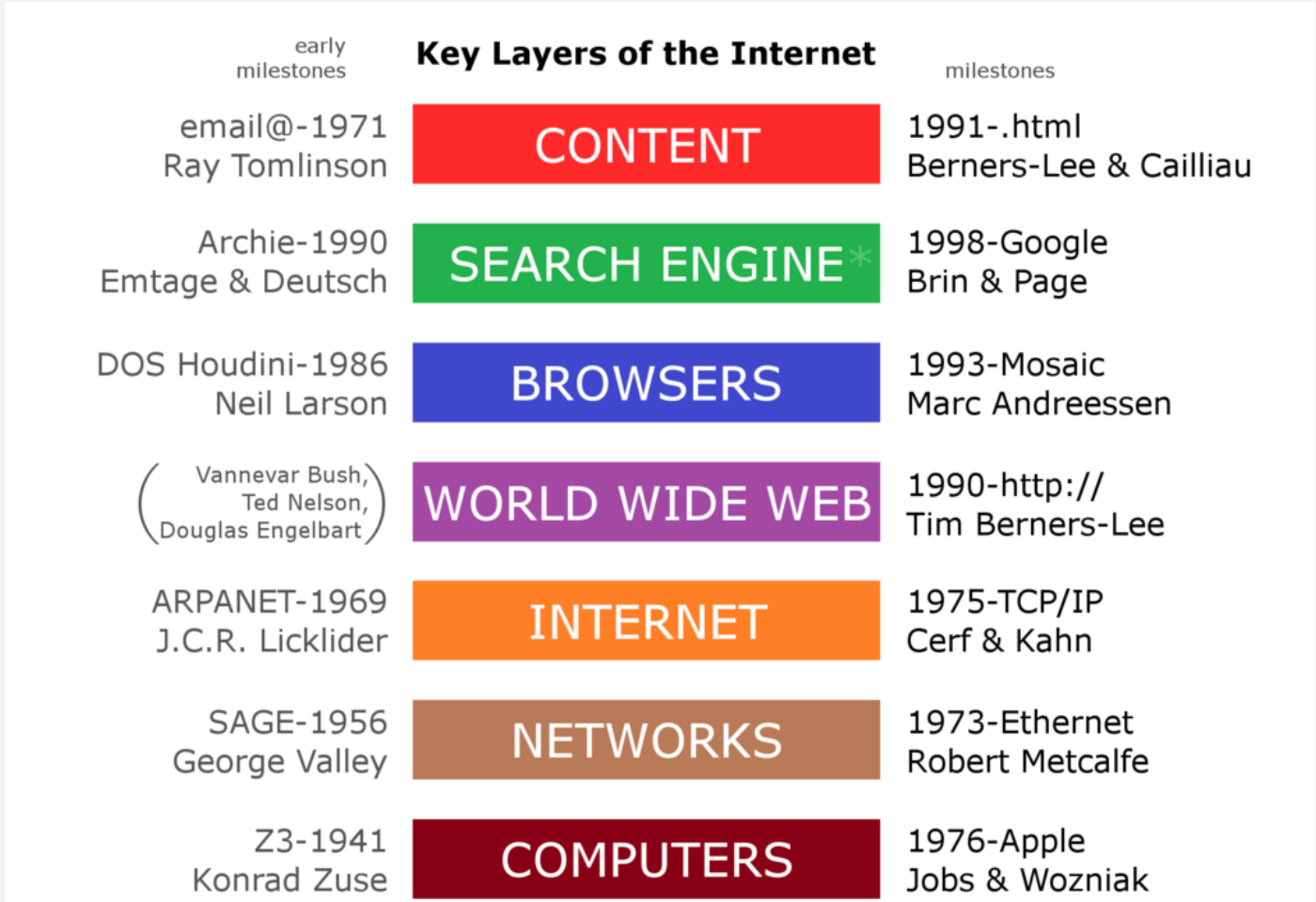
The world is your oyster

Calling REST services from ABL

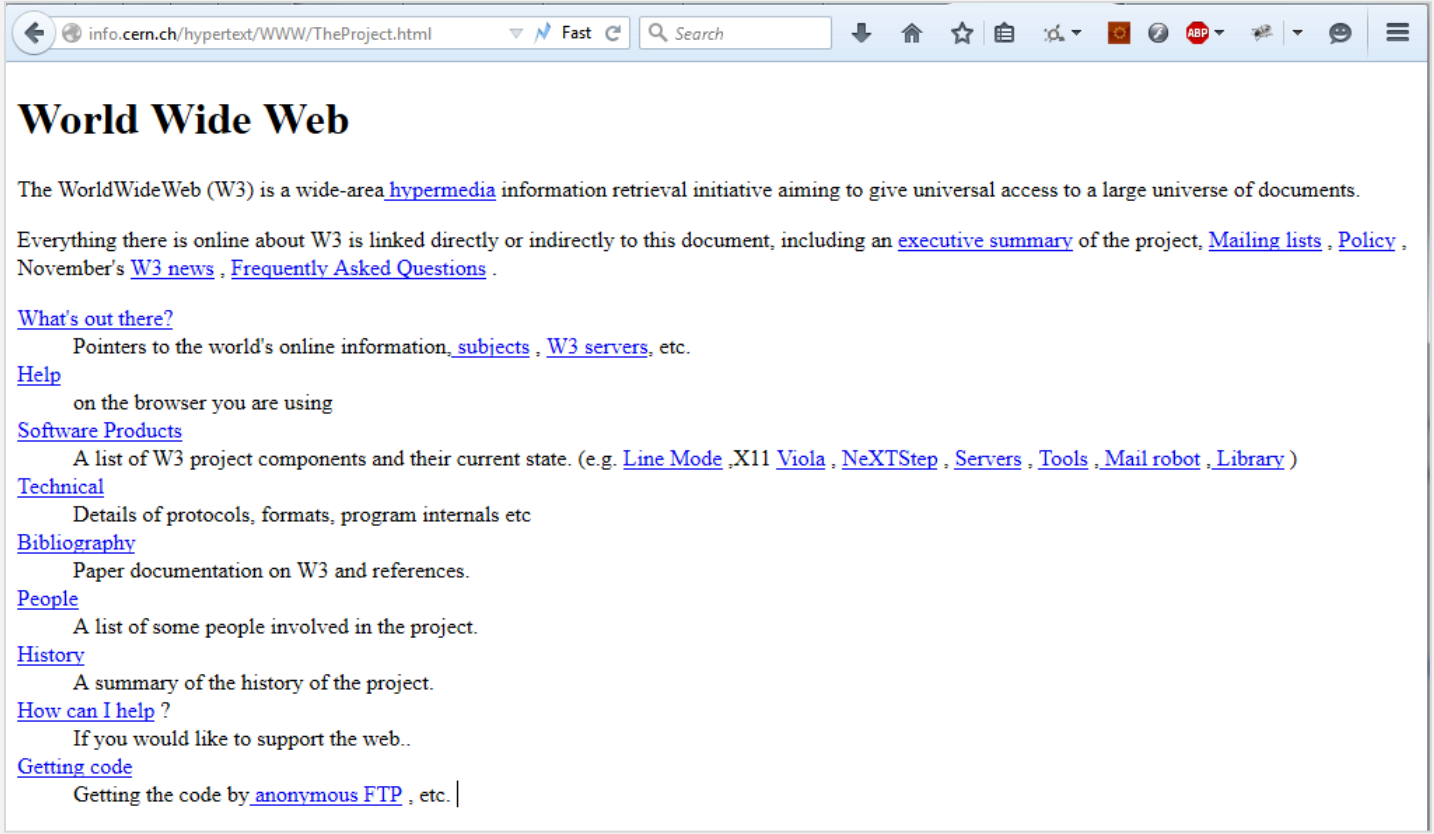
Peter Judge
pjudge@progress.com

PUG CHALLENGE EXCHANGE AMERICAS

Once upon a time …



The very first web page (still there)



So. Many. RFCs.

