Proper Error Handling

Overview

Task :

1. Learn how to configure customErrors ASP.NET Core Razor.
2. Learn how to handle program exception effectively and gracefully.

Objectives

- Setup Customed 404 error page in ASP.NET Core.
- Implement Try .. Catch codes for possible runtime error.

Examining the Three Types of Error Pages

Exceptions in .NET represent an error condition in an executing program. Error conditions can come about because of a large number of causes, each represented by its own exception type. Most exceptions arise from logical errors in code, such as an attempt to work with an object that has not been instantiated (NullRefrenceException), or to divide by zero (DivideByZeroException). Other types of exceptions result from technical issues, which may or may not be temporary. Examples of such issues might include a database or mail server being unavailable, or insufficient file system permissions.

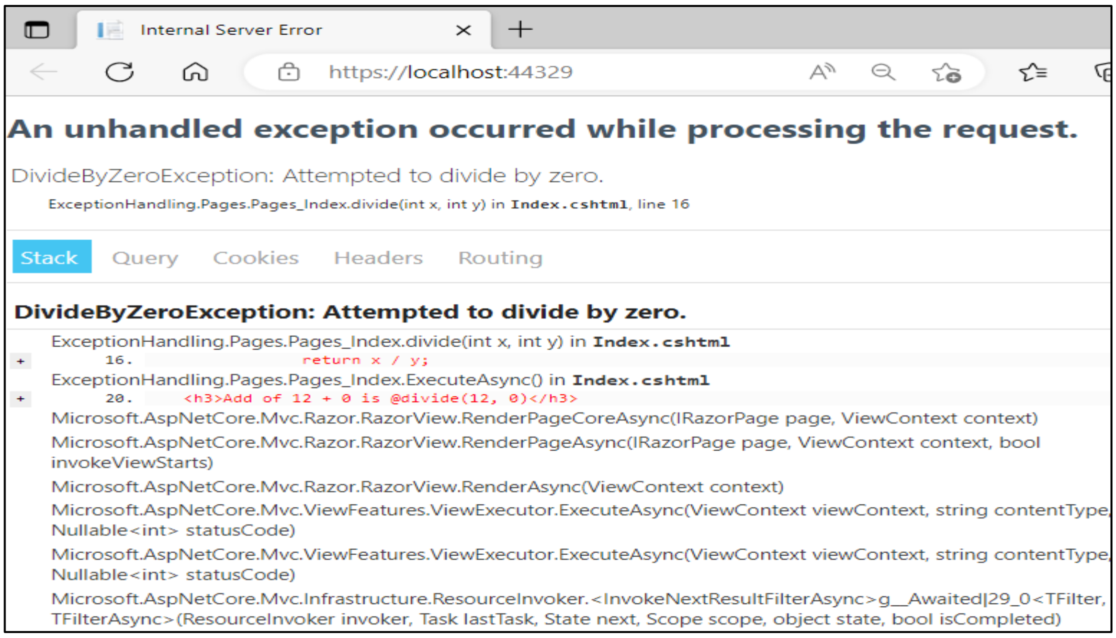
The recommendation is that you should try to minimise the impact of exceptions (otherwise known as handling them) by coding defensively rather than allowing them to crash your application. Options include wrapping code that might raise exceptions in try-catch blocks, and validating user input instead of assuming that it conforms to expectations. Despite your best efforts, however, chances are that even in a moderately complex application, there will be something that you overlooked that will go wrong.

When an unhandled exception arises in an ASP.NET Core application, the following are possible error pages:

- The Exception Details or Runtime error page,
- HTTP error page

1. The Exception Details or Runtime error page,

   The Exception provides details about the exception - the type, the message, and the stack trace.



An unhandled exception occurred while processing the request.

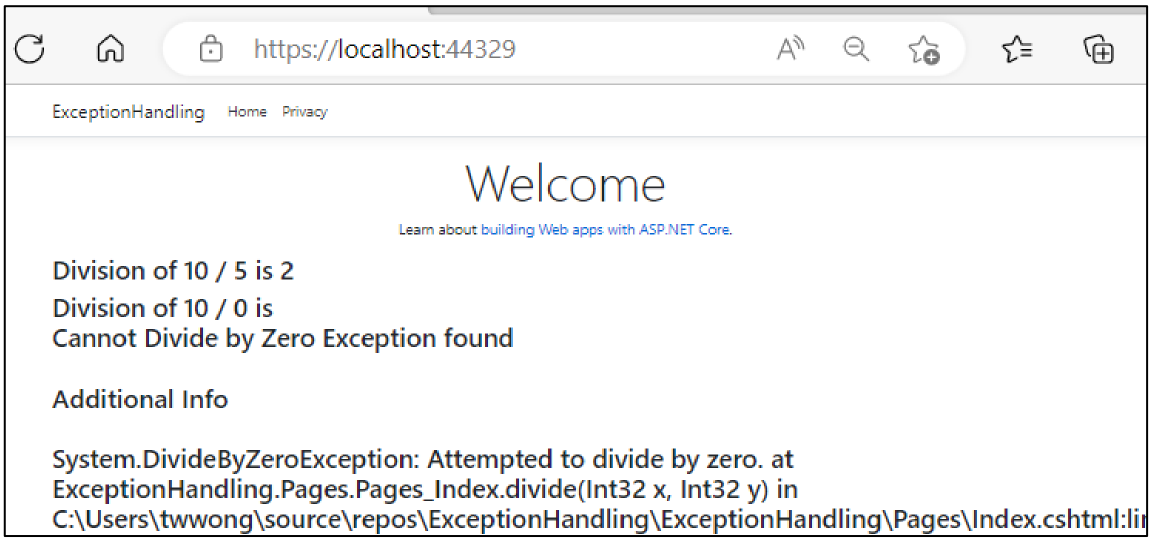
DivideByZeroException: Attempted to divide by zero.

