Consultingwerk
software architecture and development

Web handler – the best thing since sliced bread

Peter Judge, Consultingwerk

Peter Judge

- Senior Architect at Consultingwerk
- Writing 4GL since 1996, working on a variety of frameworks and applications. More recently have worked on a lot of integration-y stuff: Authentication Gateway, HTTP Client, Web Handlers. Dabble in PASOE migrations.
- Active participator in Progress communities, PUGs and other events

Consultingwerk Software Services Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **related technology**
- Located in Cologne, Germany, subsidiaries in UK, USA and Romania
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
- Experts in OpenEdge Application Modernization

Services Portfolio, Progress Software

- OpenEdge (ABL, Developer Tools, Database, PASOE, …)
- Telerik DevCraft (.NET, Kendo UI, Angular, …), Telerik Reporting
- OpenEdge UltraControls (Infragistics .NET)
- Telerik Sitefinity CMS (incl. integration with OpenEdge applications)
- Kinvey Plattform, NativeScript
- Corticon BRMS
- WhatsUp Gold infrastructure-, network- and application monitoring
- Kemp Loadmaster
- …

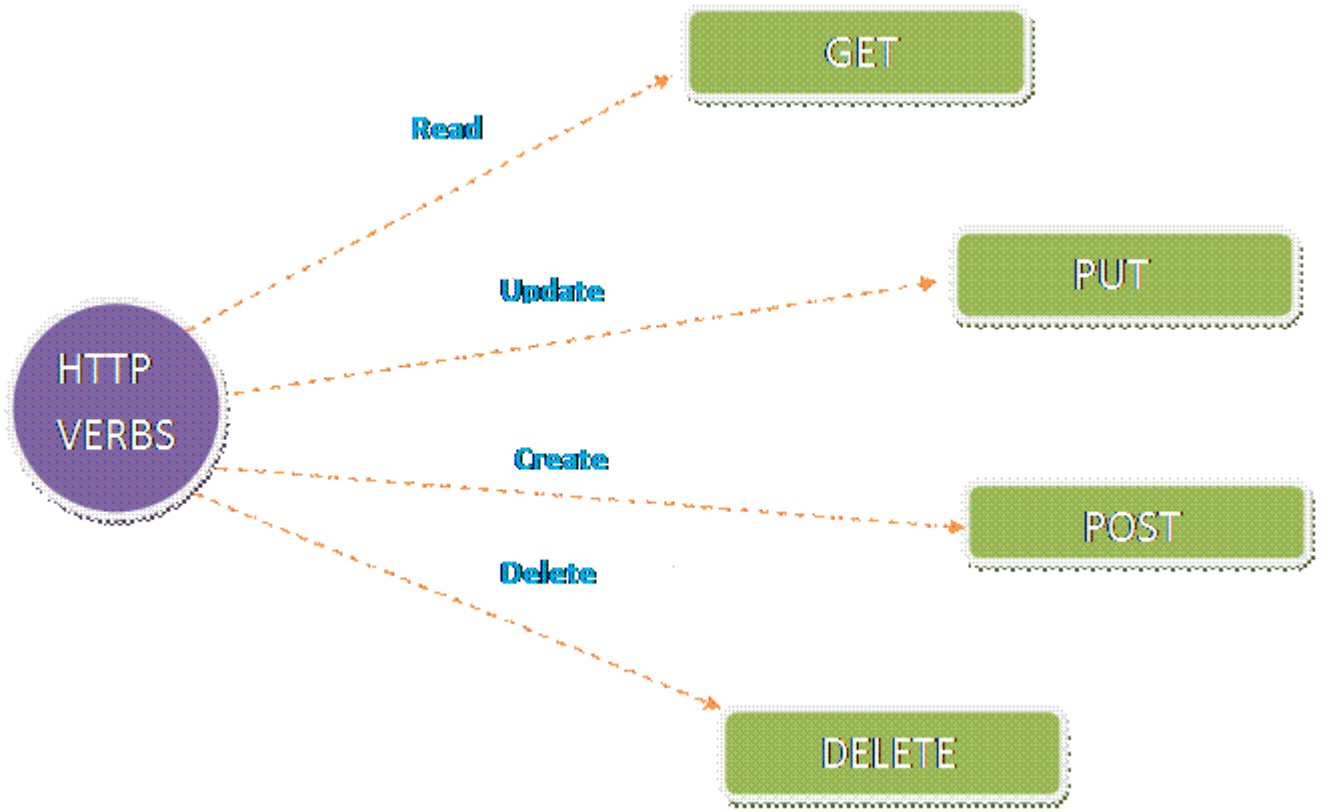
Services Portfolio, related products

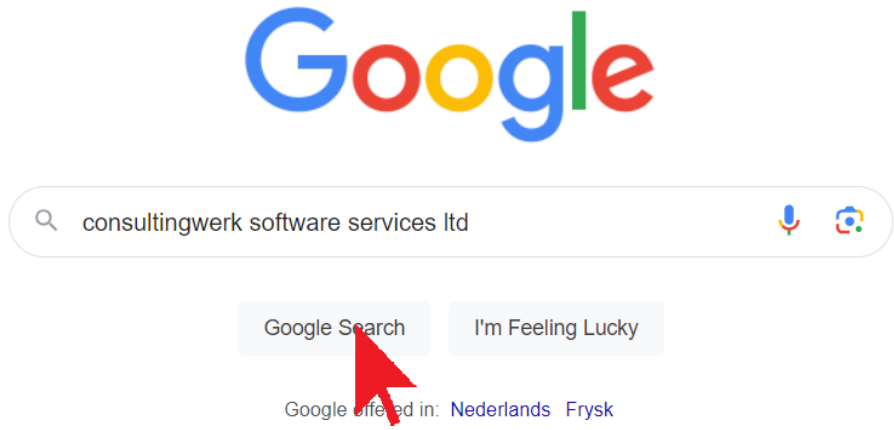
- Protop Database Monitoring
- Combit List & Label
- Web frameworks, e.g. Angular
- .NET
- Java
- ElasticSearch, Lucene
- Amazon AWS, Azure
- DevOps, Docker, Jenkins, ANT, Gradle, JIRA, …
- …