- http://tools.ietf.org/html/rfc2616
  - And many, many, many, many friends
  - Obsoleted now by http://tools.ietf.org/html/rfc7230
- These RFCs define the HTTP protocol and supporting structures
  - URIs
  - Media types
  - Authorisation schemes
  - Cookies

Flow of HTTP


```
request = request-line ;
*(( general-header ; | request-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

REQUEST

Flow of HTTP


```
request = request-line ;
*(( general-header ; | request-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

REQUEST

```
POST /oemanager/applications/webtest/props HTTP/1.1 ↵
```

Flow of HTTP


```
request = request-line ;
*(( general-header ; | request-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

REQUEST

```
POST /oemanager/applications/webtest/props HTTP/1.1 ↵
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Host: oelxdev06:8881
User-Agent: OpenEdge-HttpClient/0.3.0
Content-Length: 18 ↵
↵
```

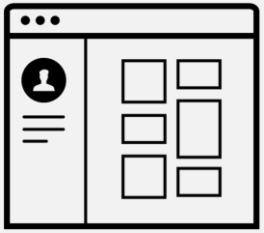
Flow of HTTP


```
request = request-line ;
*(( general-header ; | request-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

REQUEST

```
POST /oemanager/applications/webtest/props HTTP/1.1 ↵
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Host: oelxdev06:8881
User-Agent: OpenEdge-HttpClient/0.3.0
Content-Length: 18 ↵
↵
{"request": "data"}
```

Flow of HTTP

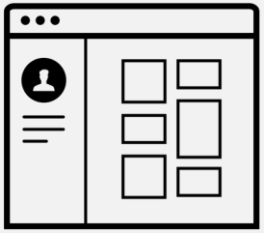


```
response = status-line ;
*(( general-header ; | response-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

RESPONSE

HTTP/1.1 200 OK ↵

Flow of HTTP

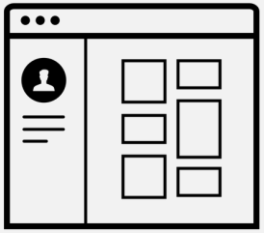


```
response = status-line ;
*(( general-header ; | response-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

RESPONSE

```
HTTP/1.1 200 OK ↵
Server: Apache-Coyote/1.1
Date: Tue, 24 Mar 2015 18:31:29 GMT
Content-Type: application/json
Content-Length: 19 ↵
↵
```

Flow of HTTP



```
response = status-line ;
*(( general-header ; | response-header ; |
entity-header ) CRLF) ; CRLF
[ message-body ] ;
```

RESPONSE

```
HTTP/1.1 200 OK ↵
Server: Apache-Coyote/1.1
Date: Tue, 24 Mar 2015 18:31:29 GMT
Content-Type: application/json
Content-Length: 19 ↵
↵
{"result": "data"}
```

JSON

JSON – http://json.org


```
{ "result":
  { "applications": ["webtest", "abltest"],
    "numInitialAgents": "1",
    "collectMetrics": "1",
    "agentExecFile": "/usr1/dlc/bin/_mproapsv",
    "maxAgents": "10",
    "agentStartupParam": "-T /usr1/wrk/webtest/temp",
    "maxABLSessionsPerAgent": "200",
  },
  "outcome": "SUCCESS",
  "errmsg": "",
  "versionStr": "PASOE 11.5.0",
  "versionNo": 1,
  "operation": "GET BROKER PROPERTIES"
}
```

What is REST?

- REST is an architecture style for designing networked applications
- **Client-Server**: pull- or request-based
- **Stateless**: each request must contain all information necessary to understand the request
- **Cacheable**
- **Uniform interface**: all resources accessed via generic interface (GET/PUT/POST/DELETE)
- **Named resources**: resources are named using a URL
- **Interconnected resource representations**: the representations of the resources are interconnected using URLs
- **Layered components** : intermediaries can be inserted between clients and resources to support performance, security, etc.

http://rest.elkstein.org/2008/02/what-is-rest.html and http://www.xfront.com/REST-Web-Services.html

What is REST?

