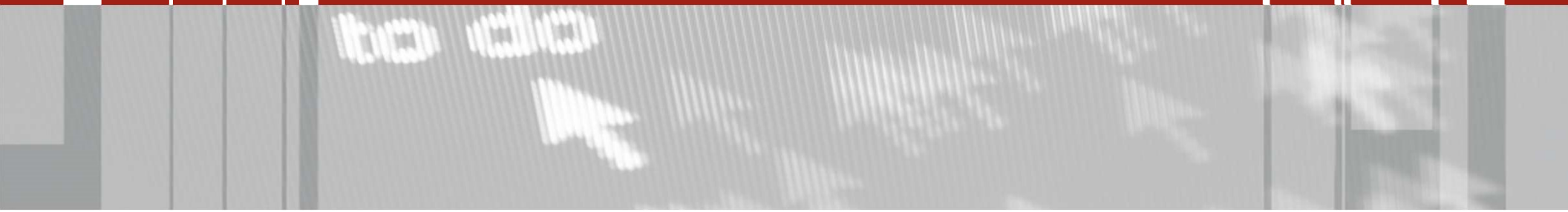
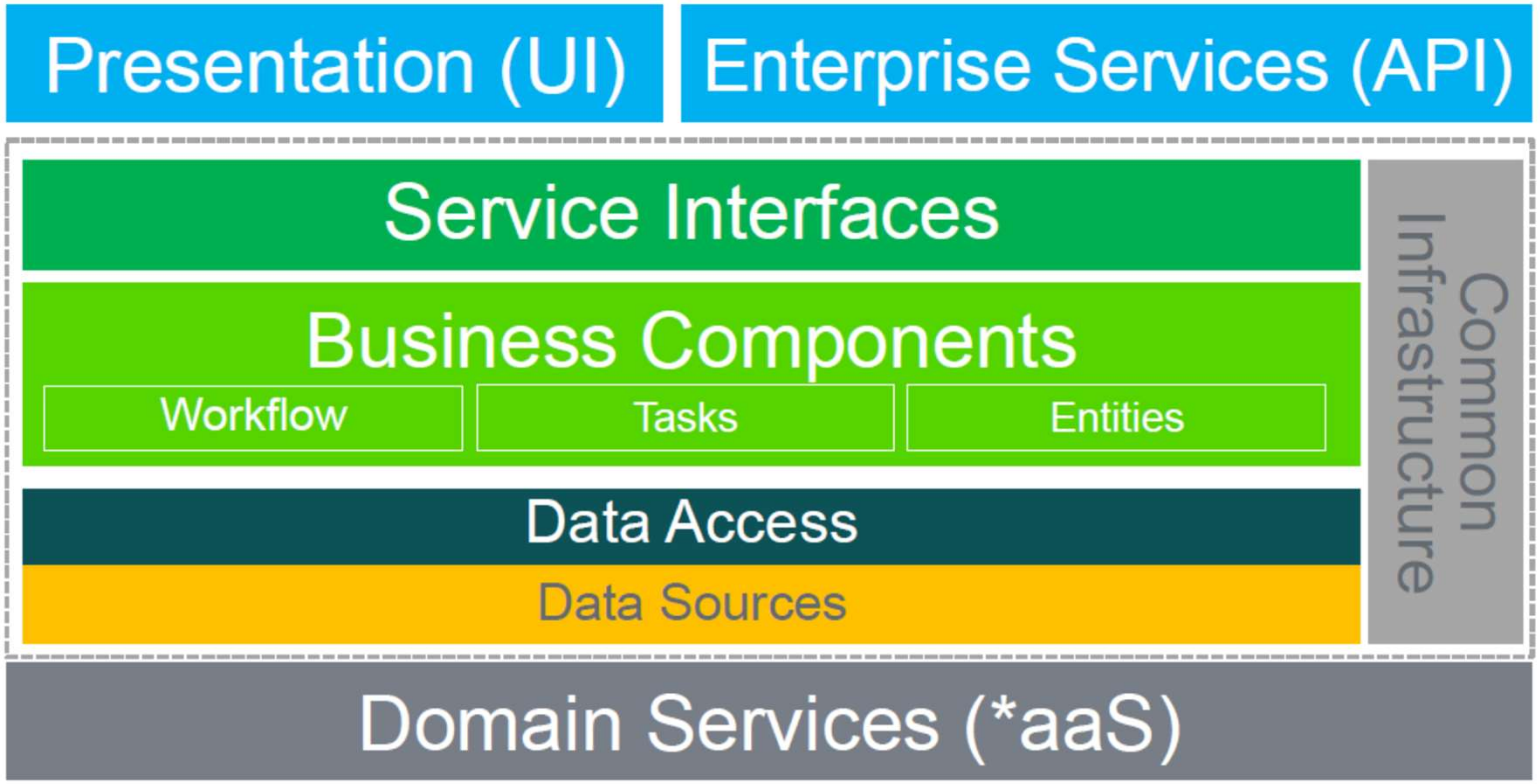


