

Exception Handling Lab

Perform these labs on your own computer using Visual Studio 2022 to ensure you understand the lessons presented in the corresponding videos and lectures.

Lab 1: Built-In Global Exception Handling

Open the **CustomerController.cs** file and in the `Get()` method, add the following line of code just after the line **List<Customer> list;**

```
// Intentionally Cause an Exception  
throw new ApplicationException("ERROR!");
```

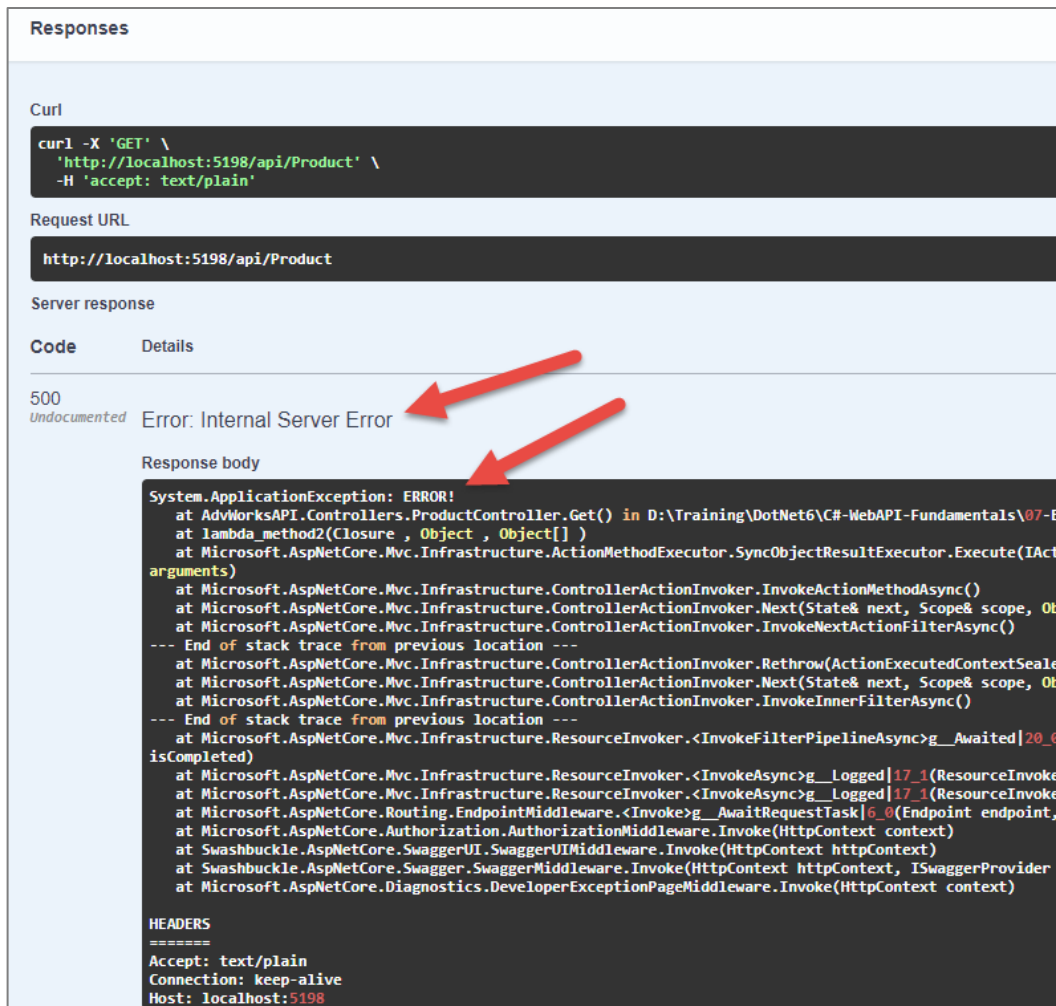
Try it Out

Run the application.

Click on the **GET /api/Customer** button.

