Developing REST Clients

In the ABL using the HTTP client classes
Speaker Wim HAM

- Bsc. Informatics Wilhelmus van der Ham (Wim)
- Born in the Netherlands (1966)
- For 25 years living and working in Italy
  - 1 wife (since 1993)
  - 2 daughters (since 1995 and 2000)
- This year celebrating **30 years** of «Work in Progress»
- At first freelance in Italy
- Since **2000** director of WITS - Worldwide IT Solutions
- In **2012** (21-12-2012) co-founder and vice president of PUG Italia
- Since **2014** Software Developer and Software Architect for EcoSafe
- **Mission:** Offer services and consultancy on all products of the Progress OpenEdge family.
Introduction

- In this presentation we will see how to consume REST services from the 4GL
- We will start with a short overview of JSON
- Then we talk about the HTTP Client classes in the Progress 4GL
- Finally we will show a demo of how we integrated two applications (YouTrack and VersAp) to register time tracking data
**JavaScript Object Notation (JSON)**: a simple format to exchange data.

- It starts with a curly brace `{`
- Followed by pairs of «Attribute»: «Value»
- After every pair a comma , or a curly brace }`
- Every «Value» can either be «simple» or a collection of other objects (array)
- To start an array you use the square bracket `[`
- To close the array another square bracket `]`
Schema
Datatypes

Available datatypes:
- Number
- String
- Boolean
- Null
Unavailable datatypes

- **Date**
  ISO 8601 date in the "YYYY-MM-DD" format

- **Datetime**
  An element can be parsed as an ISO 8601 date and time in the format: "YYYY-MM-DD[THH[:MM[:SS[.sss]]]]"

- **A date can be a number**

```plaintext
FUNCTION getDate RETURNS INT64
    (INPUT ipdtDateTime AS DATETIME ):
    /*
        Purpose: Convert datetime into msecs after 1/1/1970
        Notes:
    */
    DEFINE VARIABLE daData AS DATE NO-UNDO.
    DEFINE VARIABLE iGG AS INTEGER NO-UNDO.
    DEFINE VARIABLE iData AS INT64 NO-UNDO.

    daData = DATE(ipdtDateTime).
    iGG = daData - date(1,1,1970).
    iData = iGG * 24 * 60 * 60 * 1000.

    RETURN iData.
END FUNCTION.
```
With Progress 11 it’s possible to export and import the following objects to and from JSON in a native manner:

- Temp-table (whole table)
- Temp-table buffer (single buffer)
- ProDataSet (a set of temp-tables)

```sql
DEFINE TEMP-TABLE customer LIKE klant.
FIND FIRST klant WHERE klant.zoeknaam BEGINS "Pr" NO-ERROR.
IF AVAILABLE klant THEN DO:
    CREATE customer.
    BUFFER-COPY klant TO customer.
END.

TEMP-TABLE customer:WRITE-JSON ("LONGCHAR", cJSON, TRUE).
```

```json
Progress Software EMEA
{"customer": [
  {
    "zoeknaam": "PROEMEA",
    "klantnaam": "Progress Software EMEA",
    "bezoekadres": "Schorpioenstraat 67",
    "bezoekpostk": "3067 GG",
    "bezoekplaats": "Rotterdam",
    "bezoekland": "Nederland",
    "telefoon": "010-286 5700",
    "telefax": "010-286 5225"
  }
]}
```
New objects:
- **JSONObject**
  A single object from which all the attributes can be parsed
- **JSONArray**
  Allows access to the JSON objects contained in a JSON array

Method on JSONArray
- getJSONObject(index)

Methods on JSONObject
- getCharacter
- getInteger
- getLogical
REST

- Definition: **RE**presentational **S**tate **T**ransfer
- Protocol used on HTTP transport
- Parole chiavi
  - GET
  - PUT
  - POST
  - DELETE
  - PATCH
- URI containing Entity and Operation
REST HTTP client in Progress 4GL

- HTTP Client (IHttp.Client)
- HTTP Request (IHttpRequest)
- HTTP Response (IHttpResponse)
- Requestbody (CHAR)
Components we need:
- URL AS Character
- oRequestBody AS String
- oRequest AS IHttpRequest
- oResponse AS IHttpResponse
- oEntity AS Object

Build the request:
- RequestBuilder:Post(cURL, oRequestBody)

Execute with:
- oResponse = ClientBuilder:Build():Client:Execute (oRequest)

To receive output in oResponse

The format of the oResponse can be:
- JsonObject
- XML document
- Longchar
Example of Implementation

YouTrack
- Web Portal for issue tracking
- Has possibility of TimeTracking per issue
- Documented REST API

VersAp
- «Home-grown» application to manage applications, customers, change requests and …
- Activities

Exchange data with REST & JSON
YouTrack – API

http://tinyurl.com/YouTrack-REST-API
http://tinyurl.com/YouTrack-REST-NewWorkItem

Create New Work Item

POST /rest/issue/\{issue\}/timetracking/workitem

Create a new work item for a particular issue.

Parameters:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>issue</td>
<td>issue id</td>
<td>Issue id to which the work item is added</td>
</tr>
</tbody>
</table>

In the request code, provide the following parameters for a new work item (a work item is designated with workitem tag):

1. \{date\} – date of the new work item in Unix Epoch time format
2. \{duration\} – duration of the new work item, in minutes
3. \{description\} – activity description
4. \{worktype\} – work item type

CREATE (* First time this work item: Create *

ASSIGN

cFullURL = SUBSTITUTE("\{api\}/rest/issue/\{issue\}/timetracking/workitem", ipHost, ttWorkItem.cID)

oRequestString = SUBSTITUTE("\{workitem\}<\{date\}\{\{duration\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{duration\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\}\}\{\{description\}\}\{\{worktype\}\}\{\{id\}\}\{\{type\\}
Interface between VersAp - YouTrack
Conclusion

- REST client with Progress 11 is easy and fast
- Now you can integrate with other applications in a **non invasive** way
- Basis for creating (Web)Apps
- Perfect way to open up Progress towards others «worlds»

Implementations done so far:
- YouTrack and VersAp
- CRM Sugar
Question time
Contact

- Wim van der Ham – WITS
- wim@wits.it
- +39 335-68 77 283

Ing. Wilhelmus van der Ham

WITS - Worldwide IT Solutions sas
di Wilhelmus van der Ham & C.
via Motrassino, 2
10078 Venaria-Reale (TO) - Italy

Support:  http://www.ntrsupport.com/wits
Skype:  wim.vanderham

EcoSafe

- wim@ecosafe.it
Grazie!