Міністерство освіти і науки України
Національний університет водного господарства та природокористування

Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра обчислювальної техніки

04-05-86М

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з навчальної дисципліни
«Високорівневі мови програмування» (частина 1)
для здобувачів вищої освіти другого (магістерського) рівня
за освітньо-професійною програмою
«Інформаційні технології в бізнесі»
спеціальності 126 «Інформаційні системи та технології»
денної та заочної форми навчання

Рекомендовано
науково-методичною радою
з якості ННІКІТІ
Протокол № 8 від 27.06.2024 р.

Рівне – 2024

Методичні вказівки до лабораторних робіт з навчальної дисципліни «Високорівневі мови програмування» (частина 1) для здобувачів вищої освіти другого (магістерського) рівня за освітньо-професійною програмою «Інформаційні технології в бізнесі» спеціальності 126 «Інформаційні системи та технології» денної та заочної форми навчання. [Електронне видання] / Бойчура М. В. – Рівне : НУВГП, 2024. – 37 с.

Укладач: Бойчура М. В., к.т.н., доцент кафедри обчислювальної техніки.

Відповідальний за випуск: Сидор А. І., к.т.н., в.о. завідувача кафедри обчислювальної техніки.

Керівник групи забезпечення
спеціальності 126 «Інформаційні
системи та технології» Барановський С. В.

ЗМІСТ

Вступ

Освітні компоненти магістерського рівня вищої освіти зазвичай повинні узагальнювати увесь той багаж знань, який викладається на бакалавраті. Тому у межах дисципліни, присвяченої високорівневому програмуванню, доцільно, щоб у єдине ціле об'єднувались (систематизувались) знання по роботі з базами даних, *API*, *HTML*, *CSS*, *JavaScript*, іншими мовами програмування, архітектурами програмного забезпечення, авторизацією і автентифікацією, патернами, роботою з системами керування проектами і версіями тощо. Дуже важливо, щоб при цьому студенти покращували свої м'які навички. Особливо актуальними в *IT*-сфері є вміння працювати у команді та спілкуватись англійською мовою. Дуже цінуються знання та навички, необхідні для професійної діяльності з планування, проектування та розробки веб-застосунків. Усе це (і навіть більше) передбачається у навчальній дисципліні «Високорівневі мови програмування». Таким чином, здійснюватиметься комплексна підготовка фахівця з веб-програмування у всіх аспектах набуття професійних навичок, визначених у курсі магістра з інформаційних технологій в бізнесі.

Зокрема у першій частині даних методичних вказівок до виконання лабораторних робіт відбувається знайомство з фреймворком *ASP.NET Core*, вивчення патерну *MVC*, передбачається виконання *CRUD*-операцій у контексті веб-програмування, вивчається пошарова архітектура програмного забезпечення та пояснюються особливості командної розробки на основі *GitHub* та *Jira*. Щодо кожної лабораторної роботи вказано її мету, завдання та максимальний відсоток балів за правильне їх виконання, вимоги та рекомендації до програмної реалізації, посилання

на додаткові відеоматеріали і наведено контрольні запитання. Лабораторні роботи тісно пов'язані між собою. Студент може ще на першому занятті обрати тему проекту (або запропонувати свою) та наповнювати відповідний код новим функціоналом в межах всіх наступних лабораторних робіт.

Кожна лабораторна робота оцінюється максимум в 3 бали. Виставлення оцінок відбувається на основі продемонстрованого коду програми, а також рівня правильності наповненості *GitHub* та *Jira*. Але для отримання навіть мінімального балу студент повинен пояснити кожен рядок коду програми.

Як альтернатива, студенти можуть виконувати й інші завдання, навіть за допомогою інших технологій веб-програмування. Також передбачений варіант перезарахування балів за результатами неформальної освіти. Але усі нюанси варто попередньо узгодити з викладачем.

Дисципліна «Високорівневі мови програмування» є вибірковою компонентою освітньо-професійної програми «Інформаційні технології в бізнесі» магістерського рівня вищої освіти. Вивчається у 2-му семестрі.

Метою першого модуля освітньої компоненти «Високорівневі мови програмування» є оволодіння студентами основами веб-розробки на базі фреймворку *ASP.NET Core MVC*.

В результаті виконання усіх лабораторних робіт першого модуля студенти повинні:

Знати:

- основи фреймворку ASP.NET Core та його архітектуру;
- принципи роботи з патерном MVC;
- CRUD-операції в контексті веб-програмування;

– пошарову архітектуру програмного забезпечення;

– особливості командної розробки з використанням GitHub та Jira.

Вміти:

– створювати та редагувати проекти на базі ASP.NET Core MVC;

– виконувати CRUD-операції для маніпуляції даними;

– використовувати фреймворк Bootstrap для оформлення веб-сторінок;

– розробляти проекти з урахуванням принципів пошарової архітектури;

– ефективно співпрацювати в команді, використовуючи інструменти управління проектами та версіями коду (GitHub, Jira).

Дані методичні вказівки побудовані у відповідності до побажань та рекомендацій студентів та стейкхолдерів. Зокрема, базуючись на матеріалах курсу .NET Advanced for Teachers, що проводився IT-компанією SoftServe, в межах даної освітньої компоненти передбачається поглиблене вивчення командної веб-розробки на основі фреймворку ASP.NET Core MVC.

Лабораторна робота №1
Вступ. Огляд шаблону ASP.NET Core MVC

Мета

1. Навчитись створювати та редагувати основні файли проекту *ASP.NET Core MVC*.
2. Пригадати фреймворк *Bootstrap*.

Завдання

1. *(40% балів)* Встановіть компоненти (рис. 1), важливі для роботи з проектом *ASP.NET Core MVC*.
2. *(5% балів)* Створіть порожній проект *ASP.NET Core MVC* із готовим шаблоном для реєстрації та автентифікації.
3. *(35% балів)* Внесіть невеликі зміни в *header*, *content* та *footer* згідно запропонованого варіанту або узгодьте із викладачем власну тему.
4. *(20% балів)* Додайте будь-які кнопку, картку та текстове поле із використанням фреймворку *Bootstrap*.

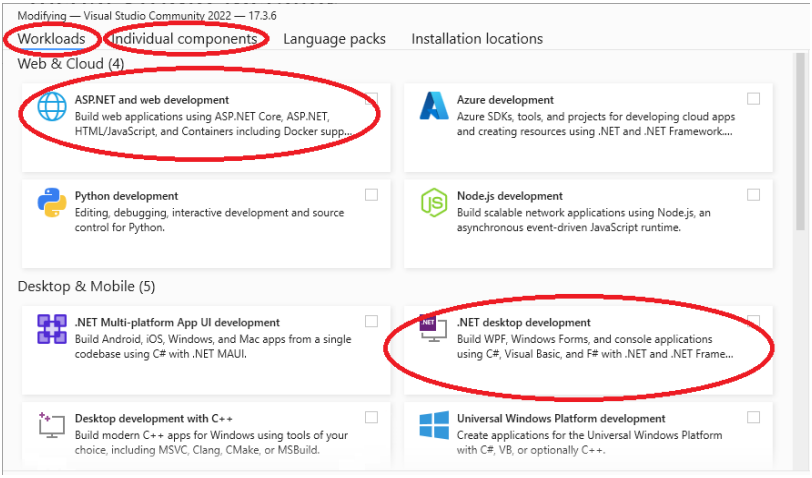


Рис. 1. Вікно *Microsoft Visual Studio Installer*; відмічено частину компонентів, які рекомендується встановити

Варіанти до виконання поставленого завдання

Варіант 1. Потрібно створити сторінку із автобіографією. Там описати свої хобі (із фотографіями), результати поточної успішності (у вигляді таблиці) і т.д.