Agenda

- **HTTP Basics**
- **PASOE**
- **Web Handler**

Webserver HTTP verbs





HTTP request methods (basic webserver behavior) I

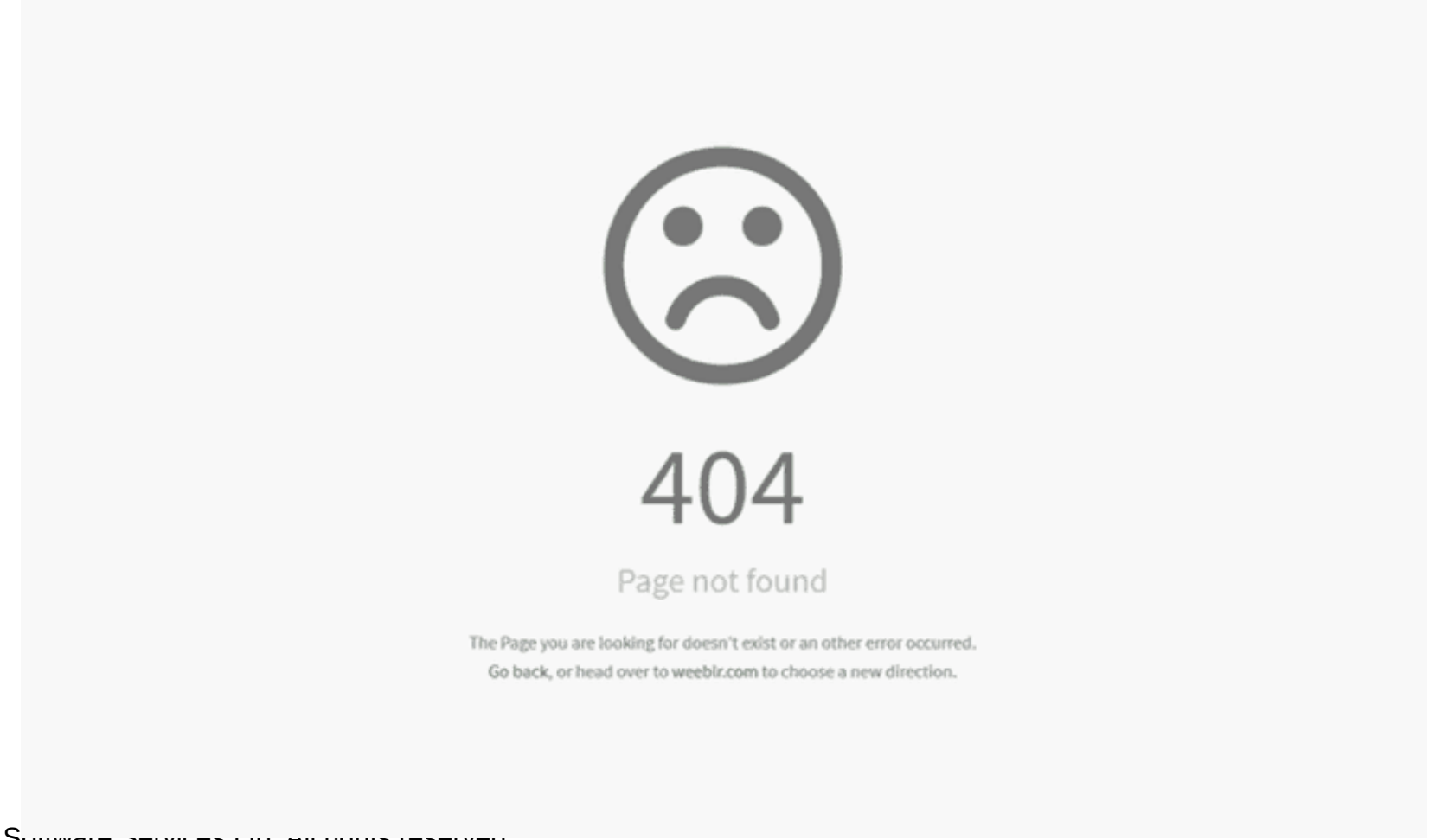
- GET
  - The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

- POST
  - The POST method submits an entity to the specified resource, often causing a change in state or side effects on the server.

HTTP request methods (basic webserver behavior) II

- PUT
  - The PUT method replaces all current representations of the target resource with the request payload.
- DELETE
  - The DELETE method deletes the specified resource.
- PATCH
  - The PATCH method applies partial modifications to a resource.

HTTP Status codes

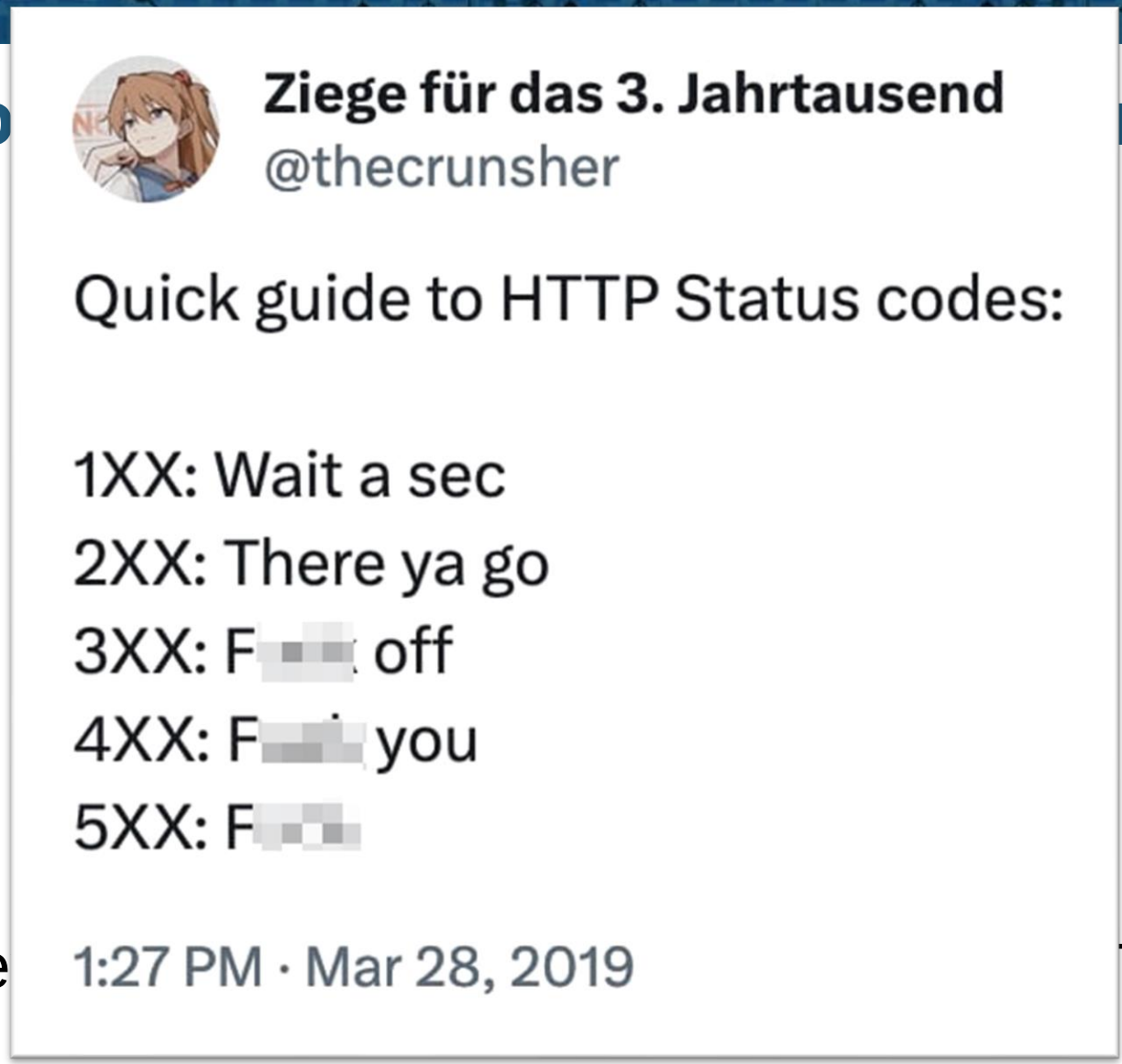


Status codes stuff we already know from browsing the Internet

- 200 OK - Everything went fine
- 201 - Data created
- 400 - Bad Request
- 403 - Forbidden
- 404 - Not Found
- 405 - Method not allowed
- 500 - Internal Server Error
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

