# Curso MVC 5

## Conceptos basicos

HomeController

Es una clase que hereda de la clase base controllery tiene distintos metodos (Index, About,Contact) los metodos retornan un

ActionResult

Como un link dentro da aplicación. Encapsula el resultado de un método de acción y se usa para realizar una operación de nivel de marco en nombre del método de acción.

AccionResult e generico e con el podemos retornar vistas,archivos, json

Pero podemos ser mais especifico e indicarlle que retorne un archivo por ejemplo con FileResult, para retornar vista ponemos ViewResult

Vistas:

Cada controlador ten as sua carpeta de ***vistas***:

-Ejemplo: As vistas de Homecontroler estan en view ../view/home

Routeconfig:

Donde configuramos las rutas.

url: "{controller}/{action}/{id}":como se compon a url.Os nombres que son variables van ente{}

decimos que temos que indicar o controlador e a accion.

Razor

Razor é unha sintaxe de programación ASP.NET usada para crear páxinas web dinámicas, permite escribir código C# mezclado con html nas vistas para eso solo temos que poñer un @ diante. para usar as variables en html poñemos @nombre da variable.

AcountController y el ManageControler

Engloban la logica de la adminstracion de usuarios.

#### ContentResult

Devolver desde un action un string libre

return Content("asdad","aplication/json"); poñemos o texto logo o tipo(se non poñemos o tipo colleo por defecto)

Podemos devolver contenido html \*/

#### Query String

Sirve para pasar parámetros dende o navegador a un action.

Ejemplo: localhost:port?nombre=1&apellido=1

O signo ? indica que se va a utilizar un query string,a continuacion as variables separadas por &. Si el action no usa los string simplemente los ignora.

Los valores tipo nn poden ter nulos (ejemplo int).

#### ViewBag e ViewData

**ViewBag:**objeto que permite enviar información do action hacia a vista mediante variables ejemplo ViewBag.Message="dadssa"

**ViewData**:igual que el ViewBag pero con distinta sintaxis

El valor de la variable solo se mantiene en la vista, el tiempo de vida es muy corto.

#### Layout y shared

Layout es lo mismo que la master page.

En la carpeta \Views\Shared colocaremos os elementos das vistas que teñan que co visual que necesitemos compartir entre todas as vistas.

@Html.Partial("\_LoginPartial") Sirve para cargar unha vista parcial

@RenderBody()donde se carga a vista que generamos nese momento

#### FileResult

Descargar Archivos

#### Metodo Raw

si poñemos lenguaje html nn funciona na vista porque MVC nos protege de mostrar codigo html que ven de fora das vistas.Para proteger de posibles ataques.

Poderiase usar html nas vistas se escribimos @Html.Raw(pelicula.titulo) pero nn é recomentable. Moi perigoso, porque poden ejecutar javascript.

#### ActionLink

Anchor: código de enlace de como vincular a unha página en HTML. <li>@Html.ActionLink("Inicio", "Index", "Home")</li>

#### RenderAction

RenderAction sirve para cargar action dentro de una vista que no es de esa acción.

#### Display

sirve para pasar calquera tipo do controlador a vista

Ol poder de display está en que se poden pasar objetos o clases

##### DisplayTemplates

é o nome dunha carpeta na cal colocamos que é o que se supon que que mvc .net poña en certo tipos de datos determinados ejemplo datetime que o poña como date o boolean como si/no.

Se a poñemos na vista Home solo tendra efecto en Home pero se a poñemos na vista shared ten efecto en todas.

Editor

Nos mostra unha clase ou objeto con textbox de edicion

##### EditorTemples

para modificar as propiedades de Editor.

#### BeginForm

Facer un formulario utilizando o metodo BeginForm.Para usalo temos que facer Post.

BeginForm</stron>por defecto va a ir hacia el mismo action lo esta llamando pero podemos indicar que action ejecuta, metodo(get,post...). Se encarga de facer a relacion entre cada texbox e a propiedad que lle corresponda

@using (Html.BeginForm())//Html.BeginForm("EjemploBeginForm","Home") podemos indicar a action e o controler por si tivese outro nome distinto ó get:

{

@\*<div>

@Html.LabelFor(Model => Model.Titulo)

@Html.EditorFor(Model => Model.Titulo) </div> @\*Usamos EditorFor en lugar de Editor porque temos un modelo\*@

@Html.EditorForModel(); //crea o formulario solo, podemos modificalo en EditorTemplates

<input type="submit" value="Enviar" />

}

## LECCIONS

### APLI\_INTRO

#### 1-Introduccion

#### 2-Controladores y Acciones

#### 3-Routing y RoutingConfig

#### 4-ActionResult

#### 5-ContenctResult

#### 6-Json y JsonResult

#### 7-Redirect,RedirectToAction y RedirectToRoute

#### 8-HTTPStatus y HTTPStatusCodeResult

#### 9-FileResult

#### 10-QuertString

#### 11-HttpGet y HttpPost

#### 12-Viewbag yViewData

#### 13-Introduccion a vistas

#### 14-Loyout y shared folder

#### 15-Código en vistas razor

#### 16-Metodo Raw

#### 17-ActionLink

#### 18-RenderAction

#### 19-Display y DisplayTemplates

#### 20- Edito y EditorTemplates

#### 21-DropDownList

#### 22-DropDownList desde un enum sin modelo

#### 23-BeginForm en razor

#### 24-PartialView

#### 25-Url por defecto al ejecutar el aplicativo

#### 26-Modelos, Scaffolding y Entity Framework

#### 27-Introducción al Scaffolding: Controladores y Vistas | Modelos

#### 28- Analizando Scaffolding Create | Modelos

#### 29- Analizando Scaffolding Edit | Modelos

#### 30-Analizando Scaffolding List

#### 31-Analizando Scaffolding Delete

#### 32-Analizando Scaffolding Details