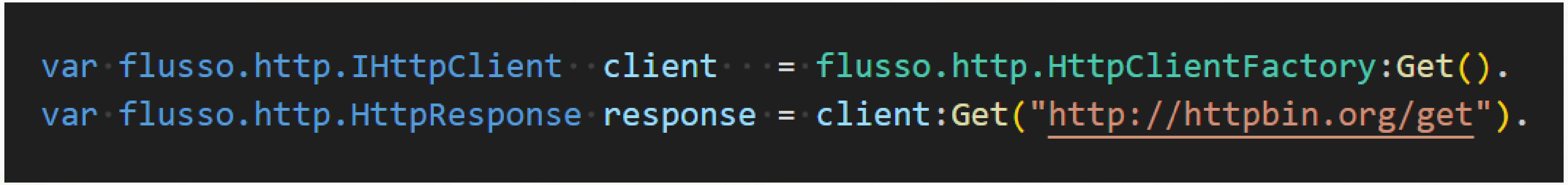
HttpClient for OpenEdge

With ABL or .NET on Windows and Linux



```
var flusso.http.IHttpClient  client   = flusso.http.HttpClientFactory:Get().
var flusso.http.HttpResponse response = client:Get("http://httpbin.org/get").
```

Introduction

- Arno van der Ende
- Richard Kelters

- Flusso B.V.

Flusso

- Flusso is a technical IT-partner with focus on durable and future proof software solutions for
    - Progress
    - Java
    - frontend development
    - OutSystems
    - cloud-native development
- We support businesses with their digital transformation
    - modernising existing applications
    - develop new applications
    - integrate applications in complex environments
    - connect technology, teams and processes

Flusso

- We develop by high standards and make use of
  - AI-tools like Codeium in Windsurf IDE
  - containerized developement environments with Kubernetes
  - Keycloak for identity and access management
  - low-code solutions with OutSystems
- At Flusso colaboration is our main focus. We believe in
  - transparant communication
  - shared responsibility
  - deliver solutions that really work
  - colaboration without fuss

Check out our podcasts on LinkedIn!

Agenda

- What is an HttpClient
- Use case
- Http recap
- OpenEdge
- .NET
- Installation
- Solution
- Demo
- Results
- Questions

What is an HttpClient?

With an HttpClient you can make http-requests within your application

You can exchange data with other parties, other applications and systems

Similar as:

- webbrowsers
- tools: postman, soapui, rest-client (vscode extension), etc.
- command line: curl

(HTTP=Hyper Text Transport Protocol)

Use case HttpClient

Connecting the world

- Interaction with applications
- Get information from internet
- REST
- Secure communication

Alternatives

Before we had an HttpClient we would solve exchanging information with other systems by

- file / shared folders
- mail / smtp / pop3
- ftp / sftp
- jms (Sonic adapter)
- homebrewn socket solution (stomp adapter)

In most cases third party solutions were involved to accommodate the above.

Why this HttpClient

- We had some performance issues with previous OpenEdge versions during a project and had to resolve to other technology.

- With the .NET Core support on Linux for OpenEdge we thought it would be an interesting project to facilitate a solution for the community that could be useful in all environments

Http Basics

- Method
- Url
- Headers
- Body

Methods

Get	read
Post	create
Put	update
Delete	delete
Patch	update, only changed properties in request
(Query)	read, with request body
other	head, trace, options and connect

flusso IT

Url

https://www.github.com/Company/Project?field=name&table=user

1. protocol
2. domain/host
3. path
4. query parameters

https://datatracker.ietf.org/doc/html/rfc3986

Headers

- Accept
- Content-Type
- Content-Length
- Cache-Control
- Authorization

Mime-Types (Media Types)

```
text/plain
text/html
text/xml
application/xml
application/json
application/zip
application/pdf
image/png
image/jpeg
audio/mpeg
video/mp4
```

Http Status Code

- 200 Ok
- 201 Created
- 204 No Content
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden
- 404 Not Found
- 405 Method Not Allowed
- 409 Conflict
- 410 Gone
- 418 I'm a teapot
- 422 Unprocessable Content
- 500 Internal Server Error

https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference

flusso IT

History OpenEdge

Relevant to HttpClient

1999	●	9.1A Sockets
2003	●	10.1A Object Oriented
2008	●	10.2 GUI for .NET
2015	●	11.5.1 OpenEdge.Net.HTTP.*
2023	●	12.7 .NET Core Windows
2024	●	12.8 .NET Core on Linux

The world is your oyster

Calling REST services from ABL

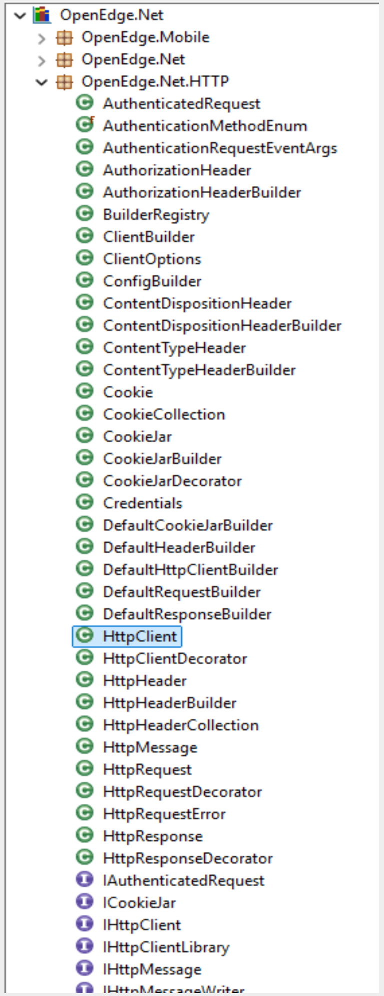
Peter Judge
pjudge@progress.com

flusso IT

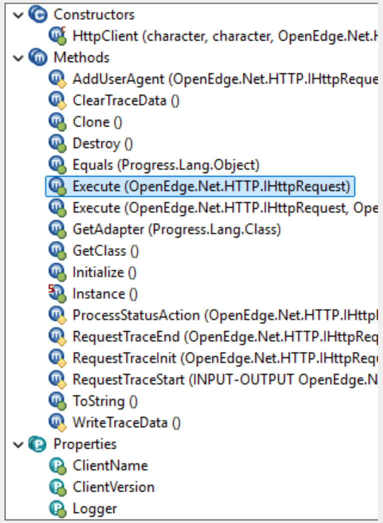
OpenEdge.Net.HTTP.HttpClient

`$DLC/[src|tty|gui]/netlib/OpenEdge.Net.pl`
also source code

`$DLC/[tty|gui]/netlib/OpenEdge.Net.apl`
signed program library



OpenEdge.Net
- OpenEdge.Mobile
- OpenEdge.Net
- OpenEdge.Net.HTTP
  - AuthenticatedRequest
  - AuthenticationMethodEnum
  - AuthenticationRequestEventArgs
  - AuthorizationHeader
  - AuthorizationHeaderBuilder
  - BuilderRegistry
  - ClientBuilder
  - ClientOptions
  - ConfigBuilder
  - ContentDispositionHeader
  - ContentDispositionHeaderBuilder
  - ContentTypeHeader
  - ContentTypeHeaderBuilder
  - Cookie
  - CookieCollection
  - CookieJar
  - CookieJarBuilder
  - CookieJarDecorator
  - Credentials
  - DefaultCookieJarBuilder
  - DefaultHeaderBuilder
  - DefaultHttpClientBuilder
  - DefaultRequestBuilder
  - DefaultResponseBuilder
  - HttpClient
  - HttpClientDecorator
  - HttpHeader
  - HttpHeaderBuilder
  - HttpHeaderCollection
  - HttpMessage
  - HttpRequest
  - HttpRequestDecorator
  - HttpRequestError
  - HttpResponse
  - HttpResponseDecorator
  - IAuthenticatedRequest
  - ICookieJar
  - IHttpClient
  - IHttpClientLibrary
  - IHttpMessage
  - IHttpMessageWriter

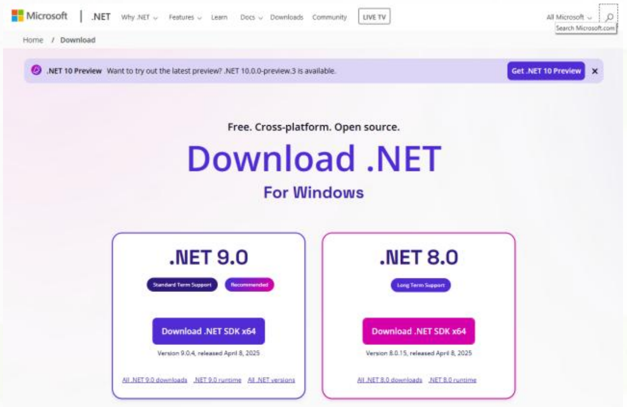


- Constructors
  - HttpClient (character, character, OpenEdge.Net.
- Methods
  - AddUserAgent (OpenEdge.Net.HTTP.IHttpReque
  - ClearTraceData ()
  - Clone ()
  - Destroy ()
  - Equals (Progress.Lang.Object)
  - Execute (OpenEdge.Net.HTTP.IHttpRequest)
  - Execute (OpenEdge.Net.HTTP.IHttpRequest, Ope
  - GetAdapter (Progress.Lang.Class)
  - GetClass ()
  - Initialize ()
  - Instance ()
  - ProcessStatusAction (OpenEdge.Net.HTTP.IHttp
  - RequestTraceEnd (OpenEdge.Net.HTTP.IHttpReq
  - RequestTraceInit (OpenEdge.Net.HTTP.IHttpReq
  - RequestTraceStart (INPUT-OUTPUT OpenEdge.N
  - ToString ()
  - WriteTraceData ()
- Properties
  - ClientName
  - ClientVersion
  - Logger

Add it to the propath!

.NET

https://dotnet.microsoft.com/en-us/download



.NET 10 Preview Want to try out the latest preview? .NET 10.0.0-preview.3 is available.

Free. Cross-platform. Open source.