Status co m browsing the Internet

- 200 OK
- 201
- 400
- 403
- 404
- 405
- 500
- https://de TTP/Status



HTTP status codes and verbs

- What works well for a plain webserver works well for REST.
- We need to GET / PUT / DELETE / PATCH data too
- We need status codes too

Agenda

- HTTP Basics
- PASOE
- Web Handler

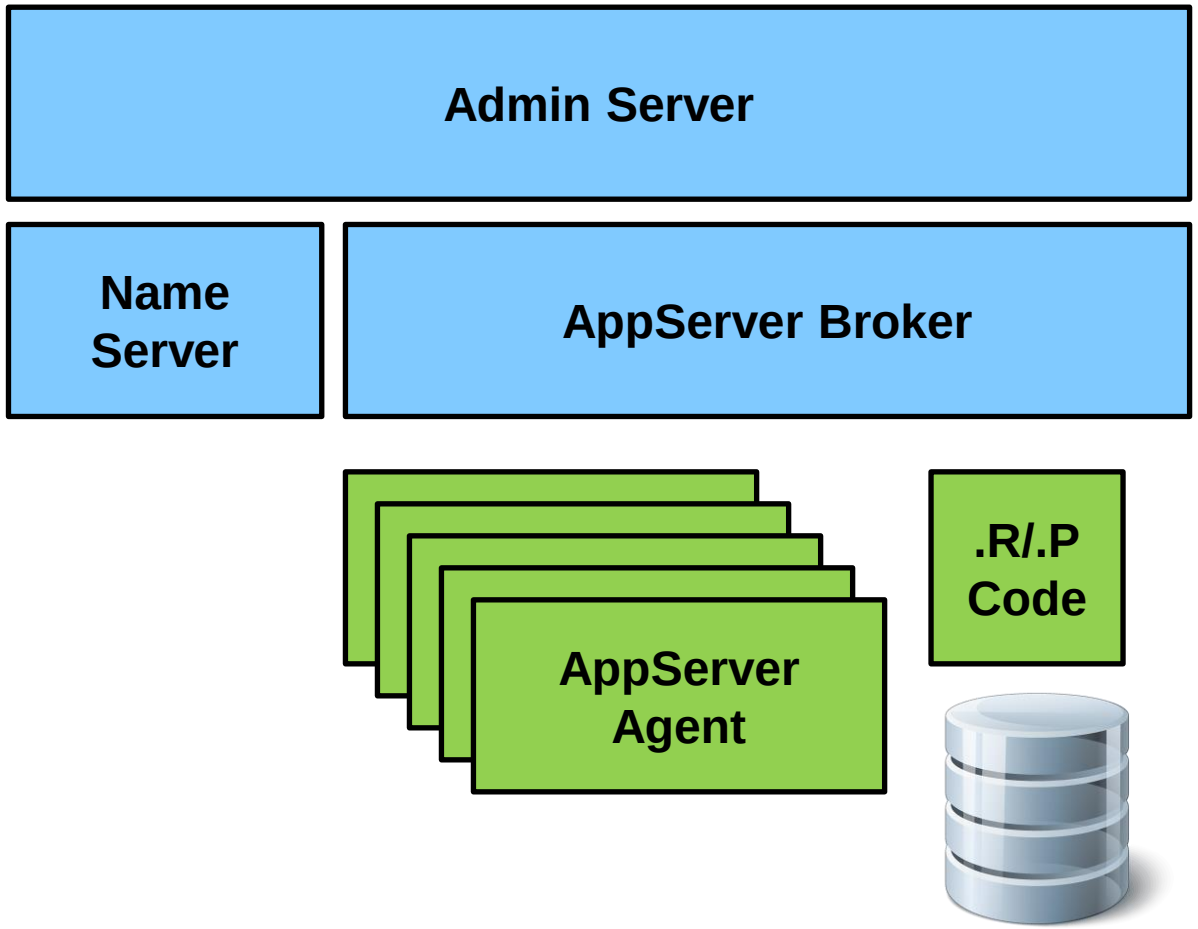
PASOE

- Progress Application Server for OpenEdge
- PASOE is the new AppServer for OpenEdge
- Introduced in OpenEdge 11.5 with APSV, REST and SOAP transport
- WEB transport added in OpenEdge 11.6
- Enhanced in OpenEdge 11.7
- Starting OpenEdge 12.0 the only AppServer for OpenEdge as the classic AppServer is retired

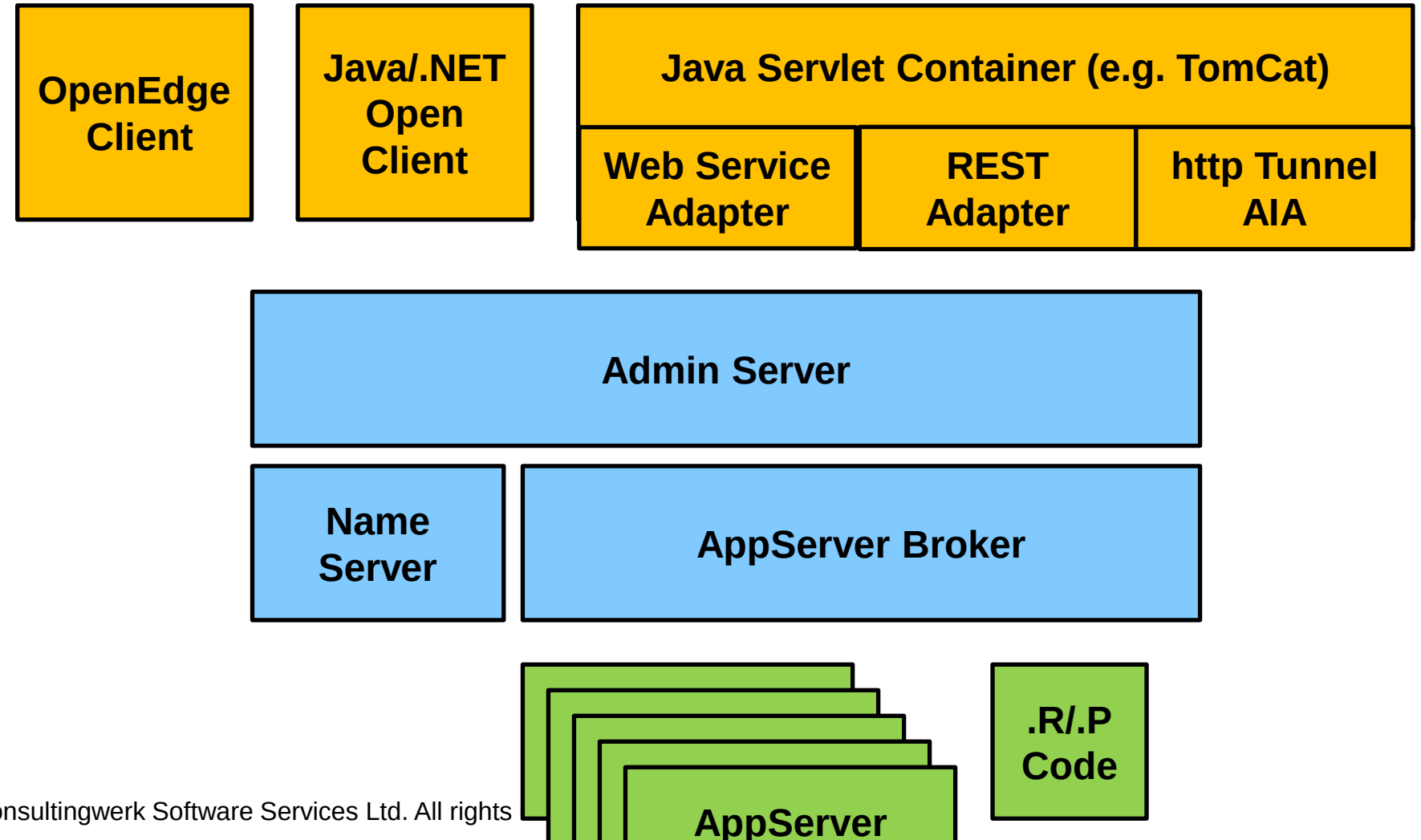
PASOE

- **64 bit only** for AppServer
- 32/64 bit for Clients (e.g. ABL GUI)
- Supported Docker images for development and production
- Build on top of Apache Tomcat – also to leverage wide set of authentication and authorization options from the Tomcat ecosystem
- No specific built-in load-balancing or fail-over solution
- **No dependency on Admin-Server framework or OpenEdge management at all**

