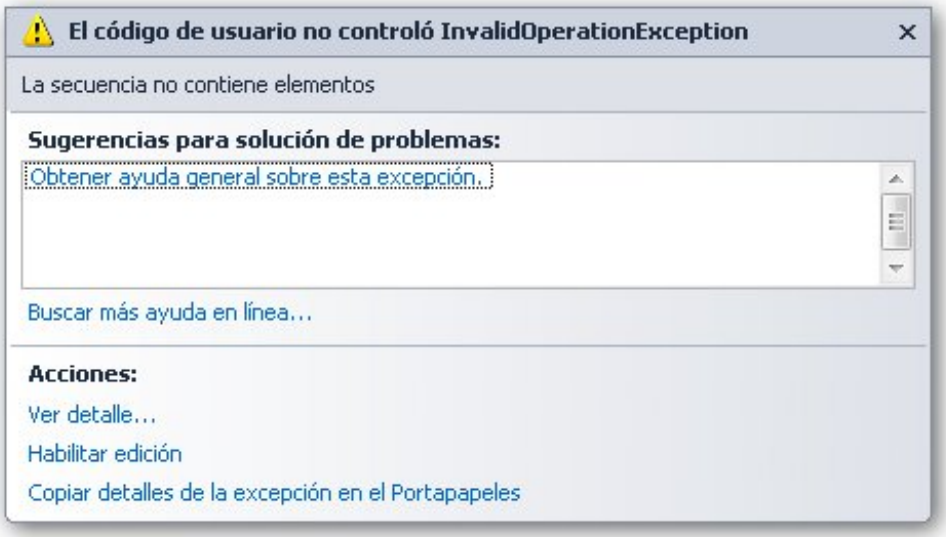
Controlando los errores que podrían producirse « afelipelc Blog

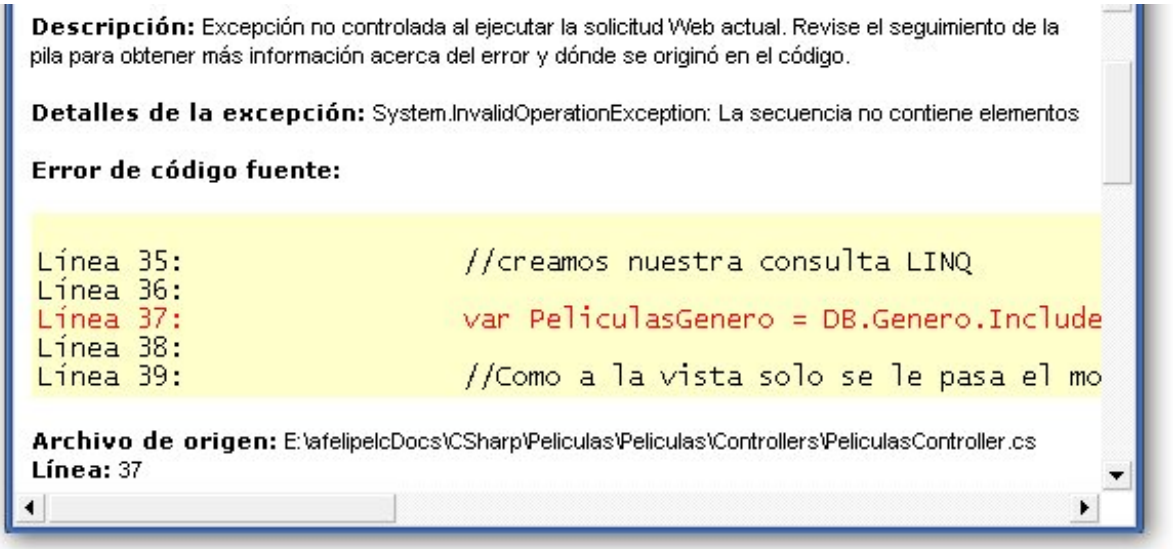
Cuando probamos la funcionalidad de nuestro proyecto de una manera normal (por decirlo así), no se produce ningún error, ya que accedemos a los vínculos creados automáticamente por el sistema.