Download .NET

For Windows

.NET 9.0

Standard Term Support · Recommended

Download .NET SDK x64

Version 9.0.4, released April 8, 2025

All .NET 9.0 downloads · .NET 9.0 runtime · All .NET versions

.NET 8.0

Long Term Support

Download .NET SDK x64

Version 8.0.15, released April 8, 2025

All .NET 8.0 downloads · .NET 8.0 runtime

- .NET Framework 3.5 (†)
- .NET Framework 4.0 (†)
- .NET Framework 4.8
- .NET Core 3.1 (†)
- .NET 6.0 (†)
- .NET 8.0

How to get started

- Install OpenEdge 12.8 +
- Update propath

  `OpenEdge.Net.(a)pl`
- Install .NET 8.0
- Update in $DLC/bin

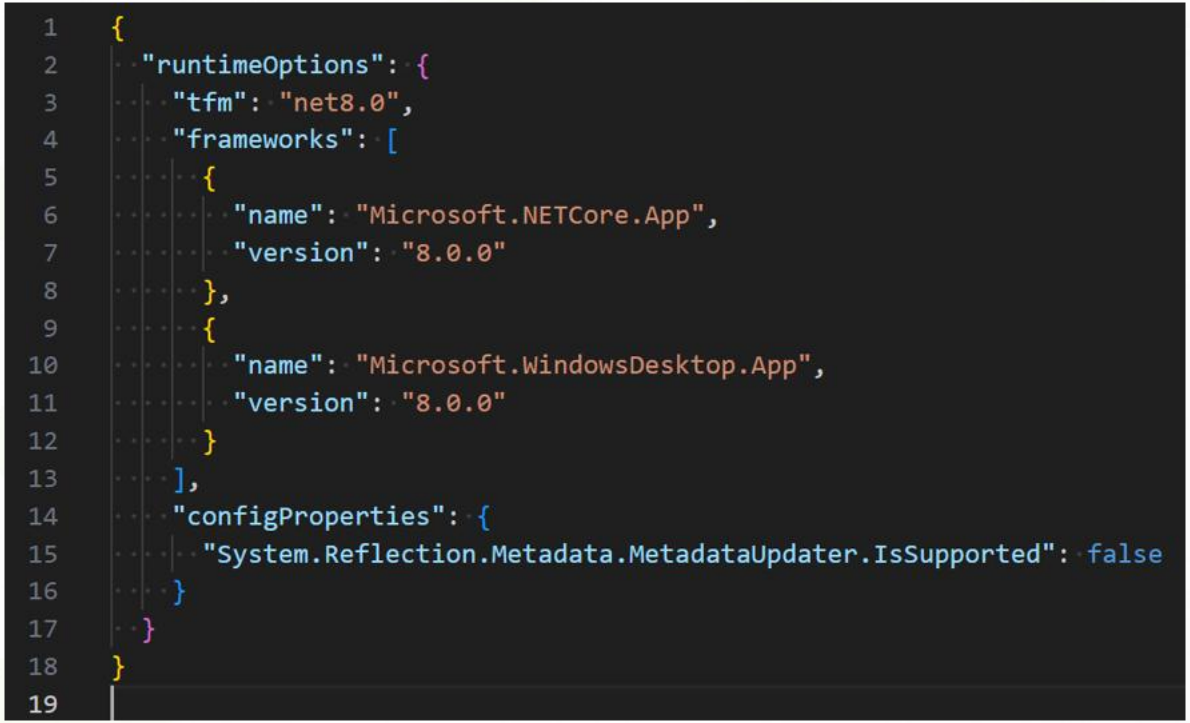
  `Progress.clrbridge.netcore.runtimeconfig.json`
- -clrnetcore (session parameter)
- -assemblies
- add assemblies.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"
<references>
   <assembly name="System, Version=4.0.0.0, Culture=
</references>
```

flusso IT

Windows

`%DLC%\bin\Progress.clrbridge.netcore.runtimeconfig.json`

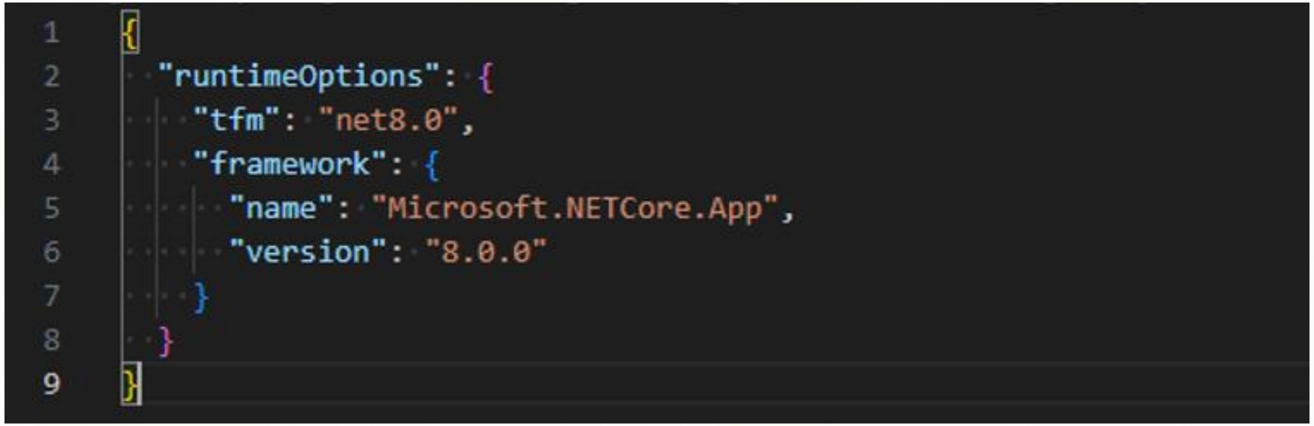


```json
{
  "runtimeOptions": {
    "tfm": "net8.0",
    "frameworks": [
      {
        "name": "Microsoft.NETCore.App",
        "version": "8.0.0"
      },
      {
        "name": "Microsoft.WindowsDesktop.App",
        "version": "8.0.0"
      }
    ],
    "configProperties": {
      "System.Reflection.Metadata.MetadataUpdater.IsSupported": false
    }
  }
}

```

flusso IT

Linux

`$DLC/bin/Progress.clrbridge.netcore.runtimeconfig.json`



```json
{
  "runtimeOptions": {
    "tfm": "net8.0",
    "framework": {
      "name": "Microsoft.NETCore.App",
      "version": "8.0.0"
    }
  }
}
```

- install .NET 8
- docker image as alternative

  `devbfvio/openedge-compiler:12.8.9-dotnet8`

flusso IT

Goal

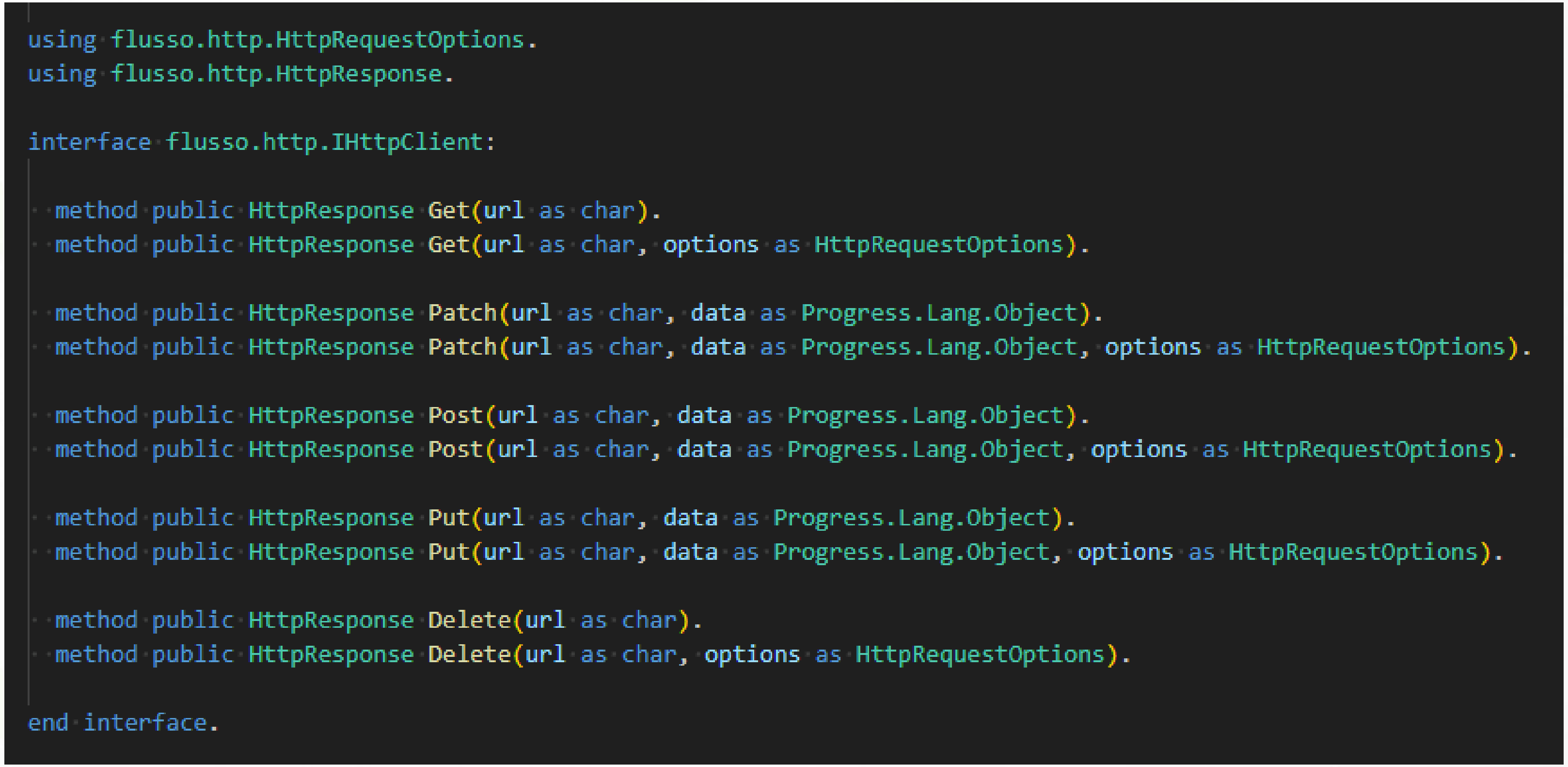
Provide solution for OpenEdge developers to easily switch between ABL and .NET HttpClient indepent of OS

- hide implementation
- HttpClient based on configuration
- simple interface
- support all common use cases

Tools

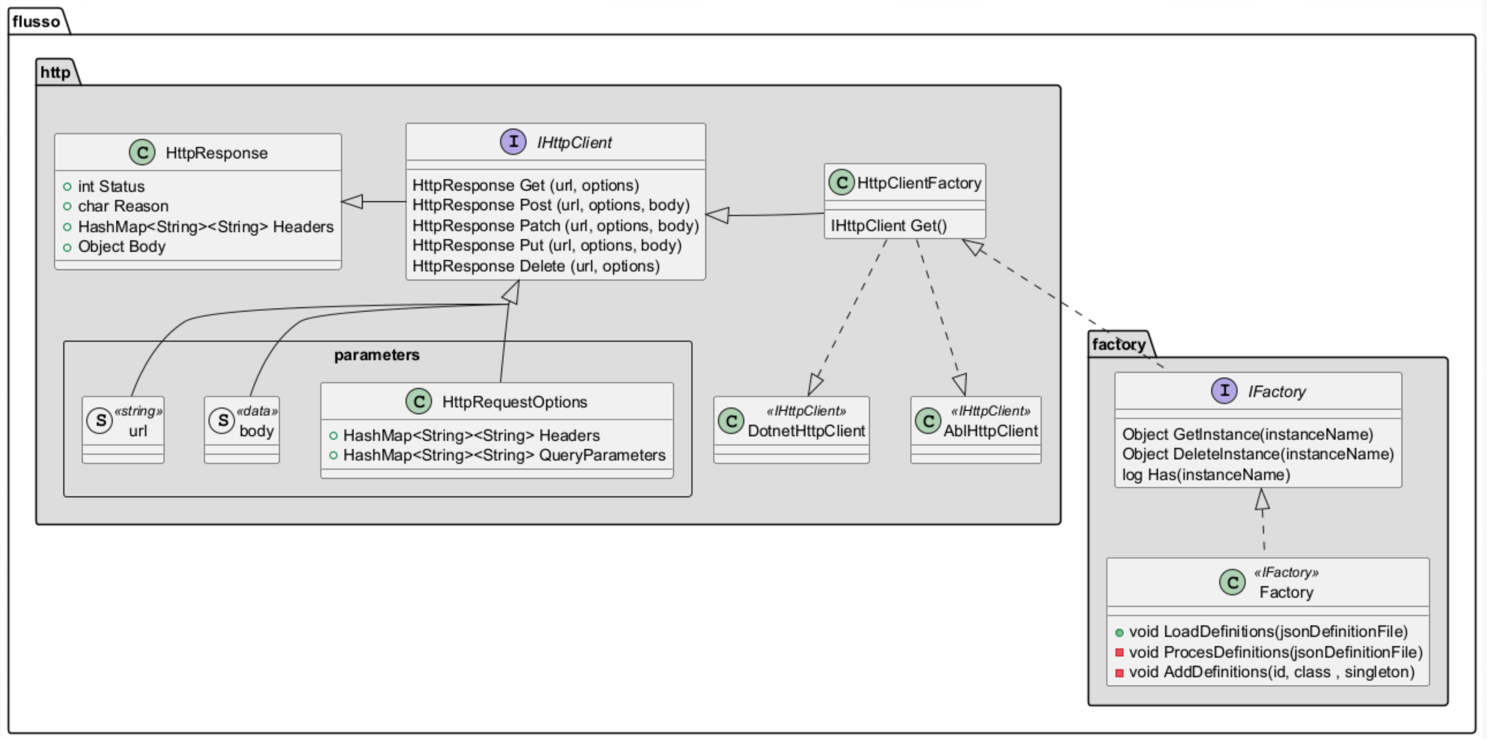
- DeveloperStudio
- vscode
- plantuml
- docker for Linux
- petstore container
- github for code
- (unittest)
- patterns

Interface



