Progress®

PASOE : Générer automatiquement des API Rest ou Microservices à partir de votre code existant

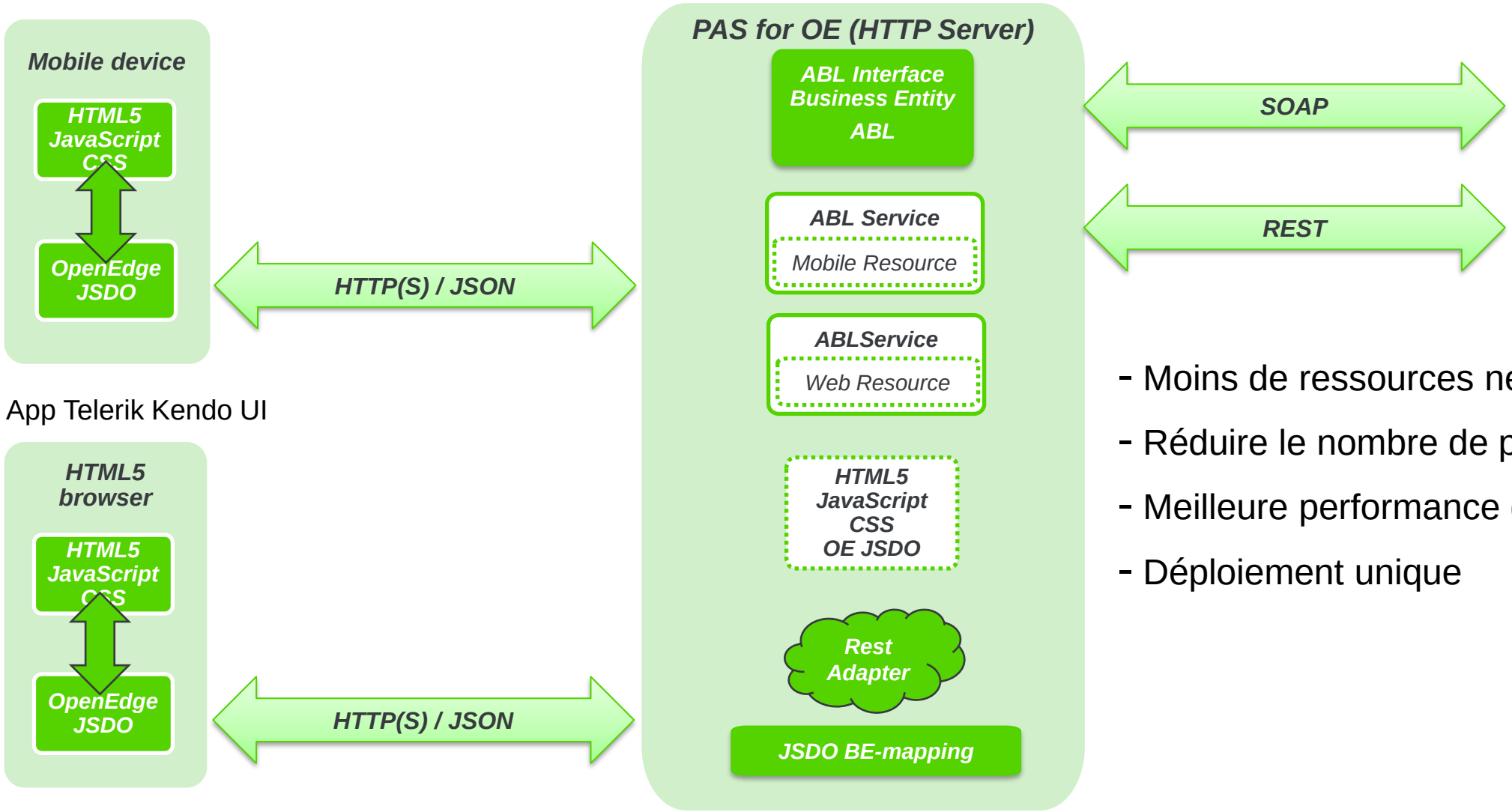
“It’s a Kind of Magic ! ”