Classic AppServer Architecture



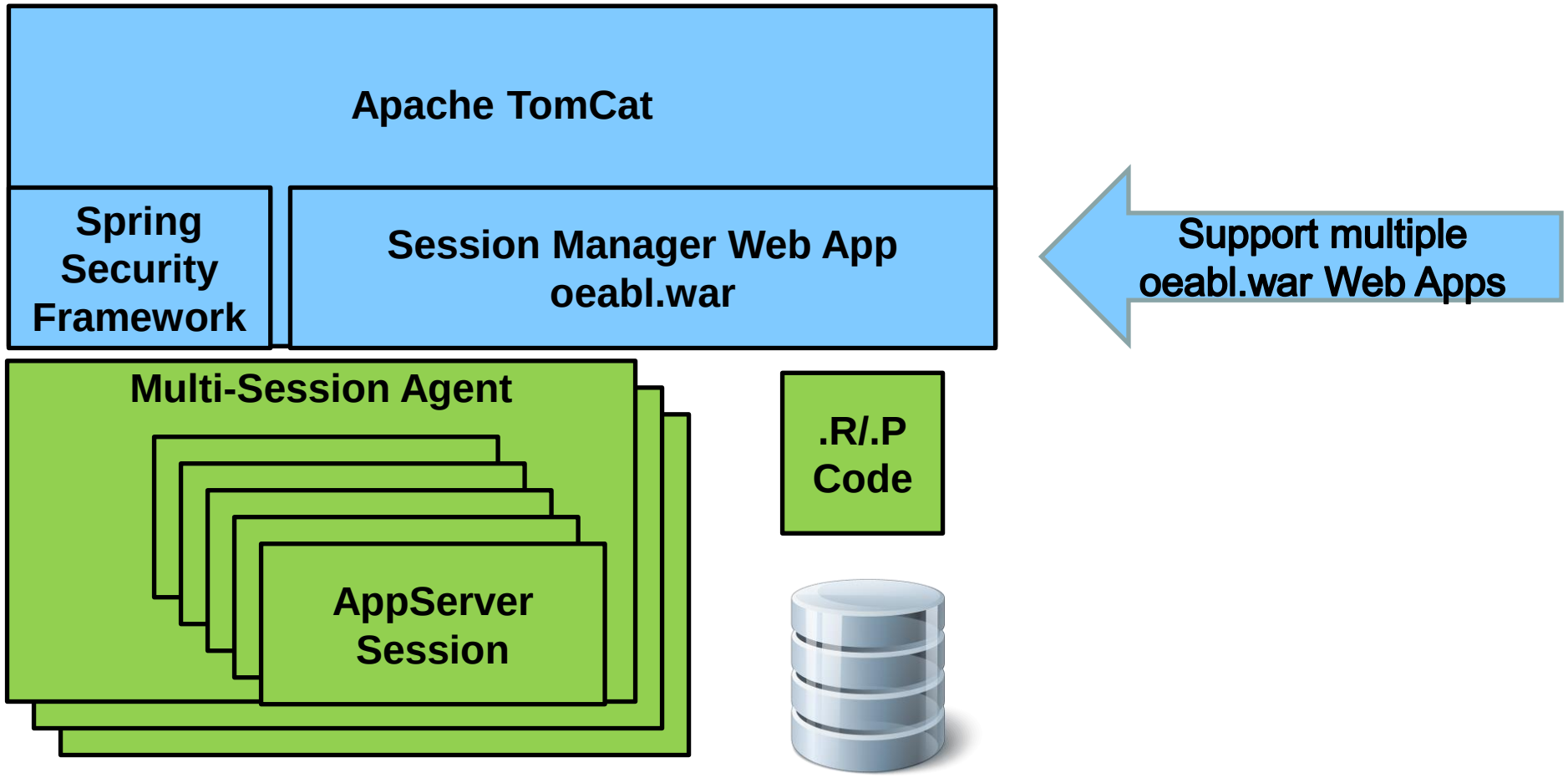
Classic AppServer Architecture



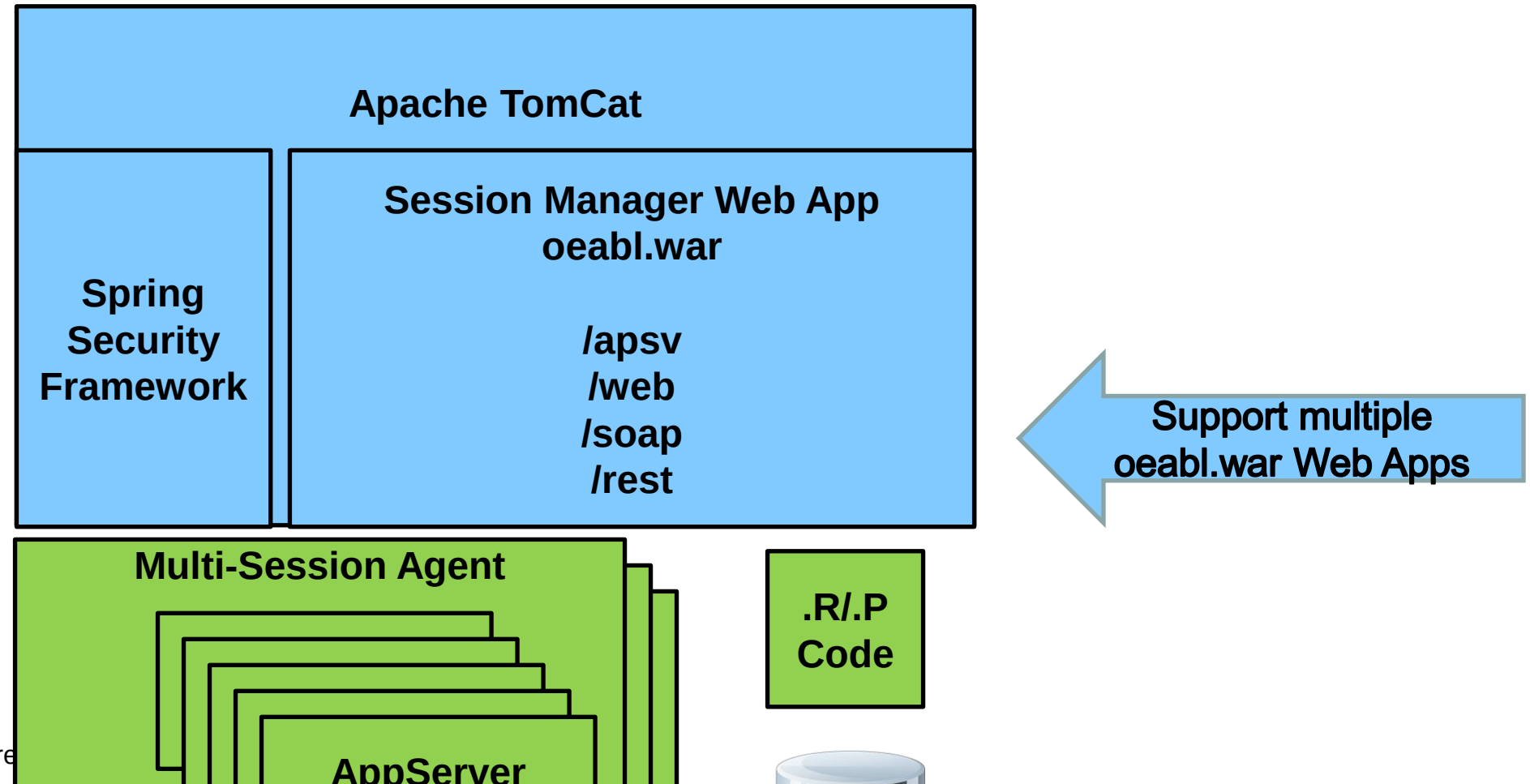
PASOE in contrast

- AppServer Broker replaced with Session Manager Tomcat webapp
- AppServer agent (C++) now supports **multi-session**
- Agent capable of executing multiple requests at a time – in **parallel threads**
- Agent capable of maintaining multiple sessions at a time (default configuration is 200 sessions per agent)
- To scale up (execute more concurrent requests) choice of using more agent processes or threads within the agents' processes
- Scaling more lightweight

PASOE Architecture



PASOE Architecture



Agenda

- http Basics
- PASOE
- **Web Handler**

PASOE Web handler

- Available since OpenEdge 11.6
- Web handlers provide a very flexible way to handle web requests
- Synchronous request-response pattern
- Supports html page generation
- Supports service requests as well
- Flexible enough to provide an alternative to the REST Adapter and Web Services Adapter (SOAP)
- ABL classes, extending OpenEdge.Web.WebHandler

PASOE Web Handler

- PASOE Web Handler provide foundation for WebSpeed support
- Supported programming models
  - CGI Wrapper
  - Embedded SpeedScript
- Support for mapped web objects added in OpenEdge 12.7
- Modern web applications typically don’t rely on server-side HTML page generation anymore
- Web Handler designed to turn PASOE into an API server for modern web applications

PASOE Web Handler

- OOABL Classes
- Inheriting from OpenEdge.Web.WebHandler
- Providing overridable ABL methods for handling HTTP verbs
  - GET → HandleGet
  - POST → HandlePost
  - PUT → HandlePut
  - …
- Shares same object model for request/response as the OpenEdge HTTP Client (available also since OpenEdge 11.6)

OpenEdge.Web.WebHandler

Method Summary

Options	Name
	INTEGER **HandleDelete (IWebRequest)**
	INTEGER **HandleGet (IWebRequest)**
	INTEGER **HandleHead (IWebRequest)**
A	INTEGER **HandleNotAllowedMethod (IWebRequest)**
A	INTEGER **HandleNotImplemented (IWebRequest)**
	INTEGER **HandleOptions (IWebRequest)**
	INTEGER **HandlePatch (IWebRequest)**
	INTEGER **HandlePost (IWebRequest)**
	INTEGER **HandlePut (IWebRequest)**
#	INTEGER **HandleRequest ()**
	INTEGER **HandleTrace (IWebRequest)**

Web handler

- WebSpeed in PASOE brings request handler mapping out of the box (classic Web Speed requires customization of web-disp.p for this)
- Based on configuration in openedge.properties
- New PDSOE project type ABL Web Application creates and registers a single handler
