить генерация поздравления с днем рождения с учетом входящих параметров.

## Работа с базами данных

Этот раздел расскажет о том, как создать новую страницу, отображающую данные по странам, полученные из таблицы countries базы данных. Для достижения этой цели вам будет необходимо настроить подключение к базе данных, создать класс [Active Record](http://stuff.cebe.cc/yii2docs-ru/guide-db-active-record.html), определить [action](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html), и создать [view](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html).

Изучив эту часть, вы научитесь:

* Настраивать подключение к БД
* Определять класс Active Record
* Запрашивать данные, используя класс Active Record
* Отображать данные во view с использованием пагинации

Обратите внимание, чтобы усвоить этот раздел, вы должны иметь базовые знания и навыки использования баз данных. В частности, вы должны знать, как создать базу данных, и как выполнять SQL запросы, используя клиентские инструменты для работы с БД.

### Подготавливаем базу данных

Для начала, создайте базу данных под названием yii2basic, из которой вы будете получать данные в вашем приложении. Вы можете создать базу данных SQLite, MySQL, PostgreSQL, MSSQL или Oracle, так как Yii имеет встроенную поддержку для многих баз данных. Для простоты, в дальнейшем описании будет подразумеваться MySQL.

После этого создайте в базе данных таблицу country, и добавьте в неё немного демонстрационных данных. Вы можете запустить следующую SQL инструкцию, чтобы сделать это:

CREATE TABLE `country` (

`code` CHAR(2) NOT NULL PRIMARY KEY,

`name` CHAR(52) NOT NULL,

`population` INT(11) NOT NULL DEFAULT '0'

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `country` VALUES ('AU','Australia',24016400);

INSERT INTO `country` VALUES ('BR','Brazil',205722000);

INSERT INTO `country` VALUES ('CA','Canada',35985751);

INSERT INTO `country` VALUES ('CN','China',1375210000);

INSERT INTO `country` VALUES ('DE','Germany',81459000);

INSERT INTO `country` VALUES ('FR','France',64513242);

INSERT INTO `country` VALUES ('GB','United Kingdom',65097000);

INSERT INTO `country` VALUES ('IN','India',1285400000);

INSERT INTO `country` VALUES ('RU','Russia',146519759);

INSERT INTO `country` VALUES ('US','United States',322976000);

На данный момент у вас есть база данных под названием yii2basic, и внутри неё таблица country с тремя столбцами, содержащими десять строк данных.

### Настраиваем подключение к БД

Откройте файл config/db.php и измените параметры на верные для вашей базы данных. По умолчанию этот файл содержит следующее:

<?php

return [

'class' => 'yii\db\Connection',

'dsn' => 'mysql:host=localhost;dbname=yii2basic',

'username' => 'root',

'password' => '',

'charset' => 'utf8',

];

Файл config/db.php - типичный [конфигурационный](http://stuff.cebe.cc/yii2docs-ru/guide-concept-configurations.html) инструмент, базирующийся на файлах. Данный конфигурационный файл определяет параметры, необходимые для создания и инициализации экземпляра [yii\db\Connection](http://stuff.cebe.cc/yii2docs-ru/yii-db-connection.html), через который вы можете делать SQL запросы к подразумеваемой базе данных.

Подключение к БД, настроенное выше, доступно в коде приложения через выражение Yii::$app->db.

**Информация:**файл config/db.php будет подключен главной конфигурацией приложения config/web.php, описывающей то, как экземпляр [приложения](http://stuff.cebe.cc/yii2docs-ru/guide-structure-applications.html) должен быть инициализирован.

### Создаём потомка Active Record

Чтобы представлять и получать данные из таблицы country, создайте класс - потомок [Active Record](http://stuff.cebe.cc/yii2docs-ru/guide-db-active-record.html), под названием Country, и сохраните его в файле models/Country.php.

<?php

namespace app\models;

use yii\db\ActiveRecord;

class Country extends ActiveRecord

{

}

Класс Country наследуется от [yii\db\ActiveRecord](http://stuff.cebe.cc/yii2docs-ru/yii-db-activerecord.html). Вам не нужно писать ни строчки кода внутри него! С кодом, приведённым выше, Yii свяжет имя таблицы с именем класса.

**Информация:**Если нет возможности задать прямой зависимости между именем таблицы и именем класса, вы можете переопределить метод [yii\db\ActiveRecord::tableName()](http://stuff.cebe.cc/yii2docs-ru/yii-db-activerecord.html" \l "tableName()-detail), чтобы явно задать имя связанной таблицы.

Используя класс Country, вы можете легко манипулировать данными в таблице country, как показано в этих фрагментах:

use app\models\Country;

// получаем все строки из таблицы "country" и сортируем их по "name"

$countries = Country::find()->orderBy('name')->all();

// получаем строку с первичным ключом "US"

$country = Country::findOne('US');

// отобразит "United States"

echo $country->name;

// меняем имя страны на "U.S.A." и сохраняем в базу данных

$country->name = 'U.S.A.';

$country->save();

### Создаём Action

Для того, чтобы показать данные по странам конечным пользователям, вам надо создать новый action. Вместо размещения нового action'a в контроллере site, как вы делали в предыдущих разделах, будет иметь больше смысла создать новый контроллер специально для всех действий, относящихся к данным по странам. Назовите новый контроллер CountryController, и создайте action index внутри него, как показано ниже.

<?php

namespace app\controllers;

use yii\web\Controller;

use yii\data\Pagination;

use app\models\Country;

class CountryController extends Controller

{

public function actionIndex()

{

$query = Country::find();

$pagination = new Pagination([

'defaultPageSize' => 5,

'totalCount' => $query->count(),

]);

$countries = $query->orderBy('name')

->offset($pagination->offset)

->limit($pagination->limit)

->all();

return $this->render('index', [

'countries' => $countries,

'pagination' => $pagination,

]);

}

}

Сохраните код выше в файле controllers/CountryController.php.

Action index вызывает Country::find(). Данный метод Active Record строит запрос к БД и извлекает все данные из таблицы country. Чтобы ограничить количество стран, возвращаемых каждым запросом, запрос разбивается на страницы с помощью объекта [yii\data\Pagination](http://stuff.cebe.cc/yii2docs-ru/yii-data-pagination.html). Объект Pagination служит двум целям:

* Устанавливает пункты offset и limit для SQL инструкции, представленной запросом, чтобы она возвращала только одну страницу данных за раз (в нашем случае максимум 5 строк на страницу).
* Он используется во view для отображения пагинатора, состоящего из набора кнопок с номерами страниц, это будет разъяснено в следующем подразделе.

В конце кода action index выводит view с именем index, и передаёт в него данные по странам вместе c информацией о пагинации.

### Создаём View

Первым делом создайте поддиректорию с именем country внутри директории views. Эта папка будет использоваться для хранения всех view, выводимых контроллером country. Внутри директории views/country создайте файл с именем index.php, содержащий следующий код:

<?php

use yii\helpers\Html;

use yii\widgets\LinkPager;

?>

<h1>Countries</h1>

<ul>

<?php foreach ($countries as $country): ?>

<li>

<?= Html::encode("{$country->name} ({$country->code})") ?>:

