Зміст

ВСТУП

Під час вивчення курсу студенти отримають глибинні знання зі швидкої розробки веб-сайтів та використання можливості сучасного PHP-фреймворку Yii2. Курс орієнтований на студентів, що володіють знаннями мови PHP, ООП, мають уявлення про MVC.

Мета вивчення дисципліни – засвоєння студентами розширених знань про мову програмування PHP та про структуру сучасного PHP-фреймворку, його функціональні можливості та повноцінне використання при розробці веб-сайтів

Ціль дисципліни – набути навичок розробки, тестування та розгортання веб-додатків засобами PHP-фреймворку Yii2.

Завданням вивчення навчальної дисципліни «Інтернет-програмування: PHP» є теоретична та практична підготовка майбутніх фахівців із таких питань:

- Особливості PHP-фреймворку. Загальна компонентна структура веб-застосунку. MVC-схема.
- Цикл обробки запитів. Маршрутизація.
- Контролери.
- Моделі даних. Моделі форм. Валідація даних.
- Механізми роботи з БД.
- Представлення. Використання віджетів у представленнях.
- Особливості обробки асинхронних запитів.
- Завантаження файлів.
- Автентифікація та авторизація, механізм RBAC.
- Кешування даних.
- Конфігурування пошти, створення та відправка поштових повідомлень
- Огляд підходів щодо організації пошуку
- Розробка REST API

Практичні навички, що отримає студент після курсу:

Розробка, тестування та впровадження веб-додатків з використанням PHP-фреймворку, робота з Composer (менеджером залежностей для PHP), розробка

моделей, валідація даних, робота з базою даних, налаштування маршрутизації, сервісу електронної пошти, реалізація автентифікації, розробка системи ролей тощо

Результатом вивчення дисципліни є набуття студентами таких *компетенцій*:

- Знати і системно застосовувати методи аналізу та моделювання прикладної області, виявлення інформаційних потреб і збору вихідних даних для проектування програмного забезпечення.
- Знати і застосовувати базові концепції і методології моделювання інформаційних процесів.
- Оцінювати і вибирати методи і моделі розробки, впровадження, експлуатації програмних засобів та управління ними на всіх етапах життєвого циклу.
- Розробляти і оцінювати стратегії проектування програмних засобів; обґрунтовувати, аналізувати і оцінювати прийняті проектні рішення з точки зору якості кінцевого програмного продукту.
- Аналізувати, оцінювати і вибирати методи, сучасні програмно-апаратні інструментальні та обчислювальні засоби, технології, алгоритмічні та програмні рішення для ефективного виконання конкретних виробничих задач з програмної інженерії.
- Обґрунтовано вибирати парадигми і мови програмування для вирішення прикладних завдань; застосовувати на практиці системні та спеціалізовані засоби, компонентні технології (платформи) та інтегровані середовища розробки програмного забезпечення.

Лабораторна робота №1. Знайомство з Yii2. Встановлення фреймворку. Розробка першої сторінки. Робота з формами. Початок роботи з БД

Мета: отримати знання та практичні навички щодо встановлення фреймворку, компонентної структури та циклу обробки запитів. Навчитись працювати з формами та БД.

Обладнання: ПК, OpenServer, Yii2

Завдання

1. Створити 2 проекти з фреймворком Yii2 (Basic та Advanced відповідно)
2. Розробити сторінку з привітанням
3. Розробити форму для входу
4. Налаштувати підключення до БД, розробити сторінку для виводу даних з БД

Теоретичні відомості

Фреймворк – (каркас, структура) – заготовки, шаблони, що визначають структуру додатку. *Фреймворк Yii2*:

- використовує MVC-архітектуру
- Призначений для швидкої розробки складних веб-додатків: порталів, інтернет-магазинів, соціальних мереж

Інструменти Yii2:

- Форми і валідація даних
- Робота з базою даних: DAO, Query Builder, Active Record
- Генерація коду, інструмент Gii
- Маршрутизація запитів
- Відображення даних: посторінкова навігація, провайдери даних, віджети
- Автентифікація та авторизація
- Консольні додатки
- Обробка асинхронних запитів, інструмент Pjax

- Завантаження файлів
- Автентифікація та авторизація, механізм RBAC
- Кешування даних
- Конфігурування mailer. Створення та відправка поштових повідомлень
- Організація пошуку
- REST API

Посилання:

- Офіційний сайт: *http://www.yiiframework.com/*
- Повне керівництво: *http://www.yiiframework.com/doc-2.0/guide-index.html*
- Українська спільнота: *https://yiiframework.com.ua/uk/*

Підготовчий етап

- ✓ Встановіть та налаштуйте локальний сервер. В методичних рекомендаціях використовуватиметься OpenServer
- ✓ Створіть бази даних *yii_basic* та *yii_advanced* для двох проектів відповідно

Хід роботи

Завдання 1. Встановлення Yii2

Встановіть фреймворк з двома шаблонами *Basic Project Template* та *Advanced Project Template* в два різні проекти

Встановити Yii2 можна двома способами:

- ✓ Завантажити архів Yii2
- ✓ Використати Composer (менеджер розширень для PHP)

Для встановлення Yii2 потрібно вибрати один із шаблонів проекту. Існує два шаблони проекту:

- Базовий шаблон проекту (Basic Project Template)
- Розширений шаблон проекту (Advanced Project Template)

Базовий шаблон проекту (yii2-app-basic)

- Містить деякі базові можливості: логін, форма зворотного зв'язку тощо
- Зручний для початківців
- Може бути використаний для невеликих проектів

Розширений шаблон проекту (yii2-app-advanced)

- Розділений на дві частини: користувацька (frontend) та адміністративна (backend)
- Використовується в командній розробці
- Призначений для великих проектів

Встановлення Yii2 з базовим шаблоном проекту

✓ Запустити командний рядок

✓ Перейти в папку *domains* з локальними проектами:

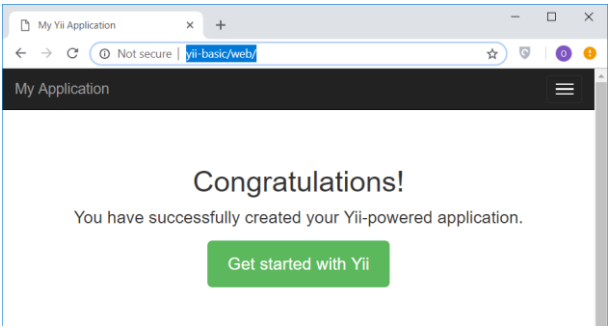
```
cd domains
```

✓ Виконати команду для встановлення Yii2 в папку *yii-basic*:

```
composer create-project --prefer-dist yiisoft/yii2-app-basic yii-basic
```

Дана команда встановить останню стабільну версію Yii2 в директорію *yii-basic*

Після встановлення додаток буде доступний по URL: *http://yii-basic/web/*



Встановлення Yii2 з розширеним шаблоном проекту

✓ Виконати команду для встановлення Yii2 в папку *yii-advanced*:

```
composer create-project --prefer-dist yiisoft/yii2-app-advanced yii-advanced
```

- ✓ Після встановлення додатку, його потрібно ініціалізувати. Для цього потрібно перейти в папку з проектом та виконати команду ініціалізації проекту:

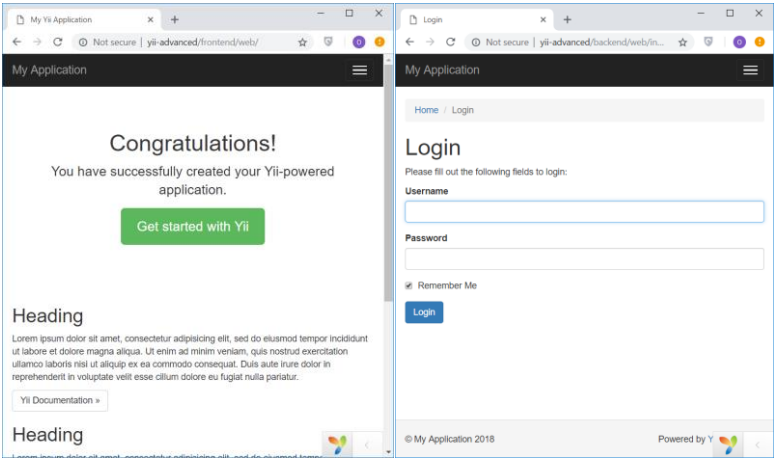
```
cd yii-advanced
php init
```

- ✓ Вибрати та підтвердити режим оточення *Development*.