- Additional handlers can be set up in OpenEdge Management

URL Mapping

- Configuration based
- Tomcat parses request URI for patterns
- http://localhost/web/Customers/1
- Easy to create „rest-style“ URI‘s
- Higher ranking in search engines compared to classic WebSpeed
  http://localhost/cgi-bin/cgiip.exe/Customers.w?CustNum=1

- Web handlers are specialized ABL classes

WebHandler URL mapping

- openedge.properties

```
[restfulpasoe.ROOT.WEB]
    adapterEnabled=1
    defaultCookieDomain=
    defaultCookiePath=
    defaultHandler=OpenEdge.Web.CompatibilityHandler
    handler1=Customer.Customers: /Salesreps/{Salesrep}/Customers
    handler2=Salesrep.Salesreps: /Salesreps/{Salesrep}
    handler3=Salesrep.Salesreps: /Salesreps
    handler4=Customer.Customers: /Customers/{CustNum}
    handler5=Customer.Customers: /Customers
    srvrAppMode=development
    srvrDebug=1
    wsRoot=/static/webspeed
```

URL Mapping in openedge.properties

- Section [<ABL app name>.<web app name>.WEB]
- Handler numbers starting from 1, no gaps, no duplicates!
- Handler entries are processed in order, looking for ~BEGINS
- First matching handler (based on URL) is used to process request

- Can be hard to determine which handler will be used
  - Log messages can help
- Hard to update via scripts

<service>.handlers file – starting OpenEdge 12.2

- Alternative to entries in openedge.properties
- JSON file in webapps/<web app name>/WEB-INF/adapters/web/<service>
- File named <service>.handlers
  - ROOT.handlers a special case / direct replacement for openedge.properties values

Webapps/ROOT/WEB-INF/adapters/web/ROOT/ROOT.handlers

ROOT.handlers

C: > Work_STREAM > SmartComponentLibrary > Develop127 > smartpas_stream > webapps > ROOT > WEB-INF > adapters > web > ROOT > ROOT.handler

```
{
    "version": "2.0",
    "serviceName": "",
    "handlers": [
        {
            "class": "Consultingwerk.OERA.JsdoGenericService.WebHandler.CatalogWebHandler",
            "uri": "/Catalog/{EntityName}",
            "enabled": true
        },
        {
            "class": "Consultingwerk.OERA.JsdoGenericService.WebHandler.CatalogsWebHandler",
            "uri": "/Catalogs/{PackageName}",
            "enabled": true
        },
        {
            "class": "Consultingwerk.OERA.JsdoGenericService.WebHandler.CountWebHandler",
            "uri": "/Resource/{EntityName}/count",
            "enabled": true
        },
        {
            "class": "Consultingwerk.OERA.JsdoGenericService.WebHandler.ResourceSubmitWebHandler",
            "uri": "/Resource/{EntityName}/SubmitData",
            "enabled": true
```

