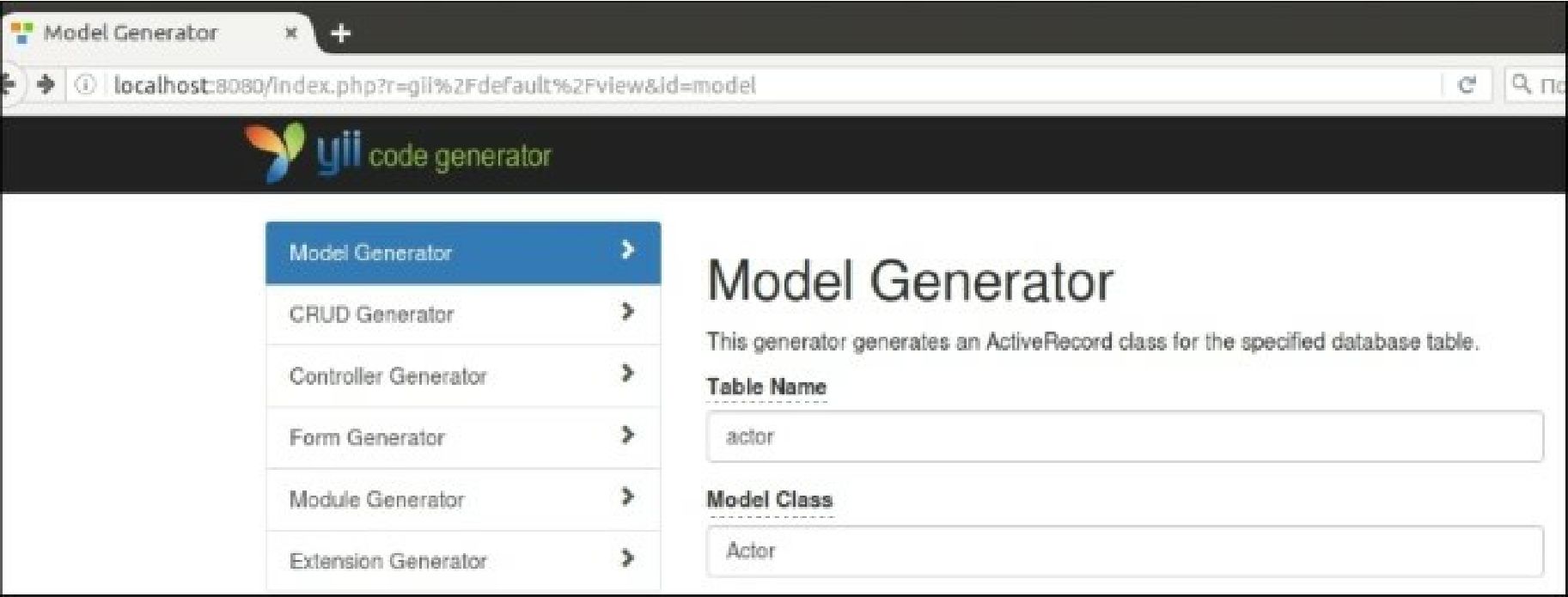
Генерация кода

Yii2 обеспечивает мощный модуль Gii для генерации моделей, контроллеров и представлений, которые вы можете легко изменять и настраивать. Это действительно полезный инструмент для быстрого развития.