Примітка. В режимі розробника буде доступна панель відлагоджувача внизу сторінок, а також відображатимуться можливі помилки.

Після встановлення розширеного шаблону проекту Yii2, доступ до сайту здійснюється за посиланнями:

- o Клієнтська частина *http://yii-advanced/frontend/web/*
- o Адміністративна частина *http://yii-advanced/backend/web/*



Завдання 2. Hello, World!

Створимо сторінку з текстом *"Hello, World!"*. В даному прикладі ми навчимось:

- ✓ створювати *action* (метод дії контролера для обробки запиту)
- ✓ створювати шаблон представлення (вигляд)
- ✓ як надсилати дані запиту в вигляд

Створення методу дії

Для роботи з діями, їх потрібно створити: визначити ідентифікатор дії та реалізувати для неї відповідний метод у контролері. Всі ідентифікатори дій задаються в нижньому регістрі. Декілька слів у ідентифікаторі повинні розділятись символом "-". Всі методи дій оголошуються в контролерах. Для задання імені методу дії видаляються всі дефіси, перші літери слів перетворюються в верхній регістр і додається префікс *action*. Наведемо приклади дій та імен відповідних методів:

Дія	*Ім'я методу дії*
index	actionIndex
create	actionCreate
create-comment	actionCreateComment

1. Створимо метод дії (*action*) в контролері. Для цього в контролері *site (controllers/SiteController.php)* додайте метод дії *say*:

Лістинг 1.1. Метод дії для обробки запиту на сторінку з привітанням

```
public function actionSay($message = "Hello")
{
    return $this->render('say', ['message' => $message]);
}
```

Пояснення до коду. Безпосередньо вивід здійснюється викликом методу `render`. В нього передаємо ім'я шаблону *say*, а також масив з елементом *message* для його подальшого відображення на сторінці. Метод потрібно вернути оператором `return`.

2. Створимо представлення. Для зберігання файлів представлення, використовуємо шлях *views/ControllerID/ViewName.php*. Наприклад, для шаблону *say* контролера *site*, даний файл повинен міститись в папці *views/site/say.php*

Лістинг 1.2. Метод дії для обробки запиту на сторінку з привітанням

```
<?php
use yii\helpers\Html;
?>

<h2><?= Html::encode($message); ?></h2>
```

3. Перевіримо роботу створеної сторінки. Для цього потрібно виконати запит:

http://yiibasic.dev/web/index.php?r=site/say&message=Hello,+world!

Пояснення. Аналогічно до ідентифікаторів дій, контролери також мають ідентифікатори. Імена класів отримуються або задаються за таким правилом. Ім'я класу містить ідентифікатор дії, приведений в регістр CamelCase. В кінець імені класу додається слово Controller. Наприклад, ідентифікатору контролера post-comment відповідає клас контролера PostCommentController.

Завдання 3. Створити сторінку з формою

Створимо сторінку, що міститиме форму з полями для введення логіну та email. В даному прикладі ми навчимось:

- ✓ створювати модель для даних, введених користувачем
- ✓ оголошувати правила для перевірки введених даних
- ✓ створювати html-форми

1. Створимо модель форми. Створіть та опишіть клас моделі форми *EntryForm* в файлі *models/EntryForm.php*:

Лістинг 1.3. Приклад моделі для форми входу (*models/EntryForm.php*)

```
<?php

namespace app\models;

use Yii;
use yii\base\Model;

class EntryForm extends Model
{
    public $name;
    public $email;

    public function rules()
    {
        return [
            [['name', 'email'], 'required', 'message' => 'Must be
required'],
            ['email', 'email', 'message' => 'Wrong Email'],
        ];
    }
}
```

2. Створимо дію ***entry*** в контролері ***site***. Для використання моделі форми потрібно імпортувати клас `EntryForm` з простору імен `app\models`:

Лістинг 1.4. Приклад методу дії в контролері для роботи формою

```
public function actionEntry()
{
    $model = new EntryForm();

    if ($model->load(Yii::$app->request->post())) {
        // Якщо дані передані в модель з post-запиту
        if ($model->validate()) {

            // можна обробити ці дані і зберегти в БД ...

            // генеруємо повідомлення з успіхом виконаної операції
            Yii::$app->session->setFlash("success", "Data received!");

            // відображаємо шаблон entry-confirm з даними моделі
            return $this->render('entry-confirm', ['model' => $model]);
        }
    }
    // відображаємо шаблон entry з формою
    return $this->render('entry', ['model' => $model]);
}
```

3. Створимо шаблони сторінок *entry.php* і *entry-confirm.php*. Шаблон *entry.php* генерує сторінку з формою:

Лістинг 1.5. Приклад шаблону з формою

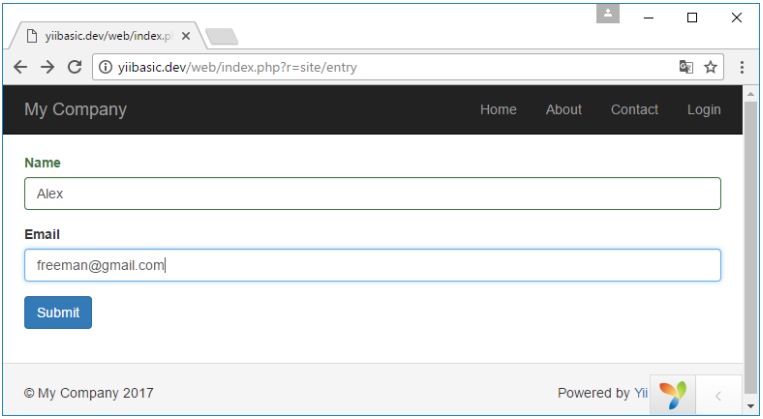
```
<?php
use yii\helpers\Html;
use yii\widgets\ActiveForm;
?>

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

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

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

<div class="form-group">
    <?= Html::submitButton('Submit', ['class' => 'btn btn-primary']) ?>
</div>

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

Другий шаблон *entry-confirm.php* генерує сторінку для відображення результату відправки форми:

Лістинг 1.6. Приклад шаблону з підтвердженням відправлених даних

```
<?php
use yii\helpers\Html;
?>
<p>You have entered the following information:</p>

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

4. Протестуйте роботу форми, клієнтську та серверну валідацію даних:



Завдання 4. Створити сторінку з виводом даних, отриманих з БД

1. Переконайтесь у наявності бази даних *yii_basic*
2. Створіть таблицю *country* з даними:

```
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 ('UA','Ukraine',42500000);
INSERT INTO `country` VALUES ('US','United States',322976000);
```

3. Налаштуйте підключення до БД в файлі *config/db.php*. Робота з БД доступна в коді через компоненту *db*: `Yii::$app->db`

4. Для роботи з даними таблиці *country*, створимо клас `Country`, що наслідується від класу `ActiveRecord` (клас для роботи з таблицями БД). Збережемо його в файлі *models/Country.php*:

Лістинг 1.7. Модель Country.php

```
<?php

namespace app\models;

use yii\db\ActiveRecord;

class Country extends ActiveRecord
{
}
```

5. Забезпечимо обробку запиту на сторінку зі списком країн. Для цього створіть контролер *country* та дію *index* в ньому:

Лістинг 1.8. Контролер CountryController.php

```
<?php

namespace app\controllers;

use yii\web\Controller;
use yii\data\Pagination;
use app\models\Country;

class CountryController extends Controller
{
    public function actionIndex()
```

```
    {
        //Створюємо запит
        $query = Country::find();

        //Запезпечимо посторінкове відображення даних
        $pagination = new Pagination([
            'defaultPageSize' => 5,
            'totalCount' => $query->count(),
        ]);

        //Формуємо та виконуємо запит
        $countries = $query->orderBy('name')
            ->offset($pagination->offset)
            ->limit($pagination->limit)
            ->all();

        //Відображаємо дані в шаблоні представлення
        return $this->render('index', [
            'countries' => $countries,
            'pagination' => $pagination,
        ]);
    }
}
```