<?= $country->population ?>

</li>

<?php endforeach; ?>

</ul>

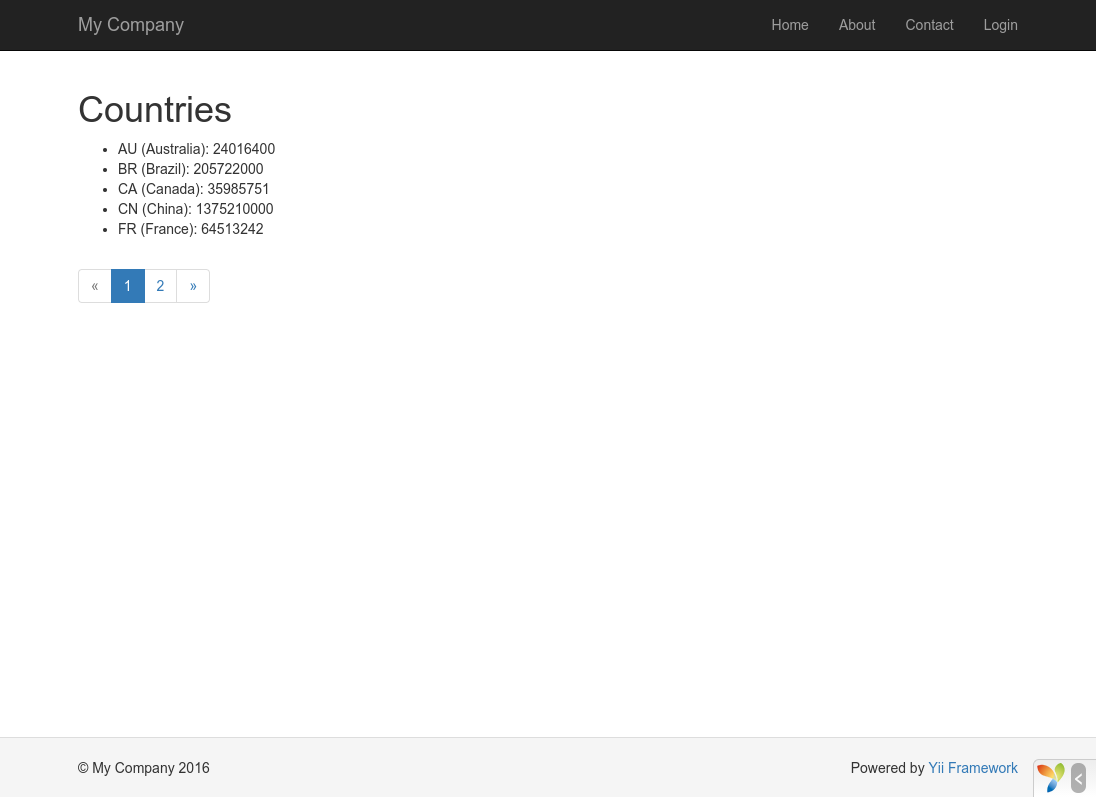
<?= LinkPager::widget(['pagination' => $pagination]) ?>

View имеет 2 части относительно отображения данных по странам. В первой части предоставленные данные по странам выводятся как неупорядоченный HTML-список. Во второй части выводится виджет [yii\widgets\LinkPager](http://stuff.cebe.cc/yii2docs-ru/yii-widgets-linkpager.html), используя информацию о пагинации, переданную из action во view. Виджет LinkPager отображает набор постраничных кнопок. Клик по любой из них обновит данные по странам в соответствующей странице.

### Испытываем в действии

Чтобы увидеть, как работает весь вышеприведённый код, перейдите по следующей ссылке в своём браузере:

http://hostname/index.php?r=country%2Findex



В начале вы увидите страницу, показывающую пять стран. Под странами вы увидите пагинатор с четырьмя кнопками. Если вы кликните по кнопке "2", то увидите страницу, отображающую другие пять стран из базы данных: вторая страница записей. Посмотрев внимательней, вы увидите, что URL в браузере тоже сменилось на

http://hostname/index.php?r=country%2Findex&page=2

За кадром, [Pagination](http://stuff.cebe.cc/yii2docs-ru/yii-data-pagination.html) предоставляет всю необходимую функциональность для постраничной разбивки набора данных:

* В начале [Pagination](http://stuff.cebe.cc/yii2docs-ru/yii-data-pagination.html) показывает первую страницу, которая отражает SELECT запрос стран с параметрами LIMIT 5 OFFSET 0. Как результат, первые пять стран будут получены и отображены.
* Виджет [LinkPager](http://stuff.cebe.cc/yii2docs-ru/yii-widgets-linkpager.html) выводит кнопки страниц используя URL'ы, созданные [Pagination](http://stuff.cebe.cc/yii2docs-ru/yii-data-pagination.html" \l "createUrl()-detail). Эти URL'ы будут содержать параметр запроса page, который представляет различные номера страниц.
* Если вы кликните по кнопке "2", сработает и обработается новый запрос для маршрута country/index. Таким образом новый запрос стран будет иметь параметры LIMIT 5 OFFSET 5 и вернет следующие пять стран для отображения.

**Задание:**

1. Реализуйте все примеры, описанные выше.
2. Изучите особенности работы с ActiveRecord. <http://stuff.cebe.cc/yii2docs-ru/yii-db-activerecord.html>

Создайте новое действие, в котором в зависимости от передаваемого параметра, происходит сортировка по разным полям + выбираются страны, с количеством жителей большем передаваемого параметра.

# Лабораторная работа №2. Работа с формами. Генерация кода при помощи Gii.

## Работа с формами

В данном разделе мы обсудим получение данных от пользователя. На странице будет располагаться форма с полями для ввода имени и email. Полученные данные будут показаны на странице для их подтверждения.

Чтобы достичь этой цели, помимо создания [действия](http://stuff.cebe.cc/yii2docs-ru/guide-structure-controllers.html) и двух [представлений](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html) вы создадите [модель](http://stuff.cebe.cc/yii2docs-ru/guide-structure-models.html).

В данном руководстве вы изучите:

* Как создать [модель](http://stuff.cebe.cc/yii2docs-ru/guide-structure-models.html) для данных, введённых пользователем;
* Как объявить правила проверки введённых данных;
* Как создать HTML форму в [представлении](http://stuff.cebe.cc/yii2docs-ru/guide-structure-views.html).

### Создание модели

В файле models/EntryForm.php создайте класс модели EntryForm как показано ниже. Он будет использоваться для хранения данных, введённых пользователем.

<?php

namespace app\models;

use yii\base\Model;

class EntryForm extends Model

{

public $name;

public $email;

public function rules()

{

return [

[['name', 'email'], 'required'],

['email', 'email'],

];

}

}

Данный класс расширяет класс [yii\base\Model](http://stuff.cebe.cc/yii2docs-ru/yii-base-model.html), который является частью фреймворка и обычно используется для работы с данными форм.

Класс содержит 2 публичных свойства name и email, которые используются для хранения данных, введённых пользователем. Он также содержит метод rules(), который возвращает набор правил проверки данных. Правила, объявленные в коде выше означают следующее:

* Поля name и email обязательны для заполнения;
* В поле email должен быть правильный адрес email.

Если объект EntryForm заполнен пользовательскими данными, то для их проверки вы можете вызвать метод [validate()](http://stuff.cebe.cc/yii2docs-ru/yii-base-model.html" \l "validate()-detail). В случае неудачной проверки свойство [hasErrors](http://stuff.cebe.cc/yii2docs-ru/yii-base-model.html" \l "hasErrors()-detail) станет равным true. С помощью [errors](http://stuff.cebe.cc/yii2docs-ru/yii-base-model.html" \l "getErrors()-detail) можно узнать, какие именно ошибки возникли.

### Создание действия

Далее создайте действие entry в контроллере site, точно так же, как вы делали это ранее.

<?php

namespace app\controllers;

use Yii;

use yii\web\Controller;

use app\models\EntryForm;

class SiteController extends Controller

{

// ...существующий код...

public function actionEntry()

{

$model = new EntryForm();

if ($model->load(Yii::$app->request->post()) && $model->validate()) {

// данные в $model удачно проверены

// делаем что-то полезное с $model ...

return $this->render('entry-confirm', ['model' => $model]);

} else {

// либо страница отображается первый раз, либо есть ошибка в данных

return $this->render('entry', ['model' => $model]);

}

}

}

Действие создает объект EntryForm. Затем оно пытается заполнить модель данными из массива $\_POST, доступ к которому обеспечивает Yii при помощи [yii\web\Request::post()](http://stuff.cebe.cc/yii2docs-ru/yii-web-request.html" \l "post()-detail). Если модель успешно заполнена, то есть пользователь отправил данные из HTML формы, то для проверки данных будет вызван метод [validate()](http://stuff.cebe.cc/yii2docs-ru/yii-base-model.html" \l "validate()-detail).