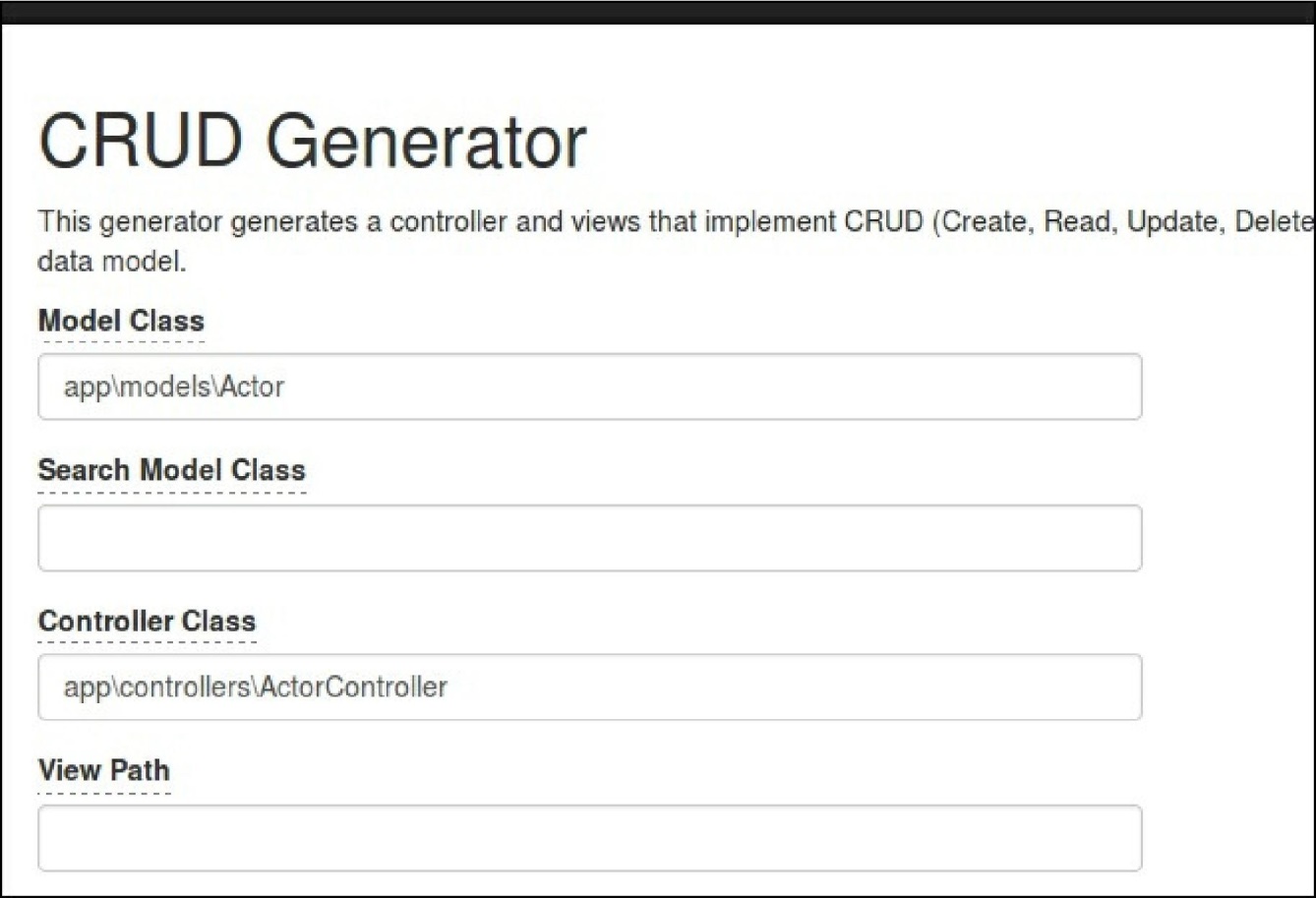
В этом разделе мы рассмотрим, как использовать Gii и генерировать код. Например, у вас есть база данных с одной таблицей film, и вы хотели бы создать приложение с операциями CRUD для этой таблицы. Это просто.

