Exception Handling

ASP .NET Core

Yash Lathiya

Table of Contents

UseDeveloperExceptionPage

Middleware component that provides detailed error information during development.

1. **Purpose**:

   To display detailed information about exceptions that occur in the application, aiding developers in diagnosing and fixing errors quickly.

2. **Development Environment**:

   This middleware is typically used in the development environment.

   It should be included in the Configure method of the Startup class when the application is running in a development setting.

3. **Detailed Error Page**:

   When an exception occurs, the Developer Exception Page provides a detailed error page that includes the exception message, stack trace, and source code where the exception was thrown. This information is invaluable for debugging.

4. **Security Considerations**:

   Since the Developer Exception Page exposes sensitive information about the application's internal workings, it is crucial to ensure that it is not enabled in production environments. Exposing detailed error information in production can lead to security vulnerabilities.

5. **Configuration**:

   No additional configuration is necessary to use the Developer Exception Page beyond calling app.UseDeveloperExceptionPage() within the appropriate environment check.

6. **Summary**: crucial tool for ASP.NET Core developers, providing detailed error information during development to facilitate rapid debugging and issue resolution, while ensuring such information is not exposed in production environments.

UseExceptionHandler

Middleware component used to handle exceptions in a centralized manner in production environments.

1. **Purpose**:

   To provide a way to handle exceptions gracefully and display a user-friendly error page or response when an unhandled exception occurs in the application.

2. **Production Environment**:

   Used in production environments to ensure that detailed error information is not exposed to end-users, which could potentially reveal sensitive information and pose security risks.

3. **Error Handling Path**:

   Can generate a user-friendly error page. This path is typically mapped to an action in a controller or a Razor page that displays an error message.

4. **Custom Error Pages**:

   Developers can create custom error pages that provide a more pleasant user experience. These pages can include helpful messages and navigation options, while ensuring sensitive details about the exception are not exposed.

5. **Logging Exceptions**:

   It's common practice to log the details of the exceptions for further analysis. This can be done within the custom error handling endpoint or through middleware that logs exceptions before they are handled.

6. **Fallback Behavior**:

   If no specific path is provided, app.UseExceptionHandler() will use a default behavior to handle exceptions. However, specifying a custom error handling path allows for more control and customization over the user experience.

7. **Security Considerations**:

   By handling exceptions in a centralized manner and providing user-friendly error pages, app.UseExceptionHandler() helps protect the application from exposing sensitive information and maintains a consistent user experience during unexpected errors.

8. **Summary** : app.UseExceptionHandler() is an essential middleware for handling exceptions in ASP.NET Core applications, particularly in production environments. *It ensures that detailed error information is hidden from end-users*, provides a way to display user-friendly error messages, and supports logging for further analysis.