Если всё в порядке, действие отобразит представление entry-confirm, которое показывает пользователю введенные им данные. В противном случае будет отображено представление entry, которое содержит HTML форму и ошибки проверки данных, если они есть.

**Информация:**Yii::$app представляет собой глобально доступный экземпляр-одиночку [приложения](http://stuff.cebe.cc/yii2docs-ru/guide-structure-applications.html) (singleton). Одновременно это [Service Locator](http://stuff.cebe.cc/yii2docs-ru/guide-concept-service-locator.html), дающий доступ к компонентам вроде request, response, db и так далее. В коде выше для доступа к данным из $\_POST был использован компонент request.

### Создание представления

В заключение, создаём два представления с именами entry-confirm и entry, которые отображаются действием entry из предыдущего подраздела.

Представление entry-confirm просто отображает имя и email. Оно должно быть сохранено в файле views/site/entry-confirm.php.

<?php

use yii\helpers\Html;

?>

<p>Вы ввели следующую информацию:</p>

<ul>

<li><label>Name</label>: <?= Html::encode($model->name) ?></li>

<li><label>Email</label>: <?= Html::encode($model->email) ?></li>

</ul>

Представление entry отображает HTML форму. Оно должно быть сохранено в файле views/site/entry.php.

<?php

use yii\helpers\Html;

use yii\widgets\ActiveForm;

?>

<?php $form = ActiveForm::begin(); ?>

<?= $form->field($model, 'name') ?>

<?= $form->field($model, 'email') ?>

<div class="form-group">

<?= Html::submitButton('Отправить', ['class' => 'btn btn-primary']) ?>

</div>

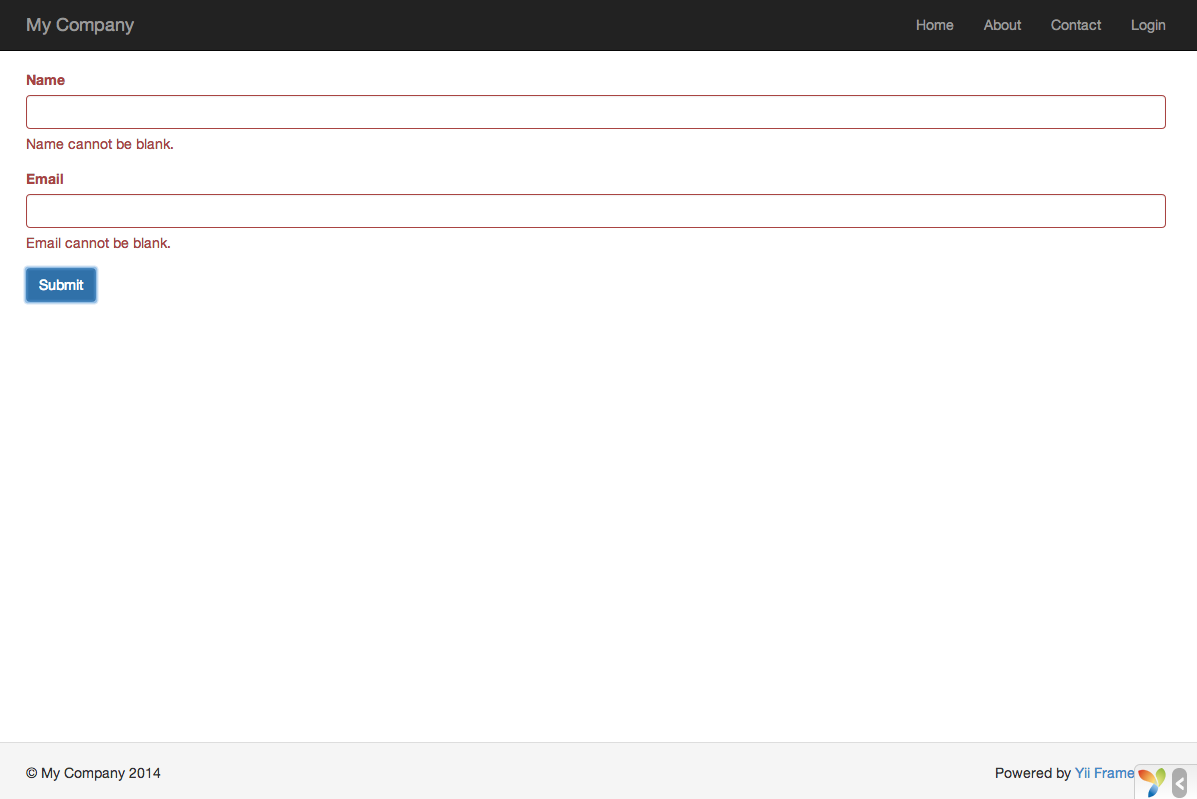
<?php ActiveForm::end(); ?>

Для построения HTML формы представление использует мощный [виджет](http://stuff.cebe.cc/yii2docs-ru/guide-structure-widgets.html) [ActiveForm](http://stuff.cebe.cc/yii2docs-ru/yii-widgets-activeform.html). Методы begin() и end() выводят открывающий и закрывающий теги формы. Между этими вызовами создаются поля ввода при помощи метода [field()](http://stuff.cebe.cc/yii2docs-ru/yii-widgets-activeform.html" \l "field()-detail). Первым идёт поле для "name", вторым — для "email". Далее для генерации кнопки отправки данных вызывается метод [yii\helpers\Html::submitButton()](http://stuff.cebe.cc/yii2docs-ru/yii-helpers-basehtml.html" \l "submitButton()-detail).

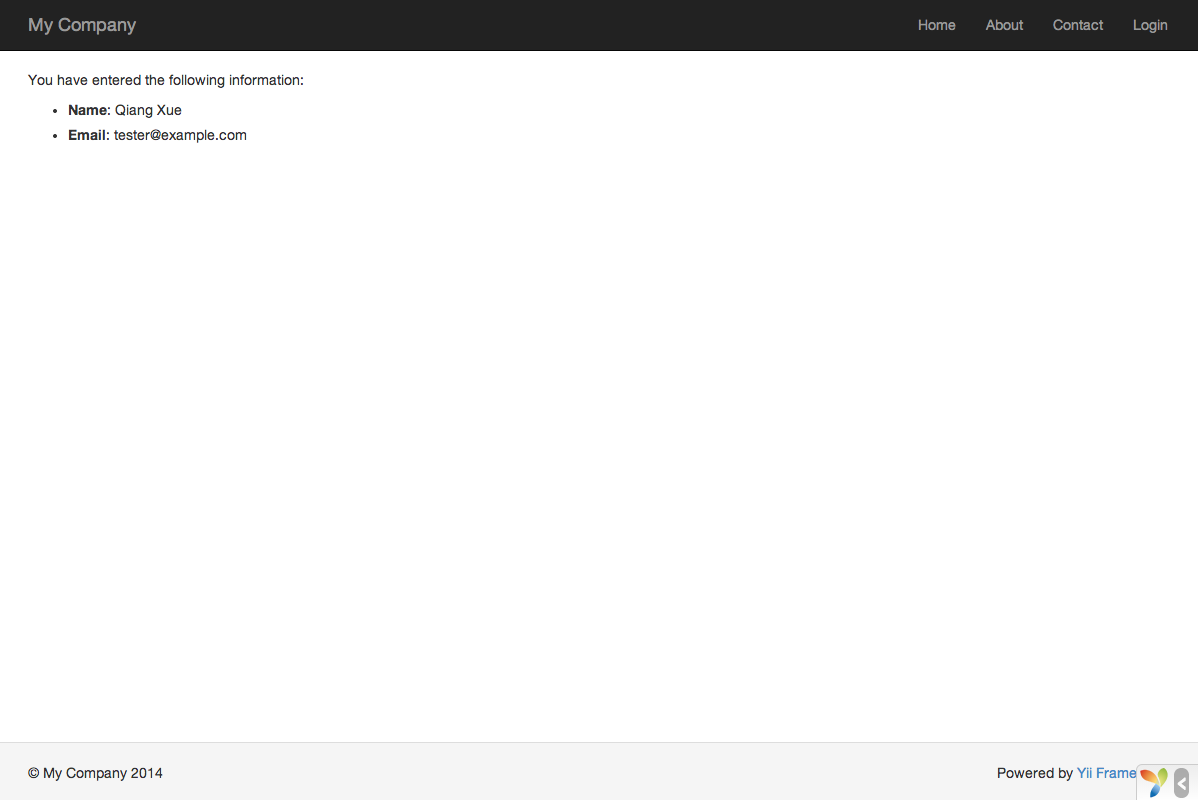
Чтобы увидеть всё созданное в работе, откройте в браузере следующий URL:

http://hostname/index.php?r=site%2Fentry

Вы увидите страницу с формой и двумя полями для ввода. Перед каждым полем имеется подпись, которая указывает, какую информацию следует вводить. Если вы нажмёте на кнопку отправки без ввода данных или если вы введете email в неверном формате, вы увидите сообщение с ошибкой рядом с каждым проблемным полем.



