Exception Management Lab

Lab 1: Built-In Global Exception

Open **HomeController.cs** and add the code shown in bold below.

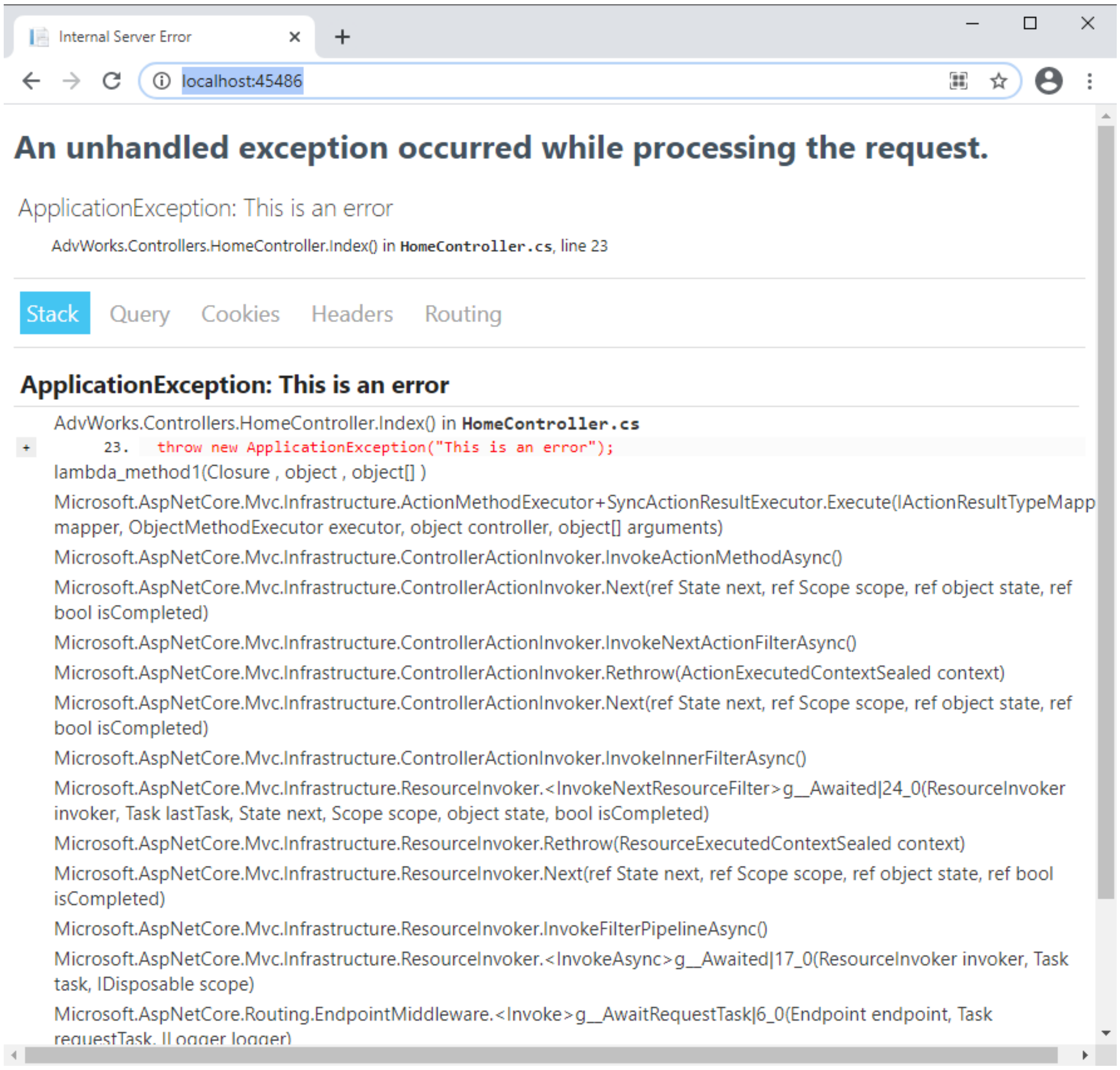
```
[HttpGet]
public IActionResult Index()
{
  throw new ApplicationException("This is an error");

  // REST OF THE CODE HERE

  return View();
}
```

Try it Out

Run the application

The debugger will most likely stop on the *throw* line

Simply click the **Continue** button in the Visual Studio Toolbar to display the error page



Internal Server Error

localhost:45486

An unhandled exception occurred while processing the request.

