ASP.NET Core MVC

Guía para Principiantes

Sumario

Aquí tienes una guía paso a paso que puedes utilizar para aplicar lo que se enseña en el curso de ASP.NET Core MVC a tus proyectos, especialmente si estás empezando en este tipo de programación:

1. Configuración del entorno de desarrollo:

- **Descarga e instala .NET 9 SDK**. Puedes encontrar el instalador en el sitio web de Microsoft. Asegúrate de elegir la versión correcta para tu sistema operativo.
- **Instala o actualiza Visual Studio**. El curso recomienda usar la versión más reciente de Visual Studio. Si tienes una versión anterior, actualízala para asegurar la compatibilidad con .NET 9. Si estás usando una versión *preview* de .NET 9, necesitarás habilitar las *preview features* en la configuración de Visual Studio.
- Abre Visual Studio y crea un nuevo proyecto de tipo **"ASP.NET Core Web App (Model-View-Controller)"**, asegurándote de que el lenguaje seleccionado sea C#.

2. Entendiendo la Arquitectura MVC

MVC (Modelo-Vista-Controlador) es un patrón arquitectónico que divide una aplicación en tres partes principales:

- **Modelos:** Representan los datos y la lógica de negocio de la aplicación. Por ejemplo, si tienes una aplicación para gestionar libros, el modelo representaría la información de cada libro y las reglas para añadir, actualizar o eliminar libros.
- **Vistas:** Son la parte de la aplicación que muestra los datos al usuario. Las vistas se crean usando archivos Razor (.cshtml) que combinan HTML y código C#.
- **Controladores:** Gestionan la interacción del usuario, determinan qué modelos usar y qué vistas mostrar. Los controladores actúan como intermediarios entre el modelo y las vistas.

3. Estructura de un proyecto ASP.NET Core MVC

Program.cs: Es el punto de entrada de la aplicación. Aquí se configura el *pipeline* de peticiones HTTP y los servicios de la aplicación.

- Inicializa un `WebApplicationBuilder` para configurar servicios, el servidor web y otros ajustes.
- Añade servicios MVC al contenedor de inyección de dependencias para habilitar el uso de controladores y vistas.
- Configura el *pipeline* de peticiones HTTP para redirección a HTTPS, enrutamiento, autorización y manejo de excepciones.

- **Carpeta "Controllers":** Contiene las clases de controladores que gestionan las peticiones HTTP.
- **Carpeta "Models":** Contiene las clases que representan los datos de la aplicación.
- **Carpeta "Views":** Contiene las vistas (.cshtml) que se usan para generar HTML que se envía al navegador.

    - Contiene una subcarpeta "Shared" para vistas compartidas como el *layout* y scripts de validación.
- **Carpeta "wwwroot":** Contiene archivos estáticos como hojas de estilo CSS, scripts de JavaScript e imágenes.
- **appsettings.json:** Archivo de configuración principal para la aplicación.
- **launchSettings.json:** Archivo de configuración para como se lanza la aplicación durante el desarrollo.

4. Trabajando con Modelos, Vistas y Controladores

- **Creación de un modelo:** Crea una clase C# en la carpeta "Models" que represente los datos que necesitas. Por ejemplo, una clase `Item` con propiedades como `Id`, `Name` y `Price`.
    - Cada propiedad en la clase representa una columna en la tabla de la base de datos.
- **Creación de un controlador:** Crea una clase C# en la carpeta "Controllers" que herede de `ControllerBase`.
    - Un controlador contiene *acciones*, que son métodos públicos que retornan un `IActionResult`.
    - Las acciones pueden devolver una vista, una redirección, un archivo JSON, o un mensaje simple.
- **Creación de una vista:** Crea un archivo `.cshtml` en la carpeta "Views", usualmente dentro de una subcarpeta que tiene el mismo nombre que el controlador. Por ejemplo, una vista `Overview.cshtml` para la acción `Overview` de un controlador llamado `ItemsController`.
    - En las vistas, puedes usar sintaxis Razor para integrar código C# con HTML, para mostrar datos del modelo o para incluir directivas.

5. Interacción entre Modelos, Vistas y Controladores

- **Controlador a Modelo:** En una acción de un controlador, crea una instancia de un modelo, configura sus propiedades y luego pasa ese modelo a una vista.
- **Controlador a Vista:** La acción de un controlador usa la función `View()` para especificar qué vista se mostrará y le pasa los datos que necesita.
- **Vista a Modelo:** En la vista, usa `@model` para especificar el tipo de modelo que va a recibir. Luego utiliza la sintaxis Razor para mostrar los datos de ese modelo.