Варіант 2. Розробити сторінку інтернет-магазину. Тут описати інформацію про наявні продукти (із фотографіями), способи доставки, вартість товарів (у вигляді таблиці) і т.д.

Варіант 3. Розробити сторінку кафедри. Тут описати інформацію про викладачів (із фотографіями), предмети, які викладають викладачі кафедри (у вигляді таблиці) і т.д.

Варіант 4. Розробити сторінку аптеки. Тут описати перелік ліків (із фотографіями), їх кількості та ціни (у вигляді таблиці) і т.д.

Варіант 5. Розробити сторінку футбольного комітету. Тут описати керівний склад комітету (із фотографіями), список команд з додатковою інформацією (у вигляді таблиці) і т.д.

Варіант 6. Розробити сторінку громадської організації. Тут описати основних учасників організації (із фотографіями), напрямки діяльності з вказанням відповідальних учасників (у вигляді таблиці) і т.д.

Варіант 7. Розробити сторінку студентського гуртка. Тут описати основних учасників гуртка (із фотографіями), розклад заходів у поточному році (у вигляді таблиці) і т.д.

Варіант 8. Розробити сторінку косметичної компанії. Тут описати наявність розповсюджуваної продукції (із фотографіями), опис характеристик та ціну (у вигляді таблиці) і т.д.

Варіант 9. Розробити сторінку політичної організації. Тут описати керівний склад організації (із фотографіями), напрямки діяльності та відповідальних виконавців (у вигляді таблиці) і т.д.

Варіант 10. Розробити сторінку з заголовком «Олімп». Тут описати грецьких або римських богів (із фотографіями), сфери відповідальності кожного з них (у вигляді таблиці) і т.д.

Варіант 11. Розробити сторінку інформаційного відділу підприємства. Тут описати персональний склад відділу (із фотографіями), перелік послуг з вказанням їх вартості (у вигляді таблиці) і т.д.

Варіант 12. Розробити сторінку Вашої студентської групи. Тут описати перелік студентів, які навчаються у даній академічній групі (із фотографіями), персональні дані (номер залікової крижки, дата народження, адреса проживання – у вигляді таблиці) і т.д.

Варіант 13. Розробити сторінку банку. Тут описати базу даних клієнтів банку (із фотографіями), наявні послуги з вказанням цінової політики (у вигляді таблиці) і т.д.

Варіант 14. Розробити Web-сторінку кінотеатру. Тут описати перелік фільмів у прокаті (із фотографіями), прайс-лист (у вигляді таблиці) і т.д.

Варіант 15. Розробити сторінку асоціації кінолюбителів. Навести перелік членів асоціації (із фотографіями), перелічити переможців різних кінопремій за останні 5 років (у вигляді таблиць), розмістити на сторінці анонси найближчих відомих фільмів.

Варіант 16. Розробити сторінку «Сайт наукових новин». Тут описати поточні новини (із фотографіями), склад команди журналістів з інформацією про них (у вигляді таблиці) і т.д.

Варіант 17. Розробити сторінку агентства нерухомості. Розмістити на ній всі наявні пропозиції щодо продажу та купівлі житла (із фотографіями), навести перелік ріелторів, їх номери телефонів та досвід роботи.

Варіант 18. Розробити сторінку автомобільного салону. Навести перелік автомобілів з їх фотографіями та цінами (у вигляді таблиці), розмістити інформацію із подяками від задоволених покупців тощо.

Варіант 19. Розробити сторінку асоціації книголюбів. Тут навести перелік її осередків у країні, склад кожного з осередків (із фотографіями), перелічити рекомендовані асоціацією книги для прочитання (у вигляді таблиці) тощо.

Варіант 20. Розробити сторінку студентської ради. Тут навести склад ради і перелік асоційованих членів із фотографіями (у вигляді таблиці), джерела фінансування, проекти ухвалених рішень, статут тощо.

Вимоги та рекомендації до програмної реалізації

Рекомендується використовувати середовище розробки *Microsoft Visual Studio 2022* версії *Community*. Відповідні файли для встановлення можна завантажити з офіційного сайту *Microsoft*: https://visualstudio.microsoft.com/downloads/.

Для коректної роботи усього функціоналу, який буде зустрічатись в межах даного предмету, важливо у вікні *Microsoft Visual Studio Installer* довстановити 2 основні компоненти (див. рис. 1):

– "*.Net desktop development*";

– "*ASP.NET and web development*"

та 7 додаткових компонент:

– "*Перевірка залежностей*" (англ. *Dependency Validation*);

– "*Конструктор класів*" (англ. *Class Designer*);

– "*Підтримка мов JavaScript і TypeScript*" (англ. *JavaScript and TypeScript language support*);

– "*Розширені можливості ASP.NET*" (англ. *Advanced ASP.NET features*);

– "*Git для Windows*" (англ. *Git for Windows*);
– "*Інструменти LINQ to SQL*" (англ. *LINQ to SQL tools*);
– "*ASP.NET MVC 4*".

Створення першого проекту на фреймворку *ASP.NET Core MVC* відбувається за допомогою шаблону *ASP.NET Core Web App (Model-View-Controller)*. При цьому для автоматичного додавання інтерфейсу реєстрації та автентифікації необхідно стосовно *Authentication type* обрати *Individual Accounts*. Бажано, щоб пункт *Do not use top-level statements* був не активним. Рекомендовані налаштування при створенні проекту *ASP.NET Core MVC* наведені на рис. 2.

