# Gestione degli errori web.api (ApiException)

## Ambiente

L’ambiente in cui andiamo ad innestare questa gestione è in .Net Core 2.2 che risponde, nello specifico, ad un’applicazione Angular ma è sfruttabile da qualsiasi sistema in grado di consumere un web service REST.

### Intercettare un errore

E’ possibile intercettare gli errori posizionando la chiamata UseExceptionHandler prima di qualsiasi middleware in cui si voglio rilevare le eccezioni.

Quindi nella classe ***Startup.cs*** ho aggiunto quanto segue, prima di ogni altro middleware :

app.UseExceptionHandler(errorApp =>

{

errorApp.Run(async context =>

{

var exceptionHandlerPathFeature =

context.Features.Get<IExceptionHandlerPathFeature>();

**if (exceptionHandlerPathFeature?.Error is ApiException myError)**

{

context.Response.StatusCode = (int)(myError).StatusCode;

context.Response.ContentType = "text/json";

ErrorResponseModel erm = new ErrorResponseModel();

erm.HttpStatusCode = (int)(myError).StatusCode;

erm.ErrorData = myError.ErrorDataList;

var json = JsonConvert.SerializeObject(erm);

await context.Response.WriteAsync(json);

}

else

{

context.Response.StatusCode = 500;

context.Response.ContentType = "text/html";

await context.Response.WriteAsync("<html lang=\"en\"><body>\r\n");

await context.Response.WriteAsync("ERROR!<br><br>\r\n");

// Use exceptionHandlerPathFeature to process the exception (for example,

// logging), but do NOT expose sensitive error information directly to

// the client.

if (exceptionHandlerPathFeature?.Error is FileNotFoundException)

{

await context.Response.WriteAsync("File error thrown!<br><br>\r\n");

}

await context.Response.WriteAsync("<a href=\"/\">Home</a><br>\r\n");

await context.Response.WriteAsync(string.Format("<span>{0}</span>", (exceptionHandlerPathFeature?.Error).Message));

await context.Response.WriteAsync("</body></html>\r\n");

await context.Response.WriteAsync(new string(' ', 512)); // IE padding

}

});

});

Il fulcro della soluzione risiede nella if evidenziata, che sostanzialmente discrimina gli errori gestiti nella applicazione, che forniscono un oggetto ApiException (che eredita da Exception), dagli errori generici.

In quest’ultima caso verrà restituito un errore siffatto :

* Codice Errore 500
* Tipo restituito text/html
* Messaggio restituito : ERROR

Nel caso di un errore ApiException invece avremo a disposizione il seguente oggetto :

public HttpStatusCode StatusCode { get; }

public List<ErrorData> ErrorDataList { get; }

/// <summary>

/// Error data details as string message.

/// </summary>

public virtual string ErrorDataMessage

{

get

{

StringBuilder sb = new StringBuilder();

foreach (var item in ErrorDataList)

{

sb.AppendLine(string.Format("ErrorCode: {0}, Field: {1},Description {2}", item.ErrorCode, item.Field, item.ErrorDescription));

}

return sb.ToString();

}

}

ErrorDataList è la lista che contiene gli errori, così che possa indicare in un’unica eccezione più errori.

L’oggetto ErrorData è il seguente :

public string Field { get; set; }

//

// Summary:

// Application error code

public string ErrorCode { get; set; }

//

// Summary:

// Error descriptions

public string ErrorDescription { get; set; }

L’ApiException ha diversi costruttori che ho indicato di seguito :

protected ApiException() : base(HttpStatusCode.InternalServerError.ToString())

{

StatusCode = HttpStatusCode.InternalServerError;

ErrorDataList = new List<ErrorData>();

}

public ApiException(HttpStatusCode statusCode) : base(statusCode.ToString())

{

StatusCode = statusCode;

ErrorDataList = new List<ErrorData>();

}

public ApiException(HttpStatusCode statusCode, Exception innerException) : base(statusCode.ToString(), innerException)

{

StatusCode = statusCode;

ErrorDataList = new List<ErrorData>();

}

public ApiException(HttpStatusCode statusCode, List<ErrorData> errorDataList) : this(statusCode)

{

ErrorDataList = errorDataList;

}

public ApiException(HttpStatusCode statusCode, List<ErrorData> errorDataList, Exception innerException) : this(statusCode, innerException)

{

ErrorDataList = errorDataList;

}

Adesso non rimane che vedere come è possibile sollevare un’eccezione di tipo ApiException :

….

}

catch (Exception err)

{

List<ErrorData> errorDataList = new List<ErrorData>();

ErrorData errore1 = new ErrorData("field", "999", err.Message);

errorDataList.Add(errore1);

throw new ApiException(HttpStatusCode.BadRequest, errorDataList);

}