ExceptionHandling.Pages.Pages_Index.divide(int x, int y) in Index.cshtml, line 16

DivideByZeroException: Attempted to divide by zero.

```
ExceptionHandling.Pages.Pages_Index.divide(int x, int y) in Index.cshtml
+     16.                return x / y;
ExceptionHandling.Pages.Pages_Index.ExecuteAsync() in Index.cshtml
+     20.       <h3>Add of 12 + 0 is @divide(12, 0)</h3>
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageCoreAsync(IRazorPage page, ViewContext context)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageAsync(IRazorPage page, ViewContext context, bool invokeViewStarts)
Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderAsync(ViewContext context)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ViewContext viewContext, string contentType, Nullable<int> statusCode)
Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ViewContext viewContext, string contentType, Nullable<int> statusCode)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeNextResultFilterAsync>g__Awaited|29_0<TFilter, TFilterAsync>(ResourceInvoker invoker, Task lastTask, State next, Scope scope, object state, bool isCompleted)
```

Exception being handled



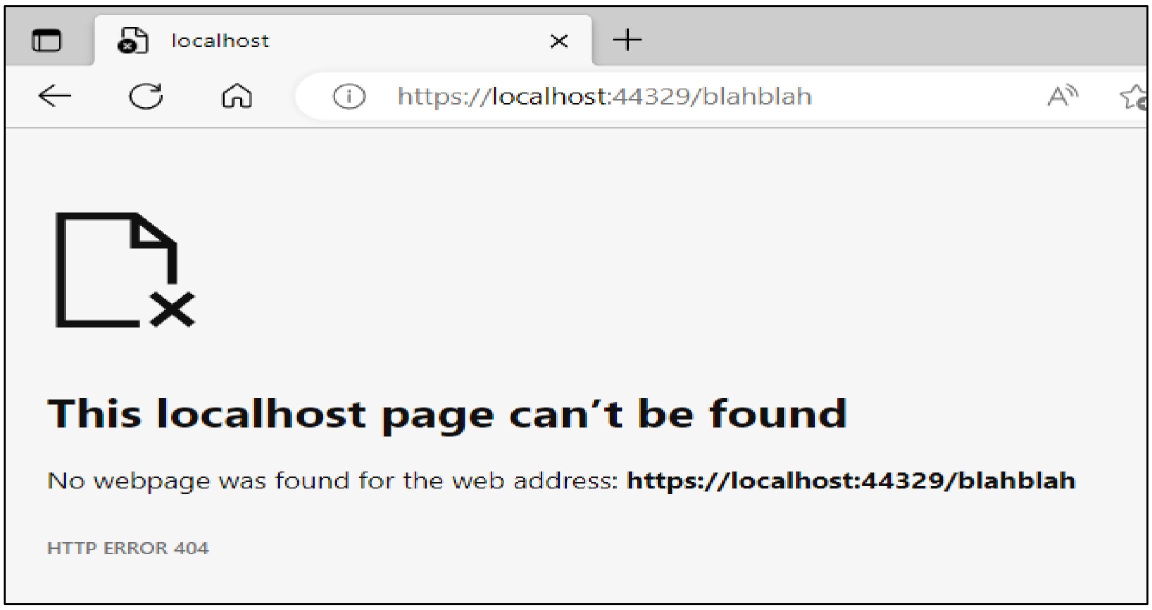
Welcome

Learn about building Web apps with ASP.NET Core.

```
Division of 10 / 5 is 2
Division of 10 / 0 is
Cannot Divide by Zero Exception found

Additional Info

System.DivideByZeroException: Attempted to divide by zero. at
ExceptionHandling.Pages.Pages_Index.divide(Int32 x, Int32 y) in
C:\Users\twwong\source\repos\ExceptionHandling\ExceptionHandling\Pages\Index.cshtml:li
```

2. HTTP error page

Default HTTP error page



localhost

https://localhost:44329/blahblah

This localhost page can't be found

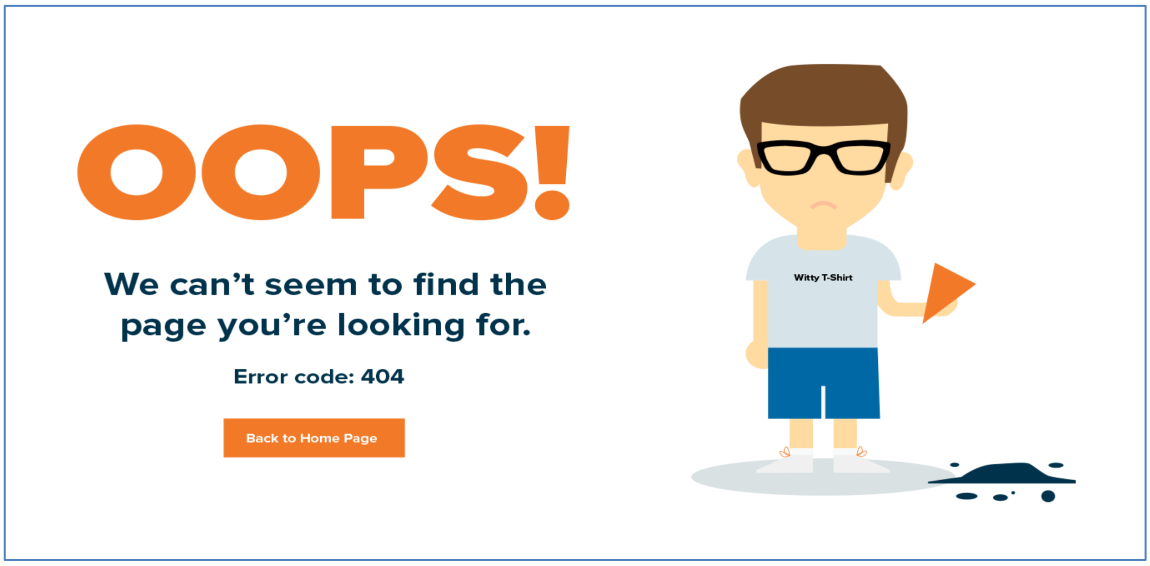
No webpage was found for the web address: **https://localhost:44329/blahblah**

HTTP ERROR 404

Custom HTTP error page (e.g 404)

The benefit of a custom error page is that you have complete control over the information that is displayed to the user along with the page's look and feel; the custom error page can use the same master page and styles as your other pages.