6. Uso de Entity Framework Core (EF Core)

- **Instala los paquetes NuGet necesarios:** `Microsoft.EntityFrameworkCore`, `Microsoft.EntityFrameworkCore.Tools` y `Microsoft.EntityFrameworkCore.SqlServer`.
- **Crea una clase de contexto:** Crea una clase que herede de `DbContext` en una carpeta llamada "Data". Esta clase es el puente entre la aplicación y la base de datos.
    - En el constructor, configura las opciones de conexión con la base de datos.
    - Añade `DbSet` para cada modelo que quieras almacenar en la base de datos.
- **Configura el contexto en `Program.cs`:** Agrega tu contexto al contenedor de servicios.

- **Usa el enfoque *Code First*:** Define el esquema de la base de datos utilizando clases C# (modelos).
    - Añade migraciones usando la consola del Administrador de paquetes de NuGet: `Add-Migration <NombreDeLaMigración>`.
    - Aplica las migraciones a la base de datos: `Update-Database`.
- **Consulta y manipula datos:** Usa el contexto en tus controladores para acceder a la base de datos y realizar operaciones CRUD (Crear, Leer, Actualizar y Eliminar).
    - Utiliza métodos asíncronos (`async`, `await`) para evitar bloqueos mientras se accede a la base de datos.

7. Enrutamiento

- Por defecto, ASP.NET Core MVC mapea las peticiones URL a los controladores y acciones según el patrón `{controller}/{action}/{id?}`.
    - `controller` es el nombre del controlador (sin el sufijo "Controller").
    - `action` es el nombre de la acción.
    - `id` es un parámetro opcional.
- Puedes modificar las rutas en `Program.cs` si necesitas un enrutamiento personalizado.

8. Parámetros de acción

- Las acciones en un controlador pueden recibir datos desde la URL, *query strings* (la parte de la URL que viene después del signo de interrogación "?"), o formularios HTML.
- Puedes recibir parámetros desde la URL usando segmentos de la ruta como `/items/edit/123` para un ID de 123.
- Puedes recibir parámetros desde un *query string* como `/items/search?name=book`.

9. Directivas Razor y Tag Helpers

- **Directivas Razor:** Son instrucciones especiales para el motor Razor, siempre empiezan con `@`.
    - `@model`: Especifica el tipo de datos que la vista espera recibir.
    - `@using`: Importa espacios de nombres.
    - `@foreach`, `@if`, etc.: Permiten usar estructuras de control de C# dentro de las vistas.
- ***Tag Helpers***: Permiten añadir comportamientos dinámicos a elementos HTML, usando atributos especiales que empiezan por `asp-`.
    - `asp-action`: Especifica a qué acción de un controlador se debe redirigir un enlace o formulario.
    - `asp-route-id`: Añade un parámetro a la ruta.
    - `asp-for`: Enlaza un elemento de un formulario con una propiedad de un modelo.

10. Tipos de Relaciones en Bases de Datos

- **Uno a uno:** Cada registro en una tabla se asocia con un único registro en otra tabla.

- **Uno a muchos:** Un registro en una tabla se asocia con múltiples registros en otra tabla, pero cada registro en la segunda tabla se asocia con un solo registro en la primera.
- **Muchos a muchos:** Múltiples registros en una tabla se asocian con múltiples registros en otra tabla. Esto requiere una tabla intermedia (tabla de unión) para manejar las relaciones.

Consejos Adicionales

- **Empieza con proyectos pequeños:** No intentes crear aplicaciones complejas desde el principio. Comienza con funcionalidades sencillas y luego ve añadiendo complejidad.
- **Usa el depurador de Visual Studio:** El depurador es una herramienta muy útil para entender el flujo de la aplicación e identificar problemas.
- **Consulta la documentación:** La documentación de Microsoft es muy útil para aprender sobre ASP.NET Core MVC y Entity Framework Core.
- **Practica constantemente:** La mejor forma de aprender es practicando y construyendo tus propios proyectos.

Este curso proporciona una base sólida para construir aplicaciones web con ASP.NET Core MVC. Aplica esta guía paso a paso, experimenta con diferentes funcionalidades y ¡no dudes en hacer preguntas si necesitas ayuda!