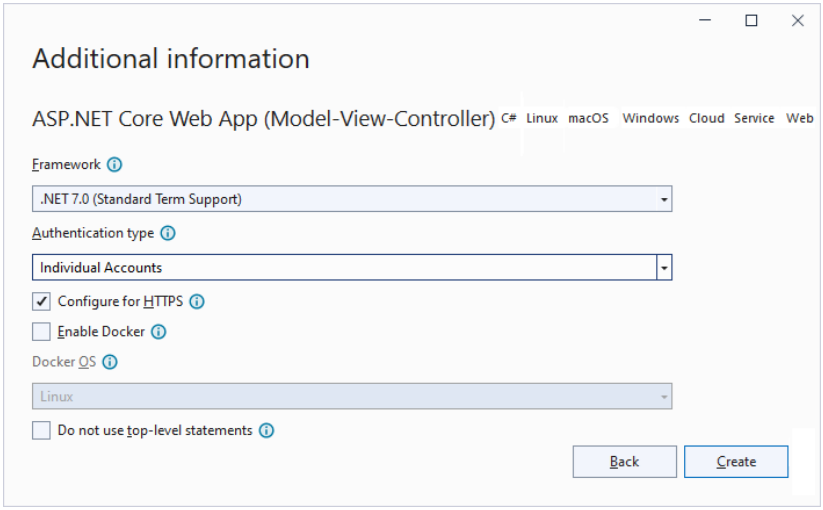
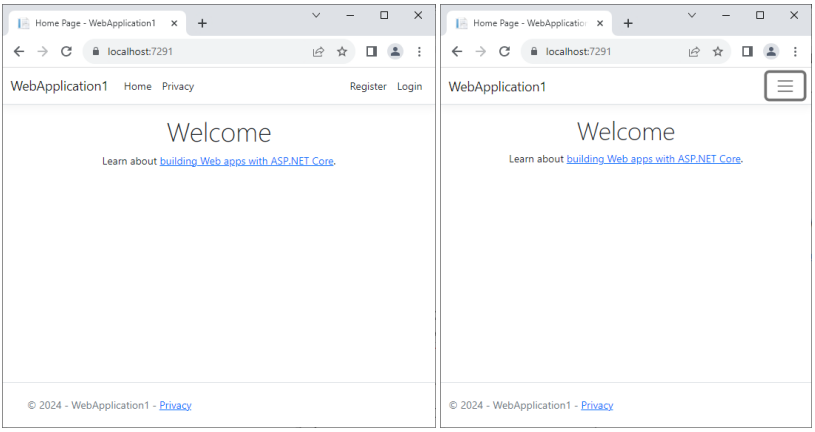


Рис. 2. Рекомендовані налаштування при створенні проекту *ASP.NET Core MVC*

Під час першого запуску проекту (рис. 3) може виникнути попередження стосовно *SSL* сертифікатів

(рис. 4). Рекомендується на всі запитання давати відповідь *Yes*.



а) б)

Рис. 3. Домашня сторінка при першому запуску проекту із продемонстрованою адаптивністю дизайну

Внесення змін у *header*, *content* та *footer* вимагає роботи з файлами: *_Layout.cshtml*, *Index.cshtml*, *HomeController.cs* (рис. 5).

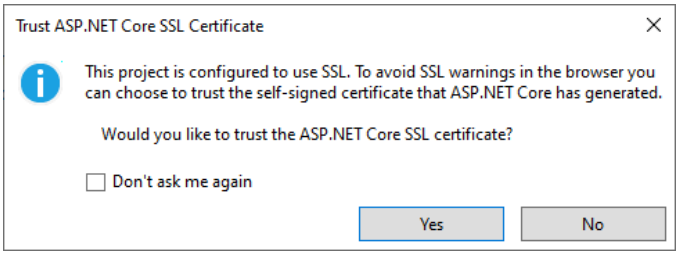
Офіційний сайт фреймворку *Bootstrap* наступний: https://getbootstrap.com/. Найбільш зручно додавати компоненти з необхідними стилями є за допомогою інструменту пошуку, розміщеного на головній сторінці сайту. Наприклад, для добавлення кнопки необхідно виконати наступну послідовність кроків:

1) у рядок пошуку ввести слово *button* та натиснути *Enter*;

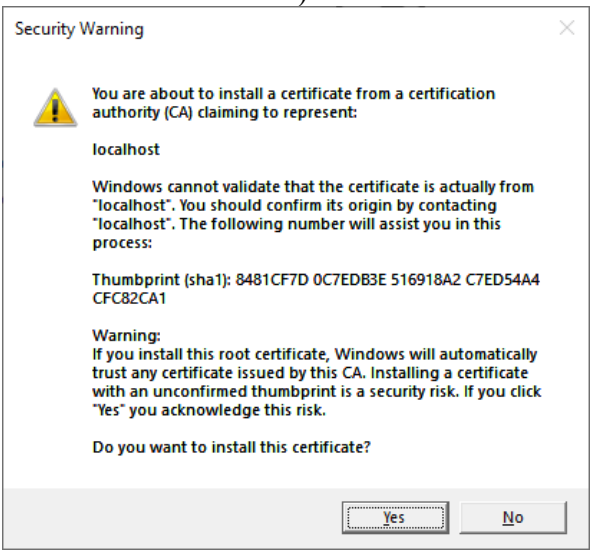
2) серед наведеного переліку кнопок знайти ту, яка найбільш підходить для конкретного випадку;

3) скопіювати код, наведений нижче кнопки, після чого вставити його у потрібний файл проекту.

Аналогічну послідовність дій можна виконати й для карток (*card*) та текстового поля (*text*).



а)



б)

Рис. 4. Попередження стосовно *SSL* сертифікатів

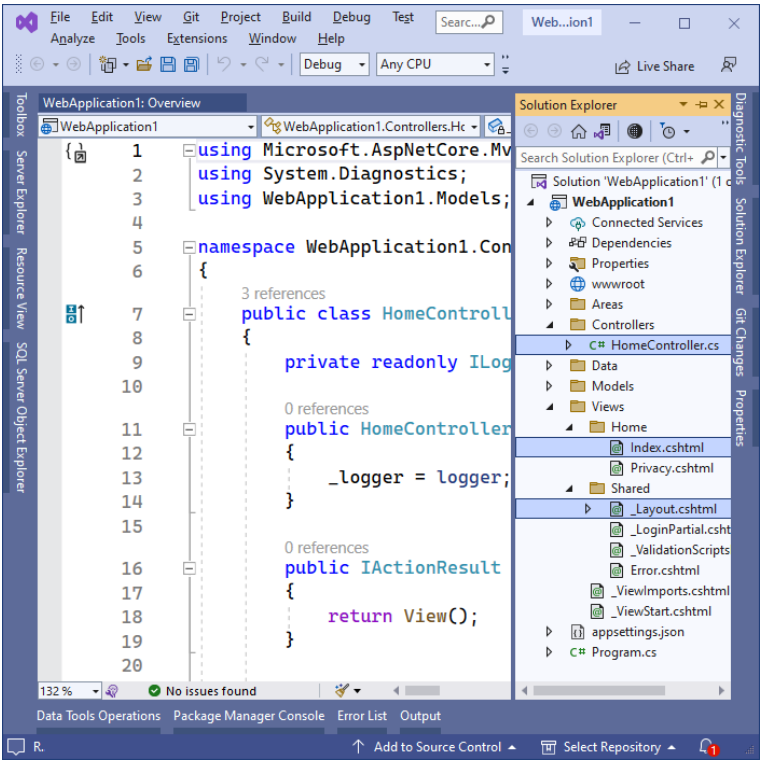


Рис. 5. Вікно *Solution Explorer* із відміченими файлами *_Layout.cshtml*, *Index.cshtml*, *HomeController.cs*

Додаткові матеріали

1. https://www.youtube.com/watch?v=AikUQpFUU-4&list=PLYn-WYVrzkRzhCxZXrvKw8CObmi3qIquX&index=2.
2. https://www.youtube.com/watch?v=phyV-OQNeRM&list=PLLWMQd6PeGY1C3YLCyAUIDJrPS9S3WdLK&index=11.
3. https://www.youtube.com/watch?v=vRqz_zUiJTw&list=PLLWMQd6PeGY1C3YLCyAUIDJrPS9S3WdLK&index=11.