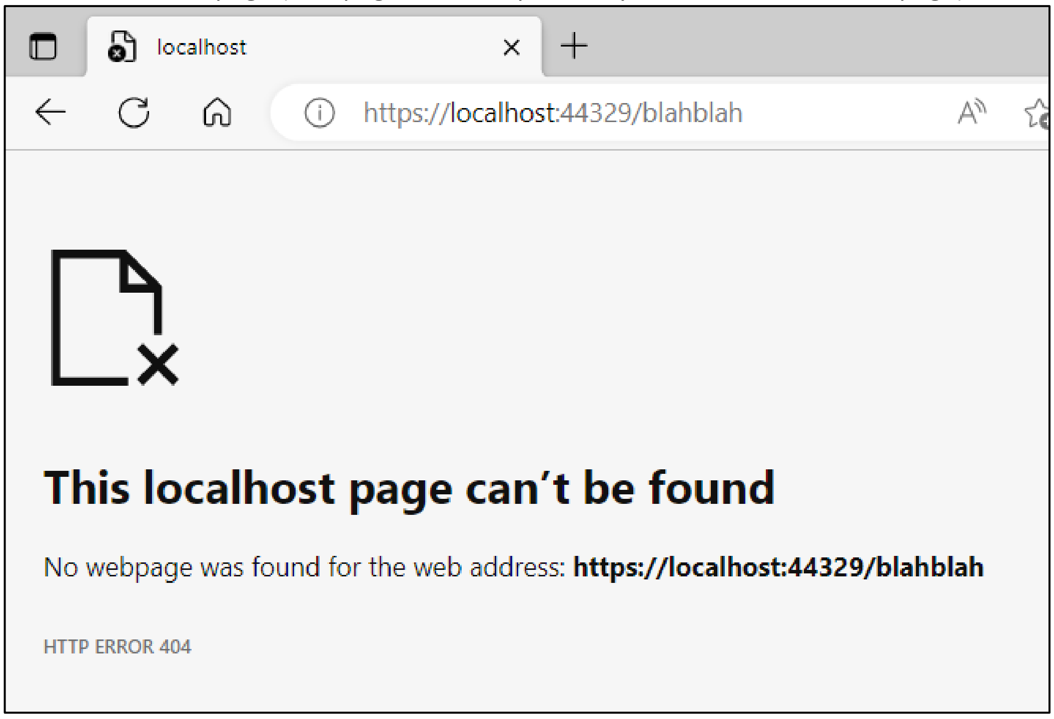
The figure below offers a sneak peak of this custom error page. As you can see, the look and feel of the error page is much more professional-looking than either of the Yellow Screens of Death shown above.



Task 1 : Creating Custom HTTP Error page

- In this section we will create a custom static 404 HTTP Error page to replace the default one.
- Add codes to program.cs

Default 404 error page (This page will be replaced by our custom 404 error page)

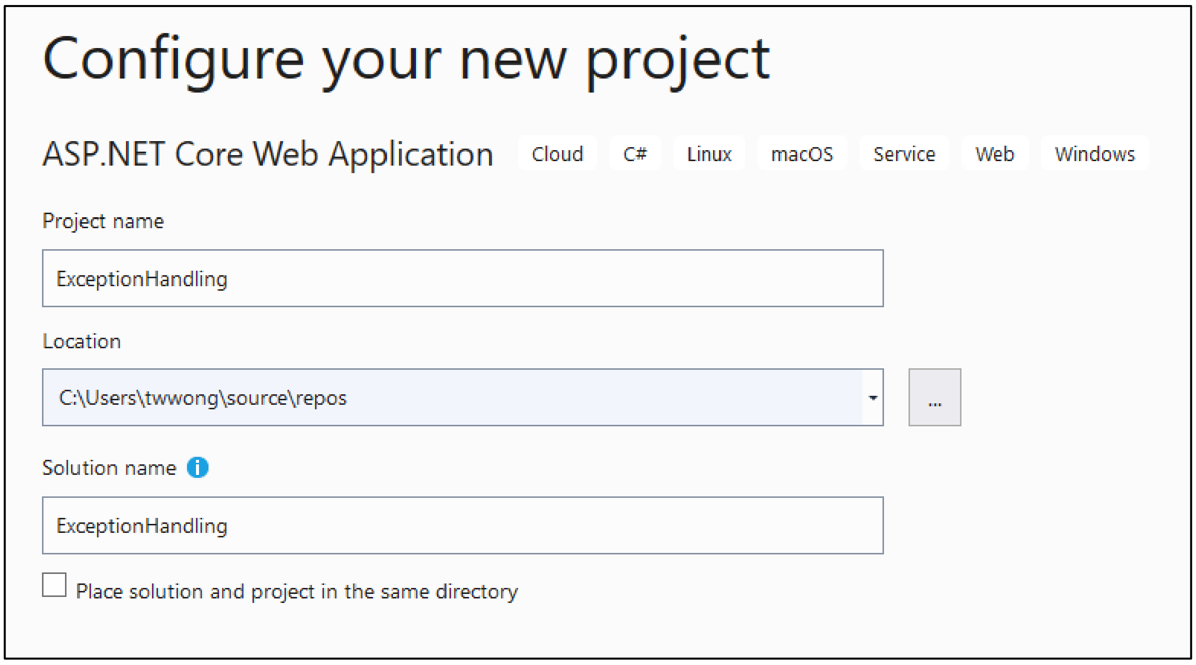


This localhost page can't be found

No webpage was found for the web address: **https://localhost:44329/blahblah**

HTTP ERROR 404

1. Create a new ASP.NET Core Web Application with the project name "**ExceptionHandling**"



Configure your new project

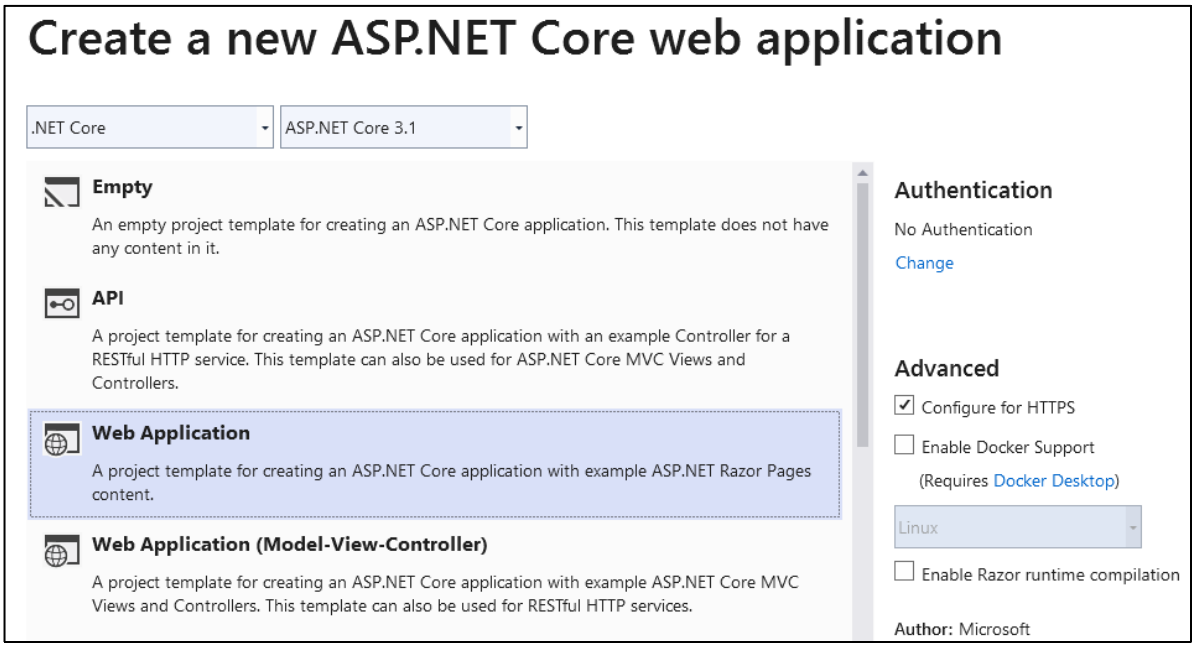
ASP.NET Core Web Application

Project name: ExceptionHandling

Location: C:\Users\twwong\source\repos

Solution name: ExceptionHandling

☐ Place solution and project in the same directory



Create a new ASP.NET Core web application

.NET Core | ASP.NET Core 3.1

Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API
A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

