Global Exception Handler

Using IExceptionHandler abstraction helps to implement the custom global exception handler.

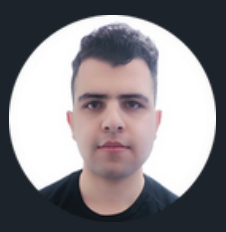
```
internal sealed class ExceptionHandler(ILogger<ExceptionHandler> logger)
  : IExceptionHandler
{
  public async ValueTask<bool> TryHandleAsync(
    HttpContext httpContext,
    Exception exception,
    CancellationToken cancellationToken)
  {
    logger.LogError(
      exception, "Exception Message: {Message}", exception.Message);

    var problemDetails = new ProblemDetails
    {
      Status = StatusCodes.Status500InternalServerError,
      Title = "Server failure",
      Detail = "An unexpected error occurred on the server.",
      Instance = httpContext.Request.Path,
      Type = "https://example.com/error/server-failure"
    };

    httpContext.Response.StatusCode = problemDetails.Status.Value;

    await httpContext.Response
      .WriteAsJsonAsync(problemDetails, cancellationToken);

    return true;
  }
}
```



Elliot One

Configuring Exception Handler

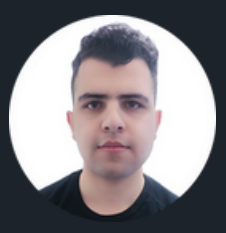
Registering IExceptionHandler implementation alongside Problem Details helps to capture exceptions.

```
builder.Services.AddExceptionHandler<ExceptionHandler>();
builder.Services.AddProblemDetails();
```

```
app.UseExceptionHandler();
```

When an exception occurs:

```
{
  "type": "https://example.com/error/server-failure",
  "title": "Server failure",
  "status": 500,
  "detail": "An unexpected error occurred on the server.",
  "instance": "/"
}
```



Elliot One

Elliot One

Enjoyed Reading This?

Reshare and Spread Knowledge.