```
using flusso.http.HttpRequestOptions.
using flusso.http.HttpResponse.

interface flusso.http.IHttpClient:

  method public HttpResponse Get(url as char).
  method public HttpResponse Get(url as char, options as HttpRequestOptions).

  method public HttpResponse Patch(url as char, data as Progress.Lang.Object).
  method public HttpResponse Patch(url as char, data as Progress.Lang.Object, options as HttpRequestOptions).

  method public HttpResponse Post(url as char, data as Progress.Lang.Object).
  method public HttpResponse Post(url as char, data as Progress.Lang.Object, options as HttpRequestOptions).

  method public HttpResponse Put(url as char, data as Progress.Lang.Object).
  method public HttpResponse Put(url as char, data as Progress.Lang.Object, options as HttpRequestOptions).

  method public HttpResponse Delete(url as char).
  method public HttpResponse Delete(url as char, options as HttpRequestOptions).

end interface.
```

Class diagram



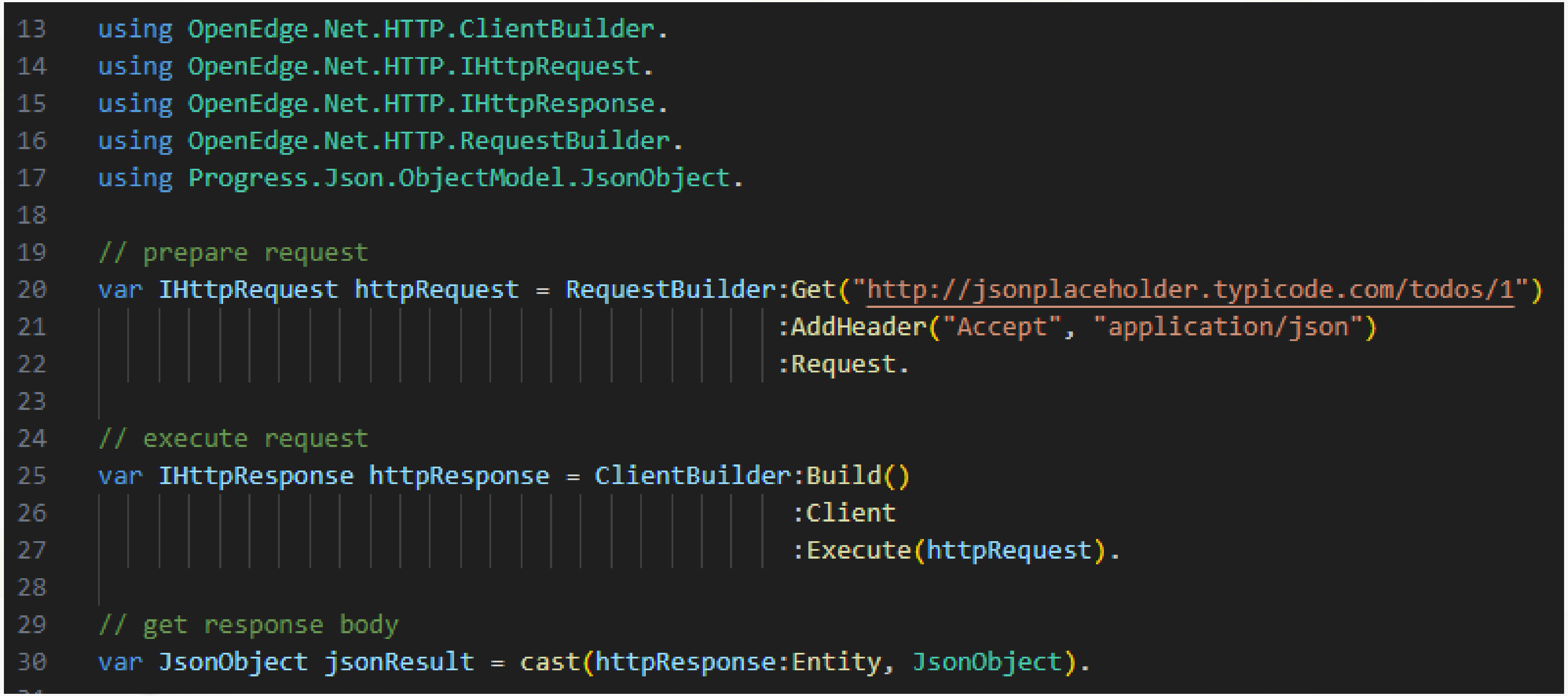
flusso
http
HttpResponse
int Status
char Reason
HashMap<String><String> Headers
Object Body
IHttpClient
HttpResponse Get (url, options)
HttpResponse Post (url, options, body)
HttpResponse Patch (url, options, body)
HttpResponse Put (url, options, body)
HttpResponse Delete (url, options)
HttpClientFactory
IHttpClient Get()
parameters
«string» url
«data» body
HttpRequestOptions
HashMap<String><String> Headers
HashMap<String><String> QueryParameters
«IHttpClient» DotnetHttpClient
«IHttpClient» AblHttpClient
factory
IFactory
Object GetInstance(instanceName)
Object DeleteInstance(instanceName)
log Has(instanceName)
«IFactory» Factory
void LoadDefinitions(jsonDefinitionFile)
void ProcesDefinitions(jsonDefinitionFile)
void AddDefinitions(id, class , singleton)

flusso IT

Solution

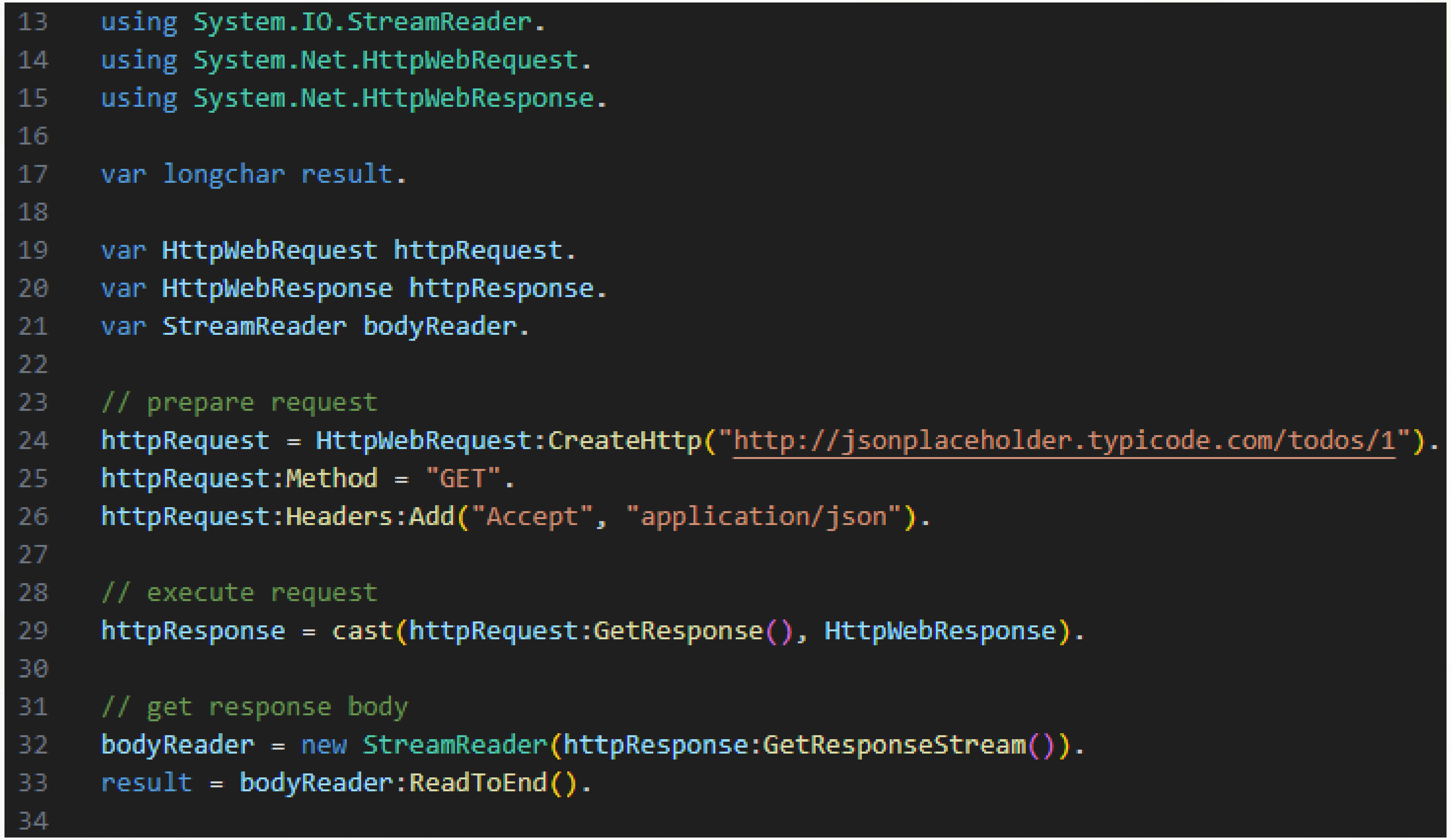
- Code examples of both HttpClients
- Patterns
- Demo
- Summary

OpenEdge HttpClient example