You may need to go back to Visual Studio and click **Continue**.

View the exception that is returned (500 Error: Internal Server Error) similar to that in the screen shot below.



Responses

Curl

```
curl -X 'GET' \
'http://localhost:5198/api/Product' \
-H 'accept: text/plain'
```

Request URL

```
http://localhost:5198/api/Product
```

Server response

Code	Details
500 <i>Undocumented</i>	Error: Internal Server Error

Response body

```
System.ApplicationException: ERROR!
  at AdvWorksAPI.Controllers.ProductController.Get() in D:\Training\DotNet6\C#-WebAPI-Fundamentals\07-E
  at lambda_method2(Closure, Object, Object[])
  at Microsoft.AspNetCore.Mvc.Infrastructure.ActionMethodExecutor.SyncObjectResultExecutor.Execute(IAct
arguments)
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeActionMethodAsync()
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Ob
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeNextActionFilterAsync()
--- End of stack trace from previous location ---
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Rethrow(ActionExecutedContextSeale
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.Next(State& next, Scope& scope, Ob
  at Microsoft.AspNetCore.Mvc.Infrastructure.ControllerActionInvoker.InvokeInnerFilterAsync()
--- End of stack trace from previous location ---
  at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeFilterPipelineAsync>g__Awaited|20_0
isCompleted)
  at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Logged|17_1(ResourceInvoke
  at Microsoft.AspNetCore.Mvc.Infrastructure.ResourceInvoker.<InvokeAsync>g__Logged|17_1(ResourceInvoke
  at Microsoft.AspNetCore.Routing.EndpointMiddleware.<Invoke>g__AwaitRequestTask|6_0(Endpoint endpoint,
  at Microsoft.AspNetCore.Authorization.AuthorizationMiddleware.Invoke(HttpContext context)
  at Swashbuckle.AspNetCore.SwaggerUI.SwaggerUIMiddleware.Invoke(HttpContext httpContext)
  at Swashbuckle.AspNetCore.Swagger.SwaggerMiddleware.Invoke(HttpContext httpContext, ISwaggerProvider
  at Microsoft.AspNetCore.Diagnostics.DeveloperExceptionPageMiddleware.Invoke(HttpContext context)
```

HEADERS

```
=====
Accept: text/plain
Connection: keep-alive
Host: localhost:5198
```

Lab 2: Add Error Handling Controller

Right mouse-click on the **Controllers** folder and add a class named **ErrorController**.

Replace the contents of this file with the following code.

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

namespace AdvWorksAPI.Controllers;

public class ErrorController : ControllerBase
{
    [Route("/ProductionError")]
    [ApiExplorerSettings(IgnoreApi = true)]
    public IActionResult ProductionErrorHandler()
    {
        string msg = "Unknown Error";

        var features =
            HttpContext.Features.Get<IExceptionHandlerFeature>();
        if (features != null) {
            msg = features.Error.Message;
        }

        return
            StatusCode(StatusCode.Status500InternalServerError,
                msg);
    }
}
```

Open the **Program.cs** file and add the following line of code just **before** the call to the **app.UseAuthorization()** method.

```
// Enable Exception Handling Middleware
app.UseExceptionHandler("/ProductionError");
```

Try it Out

Run the application and click on the **GET /api/Customers** button.

You may need to go back to Visual Studio and click **Continue**.

View the exception that is returned (500 Error: Internal Server Error) similar to that in the screen shot below.

Responses

Curl

```
curl -X 'GET' \  
'http://localhost:5198/api/Product' \  
-H 'accept: text/plain'
```

Request URL

```
http://localhost:5198/api/Product
```

Server response

Code	Details
500 <i>Undocumented</i>	Error: Internal Server Error

Response body

```
ERROR!
```

Lab 3: Development and Production Exception Handling

Open the **Program.cs** file and replace the code from the last lab to the following lines of code.