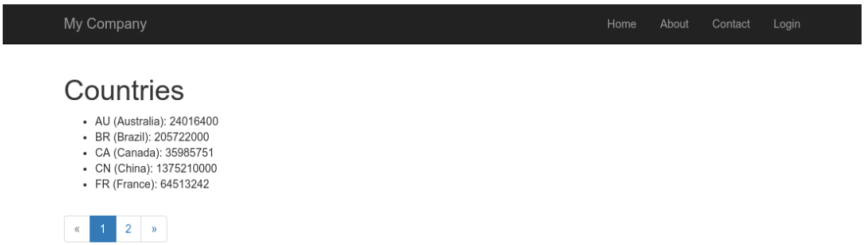
6. Створюємо шаблон представлення:

Лістинг 1.9. Шаблон представлення *views/country/index.php*

```
<?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]) ?>
```

7. Перевірте роботу сторінки, виконавши запит у браузері:

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



Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Лабораторна робота №2. Робота з формами. Робота з БД. Генерація коду з Gii. Реєстрація користувачів

Мета: закріпити знання та практичні навички роботи з формами та БД. Отримати навички роботи з автоматичною генерацією коду. Протестувати механізм реєстрації та аутентифікації користувачів.

Обладнання: ПК, OpenServer, Yii2

Завдання

Реалізувати форму з валідацією полів. Здійснити вивід даних, отриманих з таблиці БД, з посторінковою навігацією. Згенерувати CRUD-код. Протестувати механізм реєстрації та автентифікації користувачів. Додати два власні поля до існуючих даних користувача.

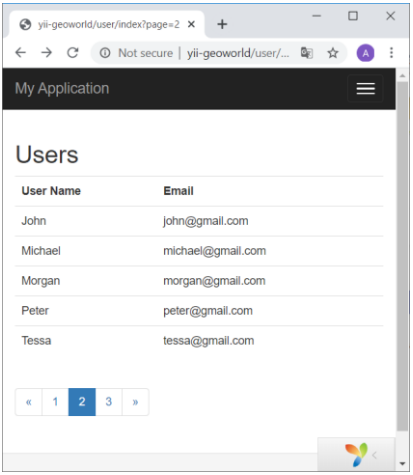
Підготовчий етап

- ✓ Встановіть та налаштуйте локальний сервер. В методичних рекомендаціях використовуватиметься *OpenServer*;
- ✓ Переконайтесь в наявності двох проектів *yii-basic*, *yii-advanced* та двох БД *yii_basic* та *yii_advanced* відповідно.

Хід роботи

Завдання 1. Таблиця користувачів

Виведіть дані таблиці *user* на сторінці за таким зразком:



1. Створіть та наповніть таблицю *user*

Структура таблиці:

Имя	Тип	Сравнение
user_id	int(11)	
username	varchar(100)	utf8_general_ci
email	varchar(100)	utf8_general_ci

Дані таблиці:

user_id	username	email
1	Jane	jane@gmail.com
2	John	john@gmail.com
3	Peter	peter@gmail.com
4	Michael	michael@gmail.com
5	Alex	alex@gmail.com
6	Vanessa	vanessa@gmail.com
7	Joahn	joahn@gmail.com
8	Morgan	morgan@gmail.com
9	Tessa	tessa@gmail.com
10	Harry	harry@gmail.com
11	Jim	jim@gmail.com

2. Створіть модель *User*, що наслідується від класу *ActiveRecord*

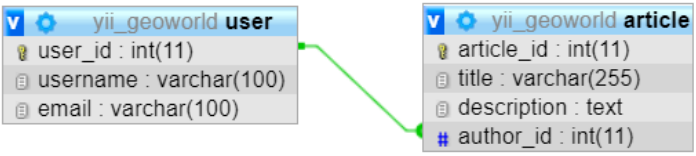
3. Створіть контролер *user* та в ньому реалізуйте метод дії *index*. В цьому методі отримайте дані з таблиці *user*, сформуйте дані посторінкової навігації та передайте у шаблон вигляду *index*.
4. Створіть шаблон *index*. Подумайте, в якій папці він повинен бути. В цьому шаблоні створіть розмітку за зразком, наведеним вище в цьому завданні.

Завдання 2. Генерація CRUD-коду

1. Створіть таблицю *article*. Структура, дані та зв'язки наведені на наступних зображеннях:

Имя	Тип	Сравнение
article_id	int(11)	
title	varchar(255)	utf8_general_ci
description	text	utf8_general_ci
author_id	int(11)	

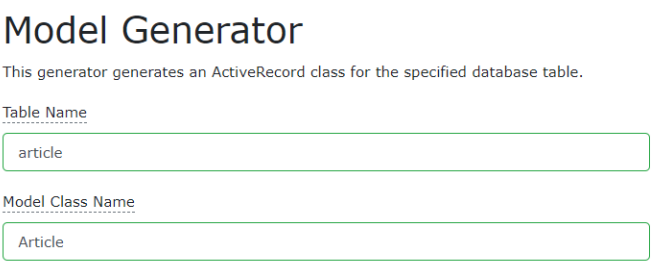
article_id	title	description	author_id
1	Desert Rose	A sun-loving plant	1
2	Bamboos	are evergreen perennial flowering plants	2



2. Згенеруємо модель Article для таблиці. Для цього

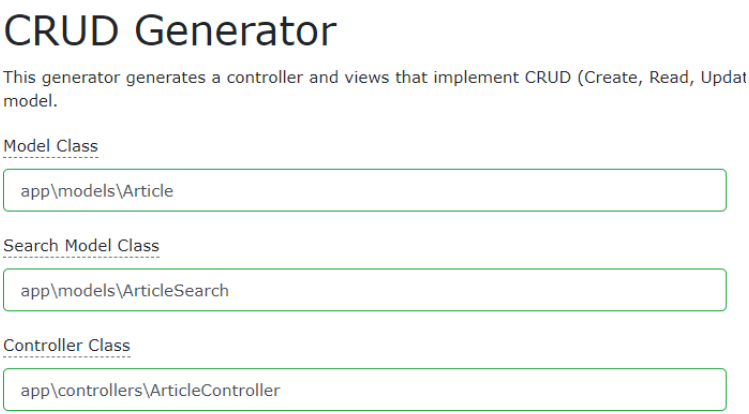
✓ Запустимо інструмент автоматичної генерації коду Gii (контролер gii): *http://{site}/web/index.php?r=gii*

✓ Перейдемо в генератор моделі та введемо:



✓ Згенеруємо модель. Пересвідчимось, що в проекті з'явився файл *models/Article.php*

3. Згенеруємо CRUD код. При цьому обов'язкове вказання просторів імен відповідних класів

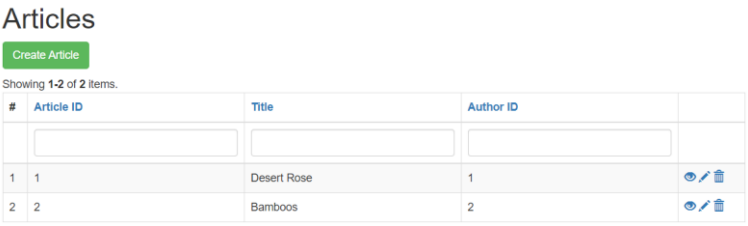


4. Переконаємось, що в проекті з'явились файли для роботи зі статтями:

controllers\ArticleController.php	Контролер для керування обробкою запитів
models\Article.php	Модель для роботи з даними статті
models\ArticleSearch.php	Модель для пошуку, фільтрації, сортування статей
views\article_form.php	Шаблон форми для створення/редагування статті
views\article_search.php	Шаблон форми для пошуку статей
views\article\create.php	Шаблон сторінки створення статті, що вбудовує форму з шаблону _form.php

views\article\index.php	Шаблон сторінки зі списком статей, що вбудовує форму з шаблону _search.php
views\article\update.php	Шаблон сторінки редагування статті, що вбудовує форму з шаблону _form.php
views\article\view.php	Шаблон сторінки перегляду даних про статтю

