Mi primera aplicación ASP.NET MVC 2 paso a paso – parte 3 « afelipelc Blog

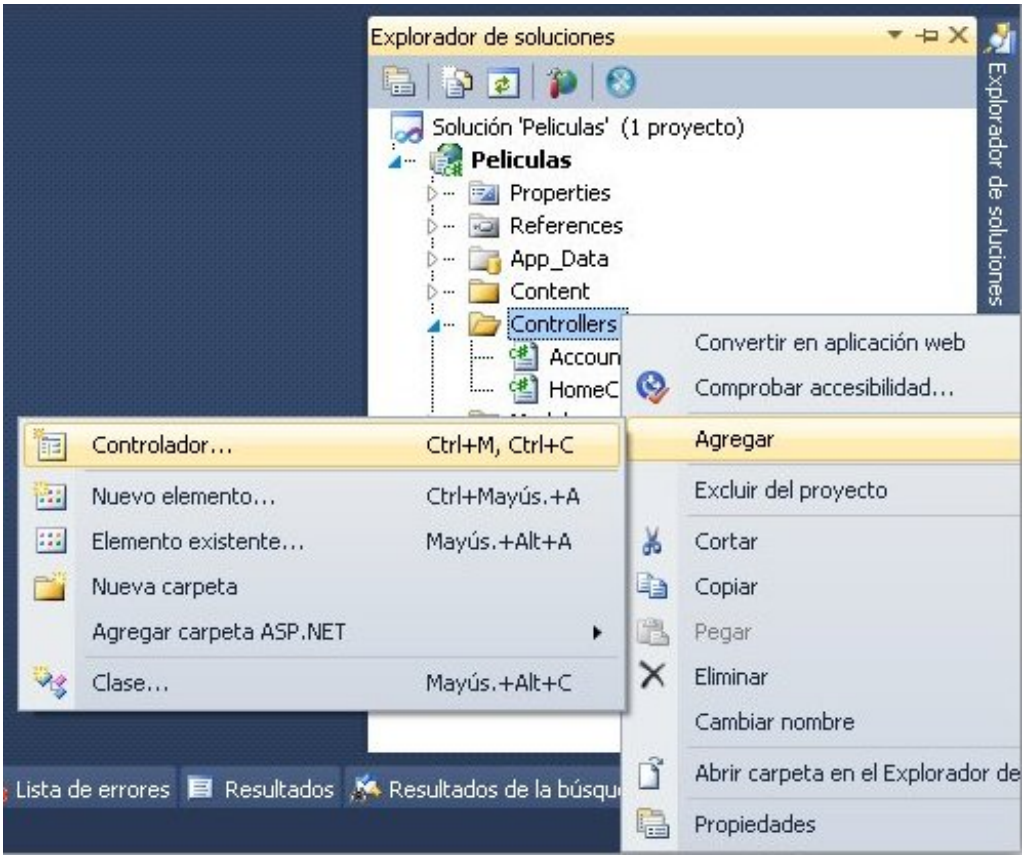
Continuando con el desarrollo de nuestra aplicación ASP.NET MVC 2… del tutorial original : Build your First ASP.NET MVC Application :: http://www.asp.net/mvc/tutorials/getting-started-with-mvc-part1

En esta parte, crearemos el controlador (Peliculas) que en la página principal mostrara la lista de géneros, donde al elegir alguno, cargara la lista de las Películas de dicho Género.

Agregar una clase controlador >> Controller

Los controladores definen métodos también llamados acciones (Un controlador aparte de ejecutar todas las acciones definidas, podemos decir que nos creará otro nivel (sección) en nuestro sitio). Como en la 1ra parte comentamos que el controlador Home, es el que ejecuta las acciones de la página principal de nuestro sitio http://localhost/ que es igual a http://localhost/Home entonces si agregamos un controlador llamado Peliculas, como resultado tendremos la dirección http://localhost/Peliculas

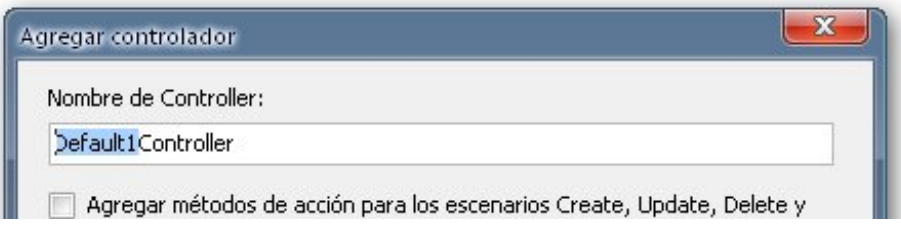
Para agregar el controlador llamado Peliculas, nos ubicamos sobre la carpeta Controllers, clic derecho, Agregar / Controlador.