Samples

- https://github.com/consultingwerk/RESTful-Samples

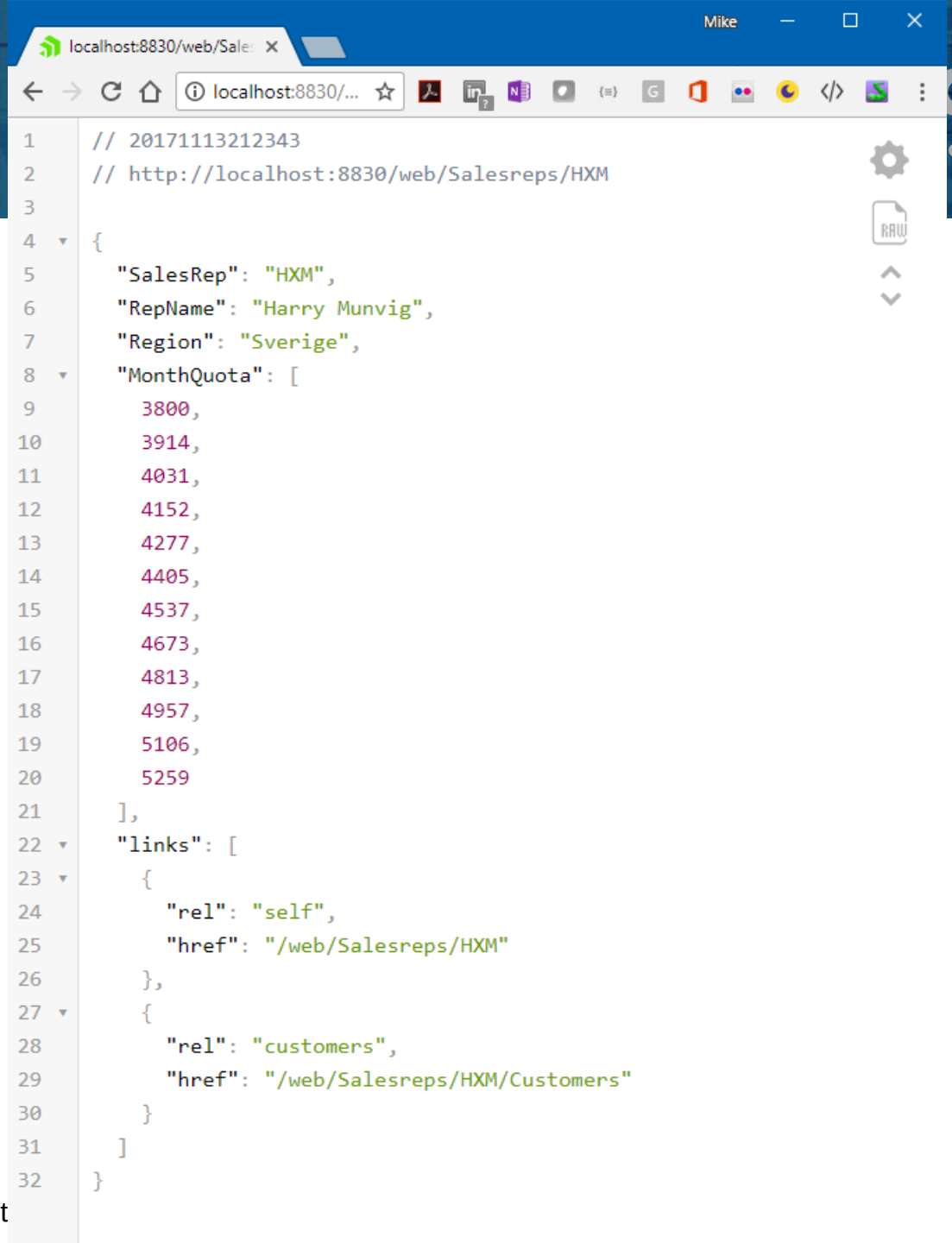
Providing the response

- Instance of OpenEdge.Web.WebResponse
- Property for ContentType (application/json, text/html, text/plain, image/png, etc.)
- Property for response payload: Entity
- Entity is of type Progress.Lang.Object
- Supports
  - Progress.Json.ObjectModel.JsonObject, JsonArray
  - OpenEdge.Core.String
  - OpenEdge.Core.Memptr (binary data)
  - OpenEdge.Core.WidgetHandle (XML document)

```
METHOD OVERRIDE PROTECTED INTEGER HandleGet (poRequest AS OpenEdge.Web.IWebRequest):

    DEFINE VARIABLE cSalesrep   AS CHARACTER     NO-UNDO .
    DEFINE VARIABLE oJson       AS JsonConstruct NO-UNDO .
    DEFINE VARIABLE oResponse   AS IHttpResponse NO-UNDO .

    ASSIGN oResponse              = NEW OpenEdge.Web.WebResponse ()
           oResponse:ContentType = 'application/json':U
           cSalesrep              = poRequest:GetPathParameter("Salesrep") .

    IF cSalesrep > "" THEN
        oJson = THIS-OBJECT:GetSalesrep (cSalesrep) .
    ELSE
        oJson = THIS-OBJECT:GetSalesreps () .

    oResponse:Entity = oJson .

    THIS-OBJECT:WriteResponse (oResponse).

    RETURN 200 .

    CATCH err AS Progress.Lang.Error:
        IF TYPE-OF (err, NotFoundException) THEN DO:
            oJson = NEW JsonObject () .
            CAST (oJson, JsonObject):Add ("error", err:GetMessage(1)) .
            oResponse:Entity = oJson .
            THIS-OBJECT:WriteResponse (oResponse).

            RETURN 404 . /* not found */
        END.
        ELSE DO:
            oJson = NEW JsonObject () .
            CAST (oJson, JsonObject):Add ("error", err:GetMessage(1)) .
            oResponse:Entity = oJson .
            THIS-OBJECT:WriteResponse (oResponse).

            RETURN 500 . /* internal server error */
        END.
    END CATCH.
```

```
METHOD PROTECTED JsonObject GetSalesrep (pcSalesrep AS CHARACTER):

    DEFINE VARIABLE oJson AS JsonObject NO-UNDO.

    EMPTY TEMP-TABLE ttSalesrep.

    FIND Salesrep WHERE Salesrep.Salesrep = pcSalesrep NO-LOCK NO-ERROR .

    IF NOT AVAILABLE Salesrep THEN
        UNDO, THROW NEW NotFoundException () .

    oJson = NEW JsonObject () .

    CREATE ttSalesrep .
    BUFFER-COPY Salesrep TO ttSalesrep .

    oJson:Read (BUFFER ttSalesrep:HANDLE) .

    oJson:Add ("links", THIS-OBJECT:GetLinks (ttSalesrep.SalesRep)) .

    RETURN oJson .

    FINALLY:
        EMPTY TEMP-TABLE ttSalesrep.
    END FINALLY.

END METHOD.
```

```
METHOD PROTECTED JsonArray GetLinks (pcSalesrep AS CHARACTER):

    DEFINE VARIABLE oLinks AS JsonArray  NO-UNDO .
    DEFINE VARIABLE oLink  AS JsonObject NO-UNDO .

    oLinks = NEW JsonArray () .
    oLink = NEW JsonObject () .

    oLink:Add ("rel", "self") .
    oLink:Add ("href", SUBSTITUTE ("/web/Salesreps/&1", pcSalesrep)) .

    oLinks:Add (oLink) .

    oLink = NEW JsonObject () .

    oLink:Add ("rel", "customers") .
    oLink:Add ("href", SUBSTITUTE ("/web/Salesreps/&1/Customers", pcSalesrep)) .

    oLinks:Add (oLink) .

    RETURN oLinks .

END METHOD.
```



Support for different media types

- Based on properties ContentType and Entity a Web Handler can support any media type
- To return images:
  - ContentType: image/png
  - Entity: OpenEdge.Core.Memptr instance wrapping a MEMPTR with image data
- Support for multi-part responses by returning instance of OpenEdge.Net.MultipartEntity