```
// Enable Exception Handling Middleware  
if (app.Environment.IsDevelopment()) {  
    app.UseExceptionHandler("/DevelopmentError");  
}  
else {  
    app.UseExceptionHandler("/ProductionError");  
}
```

Open the **ErrorController.cs** file and add a new method named the **DevelopmentErrorHandler()** method.

```
[Route("/DevelopmentError")]
[ApiExplorerSettings(IgnoreApi = true)]
public IActionResult DevelopmentErrorHandler()
{
    string msg = "Unknown Error";

    var features =
HttpContext.Features.Get<IExceptionHandlerFeature>()!;

    if (features != null) {
        msg = "Message: " + features.Error.Message;
        msg += Environment.NewLine + "Source: " +
features.Error.Source;
        msg += Environment.NewLine +
features.Error.StackTrace;
    }

    return
StatusCode(StatusCode.Status500InternalServerError,
msg);
}
```

Try it Out

Run the application and click on the **GET /api/Customer** route to cause an exception.

You should now see the message, the source, and the stack trace.

Lab 4: Add Production Profile

Open the `\Properties\launchSettings.json` file.

Copy the JSON object "AdvWorksAPI" to a new JSON object named **"AdvWorksAPI Production"**.

Change the areas in the copied JSON object with the values in **bold** below.

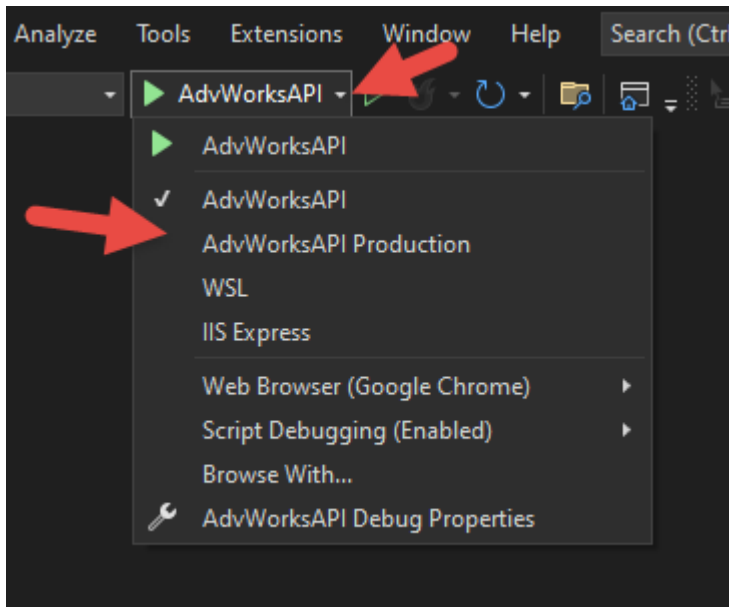
Replace the **nnnn** with your port number.