- Practically, JSON payloads to and fro over HTTP, using a particular style of URL
- JSON content types

  ```
  application/json
  application/vnd.[vendor]+json
  ```

- Response contains URLs to other resources

REST Example: request

```
GET /users/nwahmaet HTTP/1.1
Host: api.github.com
Accept: application/vnd.github.v3+json
User-Agent: OpenEdge-HttpClient/0.3.0
```

REST Example: response

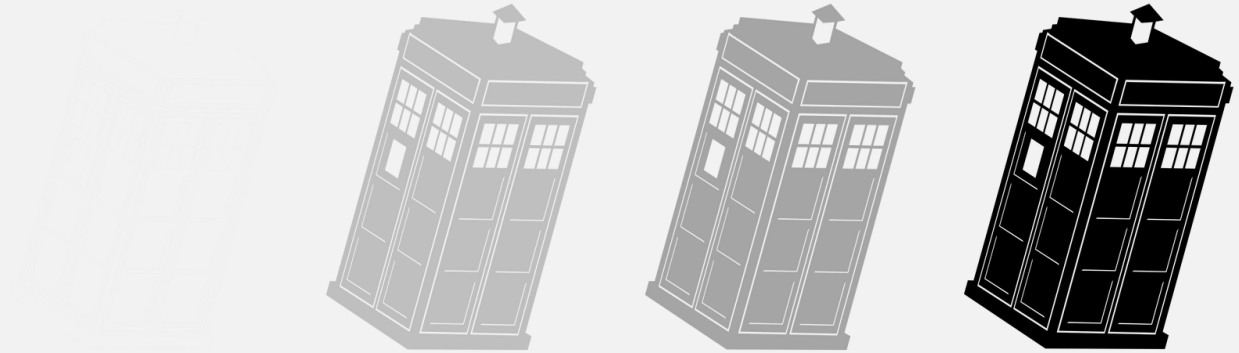
```
HTTP/1.1 200 OK
Cache-Control: public, max-age=60, s-maxage=60
Content-Encoding: gzip
Content-Type: application/json; charset=utf-8
Server: GitHub.com

{ "login": "nwahmaet",
  "id": 2736095,
  "url": "https://api.github.com/users/nwahmaet",
  "html_url": "https://github.com/nwahmaet",
  "repos_url": "https://api.github.com/users/nwahmaet/repos",
  "type": "User",
  "hireable": true,
  "created_at": "2012-11-06T15:35:09Z",
  "updated_at": "2015-03-04T20:44:24Z"
}
```

Cool! How do I do that in ABL?

Client socket support (v9.1A)

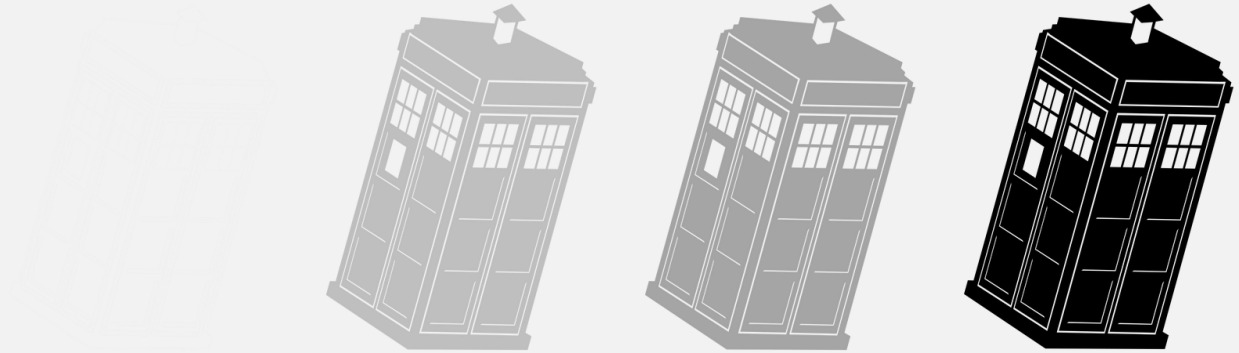
Cool! How do I do that in ABL?