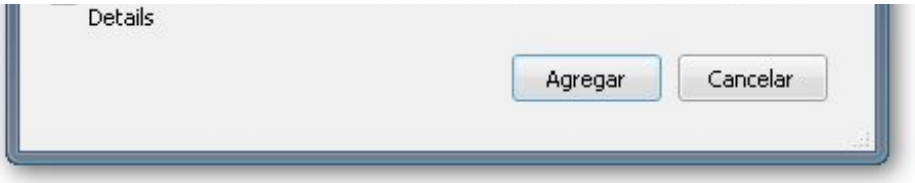
Img. 1.- Agregar un controlador

En la ventana especificamos el nombre

Img. 2.- Asignar nombre al controlador



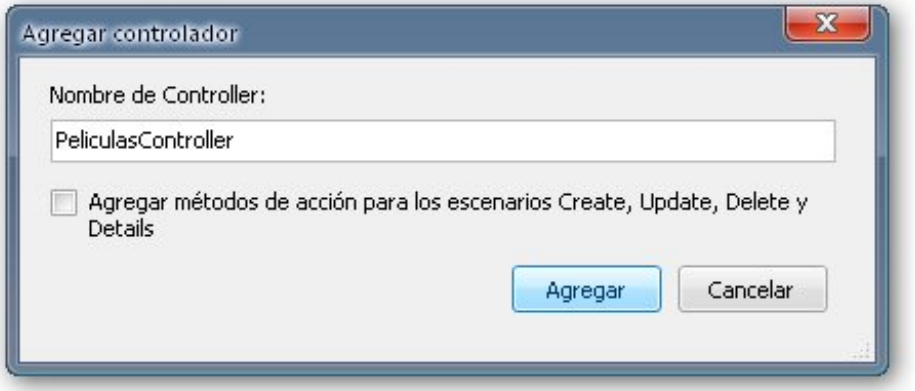
Importante: Al mostrarse esta ventana, automáticamente nos agrega el sufijo "**Controller**" al nombre del controlador que aparece con resalto "DefaultN",



entonces si nuestro controlador va a llamarse **Peliculas**, cambiaremos DefaultN por Peliculas quedando **PeliculasController** y damos Agregar **sin** activar “Agregar métodos de acción…” ya que nosotros agregaremos acciones personalizadas.

Nota: la palabra **Controller No debe ser eliminada**, ASP.NET busca las clases que llevan este sufijo para identificarlas como controladores.

Img. 3.- Agregar controlador Peliculas



La clase controlador llamada Peliculas ha sido creada y empezaremos a agregar nuestro código.

Debemos agregar el using del (los) Modelo(s). Que son todas las clases definidas dentro del espacio de nombres **Peliculas.Models** (Se encuentran dentro de la carpeta Models)

```
using Peliculas.Models;

//Dentro de
public class PeliculasController: Controller
{
//Declaramos una instancia de nuestro EntityModel llamado PeliculasEntities
PeliculasEntities DB = new PeliculasEntities();
```

Nota: Cabe mencionar que al tener un elemento ADO.NET Entity Data Model, las tablas son llamadas Entidades, que dentro de un modelo, **una entidad es una clase** donde **los campos se convierten en propiedades** que podemos llamarlas y asignarles un valor igual a una propiedad de cualquier otra clase.

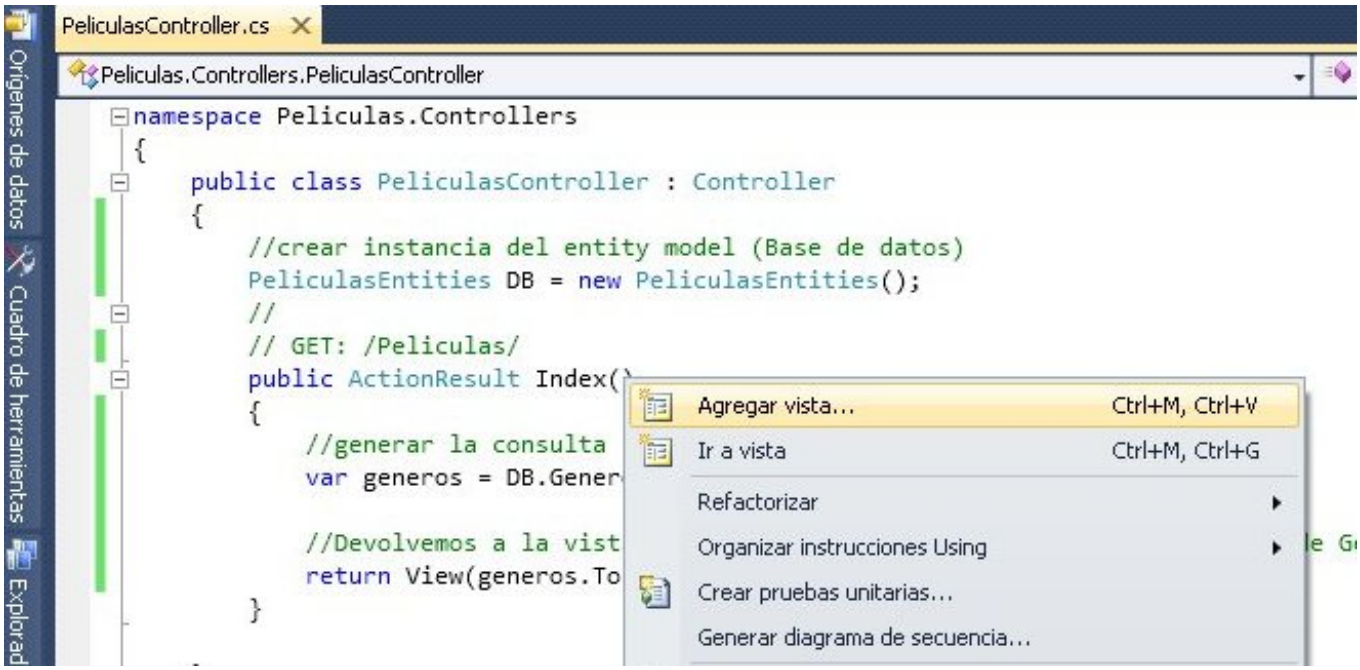
Entonces como lo primero que debe mostrase en http://localhost/Peliculas será la lista de Géneros disponibles, en la acción (Metodo) Index() que es la accion por defecto del controlador creado, creamos una variable que cargue el contenido de la entidad Genero de nuestro modelo.
El código del controlador Peliculas queda así:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