Img 1.- Funcionando de manera normal…

Como se observa en la imagen anterior, al acceder a algún género, no pasa nada, pero si modificamos el parámetro **?genero=Ficcion** por **?genero=FiccionEDCVRR**
Se producirá el error:

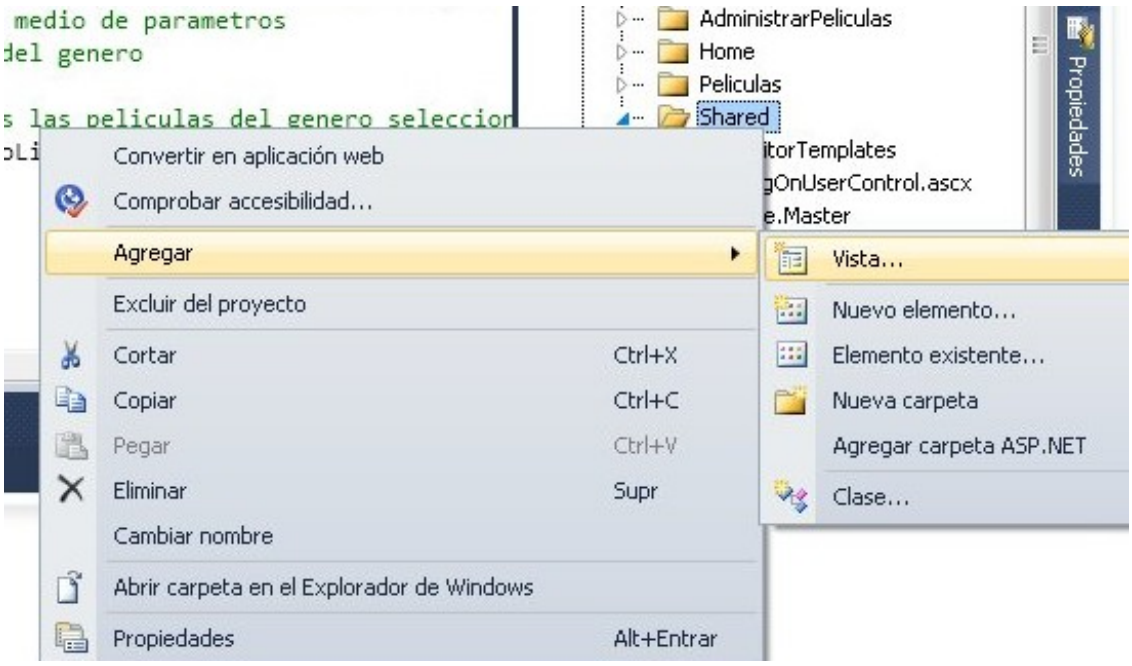
Img 2.- Error: La secuencia no contiene elementos…





Img 3.- Error mostrado en el navegador.

Ya que como el género **FiccionEDCVRR** no existe en nuestra BD, En la acción Genero el error se produce porque debería devolver un objeto como resultado.
Para controlar los errores “La secuencia no contiene elementos”
Crearemos una vista llamada Error.aspx en la carpeta Shared, de esta forma, los demás controladores podrán acceder a esta vista.



Img 4.- Agregar la vista Error en la carpeta Shared.

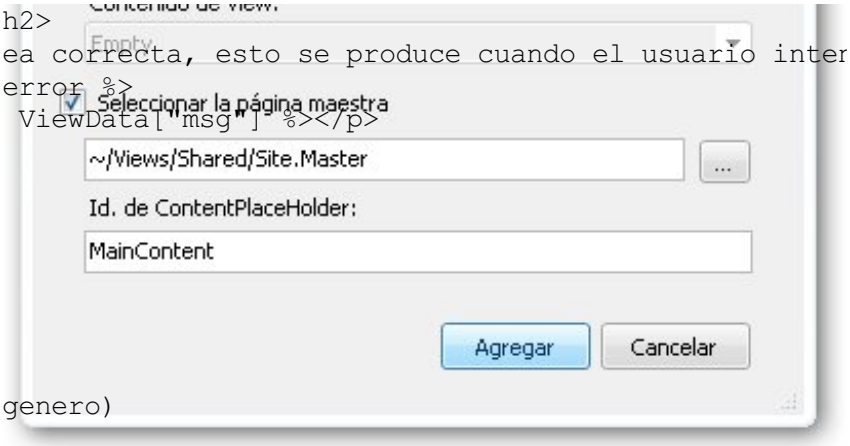
Creamos la vista Error vacía.

