Tema 3: Fundamentos de ASP.NET

Asignatura: Desarrollo web en entorno servidor

CS Desarrollo de Aplicaciones Web

Introducción

- En este capítulo veremos aspectos como:
  - Qué es el patrón MVC
  - Qué es ASP.NET Core MVC
  - Primer proyecto ASP.NET Core MVC

FORMACIÓN PROFESIONAL MONTECASTELO

INTRODUCCIÓN AL PATRÓN MVC

El patrón MVC

Patrón MVC (Model-View-Controller)

- **MVC es un patrón arquitectural, un modelo o guía que expresa cómo organizar y estructurar los componentes de un sistema software, sus responsabilidades y las relaciones existentes entre cada uno de ellos.**
- Su nombre, MVC, parte de las iniciales de Modelo-Vista-Controlador (Model-View-Controller, en inglés), que son las capas o grupos de componentes en los que organizaremos nuestras aplicaciones bajo este paradigma.

El patrón MVC

- La arquitectura MVC propone, independientemente de las tecnologías o entornos en los que se base el sistema a desarrollar, la separación de los componentes de una aplicación en tres grupos (o capas) principales:
  - **el modelo**
  - **la vista**
  - **el controlador**

El patrón MVC.

El Modelo

- **Aquellas entidades que nos servirán para almacenar información del sistema que estamos desarrollando**.
- Por ejemplo, si estamos desarrollando una aplicación de facturación, en el modelo existirán las clases Factura, Cliente o Proveedor, entre otras.

El patrón MVC.

- Por último, el Modelo será también el encargado de gestionar el almacenamiento y recuperación de las entidades del dominio, es decir, incluirá mecanismos de persistencia o será capaz de interactuar con ellos.
- **El Modelo contiene las entidades que representan el dominio, la lógica de negocio, y los mecanismos de persistencia de nuestro sistema.**

El patrón MVC.

La Vista

- **Los componentes de la Vista son los responsables de generar el interfaz de nuestra aplicación**, es decir, de componer las pantallas, páginas, o cualquier tipo de resultado utilizable por el usuario o cliente del sistema.
- Por ejemplo, en una aplicación web, si el usuario está consultando una factura, la Vista se encargará de representar visualmente el estado de la misma en forma de página visualizable en su navegador, en archivos **.aspx**, **.cshtml** de Razor, y emplearemos como lenguaje de marcado HTML o alguna de sus variantes.

El patrón MVC.

El Controlador

- **La misión principal de los componentes incluidos en el Controlador es actuar como intermediarios entre el usuario y el sistema.**
- Serán capaces de capturar las acciones de éste, como puede ser la pulsación de un botón o la selección de una opción de menú, interpretarlas y actuar en función de ellas, por ejemplo, retornando al usuario una nueva Vista que represente el estado actual del sistema, o invocando a acciones definidas en el Modelo para consultar o actualizar información.

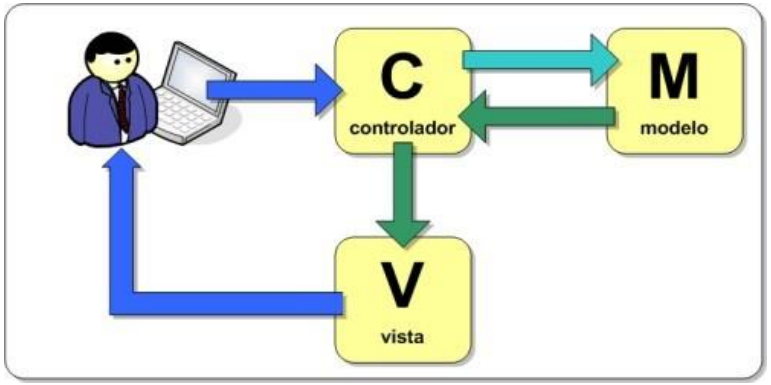
El patrón MVC.

- Realizarán también tareas de transformación de datos, traduciendo la información enviada desde el interfaz, por ejemplo valores de campos de formulario recibidos mediante el protocolo HTTP, a formatos que puedan ser comprendidos por el Modelo, como pueden ser tipos propios de .NET, o a entidades del dominio.
- **En el Controlador se encuentran los componentes capaces de procesar las interacciones del usuario, consultar o actualizar el modelo, y seleccionar las vistas apropiadas en cada momento.**

El patrón MVC.

Relación entre Modelo, Vista y Controlador

- El siguiente diagrama refleja las relaciones existentes entre los componentes del Modelo, Vista y Controlador, y de éstos a su vez con el usuario, del sistema:



El patrón MVC.

- Como se muestra en el diagrama, **las acciones e información procedentes del usuario serán recogidas exclusivamente por los Controladores.**
- **Ningún componente de otra capa debe acceder a los datos generados desde el cliente**, de la misma forma que sólo los componentes de la Vista estarán autorizados a generar interfaces de usuario con los que enviar información de retorno.

El patrón MVC.

- **El Controlador tiene acceso bidireccional al Modelo**, es decir, **será capaz tanto de actualizar su estado**, invocando por ejemplo métodos o acciones incluidos en su lógica de negocio, **como de consultar la información que sea necesaria** para completar sus tareas.

El patrón MVC.

- Por otra parte, **el Controlador es el encargado de seleccionar la Vista más apropiada en función de la acción llevada a cabo por el usuario**.

