HERRAMIENTAS DE PROGRAMACION II

DOCENTE: LIÑAN RODRIGUEZ JULIO CESAR

SEMANAS: 01, 02 Y 03

LOGRO

- **Construye la estructura de un proyecto web usando las partes de la estructura del proyecto aprendido.**

TEMARIO

1 Tema: ASP.NET Core MVC

1.1. ASP.NET Core

1.2. ASP.NET Core MVC

1.3. Estructura del proyecto

1.4. Estructura MVC en ASP.NET Core

1.4.1. Controllers

1.4.2. Vistas

1.4.3. ViewModel

1.5. Sintaxis Razor y Scaffolding

1.6. Transferencia de datos: ViewBag, ViewData y TempData

1.7. Implementando acciones (GET/POST)

1.8. Url de enrutamiento

ASP .NET CORE

- Es un nuevo framework de código abierto y multiplataforma para la creación de aplicaciones modernas conectadas a Internet y basadas en la nube.
- A través de este marco de trabajo, es muy posible crear aplicaciones web y servicios más efectivos junto con back-ends de aplicaciones móviles e incluso aplicaciones de IoT.

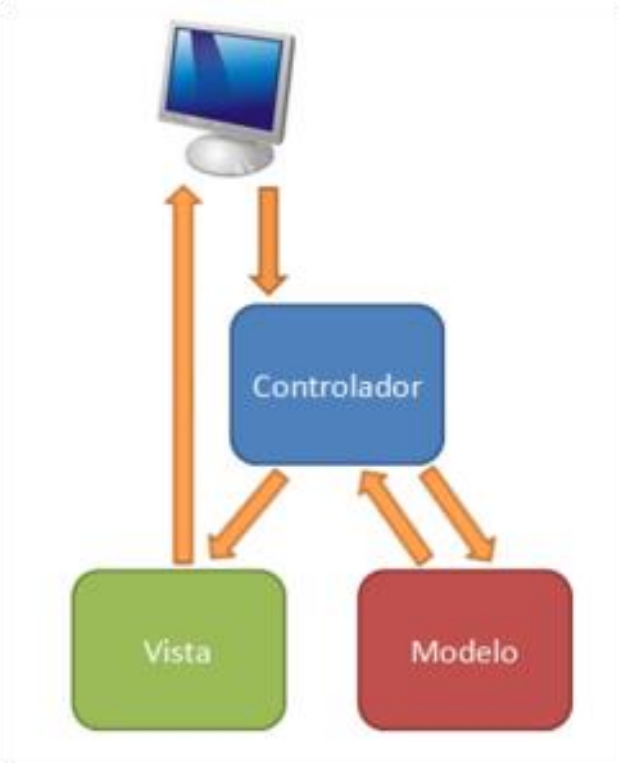
Continuación

Beneficios

- Rendimiento mejorado: Su compilador optimizará todo el código cada vez que el código se vuelva a compilar.
- Soporte de multiplataforma: Permite crear aplicaciones web que se ejecutan en Windows, Linux y Mac. El backend usará C#.
- Menor código: Pueden optimizar fácilmente su código escribiendo declaraciones mucho menores.
- Mantenimiento más sencillo: Es más fácil administrarla y mantenerla de manera efectiva.
- Asistencia para el desarrollo de aplicaciones web basadas en la nube: En caso de que tenga un negocio, es una mejor opción crear una aplicación basada en la nube en la era moderna actual.