Laurent KIEFFER

14 Octobre 2021

Architecture – Progress Application Server for OE

Mobile Apps Natives



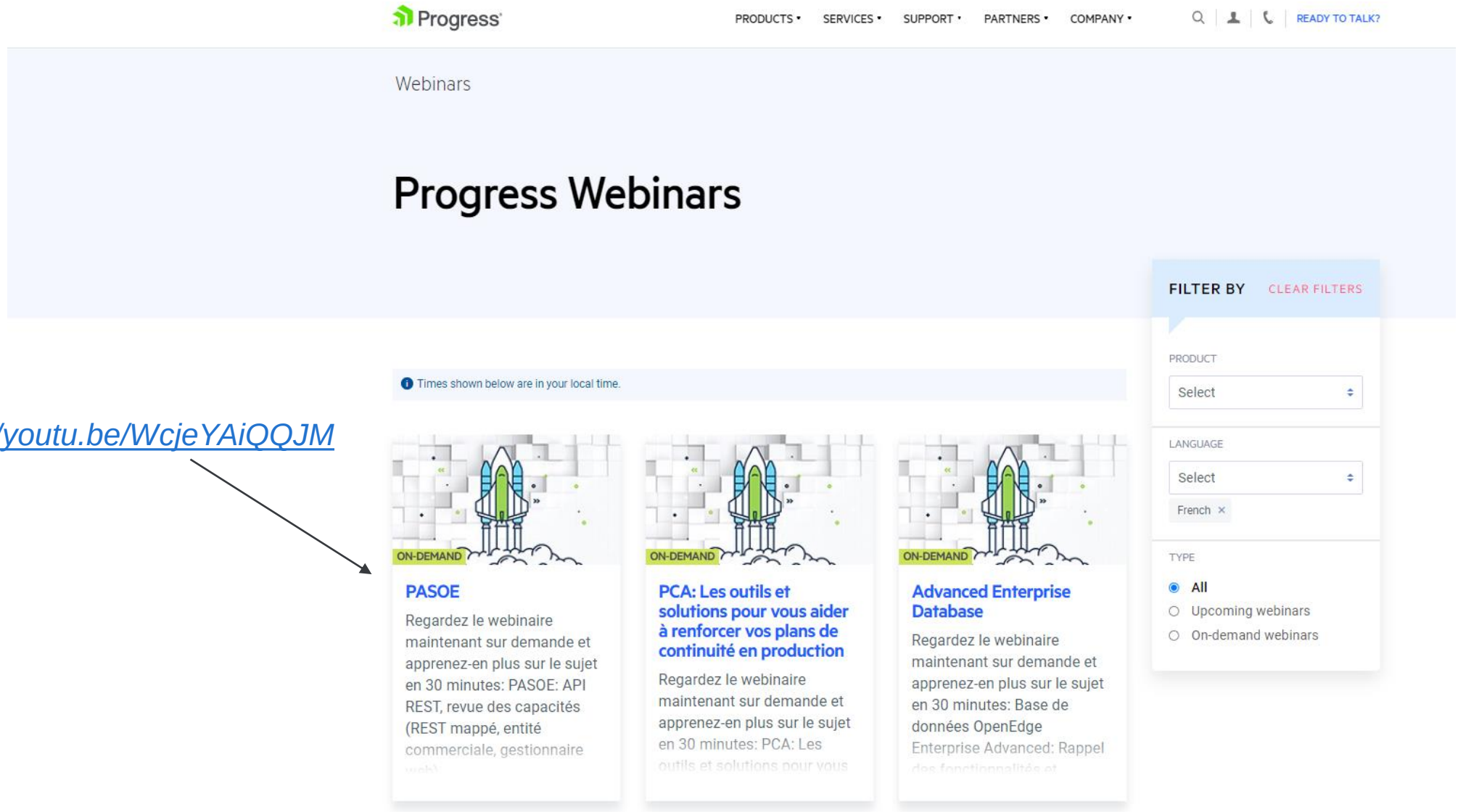
Web App Telerik Kendo UI

- Moins de ressources nécessaires
- Réduire le nombre de process OS
- Meilleure performance globale
- Déploiement unique

Progress®

Quelles sont les options pour ‘RESTifier’ OpenEdge?

Lien sur Webinaires : https://www.progress.com/webinars?filter=language French

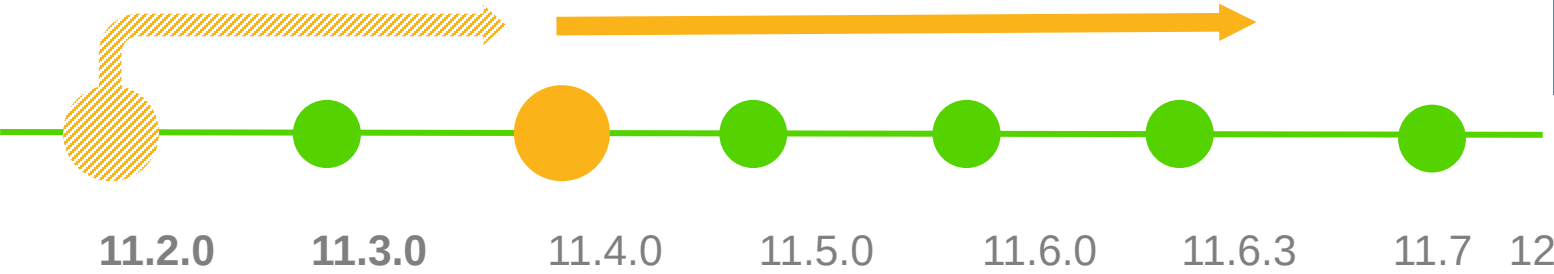


https://youtu.be/WcjeYAiQQJM

Approches Interface de Service

- a **Data Object (REST)**
- b Data Object (WebHandler)
- c REST (Mapped RPC)
- d WebHandler
- e DataObjectHandler

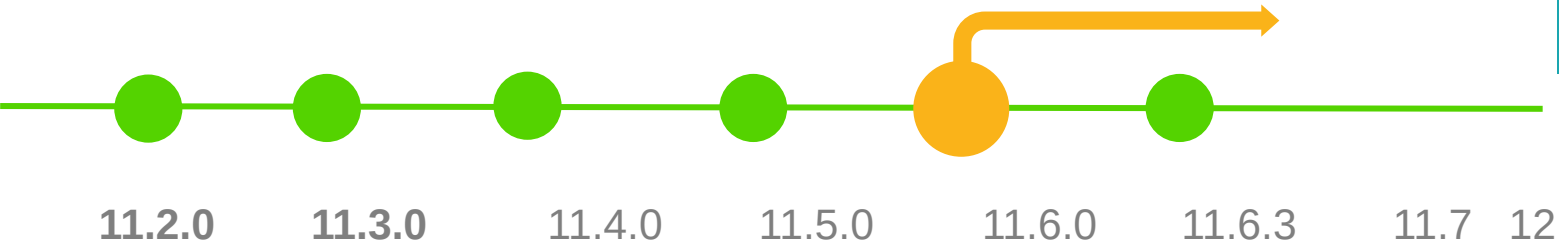
- Autrefois Services mobiles
- Annoter certaines méthodes (w/ particular signatures)
- Très normative
  - Modèle de programmation
  - URI paths
- Utilise le transport REST
- Crée un catalogue de services de données en tant qu'API public



Approches Interface de Service

a	Data Object (REST)
b	**Data Object (WebHandler)**
c	REST (Mapped RPC)
d	WebHandler
e	DataObjectHandler

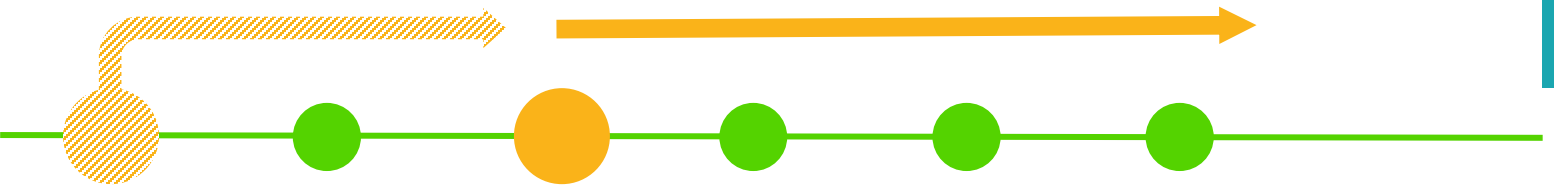
- Depuis 11.6.3
- Annoter certaines méthodes (w/ particular signatures)
- Assez normatif
- Plus de flexibilité que le mapping
- Utilise le transport WEB
- Exige PAS for OpenEdge
- Crée un catalogue de services de données en tant qu'API public
-