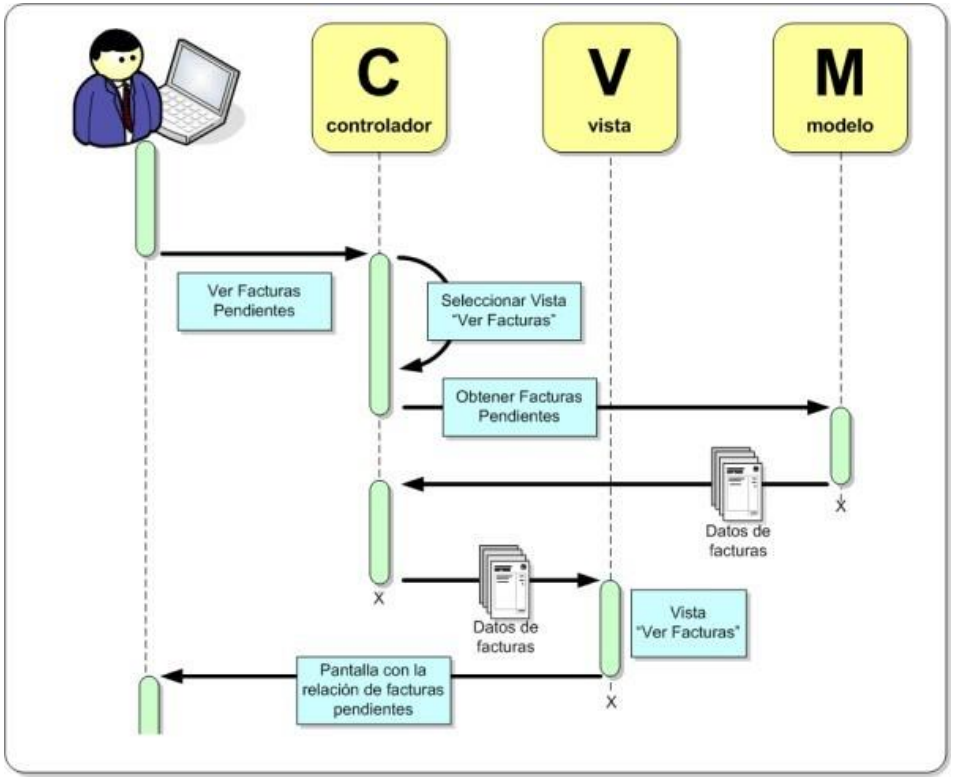
- **Una Vista, por tanto, no puede en ningún momento comunicarse directamente con el Modelo**, invocar sus acciones u obtener datos directamente desde éste, y, como en el caso anterior, será totalmente ajena a los controladores que la utilicen.

El patrón MVC.

Ejemplo de funcionamiento

- Imaginemos que un usuario está conectado a nuestra aplicación y pulsa una opción de menú destinada a visualizar facturas pendientes de pago. Una posible secuencia de acontecimientos podría ser la siguiente:
  - El Controlador recibe la orden, es decir, la pulsación del menú de la aplicación.
  - En función de la acción requerida (visualización de las facturas pendientes de pago), determina la Vista a retornar al usuario, es decir, selecciona el componente que se encargará de maquetar el interfaz de usuario.
  - Dado que el Controlador conoce que esta Vista requiere información para componer el interfaz, concretamente una relación de facturas en estado "pendiente de pago", invoca al Modelo para solicitarle dicha información.
  - El Controlador recibe los datos de las facturas del Modelo y se los envía a la Vista que previamente ha seleccionado.
  - La Vista compone (o renderiza) el interfaz utilizando los datos de que ya dispone.
  - El resultado de la maquetación es enviado al usuario.
- Una vez finalizado este proceso, el usuario estaría de nuevo en disposición de interactuar con el sistema, lo que volvería a iniciar la secuencia.

El patrón MVC.



FORMACIÓN PROFESIONAL MONTECASTELO

Vídeo: MVC desde cero

https://www.youtube.com/watch?v=UU8AKk8Slqg

FORMACIÓN PROFESIONAL MONTECASTELO

INTRODUCCIÓN A ASP.NET Core MVC

FORMACIÓN PROFESIONAL MONTECASTELO

ASP.NET Core MVC

- ASP.NET Core MVC es un framework creado por Microsoft que permite desarrollar aplicaciones para la web con arquitectura Modelo-Vista-Controlador sobre la plataforma ASP.NET.

FORMACIÓN PROFESIONAL MONTECASTELO

Ventajas de ASP.NET Core MVC

- ASP.NET Core MVC ofrece varias ventajas, que comentamos a continuación:
  - Separación de responsabilidades
  - Facilidad para realización de pruebas unitarias
  - Control total sobre el marcado
  - Control total sobre las URL
  - Uso de convención sobre configuración
  - Es flexible y extensible
  - ¡Es software libre!

ASP.NET Core MVC

- **Separación de responsabilidades**
  - La clara separación de responsabilidades impuesta por el uso del patrón MVC hace que los componentes de nuestras aplicaciones tengan sus misiones bien definidas.
- **Facilidad para realización de pruebas unitarias**
  - Como hemos visto anteriormente, los componentes del Modelo, Vista y Controlador son muy independientes entre sí y pueden ser sometidos a pruebas unitarias con relativa facilidad.
  - **ASP.NET Core MVC ha sido diseñado persiguiendo el objetivo de facilitar al máximo la realización de pruebas unitarias.**

FORMACIÓN PROFESIONAL MONTECASTELO

Vídeo: ¿Qué es una prueba unitaria?

https://www.youtube.com/watch?v=wA_y-72rLs0

ASP.NET Core MVC

- **Control total sobre el marcado**
  - En ASP.NET Core MVC, **no hay código de cliente generado de forma automática**. Somos nosotros los que nos encargamos totalmente de desarrollar el código cliente.

