

Ing José Muñoz

Asp .Net MVC

MVC

Ing José Muñoz

- MVC fue introducido por [Trygve Reenskaug](#) ([web personal](#)) en [Smalltalk-76](#) durante su visita a [Xerox Parc](#)⁶⁷ en los años 70, seguidamente, en los años 80, Jim Althoff y otros implementaron una versión de MVC para la biblioteca de clases de Smalltalk-80.⁸ Solo más tarde, en 1988, MVC se expresó como un concepto general en un artículo⁹ sobre Smalltalk-80.
- En esta primera definición de MVC el controlador se definía como «el módulo que se ocupa de la entrada» (de forma similar a como la vista «se ocupa de la salida»). Esta definición no tiene cabida en las aplicaciones modernas en las que esta funcionalidad es asumida por una combinación de la 'vista' y algún [framework](#) moderno para desarrollo. El 'controlador', en las aplicaciones modernas de la década de 2000, es un módulo o una sección intermedia de código, que hace de intermediario de la comunicación entre el 'modelo' y la 'vista', y unifica la validación (utilizando llamadas directas o el «[observer](#)» para desacoplar el 'modelo' de la 'vista' en el 'modelo' activo¹⁰).

MVC

- El MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Modelos y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación.

Versiones de MVC

Date	Version
Diciembre 2007	ASP.NET MVC CT
Marzo 2009	ASP.NET MVC 1.0
Marzo 2010	ASP.NET MVC 2
Enero 2011	ASP.NET MVC 3
Agosto 2012	ASP.NET MVC 4
Octubre 2013	ASP.NET MVC 5
Enero 2014	ASP.NET MVC 5.1
Febrero 2014	ASP.NET MVC 5.1.1
Abril 2014	ASP.NET MVC 5.1.2
Junio 2014	ASP.NET MVC 5.1.3
Julio 2014	ASP.NET MVC 5.2.0
Agosto 2014	ASP.NET MVC 5.2.2
Febrero 2015	ASP.NET MVC 5.2.3
Febrero 2018	ASP.NET MVC 5.2.4
Mayo 2018	ASP.NET MVC 5.2.5
Mayo 2018	ASP.NET MVC 5.2.6
Noviembre 2018	ASP.NET MVC 5.2.7

ASP.NET MVC3

Ing José Muñoz

- Templates de nuevos proyectos con soporte para HTML 5 y CSS 3.
- Validación mejorada de modelo.
- Razor View Engine
- Soporte para varios motores de Vistas i.e. Web Forms view engine, Razor o open source..
- Controller con características como ViewBag y ActionResult
- Discreto acercamiento JavaScript
- Acercamiento a inyección de dependencias con IDependencyResolver.
- Partial page output caching.

ASP.NET MVC 4

Ing José Muñoz

- **ASP.NET Web API.**
- **Renderizado adaptativo.**
- Un verdadero Template de Proyecto vacío.
- Nuevo **Mobile Project Template.**
- Soporte para agregar controladores y otras carpetas.
- Tareas soportadas por **Controlers Asincronos.**
- Control de [Bundling and Minification](#) a través de web.config.
- Soporte para accesos **OAuth y OpenID** utilizando la librería *DotNetOpenAuth*.
- Soporte para **Windows Azure SDK 1.6** y nuevos lanzamientos.