Approches Interface de Service

a	Data Object (REST)
b	Data Object (WebHandler)
c	**REST (Mapped RPC)**
d	WebHandler
e	DataObjectHandler

- Anciennement REST Services
-
- Outil de mapping graphique
-
- Utilise le transport REST
-
- Flexible dans les Paths URI
-



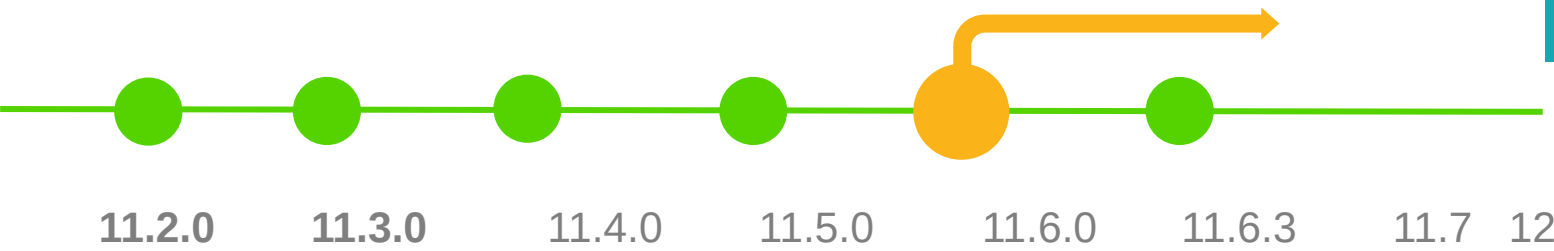
11.2.0 **11.3.0** 11.4.0 11.5.0 11.6.0 11.6.3 11.7 12

Progress

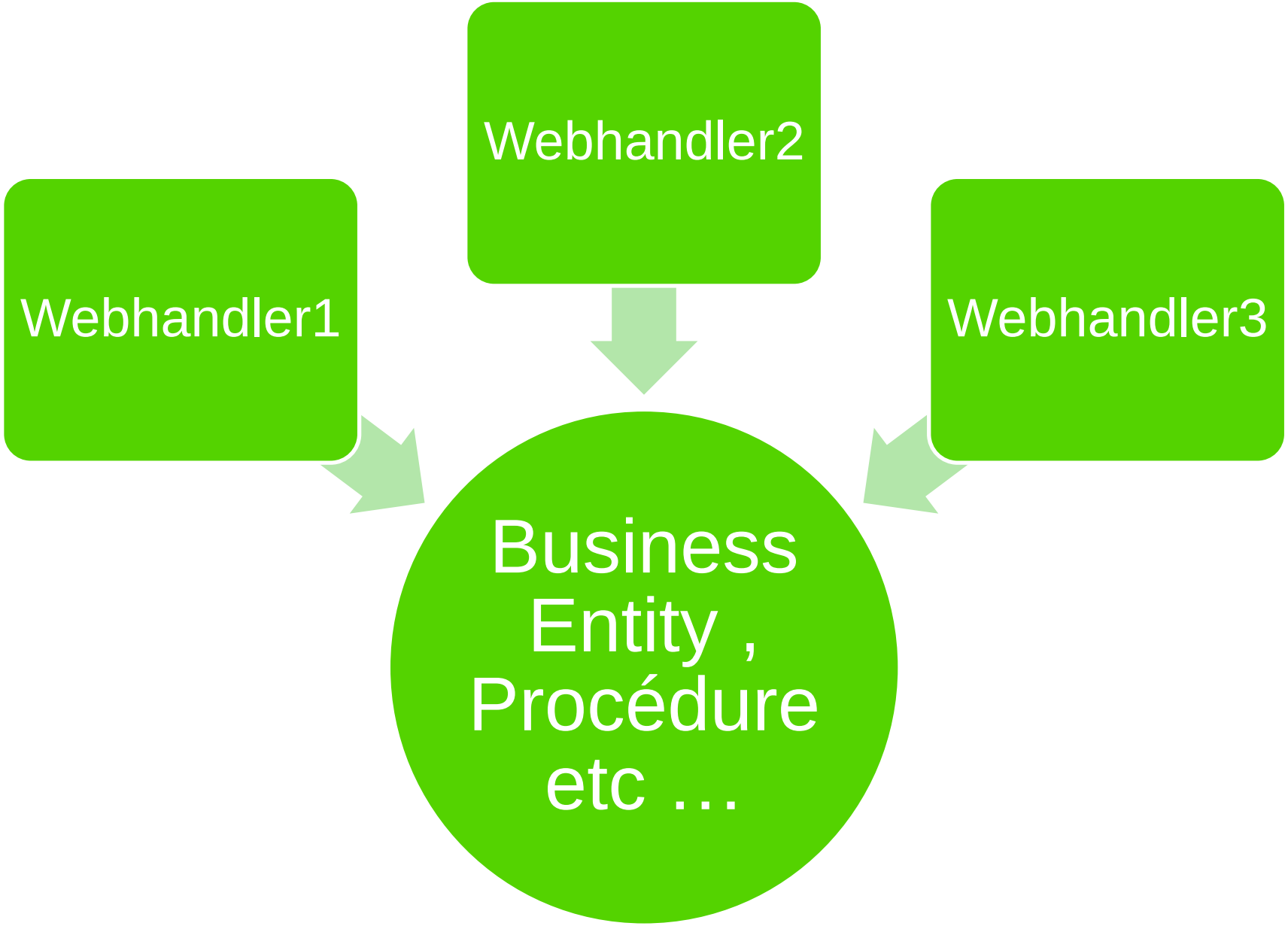
Approches Interface de Service

- a — Data Object (REST)
- b — Data Object (WebHandler)
- c — REST (Mapped RPC)
- d — **WebHandler**
- e — DataObjectHandler

- Associer une classe WebHandler OOABL à un modèle URI
- Utilise le transport WEB
- Nécessite PAS for OpenEdge
- TRÈS flexible, URI adaptable
- Faites ce que vous voulez dans le code/ABL
- Versions in-the-box
  - **OpenEdge.Web.WebHandler**
  - OpenEdge.Web.CompatibilityHandler
  - OpenEdge.Web.DefaultHandler