- **Control total sobre las URL**
  - ASP.NET Core MVC utiliza un potente sistema de routing, que nos permite definir para nuestras aplicaciones direcciones de acceso a los recursos muy amigables y muy fácilmente intuibles por los usuarios, como:
    - **http://miservidor.com/productos/fotografia** para acceder al catálogo de fotografías de un comercio
    - **http://miservidor.com/nikon-coolpix-p90** para consultar la ficha de un producto concreto
    - **http://miservidor.com/productos/editar/883** para acceder a editar el producto con código 883

ASP.NET Core MVC

- **Uso de convención sobre configuración**
  - El concepto **Convención sobre Configuración** es una filosofía de desarrollo basada en eliminar, o al menos reducir, el número de decisiones que debe tomar un desarrollador ofreciendo comportamientos que por defecto que serán válidos y útiles para la mayor parte de los casos.

  - Por ejemplo, en ASP.NET Core MVC, se asume que **los controladores serán clases cuyo nombre será siempre de la forma {Nombre}Controller**, como ProductController o UserController. De esta forma, facilita mucho al desarrollador el no tener que elegir un nombre apropiado, pues ya lo decide el framework por él.

ASP.NET Core MVC

- **Es flexible y extensible**
  - ASP.NET Core MVC framework está diseñado para ser flexible, muy flexible. Prácticamente cada componente del mismo puede ser sustituido por otro que se ajuste más a nuestras necesidades, o amplíe el alcance del original.

- **¡Es software libre!**
  - Está publicado bajo licencia Apache 2.0, lo que quiere decir que podemos utilizarlo en todo tipo de sistemas, sean abiertos o comerciales, sobre la plataforma Microsoft ASP.NET

Inconvenientes de ASP.NET Core MVC

- Algunos de los inconvenientes más destacados de ASP.NET Core MVC son:
  - Curva de aprendizaje
  - Menor nivel de abstracción
  - Hay que ceñirse a las convenciones y al patrón
  - Anulación del diseñador visual
  - El tag soup
  - No soporta controles ASP.NET
  - No se pueden convertir sitios existentes
  - La comunidad de desarrolladores no es tan amplia

ASP.NET Core MVC

- **Curva de aprendizaje**
  - ASP.NET Core MVC requiere un mayor conocimiento de las tecnologías relacionadas con la web, como HTML, CSS, Javascript, o incluso el funcionamiento del protocolo HTTP.

- **Menor nivel de abstracción**
  - En ASP.NET Core MVC trabajamos más pegados a la realidad desconectada de la web. En otras palabras, nosotros tenemos que hacer el trabajo sucio, no esperar que arrastrando un botón ya nos venga configurado.

ASP.NET Core MVC

- **Hay que ceñirse a las convenciones y al patrón**
  - Por ejemplo, **una página que pregunte el nombre del usuario con un cuadro de texto y un botón**, y que muestre una pantalla saludándolo amablemente al pulsar éste, se puede hacer en unos pocos segundos, dentro de un único formulario.
  - En ASP.NET Core MVC, **este mismo desarrollo podría implicar la creación de dos vistas** (una para solicitar los datos y otra para mostrar el mensaje) **y un controlador con un par de métodos de acción**. Vamos a tardar más tiempo en desarrollarlo, y tendremos que generar más archivos. A cambio, eso sí, tendremos una aplicación perfectamente estructurada y con una total separación de responsabilidades, con las ventajas que ello conlleva.

ASP.NET Core MVC

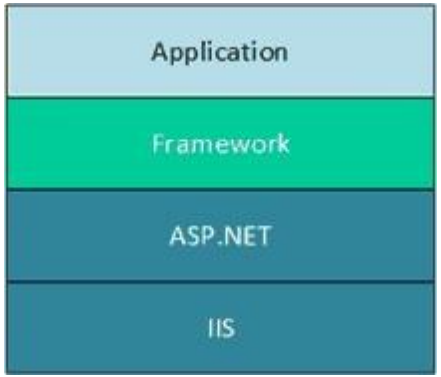
- **Anulación del diseñador visual**
  - ASP.NET Core MVC no pone fácil la maquetación usando drag&drop; la forma ideal de componer los interfaces será a nivel de código.

- **El tag soup**
  - La mezcla de etiquetas HTML, C#, CSS y Javascript puede ser complicado de entender y de mantener.
  - Por esta razón, entre otras, ASP.NET Core MVC 3 introdujo el **motor de vistas Razor**, que permite simplificar en gran medida la codificación de vistas integrando de forma muy natural el código de servidor con el lenguaje de marcado.