# REST APIs for OpenEdge: DataObject Handler

# Agenda

- **Services interfaces**
- **Data Object Handler architecture**
- **Create & configure ABL Services using the DOH**
- **Customising & extending behavior**
- **Demo**

# Service Interfaces



# What is REST

**REST = REpresentational State Transfer**

**REST is an architectural style for network based software that requires stateless, cacheable, client-server communication via a uniform interface between components.**

**Resources are named using a URL**

**Supports many representations of data: JSON, XML, multi-part**

**Uniform interface of HTTP Verbs: GET, PUT, POST, DELETE…etc.**

**Stateless, no client context between requests**

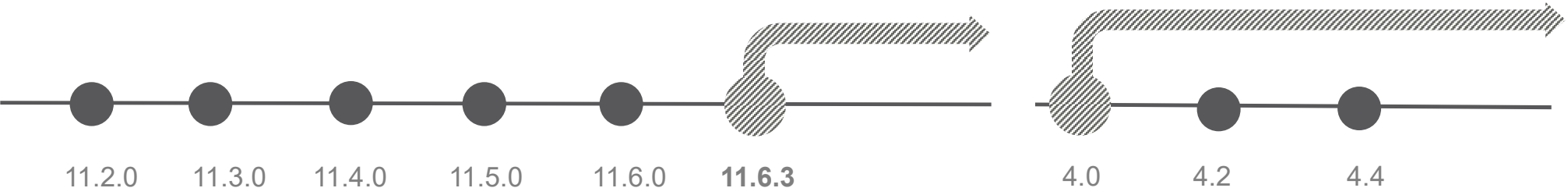
**Designed for performance & scalability, i.e. supports caching**