```
using OpenEdge.Net.HTTP.ClientBuilder.
using OpenEdge.Net.HTTP.IHttpRequest.
using OpenEdge.Net.HTTP.IHttpResponse.
using OpenEdge.Net.HTTP.RequestBuilder.
using Progress.Json.ObjectModel.JsonObject.

// prepare request
var IHttpRequest httpRequest = RequestBuilder:Get("http://jsonplaceholder.typicode.com/todos/1")
                                              :AddHeader("Accept", "application/json")
                                              :Request.

// execute request
var IHttpResponse httpResponse = ClientBuilder:Build()
                                               :Client
                                               :Execute(httpRequest).

// get response body
var JsonObject jsonResult = cast(httpResponse:Entity, JsonObject).
```

DotNet HttpClient example



```
using System.IO.StreamReader.
using System.Net.HttpWebRequest.
using System.Net.HttpWebResponse.

var longchar result.

var HttpWebRequest httpRequest.
var HttpWebResponse httpResponse.
var StreamReader bodyReader.

// prepare request
httpRequest = HttpWebRequest:CreateHttp("http://jsonplaceholder.typicode.com/todos/1").
httpRequest:Method = "GET".
httpRequest:Headers:Add("Accept", "application/json").

// execute request
httpResponse = cast(httpRequest:GetResponse(), HttpWebResponse).

// get response body
bodyReader = new StreamReader(httpResponse:GetResponseStream()).
result = bodyReader:ReadToEnd().

```

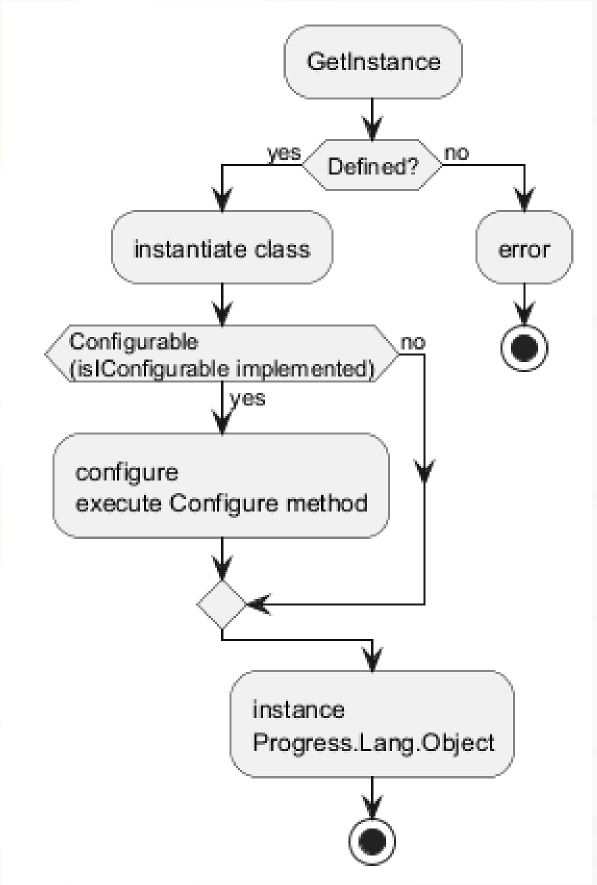
Facade pattern

- Simple interface to a complex subsystem
  - Hide complexity
  - Reduces dependencies
  - Centralizing access
- Every project has them
  - `*Wrapper` `*Service`
  - `*Helper` `*Facade`
  - `*Manager`
- `AblHttpClient` / `DotNetHttpClient`

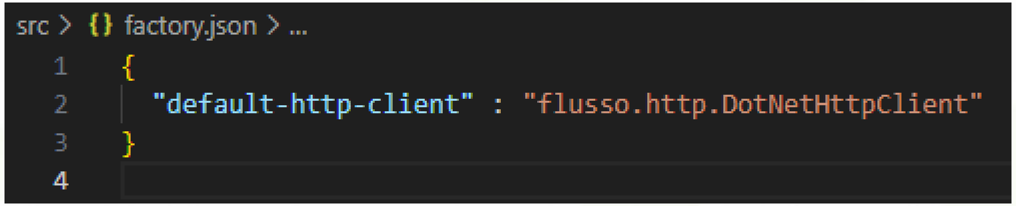
flusso IT

Factory pattern

- Create object without knowing the exact class
- Abstracts the real implementation
- Loosely coupled code
- Can be config-driven (i.e. `factory.json`)
  - Set context properties (`IConfigurable`)
- Easy to:
  - Extend (i.e. `JavaHttpClient`)
  - Mock in unit tests

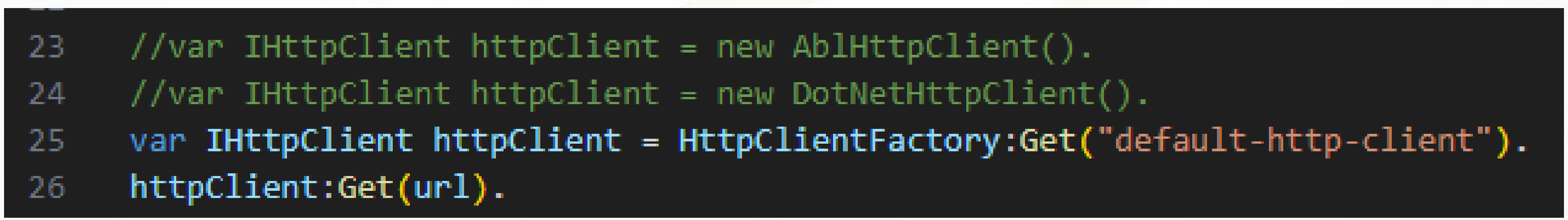


Factory pattern



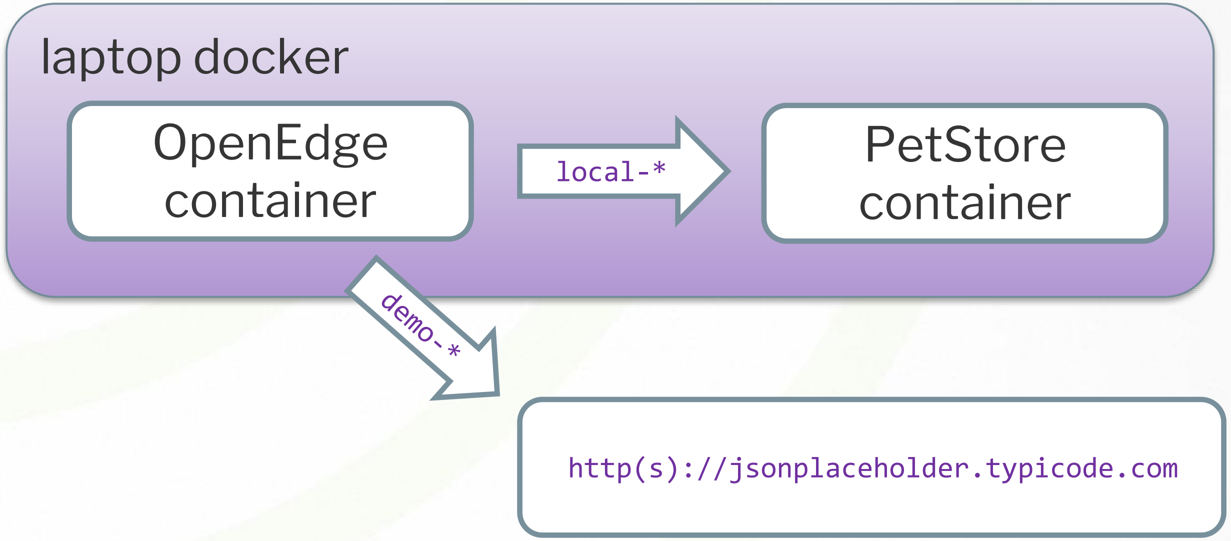
src > {} factory.json > ...