Img 5.- Crear la vista Error (Vacia)

Personalizamos el código para mostrar al usuario:


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

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

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

```
    <h2>Se ha producido un error</h2>
    <p>Verifica que la dirección sea correcta, esto se produce cuando el usuario inter
    <% //Mostramos el mensaje del error %>
    <p><b>Datos del error:</b> <%: ViewData["msg"] %></p>
</asp:Content>
```



Agregamos el bloque Try Catch y regresamos la vista Error, el código de la acción Genero ya modificado quedaría así:

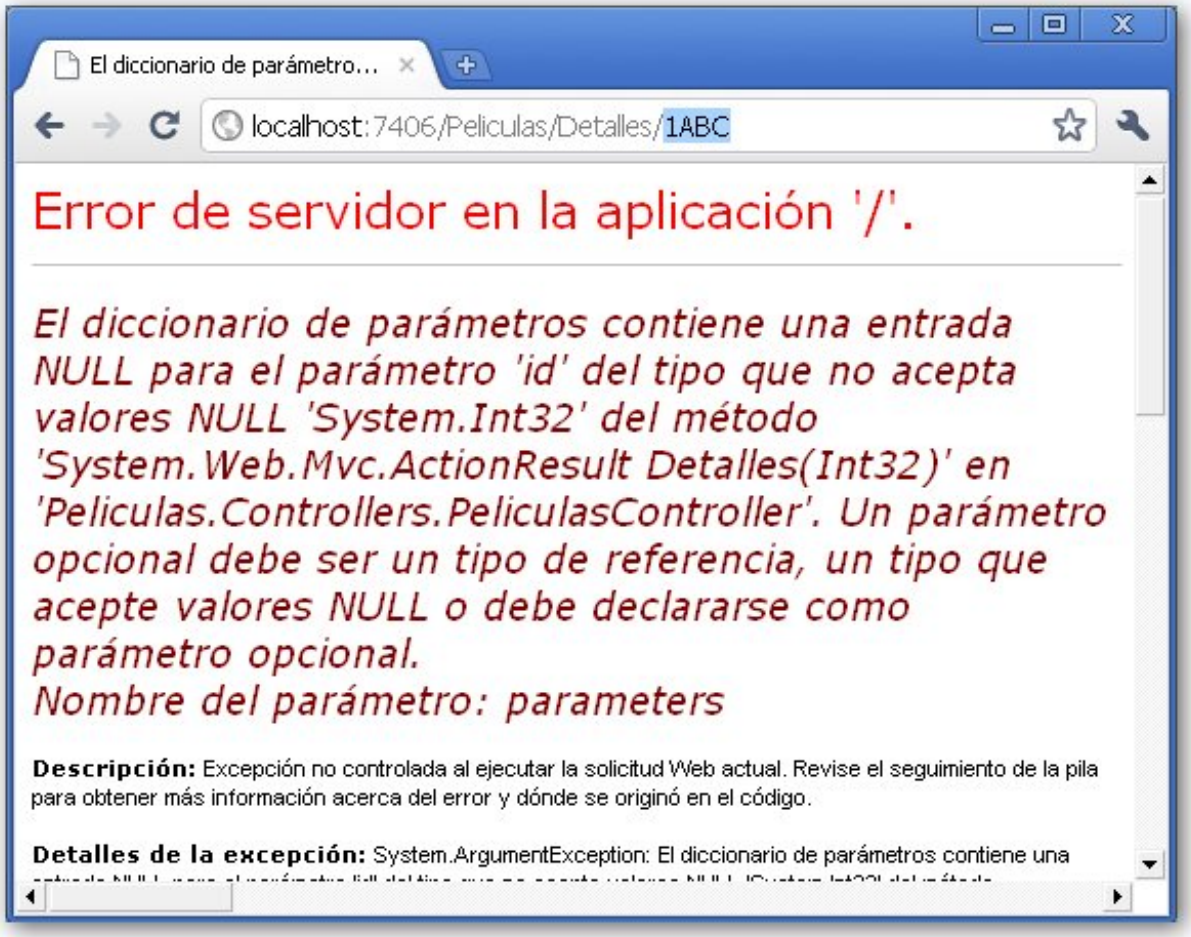
```
// GET: /Peliculas/Genero/
public ActionResult Genero(string genero)
{
    try
    {
        //hacer una consulta al modelo, cargando el genero buscado incluyendo
        //todas las peliculas relacionadas a ese genero

        //creamos nuestra consulta LINQ

        var PeliculasGenero = DB.Genero.Include("Pelicula").Single(g =&gt; g.Nombre ==

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

        //Devolver el modelo que contiene todas las peliculas del genero seleccionado
        return View(PeliculasGenero.Pelicula.ToList());
    }
    catch (Exception e)
    {
        ViewData["msg"] = e.Message;
        return View("Error");
    }
}
```