ASP.NET MVC 5

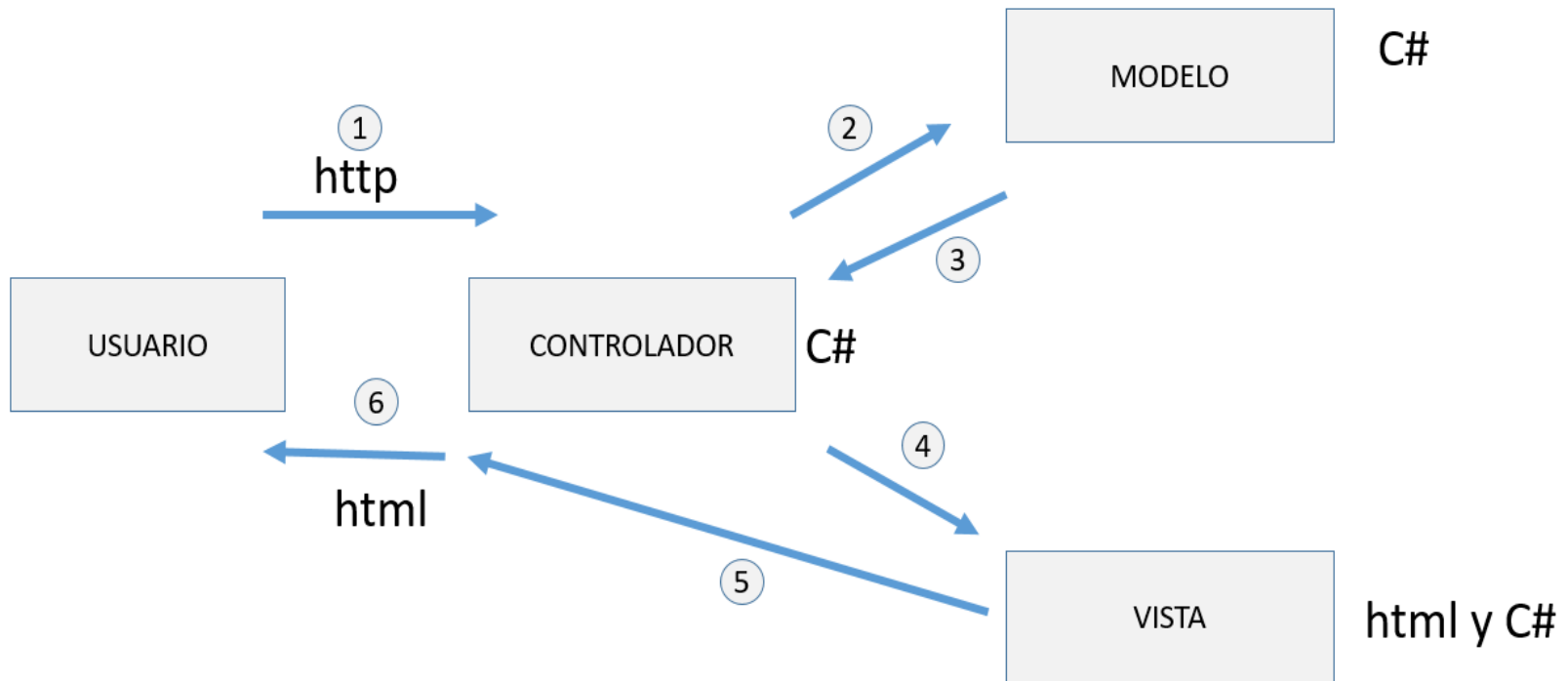
Ing José Muñoz

- **ASP.NET Identity** .
- **Authentication Filters** .
- Con ayuda de sobreescritura de **Filtros**.
- **Bootstrap** reemplaza el MVC por default.
- **Attribute Routing** integrado en MVC5..

Flujo MVC

Ing José Muñoz

FLUJO DE MVC



El Controlador

- Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos).

El Controlador

- Dentro de un proyecto la carpeta Controller contiene los controladores de la aplicación, cada controlador contiene métodos que representan paginas dentro de la aplicación similar a un link.

El Controlador

- Cada controlador contiene una carpeta con su nombre dentro de la carpeta Views, y a su vez cada acción del controlador que devuelve una vista contiene una vista en la carpeta de vistas del controlador.

El Controlador

- Una acción es un método de un controlador que le retorna contenido al usuario archivo, texto, vista.

