Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебный предмет
«Конструирование программ и языки программирования»

Инструкция
по выполнению лабораторной работы №41
«Исследование работы MVC-контроллеров. Создание MVC-page»

Минск 2025 г.

Лабораторная работа № 41

Тема работы: «Исследование работы MVC-контроллеров. Создание MVC-page»

1 Цель работы

Сформировать умения использовать MVC-контроллеры при разработке приложений, создавать MVC-page.

2 Задание

Номер варианта соответствует номеру по списку в журнале.

1. Создайте новое MVC-приложение в Visual Studio. Реализуйте контроллер HomeController с методом Index, который возвращает представление Index.cshtml с текстом "Welcome to MVC!"..
2. Создайте контроллер ProductController с действиями List и Details, где List возвращает список продуктов, а Details отображает информацию о конкретном продукте. Создайте соответствующие представления List.cshtml и Details.cshtml.
3. Реализуйте контроллер UserController с действиями Register и Login. Создайте представления Register.cshtml и Login.cshtml с формами для регистрации и входа.
4. Создайте контроллер BlogController с действиями Index, Create и Details. Реализуйте представления Index.cshtml, Create.cshtml и Details.cshtml для отображения блогов и формы создания нового блога.
5. Создайте контроллер ContactController с методом Index, который возвращает представление с формой обратной связи. Создайте представление Index.cshtml, включающее поля для имени, email и сообщения.
6. Реализуйте контроллер OrderController с действиями Index, Create и Confirm. Создайте представления Index.cshtml, Create.cshtml и Confirm.cshtml для работы с заказами.
7. Создайте контроллер CategoryController с действиями Index и Details. Реализуйте представления Index.cshtml для отображения списка категорий и Details.cshtml для подробной информации о выбранной категории.
8. Создайте контроллер ReviewController с действиями Index и Create. Создайте представления Index.cshtml для отображения отзывов пользователей и Create.cshtml для формы добавления нового отзыва.
9. Реализуйте контроллер GalleryController с действиями Index и Upload. Создайте представления Index.cshtml для отображения изображений и Upload.cshtml для формы загрузки новых изображений.
10. Создайте контроллер EventController с действиями Index, Create и Details. Реализуйте представления Index.cshtml для отображения предстоящих событий, Create.cshtml для создания нового события и Details.cshtml для подробной информации о событии.

3 Оснащение работы

ПК, среда Visual Studio 2019, MSword.

4 Основные теоретические сведения

Архитектура MVC (Model-View-Controller) является одним из наиболее популярных шаблонов проектирования для создания веб-приложений. Она разделяет приложение на три компонента, что упрощает поддержку и развитие кода.

Модель (Model) представляет данные приложения и бизнес-логику. Модели могут быть простыми классами, которые содержат свойства и методы для работы с данными, а также могут взаимодействовать с базой данных. Например, модель Product может выглядеть следующим образом:

```
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
    public decimal Price { get; set; }
}
```

Представление (View) отвечает за отображение данных пользователю. Оно берет данные из модели и форматирует их в удобочитаемом виде, обычно в формате HTML. Представления в ASP.NET MVC используют Razor-синтаксис, который позволяет внедрять C# код в HTML. Пример представления, отображающего список продуктов, может выглядеть так:

```
@model IEnumerable<Product>

<h2>Product List</h2>
<table>
    <tr>
        <th>Name</th>
        <th>Price</th>
    </tr>
    @foreach (var product in Model)
    {
        <tr>
            <td>@product.Name</td>
            <td>@product.Price.ToString("C")</td>
        </tr>
    }
</table>
```

Контроллер (Controller) является связующим звеном между моделью и представлением. Он обрабатывает входящие запросы от пользователя, взаимодействует с моделью и возвращает соответствующее представление. Контроллеры содержат методы (действия), которые соответствуют различным URL-адресам. Например, контроллер ProductController может выглядеть так:

```
public class ProductController : Controller
{
    private List<Product> products = new List<Product>
    {
        new Product { Id = 1, Name = "Product A", Price = 10.0m },
        new Product { Id = 2, Name = "Product B", Price = 20.0m }
    };

    public IActionResult List()
    {
        return View(products);
```

```
    }
}
```

Жизненный цикл запроса

Когда пользователь делает HTTP-запрос к приложению, ASP.NET MVC проходит через несколько этапов:

Маршрутизация: Запрос сопоставляется с маршрутом, который определяет, какой контроллер и метод должны быть вызваны. Это происходит в файле Startup.cs или RouteConfig.cs.

