Error : When something goes error

1. Syntax Error ; { }, Main main
2. Logical Error , in this case , we always get some result, but that’s not correct
3. Exception / Run Time Errors

* A: Exception Handling in Development Environment for ASP.NET Core MVC

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using ExceptionHandling\_LoggingDemo.Models;

using System.IO;

namespace ExceptionHandling\_LoggingDemo.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

**1st Demo**

**public IActionResult Index(int? id)**

**{**

**throw new Exception("Error");**

}

**2nd Demo**

**public IActionResult Index(int? id)**

**{**

**if(id.HasValue)**

**{**

**if(id==1)**

**throw new FileNotFoundException("File not found exception thrown in index.chtml");**

**else if (id == 2)**

**{**

**return StatusCode(500);**

**}**

**}**

**return View();**

**}**

}

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

Index.cshtml

@{

ViewData["Title"] = "Home Page";

}

<div class="text-left">

<p>

<a href="/NoSuchPage">

Request an endpoint that doesn't exist. Trigger a 404

</a>.

</p>

<p><a href="/home/index/1">Trigger an exceptionn</a>.</p>

<p><a href="/home/index/2">Return a 500 error.</a>.</p>

</div>

**In Startup.cs file**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Diagnostics;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace ExceptionHandling\_LoggingDemo

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

**app.UseDeveloperExceptionPage(); // For Developers**

}

//else

//{

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

// // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

// app.UseHsts();

//}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

* B: Exception Handling in Production Environment for ASP.NET Core MVC
  + Approach 1: UseExceptionHandler
    - 1: Exception Handler Page

launchSettings.json

{

"iisSettings": {

"windowsAuthentication": false,

"anonymousAuthentication": true,

"iisExpress": {

"applicationUrl": "http://localhost:52667",

"sslPort": 44311

}

},

**"profiles": {**

**"IIS Express": {**

**"commandName": "IISExpress",**

**"launchBrowser": true,**

**"environmentVariables": {**

**"ASPNETCORE\_ENVIRONMENT": "Production"**

**}**

},

"ExceptionHandling\_LoggingDemo": {

"commandName": "Project",

"launchBrowser": true,

"applicationUrl": "https://localhost:5001;http://localhost:5000",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Production"

}

}

}

}

In Startup.cs file

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Diagnostics;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace ExceptionHandling\_LoggingDemo

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

**app.UseDeveloperExceptionPage(); // For Developers**

}

**else**

**{**

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

**// // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.**

**app.UseHsts();**

**}**

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

* Exception Handling in Production Environment for ASP.NET Core MVC
  + Approach 1: UseExceptionHandler
    - 1: Exception Handler Page
    - **2: Exception Handler Lambda**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using ExceptionHandling\_LoggingDemo.Models;

using System.IO;

namespace ExceptionHandling\_LoggingDemo.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index(int? id)

{

**if (id.HasValue)**

**{**

**if (id == 0)**

**{**

**int x = 10;**

**int? res = x / id;**

**throw new DivideByZeroException();**

**}**

**if (id == 1)**

**throw new FileNotFoundException("File not found exception thrown in index.chtml");**

**else if (id == 2)**

**{**

**return StatusCode(500);**

**}**

**}**

**return View();**

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Diagnostics;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace ExceptionHandling\_LoggingDemo

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

**else**

**{**

**app.UseExceptionHandler(errorApp =>**

**{**

**errorApp.Run(async context =>**

**{**

**context.Response.StatusCode = 500;**

**context.Response.ContentType = "text/html";**

**await context.Response.WriteAsync("<html lang=\"en\"><body>\r\n");**

**await context.Response.WriteAsync("ERROR!<br><br>\r\n");**

**var exceptionHandlerPathFeature =**

**context.Features.Get<IExceptionHandlerPathFeature>();**

**if (exceptionHandlerPathFeature?.Error is FileNotFoundException)**

**{**

**await context.Response.WriteAsync(**

**"File error thrown!<br><br>\r\n");**

**}**

**if (exceptionHandlerPathFeature?.Error is DivideByZeroException)**

**{**

**await context.Response.WriteAsync(**

**"No.cant be divided by 0 !<br><br>\r\n");**

**}**

**await context.Response.WriteAsync(**

**"<a href=\"/\">Home</a><br>\r\n");**

**await context.Response.WriteAsync("</body></html>\r\n");**

**await context.Response.WriteAsync(new string(' ', 512));**

**});**

**});**

**app.UseHsts();**

**}**

//else

//{

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

// // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

// app.UseHsts();

//}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

* + Approach 2: UseStatusCodePages

We can use

app.UseStatusCodePages(

"text/plain", "Status code page, status code: {0}");

Create Separate Controller : Error , For Globally handling Exceptions

[Route("Error")]

public IActionResult Error()

{

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

ViewBag.ExceptionPath = ExecptionDetails.Error;

ViewBag.ExceptionMessage = ExecptionDetails.Error.Message;

ViewBag.StaclTrace = ExecptionDetails.Error.StackTrace;

return View("Error");

}