После ввода верных данных и их отправки, вы увидите страницу с данными, которые вы только что ввели.



### Как работает вся эта «магия»

Вы, скорее всего, задаётесь вопросом о том, как же эта HTML форма работает на самом деле. Весь процесс может показаться немного волшебным: то как показываются подписи к полям, ошибки проверки данных при некорректном вводе и то что всё это происходит без перезагрузки страницы.

Да, проверка данных на самом деле происходит и на стороне клиента при помощи JavaScript и на стороне сервера.[yii\widgets\ActiveForm](http://stuff.cebe.cc/yii2docs-ru/yii-widgets-activeform.html) достаточно продуман, чтобы взять правила проверки, которые вы объявили в EntryForm, преобразовать их в JavaScript код и использовать его для проведения проверок. На случай отключения JavaScript в браузере валидация проводится и на стороне сервера как показано в методе actionEntry(). Это даёт уверенность в том, что данные корректны при любых обстоятельствах.

Подписи для полей генерируются методом field(), на основе имён свойств модели. Например, подпись Name генерируется для свойства name. Вы можете модифицировать подписи следующим образом:

<?= $form->field($model, 'name')->label('Ваше имя') ?>

<?= $form->field($model, 'email')->label('Ваш Email') ?>

**Задание:**

1. Реализуйте все примеры, описанные выше.
2. Создайте нужную модель, действие и представление для того, чтобы можно было добавлять записи с помощью формы в таблицу country, созданную в предыдущей лабораторной

## Генерация кода при помощи Gii

В этом разделе мы опишем, как использовать [Gii](https://github.com/yiisoft/yii2-gii/blob/master/docs/guide-ru/README.md) для автоматической генерации кода, реализующего некоторые общие функции вебсайта. Для достижения этой цели всё, что вам нужно, это просто ввести необходимую информацию в соответствии с инструкциями, отображаемыми на веб-страницах Gii.

В этом руководстве вы узнаете:

* Как активировать Gii в приложении;
* Как использовать Gii для создания Active Record класса;
* Как использовать Gii для генерации кода, реализующего CRUD для таблицы БД.
* Как настроить код, генерируемый Gii.

### Запускаем Gii

[Gii](https://github.com/yiisoft/yii2-gii/blob/master/docs/guide-ru/README.md) представлен в Yii как [модуль](http://stuff.cebe.cc/yii2docs-ru/guide-structure-modules.html). Вы можете активировать Gii, настроив его в свойстве yii\base\Application::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](http://stuff.cebe.cc/yii2docs-ru/guide-concept-configurations.html" \l "environment-constants), приложение должно включать в себя модуль с именем gii, который реализует класс [yii\gii\Module](http://stuff.cebe.cc/yii2docs-ru/yii-gii-module.html).

Если вы посмотрите [входной скрипт](http://stuff.cebe.cc/yii2docs-ru/guide-structure-entry-scripts.html) 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

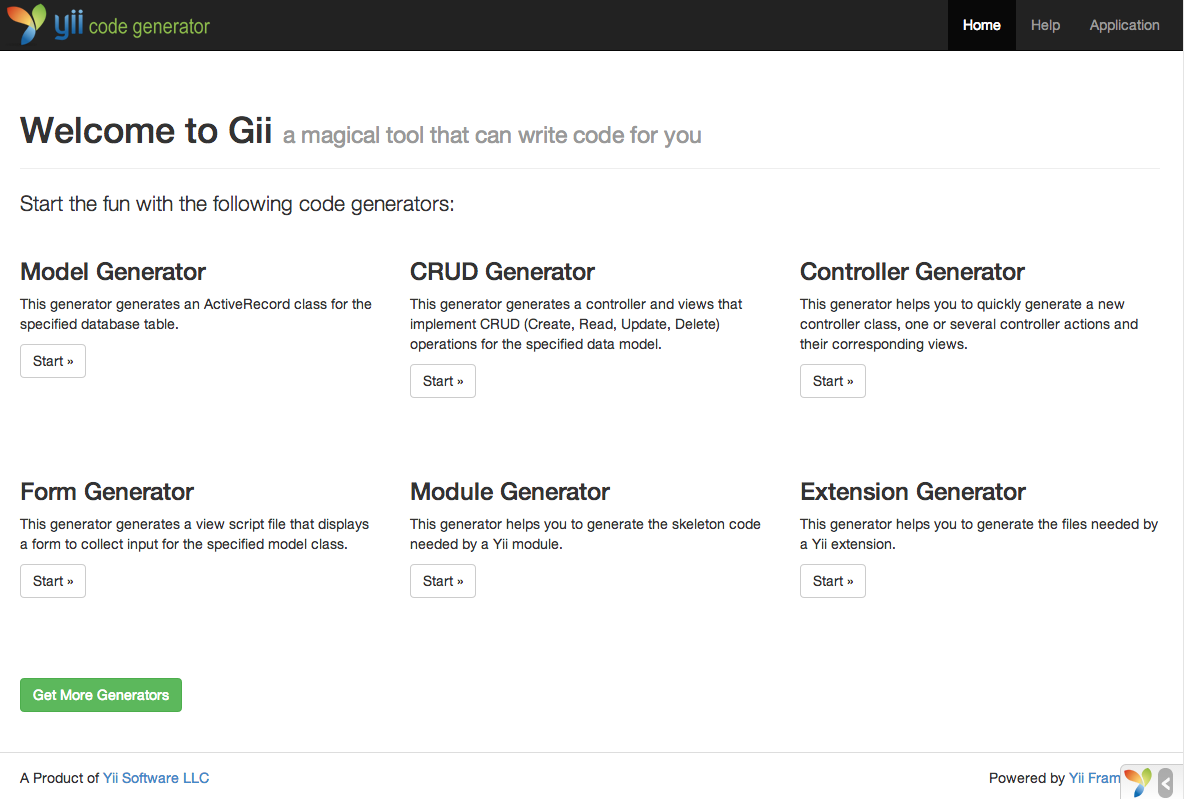
**Примечание:**Если вы пытаетесь получить доступ к Gii не с локального хоста, по умолчанию, в целях обеспечения безопасности, доступ будет запрещён. Вы можете изменить настройки Gii, чтобы добавить разрешённые IP адреса, как указано ниже

'gii' => [

'class' => 'yii\gii\Module',

'allowedIPs' => ['127.0.0.1', '::1', '192.168.0.\*', '192.168.178.20'] // регулируйте в соответствии со своими нуждами