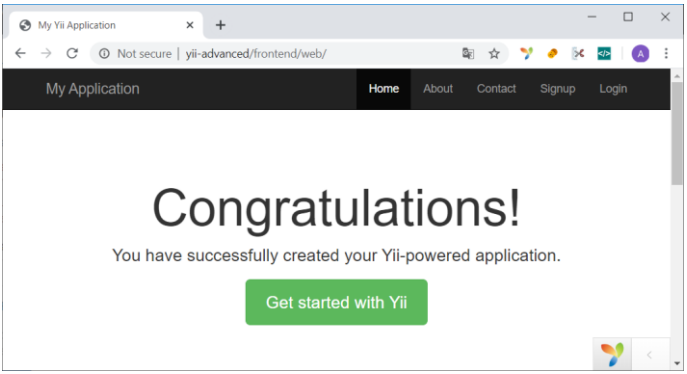
5. Виконайте запит *http://yii-basic/web/index.php?r=article*. З'явиться сторінка зі списком статей та посиланнями на сторінки перегляду, редагування, створення, а також можливістю видалення статей
6. На сторінці списку статей видаліть поле *Description*:



Завдання 4. Реєстрація користувача. Додавання та налаштування додаткових полів

Дане завдання потрібно виконати в проекті із розширеним шаблоном (проект *yii-advanced*), який було встановлено в лабораторній роботі №1.

1. Перевірте роботу веб-сайту, ввівши URL:



2. Перевірте наявність створеної бази даних *yii_advanced*. При потребі створіть дану базу.
3. Відкрийте проект *yii-advanced* в середовищі розробки та налаштуйте з'єднання з БД (файл *common/config/main-local.php*)
4. В командному рядку зробіть міграцію бази даних:

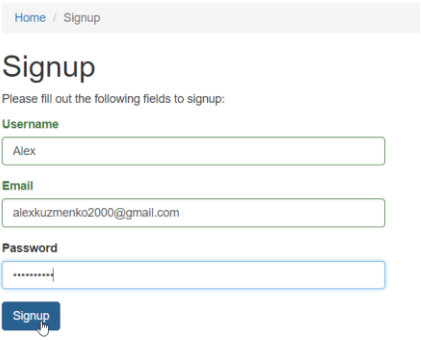
```
cd yii-advanced
yii migrate
```

Примітка. Після виконання міграції в базі даних створяться таблиці *user* та *migration*.

5. В Yii2 передбачено при реєстрації, активацію користувача через електронну пошту. Нам не потрібно поки що здійснювати активацію. Тому модифікуємо метод *signup* моделі форми *SignupForm*. Задамо статус користувача «активний» та скасуємо відправку email для активації:

```
$user->generateEmailVerificationToken();
1 $user->status = User::STATUS_ACTIVE;
  //return $user->save() && $this->sendEmail($user);
2 return $user->save();
}
```

6. В користувацькій частині створіть обліковий запис



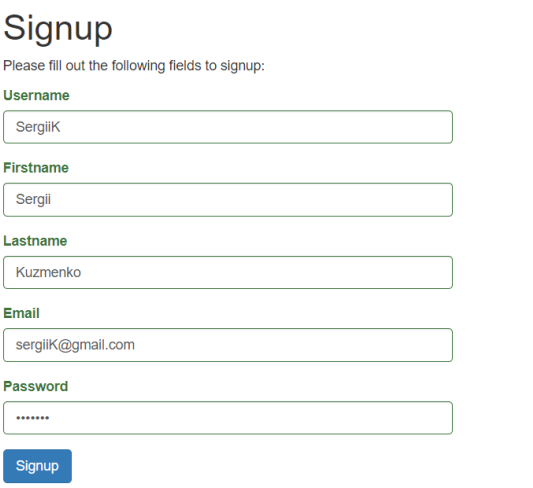
7. Перевірте роботу входу та виходу
8. Додайте два поля *firstname* і *lastname* для користувача. Для цього

- ✓ В таблицю *user* додайте два поля *firstname* і *lastname*:

☐	1	**id** 🔑	int(11)	
☐	2	**username**	varchar(255)	utf8_unicode_ci
☐	3	**firstname**	varchar(255)	utf8_unicode_ci
☐	4	**lastname**	varchar(255)	utf8_unicode_ci
☐	5	**auth_key**	varchar(32)	utf8_unicode_ci

- ✓ В клас моделі, що відповідає за реєстрацію користувачів, додайте доступні властивості `$firstname` і `$lastname`.
- ✓ В тому ж класі в метод *rules* додайте правила для валідації нових полів: вони повинні бути обов'язковими для введення.
- ✓ В правило валідації для поля *firstname* додайте елемент з ключем *message* та рядковим значенням *«Це поле обов'язкове для введення»*
- ✓ Перевірте роботу форми та збереження даних у базі:

id	username	firstname	lastname
1	AlexKuzmenko		
2	Nick		
3	SergiiK	Sergii	Kuzmenko



Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLa

Лабораторна робота №3. Розробка проекту на фреймворку Yii2

Мета: отримати глибокі практичні знання та навички про можливості фреймворку для розробки веб-сайтів з використанням архітектурного шаблону MVC

Завдання

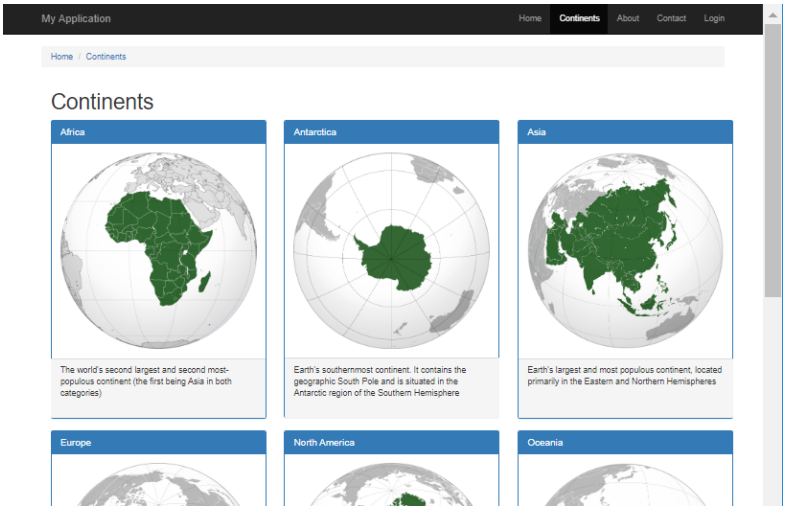
- ✓ Створіть веб-сайт на Yii-фреймворку з трьома динамічними сторінками за зразками
- ✓ Базу даних і зображення завантажити за адресою

https://github.com/alexanderkuzmenko/GeoWorld

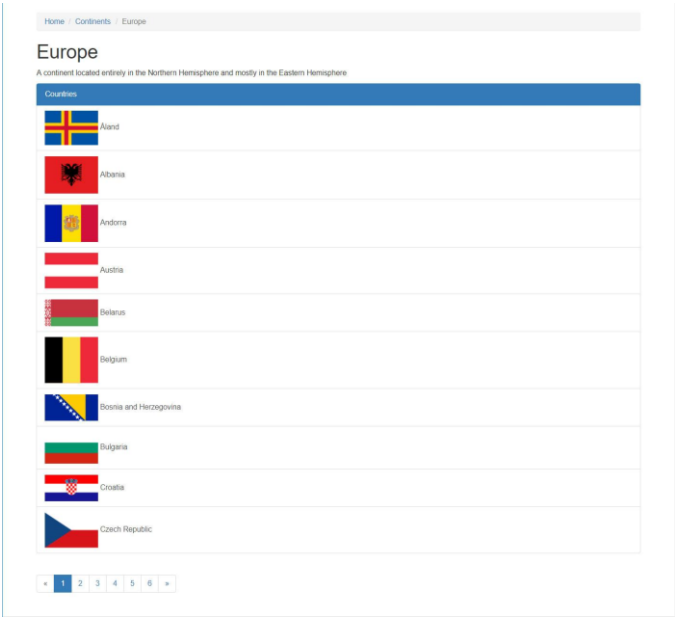
Зразки

- ✓ Панель навігації в шапці сайту. (Continents – посилання на сторінку зі списком континентів).

