Знакомство с фреймворком

Содержание

Практическая работа 1: Установка и настройка Yii2 вручную

Цель: Научиться устанавливать и настраивать Yii2 без использования Composer.

Шаги:

1. Скачайте архив с Yii2 с официального сайта (https://www.yiiframework.com/download).
2. Распакуйте архив в нужную директорию.
3. Настройте базовую конфигурацию:
    a. Откройте файл config/web.php и ознакомьтесь с основными настройками.
4. Запиши в тетрадь структуру файлов:

```
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
```

Практическая работа 2: Создание контроллеров и работа с шаблонами

Цель: Научиться создавать контроллеры, работать с шаблонами и передавать данные в контекст.

Создание контроллера

В директории controllers создайте файл OtherController.php.

Определите класс OtherController, который наследуется от yii\web\Controller.

```php
<?php
namespace app\controllers;

use yii\web\Controller;

0 references | 0 implementations
class OtherController extends Controller
{
    0 references | 0 overrides
    public function actionIndex(): mixed
    {
        // Передача данных в контекст
        $data = [
            'title' => 'Добро пожаловать в Yii2!',
            'message' => 'Введите своё ФИО и номер группы, а также свою подгруппу',
        ];

        return $this->render('index', $data);
    }

    0 references | 0 overrides
    public function actionAbout(): mixed
    {
        // Передача данных в контекст
        $data = [
            'title' => 'Информация обо мне',
            'content' => 'Введите своё ФИО, Возраст, Город проживания, Место обучения',
        ];

        return $this->render('about', $data);
    }
}

```

Создание видов

Создайте вид для отображения главной страницы:

В директории views/other создайте файл index.php.

Настройте вид для отображения данных, переданных из контроллера.

```php
<?php
/* @var $this yii\web\View */
/* @var $title string */
/* @var $message string */
?>
<div class="site-index">
    <h1><?= \yii\helpers\Html::encode($title) ?></h1>
    <p><?= \yii\helpers\Html::encode($message) ?></p>
</div>

```

Создайте вид для отображения страницы "About":

В директории views/other создайте файл about.php.

Настройте вид для отображения данных, переданных из контроллера.

```php
<?php
/* @var $this yii\web\View */
/* @var $title string */
/* @var $content string */
?>
<div class="site-about">
    <h1><?= \yii\helpers\Html::encode($title) ?></h1>
    <p><?= \yii\helpers\Html::encode($content) ?></p>
</div>

```

Настройка маршрутизации

Настройте маршрутизацию в файле config/web.php:

Добавьте маршруты для доступа к действиям контроллера.

```
'urlManager' => [
    'enablePrettyUrl' => true,
    'showScriptName' => false,
    'rules' => [
        '' => 'other/index',
        'about' => 'other/about',
    ],
],
```

Работа с формами

Создание модели

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

```php
<?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, 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(). В случае неудачной проверки свойство hasErrors станет равным true. С помощью errors можно узнать, какие именно ошибки возникли.

Создание действия

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

```
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]);
        }
    }
}
```

Не забываем импортировать EntryForm!

Действие создает объект EntryForm. Затем оно пытается заполнить модель данными из массива $_POST, доступ к которому обеспечивает Yii при помощи yii\web\Request::post(). Если модель успешно заполнена, то есть

пользователь отправил данные из HTML-формы, то для проверки данных будет вызван метод validate().

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.

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 ActiveForm. Методы begin() и end() выводят открывающий и закрывающий теги формы. Между этими вызовами создаются поля ввода при помощи метода field(). Первым идёт поле для "name", вторым — для "email". Далее для генерации кнопки отправки данных вызывается метод yii\helpers\Html::submitButton().

Попробуем

Переходим по: site/entry

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 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') ?>
```

Работа с базами данных

Этот пункт 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, определить action и создать view.

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 «ваша_группа_подгруппа_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 «ваша_группа_подгруппа_yii2basic» и внутри неё таблица country с тремя столбцами, содержащими десять строк данных.