**In practice there are 'degrees of RESTfulness'**

# What Options are there to 'RESTify' OpenEdge?

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

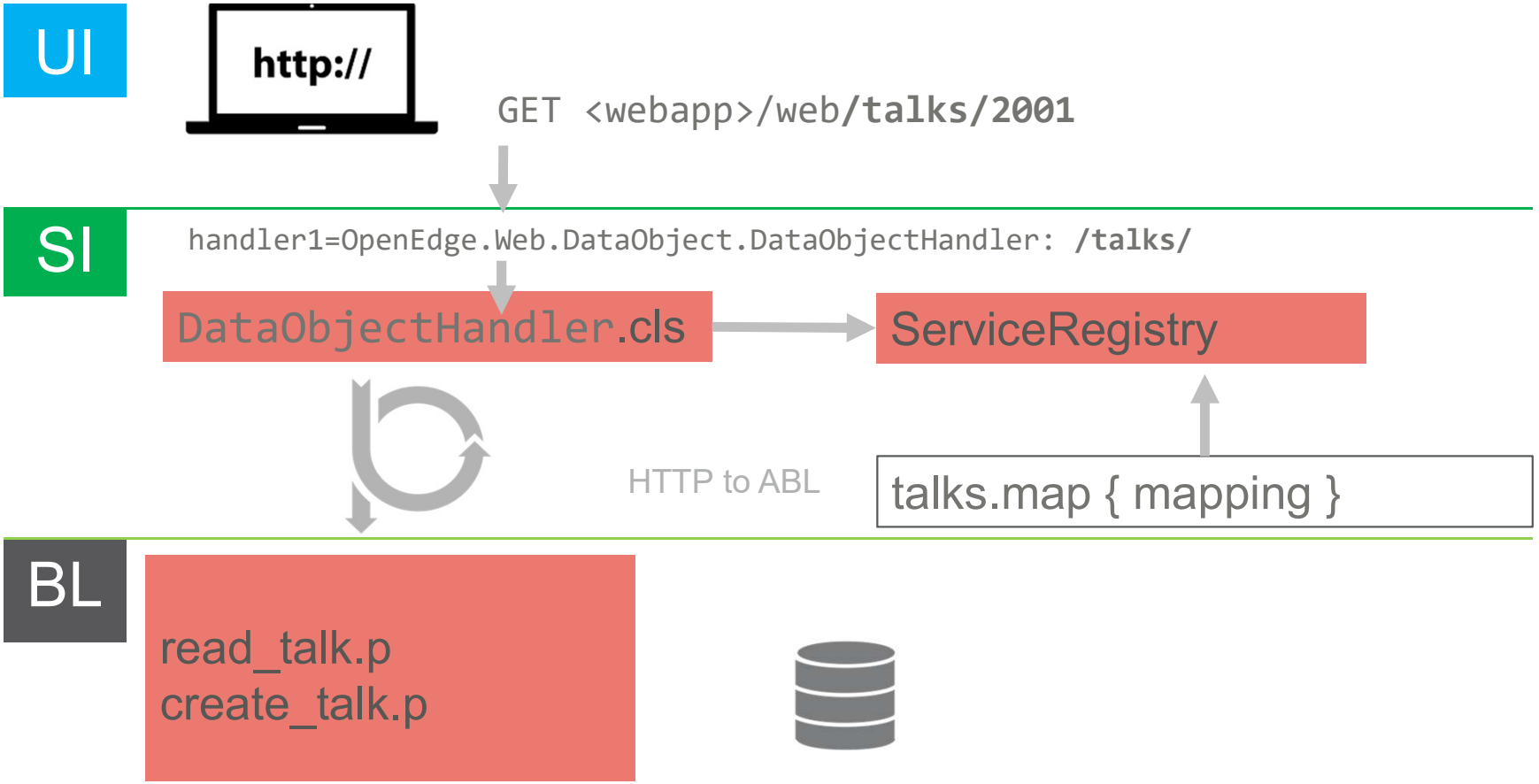
f OData view of OpenEdge ✓

# DataObject Handler

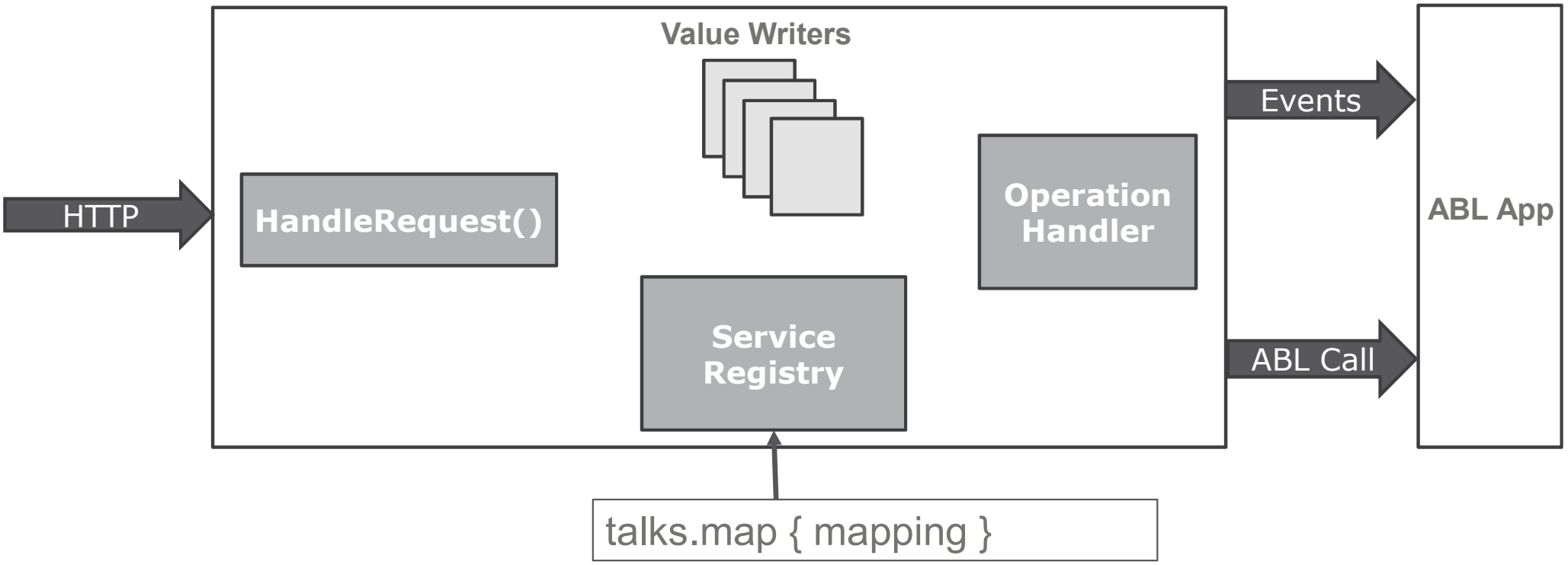
- **Pre-built generic WebHandler**
- **Mapping defined in JSON file**
- **Flexible in terms of URI**
- **Requires PAS for OpenEdge**

# DataObjectHandler WebHandler Interaction



# Architecture

## DataObject Handler



# Create&Configure

**1.Some ABL code to run**

Classes (instance/object) and/or persistent procedures

Read_talk.p
Update_speaker.p
SpeakerBE.cls

**2.Thought-out URI space**

URIs are hierarchical and case-sensitive

**3. A mapping file**

<service>.map

Describes the relationship between HTTP messages and ABL code

**4. One or more handler definitions**

Openedge.properties

Tells the ABL webapp which URIs should be passed to the DOH

**5. A set of authorization settings**

oeablSecurity.csv

Secures certain URIs using role-based access controls

# 3.Mapping File

```
"/talks/{tlk}": {
  "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"` |

# 5.Authorization

```