//Declaramos el using de nuestros modelos
using Peliculas.Models;

namespace Peliculas.Controllers
{
    public class PeliculasController : Controller
    {
        //crear instancia del entity model (Base de datos)
        PeliculasEntities DB = new PeliculasEntities();

        //
        // GET: /Peliculas/
        public ActionResult Index()
        {
```

```
            //cargar la entidad Genero en la variable generos
            var generos = DB.Genero;

            //Devolvemos a la vista Ixdex.aspx del controlador Pelicula la lista de Ge
            // que deberá mostrar al usuario
            return View(generos.ToList());
        }

    }
}
```

Cada acción de un controlador tiene definida una vista, por ejemplo la acción Index() tiene una vista llamada Index.aspx, la acción Buscar() => Buscar.aspx, Detalles() => Detalles.aspx, etc…

Para crear la vista de una acción, dar clic derecho dentro de la acción (Index() en este caso) / Agregar Vista.



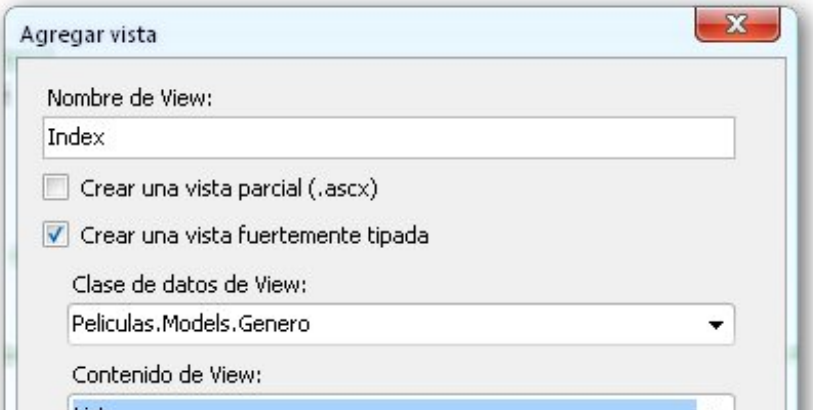
Img. 4.- Agregar la Vista de una Accion

Cuando muestre la ventana Agregar Vista, el nombre lo dejamos igual que el de la acción, como lo que vamos a mostrar es el contenido de la entidad Genero, elegimos **Crear una vista fuertemente tipada**, en Clase de datos de view, elegimos la entidad que vamos a mostrar, en este caso **Genero** (Peliculas.Models.Genero), en contenido de view, seleccionamos **List** que generará automáticamente el código que mostrara la lista de géneros en la página, dar clic en Agregar.

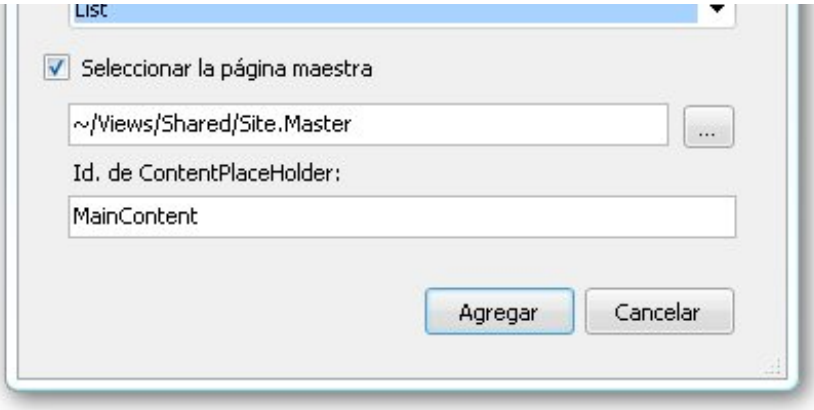
La vista Index.aspx es la pagina por defecto que se mostrara cuando ingresemos a http://localhost/Peliculas

Podemos observar que la plantilla donde se mostrara el contenido de la vista es Site.Master y el contenedor se llama MainContent (que fue lo que vimos en la primera parte).

Img. 5.- Crear la Vista Index.aspx de la acción Index() de controlador Peliculas



El código generado es



Eventos y objetos de servidor

```
<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
    Index (Titulo de nuestra Pagina)
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2>Index (Titulo del contenido de la accion Index)</h2>

    <table>
        <tr>
            <th></th>
            <th>
                GeneroId
            </th>
            <th>
                Nombre
            </th>
        </tr>

