Exception Handling

Error :

1. Syntax Error (These can be corrected)
2. Logical Error (These can be corrected)
3. Run Time Error(Exception) (These can not be corrected, they can be handled only)

Enter Name : Ajay 899

Enter Age 23

Exceptions can be handled at different levels

1. Function Level
2. Controller Level
3. Application Level

Function Level > Try / catch block

[HttpPost]

public ActionResult AddNumbers1(string txtNo1, string txtNo2)

{

try

{

int n1 = int.Parse(txtNo1);

int n2 = int.Parse(txtNo2);

int result = n1 / n2;

ViewBag.result = result;

return View();

}

catch(Exception ex)

{

ViewBag.result = ex.Message.ToString();

return View();

}}

catch(Exception ex)

{

//ViewBag.result = ex.Message.ToString();

//return View();

return View("~/Views/Shared/Error.cshtml");

}

Limitation > We have to use it in all action methods

It’s a traditional DotNet way to handle exceptions

We want not to use try-catch block so many times, we can handle exceptions at controller level or at application level

--------------------------------------------------------------------

**Controller Level : Only Add OnException() method once per controller**

OnException() > Event will be fired whenever any exception occurs in any action method within a controller

protected override void OnException(ExceptionContext filterContext)

{

Exception ex = filterContext.Exception;

filterContext.ExceptionHandled = true;

var model = new HandleErrorInfo(filterContext.Exception, "Controller", "Action");

var Result = this.View("Error", new HandleErrorInfo(ex,

filterContext.RouteData.Values["controller"].ToString(),

filterContext.RouteData.Values["action"].ToString()));

filterContext.Result = Result;

}

Error View

@model System.Web.Mvc.HandleErrorInfo

<h1 class="text-danger">Error.@Model.Exception.Message</h1>

<h2 class="text-danger">An error occurred while processing your request.</h2>

@Html.ActionLink("Back", @Model.ActionName, @Model.ControllerName)

------------------------------------------------------------------

**Handle Exceptions at Application Level**

1. **Using “HandleError” Attribute**

Add <customErrors mode="On"></customErrors> in Web.config file which is at t he route level of your application

Step 1:

<system.web>

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

Step 2:

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Start()

{

AreaRegistration.RegisterAllAreas();

**FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);**

RouteConfig.RegisterRoutes(RouteTable.Routes);

BundleConfig.RegisterBundles(BundleTable.Bundles);

}

}

Step 3:

public class FilterConfig

{

public static void RegisterGlobalFilters(GlobalFilterCollection filters)

{

filters.Add(new HandleErrorAttribute());

}

}

Step 4: Add [HandleError()] in all controllers

[HandleError()]

public class MoviesController : Controller

{

}

But what if we have 50 controllers, then is it okay to add [HandleError()] in all controllers?

**Handle Exceptions at Global Level**

For this Application\_Error() Event Handler in global.asax file which will get fired automatically whenever some exception occurs anywhere in your application

Application\_Error() Event Handler

protected void Application\_Error(object sender, EventArgs e)

{

Exception exception = Server.GetLastError();

Server.ClearError();

Response.Redirect("/Home/Error");

}

After that we can comment filters.Add(new HandleErrorAttribute());

public static void RegisterGlobalFilters(GlobalFilterCollection filters)

{

// filters.Add(new HandleErrorAttribute());

}

-------------------------------------------------------------

HandleError handles only 500 server errors.