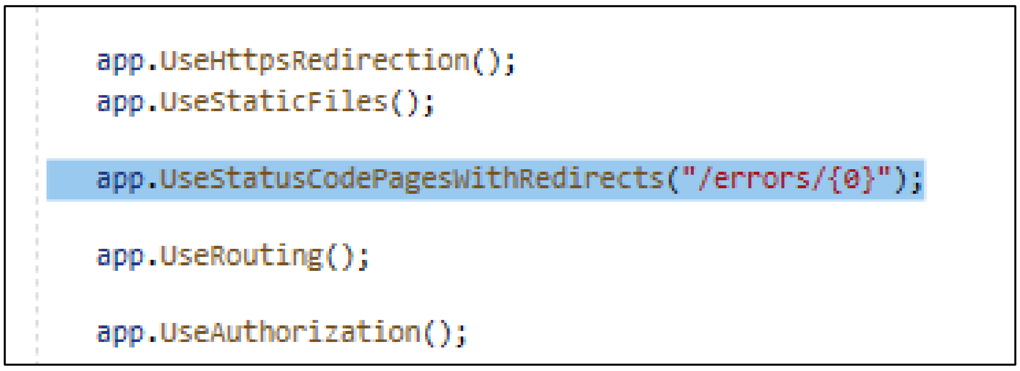
Authentication
No Authentication
Change

Advanced
☑ Configure for HTTPS
☐ Enable Docker Support
(Requires Docker Desktop)
Linux
☐ Enable Razor runtime compilation

Author: Microsoft

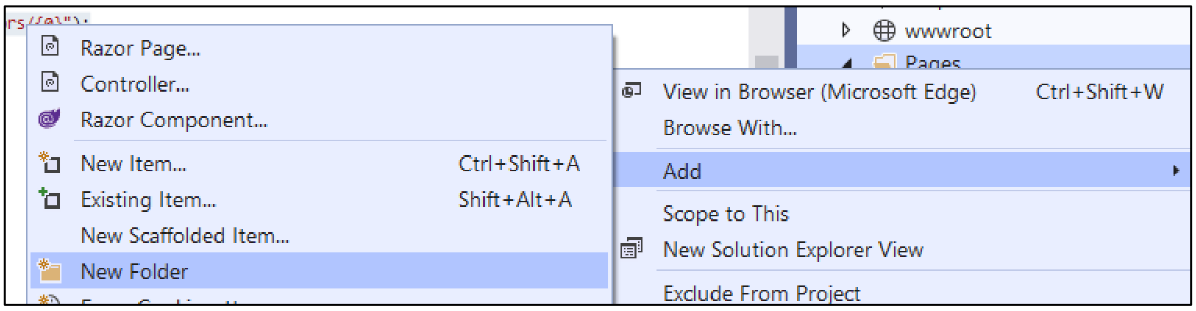
2. Add status “UseStatusCodePageWithRedirects(“/errors/{0});

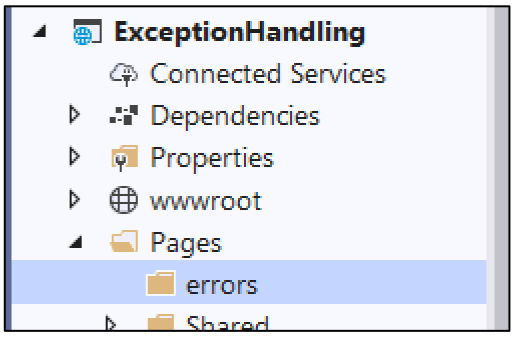
   The UseStatusCodePagesWithRedirects extension method: Sends a 302 - Found status code to the client. Redirects the client to the error handling endpoint provided in the URL template. The error handling endpoint typically displays error information and returns HTTP 200



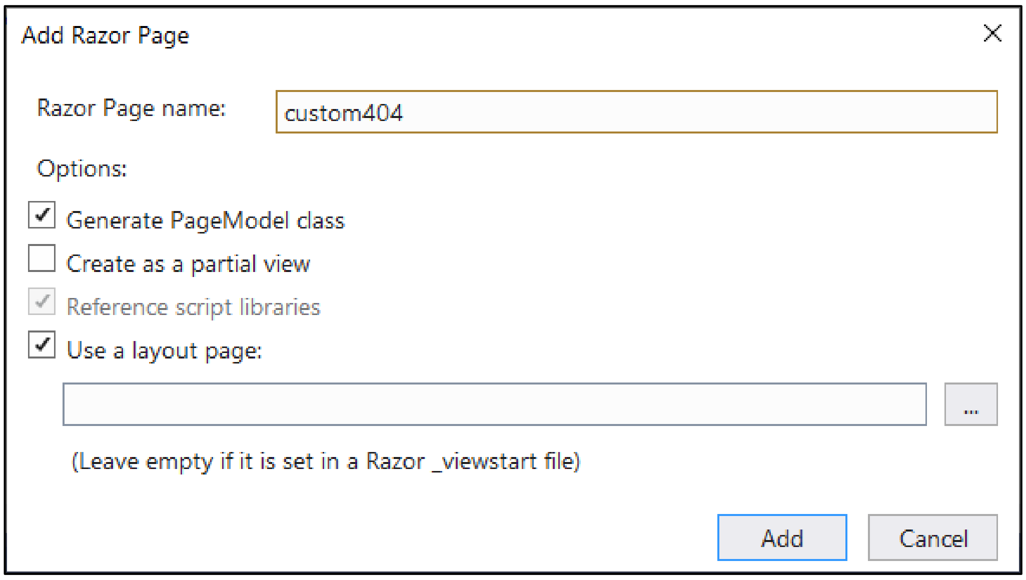
```
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseStatusCodePagesWithRedirects("/errors/{0}");

app.UseRouting();

app.UseAuthorization();
```