De la misma forma puede invocarse la vista Error en cualquier otra acción que pudiera producir esta excepción al modificar los parámetros como la acción Detalles(), en AdministrarPeliculas la acción Editar() y Eliminar().
Ejecutando el proyecto y realizando nuevamente el ataque, este es el resultado:

Img 6.- Resultado del manejo de errores.

Hasta aquí parece todo estar bien , pero hay otros errores que podrían producirse, los errores que se

producen en un sitio web pueden ser, Error 404 (Archivo no encontrado), Error 500 (Error en el servidor), etc. Incluso cuando un parámetro del tipo **ENTERO** es combinado con un **STRING**, por ejemplo al intentar ver los detalles de una película de esta forma.

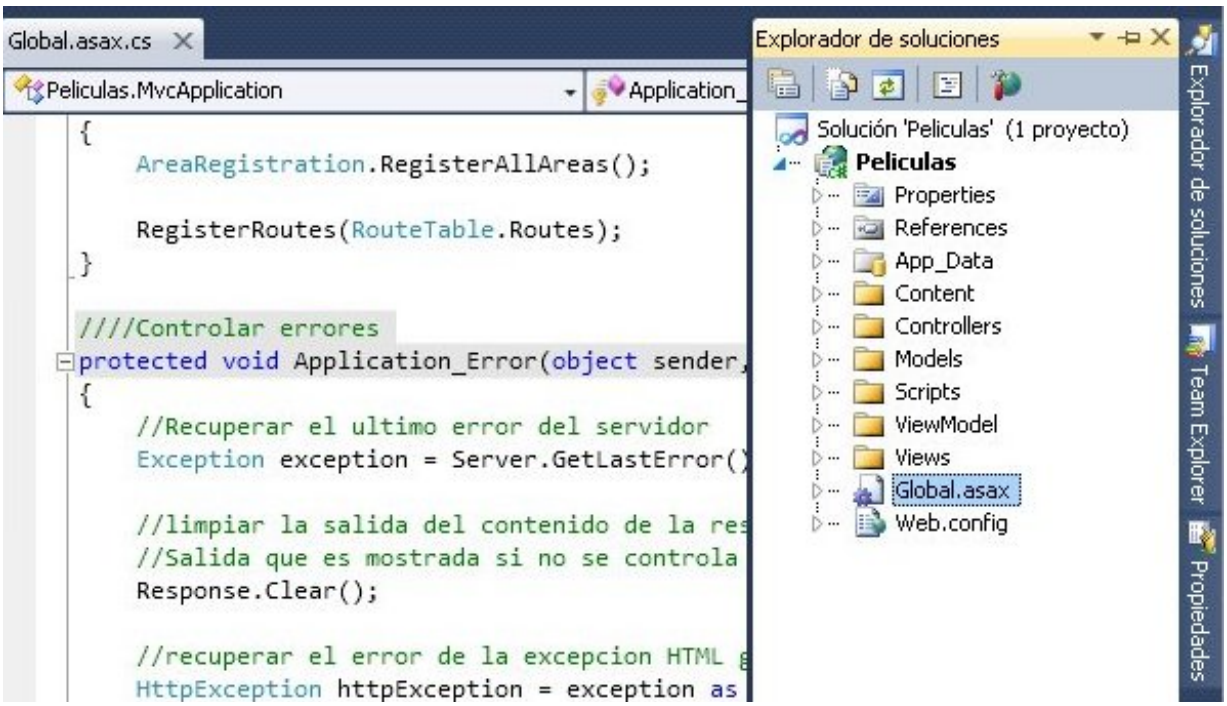


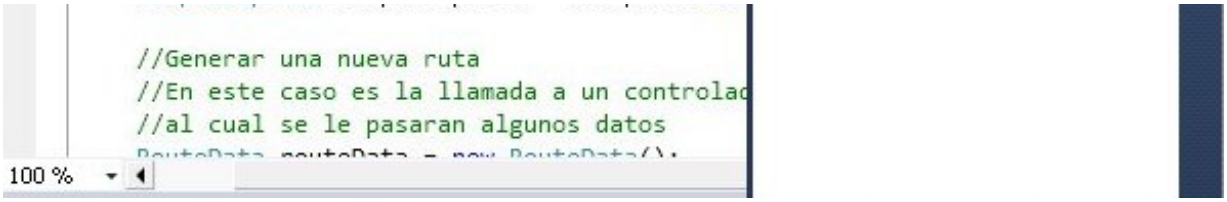
Img 7.- Error al cambiar el tipo de dato del parametro.

Aun con el Try Catch agregado a la acción Detalles, este tipo de error no es controlado porque se produce al invocar a la acción Detalles(), y no se puede agregar un try catch al inicio de la clase PeliculasController.

Para controlar **TODOS** los errores que podrían producirse en nuestro sitio, haremos lo siguiente:

Agregaremos un método al archivo Global.asax de nuestro proyecto Peliculas.





Img 8.- Abrir el archivo Global.asax

El método **Application_Error()** contendrá el código que controlara todos los errores que puedan producirse e invocar al controlador Error que posteriormente agregaremos, esto hara que en lugar de mostrar la pantalla de error en tiempo de ejecución tradicional, se muestre una vista con todo el contenido de dicho error.

Primero agregamos **using Peliculas.Controllers;** a nuestro archivo Global.asax