Контрольні запитання

1. Що означає абревіатура *ASP*?
2. Що означає абревіатура *MVC*?
3. Що таке *Bootstrap*?
4. Що потрібно зробити, щоб після створення проекту були вже інтегровані шаблони реєстрації та автентифікації?
5. Яка основна перевага програмного забезпечення версії *Community*?
6. Які помпоненти бажано встановити для коректної роботи з фреймворком *ASP.NET Core MVC*?
7. За що відповідає пункт *Do not use top-level statements* при створенні проекту?
8. За що відповідає файл *_Layout.cshtml*?
9. Яка різниця між розширенням файлів **.cs, *.html* та **.cshtml*?
10. Як встановити локальну базу даних в *Microsoft Visual Studio 2022*?
11. Як змінити вигляд для *header* веб-сайту?
12. Які файли відповідають за *content* веб-сайту?
13. Як перевірити версію *Bootstrap* у створеному проекті?
14. Які можливі проблеми при використанні класу, який не відповідає поточній версії *Bootstrap*?
15. Як зрозуміти який контролер відповідає конкретному представленню?
16. Як видалити пункт меню *Privacy* з веб-сайту?
17. Де за замовчуванням зберігаються дані про зареєстрованих користувачів?
18. Як створити першу локальну базу даних?
19. Яка різниця між файлами з розширеннями **.cs* та **.cshtml*?
20. Як створити власні *css*-класи та інтегрувати їх у представлення?

Лабораторна робота №2
Патерн MVC

Мета

Навчитись створювати власні модель, контролер і представлення та реалізовувати відповідну передачу даних між цими елементами.

Завдання

Доповнити проект, розроблений у лабораторній роботі №1, наступним:

1. *(5% балів)* Створіть новий пункт у навігаційному меню.
2. *(20% балів)* Створіть контролер та представлення, що відповідають новому пункту меню.
3. *(15% балів)* Передайте число або рядок з контролера у представлення.
4. *(20% балів)* Створіть клас-модель, яка відповідала б Вашому варіанту у лабораторній роботі №1 (або узгодьте власну тему з викладачем).
5. *(15% балів)* Здійсніть передачу моделі з контролера на представлення.
6. *(25% балів)* Здійсніть передачу моделі з представлення на контролер.

Вимоги та рекомендації до програмної реалізації

Абревіатура *MVC* розшифровується як *Model-View-Controller* (модель-представлення-контролер). По-суті це патерн, тобто одна із хороших та ефективних практик розробки програмного забезпечення. Спрощено кажучи, механізм *MVC* полягає у тому, що модель (тобто деяка структура даних) передається від контролера до

представлення та, після певного впливу на дані, повертається у якусь із дій контролерів (рис. 6).

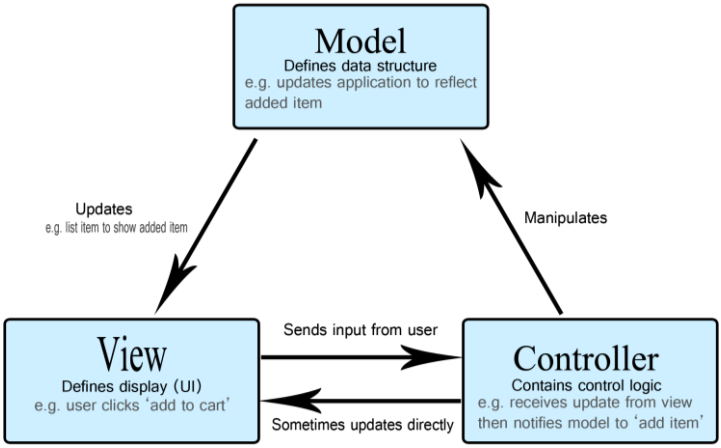


Рис. 6. Схематичне зображення патерна *MVC*

Створення нового пункту навігаційного меню виконується у файлі *_Layout.cshtml*. Найзручніше – це скопіювати існуючий пункт та підправити атрибути *asp-controller*, *asp-action* і текст між відкриваючим за закриваючим тегом.

Створення нового контролера доцільно виконувати одним із наступних способів:

1. Натиснути правою кнопкою мишкі на папку *Controller*, обрати *Add => Controller… => MVC Controller - Empty*, дати назву виду *CustomController.cs*.
2. Здублювати існуючий контролер та змінити тексти окремих рядків та назв.

Створення нового представлення доцільно виконувати одним із наступних способів:

1. Створити папку виду *Custom* (важливо, щоб ця назва була пов'язана з назвою відповідного контролера) у папці

View. Натиснути правою кнопкою мишкі на *Custom* і обрати *Add => View => Razor View - Empty*. Надати файлу назву *Index.cshtml*.

2. Здублювати папку *Home* у папці *View* та здійснити відповідні перейменування.

3. У файлі контролера натиснути правою кнопкою мишкі на слові *View*, яке знаходиться у методі *Index* та обрати *Add View… => Razor View - Empty*. Надати файлу назву *Index.cshtml*.

Передача даних від контролера до представлення здійснюється, наприклад, наступним чином:

1. У дужки функції *View()* дії *Index* контролера *Custom* поміщаємо модель (у випадку рядків потрібна явна типізація до типу *object*).

2. У перших рядках представлення вказуємо *@model тип_даних*. Тут *тип_даних* має бути сумісним з даними, які надсилаються з відповідного контролера.

Зворотну передачу даних (від представлення до контролера) можна здійснювати для різних видів моделі, наприклад, використовуючи наступний синтаксис:

1.
```
<form method="post">
        <input type="text" asp-for="@Model.Id">
        <button type="submit">Edit phone</button>
</form>
```
2.
```
<form method="post">
        <input type="text" asp-for="@Model">
        <button type="submit">Send</button>
</form>
```
3.
```
<a asp-route-phoneId="@Model.Id" asp-action="EditPhone">Edit phone</a>
```

Створення моделі здійснюється за допомогою класів. Наприклад:

```
public class Person
{
        public int Id { get; set; }
        public string Name { get; set; }
}
```

Тут *public* – це модифікатор доступу, *class* – ключове слово, *Person* – назва моделі, *Змінна { get; set; }* – властивість (властивості оголошуються з великої букви, після чого у фігурних дужках вказуються методи доступу).

Додаткові матеріали

1. https://www.youtube.com/watch?v=Ix2VE2xT54E&list=PLgRlicSxjeMP76FRmvJ6dBahsLT3jJw4B&index=3.
2. https://www.youtube.com/watch?v=phyV-OQNeRM&list=PLLWMQd6PeGY1C3YLCyAUIDJrPS9S3WdLK&index=11.

Контрольні запитання

1. Як розшифровується абревіатура *MVC*?
2. Що таке патерн?
3. Яка роль моделі в концепції *MVC*?
4. Яка роль представлення в концепції *MVC*?
5. Яка роль контролерів в концепції *MVC*?
6. У якому файлі доцільно змінювати *footer*?
7. Чим відрізняється процедура створення контролера від процедури створення представлення?
8. Як передати значення рядкової змінної з контролера на представлення?
9. Як передати модель з представлення на контролер?
10. Як створити нову модель?
11. Чим відрізняється поле від властивості?
12. Що таке модель?