```json
{
  "default-http-client" : "flusso.http.DotNetHttpClient"
}

```



```
//var IHttpClient httpClient = new AblHttpClient().
//var IHttpClient httpClient = new DotNetHttpClient().
var IHttpClient httpClient = HttpClientFactory:Get("default-http-client").
httpClient:Get(url).
```

Demo

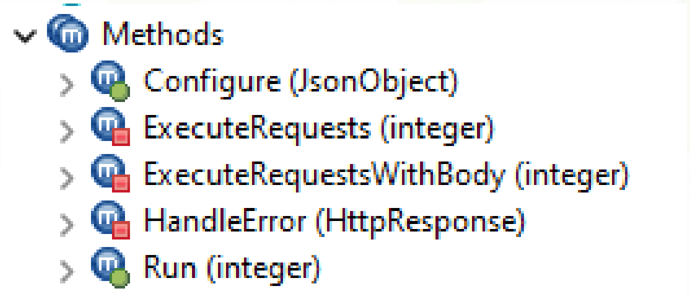


- Using 2 docker containers locally
  - Linux with OpenEdge
  - Petstore application with caddy webserver
- Measure performance
- `DemoRunner` (factory)

DemoRunner

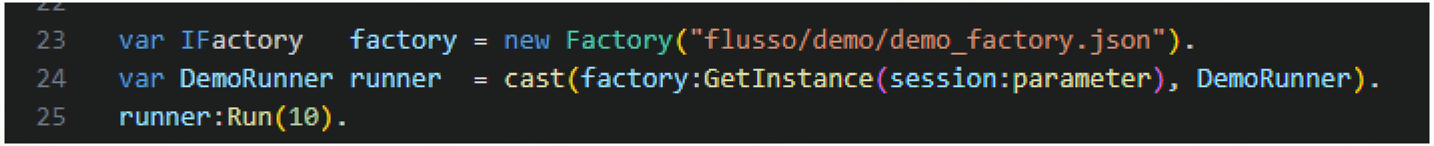
- DemoRunner

  `implements IConfigurable`



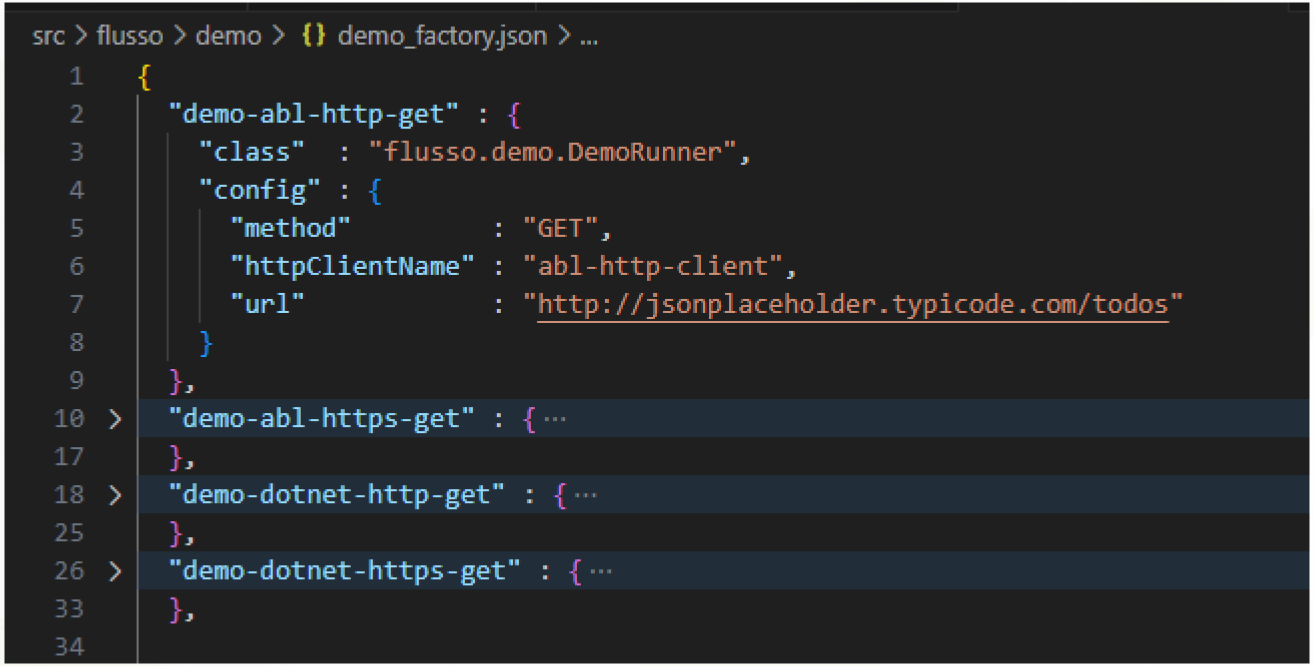
Methods
- Configure (JsonObject)
- ExecuteRequests (integer)
- ExecuteRequestsWithBody (integer)
- HandleError (HttpResponse)
- Run (integer)

- Run via command-line: `demo.sh` / `demo.bat` *`identifier`* `[nrRuns]`
- Factory



```
var IFactory   factory = new Factory("flusso/demo/demo_factory.json").
var DemoRunner runner  = cast(factory:GetInstance(session:parameter), DemoRunner).
runner:Run(10).
```

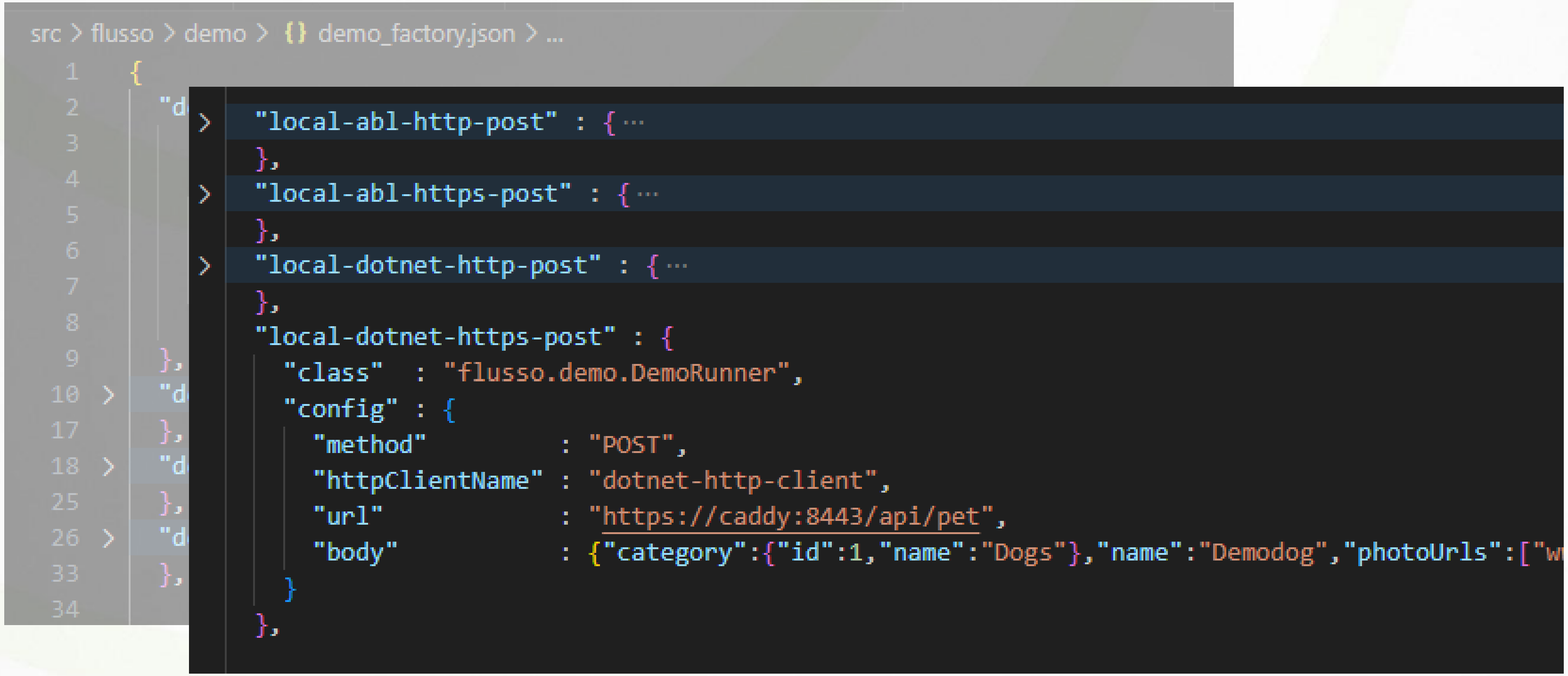
demo_factory.json



src > flusso > demo > {} demo_factory.json > ...

```
{
  "demo-abl-http-get" : {
    "class"  : "flusso.demo.DemoRunner",
    "config" : {
      "method"          : "GET",
      "httpClientName" : "abl-http-client",
      "url"             : "http://jsonplaceholder.typicode.com/todos"
    }
  },
  "demo-abl-https-get" : {⋯
  },
  "demo-dotnet-http-get" : {⋯
  },
  "demo-dotnet-https-get" : {⋯
  },

```

demo_factory.json



src > flusso > demo > {} demo_factory.json > ...