    <% foreach (var item in Model) { %>

        <tr>
            <td>
                <%: Html.ActionLink("Edit", "Edit", new { id=item.GeneroId }) %> |
                <%: Html.ActionLink("Details", "Details", new { id=item.GeneroId })%> |
                <%: Html.ActionLink("Delete", "Delete", new { id=item.GeneroId })%>
            </td>
            <td>
                <%: item.GeneroId %>
            </td>
            <td>
                <%: item.Nombre %>
            </td>
```

Img. 6.- Código de la vista Index.aspx del controlador Peliculas

Como la tabla Genero de la BD está vacía, agreguemos algunos géneros conocidos (abrir Peliculas.mdf e ir a tablas).

Después de agregar algunos registros, ejecutamos nuestra aplicación y accedemos a Peliculas en la barra de direcciones >> ejemplo: http://localhost:7406/Peliculas ya que aún no tenemos un link que nos lleve a esa dirección.

Nota: El servidor IIS con el que cuenta Visual Studio para ejecutar los proyectos ASP.NET abre un puerto de nuestro localhost para nuestra aplicación, en mi caso es el puerto 7406, si ejecuto mi aplicación, me la muestra en http://localhost:7406/

Img. 7.- La vista Index.aspx mostrando los resultados

Pero como al usuario no vamos a mostrarle los links para agregar, eliminar o editar algún registro, eliminamos la columna de la tabla generada que contiene los Html.ActionLink junto con la columna que muestra el campo GeneroId para no mostrar los índices.

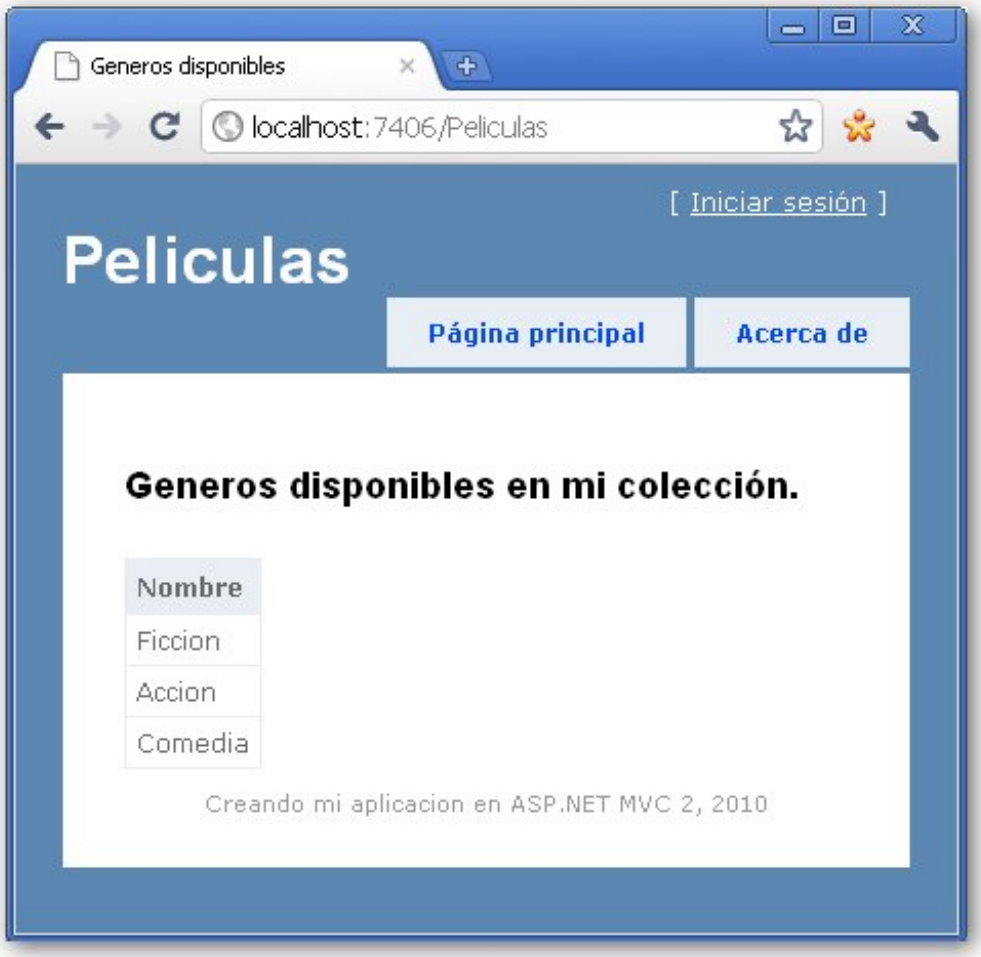
El código de nuestra vista Index.aspx del controlador Peliculas quedaría así:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master" Inherits='

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
    Generos disponibles
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2>Generos disponibles en mi colección.</h2>

    <table>
        <tr>
            <th>
                Nombre
            </th>
        </tr>

    <% foreach (var item in Model) { %>

        <tr>
            <td>
                <%: item.Nombre %>
            </td>
        </tr>

    <% } %>
```

```
    </table>
</asp:Content>
```

Observa que la lista de géneros (Obtenida del modelo pasado por la acción Index() en el controlador) es impresa recorriendo los elementos del modelo con un foreach.