## Anyone can see info about a single talk, about all talks or about one or all speakers

# This includes all the CHILD URLs so we need to give some thought to our URL scheme

"/web/conf/talks/**","GET", "permitAll()"

# no child resources here

"/web/conf/talks/streams","GET", "permitAll()"

"/web/conf/talks/*/schedule","GET", "permitAll()"

## Anyone can submit a talk

# The Mapped RPC is specified above (we added POST to the <method>)

"/web/conf/talks", "POST", "permitAll()"

## Only organisersand speakers can update talks

"/web/conf/talks/*", "PUT", "hasAnyRole('ROLE_CONF_ORG,ROLE_SPEAKER')"
```

# DEMO

# Extending the DOH's behavior (1)

## Events

- **DiscoverService** – allows you to avoid mapping files and use another repository of service mappings, like a db

  The Spark.Core.Handler.DOHEventHandler does this .

  https://github.com/progress/Spark-Toolkit/

- **LoadEntity / UnloadEntity** – lets you start your classes/procedures the way you want to, or use a cache, or …

- **Invoked** – modify a response (like adding RESTy links)

# Extending the DOH's behavior (2)

**Plug in your own …**

- **Custom / vendor-specific content type writers**
- **Operation handlers (a jsdo operation maybe?)**
- **Argument value writers for writing data to/from complex objects**