```
"AdvWorksAPI Production": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "launchUrl": "swagger",
  "applicationUrl": "http://localhost:nnnn",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Production"
  }
},
```

Save the `launchSettings.json` file.

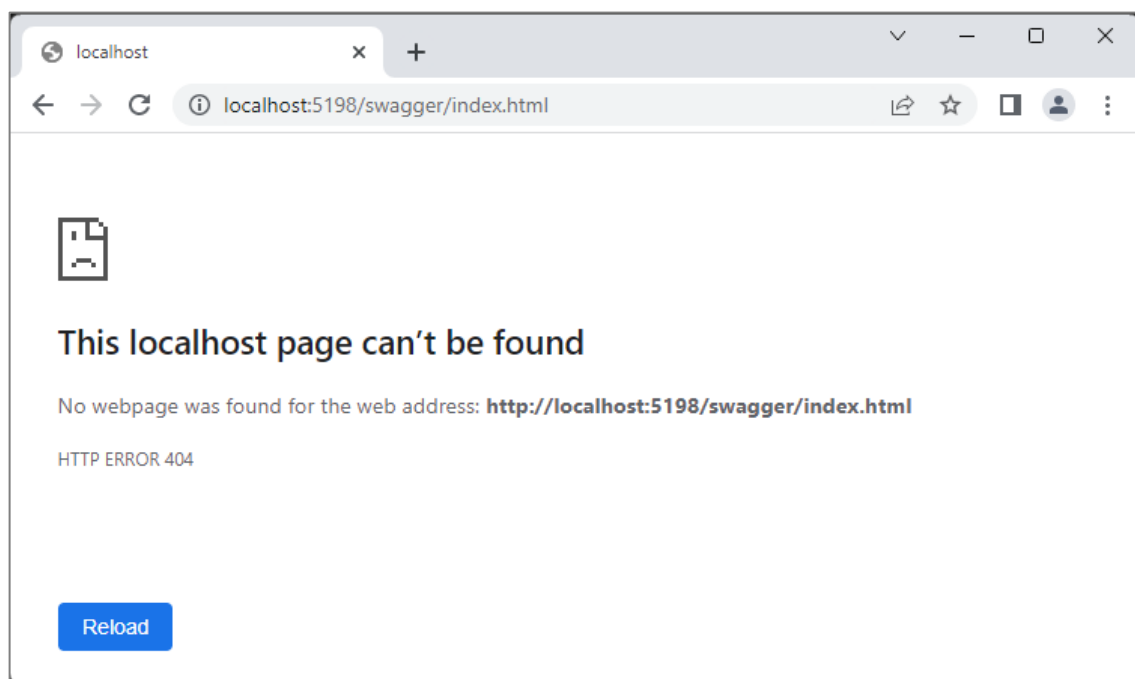
Try it Out

Click on the down arrow next to "AdvWorksAPI" on the VS command bar.



Select the "**AdvWorksAPI Production**" profile.

Run the application and you will get a 404 error.



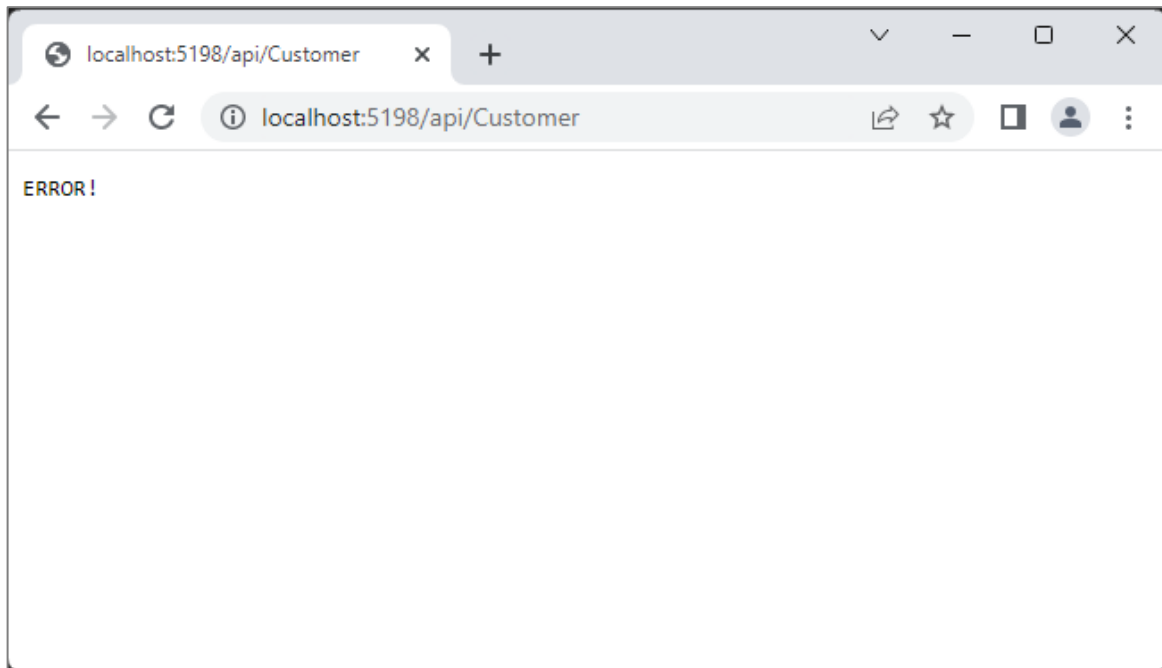
You get this error because Swagger does not appear when running in Production mode.

Type the following into the browser URL line (replacing your **PORT** number for the **nnnn**).

