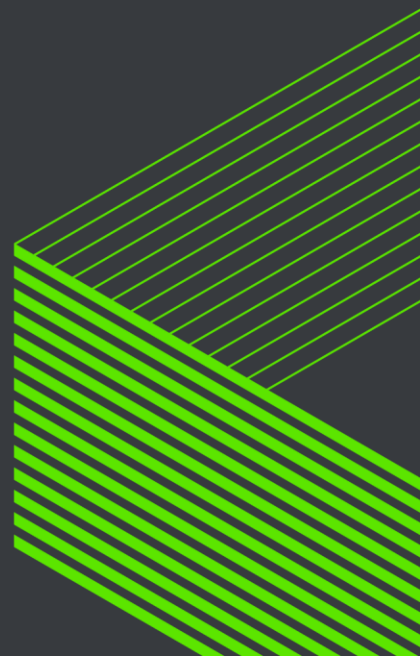
PUG CHALLENGE EXCHANGE AMERICAS

PAS for OpenEdge Web Handlers: A Deep Dive

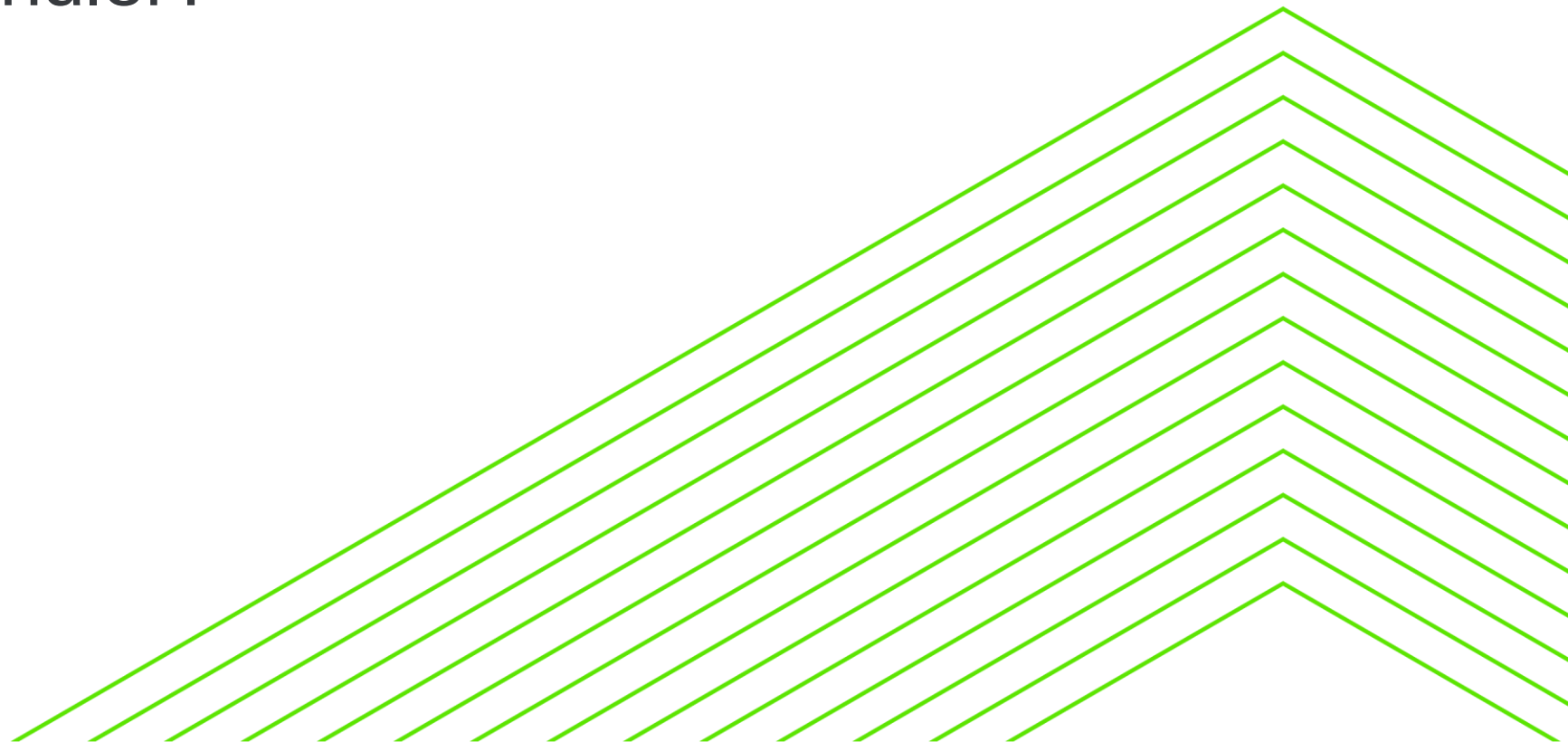
Peter Judge
pjudge@progress.com

Overview

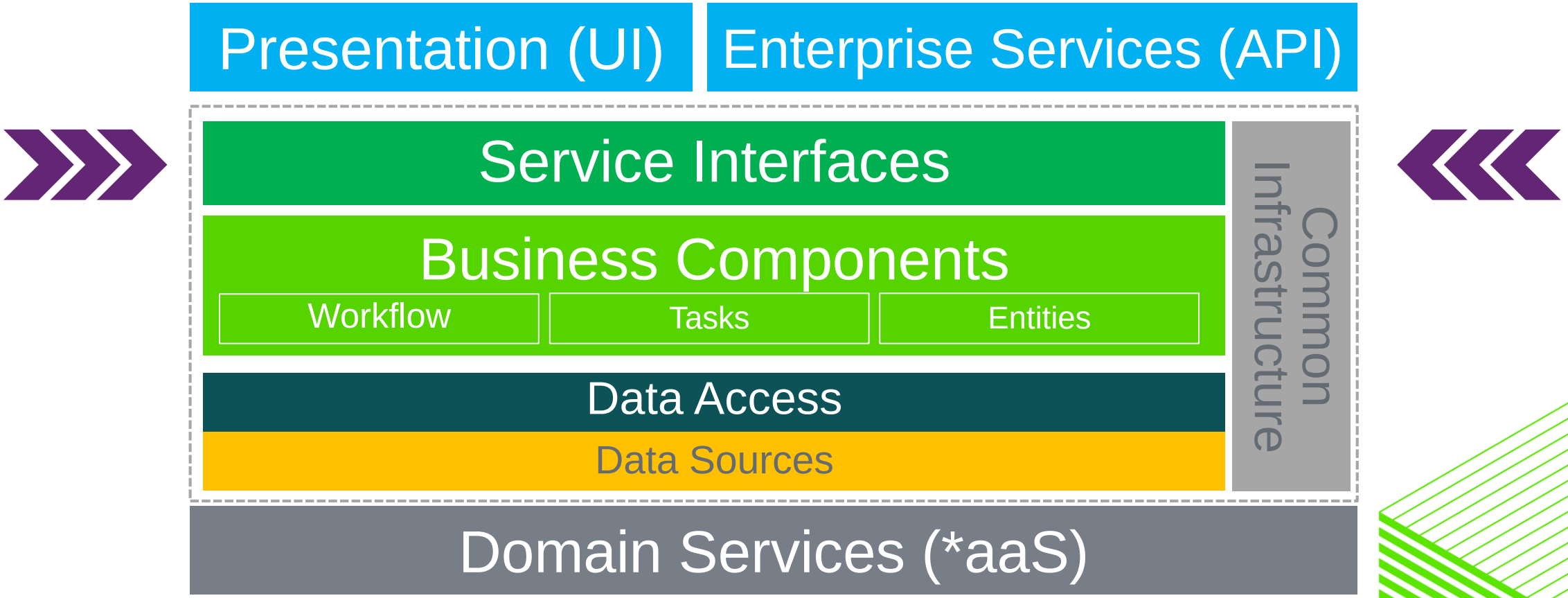
- What is a web handler
- What can you do with it
- Where should you use it



What is a Web Handler?



OpenEdge Reference Architecture (OERA)



Service Interfaces

- Service interfaces provide the translation layer between a request and the underlying business services
  - Route requests
  - Compose responses
  - Error handling
- Provide authentication and authorization
- Translate input / output formats to and from domain model
  - JSON/XML/text into ProDataSet/Temp-Table/objects
  - Data validation
- Service interfaces are NOT business domain services or logic (like tax calculations, master data maintenance, order entry, etc.)