Si volvemos a ejecutar nuestro proyecto y nos vamos otra vez a Peliculas, ahora tenemos:



Img. 8.- Personalizando la vista Index.aspx

Para tener acceso a la sección Peliculas, agregamos el link a nuestra plantilla (Site.Master en el explorador de soluciones y a la carpeta Views/Shared/) en la lista “menu”:

```
<ul id="menu">
    <li><%: Html.ActionLink("Página principal", "Index", "Home")%></li>
    <li><%: Html.ActionLink("Peliculas", "Index", "Peliculas")%></li>
    <li><%: Html.ActionLink("Acerca de", "About", "Home")%></li>
</ul>
```

Y al actualizar la página obtenemos:

Img 9.- Resultado de agregar links a la plantilla Site.Master

Ahora vamos a hacer que al dar clic sobre algún Género, nos cargue la lista de las películas de ese género.

Detener la ejecución del proyecto…

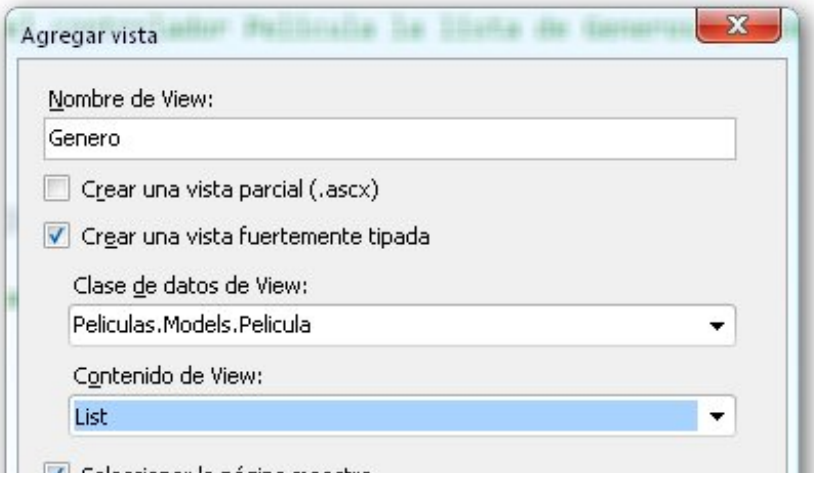
Definimos una nueva acción en el controlador Peliculas llamado **Genero()** debajo de Index() { }, que pedirá como parámetro el nombre del genero a mostrar.

```
// GET: /Peliculas/Genero/
public ActionResult Genero(string genero)
{
   //hacer una consulta al modelo, cargando el género buscado incluyendo
   //todas las películas relacionadas a ese genero

   //creamos nuestra consulta LINQ

  var PeliculasGenero = DB.Genero.Include("Pelicula").Single(g => g.Nombre == genero);

  //Como a la vista solo se le pasa el modelo que contiene la lista de películas
  //del género seleccionado
  //Pasamos a la vista algunos datos por medio de parámetros
  ViewData["Genero"] = genero; //Nombre del genero

  //Devolver el modelo que contiene todas las películas encontradas
  return View(PeliculasGenero.Pelicula.ToList());
 }
```

Creamos la vista fuertemente tipada de la clase Pelicula, con contenido List

Img. 10.- Crear la vista Genero.aspx

Antes de ver el resultado, debemos agregar algunos registros de películas manualmente a la tabla Pelicula de la BD Peliculas.mdf



PeliculaId	Titulo	FechaLanzamiento	GeneroId	Precio	Disponible
1	Identidad Sustituta ...	25/09/2009	1	80	True
2	Bourne Ultimatum ...	15/06/2007	1	80	True
3	Dejavu ...	22/02/2004	2	70	True
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

4 de 4

Img. 11.- Agregar registros a la tabla Pelicula

Nos falta generar los Links de la lista de Géneros en la vista Index.aspx

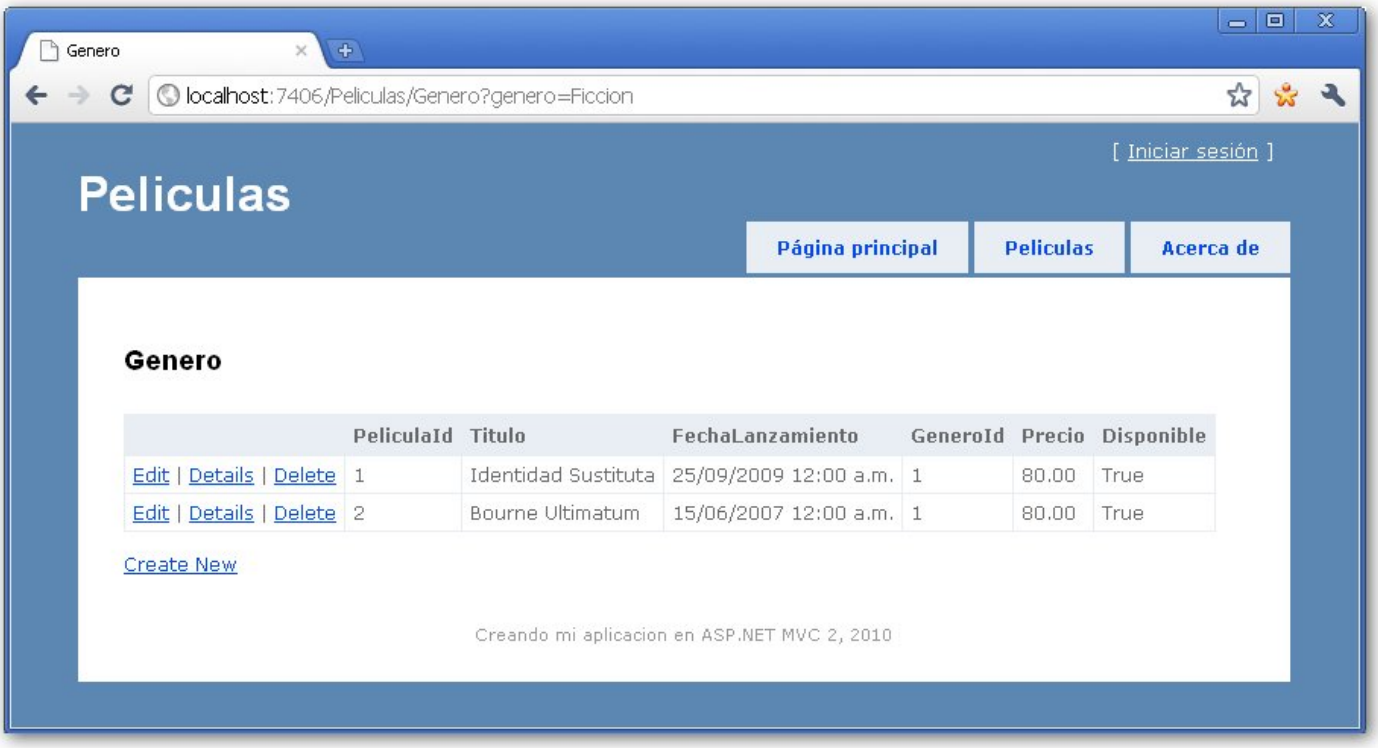
Reemplazamos el código que imprime el nombre de los generos