Client socket support (v9.1A)



Cool! How do I do that in ABL?

Client socket support (v9.1A)



OpenEdge.Net.HTTP.* (11.5.1)

OpenEdge.Net.HTTP.*

- A class library that provides support for HTTP(S)
  - Designed for API use
  - HttpClient, URI, HttpHeader, Cookie, HttpRequest, HttpResponse
  - Supports much of HTTP 1.1 spec
- Simple, extensible programming interface
- Platform-portable (built on ABL sockets)

- Foundational tech for Mobile Push Notifications in 11.4.0
- Formally QA'd, supported in 11.5.1

Using OpenEdge.Net.HTTP

1. Create Client
2. Create URI
3. Create Request
4. Execute Request
5. Process Response
6. Profit

```
using OpenEdge.Net.HTTP.*.
def var oClient as IHttpClient.
def var oRequest as IHttpRequest.
def var oResponse as IHttpResponse.

oClient = ClientBuilder:Build():Client.

oRequest = RequestBuilder
        :Get('https://www.progress.com')
        :Request.

oResponse = oClient:Execute(oRequest).

message
    oResponse:StatusCode   skip
    oResponse:StatusReason skip
view-as alert-box.
```

Basic operations

- Supports most HTTP methods

  GET, PUT, POST, DELETE, HEAD, TRACE, OPTIONS
- Payload/message body
  - In Object form in the `Entity` property
  - Built-in support for JSON, XML, binary, HTML/text, form data (name/value pairs)
- Cookies supported
- No restrictions on content type

Basic operations, cont'd

- Support for proxies per client or per request
- Redirects for

  `301/Moved Permanently`

  `302/Found`
- Transfer-encodings
  - Support for `chunked` responses
  - `zip/deflate` not supported
- Streaming not supported
- Not a rendering engine / UI

Creating a client

```
interface OpenEdge.Net.HTTP.IHttpClient:
  /** Free-text name & semantic version */
  def public property ClientName as char no-undo get.
  def public property ClientVersion as char no-undo get.

  /** Miscellanous options (like timeouts) */
  def public property Options as ClientOptions no-undo get.

  /** Executes an HTTP request. */
  method public IHttpResponse Execute (poRequest as IHttpRequest).

  method public void Execute (poRequest  as IHttpRequest,
                              poResponse as IHttpResponse).
end interface.
```

Creating a client

```
def var oHttpClient as OpenEdge.Net.HTTP.IHttpClient.


oHttpClient = ClientBuilder:Build()
      :Named(<char>, <char>)
      :UsingLibrary(<IHttpClientLibrary>)
      /* define client behaviour */
      :ViaProxy(<URI>)
      :KeepCookies(<ICookieJar>)
      /* options */
      :SetRequestTimeout(<dec>)
      :SetRetryPause (<dec>)
      :SetNumRetries (<int>)
      /* return a usable IHttpClient instance */
      :Client.
```

URI (Uniform Resource Identifier)

http://tools.ietf.org/html/rfc3986

```
<scheme>://<username>:<password>@<host>:<port>
  /<path>;<parameters>?<query>#<fragment>
```

```
http://nbbedpjudge1:8980/
                VehicleOrderSvc/rest/VehicleOrder/BrandData
                      ?filter={"brandName":"fjord"}
```

Special characters	
Reserved	; / ? : @ & = + $,
Unreserved	- _ . ! ~ * ' ( )
Unwise	{ } | \ ^ [ ] `
Excluded	< > # % "

http://www.skorks.com/2010/05/what-every-developer-should-know-about-urls/

Create a URI

```
def var oURI as OpenEdge.Net.URI.
oURI = URI:Parse('
  http://nbbedpjudge1:8980/VehicleOrderSvc/rest
  /VehicleOrder/BrandData
  ?filter={"brandName":"fjord"}').
            /* ---------- OR ---------  */
oURI = new URI('http', 'nbbedpjudge1', 8980).
oURI:Path =
  'VehicleOrderSvc/rest/VehicleOrder/BrandData'.
oURI:AddQuery('filter', '{"brandName":"fjord"}').
```