```
{
    "d
        },
        "d
        },
        "d
        },
        "d
        },
```

```
"local-abl-http-post" : {⋯
},
"local-abl-https-post" : {⋯
},
"local-dotnet-http-post" : {⋯
},
"local-dotnet-https-post" : {
  "class"  : "flusso.demo.DemoRunner",
  "config" : {
    "method"         : "POST",
    "httpClientName" : "dotnet-http-client",
    "url"            : "https://caddy:8443/api/pet",
    "body"           : {"category":{"id":1,"name":"Dogs"},"name":"Demodog","photoUrls":["w
  }
},
```

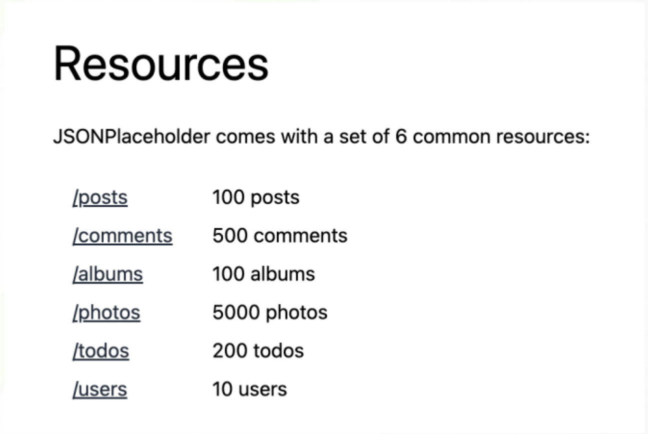
flusso IT

Demo

- Inside a docker container (Linux Ubuntu, 12.8.9, .NET 8.0.13):
    - ABL/.NET: **`http - GET`**
    - ABL/.NET: **`https - GET`**
    - ABL/.NET: **`https - POST`**
    - (ABL/.NET: **`https - DELETE)`**

Results

- Address used: http(s)://jsonplaceholder.typicode.com/todos
    - Dummy endpoints
- Linux/Windows:
    - GET http & https
    - POST http & https
    - DELETE http & https

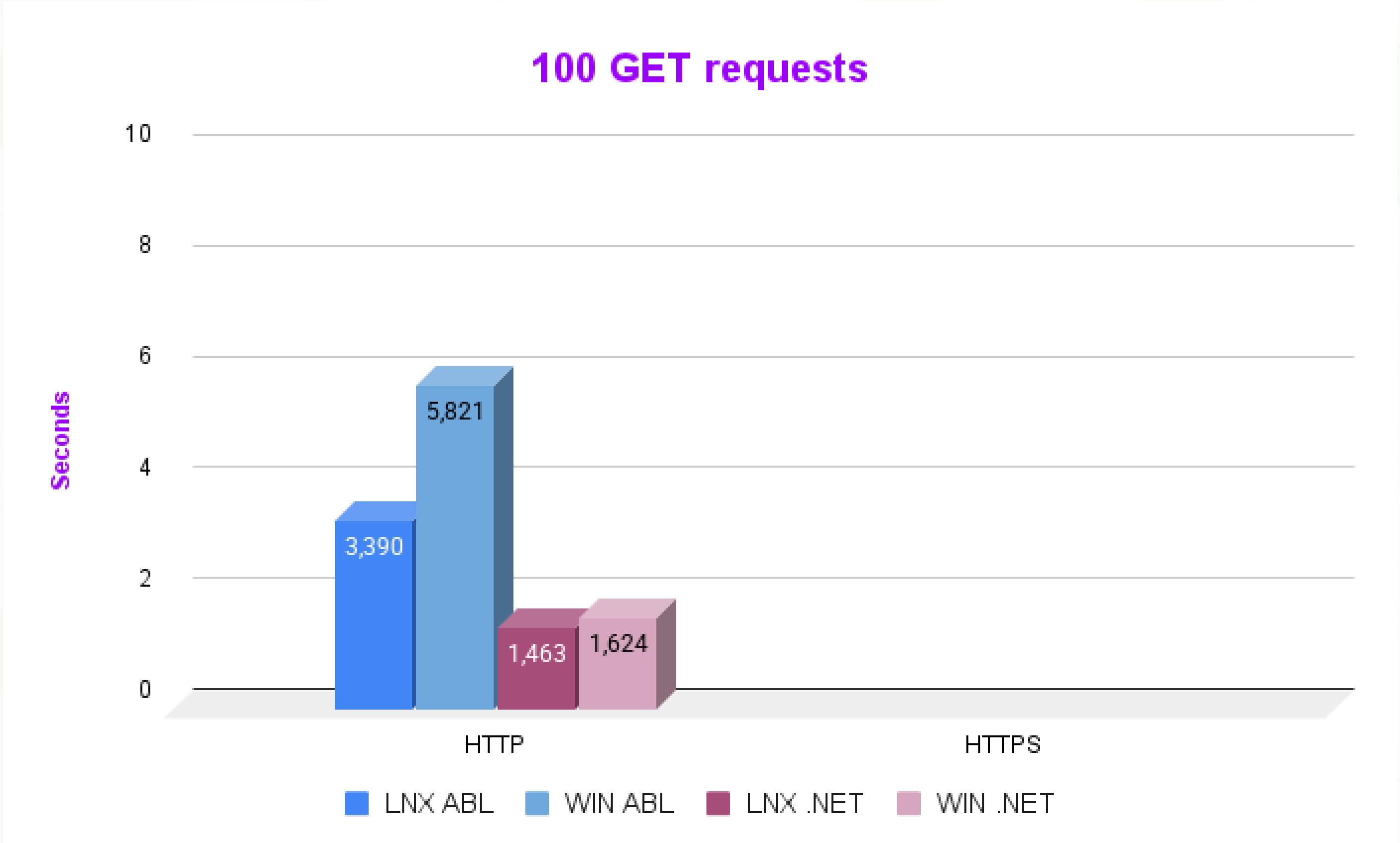


Resources

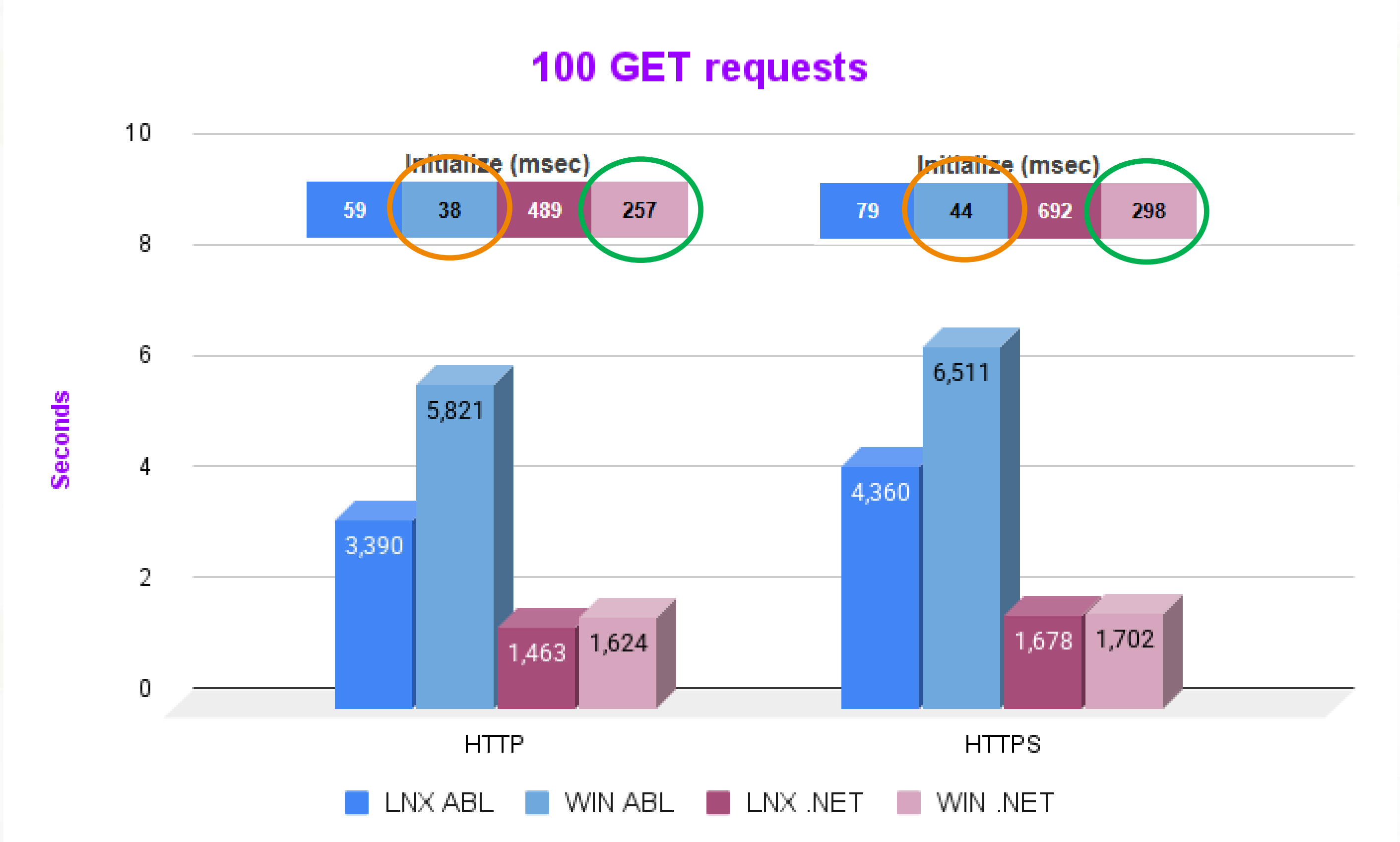
JSONPlaceholder comes with a set of 6 common resources:

/posts	100 posts
