

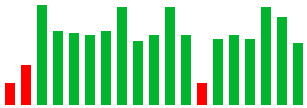
build passing quality gate failed coverage 24.8%

Build history for master (last 19 builds)

Max build time: 00:02:21

Min build time: 00:00:31

Avg build time: 00:01:40



Workshop Refit & Polly

Jacob Duijzer, January 2018

Refit

Refit: The automatic type-safe REST library for .NET Core, Xamarin and .NET

```
public interface IRemoteApi
{
    [Get("/posts")]
    Task<IEnumerable<Post>> GetAllPostsAsync();

    [Post("/posts")]
    [Headers("Authorization: Bearer")]
    Task<Post> AddPostAsync([Body]Post post);

    [Delete("/posts/{id}")]
    [Headers("Authorization: Bearer")]
    Task DeletePostByIdAsync(int id);
}
```

Refit - sample code

```
var remoteApi = RestService
    .For<IRemoteApi>("http://localhost:3000");

var posts = await remoteApi.GetAllPostsAsync();

var singlePost = await remoteApi.GetPostByIdAsync(1);
```

Refit - Scenario 1

Simple api calls

Refit - Scenario 2

Logging

Refit - Scenario 3

Authenticated api calls

Polly

Policies:

- Retry
- CircuitBreaker
- Timeout
- Bulkhead Isolation
- Cache
- Fallback
- PolicyWrap

Polly - sample code

```
// Retry multiple times, calling an action on each retry
// with the current exception, retry count and context
// provided to Execute()
var _retryPolicy = Policy.Handle<SomeExceptionType>()
    .Retry(3, (exception, retryCount, context) =>
    {
        // do something to prevent the next exception
    });

await _retryPolicy.ExecuteAsync(
    remoteApi.GetAllPostsAsync()
);
```


Polly - Scenario 1

Retry

Polly - Scenario 2

Fallback

Polly - Scenario 3

CircuitBreaker

Polly - Scenario 4

Retry with fallback

Polly - Scenario 5

CircuitBreakerWithRetryAndFallback

HttpClientHandler

- [MSDN Documentation](#)
- [Scott Hanselmans post With REFIT](#)

Polly - Unit testing

Alternatives - Flurl

```
C#!  
// Flurl will use 1 HttpClient instance per host  
var person = await "https://api.com"  
    .AppendPathSegment("person")  
    .SetQueryParams(new { a = 1, b = 2 })  
    .WithOAuthBearerToken("my_oauth_token")  
    .PostJsonAsync(new  
    {  
        first_name = "Claire",  
        last_name = "Underwood"  
    })  
    .ReceiveJson<Person>();
```

- [Flur.io](#)
- [Blog post](#)

Alternatives - DalSoft.RestClient

```
C#!
var config = new Config().UseFormUrlEncodedHandler();

dynamic restClient = new RestClient("https://jsonplaceholder.typicode.com");
var user = new User { name="foo", email="foo@bar.com" };

//POST name=foo&email=foo@bar.com https://jsonplaceholder.typicode.com/users
result = await client
    .Headers(new { ContentType = "application/x-www-form-urlencoded" })
    .User(1)
    .Post(user);
```

- [Blogpost](#)
- [Github](#)

Setup sample project

<https://github.com/jacobduijzer/RefitPollyWorkshop>

- Get the source from [GitHub](#)
- in folder src/api: npm install
- in the folder src/api: npm run start
- configure & run the console app

Links

- [Sample repository](#)
- [Refit](#)
- [Polly](#)
- [JSON Server](#)
- Presentation created from markdown with [Marp](#)