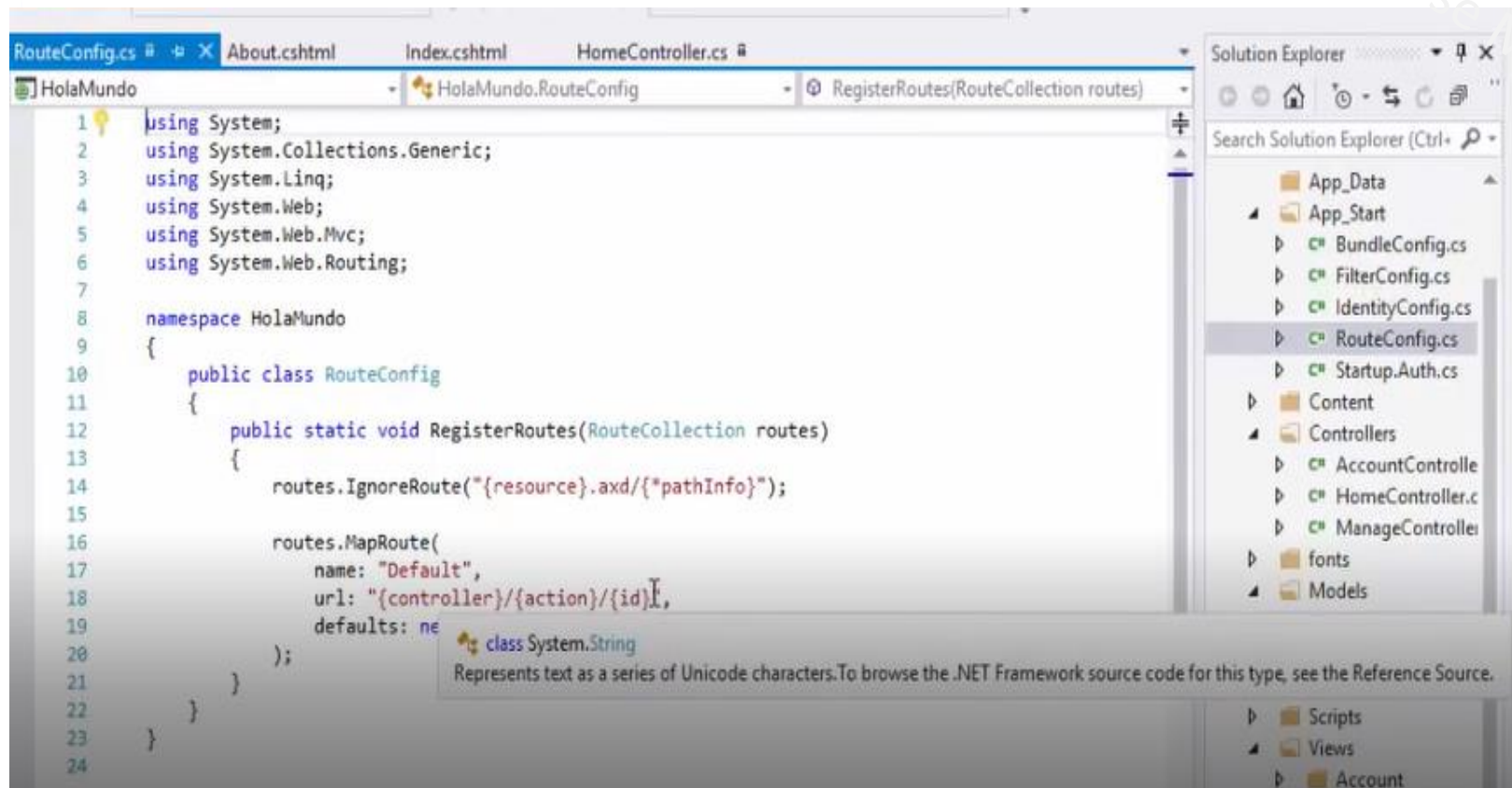
Ing José Muñoz

La clase RouteConfig

- Encapsula la funcionalidad relacionada con la redirección de peticiones, y se encontrará en App_Start/RouteConfig.cs.