/comments	500 comments
/albums	100 albums
/photos	5000 photos
/todos	200 todos
/users	10 users

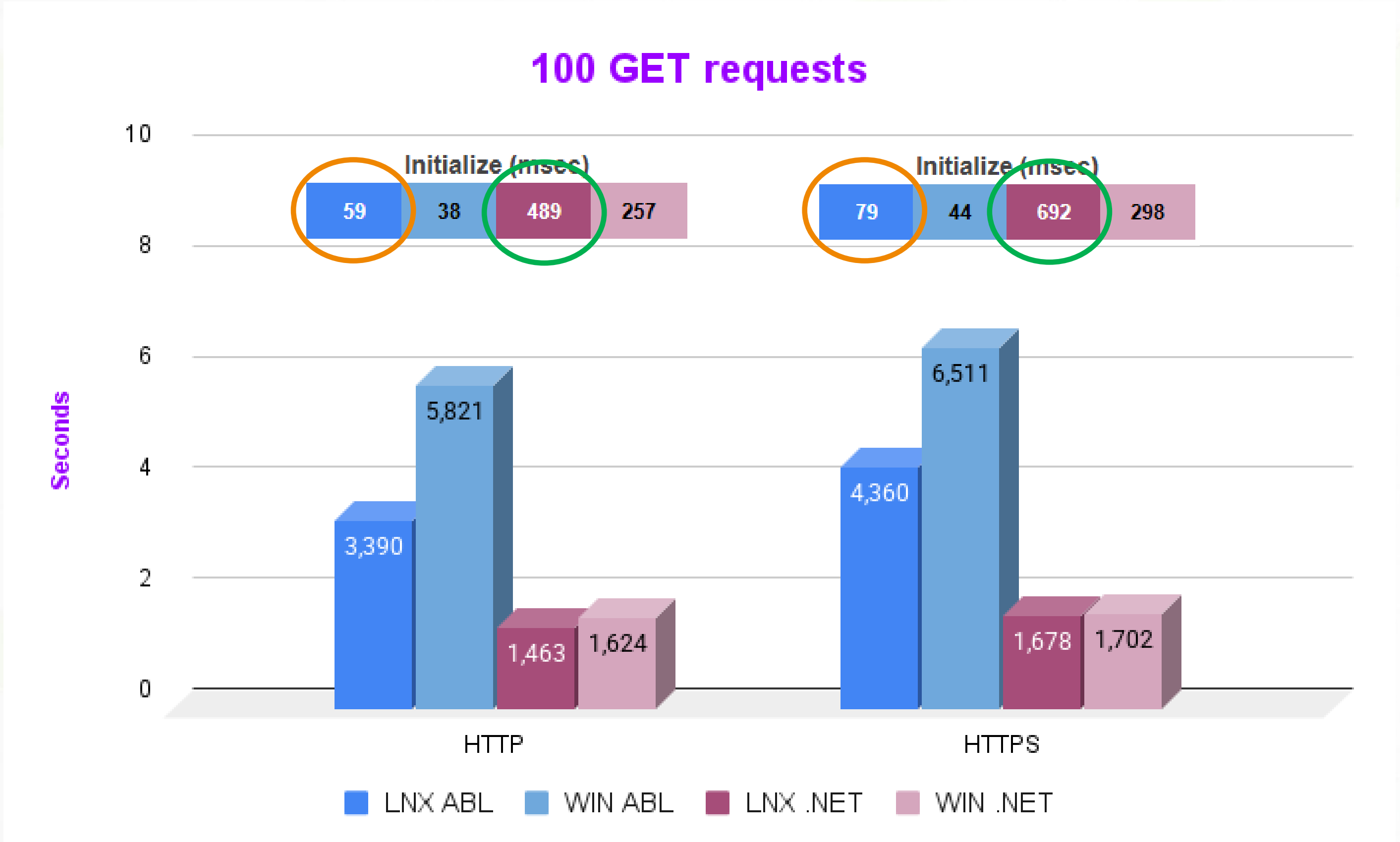
Demo summary : Performance GET http



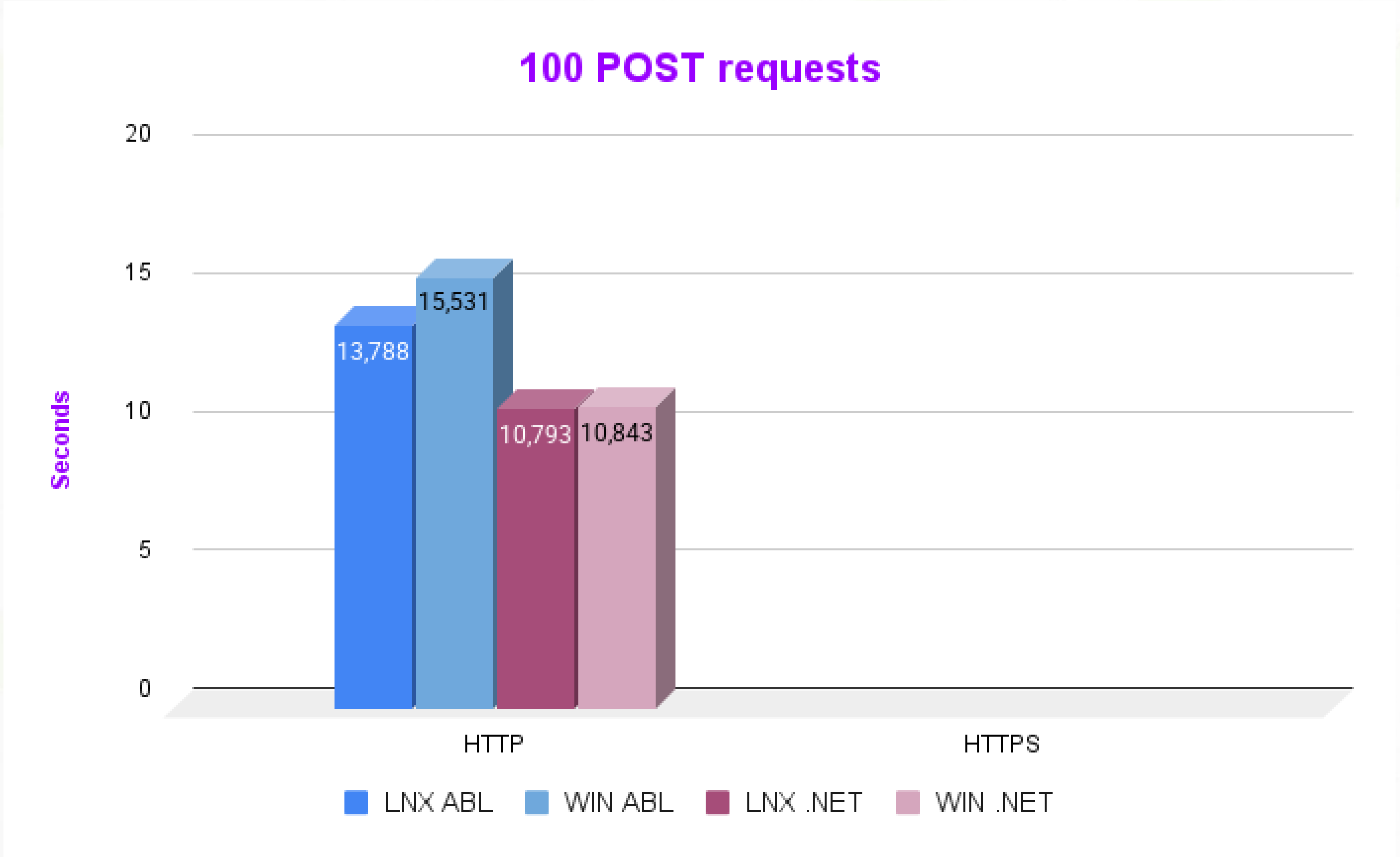
Demo summary : Performance GET http/https



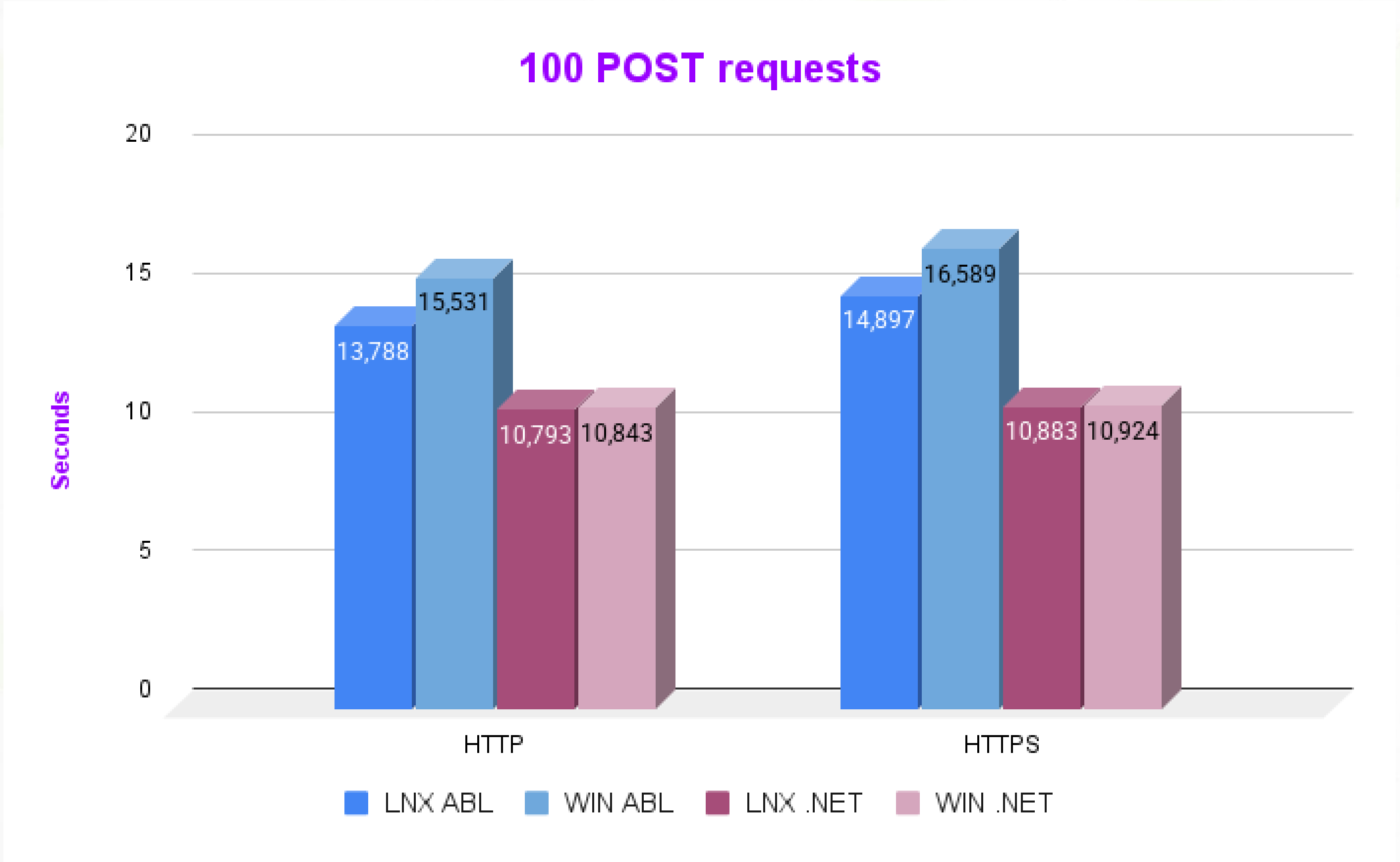
Demo summary : Performance GET http/https



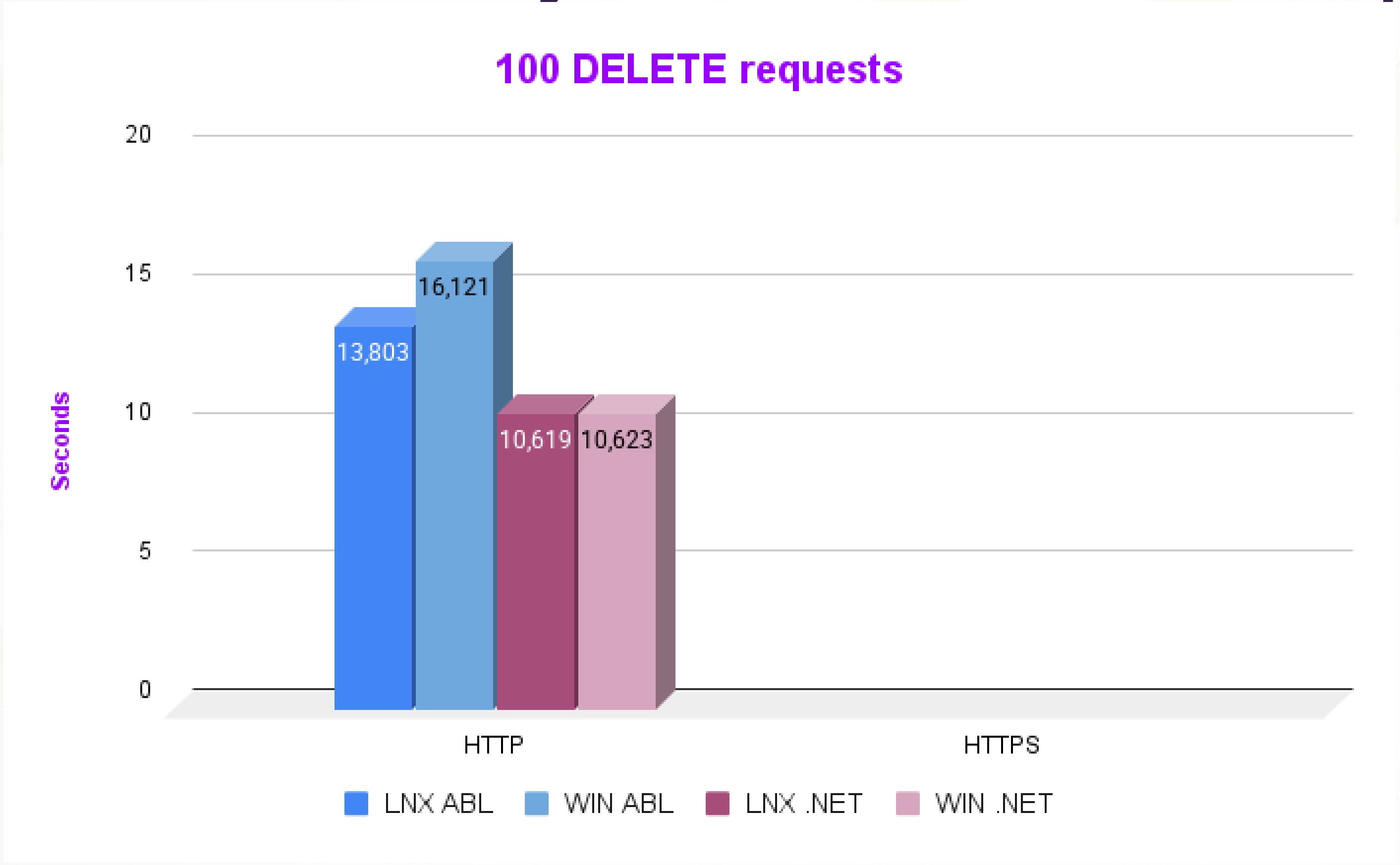
Demo summary : Performance POST http



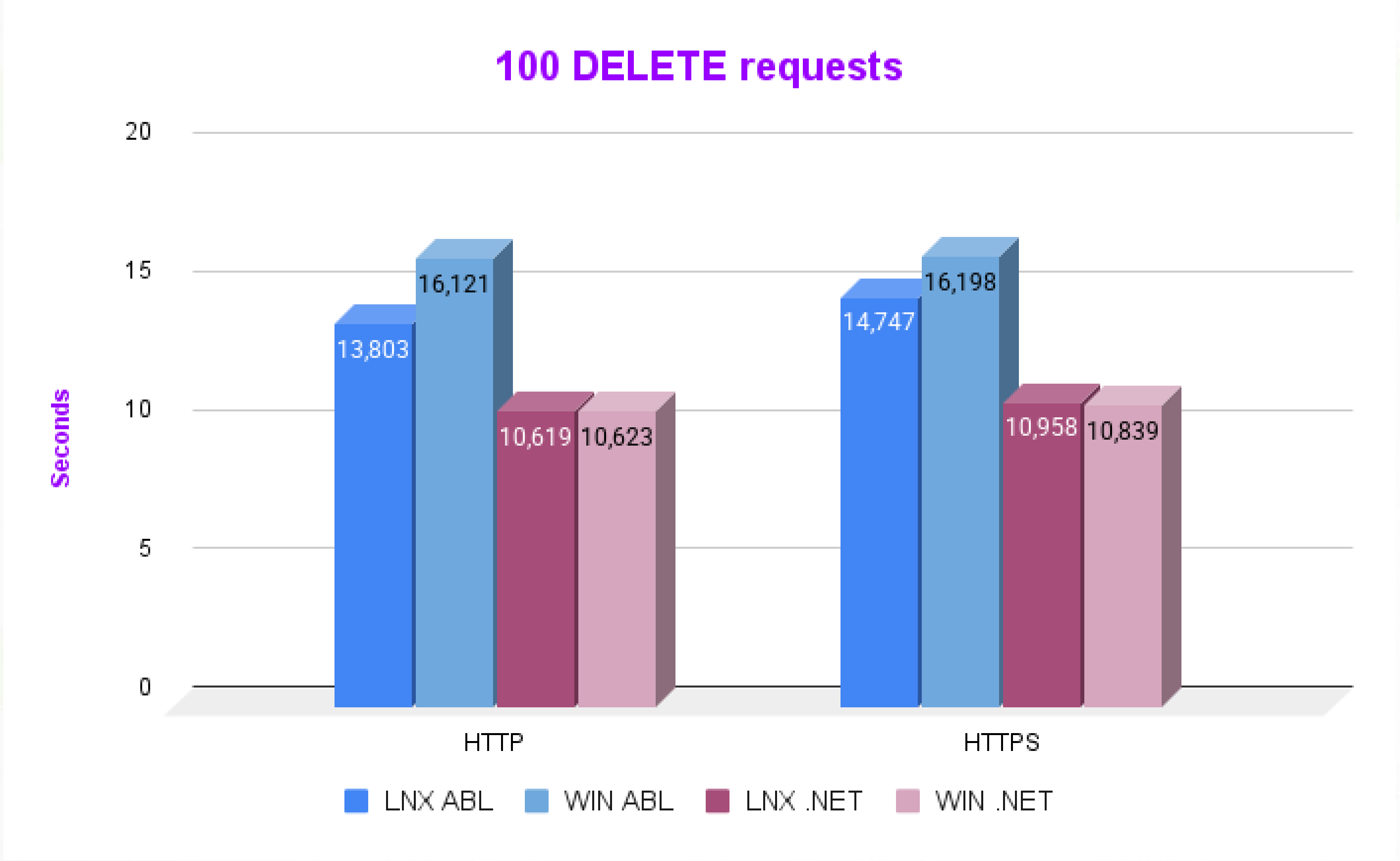
Demo summary : Performance POST http/https



Demo summary : Performance DEL http



Demo summary : Performance DEL http/https



Results - summary

Performance differences:

- .NET calls on average significant faster especially GETS
- Little to no difference overall between HTTP and HTTPS
- Between Linux and Windows, the AblHttpClient is slower on Windows
- Initialize of .NET overhead bigger
    - even more for HTTPS (especially on Linux)

Difficulties

- `-preloadCLR` fails on Linux (still under investigation)
- Convert .NET response to ABL object
- Multipart messages not implemented yet
- Assemblies advised for Linux and vscode
- ssl complexity with certificates