Creating a request

```
define variable oRequest as OpenEdge.Net.HTTP.IHttpRequest no-undo.

oRequest = RequestBuilder:Build('GET', 'http://www.progress.com')
       :AcceptContentType(<char>)
              | AcceptAll | ~FormData | ~Html | ~Json | ~Xml
       :AddHeader(<HttpHeader>)
       /* message body */
       :WithData(<Progress.Lang.Object>)
              | AddFormData(<data>)     | AddJsonData(<JsonObject>)
       :ContentType(<char>)
       /* authentication */
       :AuthCallback(<callback object> | <callback procedure>)
       :UsingBasicAuthentication(<Credentials>)
              |UsingDigestAuthentication(<Credentials>)
              |UsingCredentials(<Credentials>)
       /* proxy this request */
       :ViaProxy(<url>)
       :WithTransferEncoding(<encoding>)
   /* a usable IHttpRequest instance */
   :Request.
```

Creating a request

```
create x-document hXmlDoc.
hXmlDoc:load('file', 'soap.xml', no).
oXmlDoc = new WidgetHandle(hXmlDoc).

oPostURI =
  URI:Parse('http://httpbin.org/post').

oRequest = RequestBuilder
         :Post(oPostURI, oXmlDoc)
         :AcceptJson()
         :Request.
```

Creating a request

```
copy-lob from file 'document.pdf' to mDocument.


oInputEntity = new Memptr(
               get-pointer-value(mDocument),
               get-size(mDocument)).


oRequest = RequestBuilder
          :Post('http://httpbin.org/post',
               oInputEntity)
          :ContentType('application/pdf')
          :AcceptJson()
          :Request.
```

Request executes

```
oResponse = moHttpClient:Execute(oRequest).
```

Processing a response

```
if oResponse:StatusCode <> 200 then
  return error
         'Request error: ' + string(oResponse:StatusCode).
if not valid-object(oResponse:Entity) then
  return error 'Expected valid entity'.


if oResponse:ContentType = 'application/json' then
  cast(oResponse:Entity, JsonObject):WriteFile('data.json').


else
  return error 'Unexpected content type: ' +
               oResponse:ContentType.
```

Response status codes

- Status code for machines; reason for humans

- Success
  - 200 (**OK**): it worked, and we have a payload returned
  - 201 (**Created**): everything works well and an element was created
  - 204 (**No content**): it worked, and there's no response payload

- Failure
  - 400 (**Bad Request**): the most generic one telling that the request sent isn't correct
  - 401 (**Unauthorized)**: this occurs when we aren't authenticated
  - 404 (**Not Found)**: when trying to access a resource that doesn't exist
  - 500 (**Internal Server Error)**: something wrong within the server side processing when trying to execute the request

Some important headers

Authentication

- Library supports pre-emptive or interactive
  - Support for `401/Unauthorized`
- Provide credentials to request, or via callback to procedures & classes
- Requests implement `IAuthenticatedRequest`
- Credentials never persisted
- Customisable authentication schemes
  - Basic, Digest provided

Authentication

```
oUri = URI:Parse( 'http://oelxdev06:8881/oemanager/').
oCredentials =
     new Credentials('admin', 'tomcat', 'tomcat').
/* completely pre-emptive: credentials added immediately */
oReq = RequestBuilder:Get(oURI)
         :UsingBasicAuthentication(oCredentials)
         :Request.

/* partly pre-emptive: credentials added after negotiation */
oReq = RequestBuilder:Get(oURI)
         :UsingCredentials(oCredentials)
         :Request.
```

Authentication

```
/* completely interactive: credentials added after callback returns */
oReq = RequestBuilder:Get('http://oelxdev06:8881/oemanager/')
            :AuthCallback(new MyAuthCallback())
            :Request.


class MyAuthCallback implements IAuthFilterEventHandler:

  method public void AuthFilter_HttpCredentialRequestHandler(
              input poSender as Object,
              input poEventArgs as AuthenticationRequestEventArgs ):

   /* get values from some UI here */
   poEventArgs:Credentials = new Credentials('domain', 'user', 'pw').

  end method.

end class.
```