ApplicationException: This is an error

AdvWorks.Controllers.HomeController.Index() in HomeController.cs, line 23

ApplicationException: This is an error

```
AdvWorks.Controllers.HomeController.Index() in HomeController.cs
    23.   throw new ApplicationException("This is an error");
lambda_method1(Closure , object , object[] )
Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor+SyncActionResultExecutor.Execute(IActionResultTypeMapp
mapper, ObjectMethodExecutor executor, object controller, object[] arguments)
Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeActionMethodAsync()
Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(ref State next, ref Scope scope, ref object state, ref
bool isCompleted)
Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeNextActionFilterAsync()
Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Rethrow(ActionExecutedContextSealed context)
Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(ref State next, ref Scope scope, ref object state, ref
bool isCompleted)
Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeInnerFilterAsync()
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeNextResourceFilter>g__Awaited|24_0(ResourceInvoker
invoker, Task lastTask, State next, Scope scope, object state, bool isCompleted)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.Rethrow(ResourceExecutedContextSealed context)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.Next(ref State next, ref Scope scope, ref object state, ref bool
isCompleted)
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.InvokeFilterPipelineAsync()
Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Awaited|17_0(ResourceInvoker invoker, Task
task, IDisposable scope)
Microsoft.AspNetCore.Routing.EndpointMiddleware.<Invoke>g__AwaitRequestTask|6_0(Endpoint endpoint, Task
requestTask, ILogger logger)
```

Lab 2: Create Production Profile

Expand the **\Properties** folder

Open the Properties\launchSettings.json file

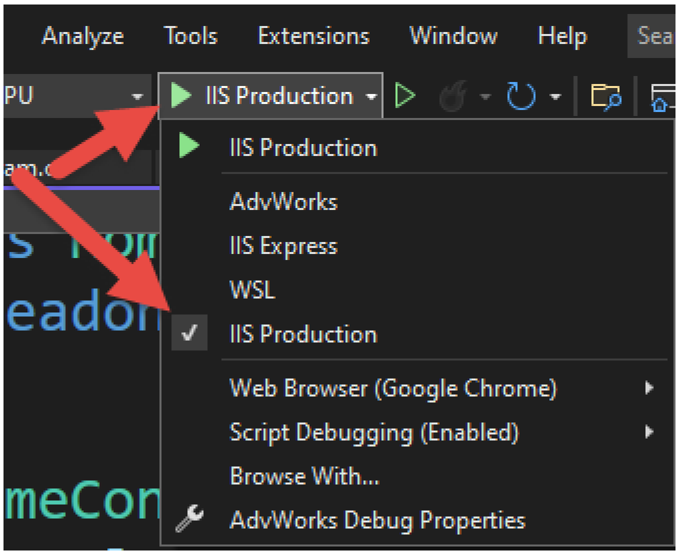
Locate the last JSON node called "IIS Express"

Copy the "IIS Express" profile to "IIS Production"