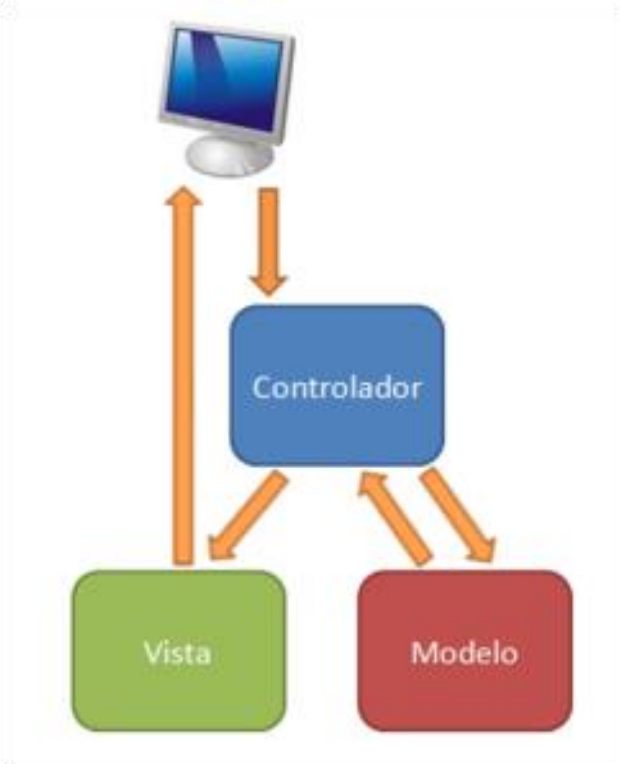
ASP .NET CORE MVC

- Es un marco liviano, de código abierto y altamente comprobable que se integra a la perfección con ASP.NET Core.
- Proporciona una forma basada en patrones para crear sitios web dinámicos. Nos da control total sobre el marcado, admite el desarrollo basado en pruebas y se adhiere a los estándares web más recientes.



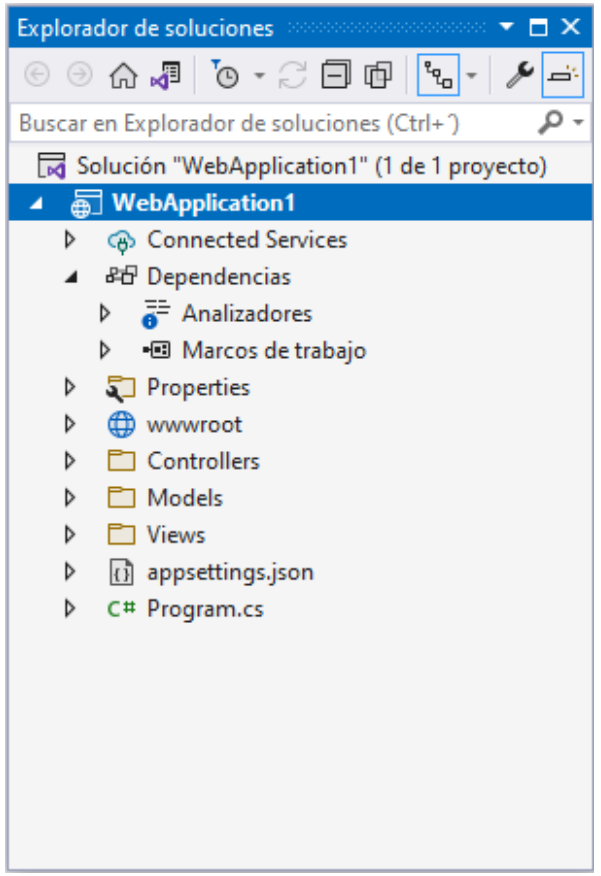
Continuación

- Con ASP.NET Core, usamos el mismo modelo de programación unificado para crear aplicaciones web MVC y Web API.
- En ambos casos, el controlador que creamos hereda de la misma clase base de controlador y devuelve **IActionResult**.
- **IActionResult** es una interfaz y tiene varias implementaciones: **ViewResult**, **JsonResult**, **etc.**