ASP.NET Core MVC

Arquitectura ASP.NET

- En esta imagen vemos la arquitectura de casi todas las aplicaciones ASP.NET:
  - Tenemos un **servidor**, normalmente IIS
  - Sobre el que corre **ASP.NET**…
  - Con el que está construido el framework (MVC, Webforms…)…
  - En el que desarrollamos nuestras aplicaciones. (Visual Studio 2022)

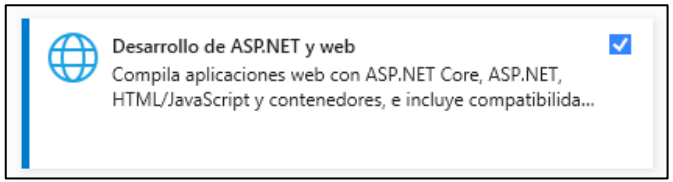


FORMACIÓN PROFESIONAL MONTECASTELO

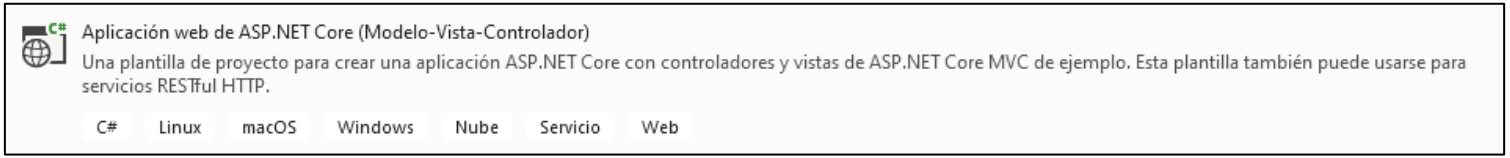
PRIMER PROYECTO ASP.NET Core MVC

Primer proyecto ASP.NET Core MVC

- Cuando vayamos a crear nuestro primer proyecto ASP.NET Core MVC, debemos:
  - **Tener instalada la opción:**



  - **Crear un proyecto:**



FORMACIÓN PROFESIONAL MONTECASTELO

Actividad 1

Crea tu primer programa web, investiga las diferentes carpetas que aparecen y busca reconocer algún patrón de arquitectura.

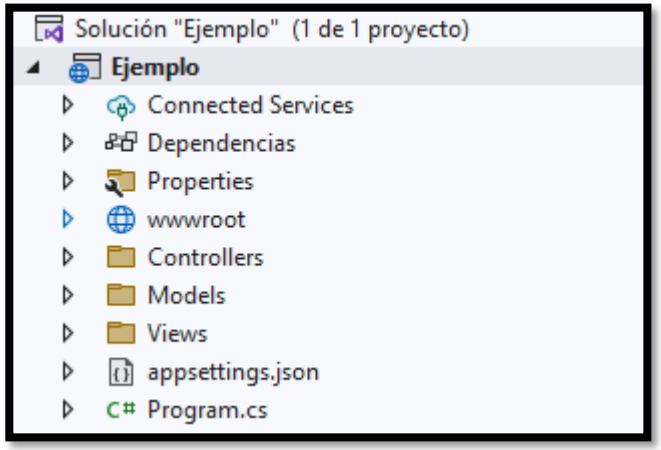
Modifica lo necesario para que la web nos pinte la imagen siguiente y despliégala.

10 MINUTOS

Primer proyecto ASP.NET Core MVC

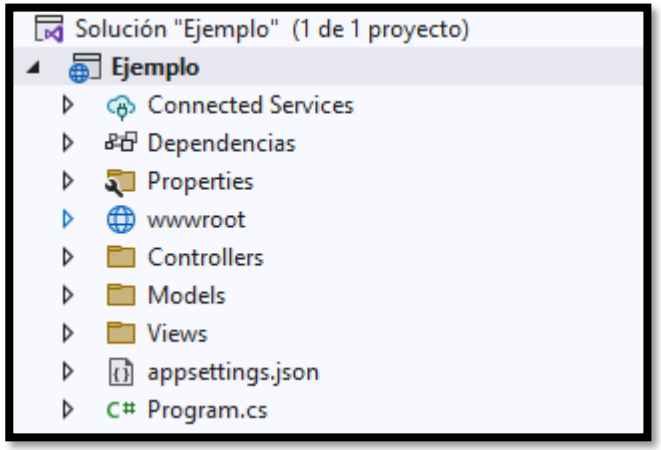
Anatomía de un proyecto ASP.NET Core MVC

- A continuación, analizaremos la estructura de un proyecto ASP.NET Core MVC.



Primer proyecto ASP.NET Core MVC

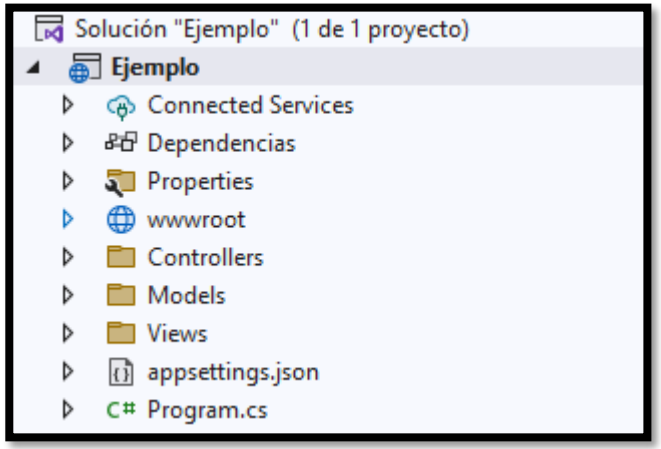
- Algunos de los elementos que nos encontraremos son:
  - ***Dependencias***, desde el que **accederemos a las referencias del proyecto** hacia ensamblados necesarios para su funcionamiento.
  - ***Properties***, desde el que **accederemos a las propiedades del proyecto**
  - Carpeta ***wwwroot***, cuyo objetivo es **almacenar los contenidos estáticos utilizados por el sitio web**. En su interior, por defecto, encontraremos las hojas de estilos utilizadas por la aplicación, pero podríamos introducir en ella todos los contenidos estáticos adicionales requeridos, imágenes, archivos HTML, PDFs, etc.