13. Як передати рядок у представлення?

14. Як передати рядок у контролер?

15. Чому варто уникати використання полів у класі-моделі?

16. Чим відрізняється синтаксис організації циклів у файлах з розширеннями **.cs* та **.cshtml*?

17. Як вивести символ ета (тобто @) на сторінці веб-сайту?

18. Як вивести масив елементів у вигляді маркованого списку?

19. Як в представленні пов’язати текстове поле з моделлю?

20. Для чого доцільно використовувати значення атрибуту *hidden* у файлі-представленні?

Лабораторна робота №3
Командна розробка веб-застосунків

Мета

Навчитись на практиці основам командної розробки з сумісним використанням *GitHub* та *Jira*.

Завдання

1. Визначтесь із темою командного проекту. Це може бути у відповідності до варіанту, наведеного у лабораторній роботі №1, запропонована викладачем тема або власна тема.
2. Сформуйте команду з 2-3 людей для подальшого формулювання та виконання проекту.
3. Зареєструйтесь на *GitHub* (https://github.com/) та *Jira* (https://www.atlassian.com/software/jira).
4. Встановіть програму *GitHub Desktop* із офіційного сайту *GitHub*: https://desktop.github.com/.
5. *(25% балів)* Налаштуйте *GitHub* та *Jira* під командну розробку конкретного проекту.
6. *(25% балів)* Помість хоча б 10 задач у *Jira* та задайте коректні *Story points*.
7. *(25% балів)* Знайдіть на *GitHub* підходящий шаблон веб-сайту, що включає *Bootstrap*, та інтегруйте це у проект.
8. *(25% балів)* За допомогою *GitHub Desktop* виконайте 2 «заливання» проекту на *GitHub* для тестування командної роботи з репозиторієм і *Pull requests* на *GitHub*, з одного боку, та *Sprint* на *Jira*, – з іншого.

Вимоги та рекомендації до програмної реалізації

При виборі теми проекту зважайте на те, що вона могла би стати хорошим фундаментом для подальшого написання кваліфікаційної магістерської роботи. А, отже, рекомендується, щоб тема була актуальною, очікувані

результати мали наукову та практичну новизну і задовольняли критеріям кваліфікації спеціальності.

Для налаштування *GitHub* під командну розробку рекомендується виконати наведену нижче послідовність кроків.

1. Усім учасникам команди потрібно зареєструватись на веб-сайті https://github.com/.
2. Хтось із членів команди повинен створити спільний командний репозиторій, натиснувши *Create Repository*. Бажано, щоб це зробив *TeamLead*.
3. Далі потрібно надати ім’я репозиторію, опис та видимість.
4. Завершення створення репозиторію здійснюється після натиснення на кнопку Create repository.
5. У якості наступного кроку власник репозиторію має натиснути на *Invite collaborators => Add people*, де добавити члена/членів майбутньої команди. Члени команди мають погодити запрошення.
6. Для здійснення подальших налаштувань необхідно ініціалізувати репозиторій будь-якими файлами.
7. Далі потрібно перейти до *Code => Branches => New branch*, де задати ім’я вітки: *develop*.
8. Для даного репозиторію потрібно задати за замовчуванням вітку *develop* (*Settings => Default branch => Switch to another branch => develop => Update*).
9. З метою заборони самовільного внесення змін у код та забезпечення інших важливих налаштувань потрібно для даного репозиторію спочатку перейти до *Settings => Branches => Add branch protection rule*. Тут задати для *Branch name pattern* ім’я *main*, зробити активними пункти *Require a pull request before merging* та *Require conversation resolution before merging*.

Наведених вище кроків могло би бути достатньо для забезпечення командної розробки. Але для підвищення комфорту роботи бажано встановити програму *GitHub Desktop* та організувати роботу хоча б з *Kanban*-дошкою. Останнє (і навіть більше) можна забезпечити з використанням *Jira*. Для налаштування *Jira* під командну розробку бажано виконати наведену нижче послідовність кроків.

1. Усім членам команди зареєструватись на веб-сайті https://www.atlassian.com/software/jira.
2. Обрати іконку *Jira Software* => *Get it free* => *Next*. Далі ввести будь-яку назву сайту (або залишити запропоновану) та натиснути *Agree*. Після чого натиснути на *Continue* та відповісти на запитання щодо цілей використання *Jira*.
3. Для забезпечення зручної роботи з *Jira, TeamLead* має обрати шаблон *Scrum*. Це дуже важливий пункт.
4. Далі потрібно відповісти на запитання щодо досвіду використання *Jira*. Після цього натиснути на кнопку *Finish*. Результуюча веб-сторінка повинна мати вигляд, зображений на рис. 7.
5. Важливим є налаштувати проект під комфортну подальшу роботу. У *Project settings* бажано зробити наступне:
   - знайти пункт *Details* і задати нове ім’я проекту та відповідну скорочу назву (*Key*);
   - знайти пункт *Features* та зробити активними *Reports* та *Issue navigator*.
6. Добавити інших членів команди у проект.

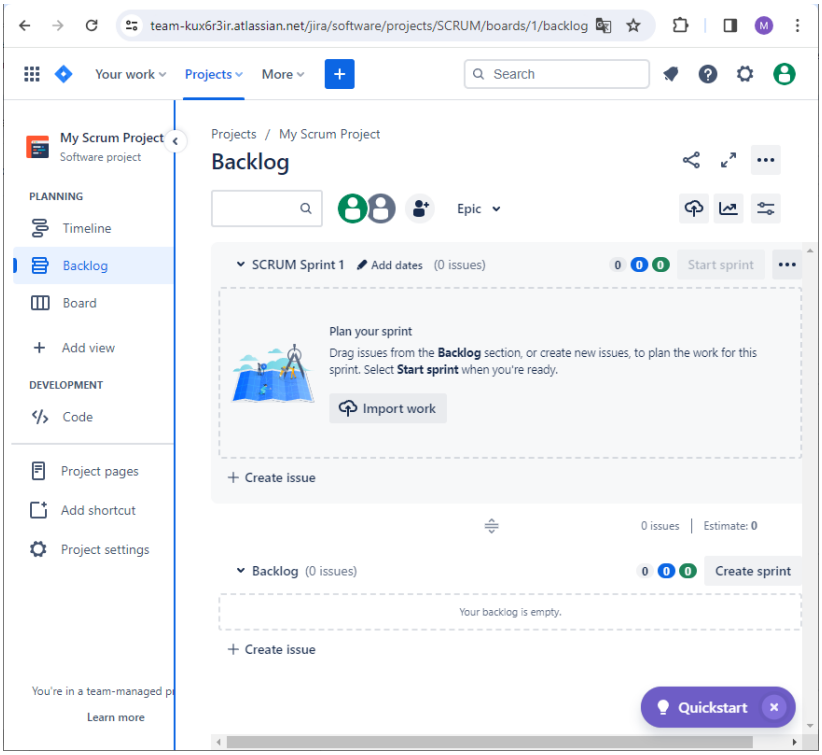


Рис. 7. Приклад зовнішнього вигляду новоствореного проекту в Jira

Ще більш комфортне сумісне використання *GitHub* та *Jira* можна організувати, виконавши наведену нижче послідовність кроків.