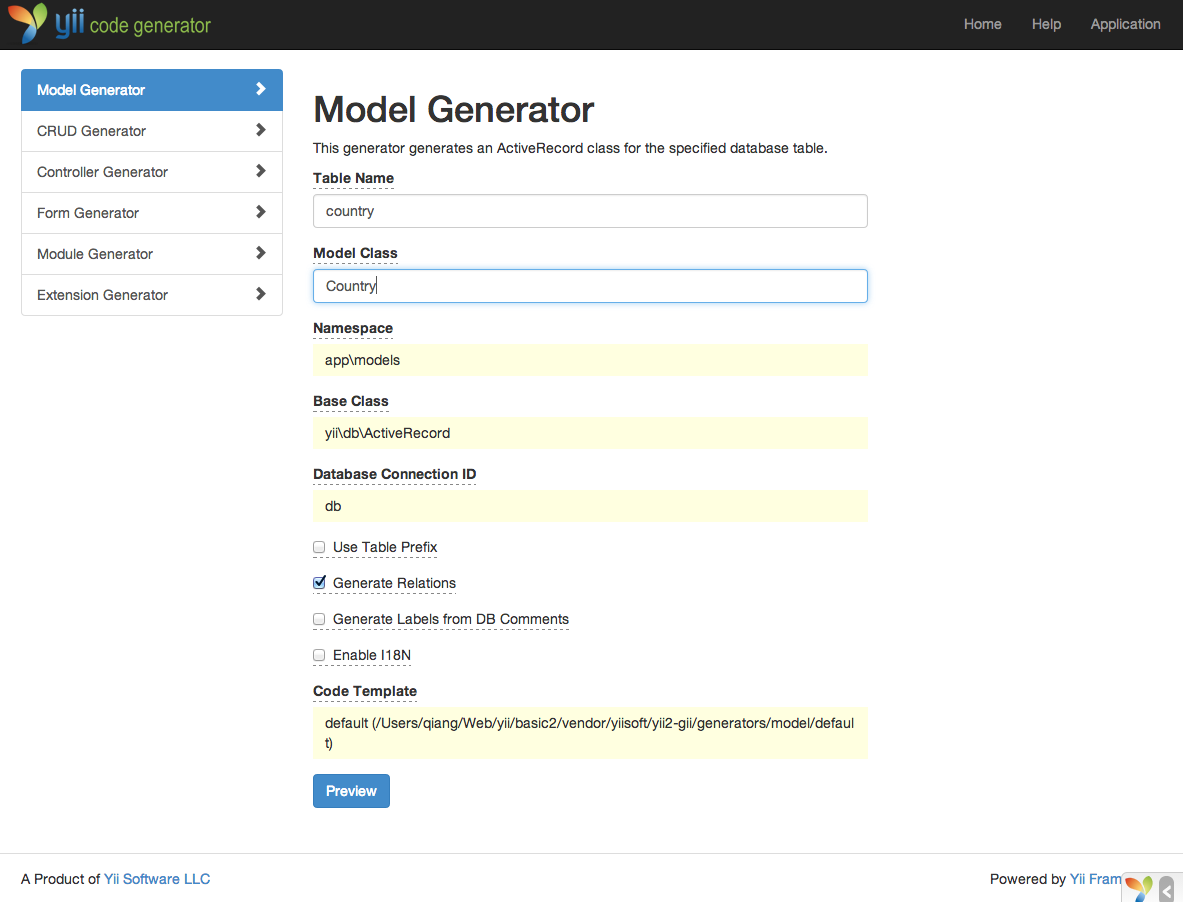
],



### Генерация класса Active Record

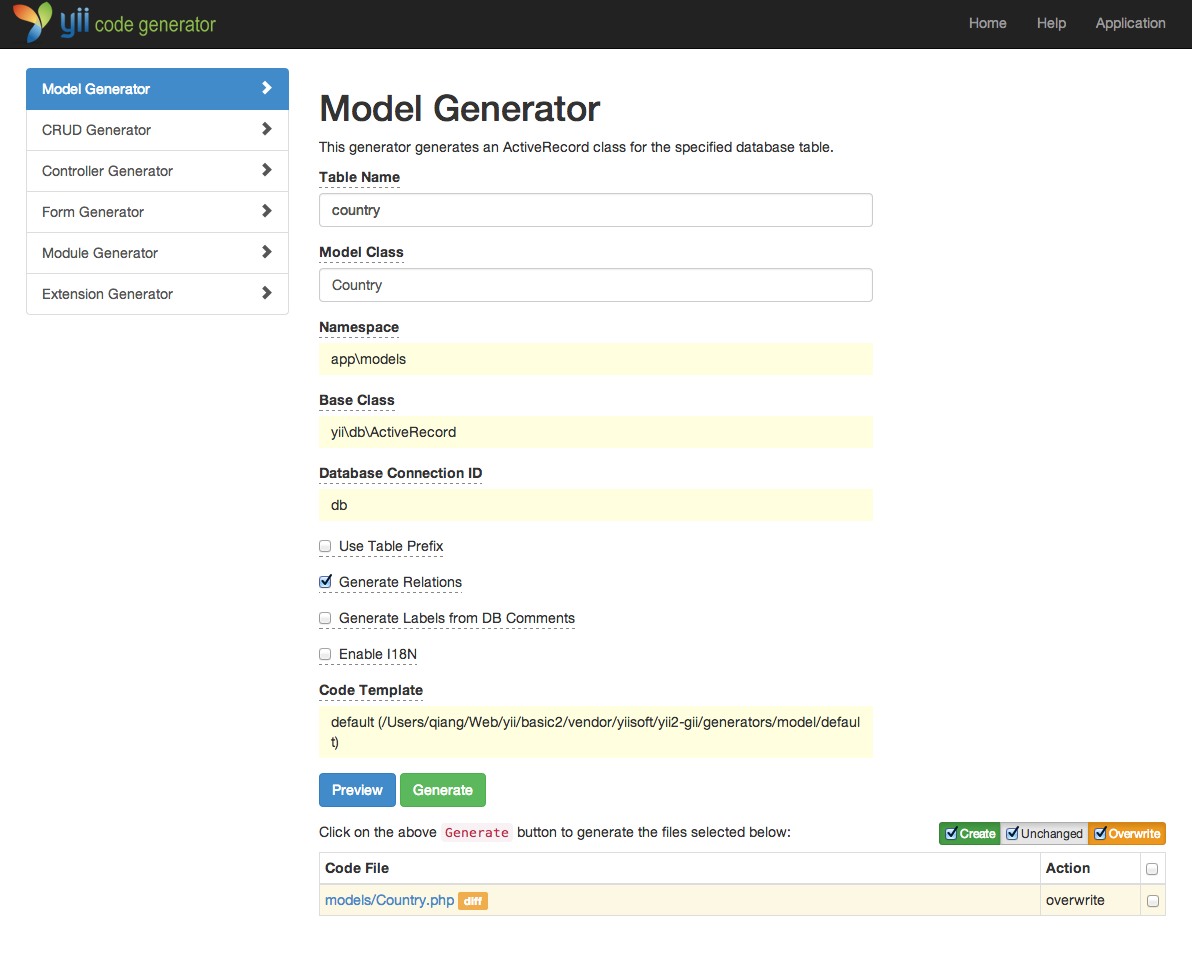
Чтобы использовать Gii для генерации класса Active Record, выберите "Генератор модели" (нажав на ссылку на главной странице Gii). И заполните форму следующим образом:

* Имя таблицы: country
* Класс модели : Country



Затем нажмите на кнопку "Предварительный просмотр". Вы увидите, что models/Country.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.



