Instructor example

Web University by Harsha Vardhan

How do you handle errors in asp.net core application?

Errors arising in controller / view / result / middleware execution can be best handled with exception middleware.

```
public class ExceptionHandlingMiddleware : IMiddleware
{
  public async Task InvokeAsync(HttpContext context, RequestDelegate next)
  {
   try
   {
    await next(context);
   }
   catch (Exception ex)
   {
    //log errors
   }
  }
}

Apply the exception handling middleware before all custom middleware:

if (builder.Environment.IsDevelopment())
{
  app.UseDeveloperExceptionPage(); //enables developer exception page on exception in “Development” environment
}
else
{
  app.UseMiddleware<ExceptionHandlingMiddleware>(); //adds ExceptionHandlingMiddleware in case of other than “D
}
```

Status code pages

To enable custom error pages on specific status codes:

```
app.UseStatusCodePagesWithRedirects("url”); //it redirects to the specified url when exception occurs
with specific status code such as 400, 500, 404 etc.
```

How do you choose between Exception Middleware and Exception filter?

Though we can also create Exception filter by implementing IExceptionFilter interface, it’s not recommended unless you actually need it.

Exception filter handles unhandled exceptions that occur in controller creation, model binding, action filters or action methods.

Exception filter doesn't handle the unhandled exceptions that occur in authorization filters, resource filters, result filters or IActionResult execution.

Exception filters are recommended to be used only when you want a different error handling and generate different result for specific controllers; otherwise, ErrorHandlingMiddleware is recommended over Exception Filters.

Your submission

Cesar David Guerrero Regino

How do you handle errors in asp.net core application?

How do you choose between Exception Middleware and Exception filter?