Введение. Шаблон MVC

№ урока: 1 **Курс:** Создание пользовательского интерфейса в ASP.NET Core

Средства обучения: Visual Studio 2019, .NET Core SDK

Обзор, цель и назначение урока

Изучить шаблон MVC (Model View Controller). Научиться настраивать пустое ASP.NET Core приложение, добавляя компоненты, необходимые для работы MVC.

Изучив материал данного занятия, учащийся сможет:

- Добавлять MVC в пустой проект ASP.NET Core.
- Выполнять основные настройки маршрутизации.
- Создавать контроллеры и представления, использовать основные возможности методов действий.
- Создавать пользовательские представления.

Содержание урока

1. Что такое MVC
2. Как настроить ASP.NET Core приложение для использования шаблона MVC
3. Работа со статическими файлами
4. Пример простого приложения с моделью и несколькими представлениями

Резюме

- MVC (Model View Controller) – один из шаблонов для построения приложений с пользовательским интерфейсом. В данный момент распространен в технологиях, связанных с построением веб ориентированных приложений.
- Основная задача шаблона MVC - отделить пользовательский интерфейс (view) от бизнес логики (controller) и в то же время вынести отдельно бизнес сущности (model). Такой подход дает большую гибкость, давая возможность менять интерфейс, не затрагивая поведения и данных, и менять данные и поведения, не внося изменения в интерфейс.
- Model – хранит состояние приложения. Содержит данные, которые нужно отобразить.
- Controller – класс, ответственный за выполнение бизнес логики, обновление модели и выбор правильного представления для отображения данных пользователю.
- View – HTML разметка, в которой отображается состояние модели.
- Если придерживаться правил MVC, то получается тестируемый код, так как написать Unit тесты на классы контроллеры, которые не связаны с интерфейсом, гораздо проще, чем на классы, которые тесно связаны с HTML разметкой и элементами управления.
- Для использования MVC в ASP.NET Core приложении нужно добавить два фрагмента кода:
    - services.AddMVC() в классе Startup в методе ConfigureServices
    - app.UseMVC() в методе Configure
- При именовании класса контроллера нужно использовать окончание Controller, например, Home**Controller**, Account**Controller**, Catalog**Controller**.

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Класс контроллера должен наследоваться от базового класса Controller или ControllerBase.
- Контроллеры должны находиться в директории Controllers (но это не является обязательным правилом).
- Открытые методы в контроллере называются методами действия или Action Method. Такие методы можно запустить с помощью правильно отправленного HTTP запроса.
- Обычно методы действия возвращают тип данных IActionResult, который может представлять разметку, перенаправление пользователя или ответ с определенным статус кодом.
- При использовании в методе действия метода View без передачи параметров, пользователю в ответ будет возвращаться представление, которое называется так же, как и текущий метод действия. При этом представление будет находиться в папке Views, во вложенной папке, которая называется так же, как и контроллер, в котором был запущен метод действия.
- Для того, чтобы приложение могло обслуживать запросы к статическим файлам (например, к HTML странице или файлу со стилями), необходимо размесить файл в директории wwwroot и добавить в middleware pipeline app.UseStaticFiles. При этом данный middleware должен быть добавлен перед middleware MVC.

Закрепление материала

- Зачем нужен шаблон MVC? Объясните, что означает M, V, C.
- Как добавить MVC в ASP.NET Core приложение?
- Каких правил необходимо придерживаться при создании контроллера?
- Что такое Action Method?
- Зачем нужен метод View в контроллере?
- Откуда берётся метод View в контроллере?
- Что такое статические файлы и как с ними работать в ASP.NET Core приложении?

Дополнительное задание

Задание
Создайте пустое ASP.NET Core приложение. Внесите в него нужные изменения, для использования MVC. Сделайте необходимые изменения в проекте, чтобы при запросе /Test/Message отображалась страница с сообщением «Hello world», а при запросе List/Info - отображался список <ul> с тремя элементами и произвольным текстом.

Самостоятельная деятельность учащегося

Задание 1.
Доработайте приложение SimpleApp. В файл data.txt добавьте дополнительную информацию о продукте – описание продукта, количество единиц на складе.
Добавьте в представление Details описание продукта и количество единиц на складе.
В представлении List сделайте так, чтобы если продукта на складе нет, отображалось сообщение напротив продукта - «нет в наличии», если количество до 5 единиц на складе – «заканчивается», если более 5 единиц – «в наличии».

Задание 2.

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Сделайте так, чтобы в приложении SimpleApp по адресу Home/Index возвращалось представление. Сделайте в этом представлении ссылку «Скачать описание урока». При клике на ссылку должен скачиваться этот файл.

Рекомендуемые ресурсы

Общие сведенья о ASP.NET Core MVC
https://docs.microsoft.com/ru-ru/aspnet/core/mvc/overview?view=aspnetcore-2.1

Справочник по синтаксису Razor
https://docs.microsoft.com/ru-ru/aspnet/core/mvc/views/razor?view=aspnetcore-2.1

Model View Controller
https://ru.wikipedia.org/wiki/Model-View-Controller

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com