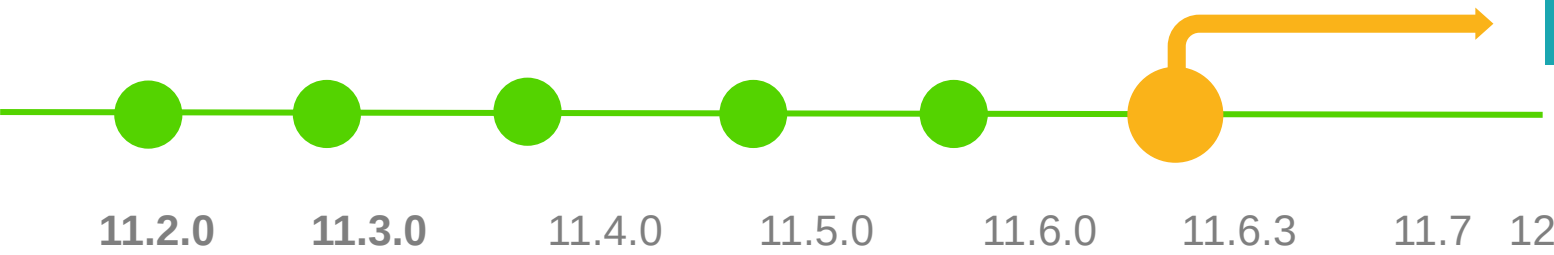
Progress



Approches Interface de Service

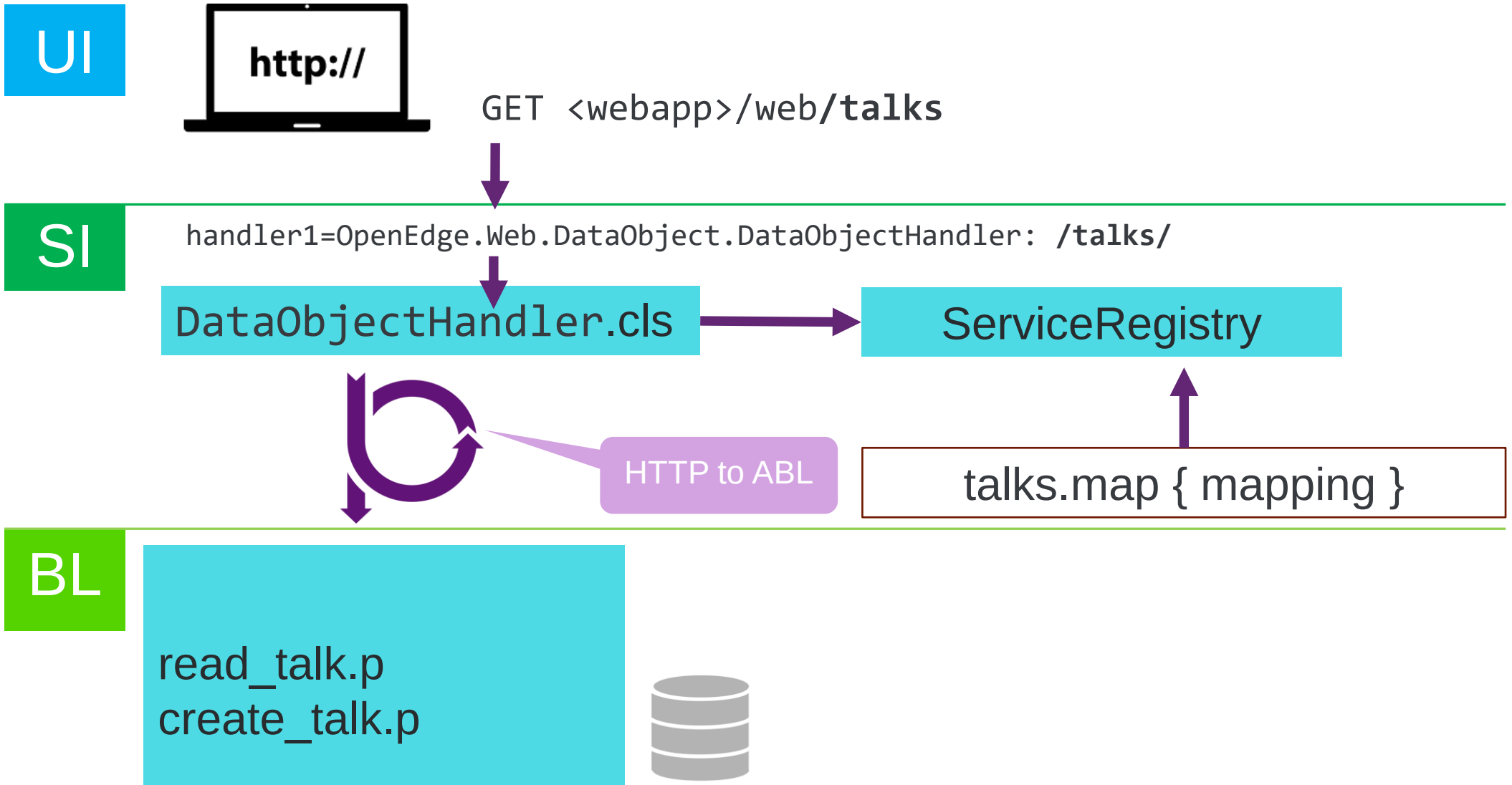
a	Data Object (REST)
b	Data Object (WebHandler)
c	REST (Mapped RPC)
d	WebHandler
e	**DataObjectHandler**

- WebHandler générique prédéfini
- Mapping défini dans ficher JSON
- TRES Flexible
- Nécessite PAS for OpenEdge