Ing José Muñoz

La clase RouteConfig



The screenshot shows the Visual Studio IDE with the 'RouteConfig.cs' file open. The file is part of a project named 'HolaMundo'. The code defines a 'RouteConfig' class with a static method 'RegisterRoutes' that configures the application's routes. The 'RegisterRoutes' method uses 'routes.IgnoreRoute' to ignore routes for '.axd' files and 'routes.MapRoute' to map a default route. The 'MapRoute' method is configured with the name 'Default', a URL pattern '{controller}/{action}/{id}', and no defaults. The 'Solution Explorer' on the right shows the project structure, including folders like 'App_Data', 'App_Start', 'Content', 'Controllers', 'fonts', and 'Models'. The 'RouteConfig.cs' file is highlighted in the 'App_Start' folder. A tooltip for 'class System.String' is visible over the 'defaults: ne' line in the 'MapRoute' method.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Web;
5 using System.Web.Mvc;
6 using System.Web.Routing;
7
8 namespace HolaMundo
9 {
10     public class RouteConfig
11     {
12         public static void RegisterRoutes(RouteCollection routes)
13         {
14             routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
15
16             routes.MapRoute(
17                 name: "Default",
18                 url: "{controller}/{action}/{id}",
19                 defaults: ne
20             );
21         }
22     }
23 }
```

class System.String
Represents text as a series of Unicode characters. To browse the .NET Framework source code for this type, see the Reference Source.

Tipos de Action Result

- ActionResult
- PartialViewResult
- ContentResult
- EmptyResult
- FileResult

Ing José Muñoz

Tipos de Action Result

- UnauthorizedResult
- JavaScriptResult
- JsonResult
- RedirectResult
- RedirectToRouteResult

Ing José Muñoz

Ejemplos de ActionResult

Ing José Muñoz

```
namespace EjercicioMVC01.Controllers
{
    public class HomeController : Controller
    {
        public HttpStatusCodeResult Index()
        {
            return new HttpStatusCodeResult(404);
        }
    }
}
```

```
namespace EjercicioMVC01.Controllers
{
    public class HomeController : Controller
    {
        public ContentResult Index()
        {
            return Content("Jose Muñoz");
        }
    }
}
```

Ejemplos de ActionResult

Ing José Muñoz

```
namespace EjercicioMVC01.Controllers
{
    public class Persona
    {
        public string Nombre { get; set; }
        public int Edad { get; set; }
    }

    public class HomeController : Controller
    {
        public JsonResult Index()
        {
            var persona1 = new Persona() { Nombre="Jose",Edad=10};
            var persona2 = new Persona() { Nombre="Juan",Edad=20};
            List < Persona > Lista = new List<Persona>();
            Lista.Add(persona1);
            Lista.Add(persona2);
            return Json(Lista, JsonRequestBehavior.AllowGet);
        }
    }
}
```

Ejemplos de ActionResult

Ing José Muñoz

```
public RedirectResult Index()  
{  
    return Redirect("https://www.google.com.mx");  
}
```

```
public RedirectToRouteResult Index()  
{  
    return RedirectToAction("About", "Home");  
}
```

```
public HttpStatusCodeResult Index()  
{  
    return new HttpStatusCodeResult(301);  
}
```

Modelo

- Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio).

Vista

- Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto requiere de dicho 'modelo' la información que debe representar como salida.

Ejemplo de Vista

Ing José Muñoz

```
@{  
    ViewBag.Title = "About";
```

```
}
```

```
<h2>@ViewBag.Title.</h2>
```

```
<h3>@ViewBag.Message</h3>
```

```
<p>Use this area to provide additional information.</p>
```

Razor

- Es una sintaxis basada en C# (aunque se puede programar en Visual Basic) que permite usarse como motor de programación en las vistas o plantillas de nuestros controladores.

Reglas de Razor

- Los bloques de código de razor van entre @{ ... }
- Las expresiones inline (variables y funciones) empiezan con @
- Las sentencias de código terminan con punto y coma
- Las variables son declaradas con var

Reglas de Razor

- Las cadenas de caracteres van entre comillas dobles
- El código de C# es sensible a mayúsculas y minúsculas.
- Los archivos de C# terminan con cshtml

Ejemplo de razor

```
@{  
    var total = 0;  
    var totalMessage = "";  
    if (IsPost)  
    {  
        var num1 = Request["text1"];  
        var num2 = Request["text2"];  
        total = num1.AsInt() + num2.AsInt();  
        totalMessage = "Total = " + total;  
    }  
}
```

VIVE LA FIME

Ejemplo de razor

