**开发者异常页面**

在开放环境中使用，发生错误时，将返回详细的开放异常页面

public class Startup

{

    public void Configure (IApplicationBuilder app, IHostingEnvironment env)

    {

        if (env.IsDevelopment ()) {

            // 使用开发者异常页

            app.UseDeveloperExceptionPage ();

        }

    }

}

**定制异常处理**

在生产环境中，我们希望返回的是我们定制的异常，而不是开发者异常

public class Program {

    public static void Main () {

        RequestDelegate handler = async context => await context.Response.WriteAsync ("Unhandled exception occurred!");

        new WebHostBuilder ()

            .UseKestrel ()

            .Configure (app => app

                // 使用异常处理器

                .UseExceptionHandler (new ExceptionHandlerOptions { ExceptionHandler = handler })

                .Run (context => Task.FromException (new InvalidOperationException ("Manually thrown exception..."))))

            .Build ()

            .Run ();

    }

}

使用错误页面

if (env.IsDevelopment ())

{

    // 开发者异常页面

    app.UseDeveloperExceptionPage ();

}

else

{

    // 定制异常页面，当发生异常时，浏览器将重定向到该页面

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

}

**针对响应状态码定制错误**

前面的处理只能捕获500服务器异常

UseStatusCodePages可以捕获如404等基于的状态码异常

Func<StatusCodeContext, Task> handler = async context => {

    var response = context.HttpContext.Response;

    if (response.StatusCode < 500) {

        await response.WriteAsync ($"Client error ({response.StatusCode})");

    } else {

        await response.WriteAsync ($"Server error ({response.StatusCode})");

    }

};

new WebHostBuilder ()

    .UseKestrel ()

    .Configure (app => app

        // 使用状态码错误处理

        .UseStatusCodePages (handler)

        .Run (context => Task.Run (() => context.Response.StatusCode = \_random.Next (400, 599))))

    .Build ()

    .Run ();