Authentication

```
oReq = RequestBuilder:Get('http://oelxdev06:8881/oemanager/')
           :AuthCallback(this-procedure)
           :Request.

procedure AuthFilter_HttpCredentialRequestHandler:
  def input param poSender as Object.
  def input param poEventArgs as AuthenticationRequestEventArgs.
  /* get values from some UI here */
  poEventArgs:Credentials =
                  new Credentials('domain', 'user', 'pw').
end procedure.
```

Cookies

- Stateful client manages cookies via a CookieJar
  - Normal client is stateless (ie no cookie cache)
- Requests & responses support cookies regardless of client
  - Cookies are transported via `Set-Cookie` and `Cookie` headers

```
Set-Cookie: JSESSIONID=<value>.webtest; Version=1;
Path="/oemanager/"; HttpOnly"

Cookie: JSESSIONID=<value>.webtest
```

Cookies

```
define variable iNumCookies as integer no-undo.
define variable oCookies as Cookie extent no-undo.

oClient = ClientBuilder:Build()
              :KeepCookies(CookieJarBuilder:Build():CookieJar)
              :Client.
oRequest = RequestBuilder:Get('http://oelxdev06:8881/oemanager/')
                 :Request.

oResponse = oClient:Execute(oRequest).
iNumCookies = oResponse:GetCookies(output oCookies).

/* iNumCookies = 2
JSESSIONIDSSO=71DEF599DE7CE03DD3ACEDE69D0A6FE1; Version=1; Path="/"; HttpOnly

JSESSIONID=7A304344764B5E83E52F58F9CA489F6C4D33DE2F11AA.webtest; Version=1;
Path="/oemanager/"; HttpOnly" */
```

Cookies

```
define variable iNumCookies as integer no-undo.
define variable oCookies as Cookie extent no-undo.

/* Continued from earlier */
iNumCookies = cast(oClient, ISupportCookies)
        :CookieJar
        :GetCookies(input oRequest:URL, output oCookies).
/* iNumCookies = 2 */

oResponse = oClient:Execute(oRequest).

/*  REQUEST HEADERS
Cookie: JSESSIONIDSSO=71DEF599DE7CE03DD3ACEDE69D0A6FE1;

JSESSIONID=7A304344764B5E83E52F58F9CA489F6C4D33DE2F11AA.webtest;
*/
```

Cookies

```
interface OpenEdge.Net.HTTP.ICookieJar:
    /** The location in which cookies are persisted */
    define public property CookieJarPath as character no-undo get. set.

    /** Returns the cookies germane to the specified URI. */
    method public integer GetCookies(input poUri as URI,
                                     output poCookies as Cookie extent).

    /** Adds a cookie to the jar, for a given domain/path. */
    method public void AddCookie(input pcDomain as character,
                                 input pcPath   as character,
                                 input poCookie as Cookie).

    method public logical RemoveCookie(input poCookie as Cookie).

    /** Removes all session (non-persistent) cookies from the jar */
    method public void ClearSessionCookies().

    /** Clears all persistent cookies */
    method public void ClearPersistentCookies().
end interface.
```

Extensibility

- Builder pattern
  - HttpClient, HttpClientLibrary
  - HttpRequest, HttpResponse, HttpHeader
  - Registries for individual builders
- Filters
  - Status
  - Authentication
  - Payload / content types
  - Headers
- Client library
  - SSL and socket options

Extensibility: content type

- Add new handler

```
My.App.DocXBodyResponseFilter
```

- Register handler with the builder

```
ContentTypeResponseWriterBuilder:Registry
  :Put('application/vnd.openxmlformats-
  officedocument.wordprocessingml.document',
       get-class(DocXBodyResponseFilter))
```