Sample: Web Handler to return Images (1/6)

- GET http://localhost:8820/web/Image/**Test/SampleImage.png**
- Web Handler retrieves file name from URL path (not query string parameter or path parameters)

```
ASSIGN oResponse              = NEW OpenEdge.Web.WebResponse ().

/* PathInfo returns URL part after /web, trim "/Image/" from path info */
ASSIGN cFileName  = SUBSTRING (poRequest:PathInfo, 8, -1, "CHARACTER":U)
       cExtension = FileHelper:FileExtension (cFileName).
```

Sample: Web Handler to return Images (2/6)

- File extension used, to determine ContentType

```
CASE cExtension:
    WHEN "png":U THEN
        ASSIGN cContentType  = "image/png":U .
    WHEN "gif":U THEN
        ASSIGN cContentType  = "image/gif":U .
    WHEN "jpg":U OR WHEN "jpeg":U THEN
        ASSIGN cContentType  = "image/jpeg":U .
    WHEN "ico":U THEN
        ASSIGN cContentType  = "image/x-icon":U .
    OTHERWISE
        UNDO, THROW NEW InvalidParameterValueException
                ("FileName":U,
                 SUBSTITUTE ("Unsupported file extension &1.",
                             cExtension)) .
END CASE .
```

Sample: Web Handler to return Images (3/6)

- Load image into MEMPTR
- Create instance of OpenEdge.Core.Memptr
- OpenEdge.Core.Memptr is assigned to Entity property
  - Entity is of type Progress.Lang.Object, to support „any“ type of payload

```
FILE-INFORMATION:FILE-NAME = cFileName.

COPY-LOB FILE FILE-INFORMATION:FULL-PATHNAME TO oMemptr.

oPicture = NEW OpenEdge.Core.Memptr (oMemptr).
```

Sample: Web Handler to return Images (4/6)

- Client may request multi-part response through „Accept“ header

```
IF poRequest:HasHeader('Accept':U) AND
   poRequest:GetHeader('Accept':U):VALUE EQ 'multipart/form-data':U THEN DO:
    ASSIGN oEntity = NEW OpenEdge.Net.MultipartEntity ()
           oEntity:Boundary = GUID
           oResponse:Entity      = oEntity
           oResponse:ContentType = 'multipart/form-data':U

           oPart = NEW OpenEdge.Net.MessagePart (cContentType, oPicture) .

           oEntity:AddPart(oPart).
END.
```

OpenEdge.Core.Memptr with image data assigned to message part

Sample: Web Handler to return Images (5/6)

- When multi-part is not requested, return Image as oEntity

```
ELSE DO:
     ASSIGN oResponse:Entity      = oPicture
            oResponse:ContentType = cContentType.
END.
```

Sample: Web Handler to return Images (6/6)

- Return oResponse to Browser, may be multi-part entity or a single image
- Don‘t forget to deallocate memory in FINALLY block!

```
oResponse:StatusCode = INTEGER (OpenEdge.Net.HTTP.StatusCodeEnum:OK).

THIS-OBJECT:WriteResponse (oResponse).

RETURN 0.

FINALLY:
    SET-SIZE (oMemptr) = 0 .
END FINALLY.
```

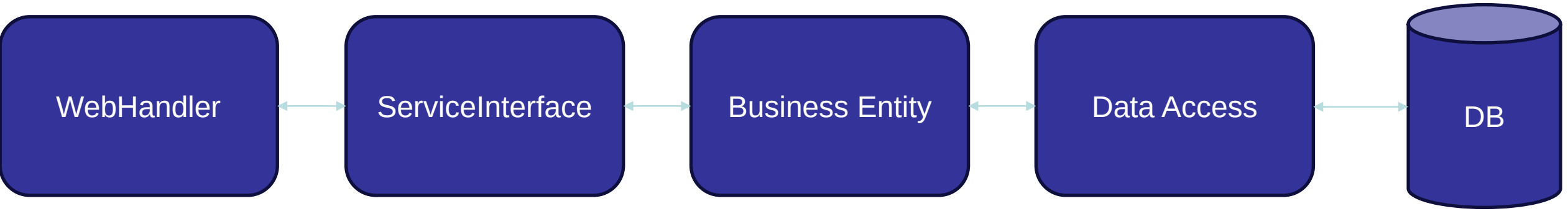
Alternative: Dynamic implementation

- When implementing a large number of RESTful resources, this will require a lot of similar code and configuration
- A dynamic approach may be advised

- In the SmartComponent Library framework, we have implemented this based on a single reusable WebHandler and Annotation based configuration
- https://documentation.consultingwerkcloud.com/display/SCL/RESTful+services

SmartComponent Library and Web Handlers

- Webhandler to access the entities: /Entities
- Sample REST endpoint http://localhost:8020/web/Entities/Customers/1



Annotation based configuration

@RestAddress annotations

```
@RestAddress (type="record", address="/Customers/~{CustNum}", tables="eCustomer,eSalesrep", id="CustNum",
              fields="eCustomer.*,eSalesrep.RepName,eSalesrep.Region", canRead="true", canUpdate="true", canDelete
              childAddresses="eSalesrep:/Salesreps/~{SalesRep}",
              links="orders:/Customers/~{CustNum}/Orders,salesrep:/Salesreps/~{SalesRep}").

@RestAddress (type="collection", address="/Customers", tables="eCustomer", id="CustNum",
              fields="Name,City,Country", canCreate="true",
              links="orders:/Customers/~{CustNum}/Orders,salesrep:/Salesreps/~{SalesRep}").
```

Business Entities define the details of the RESTful access through annotations (see The Annotation based Type Descriptor).

Richardson Maturity Model

- Steps toward the glory of REST
  - Level 1 - Resources
  - Level 2 - HTTP Verbs
  - Level 3 - Hypermedia Controls


```
Comments: "27/09/2017 20:23:12,627+02:00",
Fax: "",
EmailAddress: "info@lift-tours.com",
Flags: "C",
eSalesrep: {
    url: "http://localhost:30010/web/Entities/Salesreps/HXM",
    RepName: "Harry Munvig 333",
    Region: "West"
},
links: [
  - {
        rel: "orders",
        href: "http://localhost:30010/web/Entities/Customers/1/Orders"
    },
  - {
        rel: "salesrep",
        href: "http://localhost:30010/web/Entities/Salesreps/HXM"
    }
]
```