1. Стосовно конкретного репозиторію на *GitHub* перейдіть до *Settings => GitHub Apps => Configure =>* https://github.com/marketplace/jira-software-github *=> Install it for free =>* вкажіть Ваші дані *=> Save => Complete order and begin installation.*

2. У вкладці *Code* в *Jira* оберіть *Connect to other providers => GitHub for Jira => Get it now => check the app*

configuration => Continue => Next => Authorize Jira => Only selected repositories => обрати репозиторій => *Install*.

Важливо ще під час планування проекту наповнити вкладку *Backlog* у *Jira* низкою задач. Задачі можуть бути, наприклад, такі: створити порожній проект *ASP.NET Core MVC*, знайти в інтернеті шаблон веб-сторінки, побудувати скетч проекту, створити певні веб-сторінки тощо.

Задання *Story points* є непростою задачею. Рекомендується зіграти кілька разів у *PlanningPoker* (https://planningpokeronline.com/); варто, наприклад, врахувати, що задача складності понад 20 *Story points* повинна бути розбитою на кілька підзадач.

При пошуку на *GitHub* підходящого шаблону веб-сайту варто врахувати низку нюансів. Наприклад, якщо необхідно знайти сайт на специфічну тему з продажу павуків, доцільно ввести більш загальний запит. Зокрема електронний магазин збуту телефонів неважко буде адаптувати під специфічні потреби з продажу павуків. Також варто враховувати вид ліцензії, під яку викладено веб-сайт. Загалом запит може мати вигляд: *phone shop language:HTML bootstrap*.

Загальний цикл виконання задачі, пов'язаної з внесенням змін у код, може мати наведений нижче вигляд.

1. Певний учасник команди підв'язується під конкретну задачу *Sprint*'а, наведену у *Jira*.
2. Переносить задачу з *TO DO* в *IN PROGRESS*.
3. Виконує задачу.
4. У вікні *GitHub Desktop => Branch => New branch…* створює нову вітку виду *JiraTaskId/task/shortDescription*, в яку потрібно, щоб одночасно скопіювались усі зміни.
5. Вносить коментарі до виконаної задачі (опис змін) та натискає кнопку *Commit to JiraTaskId/task/shortDescription*.
6. Натискає на кнопку *Publish Branch*.

7. Заходить на сайт https://github.com/ та у вкладці *Pull requests* створює новий *Pull request*.

8. Члени команди перевіряють перелік внесених змін та, у позитивному випадку, натискають *Merge pull request => Confirm merge*. У протилежному випадку доведеться внести зміни у код.

9. Після перевірки будь-який член команди може натиснути *Confirm merge* для остаточного «заливання» коду на репозиторій *GitHub*. Одночасно у *Jira* варто перенести задачу з *IN PROGRESS* в *DONE*.

Можливі також неприємні випадки конфлікту коду, які вирішуються, наприклад, у процесі детального обговорення між членами команди.

Додаткові матеріали

1. https://www.youtube.com/watch?v=XxIGdL0HsUw&list=PLqWD37Mj2te2xeDcFWb4Vw1pvX5gHfWgy.
2. https://www.youtube.com/watch?v=u6RsQmlX4j0.
3. https://www.youtube.com/watch?v=yhisa9QSNO0.

Контрольні запитання

1. Яким чином доцільно використовувати *GitHub* при командній розробці програмного забезпечення?
2. Яким чином доцільно використовувати *Jira* при командній розробці програмного забезпечення?
3. Для чого призначена програма *GitHub Desktop*?
4. Яка різниця між *Pull requests* та *GitHub* та *Sprint*?
5. Як добавити вкладку *Reports* у *Jira*?
6. Яка користь з гри *PlanningPoker*?
7. Який «життєвий цикл» задачі, поставленої в *Jira*?
8. Як технічно вирішити проблему конфлікту кодів у GitHub?

9. Яким чином доцільно шукати шаблони сайтів на GitHub?

10. Задачі якої складності доцільно розбивати на менші за розміром?

11. Яка різниця між *Story* та *Story Points* в *Jira*?

12. У скільки *Story Points* Ви би оцінили подорож до м. Львів у контексті подорожі в межах Європи?

13. Який принцип формування *Story Points*?

14. Чим відрізняються *Bug*, *Story* та *Task*?

15. Як пов'язати *GitHub* та *Jira*?

16. Яка логіка формування віток в *GitHub* при командній розробці?

17. Які параметри потрібно налаштувати в *GitHub* для командної роботи?

18. Які переваги використання *Jira* в порівнянні з *GitHub Projects*?

19. Для чого потрібен файл **.gitignore*?

20. Як побачити історію комітів вітки *develop*?

Лабораторна робота №4
CRUD операції

Мета

Здобути навички роботи з базою даних засобами *ORM Entity Framework*.

Завдання

1. *(10% балів)* Оберіть *TeamLider*'а та представте скетч проекту.
2. *(20% балів)* В межах обраного проекту створіть схему бази даних, яка складатиметься хоча б з трьох пов'язаних таблиць.
3. *(20% балів)* Засобами об'єктно-орієнтованого програмування опишіть структуру бази даних та її ініціалізацію.
4. *(5% балів)* Виконайте міграцію.
5. *(35% балів)* На веб-сайті реалізуйте роботу з усіма чотирма *CRUD*-операціями.
6. *(10% балів)* Реалізуйте фільтрацію даних на веб-сторінці.

Вимоги та рекомендації до програмної реалізації

У якості *TeamLeader*-а рекомендується обирати студента, який найкраще уявляє суть проекту. Саме такий учасник команди найбільш точно орієнтуватиметься яких задач не вистачає, в якій послідовності варто їх додати в *Jira* та формуватиме схему даних.

Опис бази даних здійснюється у класі *ApplicationDbContext* засобами об'єктно-орієнтованого програмування та *ORM Entity Framework*. Створення нових таблиць забезпечується властивостями типу *DbSet<Модель>*. Властивості моделі визначають перелік

атрибутів таблиці бази даних. Ініціалізація даними реалізується у методі *OnModelCreating*.

Після проведення опису бази даних доцільно виконати міграцію послідовністю команд: *add-migration назва*, *update-database*.

Виконання *CRUD*-операцій забезпечується класом *ApplicationDbContext*. Змінні цього класу мають доступ до ключових методів *Add(object)*, *ToList()*, *Update(object)*, *Remove(object)* для роботи з базою даних.

Додавання та редагування даних зазвичай потребує зміни значень кількох полів бази даних. Тому для цього варто здійснювати перехід на окрему попередньо підготовлену сторінку за допомогою методу *get*. Далі, за допомогою методу *post* та тегу *form*, дані повинні передаватись на дію контролера, де виконуватимуться оновлення в базі даних. Варто пам'ятати про атрибути *[HttpGet]* та *[HttpPost]*.