```
http://localhost:nnnn/api/Customer
```

Click **Continue** in Visual Studio when the exception occurs.

Now you just have the error message returned in the browser as shown below.



Lab 5: Handle Status Codes

Open the **ErrorController.cs** file and add a new method.


```
[Route("/StatusCodeHandler/{code:int}")]
[ApiExplorerSettings(IgnoreApi = true)]
public IActionResult StatusCodeHandler(int code)
{
    IActionResult ret;
    string msg = $"Code is not handled: '{code}'";

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

    switch (code) {
        case 404:
            msg = $"API Route Was Not Found: '{msg}'";
            ret = StatusCode(StatusCode.Status404NotFound,
msg);
            break;
        default:
            ret =
StatusCode(StatusCode.Status500InternalServerError,
msg);
            break;
    }

    return ret;
}
```

Open **Program.cs** and just below the `app.UseExceptionHandler()` add

```
// Handle status code errors in the range 400-599
app.UseStatusCodePagesWithReExecute("/StatusCodeHandler/
{0}");
```

Try it Out

While still in production mode, run the app.

You should now see the 404 status returned because swagger is not found.

Lab 6: Log Exceptions and Informational Messages into Different Files

Open the **Program.cs** file and modify the configuration of Serilog so you have two files: one for informational messages and higher and the other for exceptions.

```
// Configure logging to Console & File using Serilog
builder.Host.UseSerilog((ctx, lc) =>
{
    // Log to Console
    lc.WriteTo.Console();
    // Log to Rolling File
    lc.WriteTo.File("Logs/InfoLog-.txt",
        rollingInterval: RollingInterval.Day,
        restrictedToMinimumLevel:
LogEventLevel.Information);
    // Log Errors to Rolling File
    lc.WriteTo.File("Logs/ErrorLog-.txt",
        rollingInterval: RollingInterval.Day,
        restrictedToMinimumLevel: LogEventLevel.Error);
});
```

Try it Out

Delete any log files under the **Logs** folder.

Switch your profile back to **AdvWorksAPI**.

Run the application and click on the **GET /api/Customer** button.

Stop the application.

View the **Logs** folder and you should see two different log files.

NOTE:	You still get exceptions in the InfoLog.txt file because you can only set the minimum level. Look up Serilog.Filters.Expressions and how to configure which log levels go into which files.
--------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Lab 7: Log Exceptions in Catch Block

Open the **CustomerController.cs** file and add a new field

```
private readonly ILogger<CustomerController> _Logger;
```

Modify the constructor.

```
public CustomerController(IRepository<Customer> repo,  
ILogger<CustomerController> logger)  
{  
    _Repo = repo;  
    _Logger = logger;  
}
```

Modify the Get() method with the code shown in bold.

```
[HttpGet]
[ProducesResponseType (StatusCodes.Status200OK) ]
[ProducesResponseType (StatusCodes.Status404NotFound) ]
[ProducesResponseType (StatusCodes.Status500InternalServerError)]
public ActionResult<IEnumerable<Customer>> Get()
{
    ActionResult<IEnumerable<Customer>> ret;
    List<Customer> list;
    string msg = "No Customers are available.";

    try {
        // Intentionally Cause an Exception
        throw new ApplicationException("ERROR!");