<%: item.Nombre %>

Por:

```
<%// Genera un link con parámetros del tipo: http://localhost:7406/Peliculas/Genero?ge
<%: Html.ActionLink(item.Nombre, "Genero", "Peliculas", new { genero = item.Nombre },
```

Ejecutamos y vemos los resultados en http://localhost:7406/Peliculas/ dando clic en algún género.



Img. 12.- Resultado de la vista Genero.aspx mostrando las peliculas del genero seleccionado.

Igual que en la lista de géneros, al usuario no vamos a mostrarle los links de Editar, Eliminar o crear nuevo registro, entonces eliminamos esa columna y la de PeliculaId. Pero si vamos a darle la opción de ver los detalles de la película, para ello convertimos en Link el título de las películas.

El código de la vista Genero.aspx mostrando también lo que recibe por el parámetro **ViewData**["Genero"] y el título de la película convertido en link con algunas otras modificaciones como mostrar SI en vez de True y No en vez de False (en Disponible), quedara así:

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master" Inherits='

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
  <% //Generamos el título de nuestra página %>
 Clasificación por género <%: ViewData["Genero"] %>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
    <% //Generamos el título del contenido de nuestra página y mostramos el total de p
    <h2>Clasificación por género <%: ViewData["Genero"] %></h2>
    <p>(<%: Model.Count()%>) películas de este género.</p>

    <table>
        <tr>
             <th>
                Titulo
            </th>
            <th>
                <%//Cambio el texto de FechaLanzamiento a Fecha de Lanzamiento %>
                Fecha de Lanzamiento
            </th>
            <th>
                Género
            </th>
            <th>
                Precio
            </th>
            <th>
                Disponible
            </th>
        </tr>

    <% foreach (var item in Model) { %>

        <tr>
            <td>

                <% //Convertimos item.Titulo en un vinculo que nos lleve a detalles de
                   //Genera el link http://localhost:7406/Peliculas/Details/N %>
                <%: Html.ActionLink( item.Titulo, "Detalles", new { id=item.PeliculaIc
            </td>
            <td>
                <% //Para que muestre el formato dd/mm/yy cambiamos de "{0:g}" a "{0:c
                <%: String.Format("{0:d}", item.FechaLanzamiento) %>
            </td>
            <td>
            <% //Para mostrar el nombre del Genero en lugar del GeneroId
               //Cambiamos item.GeneroId por item.Genero.Nombre (obteniendo los datos
                <%: item.Genero.Nombre %>
            </td>
            <td>
                <%: String.Format("{0:F}", item.Precio) %>
            </td>
            <td>
                <% //item.Disponible muestra True o False (es de tipo Boleano), debemc
                <% try
```

```
                    {
                        if ((bool)(item.Disponible))
                        { %>
                             Si
                             <%
                         }
                        else
                        { %>
                               No
                          <% }
                    }
                    catch
                    {
                    %>
                     No <% //Si Disponible es NULL, se producira un error, entonces lo
                <%
                    } %>
            </td>
        </tr>