Agregamos el método Application_Error() debajo del método Application_Start()

```
//Controlar errores
protected void Application_Error(object sender, EventArgs e)
{
//Recuperar el ultimo error del servidor
Exception exception = Server.GetLastError();

//limpiar la salida del contenido de la respuesta html
//Salida que es mostrada si no se controla el error
Response.Clear();

//recuperar el error de la excepción HTML generada
HttpException httpException = exception as HttpException;

//Generar una nueva ruta
//En este caso es la llamada a un controlador
//al cual se le pasaran algunos datos
RouteData routeData = new RouteData();
//agregar un parámetro para la ruta a invocar
//Error es el nombre del controlador a invocar
routeData.Values.Add("controller", "Error");

if (httpException == null)
{
    //Si no hay información del error, redirige a la acción
    //Index() del controlador ErrorController
    routeData.Values.Add("action", "Index");
}
else //Si se genera una Exception Http, debe ser controlada
{
    switch (httpException.GetHttpCode())
    {
        //Segun el numero de error, redirigir a una accion
        //definida en el controlador ErrorController
        case 404:
            // Pagina no encontrada
            //Redirige a la accion "action" HttpError404()
            routeData.Values.Add("action", "HttpError404");
            break;
        case 500:
            // Error del servidor.
            //Redirige a la accion "action" HttpError500()
            routeData.Values.Add("action", "HttpError500");
            break;
        // Algun otro error
        //Redirige a la accion "action" General()
        default:
```