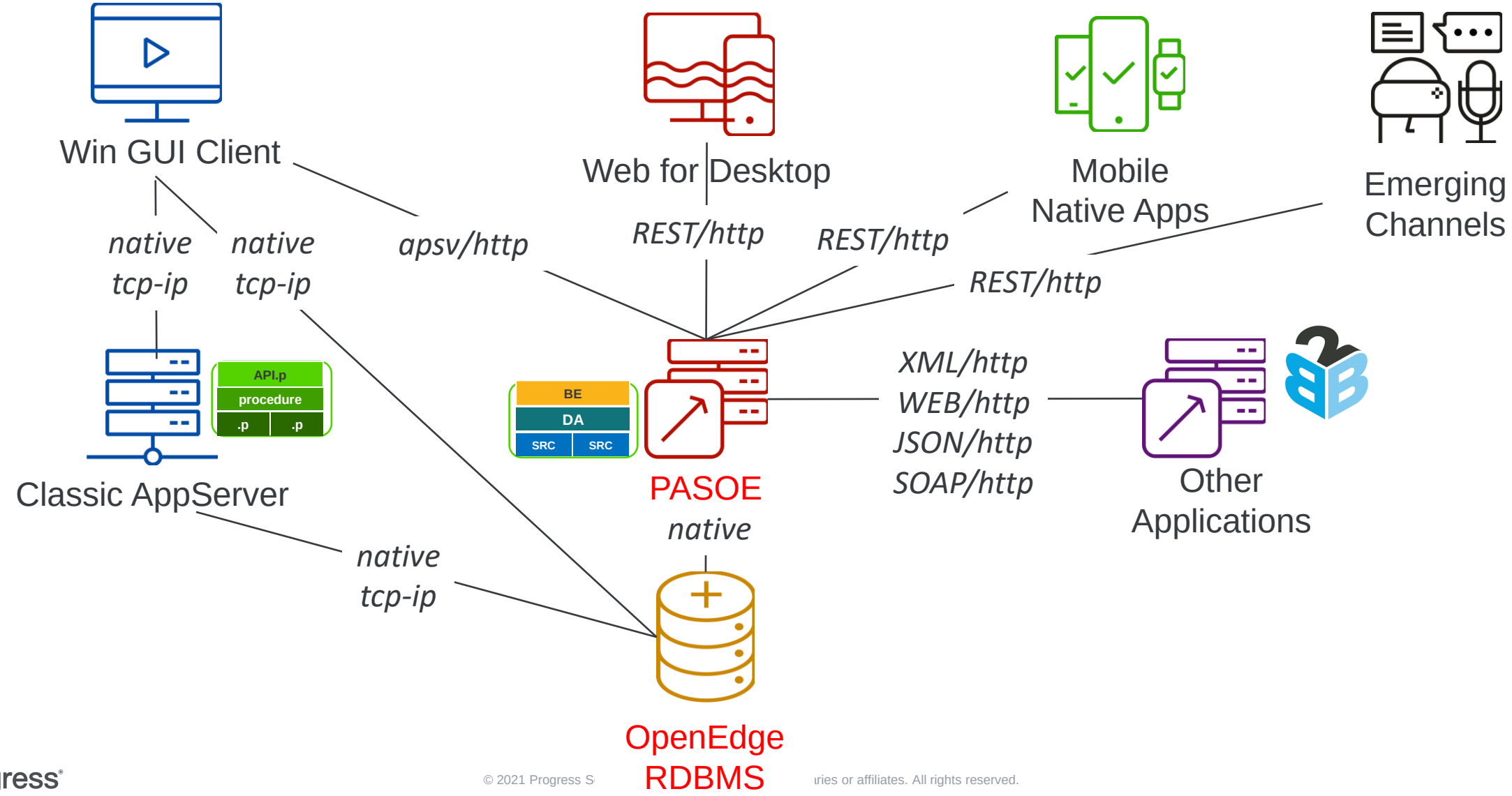


11.2.0 **11.3.0** 11.4.0 11.5.0 11.6.0 11.6.3 11.7 12

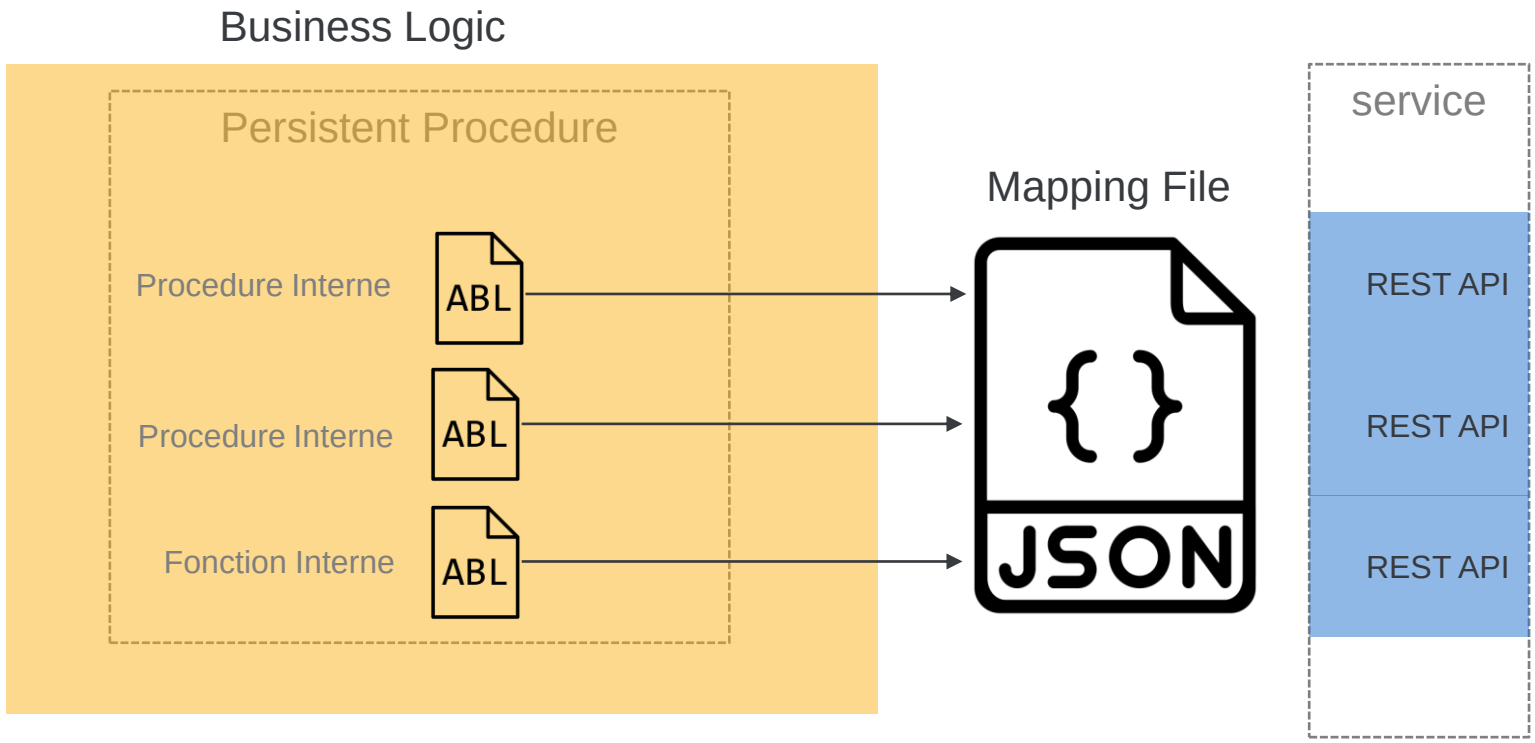
DataObjectHandler WebHandler Interaction