    <% } %>

    </table>
</asp:Content>
```

El resultado es:

Img. 13.- Vista Genero.aspx personalizada.

Hasta aquí ya tenemos nuestra tabla de datos personalizada.

En este módulo nos falta crear la acción Detalles().

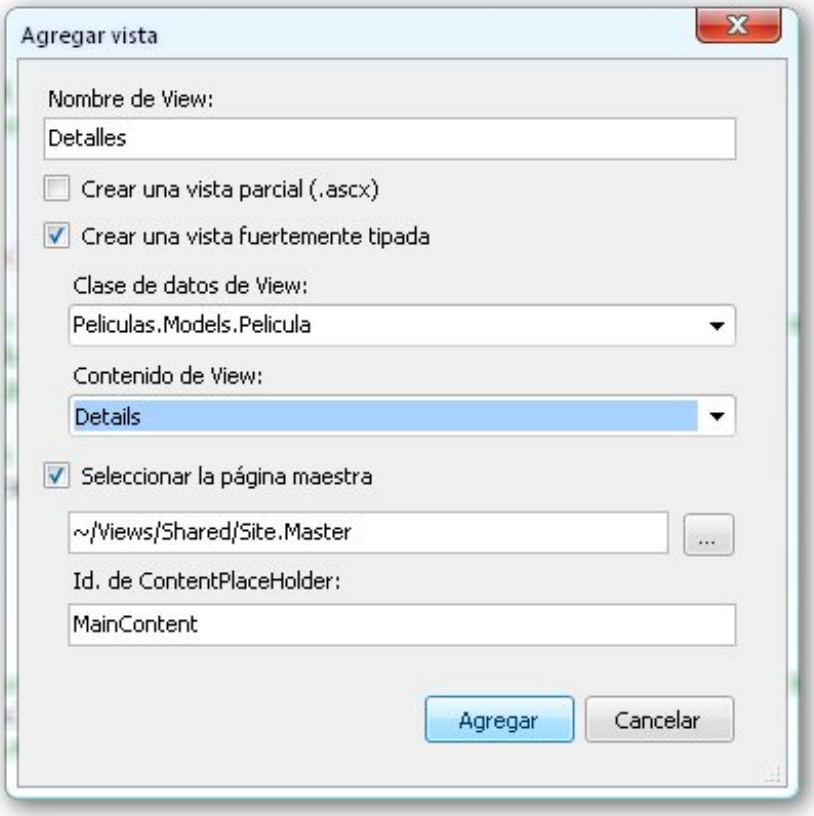
Debajo de la acción Genero() del controlador Peliculas, definimos la acción Detalles() que recibe como parámetro PeliculaId.

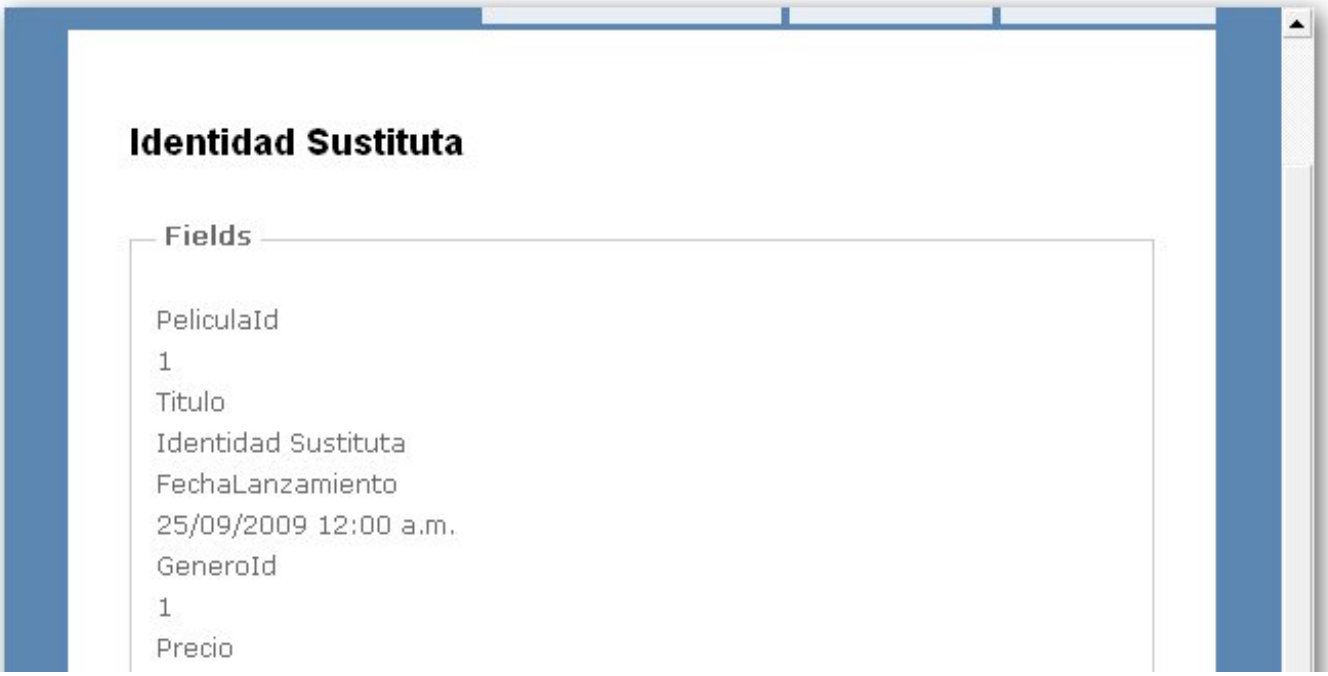
```
// GET: /Peliculas/Detalles/N
public ActionResult Detalles(int id)
{
   //Creamos la consulta LINQ que obtiene un solo registro, donde Pelicula.PeliculaId
   //igual al ID pasado en el parametro id
   var pelicula = DB.Pelicula.Single(p => p.PeliculaId == id);
   //Devolvemos el modelo (resultado) a la vista
   return View(pelicula);
}
```

Creamos la vista fuertemente tipada de la clase Pelicula y en contenido elegimos Details. Esto hará que se muestren los detalles del registro.

Img. 14.- Crear la vista Detalles.aspx

Ejecutamos y seleccionamos algún título de nuestra lista de películas.



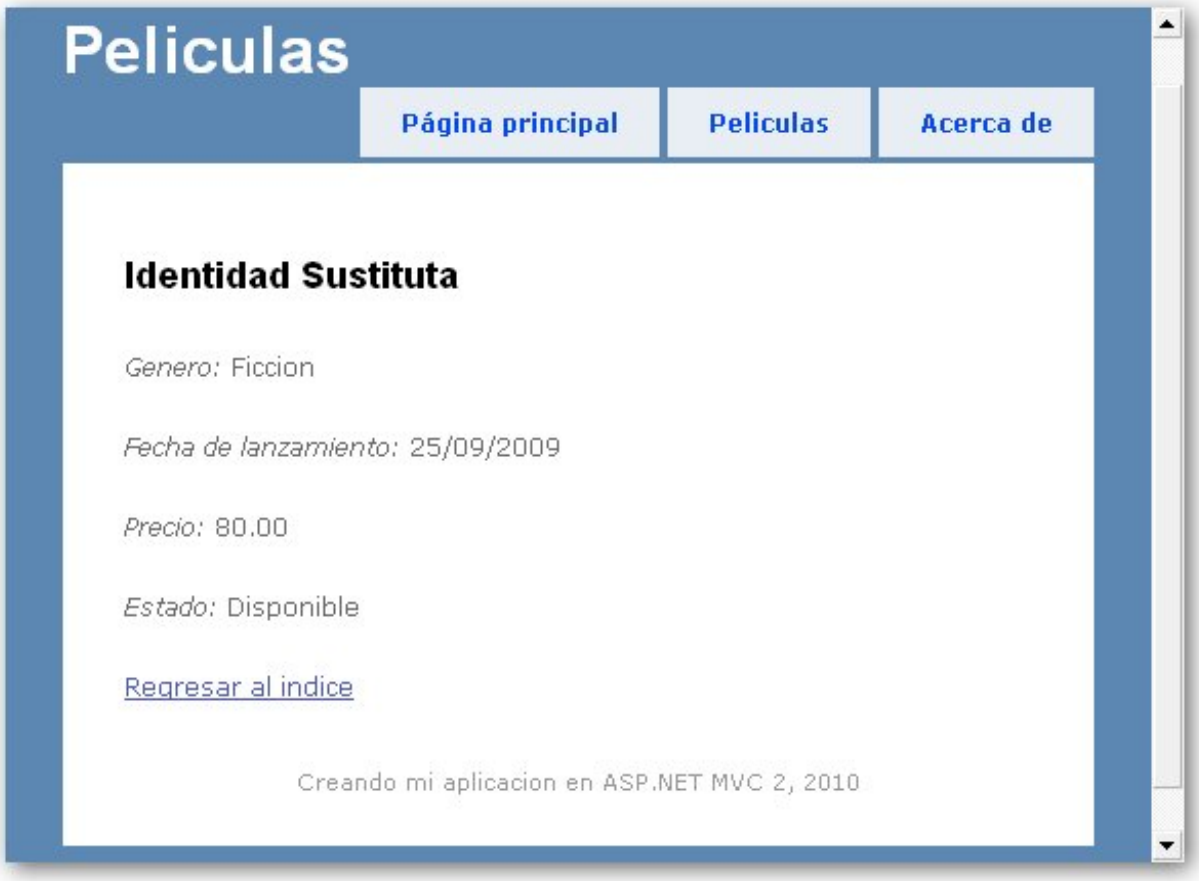


Img. 15.- Resultado de la vista Detalles.aspx

Personalizando el código de la vista Detalles.aspx

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master" Inherits='

<asp:Content ID="Content1" ContentPlaceHolderID="TitleContent" runat="server">