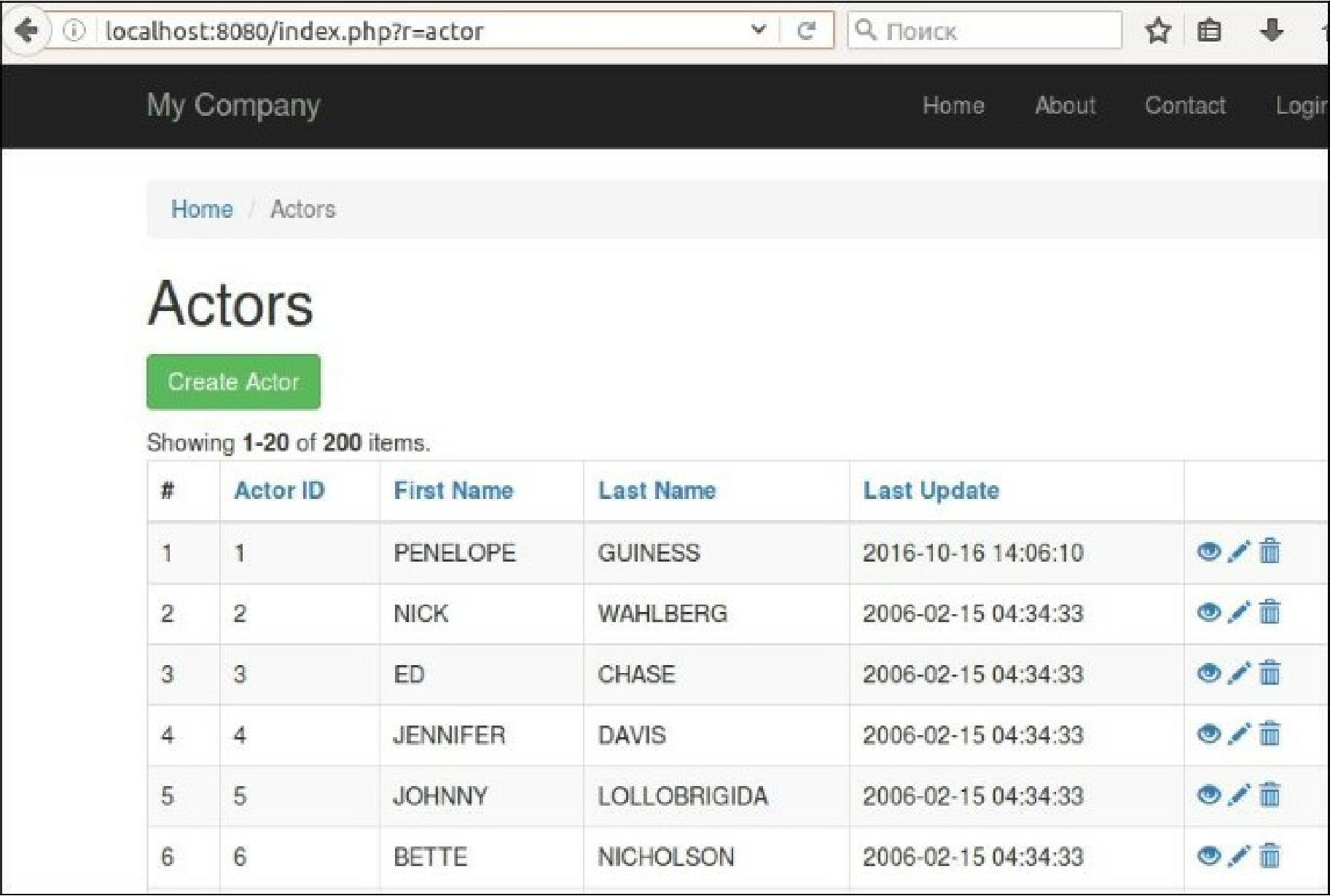


Подготовка

1. Создайте новое приложение с помощью composer, как описано в официальном руководстве по <http://www.yiiframework.com/doc-2.0/guide-start-installation.html>. (от переводчика по русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-installation> )
2. Скачать база данных "sakila" c [http://dev.mysql.com/doc/indey-other.html](http://dev.mysql.com/doc/index-other.html).
3. Выполните загрузку SQL: сначала схема, затем данные.
4. Настройте подключение к базе данных в config/main.php для использования базы данных Sakila.
5. 5. Запустите веб-сервер с помощью./yii serve. (от переводчика: или другим способом)