Réutilisation de l’existant



DOH : comment ca marche



DataObjectHandler: Ce que je peux Mapper ?

```
"/talks/{talk-id}": {
  "POST": {
    "contentType": "application/json",
    "statusCode": 201,
    "options": {
      "responseEnvelope": true
    },
    "entity": {
      "name": "logic/talk/new_talk.p",
      "function": "add_talk",
      "arg": [ {
        "ablName": "ttTalk",
        "ioMode" : "INPUT",
        "ablType": "table",
        "msgElem": { "type": "body",
                     "name": null }
      }, {
        "ablName": "pcChar",
        "ioMode" : "output",
        "ablType": "character",
        "msgElem": { "type": "header",
                     "name": "location" }
} } ] } } },
```

URI mapping

```
/talks
/{service}/data/{resource}
/{collection}/{coll-id}
```

Status codes

```
202 / Accepted
418 / I'm a teapot
```

Envelopes

```
requestEnvelope : "input"
errorEnvelope   : "oops"
```

IO Modes

```
"input" "output" "input-output" "return"
```

ABL data types (also `extent` variants)

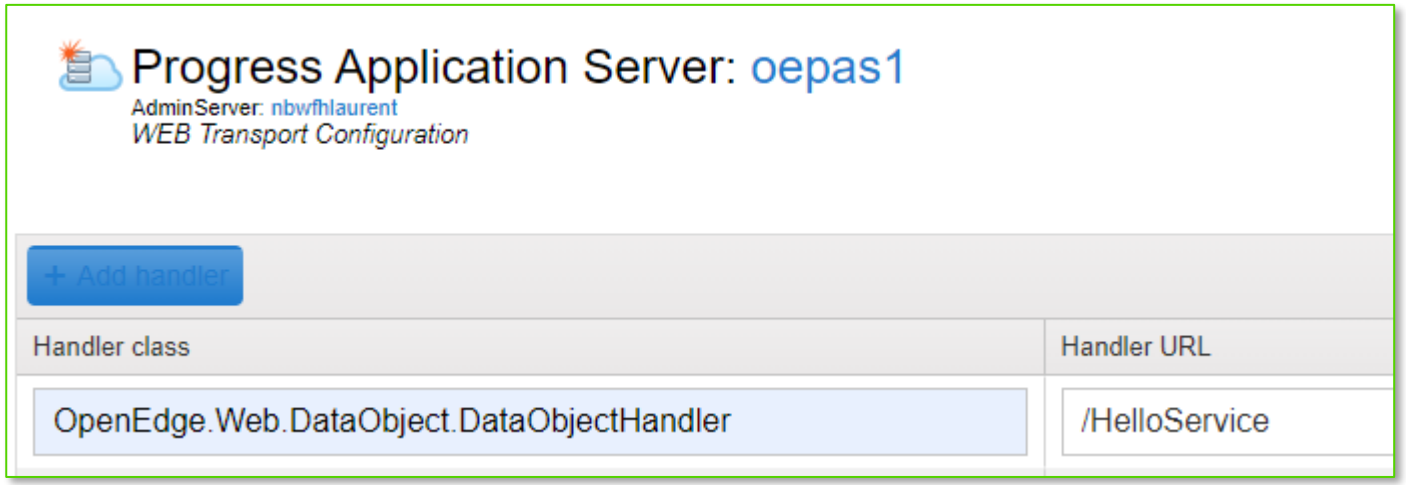
```
"character", "longchar", "integer", "int64", "decimal",
"logical", "rowid", "recid", "date", "datetime", "datetime-tz",
"raw", "memptr", "dataset", "temp-table",
"class <ooabl.type.name>"
```

HTTP Message elements

Request-only	`"path", "query", "httpMethod", "request", "constant"`
Response-only	`"none", "statusCode", "statusReason"`
Both	`"cookie", "header", "field", "body"`

Créer ce type de Service

- Dans PDSOE
  - **New > ABL Service > Web transport > Nom (HelloService) > Type Webhandler (OpenEdge.Web.DataObject.DataObjectHandler) > Resource URI (/HelloService)**.
  - Créer un fichier de mapping dans `/PASOEContent/WEB-INF/openedge` : HelloService.map
- Dans OE Explorer



Mapping example

NewService.map

```
"/Customer/{cust-num}": {
    "GET": {
        "contentType": "application/json",
        "statusCode": 200,
        "entity": {
            "arg": [{
                    "ablName": "cust-num",
                    "ablType": "integer",
                    "ioMode": "INPUT",
                    "msgElem": {
                        "type": "path",
                        "name": "cust-num"
                    }},
                {  "ablName": "ttCustomer",
                    "ioMode": "output",
                    "ablType": "table",
                    "msgElem": {
                        "type": "field",
                        "name": "data"
                    }}],
            "type": "procedure",
            "name": "Customers.p",
            "function": "get_Customer"
        }}},
```

Customers.p

```
procedure get_filtered_talks:
  define input  parameter pFilter as character.
  define input  parameter pSkipRecs as integer.
  define input  parameter pTopRecs as integer.
  define output parameter table for ttTalk.
  define output parameter pCount as integer.

procedure get_Customer:
  define input  parameter cust-num as integer.
  define output parameter table for ttCustomer.

procedure read_param_filter_response:
  define input parameter pParams as FilterParams.
  define output parameter table for ttTalk.
  define output parameter pResp as FilterResponse.
```