Primer proyecto ASP.NET Core MVC

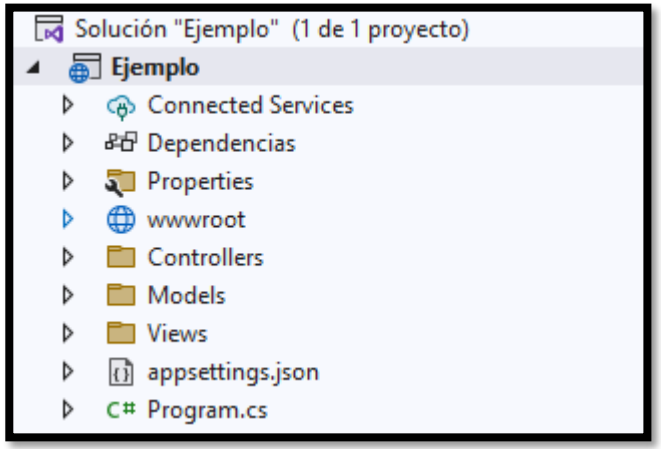
- Algunos de los elementos que nos encontraremos son:
  - Carpeta ***/Controllers***, donde **depositaremos todos los componentes pertenecientes a la capa Controlador**.

    En el proyecto por defecto aparecen un controlador:
    - **HomeController**, que procesará las peticiones relativas al sitio web (como la página inicial o la página "About“)



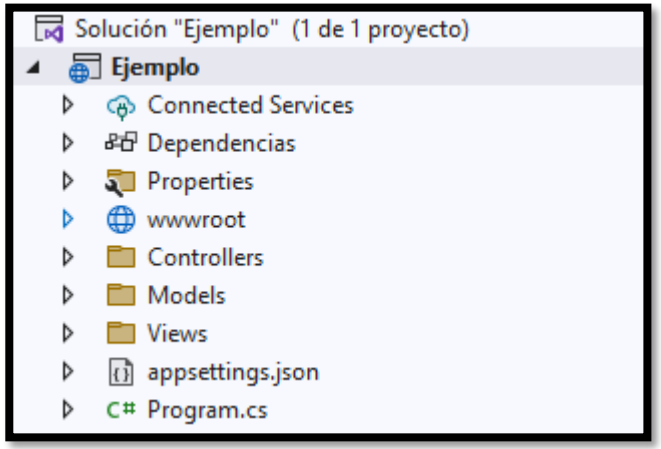
Primer proyecto ASP.NET Core MVC

- Algunos de los elementos que nos encontraremos son:
  - Carpeta ***/Models***, donde **almacenaremos las clases pertenecientes al Modelo**
  - Carpeta ***/Views***, el lugar indicado para **almacenar los componentes de la Vista**. Las vistas con las que vamos a trabajar se van a implementar **con el motor Razor**, en archivos **.cshtml** (para vistas implementadas en C#)
  - **Las vistas relacionadas exclusivamente con un controlador, serán almacenadas en una carpeta, dentro de /Views, con el nombre de dicho controlador.** Por ejemplo, una vista utilizada por un controlador llamado **Montecastelo**, debería estar en la carpeta **/Views/Montecastelo**.



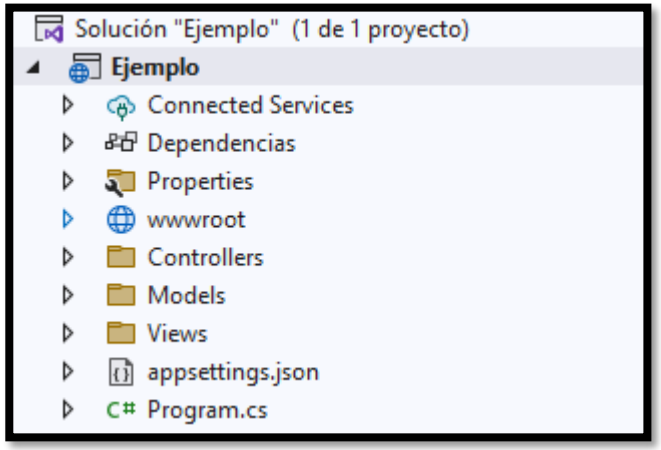
Primer proyecto ASP.NET Core MVC

- Algunos de los elementos que nos encontraremos son:
  - **Los componentes de Vista asociados a varios controladores se almacenarán en la carpeta /Views/Shared**. En el proyecto que hemos creado encontraremos los archivos:
    - **_Layout.cshtml**, que es la página maestra del sitio (llamada Layout en Razor)
    - **Error.cshtml**, la vista mostrada por defecto cuando se produce un error en nuestra aplicación.