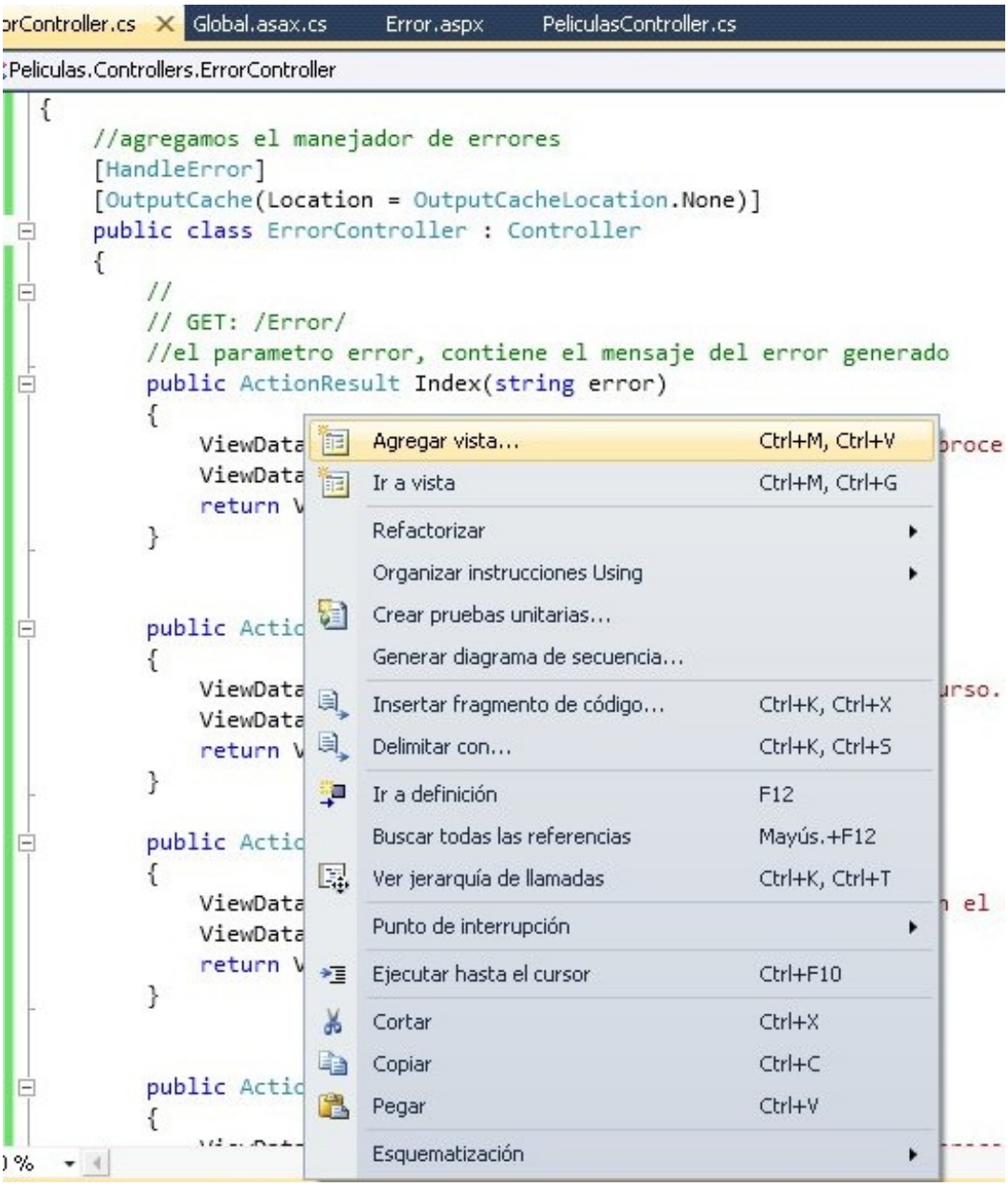
```
            routeData.Values.Add("action", "General");
            break;
    }
}

// Pasar el mensaje de la excepcion al parametro error.
routeData.Values.Add("error", exception);

// Limpiar la excepcion en el servidor.
Server.ClearError();

//crear una instancia del controlador
// llamar al controlador ErrorController pasandole los parametros.
IController errorController = new ErrorController();
errorController.Execute(new RequestContext(
    new HttpContextWrapper(Context), routeData));
}
```