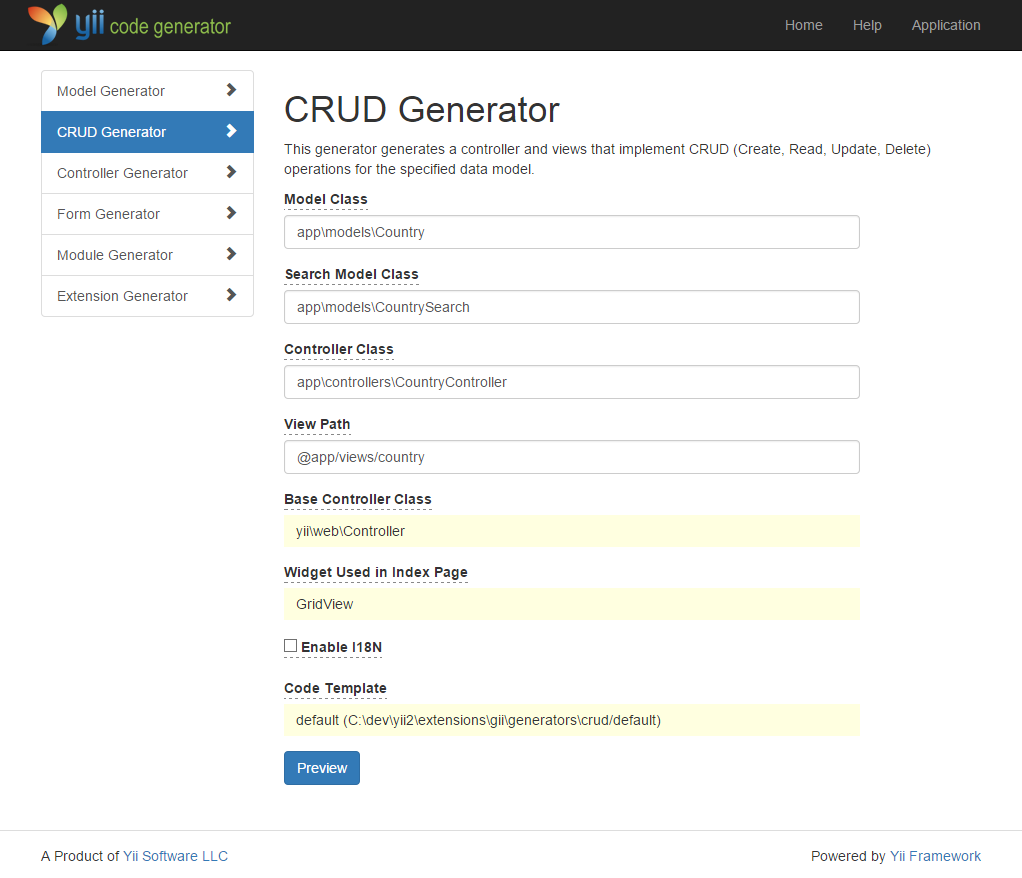
Для перезаписи существующего файла установите флажок рядом с "overwrite" и нажмите кнопку "Generate". Для создания нового файла вы можете просто нажать "Generate".

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.

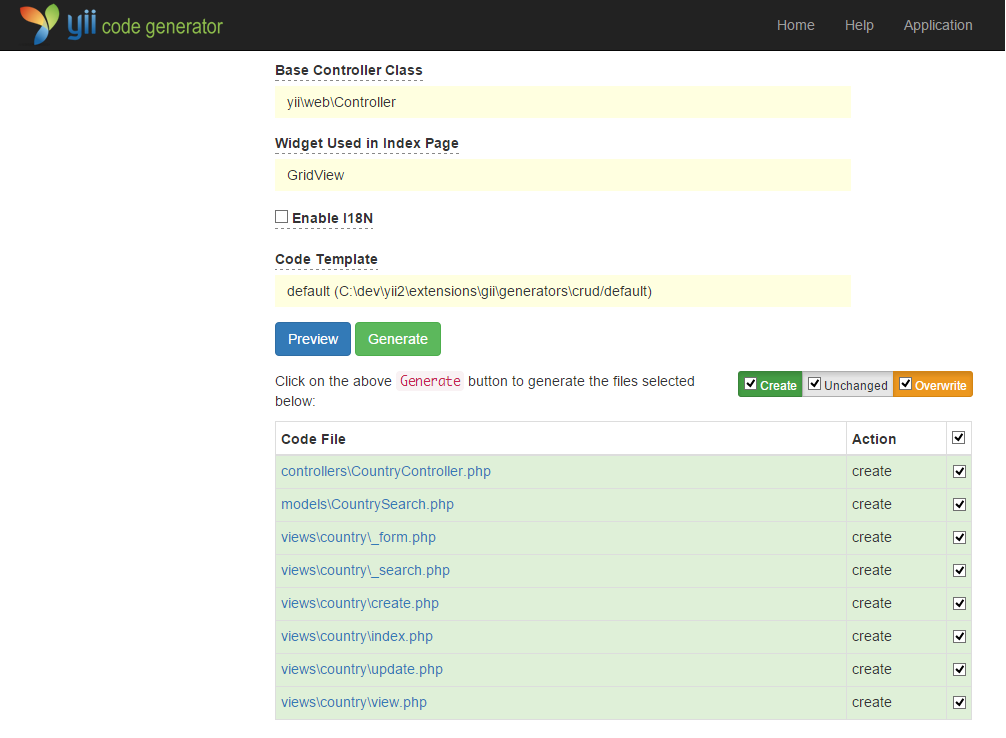
### Создание CRUD кода

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). Для нашей таблицы «country» заполните полученную форму следующим образом:

* Model Class: app\models\Country
* Search Model Class: app\models\CountrySearch
* Controller Class: app\controllers\CountryController



Затем нажмите на кнопку "Preview". Вы увидите список файлов, которые будут созданы, как показано ниже.



Если вы уже создали файлы controllers/CountryController.php и views/country/index.php, установите флажок "overwrite", чтобы заменить их. (Предыдущие версии не поддерживают CRUD полностью)