Service Interface Approaches - REST

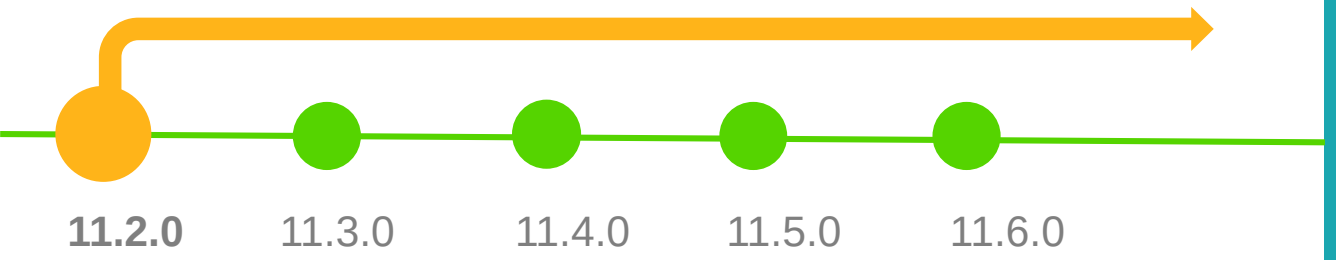
WebHandler

REST (Mapped RPC)

Data Object (REST)

- Formerly REST Services
- Use GUI tool to map HTTP elements to program input/outputs
- Flexible in URI paths

- Formerly Mobile Services
- Annotate certain methods (w/ particular signatures)
- Very prescriptive
  - Programming model
  - URI paths
- Creates Data Service Catalog as public API



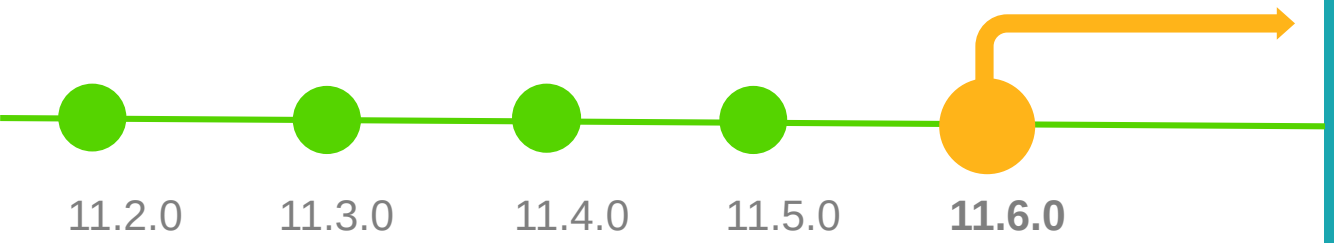
Service Interface Approaches – WEB

WebHandler

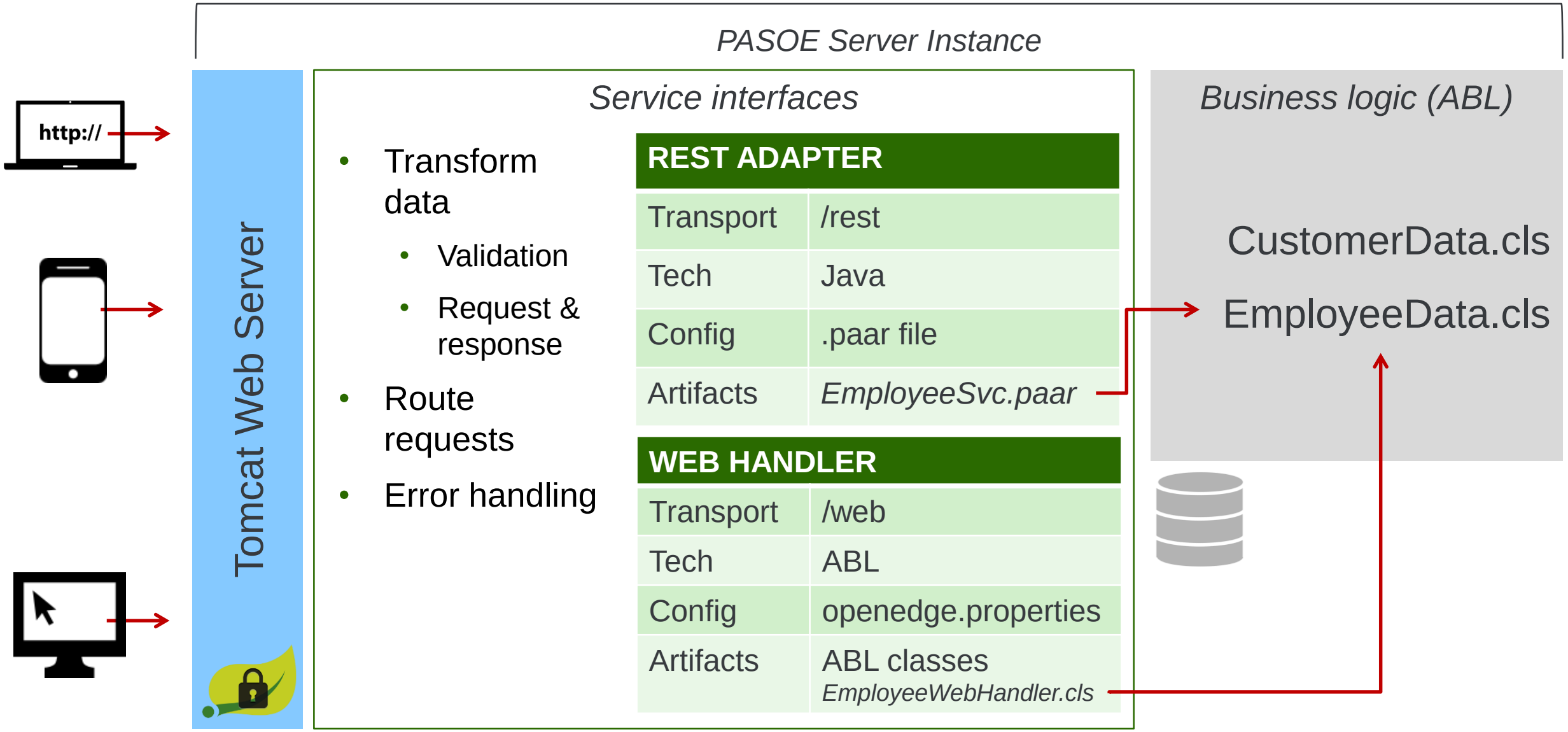
REST (Mapped RPC)