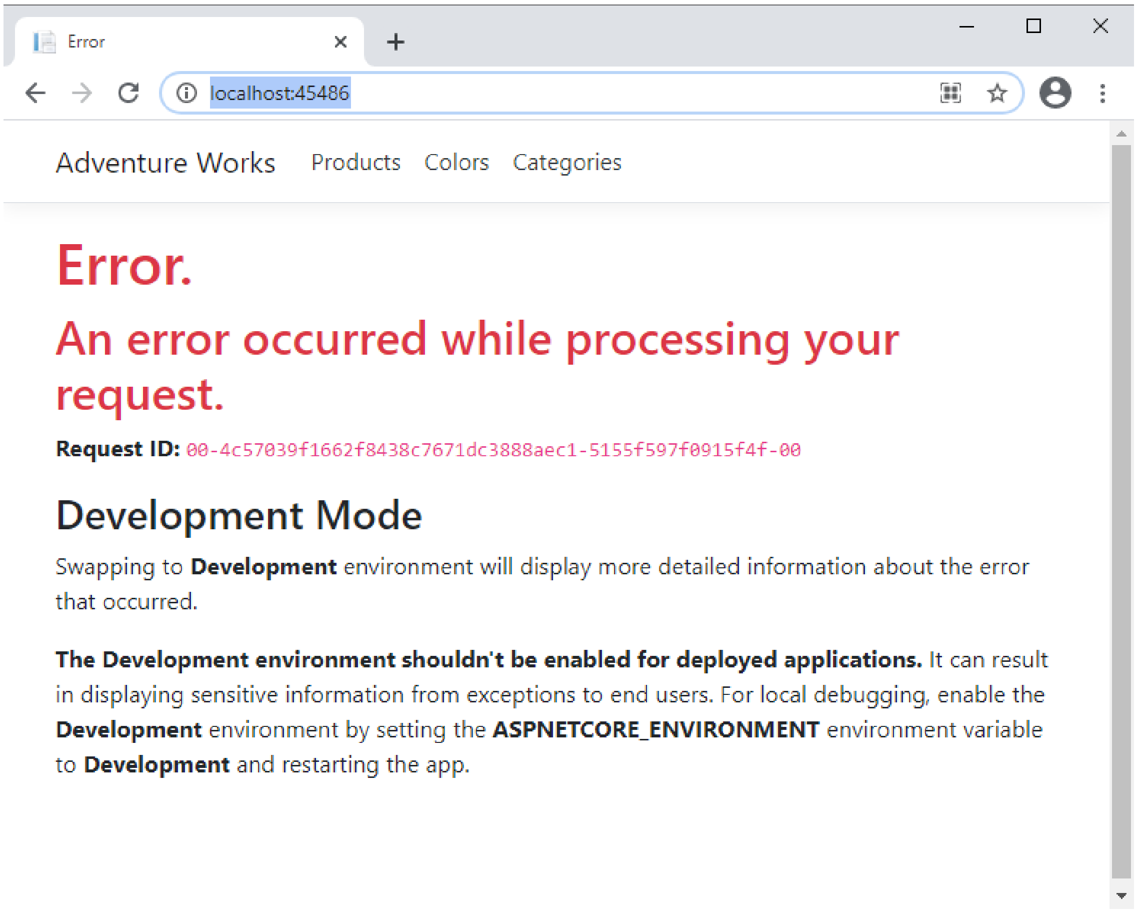
```
"IIS Production": {
  "commandName": "IISExpress",
  "launchBrowser": true,
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Production",
    "ASPNETCORE_HOSTINGSTARTUPASSEMBLIES":
"Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation"
  }
}
```

Try it Out

From the Run button on the Visual Studio Toolbar, change to the "IIS Production" profile as shown below.



Run the application to see the following page:



Error.

An error occurred while processing your request.

Request ID: 00-4c57039f1662f8438c7671dc3888aec1-5155f597f0915f4f-00

Development Mode

Swapping to **Development** environment will display more detailed information about the error that occurred.

The Development environment shouldn't be enabled for deployed applications. It can result in displaying sensitive information from exceptions to end users. For local debugging, enable the **Development** environment by setting the **ASPNETCORE_ENVIRONMENT** environment variable to **Development** and restarting the app.

Lab 3: Move Error Handling out of Home Controller

Open the **Program.cs** file and modify the following code:

```
if (!app.Environment.IsDevelopment()) {
  app.UseExceptionHandler("/Error/Error");
}
```

Right mouse-click on the **\Views** folder and create a new folder named **Error**

Move the **Error.cshtml** page from **\Shared** into **\Error**

Right mouse click on the **\Controllers** folder and add a new controller named **ErrorController.cs**.

Make the controller look like the following code:

```
using AdvWorks.Models;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;

namespace AdvWorks.Controllers {
  [AllowAnonymous]
  public class ErrorController : Controller {
    private readonly ILogger<HomeController> _logger;

    public ErrorController(ILogger<HomeController> logger) {
      _logger = logger;
    }

    [ResponseCache(Duration = 0, Location =
ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error() {
      return View(new ErrorViewModel { RequestId =
Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }
  }
}
```

Open the **HomeController.cs** file and remove the **Error()** method at the bottom of the class.