        // Get all data
        list = _Repo.Get();

        if (list != null && list.Count > 0) {
            ret = StatusCode(StatusCodes.Status200OK, list);
        }
        else {
            ret = StatusCode(StatusCodes.Status404NotFound,
msg);
        }
    }
    catch (Exception ex) {
        msg = "Error in CustomerController.Get()";
        msg += $"{Environment.NewLine}Message:
{ex.Message}";
        msg += $"{Environment.NewLine}Source: {ex.Source}";

        _Logger.LogError(ex, "{msg}", msg);

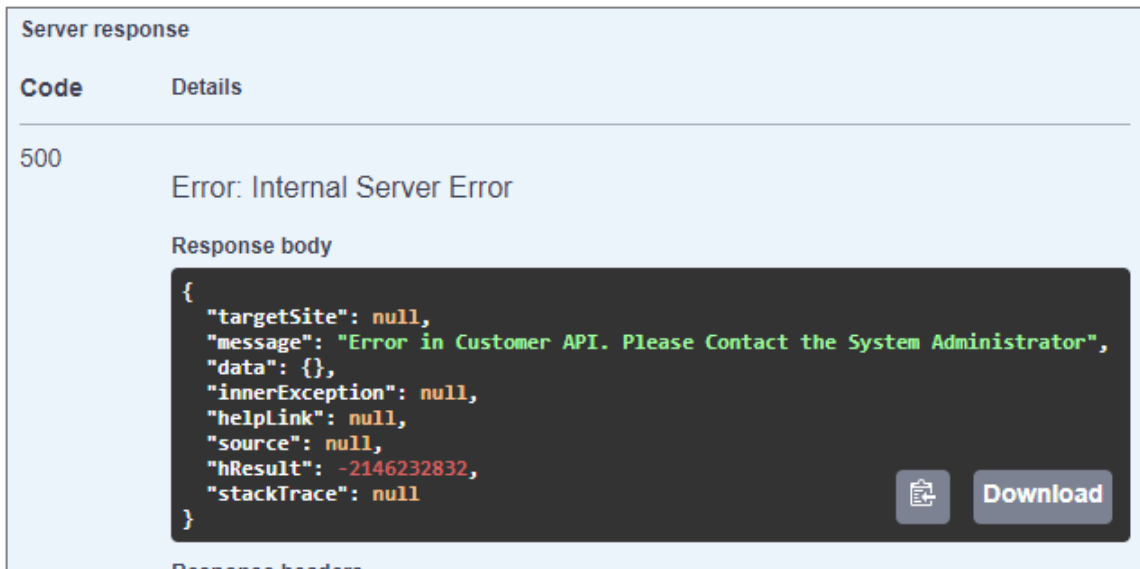
        ret =
StatusCode(StatusCodes.Status500InternalServerError,
        new ApplicationException("Error in Customer API.
Please Contact the System Administrator."));
    }

    return ret;
}
```

Try it Out

Delete any log files in the **Logs** folder.

Run the application and click on the **GET /api/Customer** button.
You should now see the error displayed.



The screenshot displays a 'Server response' window. It features a table with two columns: 'Code' and 'Details'. The 'Code' column shows '500'. The 'Details' column shows 'Error: Internal Server Error'. Below this, the 'Response body' is shown as a JSON object in a dark-themed code editor. The JSON object contains the following fields: 'targetSite' (null), 'message' ('Error in Customer API. Please Contact the System Administrator'), 'data' ({}), 'innerException' (null), 'helpLink' (null), 'source' (null), 'hResult' (-2146232832), and 'stackTrace' (null). To the right of the JSON code, there are two buttons: a 'Copy' button with a clipboard icon and a 'Download' button.

Code	Details
500	Error: Internal Server Error

Response body

```
{
  "targetSite": null,
  "message": "Error in Customer API. Please Contact the System Administrator",
  "data": {},
  "innerException": null,
  "helpLink": null,
  "source": null,
  "hResult": -2146232832,
  "stackTrace": null
}
```

Copy Download

Stop the application.

Open the **\\Logs\\ErrorLog-nnnn.txt** file and view the error information.