Видалення потребує використання лише методу *get* без додаткового створення веб-сторінок.

В усіх трьох випадках варто не забувати виконувати збереження даних у базі даних за допомогою методу *SaveChanges()*.

Додаткові матеріали

1. https://www.youtube.com/watch?v=Bu1oGMDgXuY&list=PLz3T2C4dYvQKkYsJA5GU7fzHHLa-E0sPT&index=1.
2. https://www.youtube.com/watch?v=BWOcba-XfcM&list=PLz3T2C4dYvQKkYsJA5GU7fzHHLa-E0sPT&index=2.
3. https://www.youtube.com/watch?v=VPWwDowbq5A&list=PLz3T2C4dYvQKkYsJA5GU7fzHHLa-E0sPT&index=3.
4. https://www.youtube.com/watch?v=DtLbKJogQgA&list=PLz3T2C4dYvQKkYsJA5GU7fzHHLa-E0sPT&index=4.

5. https://www.youtube.com/watch?v=N3EOjMJ63-I&list=PLz3T2C4dYvQKkYsJA5GU7fzHHLa-E0sPT&index=5.

Контрольні запитання

1. Що означає абревіатура CRUD?
2. Що тут мається на увазі ORM EF?
3. Якими якостями повинен володіти TeamLeader?
4. Що таке властивості у C#?
5. Що таке навігаційні властивості?
6. Яке призначення класу *ApplicationDbContext*?
7. Для чого призначені властивості виду *DbSet<Модель>*?
8. Як виконати міграцію?
9. Які методи фреймворка Entity Framework призначені для виконання *CRUD*-операцій?
10. Які існують атрибути дій при роботі з формами?
11. Для чого призначений метод *SaveChanges()*?
12. Що обов'язково потрібно врахувати при додаванні нового рядка у базу даних при використанні *Entity Framework*?
13. Яка різниця між кодами для виконання *Update* та *Add*?
14. Яка різниця між кодами для виконання *Delete* та *Add*?
15. Що є лишнім при виконанні *Read* у порівнянні з рештою *CRUD* операцій?
16. Яка різниця у використанні *дій* при роботі з методами *Get* та *Post*?
17. Як передати дані з представлення в контролер за допомогою тега *<a>*?
18. У якому випадку *context.Update(phone)* та *context.Add(phone)* будуть працювати однаково?
19. Яка різниця між *add-migration* та *update-database*?
20. Для чого призначений метод *HasData()*?

Лабораторна робота №5
Пошарова архітектура. Сервіси

Мета

Навчитись будувати пошарову архітектуру програмного забезпечення.

Завдання

1. *(10% балів)* Представте ступінь виконання проекту.
2. *(40% балів)* Структуруйте Ваш проект у відповідності до *Layer Architecture*. Допускається часткове залишення бізнес-логіки у проекті *UI*.
3. *(50% балів)* Створіть 2 сервіси та відповідні їм 2 контролери.

Вимоги та рекомендації до програмної реалізації

При представленні проекту потрібно показати яким чином реалізовано *CRUD* операції. Очікується, що кожен студент висловить хоча б декілька слів англійською мовою. Викладач може попросити показати наповненість *Jira* та *GitHub*. Під час представлення можуть задавати різного роду технічні запитання щодо подальшої веб-розробки.

Layer Architecture (пошарова архітектура) програмного забезпечення – це вид поділу проекту на логічно-незалежні складові, пов'язані між собою лінійно, як однозв'язний ланцюг (рис. 8). Рекомендується поділити розроблюваний проект на 3 частини: *BusinessLogic*, *DataAccess* та *UI*. Повинне бути взаємне розділення бізнес-логіки, налаштування зв'язків з базою даних та візуальної складової. Логічно було б, щоб проект *BusinessLogic* користувався лише тим, що є в *DataAccess*, а проект UI, в свою чергу, мав доступ до *BusinessLogic* та, транзитивно, – до *DataAccess*. Зворотного зв'язку не повинно бути.

Очевидно, що у процесі перенесення класів між проектами доведеться встановити низку пакетів, які є у початковому проекті, перейменувати простори імен тощо. Варто пам'ятати рекомендацію, що назва простору імен формується за наступним принципом: *Назва_проекту.Назва_папки.Назва_підпапки…Назва_підпапки*. Встановлення ж пакетів здійснюється за допомогою менеджера *Manage NuGet Packages…* (рис. 9), вікно якого можна знайти у контекстному меню проектів, перелічених в *Solution Explorer*.

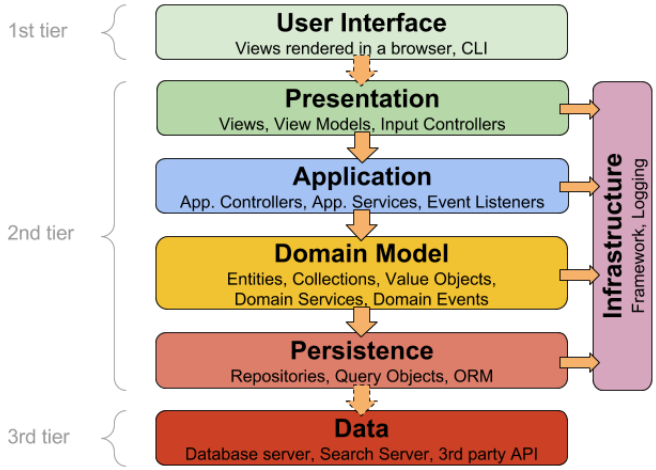


Рис. 8. Схематичне зображення пошарової архітектури програмного забезпечення

Для пошарової архітектури програмного забезпечення характерною є наступна структура:

- проект *UI*: папки *Areas Controllers*, *Models*, *Properties*, *Views*, *wwwroot*; файли *Program.cs*, *appsetting.json*;
- проект *BusinessLogic*: папка *Services*;

- проект *DataAccess*: папки *Entities*, *Migrations*; файл *ApplicationDbContext*.