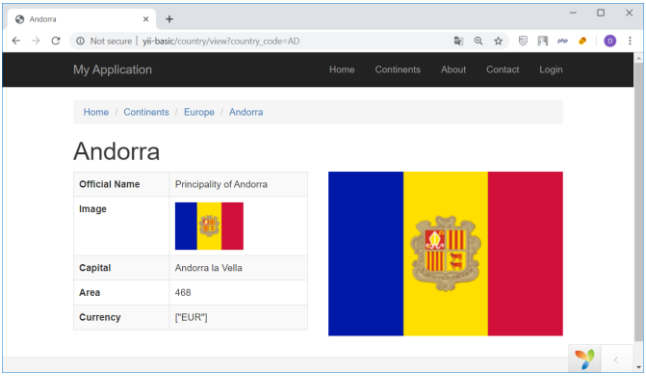
- ✓ Сторінка зі списком континентів. Вона повинна бути домашньою та доступна за URL-адресами *http://yii-basic/continent/index* та *http://yii-basic/*. Зразок сторінки:



- ✓ З сторінки списку континентів можна перейти на сторінку континенту зі списком країн, що йому належать. Сторінка континенту *http://yii-basic/continent/view?code=EU*. Зразок сторінки:

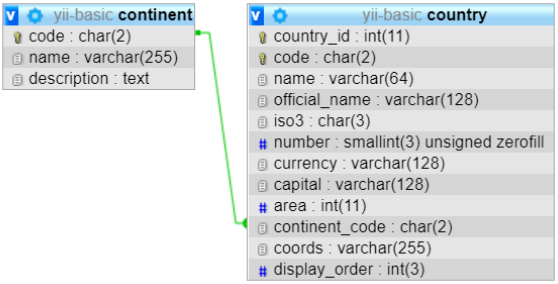


- ✓ З сторінки континенту можна перейти на сторінку з детальною інформацією про країну за посиланням *http://yii-basic/country/view?code=UA*. Зразок сторінки країни:



Підготовчий етап

1. Переконайтесь у наявності проекту з базовим шаблоном *yii-basic*
2. Переконайтесь у наявності бази даних *yii_basic*
3. Імпортуйте дані в базу з *https://github.com/alexanderkuzmenko/GeoWorld*
4. Налаштуйте підключення до БД в файлі *config/db.php*
5. Вивчіть структуру бази даних



Хід роботи

Завдання 1. Розробка сторінки зі списком континентів

1. Розробимо модель континенту. Для цього в папці models створіть клас моделі Continent, що наслідується від ActiveRecord

Лістинг 3.1. Модель Continent

```
<?php

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

class Continent extends ActiveRecord
{

}
```

2. Створимо контролер *continent* для обробки запитів, пов'язаних з континентами. Для цього в папці controllers створимо клас ContinentController з двома методами дії:

Лістинг 3.2. Контролер *continent (controllers/ContinentController.php)*

```
<?php

namespace app\controllers;

class ContinentController extends Controller
{

    public function actionIndex() {}

    public function actionView($code = "AF") {}

}
```

3. Створимо сторінку зі списком континентів

✓ Отримаємо масив континентів та передамо їх в шаблон представлення. Для цього реалізуємо метод дії *index* в контролері *continent*:

Лістинг 3.2. Контролер continent (controllers/ContinentController.php)

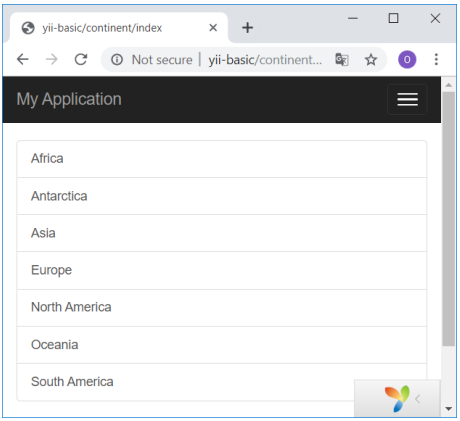
```
public function actionIndex()
{
    $continents = Continent::find()->asArray()->all();

    return $this->render('index', [
        'continents' => $continents,
    ]);
}
```

✓ Представимо отримані дані у вигляді списку. Для цього створіть файл шаблону в папці *views/continent/index.php*:

Лістинг 3.3. Шаблон сторінки зі списком континентів

```
<?php
use yii\helpers\Html;
?>
<div class="list-group">
    <?php foreach ($continents as $continentItem): ?>
        <a href="#" class="list-group-item">
            <?= Html::encode($continentItem['name']); ?>
        </a>
    <?php endforeach; ?>
</div>
```

✓ Виконаємо запит на отримання сторінки зі списком континентів:



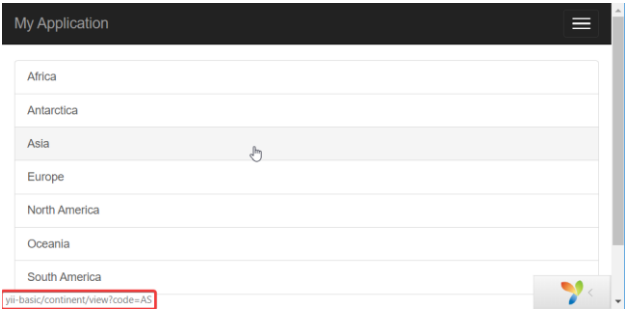
✓ Для кожного континенту згенеруємо посилання на відповідну сторінку. Для цього в елементі списку для тега посилання <a> потрібно записати значення атрибуту href:

Лістинг 3.4. Рядок посилання на сторінку континенту

```
<?= Url::to(['continent/view', 'code' => $continentItem['code']]); ?>
```

Примітка. Для роботи з класом Url потрібно імпортувати його з простору yii\helpers

✓ Перевірте правильність Url:



Завдання 2. Створіть сторінку для перегляду детальної інформації про континент

1. Для отримання даних про один континет в контролері *continent* реалізуємо метод дії *view*:

Лістинг 3.5. Метод дії *view* для обробки запиту на сторінку континенту

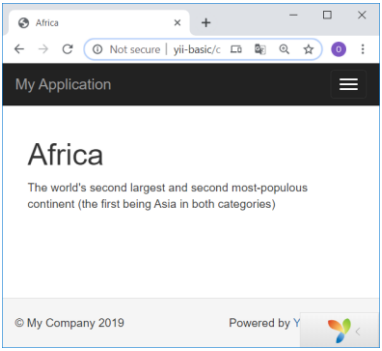
```
public function actionView($code = "AF")
{
    $continent = Continent::findOne(['code' => $code]);
    return $this->render('view', [
       'continent' => $continent
    ]);
}
```

2. Створимо файл шаблону *views/continent/view.php* з отриманими даними континенту

Лістинг 3.6. Шаблон сторінки з інформацією про континент

```
<?php

use yii\helpers\Html;

$this->title = $continent->name;
?>

<section id="continent-view">
    <div class="container">
        <h1><?= Html::encode($this->title); ?></h1>
        <p><?= Html::encode($continent->description); ?></p>
    </div>
</section>
```

3. Виконаємо запит на сторінку континенту



4. Реалізуємо отримання даних про країни даного континенту

✓ Для отримання масиву країн, що належать даному континенту, з розбивкою на сторінки потрібно використати постачальник даних ActiveDataProvider:

Лістинг 3.7. Використання постачальника даних

```
public function actionView($code = "AF")
{
    //Отримання даних про континент
    $continent = Continent::findOne(['code' => $code]);
    //Отримання даних про країни даного континенту
    //Формування постачальника даних
    $countriesDataProvider = new ActiveDataProvider([
        'query' => Country::find()->where(['continent_id' => $continent-
>continent_id]),
        'pagination' => [
            'pageSize' => 20,
        ]
    ]);

    //Передача даних в шаблон вигляду
    return $this->render('view', compact('continent',
'countriesDataProvider'));
}
```

✓ Для виводу країн в шаблоні сторінки континенту використаємо компонент (віджет) ListView:

Лістинг 3.8. Використання віджету даних ListView

```
<?= ListView::widget([
    'dataProvider' => $countriesDataProvider,
    'itemView' => '_country',
```

```
    'layout' => "{items}\n{summary}\n{pager}",
    'options' => [
            'class' => 'list-group'
    ],
    'itemOptions' => [
        'tag' => false,
    ],
]); ?>
```

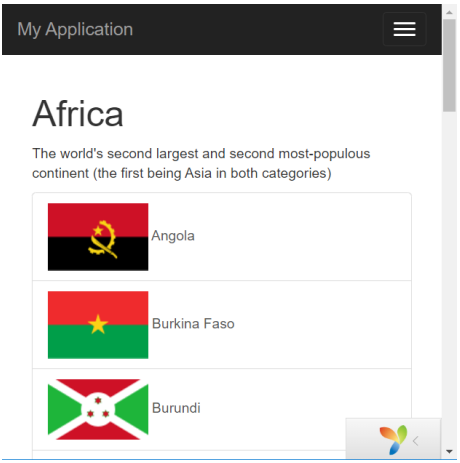
Примітка. В параметр `itemView` записано посилання на файл *_country* з шаблоном одного елемента списку

✓ Створимо файл *views/continent/_country.php* та запишемо в нього html-код елемента списку:

Лістинг 3.9. Шаблон елемента списку

```
<?php
use yii\helpers\Html;
use yii\helpers\Url;
?>
<a href="<?= Url::to(['country/view', 'country_code' => $model->code]); ?>"
class="list-group-item list-group-item-action">
    <?= Html::img('@web/images/countries/png100px/' .
strtolower($model['code']) . '.png', ['alt' => $model['name']]) ?>
    <?= Html::encode($model->name) ?>
</a>
```

✓ Зробимо перевірку:



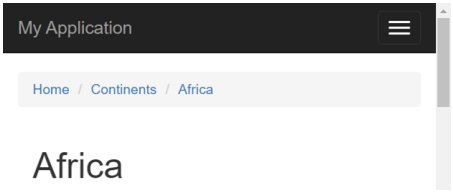
Завдання 3. Реалізуємо ланцюжки навігації

1. Створимо елементи ланцюжка на сторінці континенту:

Лістинг 3.10. Елементи ланцюжка навігації на сторінці континенту

```
$this->title = $continent->name;
$this->params['breadcrumbs'][] = ['label' => 'Continents', 'url' =>
['continent/index']];
$this->params['breadcrumbs'][] = ['label' => $continent['name'], 'url' =>
['continent/view', 'code' => $continent['code']]];
```

2. Протестуємо:



3. Аналогічно створіть такі ланцюжки на інших сторінках

Завдання 4. Шаблон сторінки зі списком континентів модифікуйте за таким зразком, наведеним в описі завдань

Завдання 5. Розробіть повний цикл обробки запиту на сторінку з даними про країну та розробіть шаблон сторінки за зразком, наведеним в описі завдань

Вимоги до звіту

Звіт повинен складатись:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Лабораторна робота №4. Генерація та розбір URL. Робота з багаторівневими даними

Мета. Навчитись працювати з механізмом генерації та розбору URL, ланцюжком навігації, ієрархією категорій

Завдання

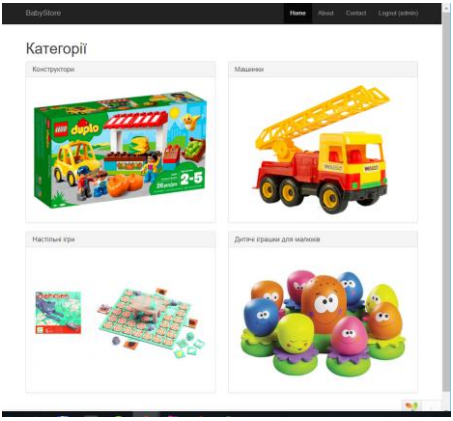
В клієнтській частині розробити дві сторінки в клієнтській частині за зразками

Підготовчий етап

- ✓ Переконатись в наявності встановленого проекту Yii2 (Advanced Template)
- ✓ Використати базу даних та файли зображень за цим посиланням *https://bit.ly/2Yjk3yH*

Хід роботи

1. *Головна сторінка* повинна містити перелік базових категорій. Розробіть повний цикл формування головної сторінки та шаблон розмітки за зразком

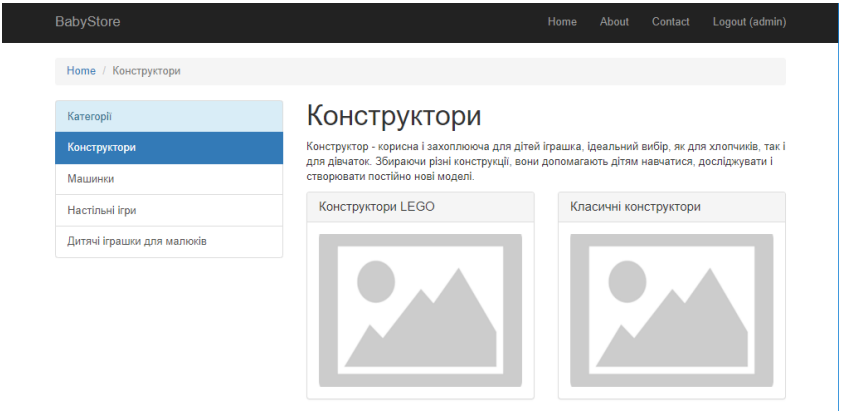


2. Доступ на головну сторінку реалізуйте за посиланнями:

- ✓ *http://{host}/*

- ✓ *http://{host}/category/index*
- ✓ *http://{host}/categories*

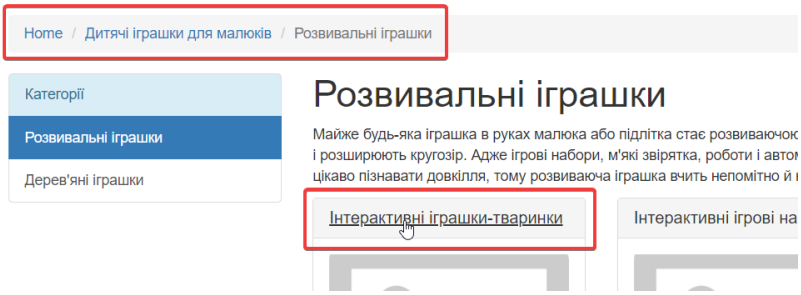
3. *Сторінка категорії*. Зліва меню категорій того ж рівня, що і дана категорія. В головній області міститься список підкатегорій

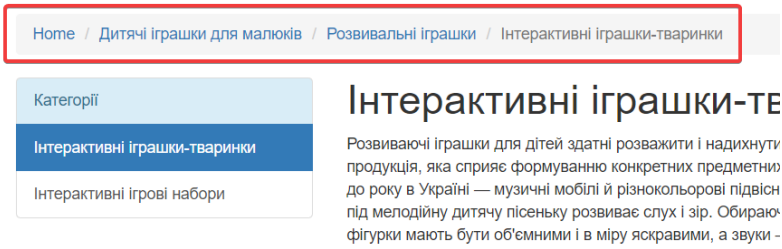


4. Доступ на сторінку категорії реалізувати за посиланнями:

- ✓ *http://{host}/category/view/?id={id}*
- ✓ *http://{host}/category/{id}*

5. Ланцюжок навігації реалізувати з врахуванням вкладеності категорій (може бути як завгодно великий рівень вкладеності)





Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Лабораторна робота №5. Розробка системи авторизації. Робота з механізмом контролю доступу на основі ролей та дозволів (RBAC)

Мета. Отримати теоретичні знання та практичні вміння використання механізму контролю доступу на основі системи ролей, а також використання фільтрів доступу.