- Run as normal

Extensibility: content type

```
class DocXBodyResponseFilter implements IHttpMessageWriter:
  define public property Message as IHttpMessage no-undo get.
      private set.
  constructor public DocXBodyResponseFilter(
                                    input poMessage as IHttpResponse).


  method public void Write( input pmData as memptr):
    define variable hDocument as handle no-undo.
    create x-document hDocument.
   hDocument:load('memptr':u, pmData, false).


    this-object:Message:Entity = new WidgetHandle(hDocument).
  end method.
end class.
```

Extensibility: authentication scheme

- Add new handler

```
App.Auth.OAuth2AuthenticationFilter
```

- Register handler with the builder

```
AuthenticationRequestWriterBuilder:Registry
  :Put('oauth',
       get-class(OAuth2AuthenticationFilter))
```

- Run as normal

Extensibility: authentication scheme

```
class OAuth2AuthenticationFilter inherits AuthenticationRequestFilter:

  method override protected void AddAuthentication():
    /* credentials retrieved from request or callback */
    assign oCredentials = GetCredentials(cRealm).

      if not valid-object(oCredentials) then
       return error new AppError('Missing credentials for realm', 0).

    assign cCredentials = <built to what OAuth expects>

    /* add to the request */
   oRequest:SetHeader(HttpHeaderBuilder:Build('Authorization')
                     :Value(cCredentials)
                           :Header).

  end method.

end class.
```

Debugging

`session:debug-alert = true`

- `request-raw.txt`
- `response-data-received.txt`

} in `session:temp-dir`

`log-manager:logfile-name = 'log.log'`

- Writes predefined `SOCKET` and `COOKIES` messages out

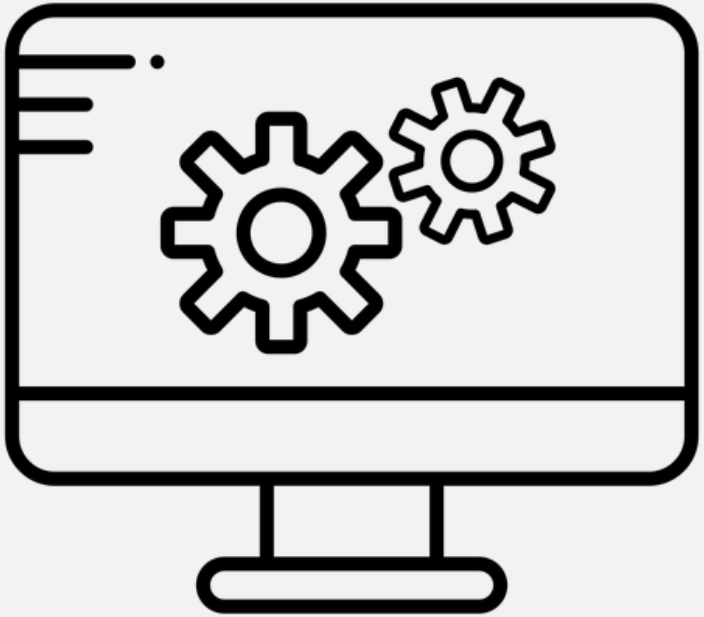
```
[<timestamp>] P-010592 T-008504 1 4GL -- Logging level set to = 1
[<timestamp>] P-010592 T-008504 1 4GL SOCKET         CONNECT: TIME=547
[<timestamp>] P-010592 T-008504 1 4GL SOCKET         WRITE: TIME=0; SIZE=160
[<timestamp>] P-010592 T-008504 1 4GL SOCKET         READ: TIME=0; SIZE=489
[<timestamp>] P-010592 T-008504 1 4GL SOCKET         TOTAL READ: TIME=70
```

Yes, but what is it for?

- PAS/OE management
- Great for QA of REST & other services
- Messaging services
- Push notification services
- Varied 3rd-party services
  - SharePoint, SalesForce, Concur
  - US Gov't, NPR, Marvel Comics Developer Portal
- Anything, really

https://developer.concur.com/docs-and-resources/documentation
http://www.usa.gov/About/developer-resources/federal-agency-directory/interactivedoc.shtml
http://developer.marvel.com/docs#!/public/

Demo



Q&A

pjudge@progress.com


PROGRESS