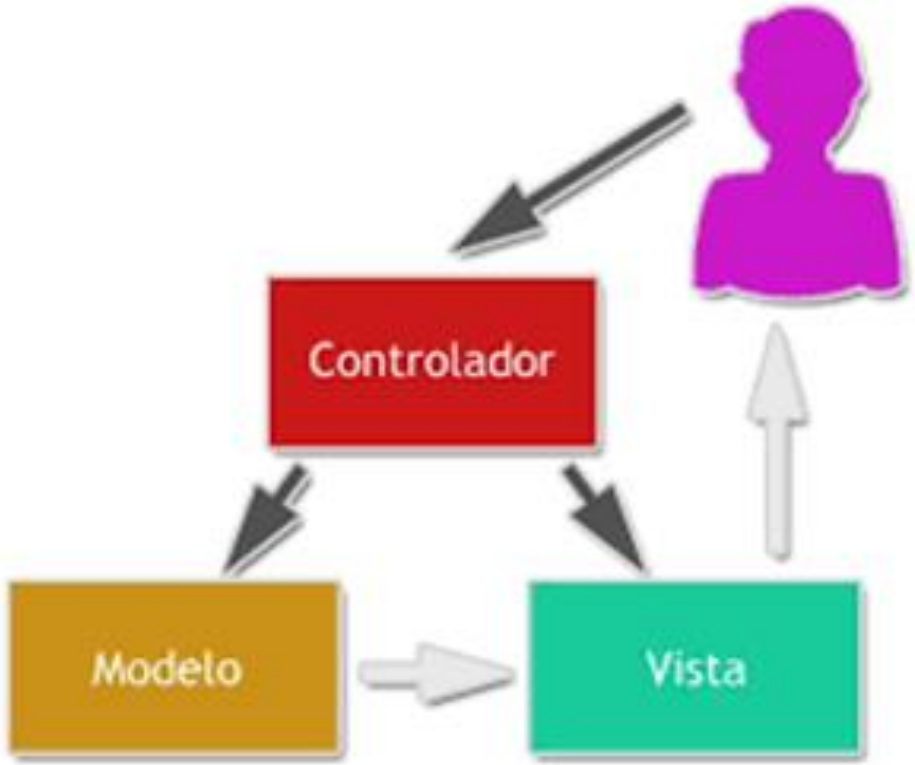


ESTRUCTURA DE UN PROYECTO

- Las dependencias se almacenan los dlls de la aplicación.
- Los archivos Program.cs y LaunchSettings.json configurarán el servidor web y la canalización de ASP.NET Core.
- Los directorios Models, Views y Controllers contienen los componentes de la arquitectura MVC.
- El directorio wwwroot contiene los archivos estáticos como CSS, Javascript e imágenes.
- El archivo appsettings.json se usa para almacenar información como cadenas de conexión o configuraciones específicas de la aplicación y estas se almacenan en el formato JSON.



Estructura MVC en ASP .NET Core



Continuación

- El usuario interactúa con la interfaz de usuario: pulsa un botón, enlace, etc. El controlador recibe la notificación de la acción solicitada por el usuario a través de un gestor de eventos (handler) o callback.
- El controlador accede manipulando al modelo, de acuerdo a la acción solicitada por el usuario.
- El modelo retorna la información solicitada por el controlador.
- El controlador envía a la vista los datos actualizados. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se refleja los cambios en el modelo.
- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

Controladores (Controllers)

- En la aplicación ASP.NET Core MVC, la clase de controlador debe heredarse de la clase base de controlador.
- Cuando el cliente envía una solicitud al servidor, esa solicitud pasa primero por el canal de procesamiento de solicitudes.
- Una vez que la solicitud pasa la canalización de procesamiento de solicitudes, llegará al controlador. Dentro del controlador, hay muchos métodos (llamados ActionResult) que realmente manejan esa solicitud HTTP entrante.
- El ActionResult dentro del controlador ejecuta la lógica comercial y prepara la respuesta que se envía al cliente que inicialmente realizó la solicitud.

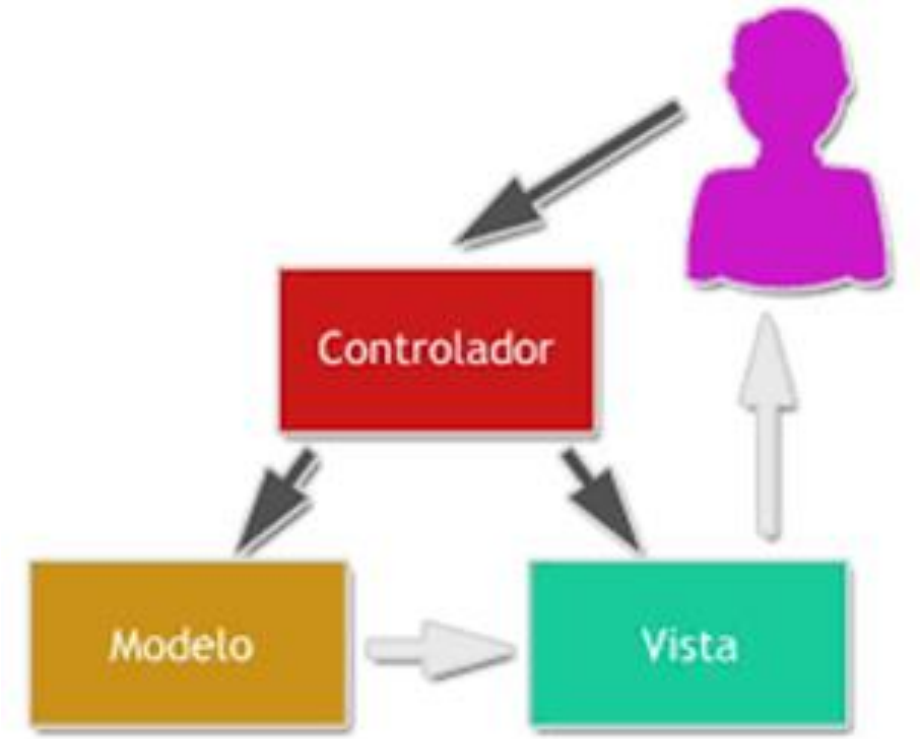
Métodos y Selectores del Action

- Una acción de controlador devuelve un resultado de acción. Un resultado de acción es lo que devuelve una acción del controlador en respuesta a una solicitud del explorador.

