4. Exception Handling

4.1 UseDeveloperExceptionPage

- The Developer Exception Page Middleware in ASP.NET Core is an essential tool for developers, particularly during the development phase of a web application.
- Purpose :
  It's designed to catch unhandled exceptions that occur during the request pipeline processing in an ASP.NET Core application. When an exception is thrown, this middleware displays a detailed error page, showing a stack trace and other debugging information.
- Use During Development :
  It's typically used only in the development environment, as exposing detailed error information in production can be a security risk.
- Customization :
  Developers can customize the display of the exception details to include things like source code snippets and variable values at the time of the exception.
- Component of DeveloperExceptionPage are as follows :
  - Stack : The Stack tab gives the information of stack trace, which indicates where exactly the exception occurred, the file name, and the line number that caused the exception.
  - Query : The query string of the request that caused the exception is shown. This is useful for understanding the parameters that were passed to the server at the time of the error.
  - Cookies : Cookies are key-value pairs sent by the server and stored in the client's browser. They are sent back to the server with each request. The Developer Exception Page shows all cookies associated with the request. This can be crucial for debugging issues related to session management, authentication, or user preferences.
  - Header : HTTP headers contain information about the request and the client. They include data like content type, response status code, client information, and more. The Developer Exception Page lists all the request headers, which can be vital for troubleshooting issues related to caching, content negotiation, authentication, etc.
  - Routing : This refers to how ASP.NET Core maps incoming requests to route handlers. The Developer Exception Page can show information about

the route that was matched (if any) for the request that caused the exception. This information is useful for debugging issues related to route configuration, URL patterns, and route constraints.

- We should Enable the Developer Exception Page Middleware only when the application runs in the Development environment because we don’t want to share detailed exception information when the application runs in production.

4.2 UseExceptionHandler

- For handling exceptions in the Production environment UseExceptionHandler extension method is used.
- The UseExceptionHandler extension method can be used to configure custom error handling routes.
- Example :

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if(env.IsDevelopment())
     {
            app.UseDeveloperExceptionPage();
     }
     else
     {
            app.UseExceptionHandler("/Home/Error");
      }
 }
```

- When an error occurs in the ASP.NET Core MVC application, the control will be redirected to /home/error.
- We can use a lambda expression in the UseExceptionHandler extension method for using a custom exception handler page.