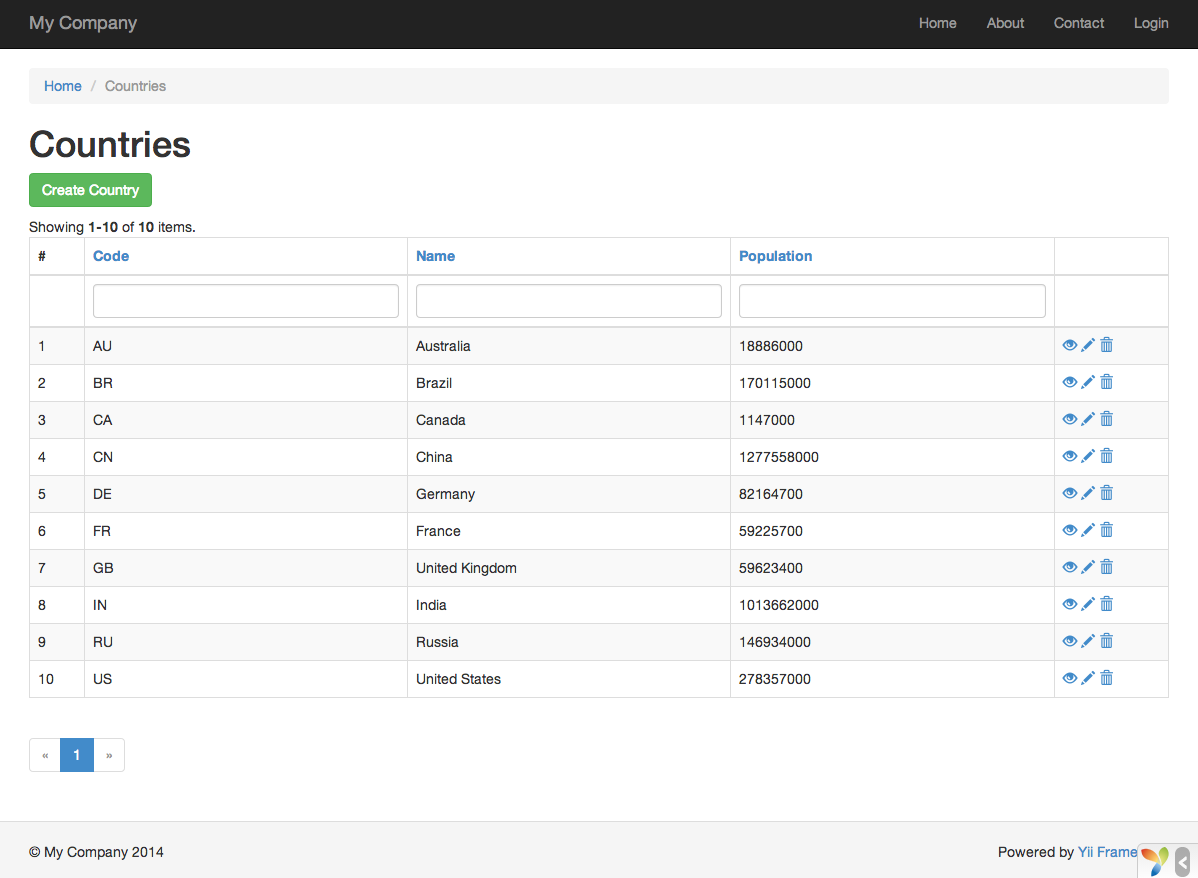
### Испытываем в действии

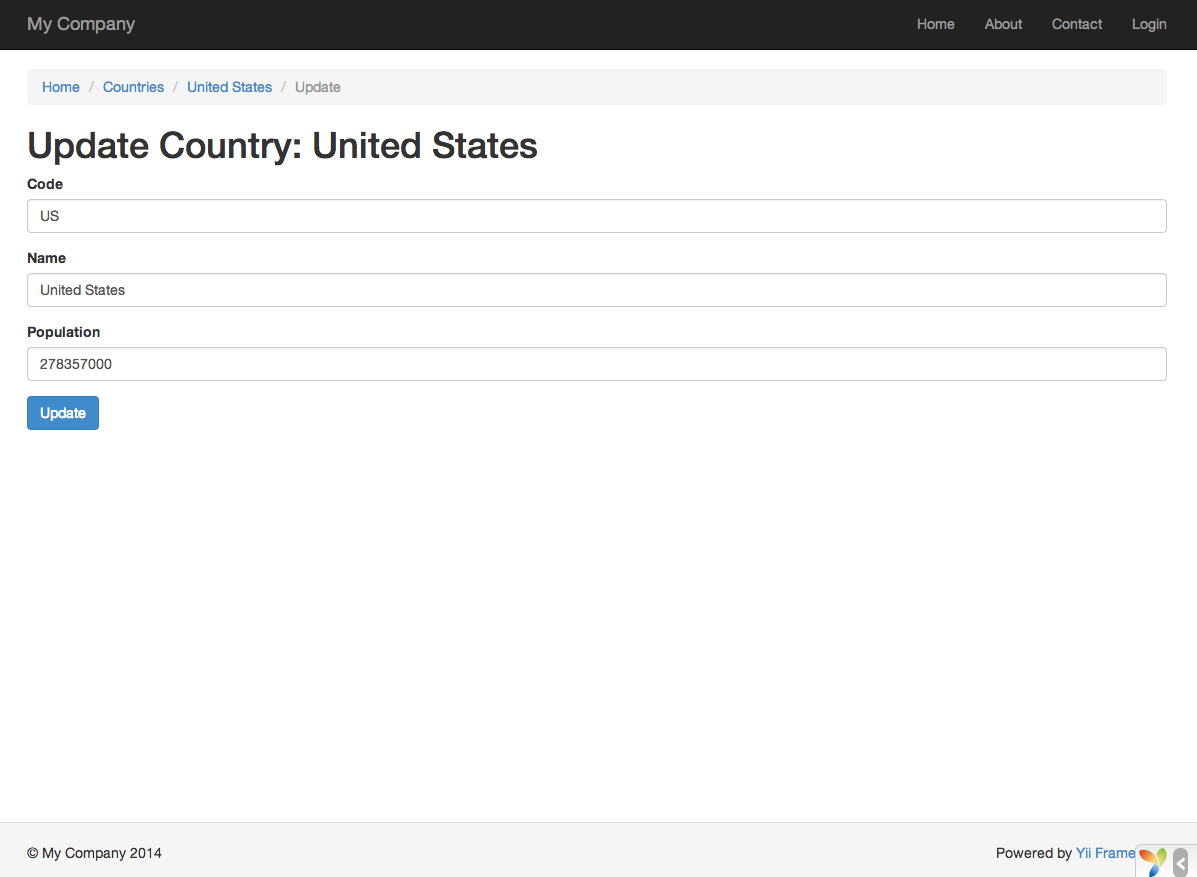
Чтобы увидеть как всё это работает, перейдите по следующему URL, используя ваш браузер:

http://hostname/index.php?r=country%2Findex

Вы увидите таблицу, показывающую страны из таблицы БД. Вы можете сортировать, а также фильтровать данные, указывая условия фильтрации в заголовках столбцов.

Для каждой отображающейся в таблице страны вы можете просмотреть подробную информацию, обновить или удалить её. Вы также можете нажать на кнопку "Создать страну" в верхней части таблицы для получения формы создания новой страны.





Ниже приведен список файлов, созданных с помощью Gii, в том случае, если вы захотите исследовать реализацию этих функций, или изменить их:

* Контроллер: controllers/CountryController.php
* Модели: models/Country.php и models/CountrySearch.php
* Вид: views/country/\*.php

**Задание:**

1. Реализуйте все примеры, описанные выше. **Внимание!!!** Сохраните файлы, созданные вами раньше.
2. Интегрируйте в созданные gii файлы код, который был вами написан на предыдущем шаге (работа с формами). Убедитесь, что данные в таблицу можно добавлять с помощью сгенерированного действия create и с помощью вашего.

# Лабораторная работа №3. Аутентификация и авторизация.

## Аутентификация

Аутентификация — это процесс проверки подлинности пользователя. Обычно используется идентификатор (например, username или адрес электронной почты) и секретный токен (например, пароль или ключ доступа), чтобы судить о том, что пользователь именно тот, за кого себя выдаёт. Аутентификация является основной функцией формы входа.

Yii предоставляет фреймворк авторизации с различными компонентами, обеспечивающими процесс входа. Для использования этого фреймворка вам нужно проделать следующее:

* Настроить компонент приложения [user](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html);
* Создать класс, реализующий интерфейс [yii\web\IdentityInterface](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html).

### Настройка [yii\web\User](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html)

Компонент [user](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html) управляет статусом аутентификации пользователя. Он требует, чтобы вы указали [identity class](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html" \l "$identityClass-detail), который будет содержать текущую логику аутентификации. В следующей конфигурации приложения, [identity class](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html" \l "$identityClass-detail) для [user](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html) задан как app\models\User, реализация которого будет объяснена в следующем разделе:

return [

'components' => [

'user' => [

'identityClass' => 'app\models\User',

],

],

];

### Реализация [yii\web\IdentityInterface](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html)

[identity class](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html" \l "$identityClass-detail) должен реализовывать [yii\web\IdentityInterface](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html), который содержит следующие методы:

* [findIdentity()](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html" \l "findIdentity()-detail): Этот метод находит экземпляр identity class, используя ID пользователя. Этот метод используется, когда необходимо поддерживать состояние аутентификации через сессии.
* [findIdentityByAccessToken()](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html" \l "findIdentityByAccessToken()-detail): Этот метод находит экземпляр identity class, используя токен доступа. Метод используется, когда требуется аутентифицировать пользователя только по секретному токену (например в RESTful приложениях, не сохраняющих состояние между запросами).
* [getId()](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html" \l "getId()-detail): Этот метод возвращает ID пользователя, представленного данным экземпляром identity.
* [getAuthKey()](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html" \l "getAuthKey()-detail): Этот метод возвращает ключ, используемый для основанной на cookie аутентификации. Ключ сохраняется в аутентификационной cookie и позже сравнивается с версией, находящейся на сервере, чтобы удостоверится, что аутентификационная cookie верная.
* [validateAuthKey()](http://stuff.cebe.cc/yii2docs-ru/yii-web-identityinterface.html" \l "validateAuthKey()-detail): Этот метод реализует логику проверки ключа для основанной на cookie аутентификации.

В следующем примере, [identity class](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html" \l "$identityClass-detail) реализован как класс [Active Record](http://stuff.cebe.cc/yii2docs-ru/guide-db-active-record.html), связанный с таблицей user.

<?php

use yii\db\ActiveRecord;

use yii\web\IdentityInterface;

class User extends ActiveRecord implements IdentityInterface

{

public static function tableName()

{

return 'user';

}

/\*\*

\* Finds an identity by the given ID.

\*

\* @param string|int $id the ID to be looked for

\* @return IdentityInterface|null the identity object that matches the given ID.

\*/

public static function findIdentity($id)

{

return static::findOne($id);

}

/\*\*

\* Finds an identity by the given token.