Mapping example

conf.map

```
"/Talks({talk-id})": {
  "GET": {
    "contentType": "application/json",
    "statusCode": 200,
    "entity": {
      "name": "read_talks.p",
      "function": "get_single_talk",
      "arg": [ {
        "ablName": "pId",
        "ioMode" : "INPUT",
        "ablType": "character",
        "msgElem": { "type": "path",
                     "name": "talk-id"}
      }, {
        "ablName": "ttTalk",
        "ioMode" : "output",
        "ablType": "table",
        "msgElem": { "type": "field",
                     "name": "data"}
    }
}} ] } } }
```

read_talks.p

```
procedure get_filtered_talks:
  define input  parameter pFilter as character.
  define input  parameter pSkipRecs as integer.
  define input  parameter pTopRecs as integer.
  define output parameter table for ttTalk.
  define output parameter pCount as integer.

procedure get_single_talk:
  define input  parameter pId as character.
  define output parameter table for ttTalk.

procedure read_param_filter_response:
  define input parameter pParams as FilterParams.
  define output parameter table for ttTalk.
  define output parameter pResp as FilterResponse.
```

Mapping example

conf.map

```
"/talks/": {
  "GET": {
    "statusCode": 200,
    "entity": {
      "name": "read_talks.p",
      "function": "get_filtered_talks",
      "arg": [ {
        "ablName": "pFilter",
        "ioMode" : "INPUT", "ablType": "character",
        "msgElem": {"type": "query", "name": "filter"}
      }, {
        "ablName": "pSkipRecs",
        "ioMode" : "input", "ablType": "integer",
        "msgElem": {"type": "query", "name": "skip"}
      }, {
        "ablName": "pTopRecs",
        "ioMode" : "input", "ablType": "integer",
        "msgElem": {"type": "query", "name": "top"}
      }, {
        "ablName": "ttTalk",
        "ioMode" : "output", "ablType": "table",
        "msgElem": {"type": "field", "name": "data"}
      }, {
        "ablName": "pCount",
        "ioMode" : "output", "ablType": "integer",
        "msgElem": {"type": "field", "name": "count"}
 }} ] } } }
```

read_talks.p

```
procedure get_filtered_talks:
  define input  parameter pFilter as character.
  define input  parameter pSkipRecs as integer.
  define input  parameter pTopRecs as integer.
  define output parameter table for ttTalk.
  define output parameter pCount as integer.


procedure get_single_talk:
  define input  parameter pId as character.
  define output parameter table for ttTalk.

procedure read_param_filter_response:
  define input parameter pParams as FilterParams.
  define output parameter table for ttTalk.
  define output parameter pResp as FilterResponse.
```

Mapping example

conf.map

```
"/talks/": {
  "GET": {
    "contentType": "application/json",
    "statusCode": 200,
    "entity": {
      "name": "read_talks.p",
      "function": "read_param_filter_response",
      "arg": [ {
        "ablName": "pFilter",
        "ioMode" : "input",
        "ablType": "class logic.FilterParams",
        "msgElem": [
          {"type": "query", "name": "filter", "ablRef": "Where" },
          {"type": "query", "name": "top",    "ablRef": "TopRecs"},
          {"type": "query", "name": "skip",   "ablRef": "SkipRecs"}
       ]
      }, {
        "ablName": "ttTalk",
        "ioMode" : "output", "ablType": "table",
        "msgElem": {"type": "field", "name": "data"}
      }, {
        "ablName": "pResp",
        "ioMode" : "output",
        "ablType": "class logic.FilterResponse",
        "msgElem":
          {"type": "field", "name": "count", "ablRef": "NumRecs"}
      }
] } } }
```

read_talks.p

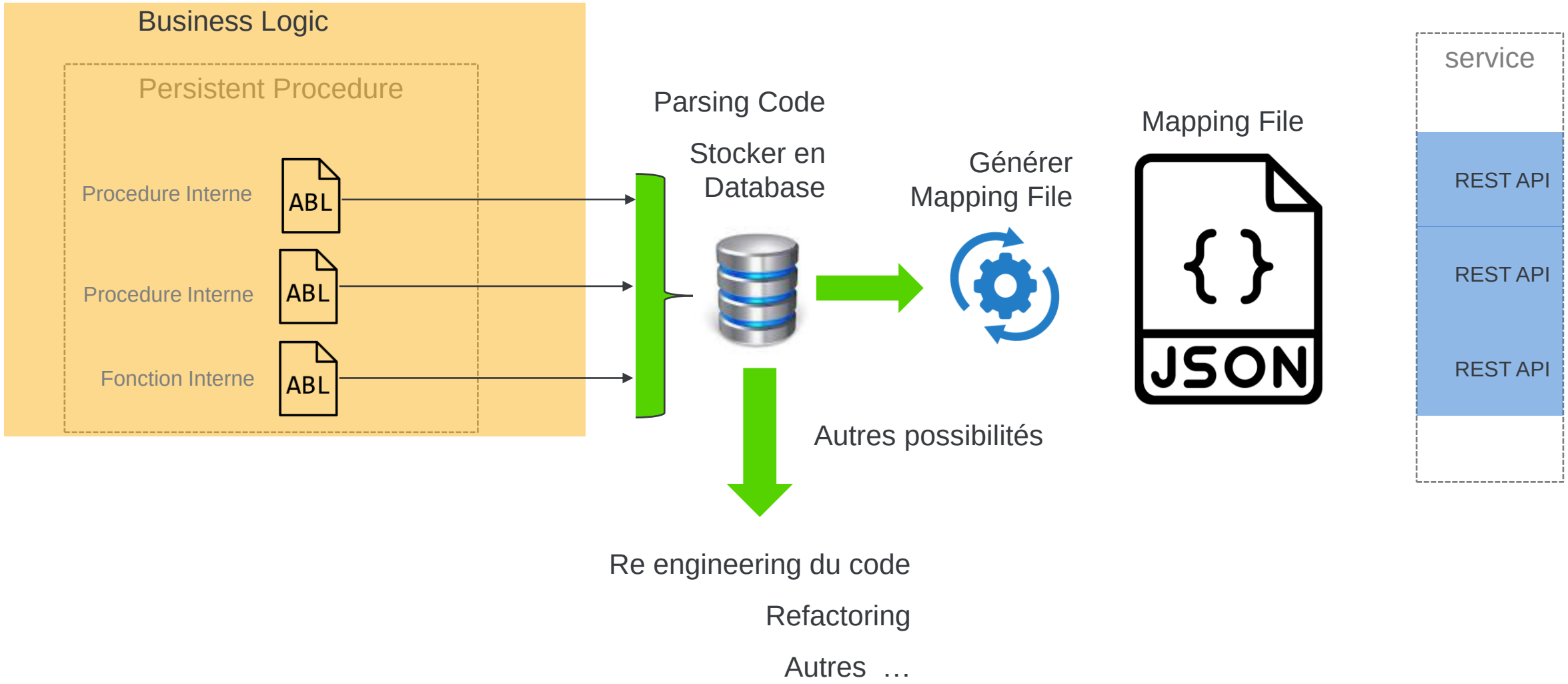
```
procedure get_filtered_talks:
  define input  parameter pFilter as character.
  define input  parameter pSkipRecs as integer.
  define input  parameter pTopRecs as integer.
  define output parameter table for ttTalk.
  define output parameter pCount as integer.

procedure get_single_talk:
  define input  parameter pId as character.
  define output parameter table for ttTalk.

procedure read_param_filter_response:
  define input parameter pParams as FilterParams.
  define output parameter table for ttTalk.
  define output parameter pResp as FilterResponse.
```