3. Create a new Folder “errors” in the Pages folder.





4. Add new Empty Razor page into the errors folder. Name the Razor page "404".

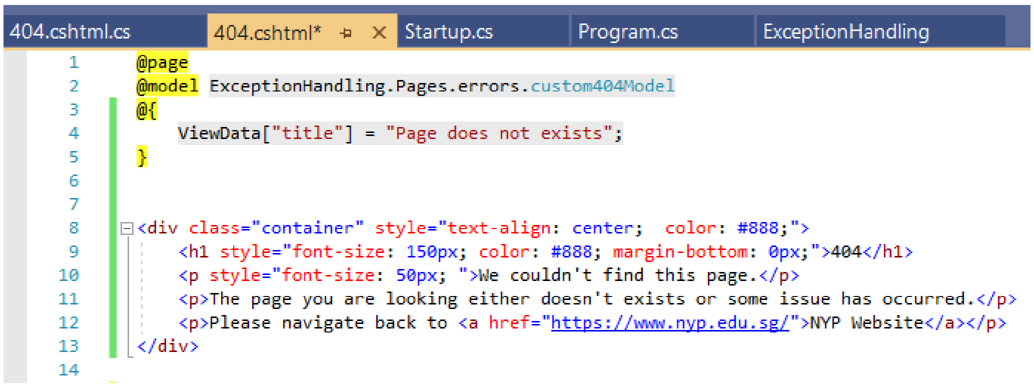


Razor Page name: custom404

Options:
- ☑ Generate PageModel class
- ☐ Create as a partial view
- ☑ Reference script libraries
- ☑ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Update custom404.cshtml with the following codes.



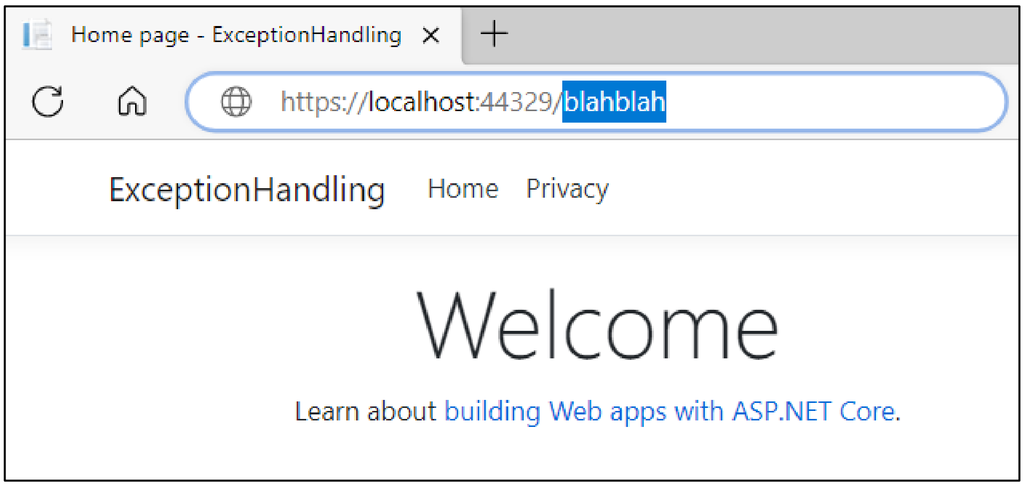
```
@page
@model ExceptionHandling.Pages.errors.custom404Model
@{
    ViewData["title"] = "Page does not exists";
}


<div class="container" style="text-align: center;  color: #888;">
    <h1 style="font-size: 150px; color: #888; margin-bottom: 0px;">404</h1>
    <p style="font-size: 50px; ">We couldn't find this page.</p>
    <p>The page you are looking either doesn't exists or some issue has occurred.</p>
    <p>Please navigate back to <a href="https://www.nyp.edu.sg/">NYP Website</a></p>
</div>
```

<u>Sample codes</u>

```
@{

  ViewData["title"] = "Page does not exists";

}

<div class="container" style="text-align: center;  color: #888;">

  <h1 style="font-size: 150px; color: #888; margin-bottom: 0px;">404</h1>

  <p style="font-size: 50px; ">We couldn't find this page.</p>

  <p>The page you are looking either doesn't exists or some issue has occurred.</p>

  <p>Please navigate back to <a href="https://www.nyp.edu.sg/">NYP Website</a></p>

</div>
```

5. Test run your app.

   Run your app and in the url just enter any invalid page.
   e.g : https://localhost:44329/blahblah



Home page - ExceptionHandling

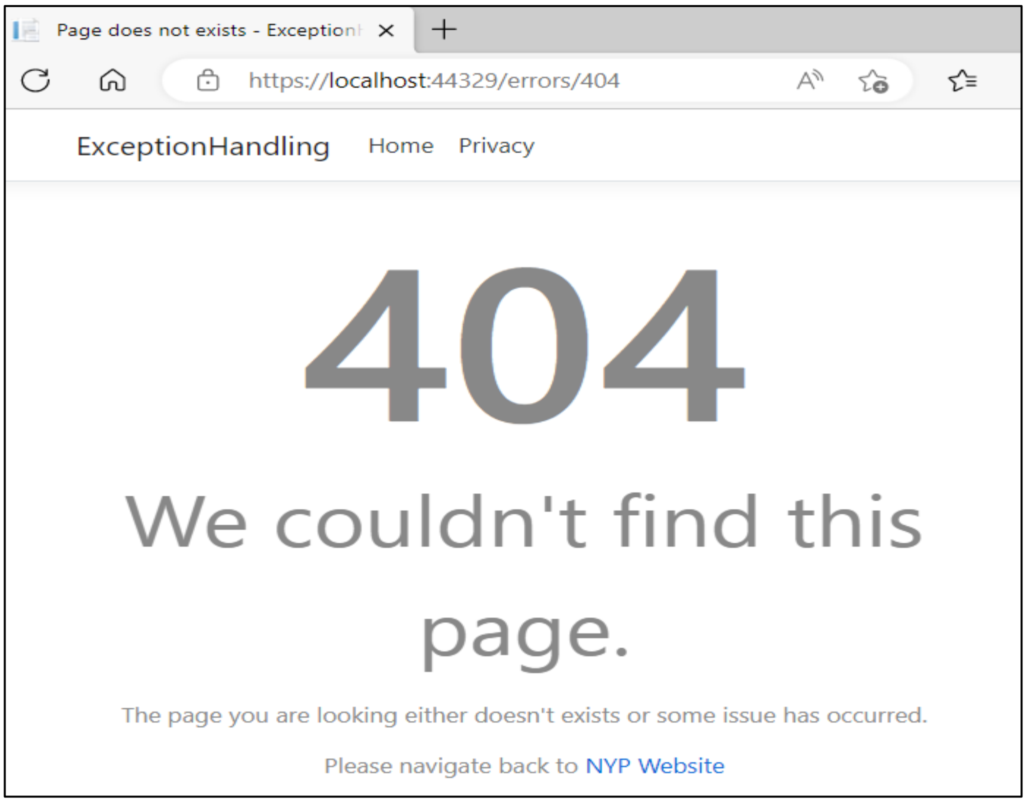
https://localhost:44329/blahblah

ExceptionHandling Home Privacy

Welcome

Learn about building Web apps with ASP.NET Core.