Try it Out

Choose the "IIS Production" profile

Run the application and make sure you still the get production error page

Switch to the "IIS Express" profile and make sure you still get the development error page

Lab 4: Handle Status Codes

Open the **HomeController.cs** file and comment out the line where you throw the ApplicationException.

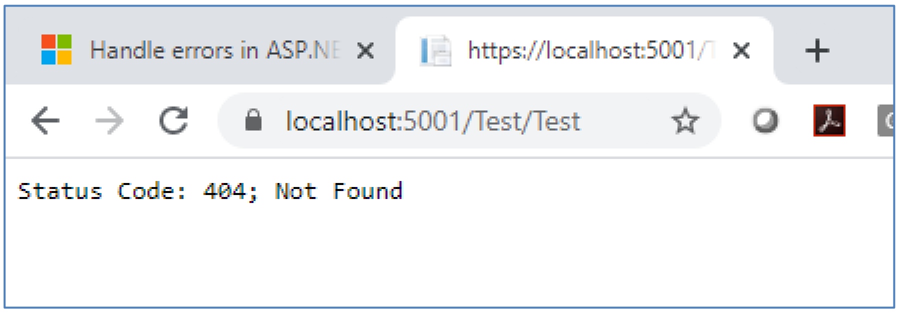
Open the **Program.cs** file and locate the app.UseStaticFiles() call and add the add the following code immediately following.

```
// Add code to display status code pages
app.UseStatusCodePages();
```

Try it Out

Run the application and after the http://localhost:nnnnn, type in **/BadPage**

You should see a page that looks like the following:



```
Status Code: 404; Not Found
```

Lab 5: Display Custom Status Page

Redirect to your own custom page when an error status code is raised.

Add the following code after the **app.UseStatusCodePages**() call you just added.

```
app.UseStatusCodePagesWithRedirects("/Error/StatusCodeRedirect?code=
{0}");
```

Open the **\Models\ErrorViewModel.cs** and add a *StatusCode* property

```
public int StatusCode { get; set; }
```

Open the **ErrorController.cs** file and create a new method named StatusCodeRedirect()

```
public IActionResult StatusCodeRedirect(int code)
{
  ErrorViewModel vm = new()
  {
    // Set status code property
    StatusCode = code
  };

  // Build page name from status code number
  return View("StatusCode" + vm.StatusCode.ToString(), vm);
}
```

Right mouse-click on the \Error folder and create a file named **StatusCode404.cshtml**

```
@model ErrorViewModel
@{
    ViewData["Title"] = "Page Not Found";
}

<h1>Page Not Found</h1>

<br />

<h2>You have reached the end of the internet!</h2>

<br />

<p>Status Code: @Model.StatusCode</p>
```

Try it Out

Run the application and after the http://localhost:nnnnn, type in **/BadPage**

You should now see the StatusCode404 page you created

Lab 6: Display Path on Status Code Page

Open **Program.cs** and add the following code just after the previous line of code you wrote

```
app.UseStatusCodePagesWithReExecute("/Error/StatusCodeReExecute",
"?code={0}");
```

COMMENT out the **app.UseStatusCodePagesWithRedirects()** method.

Open the **\Models\ErrorViewModel.cs** and add a *StatusPath* property

```
public string StatusPath { get; set; }
```

Open the **ErrorController.cs** file and add a using statement

```
using Microsoft.AspNetCore.Diagnostics;
```

Add a new method named StatusCodeReExecute()

```
public IActionResult StatusCodeReExecute(int code)
{
  ErrorViewModel vm = new()
  {
    // Set status code property
    StatusCode = code
  };

  // Get some path information
  var feature =
HttpContext.Features.Get<IStatusCodeReExecuteFeature>();
  if (feature != null) {
    vm.StatusPath =
        feature.OriginalPathBase
        + feature.OriginalPath
        + feature.OriginalQueryString;
  }

  // Build page name from status code number
  return View("StatusCode" + vm.StatusCode.ToString(), vm);
}
```

Open the **StatusCode404.cshtml** file and add the following code at the end of the file.

```
<p>Path: @Model.StatusPath</p>
```

Try it Out

Run the application and after the http://localhost:nnnnn, type in **/BadPage**

On the Status404 error page you see the page that was bad.