Customer : Exemple de Mapping

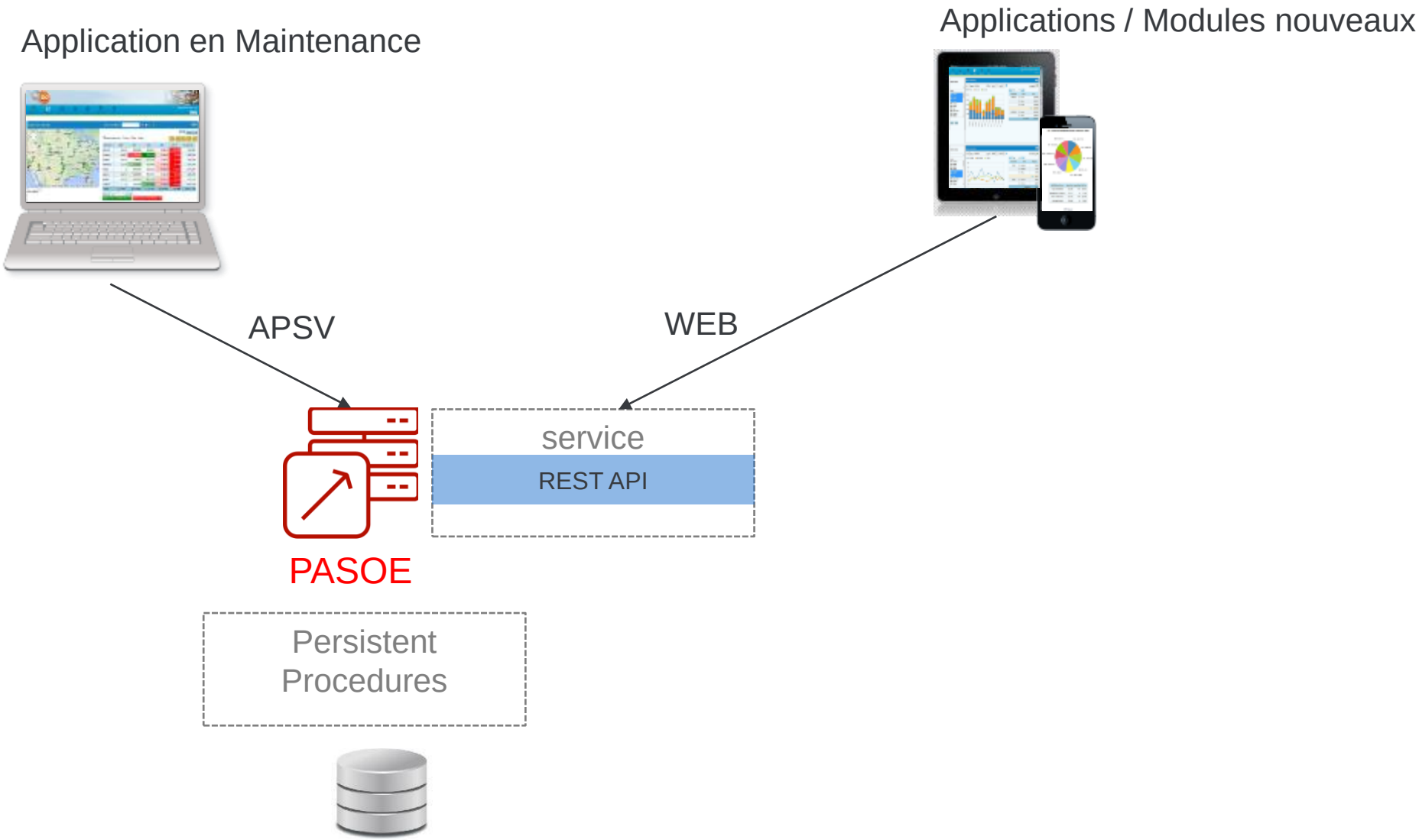
Du Manuel vers l'Automatique

- Le principe adopté : statique
  - Analyse des procédures et extraction des procédures/fonctions internes
  - Stockage dans une base de données
  - Lecture de la base totalement ou par filtre
    - Génération structure json pour mapper les procédures et paramètres
    - Copie du fichier dans PASOE
  - Création d'un Handler de type OpenEdge.Web.DataObject.DataObjectHandler
  - Démarrage PASOE

DOH : Automatisation



Exemple d’utilisation



Progress®

Et Voila !!!!! Demo

Autres liens utiles

- https://github.com/progress/Spark-Toolkit-Demos
  - Sports - Demonstrates use of the built-in DataObjectHandler (DOH) using statically-produced service mapping files from PDSOE. There is no UI for this application.
  - DynSports - Similar to the Sports project, but uses an automatic runtime discovery of classes to create necessary metadata for the DOH pattern. There is no UI for this application.
- https://github.com/progress/Spark-Toolkit
  - This repository primarily contains ABL artifacts and was built specifically for the Progress Application Server for OpenEdge to provide the back-end (server-side) support for exposing ABL logic via HTTP/HTTP.

Conclusions

- Avec PASOE l’exposition de Service REST est flexible
- Les WebHandler permettent de gérer tout type d’argument HTTP
  - GET , POST, etc
  - Entêtes
  - WebRequest
  - WebResponse
  - Status Code
- Tout traitement existant peut être exposé sous forme d’API REST
  - Procédure (.p)
  - Classes (.cls)

Conclusions

- Avec l’approche DataObjectHandler
  - On peut facilement exposer du code ancien sous forme de Services / MicroServices
  - On peut continuer la maintenance de ce code tout en modernisant
  - Pas de modification de code via Progress Developer Studio

Progress®

Q&R


Progress®