

DESARROLLO DE APLICACIONES II

Ing. Ricardo Dulanto Ramírez

rdulantor@usmp.pe

Semana 02

Formula de Evaluación

$$PF = 0.5 * PE + 0.2 * EP + 0.3 * EF$$

Donde:

- **PF** = Promedio Final
- **PE** = Promedio de evaluaciones
- **EP** = Examen parcial (Promedio del Trabajo parcial individual y grupal)
- **EF** = Examen Final (Promedio del Trabajo final individual y grupal)

$$PE = (0.40 * P1 + 0.50 * P2 + 0.1 * NP)$$

Donde:

- **P1** = Práctica calificada 1 de laboratorio
- **P2** = Práctica calificada 2 de laboratorio
- **NP** = Participación en clase

Agenda

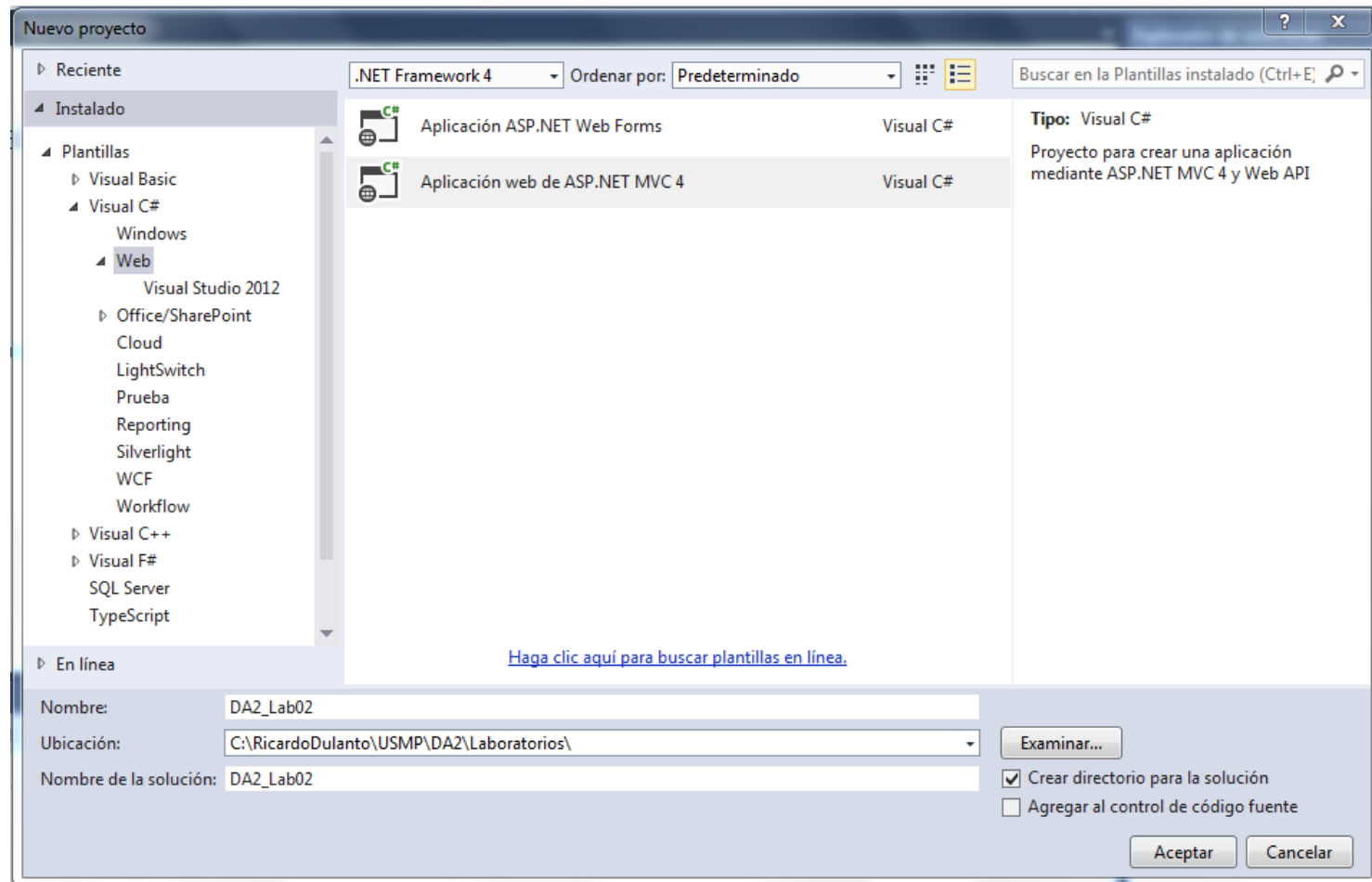


- Proyectos
- Controllers
- Routing engine

Visual Studio

PROYECTOS

Proyectos










Proyectos

Nuevo proyecto de ASP.NET MVC 4

Plantilla de proyecto

Seleccionar una plantilla:

 Vacío	 Básico	 Aplicación de Internet
 Aplicación de Intranet	 Aplicación móvil	 Web API
 Aplicación de		

Descripción:

Proyecto de ASP.NET MVC 4 vacío.

Motor de vistas:

Razor

☐ Crear proyecto de prueba unitaria

Nombre del proyecto de prueba:

MvcApplication2.Tests

Marco de pruebas:

Visual Studio Unit Test

Información adicional

Aceptar Cancelar

MVC ASP.NET

CONTROLLERS

Controladores

- El procesamiento de una petición empieza con el motor de enrutamiento.
- Este motor enlaza la petición a un método de un controlador.
- El método es ejecutado y produce el resultado que es devuelto hacia el usuario.

Conceptos y configuración

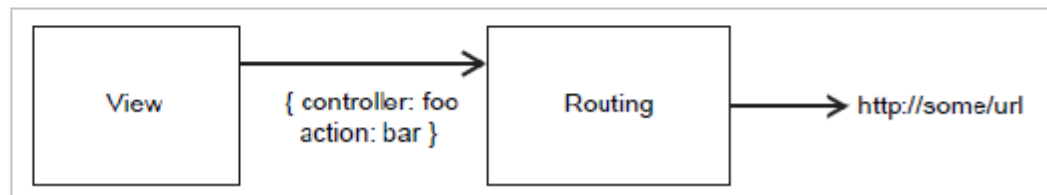
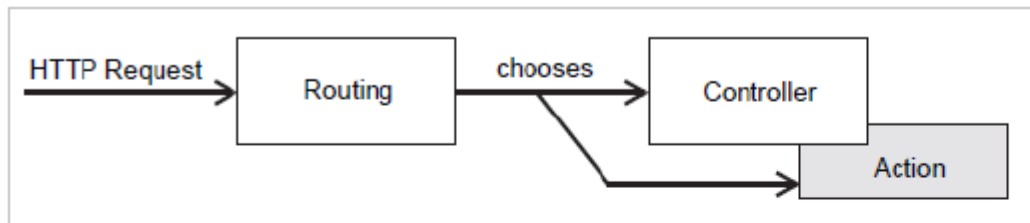
ROUTING ENGINE

Routing Engine

- En ASP.NET MVC no se espera los nombres de los archivos físicos se encuentren como parte de las URLs.
- La URL se enlaza a un método de un controlador en base al motor de enrutamiento:
 - <http://localhost/productos>
 - Controller: Productos y Método: Index
 - <http://localhost/servicios/detalle>
 - Controller: Servicios y Método: Detalle

Routing Engine

- El motor de enrutamiento de ASP.NET MVC nos permite que las URL se asignen métodos de controladores.
- De esta manera, podemos tener URL descriptivas a las acciones de los usuarios.



Routing Engine

- La plantilla de proyecto utilizada incluye dos rutas por defecto.

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}")

routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new With { .controller = "Home", .action = "Index", .id = UrlParameter.Optional }
)
```

- La primera evita el acceso a ScriptResource.axdb y WebResource.axd.
- La segunda permite enlazar las peticiones a los métodos de los controladores.

Routing Engine

- La definición de rutas se realiza en MapRoute:

Valores por defecto

Patrón de la ruta

Nombre de la ruta