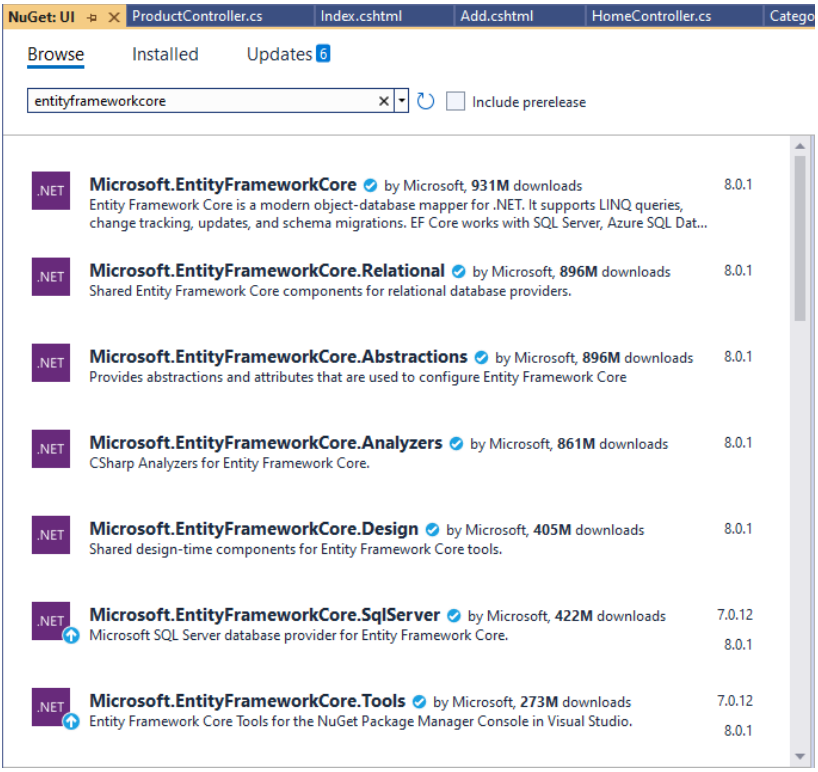


Рис. 9. Вікно менеджера пакетів *Manage NuGet Packages*

Створення сервісу передбачає додавання у проект *BusinessLogic* папки *Services* та поміщення туди нового файлу з назвою виду *HomeService.cs*. У файлі має бути клас, який складається з конструктора та низки методів, у певному сенсі перенесених з *UI*. Потрібно пам'ятати, що вся бізнес-логіка повинна бути в *BusinessLogic*, а не в *UI*. Конструктор класу *HomeService* повинен отримувати екземпляр класу *ApplicationDbContext*. Цей екземпляр має

створюватись за рахунок використання дизайн-патерну *Dependency Injection*. Простіше кажучи потрібно у код файлу *Program.cs* внести рядок виду:

```
builder.Services.AddScoped<HomeService>();
```

Додаткові матеріали

1. https://www.youtube.com/watch?v=Zpq21ErM4ZA.
2. https://www.youtube.com/watch?v=vpEHvvroNoA.
3. https://www.youtube.com/watch?v=43fg4T4iFGw.

Контрольні запитання

1. Що означає *Layer Architecture*?
2. Які переваги *Layer Architecture*?
3. Яка оптимальна кількість сервісів у проекті?
4. Яка орієнтовна структура пошарової архітектури програмного забезпечення?
5. Навіщо потрібен конструктор в сервісі?
6. Яку роль грає *Dependency Injection* при створенні екземпляра сервісу?
7. Яка роль сервісів у додатку?
8. Які пакети потребуються у проекті *BusinessLogic*?
9. Яким чином можна доступитись до проекту *DataAccess* з проекту *UI*?
10. Який варто обрати шаблон при створенні проекту *DataAccess*?
11. Що таке веб-сервіс?
12. Де повинні розміщуватись моделі у додатку, в якому реалізується *Layer Architecture*?
13. Які методи є характерними для веб-сервісу?
14. Де в проекті варто розміщувати алгоритм фільтрації назва товарів?

15. Що необхідно вказувати в *Dependencies* усіх проектів при *Layer Architecture*?

16. Яким чином вказати для *Dependency Injection*, щоб він створював екземпляр сервісу при потребі?

17. За яким принципом варто формувати назву простору імен?

18. Яка послідовність виклику класів має місце при натисненні на кнопку для видалення товару у *Layer Architecture*?

19. Яка виникає складність у перейменуванні простору імен у папці *Migrations*?

20. Яка виникає незручність у створенні міграцій, якщо файл *ApplicationDbContext.cs* перенесено у проект *DataAccess*?

Рекомендована література

Основна

1. Lock A. ASP.NET Core in Action. 3rd ed. Shelter Island : Manning, 2023. 984 p.
2. Freeman A. Pro ASP.NET Core 7. 10th ed. Shelter Island : Manning, 2023. 1256 p.
3. Danylko J.R. ASP.NET 8 Best Practices: Explore techniques, patterns, and practices to develop effective large-scale. NET web apps. Birmingham : Packt Publishing, 2023. 256 p.
4. Smith J.P. Entity Framework Core in Action. 2nd ed. Shelter Island: Manning, 2021. 624 p.
5. Мартін Р. Чистий Agile. Харків : Фабула, 2021. 224 с.
6. Омельчук Л. Л., Русіна Н. Г. Інструментальні середовища та технології програмування : лабораторний практикум. Одеса : Айс Принт, 2020. 175 с.
7. Мартін Р. Чистий код. Створення, аналіз і рефакторинг. Харків : Фабула, 2019. 368 с.
8. Troelsen A., Japikse P. Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming. 11th ed. New York : Apress, 2022. 1705 p.
9. Richter J. CLR via C#. 4th ed. Redmond : Microsoft Press, 2012. 1594 p.

Додаткова

10. Коноваленко І. В. Платформа .NET та мова програмування C# 8.0 : навчальний посібник. Тернопіль : ФОП Паляниця В. А., 2020. 320 с.
11. Freeman A. Pro ASP.NET Core 6: Develop Cloud-Ready Web Applications Using MVC, Blazor, and Razor Pages. 9th ed. New York : Apress, 2022. 1286 p.
12. Marcotte C.-H. An Atypical ASP.NET Core 6 Design Patterns Guide. 2nd ed. Birmingham : Packt Publishing, 2022. 678 p.

13. Shellman M., Afyouni H., Pratt P.J., Last M.Z. A Guide to SQL. 10th ed. Kentucky : Cengage Learning, 2020. 320 p.

14. Bootstrap · The most popular HTML, CSS, and JS library in the world. URL: https://getbootstrap.com/ (Last accessed: 02.06.2024).

15. ASP.NET Core | Open-source web framework for .NET. URL: https://dotnet.microsoft.com/en-us/apps/aspnet (Last accessed: 02.06.2024).

16. Introduction to Identity on ASP.NET Core | Microsoft Learn. URL: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio (Last accessed: 02.06.2024).

17. Manifesto for Agile Software Development. URL: https://agilemanifesto.org/ (Last access: 02.06.2024).

Корисні посилання

18. Михайло Володимирович Бойчура - YouTube. URL: https://www.youtube.com/@mvboichura (Last accessed: 02.06.2024).

19. Introduction to ASP.NET MVC in C#: Basics, Advanced Topics, Tips, Tricks, Best Practices, and More - YouTube. URL: https://www.youtube.com/watch?v=phyV-OQNeRM&list=PLLWMQd6PeGY1C3YLCyAUIDJrPS9S3WdLK&index=11 (Last accessed: 02.06.2024).

20. An Introduction To The C# Learning Cycle - YouTube. URL: https://www.youtube.com/watch?v=h7aIzCkmbl8&list=PLLWMQd6PeGY2GVsQZ-u3DPXqwwKW8MkiP (Last accessed: 02.06.2024).

21. Clean Architecture with ASP.NET Core 6 - YouTube. URL: https://www.youtube.com/watch?v=lkmvnjypENw (Last accessed: 02.06.2024).

22. Jira Tools - YouTube. URL: https://www.youtube.com/playlist?list=PLqWD37Mj2te0xiDW-Q58ZIG7XZry-1-bk (Last accessed: 02.06.2024).