Міністерство освіти України

Національний технічний університет "ХПІ"

кафедра "Інформатики та інтелектуальної власності"

**Звіт**

**Лабораторна робота 3**

з дисципліни "Проєктування серверних застосунків"

Виконав: студент групи КН-922В

Шершньов М.О.

Перевірив:

Івашко А.В.

Харків 2025

Зміст

[Завдання 3](#_Toc193187869)

[Класи request, iformcollection, weaktypification, stricttypification, mathtask 5](#_Toc193187870)

[Клас Request 5](#_Toc193187871)

[Клас IFormCollection 6](#_Toc193187872)

[Клас WeakTypification 7](#_Toc193187873)

[Клас StrongTypification 7](#_Toc193187874)

[Клас MathTask 8](#_Toc193187875)

[Висновок 9](#_Toc193187876)

**Шляхи використання можливостей форм у засобах ASP.NET MVC WEB-застосунків. Частина 2.**

**Мета роботи:** набути практику та досвід у використанні різноманітних шля хів отримання даних із форм на поданнях під час розробки та використання MVC WEB-застосунків.

Завдання

1. Створити застосунок, стартове подання (Index) якого містить форму згідно до власного варіанту. За допомогою стандартного класу Request контролер, викликаний кнопкою Відіслати приймає дані з форми та передає їх назад до подання для їхньої демонстрації під формою, кнопка Скинути має повертати форму у «дефолтний» (початковий) стан, тобто очистити текстові поля

Мій варіант:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

1. Виконати функції минулого завдання, використавши для отримання даних форми замість класу Request колекцію типу IFormCollection
2. Виконати функції першого завдання, використавши для отримання да-них форми замість класу Request слабку типізацію форми, додавши до кожного елементу форми відповідну змінну.
3. Виконати функції першого завдання, використавши для отримання даних форми замість класу Request сувору типізацію форми, додавши до елементів форми відповідну модель
4. Створити застосунок із поданням, що містить форму та розробити контролер з вирішення геометричної задачі згідно свого варіанту. У випадку введення до полів форми некоректних даних вивести відповідне повідомлення

Мій варіант:





Класи для роботи:

1. «Запит» - **Request**
2. «Колекція» – **IFormCollection**
3. «Слабка типізація» – **WeakTypification**
4. «Сувора типізація» – **StrongTypification**
5. «Математична задача» – **MathTask**

Класи request, iformcollection, weaktypification, stricttypification, mathtask

Клас Request

**Request** – це об'єкт у Controller, який містить усю інформацію про HTTP-запит (GET, POST, Cookies, Headers тощо).

Можна отримувати параметри форми (Request.Form), параметри URL (Request.Query) та заголовки запиту (Request.Headers).

using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult IndexPost()

{

var name = Request.Form["name"].ToString() ?? string.Empty;

var phone = Request.Form["phone"].ToString() ?? string.Empty;

var email = Request.Form["email"].ToString() ?? string.Empty;

var visitDate = Request.Form["visitDate"].ToString() ?? string.Empty;

var age = int.TryParse(Request.Form["age"], out int parsedAge) ? parsedAge : 0;

var favoriteCuisine = Request.Form["favoriteCuisine"].ToString() ?? string.Empty;

var desiredDishes = Request.Form["desiredDishes"].ToString() ?? string.Empty;

var reasonForChoosing = Request.Form["reason"].ToString() ?? string.Empty;

var recommendation = Request.Form["recommendation"].ToString() ?? string.Empty;

ViewData["Name"] = name;

ViewData["Phone"] = phone;

ViewData["Email"] = email;

ViewData["VisitDate"] = visitDate;

ViewData["Age"] = age;

ViewData["FavoriteCuisine"] = favoriteCuisine;

ViewData["DesiredDishes"] = desiredDishes;

ViewData["ReasonForChoosing"] = reasonForChoosing;

ViewData["Recommendation"] = recommendation;

return View("Index");

}

}

Клас IFormCollection

**IFormCollection –** це колекція всіх даних POST-форми, яка дозволяє отримувати значення у вигляді Dictionary<string, StringValues>.