Primer proyecto ASP.NET Core MVC

- Algunos de los elementos que nos encontraremos son:
  - Por último, encontraremos los siguientes archivos:
    - ***appsettings.json***, establecemos parámetros de configuración de la app.
    - ***Program.cs,*** se utiliza para configurar y construir el servidor web que ejecutará la aplicación.



Primer proyecto ASP.NET Core MVC

Anatomía de un proyecto ASP.NET Core MVC

- En resumen:
  - **Los Controladores**, o componentes de la capa Controlador, debemos incluirlos en la carpeta ***Controllers***.
  - **Las clases del Modelo**, serán almacenadas en la carpeta ***Models***.
  - Las vistas del interfaz de usuario, estarán en la carpeta ***Views***:
    - Las Vistas asociadas a un controlador concreto, aparecerán en una carpeta con el nombre de dicho controlador.
    - Las Vistas utilizadas desde más de un controlador, como una página maestra (layout) o una vista parcial, aparecerán en la carpeta ***Shared***.

Primer proyecto ASP.NET Core MVC

Cómo funciona una aplicación ASP.NET Core MVC

- A continuación vamos a ver el funcionamiento de una aplicación ASP.NET Core MVC realizando el seguimiento de una petición desde su origen, detallando los principales pasos que recorre durante su proceso.
- Sin embargo, comenzaremos un poco antes, describiendo lo que ocurre cuando arranca una aplicación MVC.

Primer proyecto ASP.NET Core MVC

Paso 0: Carga de la tabla de rutas

- ASP.NET Core MVC **utiliza el sistema de routing de ASP.NET para determinar cómo deben gestionarse las peticiones entrantes.**
- Por ello, todas las aplicaciones construidas sobre el framework MVC, durante su inicialización, cargan lo que se denomina la **tabla de rutas**, una **colección que contiene información sobre los patrones de URLs que serán capaces de entender, y cómo serán procesadas las peticiones que coincidan con los mismos.**

Primer proyecto ASP.NET Core MVC

Paso 0: Carga de la tabla de rutas

- Esta tabla muestra patrones y peticiones que coincidirían con cada uno de ellos.
- La URL indicada por el patrón puede estar formada por componentes fijos (como el "admin" del segundo caso) y componentes variables (las porciones encerradas entre llaves, como "controller", "action" o "id").

Patrón	Ejemplo de URLs coincidentes
{controller}/{action}/{id}	product/view/sony-vahio-vgn catalog/browse/netbooks
admin/{controller}/{action}/{id}	admin/users/edit/5 admin/threads/add admin/account/logout
{controller}/{action}/{year}/{month}	blog/archive/2009/12
{controller}/{action}/{forumName}	/forum/browse/ /forum/browse/ASP.NET

Primer proyecto ASP.NET Core MVC

Paso 0: Carga de la tabla de rutas

- Quedaros con el dato de que hay dos variables, o parámetros, que se repiten en todos ellos: **{controller}** y **{action}**.
- Para el escenario que vamos a ver en los siguientes pasos asumiremos que la tabla de rutas de nuestra aplicación tiene, exclusivamente, las siguientes entradas:
  - **admin/{controller}/{action}/{id}**
  - **{controller}/{action}/{forumName}**

Primer proyecto ASP.NET Core MVC

Paso 1: El usuario realiza una petición

- Un usuario, deseoso de obtener información sobre ASP.NET en nuestro foro, escribe en el navegador la dirección:

  http://www.ciclosmontecastelo.com/forum/browse/asp.net

- Éste abre una conexión contra el puerto 80 del servidor web que se encuentra funcionando en esta dirección y utiliza el protocolo HTTP para solicitar el recurso al que se pretende acceder.

Primer proyecto ASP.NET Core MVC

Paso 1: El usuario realiza una petición

- La orden para acceder a ese recurso sería:

```
GET /forum/browse/asp.net HTTP/1.1
Accept: */*
Accept-Language: es
(... omitted ...)
```

Primer proyecto ASP.NET Core MVC

Paso 2: La petición es recibida por el sistema de routing

- La petición realizada por el usuario es recibida directamente por el sistema de routing, cuya primera misión será determinar cuál de los patrones almacenados en la tabla de rutas encaja con la dirección del recurso (URL) especificado en la petición.
- **IMPORTANTE**: **el proceso de comprobación de rutas es ordenado y secuencial**. **El primer patrón que coincida** con la dirección, en el orden en que han sido definidos, **será tomado como válido.**

Primer proyecto ASP.NET Core MVC

Paso 2: La petición es recibida por el sistema de routing

- En nuestro escenario, donde la petición se realiza a **/forum/browse/ASP.NET**, el sistema de routing recorrería ordenadamente los patrones de la tabla de rutas que cargamos durante la inicialización:
  - el patrón ***admin/{controller}/{action}/{id}*** sería descartado, puesto que incluye un componente fijo ("admin/") no presente en la URL especificada.
  - el patrón ***{controller}/{action}/{forumName}*** encajaría perfectamente con la dirección suministrada, por lo éste sería dado por correcto y finalizaría la búsqueda.