```
routes.MapRoute(  
    name: "Default",  
    url: "{controller}/{action}/{id}",  
    defaults: new With { .controller = "Home", .action = "Index", .id = UrlParameter.Optional }  
)
```

Routing Engine

El motor de enrutamiento maneja las URL en dos direcciones:

1.Inbound Routing(entrada)

Recibe la URL entrante para asignarla al método y controlador correcto, así pasar los parámetros que se envían en la URL.

2.Outbound routing(salida)

Construir las URL de la aplicación en la vistas que se enlazan a algún método de un controlador, enviando los parámetros de ser necesario.

DA2_LAB02_1

LABORATORIO 01

Laboratorio 01

1. Crear el controlador Productos.
2. Para el controlador, crear los métodos y sus vistas respectivas para:
Index, List, Detail
3. Definir una ruta que permita que la URL con el formato:
<http://localhost/Catalogo/Listado>
Invoque al método **List** del controlador **Productos**
4. Definir una ruta que permita que la URL con el formato:
<http://localhost/Catalogo/Detalle>
Invoque al método **Detail** del controlador **Productos**
5. Ejecutar la aplicación

DA2_LAB02_2

LABORATORIO 02

Laboratorio 02

1. En la vista Index, crear enlaces hacia los métodos List y Detail, haciendo uso de:

- Etiqueta HTML <a>
- HelperActionLink

2. Ejecutar la aplicación y probar los enlaces.

DA2_LAB02_3

LABORATORIO 03

Laboratorio 03

Modificar las rutas creadas para que permitan que la URL con el formato:

- <http://localhost/CatalogoVirtual/Listado>
- Invoque al método **List** del controlador **Productos**

Definir una ruta que permita que la URL con el formato:

- <http://localhost/CatalogoVirtual/Detalle>
- Invoque al método **Detail** del controlador **Productos**

DA2_LAB02_4

LABORATORIO 04

Laboratorio 04

1. Crear el controlador Noticias.
2. Para el controlador, crear los métodos y sus vistas respectivas para:
Index, Listado, Detalle
3. Definir una ruta que permita que la URL con el formato:
 - <http://localhost/Archivo/Historial>
 - Invoque al método **Listado** del controlador **Noticias**
4. Definir una ruta que permita que la URL con el formato:
 - <http://localhost/Archivo/Detalle>
 - Invoque al método **Detalle** del controlador **Noticias**
5. Ejecutar la aplicación y verificar el funcionamiento de las rutas.

ROUTING

PARAMETROS EN LA URL

Parámetros

- Podemos enviar parámetros como parte de la URL, éstos pueden formar parte de la configuración de rutas.
- Se agregan como pares clave/valor al objeto de petición RouteData, para de ser necesario, acceder a los valores de la URL a través de este objeto.

```
routes.MapRoute(  
    name:="BorrarProducto",  
    url:="Producto/Borrar/{id}",  
    defaults:=New With {.controller = "Productos", .action = "Delete", .id = UrlParameter.Optional}  
)
```


Parámetros

Es posible configurar la ruta para que en el formato de la URL, los parámetros puedan tener:

- Valores por defecto.
- Indicar los valores como opcionales.

```
routes.MapRoute(  
    name:="BorrarProducto",  
    url:="Producto/Borrar/{id}",  
    defaults:=New With {.controller = "Productos", .action = "Delete", .id = 3}  
)
```

DA2_LAB02_5

LABORATORIO 05

Laboratorio 05

1. Crear el método **Delete** para el controlador **Productos**.
2. Definir una ruta para que el método pueda recibir un parámetro y la URL tenga el formato:

`http://localhost/Catalogo/Borrar/9`
3. Configurar la ruta para que el valor a recibir y se muestre en la vista correspondiente, pueda ser considerado como:
 - Parámetro opcional
 - Parámetro con valor por defecto
 - Recibir el valor en el método

MVC

RESTRICCIONES Y REGLAS

Restricciones

Podemos agregar condiciones que tienen que ser cumplidas por una petición, como por ejemplo:

- Que los valores de los parámetros de la URL tienen que cumplir alguna regla.
- Las rutas que tienen ser llamadas por GET o POST.
- Según el tipo de datos del valor del parámetro que recibe, la ruta se pueda asignar a diferentes métodos de un controlador.

Restricciones Personalizadas

- En el caso de necesitar aplicar alguna regla específica, es posible crear una clase que contenga la restricción que luego podemos utilizar dentro de la configuración de rutas.
- Se debe crear una clase que implemente **IRouteConstraint**.

Recursos

Routing

<https://msdn.microsoft.com/en-us/library/cc668201.aspx>

Regular expressions

<https://msdn.microsoft.com/en-us/library/hs600312%28v=vs.140%29.aspx>