   If you are not able to get the page below to work, rename your custom404.cshtml to 404.cshml instead.



Page does not exists - Exception

https://localhost:44329/errors/404

ExceptionHandling Home Privacy

404

We couldn't find this page.

The page you are looking either doesn't exists or some issue has occurred.

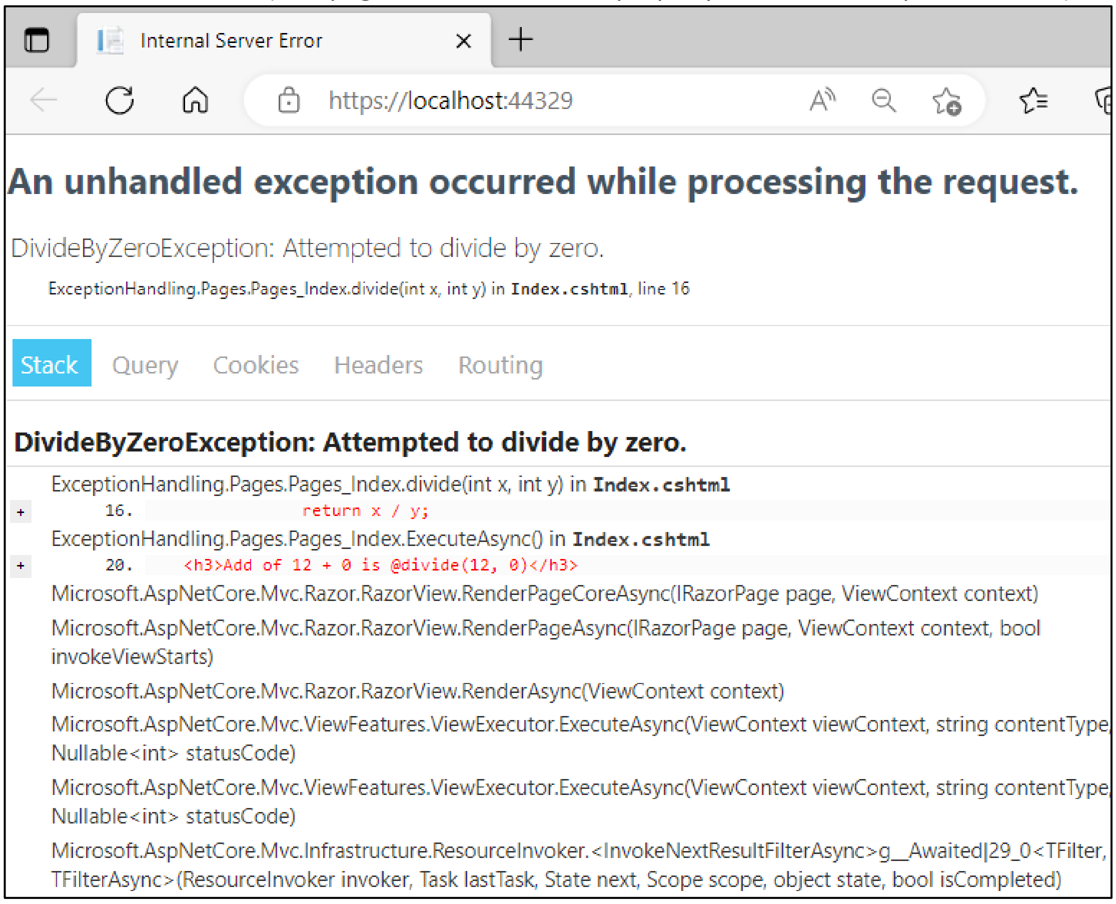
Please navigate back to NYP Website

Note : See the Appendix A for a list of HTTP error codes.

Task 2 : Handling runtime error

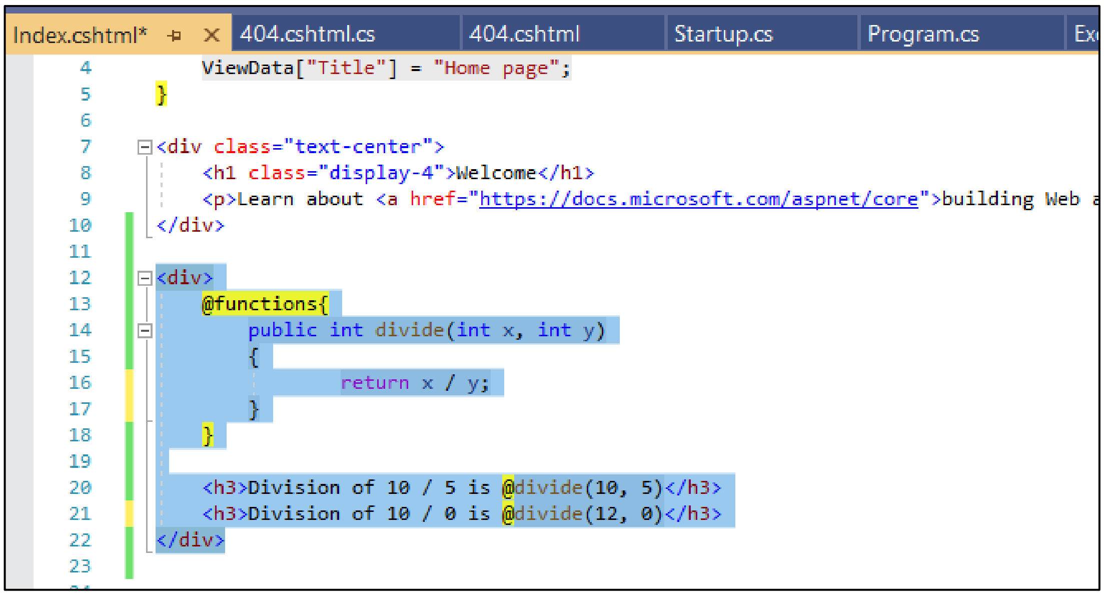
- In this section we will implement Try..catch code to handle any possible or foreseeable runtime errors in the application.