- https://www.martinfowler.com/articles/richardsonMaturityModel.html

Be nice and RESTfull

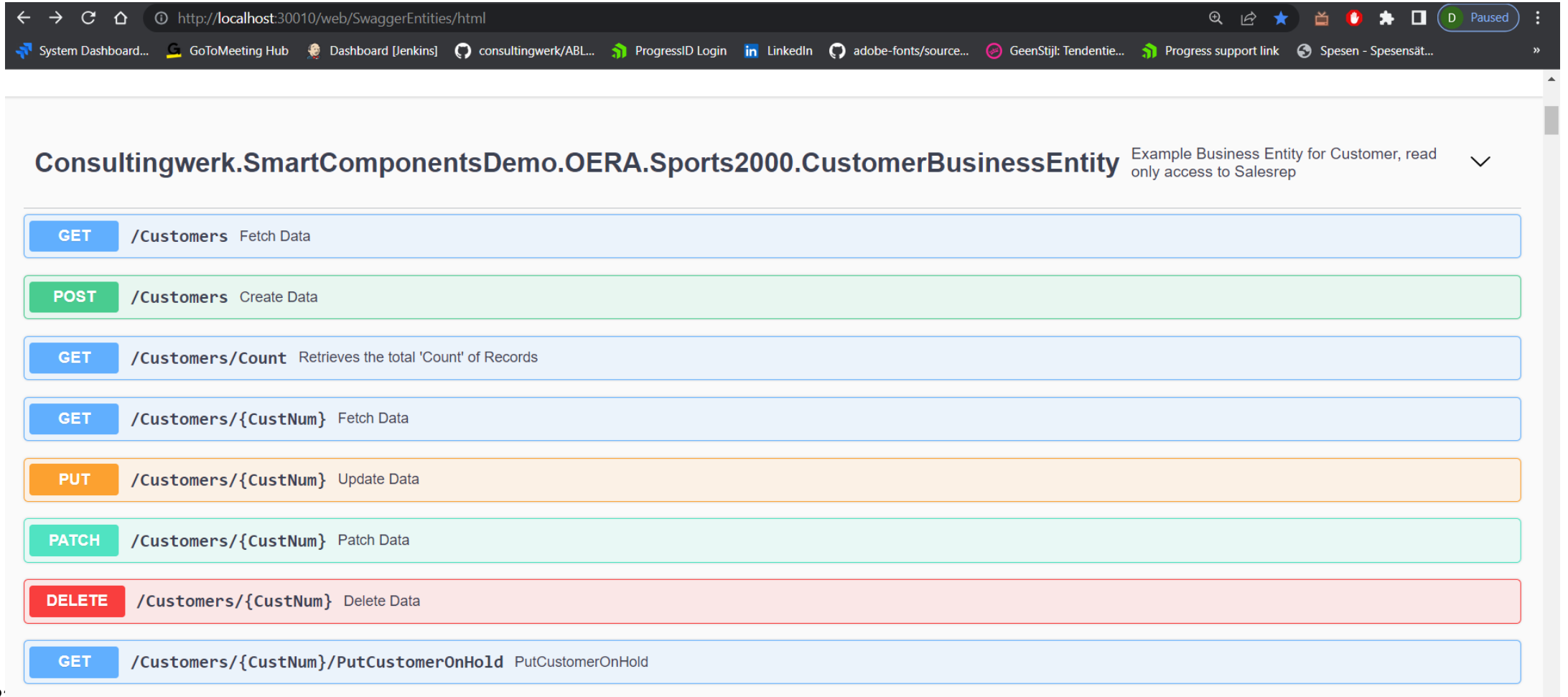
- Follow (defacto) standards of the internet and REST
- Don’t return errors as 200 OK
- REST endpoints are plural
- Both for a collection as for a Single record.

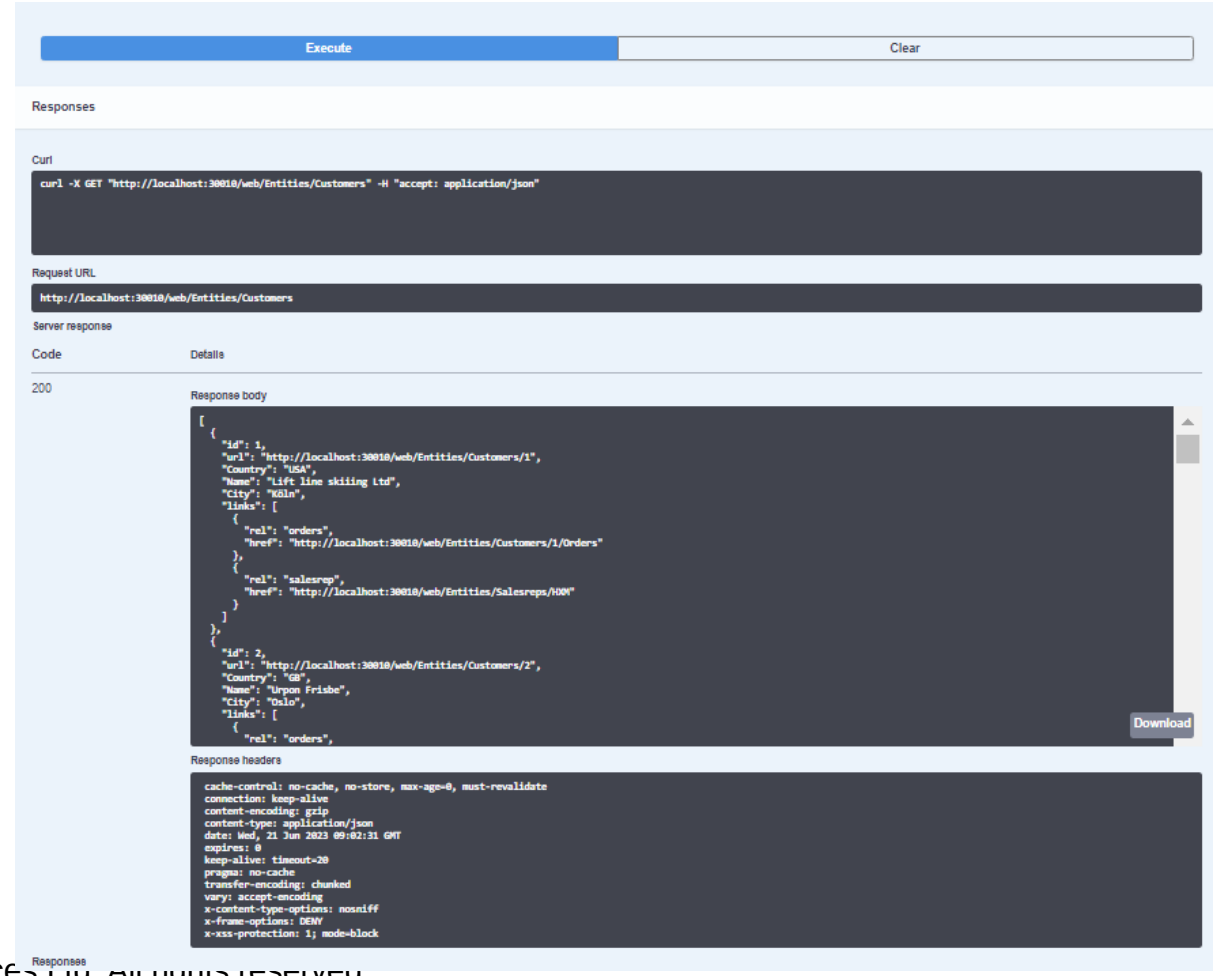
Swagger Open API

- Generate a JSON with REST Endpoints from your code.
- There is a Swagger generator which generates the Swagger page
  - https://swagger.io/tools/open-source/getting-started/

- Of course this is standard integrated in the SmartComponent Library!

Swagger OpenAPI





Can we only use JSON?

- No
  - JSON / XML / Binary / HTML can all be returned.

```
COPY-LOB FROM FILE FileHelper:FindFile(cFullPath) TO pData.

oData     = NEW  OpenEdge.Core.Memptr(pData).

poResponse:SetHeader("Content-Disposition", SUBSTITUTE ("attachment; filename=~"&1~"", cFileName)).
poResponse:ContentType    = "application/octet-stream".
poResponse:ContentLength = oData:Size.
poResponse:Entity         = oData.

THIS-OBJECT:WriteResponse(poResponse).
```

Questions


Consultingwerk
software architecture and development