Como este método invoca al controlador Error, entonces creamos ese controlador.
Código del controlador Error:

```
using System;
using System.Web;
using System.Web.Mvc;

//Agregamos:
using System.Web.UI;
//Define las propiedades, métodos y eventos que comparten todos los controles de servi

namespace Peliculas.Controllers
{
    //agregamos el manejador de errores
    [HandleError]
    [OutputCache(Location = OutputCacheLocation.None)]
    public class ErrorController : Controller
    {
        //
        // GET: /Error/
        //el parametro error, contiene el mensaje del error generado
        public ActionResult Index(string error)
        {
            ViewData["Titulo"] = "Un error ha ocurrido mientras se procesaba la solici
            ViewData["Descripcion"] = error;
            return View();
        }

        public ActionResult HttpError404(string error)
        {
            ViewData["Titulo"] = "Error 404: No en encuentra el recurso.";
            ViewData["Descripcion"] = error;
            return View("Index");
        }

        public ActionResult HttpError500(string error)
        {
            ViewData["Titulo"] = "Error 500: Un error ha ocurrido en el servidor mient
            ViewData["Descripcion"] = error;
            return View("Index");
        }

        public ActionResult General(string error)
        {
            ViewData["Titulo"] = "Un error ha ocurrido mientras se procesaba la solici
            ViewData["Descripcion"] = error;
```

```
            return View("Index");
        }
    }
}
```

Esto aún no funciona , solo falta crear la vista Index del controlador Error.



Img 9.- Crear la vista Index del ErrorController.

Img 10.- Crear la vista Index del ErrorController.

La vista Index recuperara los datos pasados por ViewData y mostrarlos al usuario:

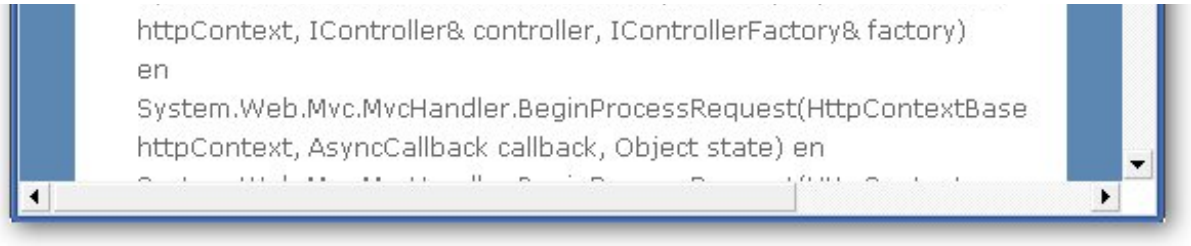

```
<%@ Page Title="" Language="C#" MasterPageFile="~/Views/Shared/Site.Master" Inherits='

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

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent" runat="server">
     <h2><%= Html.Encode(ViewData["Titulo"]) %></h2>
```

```
    <p><b>Detalles del error</b>: <br /><%= Html.Encode(ViewData["Descripcion"]) %></p>
</asp:Content>
```

Ejecutando el proyecto y probando nuevamente el ataque, este será el resultado:

Img 11.- Resultado del control de errores…

Y si intentamos acceder a una dirección que no existe:



Img 12.- Error 404 controlado.

Pues bien, hasta aquí los errores son controlados, solo que ya implementado esto en un sistema, no sería recomendable mostrar al usuario los detalles del error sino solamente un breve mensaje.

Controlando los errores que podrían producirse, 5.0 out of 5 based on 2 ratings