\*

\* @param string $token the token to be looked for

\* @return IdentityInterface|null the identity object that matches the given token.

\*/

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

{

return static::findOne(['accessToken' => $token]);

}

/\*\*

\* @return int|string current user ID

\*/

public function getId()

{

return $this->id;

}

/\*\*

\* @return string current user auth key

\*/

public function getAuthKey()

{

return $this->authKey;

}

/\*\*

\* @param string $authKey

\* @return bool if auth key is valid for current user

\*/

public function validateAuthKey($authKey)

{

return $this->getAuthKey() === $authKey;

}

}

**Примечание:**Не путайте identity класс User с классом [yii\web\User](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html). Первый является классом, реализующим логику аутентификации пользователя. Он часто реализуется как класс [Active Record](http://stuff.cebe.cc/yii2docs-ru/guide-db-active-record.html), связанный с некоторым постоянным хранилищем, где лежит информация о пользователях. Второй — это класс компонента приложения, отвечающий за управление состоянием аутентификации пользователя.

### Использование [yii\web\User](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html)

В основном класс [yii\web\User](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html) используют как компонент приложения user.

Можно получить identity текущего пользователя, используя выражение Yii::$app->user->identity. Оно вернёт экземпляр [identity class](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html" \l "$identityClass-detail), представляющий текущего аутентифицированного пользователя, или null, если текущий пользователь не аутентифицирован (например, гость). Следующий код показывает, как получить другую связанную с аутентификацией информацию из [yii\web\User](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html):

// `identity` текущего пользователя. `Null`, если пользователь не аутентифицирован.

$identity = Yii::$app->user->identity;

// ID текущего пользователя. `Null`, если пользователь не аутентифицирован.

$id = Yii::$app->user->id;

// проверка на то, что текущий пользователь гость (не аутентифицирован)

$isGuest = Yii::$app->user->isGuest;

Для залогинивания пользователя вы можете использовать следующий код:

// найти identity с указанным username.

// замечание: также вы можете проверить и пароль, если это нужно

$identity = User::findOne(['username' => $username]);

// логиним пользователя

Yii::$app->user->login($identity);

Для выхода пользователя, просто вызовите

Yii::$app->user->logout();

**Задание:**

1. Используя приведенный выше **identity class** User**,** измените класс User из базового пакета Yii (basic\models\User) так, чтобы авторизация осуществлялась через проверку логина и пароля из таблицы user.

Дамп таблицы – файл user.sql

## Авторизация

Авторизация — это процесс проверки того, что пользователь имеет достаточно прав, чтобы выполнить какие-то действия. Yii предоставляет два метода авторизации: фильтры контроля доступа (ACF) и контроль доступа на основе ролей (RBAC).

### Фильтры контроля доступа

Фильтры контроля доступа (ACF) являются простым методом, который лучше всего использовать в приложениях с простым контролем доступа. Как видно из их названия, ACF — это фильтры, которые могут присоединяться к контроллеру или модулю как поведение. ACF проверяет набор [правил доступа](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accesscontrol.html" \l "$rules-detail), чтобы убедиться, что пользователь имеет доступ к запрошенному действию.

Код ниже показывает, как использовать ACF фильтр, реализованный в [yii\filters\AccessControl](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accesscontrol.html):

use yii\filters\AccessControl;

class SiteController extends Controller

{

public function behaviors()

{

return [

'access' => [

'class' => AccessControl::className(),

'only' => ['login', 'logout', 'signup'],

'rules' => [

[

'allow' => true,

'actions' => ['login', 'signup'],

'roles' => ['?'],

],

[

'allow' => true,

'actions' => ['logout'],

'roles' => ['@'],

],

],

],

];

}

// ...

}

Код выше показывает ACF фильтр, связанный с контроллером site через поведение. Это типичный способ использования фильтров действий. Параметр only указывает, что фильтр ACF нужно применять только к действиям login, logout и signup. Параметр rules задаёт [правила доступа](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html), которые означают следующее:

* Разрешить всем гостям (ещё не прошедшим авторизацию) доступ к действиям login и signup. Опция roles содержит знак вопроса ?, это специальный токен обозначающий "гостя".
* Разрешить аутентифицированным пользователям доступ к действию logout. Символ @ — это другой специальный токен, обозначающий аутентифицированного пользователя.

Когда фильтр ACF проводит проверку авторизации, он проверяет правила по одному сверху вниз, пока не найдёт совпадение. Значение опции allow выбранного правила указывает, авторизовывать пользователя или нет. Если ни одно из правил не совпало, то пользователь считается НЕавторизованным, и фильтр ACF останавливает дальнейшее выполнение действия.

По умолчанию, когда у пользователя отсутствует доступ к текущему действию, ACF делает следующее:

* Если пользователь гость, вызывается [yii\web\User::loginRequired()](http://stuff.cebe.cc/yii2docs-ru/yii-web-user.html" \l "loginRequired()-detail), который перенаправляет браузер на страницу входа.
* Если пользователь авторизован, генерируется исключение [yii\web\ForbiddenHttpException](http://stuff.cebe.cc/yii2docs-ru/yii-web-forbiddenhttpexception.html).

[Правила доступа](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html) поддерживают набор свойств. Ниже дано краткое описание поддерживаемых опций. Вы также можете расширить [yii\filters\AccessRule](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html), чтобы создать свой собственный класс правил доступа.

* [allow](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html" \l "$allow-detail): задаёт какое это правило, "allow" или "deny".
* [actions](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html" \l "$actions-detail): задаёт действия, соответствующие этому правилу. Значение должно быть массивом идентификаторов действий. Сравнение — регистрозависимо. Если свойство пустое или не задано, то правило применяется ко всем действиям.
* [controllers](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html" \l "$controllers-detail): задаёт контроллеры, которым соответствует правило. Значение должно быть массивом с идентификаторами контроллеров. Сравнение регистрозависимо. Если свойство пустое или не задано, то правило применяется ко всем контроллерам.
* [roles](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html" \l "$roles-detail): задаёт роли пользователей, соответствующих этому правилу. Распознаются две специальные роли, которые проверяются с помощью yii\web\User::isGuest:
  + ?: соответствует гостевому пользователю (не аутентифицирован),
  + @: соответствует аутентифицированному пользователю.
* [matchCallback](http://stuff.cebe.cc/yii2docs-ru/yii-filters-accessrule.html" \l "$matchCallback-detail): задаёт PHP колбек, который вызывается для определения, что правило должно быть применено.

Ниже показан пример, показывающий использование опции matchCallback, которая позволяет писать произвольную логику проверки доступа:

use yii\filters\AccessControl;

class SiteController extends Controller

{

public function behaviors()

{

return [

'access' => [

'class' => AccessControl::className(),

'only' => ['special-callback'],

'rules' => [

[

'actions' => ['special-callback'],

'allow' => true,

'matchCallback' => function ($rule, $action) {

return date('d-m') === '31-10';

}

],

],

],

];

}

// Колбек сработал! Эта страница может быть отображена только 31-ого октября

public function actionSpecialCallback()

{

return $this->render('happy-halloween');

}

}

**Задание:**

1. Используя фильтры контроля доступа, разграничите права доступа к контроллеру CountryController по следующим правилам:
   1. Гость может просматривать список стран
   2. Зарегистрированный пользователь может добавлять страны в таблицу
   3. Администратор (пользователь с логином admin) может редактировать и удалять записи.

# Лабораторная работа №4. Итоговая работа по Yii2.

Используя полученные знания, реализуйте информационную систему.

### Общие требования:

1. Использование БД для хранения данных.
2. Присутствие не менее 3 ролей пользователей, доступ к соответствующим действиям организован с помощью фильтров контроля доступа (или RBAC).
3. Использование не стандартного шаблона.

### Полезные ресурсы:

* Русскоязычный официальный гайд <http://stuff.cebe.cc/yii2docs-ru/guide-intro-yii.html>
* Русскоязычное сообщество [http://www.yiiframework.ru](http://www.yiiframework.ru/)

### Варианты для разработки ИС:

В соответствии с темой курсового проекта