Acción	Resultado
ViewResult	Representa HTML y marcado
RedirectResult	Representa una redirección a una nueva dirección URL
JsonResult	Representa un resultado de notación de objetos JavaScript que se puede usar en una aplicación AJAX
ContentResult	Representa un resultado de texto
FileContentResult	Representa un archivo descargable (con el contenido binario)
FilePathResult	Representa un archivo descargable (con una ruta de acceso)

ViewModel

- Un ViewModel es un modelo que contiene más de un modelo de datos necesarios para una vista en particular.
- La combinación de varios objetos de modelo en un solo objeto de modelo de vista nos proporciona una mejor optimización.



Vistas (Views)

- La Vista es la interfaz con la que el usuario final puede interactuar. Son plantillas HTML con marcado Razor incorporado que generan contenido que se envía al cliente.
- En la aplicación ASP.NET Core MVC, una vista es un archivo con la extensión ".cshtml" (para lenguaje C#).
- Las vistas en ASP.NET Core MVC generalmente se devuelven desde el método de acción del controlador. Usamos el tipo de devolución ViewResult para devolver una vista desde un método de acción.

Sintaxis Razor

- La sintaxis Razor proporciona una sintaxis de programación para escribir código en páginas web donde el código se incrusta en formato HTML.
- El código Razor se ejecuta en el servidor antes que la página se envíe al explorador. Este código genera formato HTML y, lo envía al explorador junto con cualquier código HTML estático que contenga la página.

Scaffolding

- Scaffolding es un método para construir aplicaciones basadas en bases de datos.
- En esta técnica, el programador escribe una especificación que describe cómo debe ser usada la base de datos. Luego el compilador utiliza esa especificación para generar el código que la aplicación usará para crear, leer, actualizar y eliminar registros de la base de datos.
- Es utilizado para creación de Controladores (ActionResults) y Vistas.

Transferencia de Datos: ViewBag

- Es un contenedor construido alrededor de ViewData.
- Propiedad dinámica y utiliza las características dinámicas de C # 4.0.
- Durante la recuperación, no es necesario utilizar datos de conversión de tipos.
- Se utiliza para pasar valor de Controller a View. Está disponible solo para Solicitud actual. Se destruirá en la redirección.

Transferencia de Datos: ViewData

- Es un objeto de diccionario, es decir, definidos por clave y valore donde la clave es una cadena, mientras que el valor es un objeto.
- Durante la recuperación, los datos deben ser convertidos en tipo a su tipo original, ya que los datos se almacenan como objetos.
- Se utiliza para pasar valor de Controller a View. Está disponible solo para Solicitud actual. Se destruirá en la redirección.

Transferencia de Datos: TempData

- Es un objeto de diccionario, que almacena clave y valor, la clave es una cadena, mientras que el valor es un objeto.
- Durante la recuperación, los datos deben ser convertidos en tipo a su tipo original, ya que los datos se almacenan como objeto.
- Se puede utilizar para pasar valor de Controlador a Vista y también de Controlador a Controlador.
- Está disponible para solicitudes actuales y posteriores. No se destruirá en la redirección.

Implementando acciones (GET)

- El método Get (HttpGet) envía datos mediante una cadena de consulta.
- Los datos se adjuntan a la URL y son visibles para todos los usuarios. Sin embargo, no es seguro, pero es rápido.
- Se usa principalmente cuando no está publicando datos confidenciales en el servidor como nombre de usuario, contraseña, información de tarjeta de crédito, etc.

Implementando acciones (POST)

- El método Post (HttpPost) oculta información de la URL y no vincula datos a la URL. Es más seguro que el método HttpGet pero es más lento. Solo es útil cuando está pasando información confidencial al servidor.
- Más segura pero más lento que HttpGet.
- No tiene ninguna restricción para pasar datos y puede publicar variables de formulario ilimitadas.
- Se utiliza cuando se envían datos críticos.

Url de Enrutamiento

- Al crear una aplicación ASP.NET MVC se define una tabla de enrutamiento que se encarga de decidir que controlador gestiona cada petición Web basándose en la URL de dicha petición.
- En cada petición URL no se asigna un archivo físico del disco, sino que se asigna una acción de un controlador (más un parámetro), que nos mostrará una vista específica.

www.sitio.com/products/report/1/06/2008

```
using System;

public class productsController : Controller
{
    public ActionResult reports(int id, int month, int year)
    {
        View();
    }
}
```

Muchas Gracias!!!