Завдання

Створити ролі та налаштувати права, використовуючи керування доступом на основі ролей RBAC:

- ✓ Роль *guest* (користувач, що не здійснив вхід). Має право на перегляд даних про категорії/товари.
- ✓ Роль *authUser*. Надається автоматично користувачу, що реєструється. Після входу має право на перегляд товарів/категорій/виробників. Не має доступу до адміністративної частини сайту.
- ✓ Роль *manager*. Має всі права authUser. Також може додавати товари та змінювати дані лише про свої товари.
- ✓ Роль *admin*. Має всі права manager, а також може змінювати всі товари, додавати/змінювати категорії та виробників.
- ✓ Роль *superAdmin*. Має всі права admin, а також може видаляти товари/категорії/виробників.

Підготовчий етап

- ✓ Переконатись в наявності встановленого проекту Yii2 (Advanced Template) та виконаних завданнях лабораторної роботи №4
- ✓ Згенерувати в адміністративній частині проекту CRUD-код для роботи з товарами, категоріями та виробниками.

Хід роботи

1. Налаштуйте компонент *authManager* в файлі *common/config/main*:

Лістинг 5.1. Налаштування компонента менеджера авторизації

```
    'authManager' => [
        'class' => 'yii\rbac\DbManager',
    ],
```

2. Для створення таблиць механізму ролей, виконайте команду міграції:

```
yii migrate --migrationPath=@yii/rbac/migrations
```

3. В папці *console/controllers* створіть контролер *RbacController*, що розширяється від класу *Controller*, та реалізуйте в ньому метод дії *actionInit*.
4. В методі *actionInit* створіть дозволи на додавання товару *createProduct* та на редагування товару *updateProduct*
5. Далі створіть ролі *authUser, manager, admin, superAdmin*
6. Організуйте ієрархію ролей та дозволів відповідно до завдання
7. В базу даних додайте користувачів Jane, John, Alex, Peter. Призначити такі ролі користувачам:

- ✓ Jane - *superAdmin*
- ✓ John - *admin*
- ✓ Alex - *manager*
- ✓ Peter (а також всі, хто реєструється) - *authUser*

8. Оскільки менеджери повинні мати доступ до редагування лише своїх товарів, створимо окремий дозвіл *updateOwnProduct*, який є батьківським для *updateProduct*, та обмежується правилом. Правило теж потрібно створити.
9. Для задання правила, створимо папку *common/rbac*, в якій створимо клас *ManagerRule* з методом *execute*, що повертає *true*, якщо поточний користувач є «власником» поточного товару:

Лістинг 5.2. Правило *isManager* для обмеження дозвіл редагування товарів

```
<?php
namespace common\rbac;

use yii\rbac\Rule;
```

```
class ManagerRule extends Rule
{
    public $name = 'isManager';

    public function execute($user, $item, $params)
    {
        return isset($params['product']) ? $params['product']->user_id ==
$user : false;
    }
}
```

10. В контролері *rbac* створіть ще один метод дії *manager-rule*. В цьому методі створіть дозвіл *updateOwnProduct*, примініть для нього описане правило, зробіть дозвіл *updateOwnProduct* батьківським для *updateProduct*, а також призначте ролі *manager* щойно створений дозвіл.
11. Виконаємо команди для створення ролей і збереження їх в базі даних

```
yii rbac/init
yii rbac/manager-rule
```

12. Здійсніть перевірку прав доступу для методу дії actionUpdate контролера ProductController, додавши на початок код:

Лістинг 5.3. Використання правила з врахуванням дозволу в контролері

```
$model = $this->findModel($id);
if (!Yii::$app->user->can('updateProduct', ['product' => $model])) {
    throw new ForbiddenHttpException("You don't have permisson to update
product");
}
```

13. Врахуйте перевірку всіх ролей та дозволів в коді нашого проекту згідно завдання та здійсніть перевірку доступу для користувачів з різними ролями.

Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Лабораторна робота №6. Робота з файлами (зображеннями)

Мета. Познайомитись та реалізувати цикл обробки завантаження файлів на сервер

Завдання

Розробити функціонал додавання/редагування зображень для категорій, товарів та виробників.

Кожному товару, категорії та виробнику відповідає одне головне зображення, ім'я якого задається атрибутом image. Якщо поле не задане або файлу не існує, то замість зображення задати placeholder.

Зображення помістити в відповідні папки:

- ✓ {url}/images/category/
- ✓ {url}/images/product/
- ✓ {url}/images/manufacturer/

Прямий доступ до даних папок заборонити.

Файли зображень, що завантажуються, повинні проходити валідацію.

Здійснити обробку зображень під час завантаження засобами PHP. Остаточне зображення повинно мати фіксовані розміри 800x600px (або 600x800px в залежності від орієнтації). Якщо пропорції зображення інші, їх змінювати не потрібно Функції по обробці зображення винести в окремий клас Image.

Підготовчий етап

Переконатись в наявності встановленого проекту Yii2 (Advanced Template) та виконаних завданнях лабораторної роботи №4

Теоретичні відомості

- ✓ Завантаження відбувається через клас *yii\web\UploadedFile*
- ✓ Кожен завантажений файл представляється як екземпляр даного класу
- ✓ Для реалізації методів завантаження потрібно створити модель
- ✓ В html потрібно створити форму, скориставшись віджетом yii\webs\ActiveForm

*Модель***.** Для завантаження файлу можна створити клас моделі та використовувати його атрибут для збереження екземпляру об'єкта UploadedFile, що містить параметри файлу, що завантажується. Також можливе використання правил валідації для перевірки цього файлу:

Лістинг 6.1. Модель форми UploadForm

```
<?php

namespace app\models;

use yii\base\Model;
use yii\web\UploadedFile;

class UploadForm extends Model
{
    /**
     * @var UploadedFile
     */
    public $imageFile;

    public function rules()
    {
        return [
            [['imageFile'], 'file', 'skipOnEmpty' => false, 'extensions' =>
'png, jpg'],
        ];
    }

    public function upload()
    {
        if ($this->validate()) {
            $this->imageFile->saveAs('uploads/' . $this->imageFile->baseName
. '.' . $this->imageFile->extension);
            return true;
        } else {
            return false;
        }
    }
}
```

В даній моделі атрибут imageFile використовується для зберігання екземпляру файлу, що завантажується. Правило валідації file, яке за допомогою валідатора yii\validators\FileValidator, перевіряє розширення файлу, що завантажується на відповідність з png або jpg. Метод upload() виконує валідацію та зберігає файл, що завантажується на сервері.

Лістинг 6.3. Приклад шаблону форми для завантаження

```
<?php
use yii\widgets\ActiveForm;
?>

<?php $form = ActiveForm::begin(['options' => ['enctype' => 'multipart/form-
data']]) ?>
<?= $form->field($model, 'imageFile')->fileInput() ?>
<button>Submit</button>
<?php ActiveForm::end() ?>
```

Для керування процесом завантаження наведемо приклад методу дії контролера:

Лістинг 6.4. Обробка запиту завантаження файлу в контролері

```
public function actionUpload()
{
    $model = new UploadForm();

    if (Yii::$app->request->isPost) {
        $model->imageFile = UploadedFile::getInstance($model, 'imageFile');
        if ($model->upload()) {
            // file is uploaded successfully
            return;
        }
    }

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

Хід роботи

1. Створіть шаблон сторінки imageForm з формою додавання зображення
2. Створіть модель UploadForm з атрибутом imageFile для зберігання екземпляру UploadedFile
3. Створіть метод дії actionUpload
4. Реалізуйте в контролері наступний сценарій завантаження та збереження файлу зображення на сервері:

✓ Створюємо екземпляр моделі форми

- ✓ Отримуємо екземпляр товару за ідентифікатором
- ✓ Якщо файл відправлено та пройдено валідацію
    - Ініціюємо атрибут *imageFile*
    - Генеруємо ім'я для файлу, що зберігатиметься на сервері
    - Якщо файл успішно зберігся в папці
        - Запам'ятовуємо ім'я неактуального файлу
        - Записуємо нове ім'я в базу даних
        - Якщо помилка під час запиту, видаляємо новий файл
        - Видаляємо старий файл
        - Переадресовуємо запит на сторінку перегляду даних про товар з новим зображенням
- ✓ Відображаємо сторінку з формою

Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Лабораторна робота №7. Особливості роботи з відношенням «багато-до-багатьох»

Мета. Отримати практичні навички для роботи з віджетами представлень. Реалізувати форму для множинного призначення категорій товарам. Опрацювати розширення yii-faker.

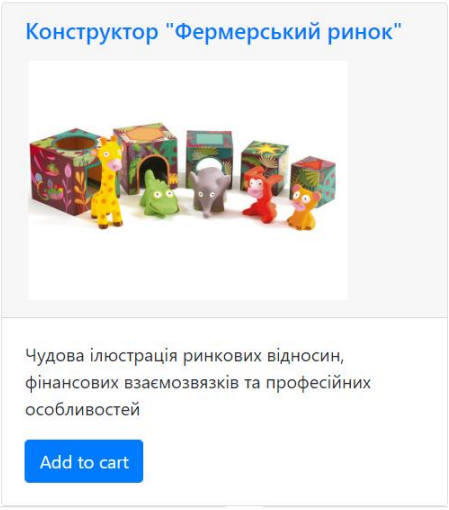
Завдання

Розробити сторінки з переглядом даних про товар. Забезпечити можливість множинного призначення категорій для товару. Реалізувати можливості розширення yii-faker для генерації демонстраційних даних

Хід роботи

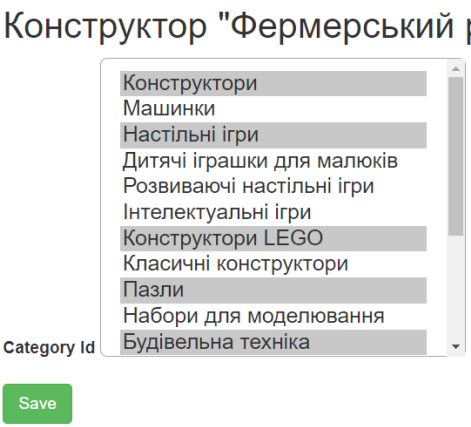
Завдання 1. В клієнтській частині веб-магазину на сторінці з даними про категорію відобразити товари, що належать безпосередньо даній категорії. Вимоги:

- ✓ Товар повинен відображатись в картці (назва товару, посилання на сторінку товару, зображення, короткий опис, ціна, кнопка “Додати”)



- ✓ В URL товару передбачити передачу ID товару та ID категорії (Категорія необхідна для *breadcrumbs*)
- ✓ Сформувати сторінку перегляду даних про товар: назва, зображення, повний опис, ціна, кнопка додати.
- ✓ На сторінці товару сформувати ланцюжок посилань категорій

Завдання 2. В адміністративній частині реалізувати можливість призначення товарів категоріям. Для форми потрібно створити модель і забезпечити зв'язування поля *CategoryId* з об'єктом моделі



Завдання 3. Використовуючи розширення *yii-faker* згенерувати 1000 товарів та занести їх у базу. Занести кожен товар у випадково вибрані категорії. Кількість таких категорій - теж випадкове число від 1 до 4.

Після виконання завдання 3 протестувати завдання 1 і 2.

Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Лабораторна робота №8. Робота з сесією та асинхронними запитами. Конфігурування компонента mailer

Мета. Здобути практичні навички роботи з сесією, асинхронними запитами та конфігурування електронної пошти.

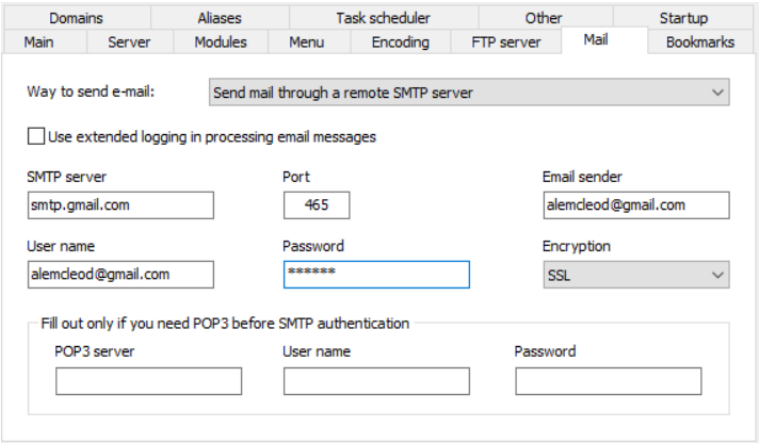
Завдання

Реалізувати можливість формування кошика замовлень та відправляти замовлення на електронну пошту.

Формування кошика замовлень реалізуйте через механізм сесії, в якій зберігається елемент *cart* з масивом товарів, що замовляються. Ідентифікатори товарів заносяться в масив як його ключі. Значення елементів - це масиви з двома елементами: ціна та кількість.

Теоретичні відомості

Конфігурація OpenServer для відправлення поштових повідомлень



Налаштування поштового компонента виглядає так:

Лістинг 8.1. Компонент mailer (збереження електронних листів в файли)

```
'components' => [
    'mailer' => [
        'class' => 'yii\swiftmailer\Mailer',
        'viewPath' => '@common/mail',
```

```
        'useFileTransport' => true
    ],
],
```

Опція *useFileTransport* означає, що відправка Email здійснюється в файли (тестовий режим)

Конфігурація *mailer* для відправки реального повідомлення. Конфігурувати сервер при цьому не потрібно.

Лістинг 8.2. Компонент mailer для реальної відправки листів

```
'useFileTransport' => false,
'transport' => [
    'class' => 'Swift_SmtpTransport',
    'host' => 'smtp.gmail.com',
    'username' => 'alemcleod@gmail.com',
    'password' => '******',
    'port' => '465',
    'encryption' => 'ssl',
    'streamOptions' => [
        'ssl' => [
            'allow_self_signed' => true,
            'verify_peer' => false,
            'verify_peer_name' => false,
        ],
    ],
],
```

Конфігурування пошти відправника на Gmail:

- ✓ Включити IMAP Access (Протокол доступу до електронної пошти) Gmail -> Settings -> Forwarding and POP/IMAP -> IMAP Access -> Enable
- ✓ Дозволити стороннім додаткам доступ до сервера електронної пошти: *https://myaccount.google.com/lesssecureapps*

Лістинг 8.3. Приклад створення поштового повідомлення

```
Yii::$app->mailer->compose()
    ->setFrom('admin@mysite.com')
    ->setTo('user@mysite.com')
    ->setSubject("Subject")
    ->setTextBody("Text of notification")
    ->send();
```

Хід роботи

1. Використовуючи механізм асинхронних запитів Pjax, обробіть запит натискання на кнопку «Замовити». При цьому кількість даного товару в замовленні повинна збільшуватись на одиницю. В шапці сайту біля іконки «Кошик» відображається повна вартість замовлення;
2. Створіть сторінку cart з можливістю оформлення замовлення. Форма для замовлення повинна виглядати так:

Cart

Name	Price	Quantity	Total
Name	0	+ 0 −	0
Totals		**0**	**0**

При збільшенні/зменшенні кількості товару, дані замовлення повинні переобчислюватись без перевантаження сторінки. Всі величини не повинні бути від'ємними.

Реалізуйте збереження замовлення в БД та відправку даних замовлення поштою користувачам, які є адміністраторами системи.

Вимоги до звіту

Звіт повинен складатись з:

- ✓ Назви, теми і мети лабораторної роботи
- ✓ Лістингів програмного коду з поясненнями

Реалізований проект згідно завдання з повним доступом для користувача @alexkuzmenko на репозиторії системи GitLab

Література

1. Повне керівництво по Yii2.0. [Електронний ресурс]. Режим доступу
2. https://www.yiiframework.com/doc/guide/2.0
3. Bill Keck. Yii 2 For Beginners
4. Mark Safronov, Jeffrey Winesett. Web Application Development with Yii 2 and PHP
5. Matteo Pescarin. Learning Yii Testing
6. Andrew Bogdanov, Dmitry Eliseev. Yii2 Application Development Cookbook

Навчально-методичне видання

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З ДИСЦИПЛІНИ “Інтернет-програмування: PHP”

Підготували
Кузьменко Олександр Вікторович
Петросян Руслан Валерікович

Технічний редактор **О.В. Кузьменко**
Комп’ютерне верстання **– О.В. Кузьменко**
Підписано до друку _____. Формат 60×84/16.
Ум. друк. арк. _____. Зам. офс.