ASP.NET Core - Exception Handling

Exception handling is one of the most important features of any application. Fortunately, ASP.NET Core includes a middleware that makes exception handling easy. In this chapter, we will learn about exception handling in ASP.NET Core application.

By default, ASP.NET Core returns a simple status code for any exception that occurs in an application. Consider the following example of Configure method which throws an error.

public class Startup

{

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

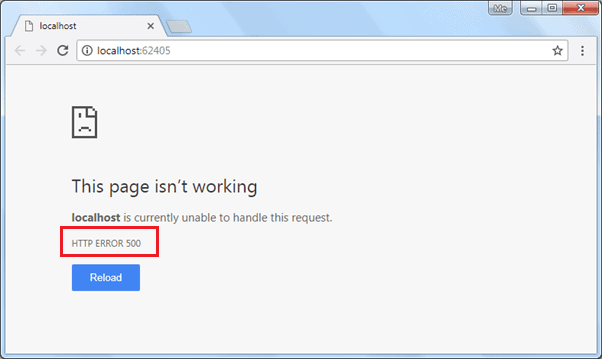
{

app.Run(context => { throw new Exception("error"); });

}

}

The above code will display the following result.

[](https://www.tutorialsteacher.com/Content/images/core/unhandled-exception-page.png)

Install Microsoft.AspNetCore.Diagnostics Package

To handle exceptions and display user friendly messages, we need to install Microsoft.AspNetCore.Diagnostics NuGet package and add middleware in the Configure() method. If you are using Visual Studio templates to create ASP.NET Core application then this package might be already installed. If not then you can add Microsoft.AspNetCore.Diagnostics package via NuGet manager.

The Microsoft.AspNetCore.Diagnostics package includes following extension methods to handle exceptions in different scenario:

1. UseDeveloperExceptionPage
2. UseExceptionHandler

UseDeveloperExceptionPage

The UseDeveloperExceptionPage extension method adds middleware into the request pipeline which displays developer friendly exception detail page. This helps developers in tracing errors that occur during development phase.

As this middleware displays sensitive information, it is advisable to add it only in development environment.

public class Startup

{

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment() || env.IsStaging())

{

app.UseDeveloperExceptionPage();

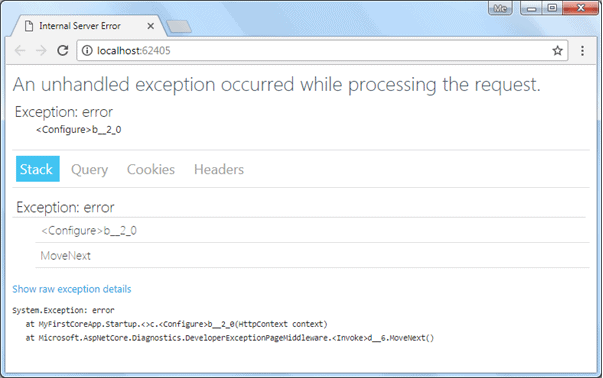
}

app.Run(context => { throw new Exception("error"); });

}

}

The above code will display the following result.

[](https://www.tutorialsteacher.com/Content/images/core/exception.png)Exception Handling

As you can see above, the developer exception page includes 4 tabs: Stack, Query, Cookies, and Headers. Stack tab displays information of stack trace, which indicates where exactly an error occurred. Query tab displays information about query string. Cookies tab displays information about cookies set by the request and Headers tab displays information about headers.

ADVERTISEMENT

UseExceptionHandler

In MVC Core application, we might want some other controller to handle all exceptions and display custom user friendly error messages. The UseExceptionHandler extension method allows us to configure custom error handling route. This is useful when an application runs under production environment.

Example: Exception Handler in MVC

 Copy

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment() || env.IsStaging())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

}

//code removed for clarity

}

In the above example, the UseExceptionHandler("/Home/Error") sets the error handler path. If an error occurred in the MVC application then it will redirect the request to /home/error, which will execute the Error action method of HomeController.

Create a simple Error action method in HomeController class as shown below.

HomeController:

 Copy

public class HomeController : Controller

{

public HomeController()

{

}

public IActionResult Error()

{

return View();

}

// other code removed for the clarity

}

The following is the content of Error.cshtml.

Error.cshtml

 Copy

@{

ViewData["Title"] = "Error";

}

<h1 class="text-danger">Error.</h1>

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

<h3>Development Mode</h3>

<p>

Swapping to <strong>Development</strong> environment will display more detailed information about the error that occurred.

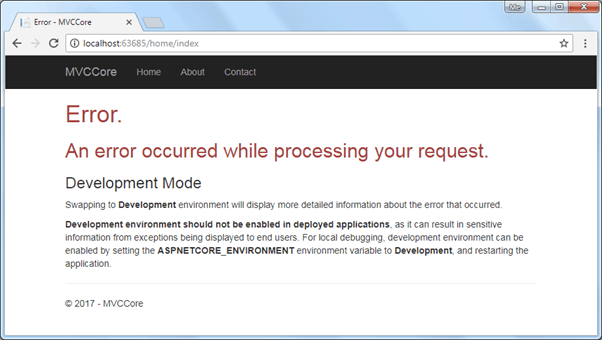
</p>

<p>

<strong>Development environment should not be enabled in deployed applications</strong>, as it can result in sensitive information from exceptions being displayed to end users. For local debugging, development environment can be enabled by setting the <strong>ASPNETCORE\_ENVIRONMENT</strong> environment variable to <strong>Development</strong>, and restarting the application.

</p>

Now, when an error occurs, it displays the page shown below.

[](https://www.tutorialsteacher.com/Content/images/core/custom-exception-page.png)Exception Handling

Thus, we can configure middleware to handle exceptions in ASP.NET Core application.

 Note:

Visual Studio automatically creates Error.cshtml under Home folder when you create ASP.NET Core project with MVC template.