Використовується для отримання всіх полів форми одним параметром.

using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult Index(IFormCollection form)

{

ViewData["Name"] = form["name"].ToString() ?? string.Empty;

ViewData["Phone"] = form["phone"].ToString() ?? string.Empty;

ViewData["Email"] = form["email"].ToString() ?? string.Empty;

ViewData["VisitDate"] = form["visitDate"].ToString() ?? string.Empty;

ViewData["Age"] = int.TryParse(form["age"], out int age) ? age : 0;

ViewData["FavoriteCuisine"] = form["favoriteCuisine"].ToString() ?? string.Empty;

ViewData["DesiredDishes"] = form["desiredDishes"].ToString() ?? string.Empty;

ViewData["ReasonForChoosing"] = form["reason"].ToString() ?? string.Empty;

ViewData["Recommendation"] = form["recommendation"].ToString() ?? string.Empty;

return View();

}

}

Клас WeakTypification

Слабка типізація означає, що дані передаються у вигляді string без явного зв’язку з типом.

Використовується Request.Form або IFormCollection.

using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult Index(

string name, string phone, string email, string visitDate, string age,

string favoriteCuisine, string desiredDishes, string reason, string recommendation)

{

ViewData["Name"] = name ?? string.Empty;

ViewData["Phone"] = phone ?? string.Empty;

ViewData["Email"] = email ?? string.Empty;

ViewData["VisitDate"] = visitDate ?? string.Empty;

ViewData["Age"] = int.TryParse(age, out int parsedAge) ? parsedAge : 0;

ViewData["FavoriteCuisine"] = favoriteCuisine ?? string.Empty;

ViewData["DesiredDishes"] = desiredDishes ?? string.Empty;

ViewData["ReasonForChoosing"] = reason ?? string.Empty;

ViewData["Recommendation"] = recommendation ?? string.Empty;

return View();

}

}

Клас StrongTypification

Сувора типізація використовує **класи-моделі** (Model), де чітко визначені типи полів.

Використовується asp-for у <input> для автоматичної прив'язки даних.

using Microsoft.AspNetCore.Mvc;

using Task\_4.Models;

public class HomeController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View(new SurveyModel());

}

[HttpPost]

public IActionResult Index([FromForm] SurveyModel model)

{

if (!ModelState.IsValid)

{

return View(model);

}

return View(model);

}

}

Клас MathTask

Клас **MathTask** обробляє HTTP-запити для сторінки Index. Він містить два методи:

Index() з атрибутом [HttpGet], який відображає сторінку.

Index(double radius, string calculationType) з атрибутом [HttpPost], який обробляє введені дані та виконує обчислення.

using Microsoft.AspNetCore.Mvc;

public class HomeController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult Index(double radius, string calculationType)

{

if (radius <= 0)

{

ViewData["Error"] = "Радіус має бути більше 0";

return View();

}

double result = calculationType == "Volume"

? (4.0 / 3.0) \* Math.PI \* Math.Pow(radius, 3)

: 4 \* Math.PI \* Math.Pow(radius, 2);

ViewData["Radius"] = radius;

ViewData["CalculationType"] = calculationType;

ViewData["Result"] = result;

return View();

}

}

Висновок

У ході виконання цієї лабораторної роботи я ознайомився з можливостями роботи з формами у ASP.NET MVC WEB-застосунках та навчився отримувати й обробляти введені користувачем дані різними способами.

Я реалізував обробку форм через контролери, використовуючи: Request.Form для безпосереднього отримання значень полів форми. IFormCollection для зручного доступу до всіх вхідних параметрів форми. Слабку типізацію (string, string[]) для передачі даних через параметри методу контролера. Сувору типізацію (моделі, наприклад, StrongTypificationModel) для структурованої передачі даних та їхньої валідації.

Також я використовував asp-for у Razor-розмітці для зв’язку елементів форми з моделями, що спростило передачу даних між поданням та контролером.

Окрім роботи з формами, я реалізував контролер для вирішення геометричної задачі, який обробляє введені значення, виконує необхідні розрахунки та виводить результат. При некоректному введенні даних контролер повертає повідомлення про помилку, що покращує взаємодію з користувачем.

Загалом, ця лабораторна робота допомогла мені освоїти принципи роботи з формами в ASP.NET MVC, навчила правильно інтегрувати їх із контролерами та вибирати найбільш підходящий метод обробки в залежності від вимог проєкту.