Primer proyecto ASP.NET Core MVC

Paso 2: La petición es recibida por el sistema de routing

- A continuación, el sistema de routing comienza a analizar el patrón seleccionado, e infiere el valor de los parámetros (las porciones encerradas entre llaves) a partir de la dirección suministrada en la petición:

Patrón de ruta	**URL**
{controller}/{action}/{forumName}	`forum/browse/asp.net`
Parámetro de ruta	**Valor**
{controller}	"forum"
{action}	"browse"
{forumName}	"asp.net"

Primer proyecto ASP.NET Core MVC

Paso 2: La petición es recibida por el sistema de routing

- **El resultado de este proceso,** junto con otra información**, es introducido en el contexto de la petición**. A partir de ese momento, cede el control al framework ASP.NET Core MVC para que continúe el proceso.
- **El sistema de routing recibe la petición y delega su proceso a controladores creados específicamente para ello.**

Primer proyecto ASP.NET Core MVC

Paso 3: Se localiza la clase controlador

- **Cuando el manejador ASP.NET Core MVC recibe el control, su primera tarea es determinar qué controlador será el encargado de gestionar la petición**, para lo que acude al contexto de la petición y obtiene el contenido del parámetro obligatorio "controller", que como sabemos, en nuestro caso contiene la cadena de texto "forum".

Primer proyecto ASP.NET Core MVC

Paso 3: Se localiza la clase controlador

- Las clases controlador, quienes finalmente contienen el código que procesará la petición, deben nombrarse siempre con la denominación del controlador seguido de la palabra "Controller".
- **La forma general de nombrado de clases controlador es NombreControladorController**, por ejemplo *ForumController*, *ProductController* o *HomeController*.

Primer proyecto ASP.NET Core MVC

Paso 3: Se localiza la clase controlador

- Por tanto, teniendo en cuenta todos estos aspectos el manejador MVC haría los siguientes pasos:
  - Localizaría la clase ForumController
  - Instanciaría un objeto de este tipo
  - Buscaría cuál es el método que se debe ejecutar, según la ruta.

Primer proyecto ASP.NET Core MVC

Paso 4: Se localiza el método de acción

- El objetivo es determinar qué método en la clase controlador procesará finalmente la solicitud del usuario.
- En nuestro escenario, el valor de dicho parámetro es "browse", por lo que el framework usará de nuevo los mecanismos de reflexión para intentar localizar un método con el mismo nombre en la clase ForumController.

Primer proyecto ASP.NET Core MVC

- A continuación, esta podría ser una implementación correcta del método Browse() en la clase controlador:

```
public class ForumController : Controller
{
    public ActionResult Browse(string forumName)
    {
        var msgs = ForumData.GetMessages(forumName);
        //Obtiene mensajes desde el foro forumName

        return View("browse", msgs);
        //Devuelve la vista "browse" con los datos obtenidos
    }
}
```

Primer proyecto ASP.NET Core MVC

Paso 4: Se localiza el método de acción

- De esta forma, ASP.NET Core MVC ya sabría cuál es el procedimiento que gestionará la solicitud, y únicamente le faltaría invocarlo.
- Sin embargo, todavía tiene un problema pendiente de resolver: **el parámetro de entrada (forumName) que requiere el método para ser ejecutado.**

Primer proyecto ASP.NET Core MVC

Paso 5: Se obtienen los valores para los parámetros del método

- Para **invocar al método de acción Browse()**, ASP.NET Core MVC **necesita asignar valores a los parámetros** del mismo, para lo que, de nuevo, debe utilizar la información almacenada en el contexto de la petición.
- El mecanismo encargado de obtener esta información desde el contexto y asociarla a los parámetros de los métodos de acción es denominado "**Binder**".
- Básicamente, lo que hace es **recorrer cada uno de los parámetros del método e ir buscando variables o parámetros presentes en el contexto de la petición con un nombre coincidente**.

Primer proyecto ASP.NET Core MVC

Paso 5: Se obtienen los valores para los parámetros del método

- En nuestro ejemplo, ASP.NET Core MVC comenzaría a analizar la signatura del método Browse(), cuya implementación hemos mostrado anteriormente, y detectaría que necesita obtener un valor para el parámetro forumName.
- En primer lugar, buscaría un parámetro de ruta con dicho nombre y, dado que existe (recordad que el sistema de routing lo había extraído desde la URL usada), le inyectaría ese valor al invocar al método; si no existiera, seguiría buscando en la colección de variables incluidas en la query string, y en los valores de campos de formulario.

Primer proyecto ASP.NET Core MVC

Paso 5: Se obtienen los valores para los parámetros del método

- En resumen: **la petición original era forum/browse/asp.net**, y **el patrón de la tabla de rutas con el que coincidía era {controller}/{action}/{forumName}.**
- A raíz de ello se ha determinado que:
  - **La clase controlador es ForumController**
  - **El método de acción es Browse(string forumName)**
  - **El valor de forumName es "asp.net“**
- Una vez realizada esta tarea, el framework puede invocar el método de acción Browse() pasándole "asp.net" como valor del parámetro forumName.