```
<html>
<head>
  <meta name="viewport" content="width=device-width" />
  <title>Index</title>
</head>
<body>
  <p>Ingresa dos numeros y despues da click <strong>Agregar</strong>.</p>
  <form action="" method="post">
    <p>
      <label for="text1">Primer Numero:</label>
      <input type="text" name="text1" />
    </p>
    <p>
      <label for="text2">Segundo Numero:</label>
      <input type="text" name="text2" />
    </p>
    <p><input type="submit" value="Agregar" /></p>
  </form>
  <p>@totalMessage</p>
</body>
</html>
```

Dictionary<TKey,TValue> Class

- Representa una colección de claves y valores.
- La [Dictionary<TKey,TValue>](#) clase genérica proporciona una asignación de un conjunto de claves a un conjunto de valores. Cada adición al diccionario consta de un valor y de su clave asociada. Recuperar un valor usando su clave es muy rápido, cerca de $O(1)$, porque la [Dictionary<TKey,TValue>](#) clase se implementa como una tabla hash.
- **Parámetros de tipo**
- **Tkey** Tipo de las claves del diccionario.
- **Tvalue** Tipo de los valores del diccionario.

Ejemplo de Diccionario

```
// Creación de diccionario de cadenas, con llaves de cadenas.
Dictionary<string, string> openWith = new Dictionary<string, string>();
// Se agregan nuevos elementos al diccionario. Ahí no hay llaves
// duplicadas, pero algunos de las valores están duplicados
openWith.Add("txt", "notepad.exe");
openWith.Add("bmp", "paint.exe");
openWith.Add("dib", "paint.exe");
openWith.Add("rtf", "wordpad.exe");
// El metodo de agregado lanza una excepcion si la nueva llave ya
// esta en el diccionario
try
{
    openWith.Add("txt", "winword.exe");
}
catch (ArgumentException)
{
    Console.WriteLine("Un elemento con la llave = \"txt\" ya existe");
}
```

Creación de Proyecto MVC

Ing José Muñoz

Visual Studio 2019

Open recent

Seleccionar: Crear
un nuevo Proyecto



Get started



Clone or check out code

Get code from an online repository like GitHub or Azure DevOps



Open a project or solution

Open a local Visual Studio project or .sln file



Open a local folder

Navigate and edit code within any folder



Create a new project

Choose a project template with code scaffolding to get started

[Continue without code →](#)

Selección de Lenguaje, Sistema Operativo y Plataforma

Ing José Muñoz

Indicar el lenguaje de código,
Sistema Operativo y Plataforma de

Create a new project

Recent project templates

- ASP.NET Web Application (.NET Framework)
- Console App (.NET Framework)
- Workflow Console Application
- ASP.NET Core Web Application
- Console App (.NET Core)
- Class Library (.NET Framework)

Search for templates (Alt+S) Clear all

C# Windows Web

Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).

C# Linux macOS Windows Cloud Web

gRPC gRPC Service
A project template for creating a gRPC ASP.NET Core service using .NET Core.

C# Linux macOS Windows Cloud Service Web

Razor Class Library
A project template for creating a Razor class library.

C# Linux macOS Windows Library Web

JUnit Test Project (.NET Core)
A project that contains NUnit tests that can run on .NET Core on Windows, Linux and MacOS.

C# Linux macOS Windows Desktop Test Web

ASP.NET Web Application (.NET Framework)
Project templates for creating ASP.NET applications. You can create ASP.NET Web Forms, MVC, or Web API applications and add many other features in ASP.NET.

C# Windows Cloud Web

Web Driver Test for Edge (.NET Core)
A project that contains unit tests that can automate UI testing of web sites within Edge browser (using Microsoft WebDriver).

De las opciones que aparecen
seleccionar Asp .Net Web Application

Back Next

Configuración de Proyecto

Ing José Muñoz

Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name
Ejercicio01_MVC

Location
C:\Users\user\source\repos

Solution name ⓘ
Ejercicio01_MVC

☐ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Configurar los valores de:
Nombre de Proyecto
Ubicación
Nombre de Solución
Versión de Framework con que se va a realizar

Back

Create

Selección de Aplicación Web MVC

Create a new ASP.NET Web Application



Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.



Web Forms

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.



MVC

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.



Web API

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.



Single Page Application

A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication

No Authentication

[Change](#)

Add folders & core references

☐ Web Forms

☒ MVC

☐ Web API

Advanced

☒ Configure for HTTPS

☐ Docker support

(Requires [Docker Desktop](#))

☐ Also create a project for unit tests

Ejercicio01_MVC.Tests

Back

Create

Entorno de desarrollo