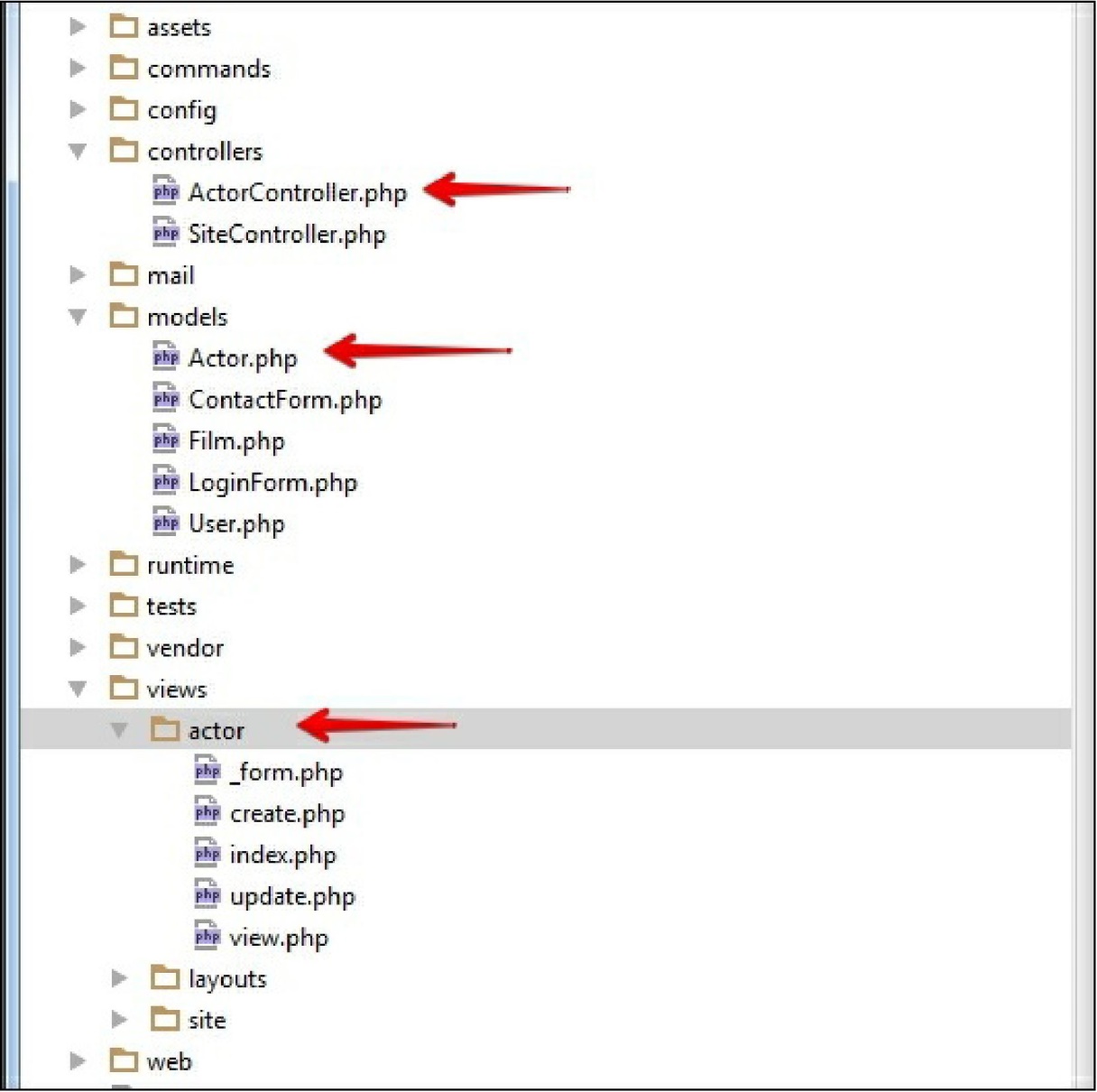
Как это сделать...

1. Перейдите по адресу http: //localhost :8080/index. php?r=gii и выберите Model Generator.
2. Заполните Table Name как actor и Model Class как Actor и нажмите кнопку Generate внизу страницы
3. Вернуться к главному меню GII, нажав на логотип в заголовке и выбрав CRUD Generator.
4. Заполнит поле Model Class как app\models\Actor и Controller Class как app\controllers\ActorController.
5.  Нажмите кнопку Preview в нижней части страницы, а затем нажмите зеленую кнопку Generate.
6. Проверьте результат по адресу http://localhost:8080/index.php?actor/create.



Как это работает…

Если вы проверите структуру проекта, вы увидите автоматически созданный код:



Сначала мы создали модель Actor. Gii автоматически создает все модельные правила, которые зависят от типов полей mysql. Например, если в Вашей таблице MySQL actor поля first\_name и last\_name имеют флаг NOT NULL, то Yii автоматически создает правило для него required и устанавливает максимальную длину 45 символов, потому что в нашей базе данных Максимальная длина этого поля установлена как 45

.

public function rules()

{

return [

[['first\_name', 'last\_name'], 'required'],

[['last\_update'], 'safe'],

[['first\_name', 'last\_name'], 'string', 'max' => 45],

}

];

Кроме того, yii автоматически создает отношения между моделями на основе внешних ключей, добавленных в базу данных. В нашем случае два отношения были созданы автоматически

.

public function getFilmActors()

{

return $this->hasMany(FilmActor::className(), ['actor\_id' => 'actor\_id']);

}

public function getFilms()

{

return $this->hasMany(Film::className(), ['film\_id' => 'film\_id'])-

>viaTable('film\_actor', ['actor\_id' => 'actor\_id']);

}

Это отношение было создано, потому что у нас есть два внешних ключа в нашей базе данных. Таблица film\_actor имеет внешний ключ fk\_film\_actor\_actor, который указывает на поля таблицы Actor actor\_id и fk\_film\_actor\_film что указывает на таблицу Film поле film\_id.

Обратите внимание, что модель FilmActor еще не создана. Поэтому, если вы будете разрабатывать полное приложение , вы должны будете генерировать также модели Film, FilmActor .

Для остальных частей обратитесь к <http://www.yiiframework.com/doc-2.0/guide-start-gii.html>. ( по русски <http://yiiframework.domain-na.me/doc/guide/2.0/ru/start-gii> )