Настраиваем подключение к БД

Перед продолжением убедитесь, что у вас установлены PHP-расширение PDO и драйвер PDO для используемой вами базы данных (например, pdo_mysql для MySQL). Это базовое требование в случае использования вашим приложением реляционной базы данных. После того, как они установлены, 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 инструмент, базирующийся на файлах. Данный конфигурационный файл определяет параметры, необходимые для создания и инициализации экземпляра yii\db\Connection, 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.

Создаём потомка Active Record

Чтобы представлять и получать данные из таблицы country, создайте класс — потомок Active Record, под названием Country и сохраните его в файле models/Country.php.

```php
<?php

namespace app\models;

use yii\db\ActiveRecord;

class Country extends ActiveRecord
{
}
```

Класс Country наследуется от yii\db\ActiveRecord. 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 прямую зависимость между именем таблицы и именем класса, вы можете переопределить метод yii\db\ActiveRecord::tableName(), чтобы явно задать имя связанной таблицы.

Создаём Action

Для того, чтобы показать данные по странам конечным пользователям, вам надо создать новый action. Вместо размещения нового action'а 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
<?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. 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 с 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->code} ({$country->name})") ?>:
        <?= $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, 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.

Попробуем

Переходим по: country/index

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:

country/index&page=2

За кадром Pagination предоставляет всю необходимую функциональность для постраничной разбивки набора данных:

- В начале Pagination 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 выводит кнопки страниц используя URL'ы, созданные Pagination. 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.

Самостоятельная работа

ВСЕ ЗАДАНИЯ ВЫПОЛНЯЮТСЯ В НОВОМ КОНТРОЛЛЕРОМ!

В каждом должны быть настроены URL (смотреть ПР – 2!)

Задание 1

По аналогии «Работа с формой», создайте:

Форму: FeedbackForm, в котором есть поля:

- username – обязательное поле
- email – обязательное поле, тип email
- message – обязательное поле, тип текст.

Создайте Действие, Создайте Представление

Задание 2

По аналогии «Работа с базой данных», создайте:

Таблицу product, создайте новый контроллер с названием: ProductsController, выведите все данные на экран с пагинацией.

Домашнее задание

ВСЕ ЗАДАНИЯ ВЫПОЛНЯЮТСЯ В НОВОМ КОНТРОЛЛЕРОМ!

В каждом должны быть настроены URL (смотреть ПР – 2!)

Задание 1

По аналогии «Работа с формой», создайте:

Форму: RegisterForm, в котором есть поля:

- username – обязательное поле, тип текст
- email – обязательное поле, тип email
- age – обязательное поле, тип число
- password – обязательное поле, тип текст

Создайте Действие, Создайте Представление

Задание 2

По аналогии «Работа с базой данных», создайте:

Таблицу book, создайте новый контроллер с названием: BooksController, выведите все данные на экран с пагинацией.

Таблица book:

```
CREATE TABLE `books` (
  `id` INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `title` VARCHAR(255) NOT NULL,
  `author` VARCHAR(255) NOT NULL,
  `published_year` INT(4) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 1', 'Author 1', 2001);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 2', 'Author 2', 2002);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 3', 'Author 3', 2003);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 4', 'Author 4', 2004);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 5', 'Author 5', 2005);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 6', 'Author 6', 2006);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 7', 'Author 7', 2007);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 8', 'Author 8', 2008);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 9', 'Author 9', 2009);
INSERT INTO `books` (`title`, `author`, `published_year`) VALUES ('Book 10', 'Author 10', 2010);
```

Видео о том, как загружать работу на хостинг:

https://rutube.ru/video/private/15ccedf099cdef1a4a444f1dac1aae74/?p=_G9xZrenOyK6WYNEBGA1bA

Создаёте 1 сайт, и там будут папки с вашими проектами!

Название сайта: yii-33-1-ваш-логин.kpk45.ru

Название проекта: pr-1