Certificates

OpenEdge

- use `certutil` from proenv
- copy certificate to `$DLC/certs`

.NET

- update local certificates
- copy certificate to `/etc/ssl/certs`

Demo

- Add public key from petstore container to OpenEdge container
  - `/etc/ssl/certs`
  - `$DLC/certs`

Summary

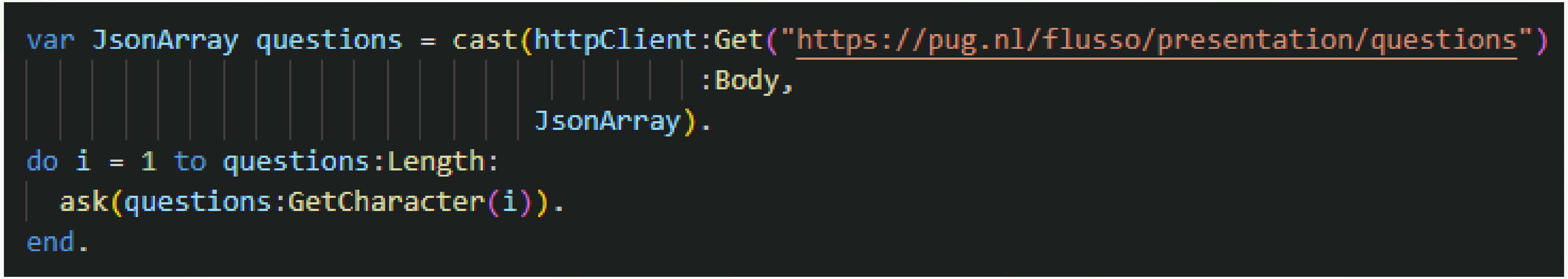
- Use case for the HttpClient
- Quick overview of the Http protocol
- History of OpenEdge and .NET
- Installation
- Our solution and tools
- Design patterns
- Demo
- Difficulties and open issues
- https://github.com/FlussoBV/HttpClientForOpenEdge

flusso IT

See other presentations on .NET

- The OpenEdge You Know, Now Even Better: Dive Into New Possibilities in ABL by Ruben Dröge Wednesday 13:45
- Seamless Integration of Word & Excel within ABL by Martyn Kemp Thursday 15:00
- The Dot Net Library by Bob Brennan Thursday 13:35

Questions



```
var JsonArray questions = cast(httpClient:Get("https://pug.nl/flusso/presentation/questions")
                                         :Body,
                               JsonArray).
do i = 1 to questions:Length:
  ask(questions:GetCharacter(i)).
end.
```