 Pelicula  <%: Model.Titulo %>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">

    <h2><%: Model.Titulo %></h2>

    <p>
        <em>Genero: </em>
        <%: Model.Genero.Nombre %>
    </p>
    <p>
        <em>Fecha de lanzamiento: </em>
        <%: String.Format("{0:d}", Model.FechaLanzamiento)%>
    </p>
    <p>
        <em>Precio: </em>
        <%: String.Format("{0:F}", Model.Precio) %>
    </p>

    <p>
        <em>Estado: </em>
    <% try
       {
           if ((bool)(Model.Disponible))
           { %>
               Disponible
              <%
           }
           else
           { %>
               No disponible
             <%
           }
       }
       catch
       {
          %>
          No disponible
          <%
       } %>
</p>
    <p>
        <%: Html.ActionLink("Regresar al indice", "Index") %>
    </p>

</asp:Content>
```

Ejecutamos el proyecto y obtenemos.



Img. 16.- Resultado de la Vista Detalles.aspx personalizada.

Hasta aquí termina nuestro controlador llamado Películas que nos genera el catálogo de Generos que el usuario puede elegir para ver el catálogo de películas de cada género.

En la siguiente parte, crearemos el controlador Administrar (donde podremos agregar, modificar y eliminar los registros)

Mi primera aplicación ASP.NET MVC 2 paso a paso – parte 3, 5.0 out of 5 based on 1 rating