Primer proyecto ASP.NET Core MVC

Paso 6: Se ejecuta el método de acción

- El siguiente paso llevado a cabo por ASP.NET Core MVC es la **invocación de nuestro método de acción**, que es donde realmente se procesará la petición.
- Volvemos a ver, por ejemplo, el código que hemos implementado previamente del método Browse():

Primer proyecto ASP.NET Core MVC

```
public class ForumController : Controller
{
    public ActionResult Browse(string forumName)
    {
        var msgs = ForumData.GetMessages(forumName);
        //Obtiene mensajes desde el foro forumName

        return View("browse", msgs);
        //Devuelve la vista "browse" con los datos obtenidos
    }
}
```

Primer proyecto ASP.NET Core MVC

Paso 6: Se ejecuta el método de acción

- El método tiene la siguiente estructura:
  - En primer lugar, **usamos el modelo para obtener información** (en este caso una lista de mensajes del foro indicado)
  - Seguidamente, **retornamos una vista que poblamos con dichos datos**.
- En particular, vamos a fijarnos en este momento en el retorno del método:
  - La última línea, **return View("browse", msgs)**, retorna la vista "browse" en forma de objeto de tipo ViewResult, y le suministra los datos que necesita ésta para maquetar el interfaz que será enviado al usuario.
- Tras ejecutarse la instrucción de retorno finaliza el código de usuario, devolviéndose el control al framework.

Primer proyecto ASP.NET Core MVC

Paso 7: Se ejecuta el resultado

- Cuando ASP.NET Core MVC recibe de nuevo el control, el resultado devuelto por el método de acción es ejecutado.
- Lo más habitual es que el proceso finalice con el envío al cliente de la vista, es decir, el código HTML.

Primer proyecto ASP.NET Core MVC

Paso 7: Se ejecuta el resultado

- En nuestro escenario, el método de acción retornó un ViewResult con la vista "browse", por lo que el motor de vistas buscará la plantilla browse.cshtml o browse.aspx
- Recordemos que su lugar será dentro de la carpeta /Views del proyecto, en una subcarpeta cuya denominación coincide con la del controlador actual ("Forum").

Primer proyecto ASP.NET Core MVC

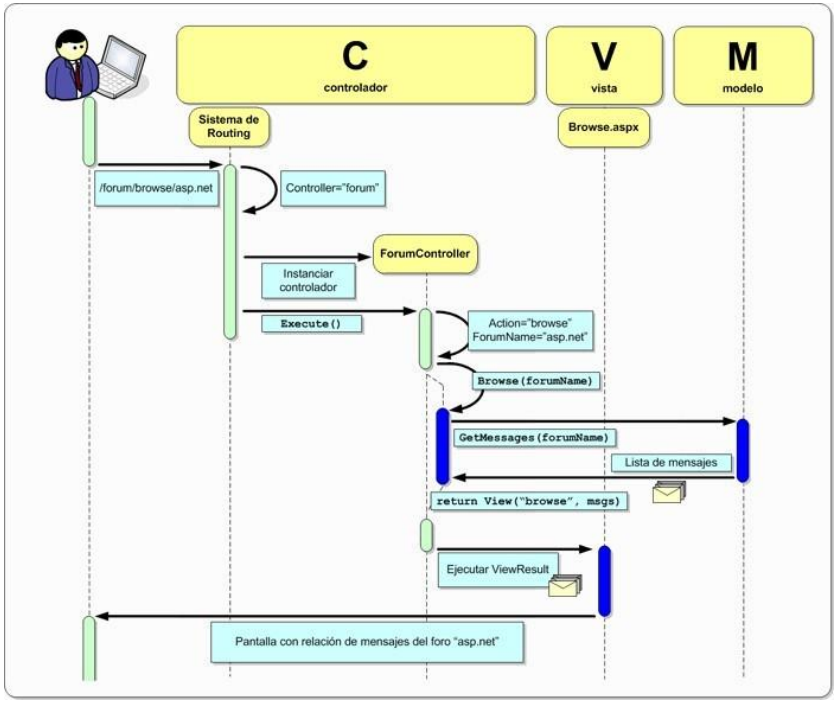
Paso 7: Se ejecuta el resultado

- Para finalizar, se procesa la plantilla (el archivo **/Views/Forum/Browse.cshtml**) enviándole todos los datos que necesita para componer el interfaz.
- En nuestro caso, esta información es una lista de mensajes, la que hemos obtenido anteriormente desde el Modelo.
- Una vez maquetado el resultado, será enviado de vuelta al cliente.

Primer proyecto ASP.NET Core MVC

Paso 8: Resumen

- En este diagrama podremos ver, de forma resumida, el ciclo de vida de la petición que hemos estado siguiendo en nuestro ejemplo.



FORMACIÓN PROFESIONAL MONTECASTELO

Vídeo: Funcionamiento de ASP.NET Core MVC

https://youtu.be/DG15OMmtjd8

FORMACIÓN PROFESIONAL MONTECASTELO

Práctica guiada

Práctica guiada 1: Aplicaciones MVC y routing

15 MINUTOS

FORMACIÓN PROFESIONAL MONTECASTELO

Práctica 1

Investiga los ejercicios del boletín 3 y, con esa información haz lo siguiente:

Crea una aplicación web que permita a los usuarios publicar y comprar productos en línea. La aplicación debe permitir a los usuarios agregar productos a su carrito de compras, realizar pagos y ver su historial de compras.

15 MINUTOS

FORMACIÓN PROFESIONAL MONTECASTELO

Práctica 2

Investiga los ejercicios del boletín 3 y, con esa información haz lo siguiente:

Crea una aplicación web que permita a los usuarios buscar y reservar citas médicas. La aplicación debe permitir a los usuarios buscar médicos por especialidad, ver horarios disponibles y programar citas.

15 MINUTOS

KEEP CALM IT'S KAHOOT TIME

Tema 3: Fundamentos de ASP.NET

Asignatura: Desarrollo web en entorno servidor

CS Desarrollo de Aplicaciones Web