

MVC FRAMEWORK - EXCEPTION HANDLING

http://www.tutorialspoint.com/mvc_framework/mvc_framework_exception_handling.htm


Copyright © tutorialspoint.com

In ASP.NET, error handling is done using the standard try catch approach or using application events. ASP.NET MVC comes with built-in support for exception handling using a feature known as exception filters. We are going to learn two approaches here: one with overriding the onException method and another by defining the HandleError filters.

Override OnException method

This approach is used when we want to handle all the exceptions across the Action methods at the controller level.

To understand this approach, create an MVC application *follow the steps covered in previous chapters*. Now add a new Controller class and add the following code which overrides the onException method and explicitly throws an error in our Action method:



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

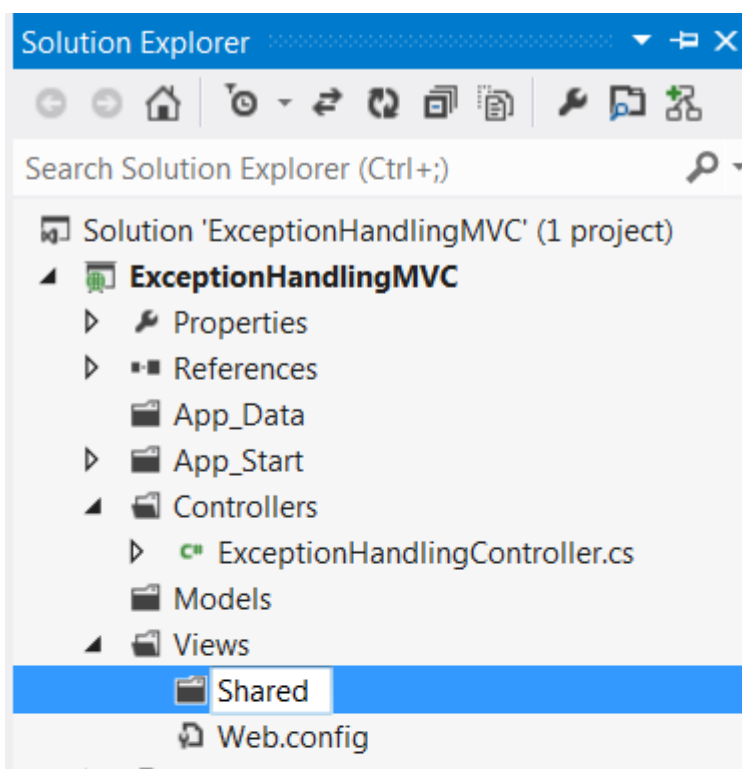
namespace ExceptionHandlingMVC.Controllers
{
    public class ExceptionHandlingController : Controller
    {
        protected override void OnException(ExceptionContext filterContext)
        {
            var e = filterContext.Exception;

            filterContext.ExceptionHandled = true;

            filterContext.Result = new ViewResult
            {
                ViewName = "Error"
            };
        }

        public ActionResult TestMethod()
        {
            throw new Exception("Test Exception");
            return View();
        }
    }
}
```

Now let us create a common View named Error which will be shown to the user when any exception happens in the application. Inside the Views folder, create a new folder called Shared and add a new View named Error.



```
Global.asax
packages.config
Web.config
```

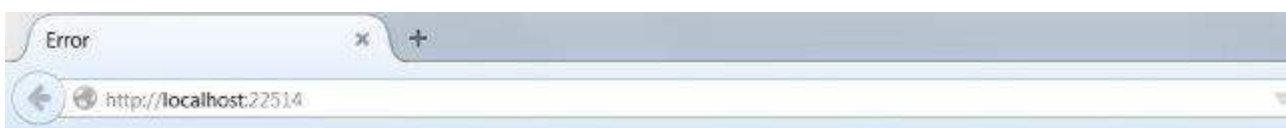
Copy the following code inside the newly created Error.cshtml:

```
RouteConfig.cs
TestMethod.cshtml
Index.cshtml
Exceptionl

@model dynamic
@{
    Layout = null;
}

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width"/>
    <title>Error</title>
</head>
<body>
    <h2>
        Sorry, an error occurred while processing your request.
    </h2>
    <ul>
        <li>
            <b> Exception:</b>@Model.Message
        </li>
        <li>
            <b> StackTrace:</b>>@Model.StackTrace
        </li>
    </ul>
</body>
</html>
```

If you try to run the application now, it will give the following result. The above code renders the Error View when any exception occurs in any of the action methods within this controller.



Sorry, an error occurred while processing your request.

The advantage of this approach is that multiple actions within the same controller can share this error handling logic. However, the disadvantage is that we cannot use the same error handling logic across multiple controllers.

HandleError Attribute

The HandleError Attribute is one of the action filters that we studied in Filters and Action Filters chapter. The HandleErrorAttribute is the default implementation of IExceptionHandler. This filter handles all the exceptions raised by controller actions, filters and views.

To use this feature, first of all turn on the customErrors section in web.config. Open the web.config and place the following code inside system.web and set its value as On.

```
<customErrors mode="On"/>
```

We already have the Error View created inside the Shared folder under Views. This time change the code of this View file to the following to strongly-type it with the HandleErrorInfo model which is present under System.Web.Mvc:

```
@model System.Web.Mvc.HandleErrorInfo
@{
    Layout = null;
}

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>Error</title>
</head>
<body>
<h2>
    Sorry, an error occurred while processing your request.
</h2>
<h2>Exception details</h2>
<p>
    Controller: @Model.ControllerName <br>
    Action: @Model.ActionName
    Exception: @Model.Exception
</p>
</body>
</html>
```

Now place the following code in your controller file which specifies [HandleError] attribute at the Controller file.

```
using System;
using System.Data.Common;
using System.Web.Mvc;

namespace ExceptionHandlingMVC.Controllers
{
    [HandleError]
    public class ExceptionHandlingController : Controller
    {
        public ActionResult TestMethod()
        {
            throw new Exception("Test Exception");
            return View();
        }
    }
}
```

```
}
```

If you try to run the application now, you will get an error similar to shown below:



As you can see, this time the error contains more information about the Controller and Action related details. In this manner, the HandleError can be used at any level and across controllers to handle such errors.

Loading [MathJax]/jax/output/HTML-CSS/jax.js