Создание контроллера: Создается экземпляр контроллера, соответствующего запросу.

Выполнение действия: Вызывается метод действия контроллера, который обрабатывает запрос.

Формирование представления: Данные, полученные от модели, передаются в представление, которое генерирует HTML-ответ.

Отправка ответа: Сформированный HTML возвращается пользователю.

Работа с формами и обработка данных

Для взаимодействия с пользователем часто используются HTML-формы. В ASP.NET MVC можно легко работать с формами, используя вспомогательные методы, такие как Html.BeginForm(). Пример формы для создания нового продукта может выглядеть следующим образом:

```
@model Product

<h2>Create Product</h2>
<form asp-action="Create" method="post">
    <div>
        <label asp-for="Name"></label>
        <input asp-for="Name" />
        <span asp-validation-for="Name"></span>
    </div>
    <div>
        <label asp-for="Price"></label>
        <input asp-for="Price" />
        <span asp-validation-for="Price"></span>
    </div>
    <button type="submit">Create</button>
</form>
```

В контроллере можно создать методы для обработки запросов GET и POST:

```
[HttpGet]
public IActionResult Create()
{
    return View();
}

[HttpPost]
public IActionResult Create(Product product)
{
    if (ModelState.IsValid)
    {
        // Сохранение продукта в базу данных (логика не показана)
        return RedirectToAction("List");
    }
    return View(product); // Вернуть форму с ошибками валидации
}
```

Валидация данных

ASP.NET MVC поддерживает валидацию данных с использованием атрибутов. Это позволяет автоматически проверять введенные данные перед их обработкой. Например, можно использовать атрибуты в модели для валидации полей:

```
public class Product
{
    [Required(ErrorMessage = "Name is required.")]
    public string Name { get; set; }

    [Range(0.01, double.MaxValue, ErrorMessage = "Price must be a
positive value.")]
    public decimal Price { get; set; }
}
```

При отправке формы, если данные не проходят валидацию, контроллер может вернуть представление с ошибками:

```
if (ModelState.IsValid)
{
    // Сохранение данных
}
return View(product); // Возврат модели с ошибками
```

Использование моделей

Модели в ASP.NET MVC могут быть связаны с базами данных с использованием Entity Framework или других ORM. Это позволяет работать с данными, не углубляясь в SQL-запросы. Пример использования модели с базой данных может выглядеть так:

```
public class ProductController : Controller
{
    private readonly ApplicationDbContext _context;

    public ProductController(ApplicationDbContext context)
    {
        _context = context;
    }

    public IActionResult Create()
    {
        return View();
    }

    [HttpPost]
    public IActionResult Create(Product product)
    {
        if (ModelState.IsValid)
        {
            _context.Products.Add(product);
            _context.SaveChanges();
            return RedirectToAction("List");
        }
        return View(product);
    }
}
```

5. Порядок выполнения работы

1. Выделить ключевые моменты задачи.

2. Построить алгоритм и теоретическую объектную модель решения задачи.

3. Запрограммировать полученные алгоритмы и объектную модель.

6. Форма отчета о работе

Лабораторная работа № ___

Номер учебной группы _____
Фамилия, инициалы учащегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Оснащение работы: _____
Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Что такое архитектура MVC и каковы основные компоненты?
2. Как происходит маршрутизация запросов в ASP.NET MVC?
3. Как валидация данных реализуется в ASP.NET MVC?
4. Как обрабатываются данные, отправленные через формы?
5. В чем разница между методами действий GET и POST?

8. Рекомендуемая литература

1. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. СПб. : Изд-во Питер, 2021. 896 с.

2. Прайс, М. Дж. C# 10 и .NET 6. Современная кросс-платформенная разработка / М. Дж. Прайс. СПб : Изд-во Питер, 2023. 848 с.

3. Васильев, А.Н. Программирование на C# для начинающих. Особенности языка / А.Н. Васильев. М. : Эксмо, 2022. 528 с.

4. Фримен, А. ASP.NET Core 3 с примерами на C# для профессионалов / А. Фримен. СПб. : Изд-во Вильямс, 2021. 1184 с.