Default runtime error (This page will have it's error properly handled with updated codes)



```
An unhandled exception occurred while processing the request.

DivideByZeroException: Attempted to divide by zero.
    ExceptionHandling.Pages.Pages_Index.divide(int x, int y) in Index.cshtml, line 16

DivideByZeroException: Attempted to divide by zero.
    ExceptionHandling.Pages.Pages_Index.divide(int x, int y) in Index.cshtml
+       16.             return x / y;
    ExceptionHandling.Pages.Pages_Index.ExecuteAsync() in Index.cshtml
+       20.     <h3>Add of 12 + 0 is @divide(12, 0)</h3>
    Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageCoreAsync(IRazorPage page, ViewContext context)
    Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderPageAsync(IRazorPage page, ViewContext context, bool
    invokeViewStarts)
    Microsoft.AspNetCore.Mvc.Razor.RazorView.RenderAsync(ViewContext context)
    Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ViewContext viewContext, string contentType
    Nullable<int> statusCode)
    Microsoft.AspNetCore.Mvc.ViewFeatures.ViewExecutor.ExecuteAsync(ViewContext viewContext, string contentType
    Nullable<int> statusCode)
    Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeNextResultFilterAsync>g__Awaited|29_0<TFilter,
    TFilterAsync>(ResourceInvoker invoker, Task lastTask, State next, Scope scope, object state, bool isCompleted)
```

1. In the same ASP.NET Core Web Application “**ExceptionHandling**”, create runtime error in the index.cshml page.



Index.cshtml* | 404.cshtml.cs | 404.cshtml | Startup.cs | Program.cs | Ex

```
        ViewData["Title"] = "Home page";
    }

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web a
</div>

<div>
    @functions{
        public int divide(int x, int y)
        {
            return x / y;
        }
    }

    <h3>Division of 10 / 5 is @divide(10, 5)</h3>
    <h3>Division of 10 / 0 is @divide(12, 0)</h3>
</div>
```

Add the divide function and the HTML codes into index.cshtml. The divide function will return an error due to divide by zero error. Test run your app.

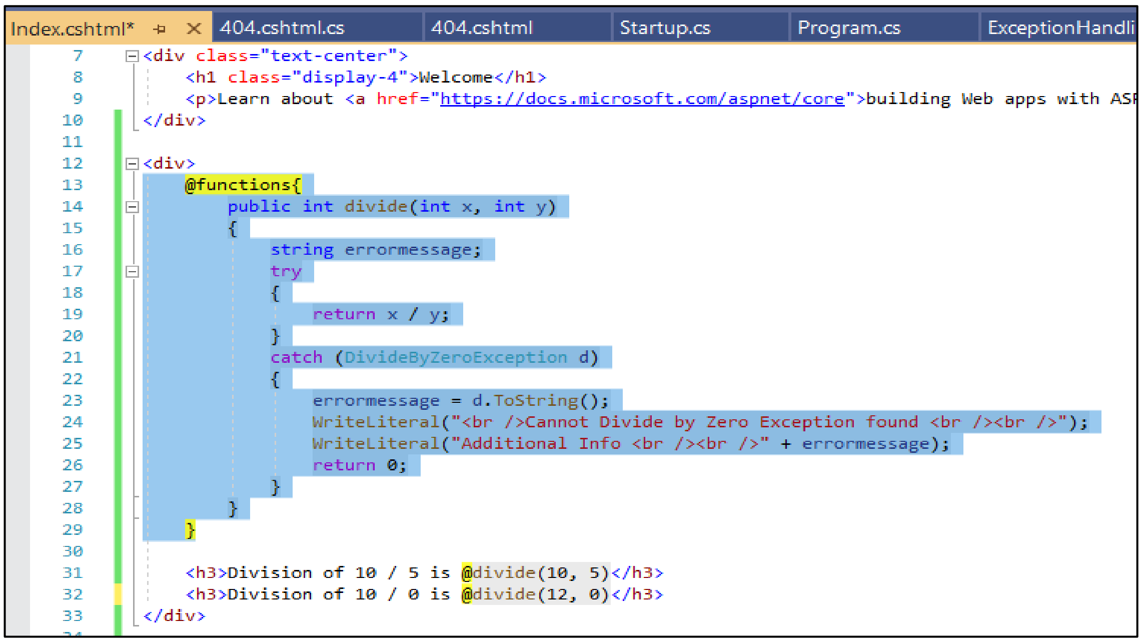
Sample

```
<div>

  @functions{
      public int divide(int x, int y)
      {
          return x / y;
      }
   }

  <h3>Division of 10 / 5 is @divide(10, 5)</h3>
  <h3>Division of 10 / 0 is @divide(12, 0)</h3>

</div>
```

2. Supressing the exception messages by implementing Try .. Catch codes.

   Using Try .. Catch block will prevent application crashes. It will help to handle your page error gracefully.

   Update the codes in index.cshml as show below :

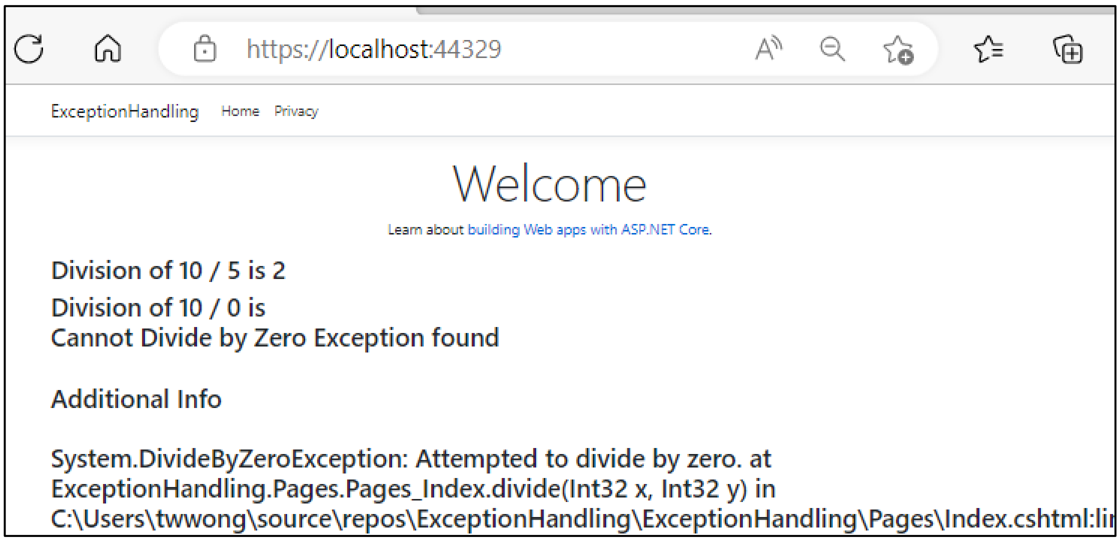


```
Index.cshtml*   404.cshtml.cs   404.cshtml   Startup.cs   Program.cs   ExceptionHandli
<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP
</div>

<div>
    @functions{
        public int divide(int x, int y)
        {
            string errormessage;
            try
            {
                return x / y;
            }
            catch (DivideByZeroException d)
            {
                errormessage = d.ToString();
                WriteLiteral("<br />Cannot Divide by Zero Exception found <br /><br />");
                WriteLiteral("Additional Info <br /><br />" + errormessage);
                return 0;
            }
        }
    }

    <h3>Division of 10 / 5 is @divide(10, 5)</h3>
    <h3>Division of 10 / 0 is @divide(12, 0)</h3>
</div>
```

Sample codes

```
@functions{

  public int divide(int x, int y)

  {
    string errormessage;
    try {
        return x / y;
    }
    catch (DivideByZeroException d)
    {
        errormessage = d.ToString();
        WriteLiteral("<br />Cannot Divide by Zero Exception found <br /><br />");
        WriteLiteral("Additional Info <br /><br />" + errormessage);
        return 0;

    }
  }
}

<h3>Division of 10 / 5 is @divide(10, 5)</h3>
<h3>Division of 10 / 0 is @divide(12, 0)</h3>
```

Test run your app again.



Welcome

Learn about building Web apps with ASP.NET Core.

Division of 10 / 5 is 2

Division of 10 / 0 is
Cannot Divide by Zero Exception found

Additional Info

```
System.DivideByZeroException: Attempted to divide by zero. at
ExceptionHandling.Pages.Pages_Index.divide(Int32 x, Int32 y) in
C:\Users\twwong\source\repos\ExceptionHandling\ExceptionHandling\Pages\Index.cshtml:li
```

APPENDIX A

Http Error page codes

There are different Http codes that your web application could return. Some errors are more often than others. You probably don't need to cover all cases. It is ok to place custom error pages for the most common error codes and place default error page for the rest.

400 Bad Request

Request is not recognized by server because of errors in syntax. Request should be changed with corrected syntax.

401 Not Authorized

This error happens when request doesn't contain authentication or authorization is refused because of bad credentials.

402 Payment Required

Not in use, it is just reserved for the future

403 Forbidden

Server refused to execute request, although it is in correct format. Server may or may not provide information why request is refused.

404 Not Found

Server can not find resource requested in URI. This is very common error, you should add custom error page for this code.

405 Method Not Allowed

There are different Http request methods, like POST, GET, HEAD, PUT, DELETE etc. that could be used by client. Web server can be configured to allow or disallow specific method. For example, if web site has static pages POST method could be disabled. There are many theoretical options but in reality this error usually occurs if POST method is not allowed.

406 Not Acceptable

Web client (browser or robot) could try to receive some data from web server. If that data are not acceptable web server will return this error. Error will not happen (or very rarely) when web browser request the page.

407 Proxy Authentication Required

This error could occur if web client accesses to web server through a proxy. If proxy authentication is required you must first login to proxy server and then navigate to wanted page. Because of that this error is similar to error 401 Not Authorized, except that here is problem with proxy authentication.

408 Request Timeout

If connection from web client and server is not established in some time period, which is defined on web server, then web server will drop connection and send error 408 Request Timeout. The reason could be usually temporarily problem with Internet connection or even to short time interval on web server.

409 Conflict

This error rarely occurs on web server. It means that web request from client is in conflict with some server or application rule on web server.

410 Gone

This error means that web server can't find requested URL. But, as opposed to error 404 Not Found which says: "That page is not existing", 410 says something like: "The page was here but not anymore". Depending of configuration of web server, after some time period server will change error message to 404 Not Found.

411 Length Required

This error is rare when web client is browser. Web server expects Content-Length parameter included in web request.

412 Precondition Failed

This is also rare error, especially if client is web browser. Error occurs if Precondition parameter is not valid for web server.

413 Request Entity Too Large

This error occurs when web request is to large. This is also very rare error, especially when request is sent by web browser.

414 Request URI Too Long

Similar like error 413, error occurs if URL in the web request is too long. This limit is usually 2048 to 4096 characters. If requested URL is longer than server's limit then this error is returned. 2048 characters is pretty much, so this error occurs rarely. If your web application produces this error, then it is possible that is something wrong with your URLs, especially if you build it dynamically with ASP.NET server side code.

415 *Unsupported Media Type*

This error occurs rarely, especially if request is sent by web browser. It could be three different reasons for this error. It is possible that requested media type doesn't match media type specified in request, or because of incapability to handle current data for the resource, or it is not compatible with requested Http method.

416 *Requested Range Not Satisfied*

This is very rare error. Client request can contain Range parameter. This parameter represents expected size of resource requested. For example, if client asks for an image, and range is between 0 and 2000, then image should not be larger from 2000 bytes. If image is larger, this error is returned. However, web page hyperlinks usually don't specify any Range value so this error rarely occurs.

417 *Expectation Failed*

This is also rare error, especially if client is web browser. There is Expect parameter of request, if this Expect is not satisfied Expectation error is returned.

500 *Internal Server Error*

This is very common error; client doesn't know what the problem is. Server only tells that there is some problem on server side. But, on the server side are usually more information provided. If server hosts ASP.NET application, then this often means that there is an error in ASP.NET application. More details about error could be logged to EventLog, database or plain text files. To see how to get error details take a look at Application Level Error Handling In ASP.NET tutorial.

501 *Not Implemented*

This is rare error. It means that web server doesn't support Http method used in request. Common Http methods are POST, GET, HEAD, TRACE etc. If some other method is used and web server can't recognize it, this error will be returned.

502 *Bad Gateway*

This error occurs when server is working as gateway and need to proceed request to upstream web server. If upstream web server response is not correct, then first server will return this error. The reason for this error is often bad Internet connection some problem with firewall, or problem in communication between servers.

503 *Service unavailable*

This error means that server is temporally down, but that is planned, usually because a maintenance. Of course, it is not completely down because it can send 503 error :), but it is not working properly. Client should expect that system administrator is working on the server and server should be up again after problem is solved.

504 *Gateway Timeout*

Similar to error 502 Bad Gateway, there is problem somewhere between server and upstream web server. In this case, upstream web server takes too long to respond and first server returned this error. This could happen for example if your Internet connection is slow, or it is slow or overloaded communication in your local network.

505 *HTTP Version Not Supported*

Web server doesn't support Http version used by web client. This should be very rare error. It could happen eventually if web client tries to access to some very old web server, which doesn't support newer Http protocol version (before v. 1.x).