Data Object (REST)

- Associate an OOABL WebHandler class with a URI pattern
- Very flexible
- Do whatever you want in ABL code
- URI is all yours too
- Create your own or use one of the in-the-box WebHandlers
  - Extend and/or replace existing behaviour



PASOE Architecture (11.6.0+)



What is a web handler?

- Tasks
  - Routing: what program & function should be run
  - Parameter mapping: convert HTTP into ABL and back

- OOABL implementation of `Progress.Web.IWebRequest` ← for advanced use only

  `OpenEdge.Web.WebHandler` ← use this for new code

  `OpenEdge.Web.CompatibilityHandler` ← 'classic' WebSpeed

  `OpenEdge.Web.DefaultWebHandler` ← locked-out version

  `OpenEdge.Web.DataObject.DataObjectHandler` ← general purpose w/JSON config

- OpenEdge.Web & OpenEdge.Net packages in $DLC/tty/netlib/OpenEdge.Net.pl
- API doc at https://documentation.progress.com/output/oehttpclient/oe117

Routing requests

HTML

```
GET /schedule HTTP/1.1
Host: pugchallenge.org
Accept: application/json


DELETE /schedule HTTP/1.1
Host: pugchallenge.org
Accept: application/json


POST /register HTTP/1.1
Host: pugchallenge.org
Content-Type: application/json


POST /feedback/session HTTP/1.1
Host: pugchallenge.org
Content-Type: application/json
```

ABL

Web.ScheduleHandler

provide_feedback.p

manage_sessions.p

register_attendee.p

Web.FeedbackHandler

Routing requests

HTML

```
GET /schedule HTTP/1.1
Host: pugchallenge.org
Accept: application/json


DELETE /schedule HTTP/1.1
Host: pugchallenge.org
Accept: application/json


POST /register HTTP/1.1
Host: pugchallenge.org
Content-Type: application/json


POST /feedback/session HTTP/1.1
Host: pugchallenge.org
Content-Type: application/json
```

ABL

Web.ScheduleHandler

provide_feedback.p

WH

manage_sessions.p

register_attendee.p

Web.FeedbackHandler

Routing requests

HTML

```
GET /schedule HTTP/1.1
Host: pugchallenge.org
Accept: application/json


DELETE /schedule HTTP/1.1
Host: pugchallenge.org
Accept: application/json


POST /register HTTP/1.1
Host: pugchallenge.org
Content-Type: application/json


POST /feedback/session HTTP/1.1
Host: pugchallenge.org
Content-Type: application/json
```

ABL

Web.ScheduleHandler

provide_feedback.p

WH

manage_sessions.p

register_attendee.p

Web.FeedbackHandler

Which handler is used?

handler*N*=ooabl.type.name : <relative-uri>

- <relative-uri> `1* [ "/" [token|text] ]`
  - Relative to **/web** ← the transport path
    - MUST have leading /
    - Either Text `Customers`
    - Or Tokens `{CustomerName}` or `{pathparam: regex}`

- Matched in (numeric) order and then by best match
  - Handlers may be reused for differing paths

- Each webapp has a default for no-match-found
  - In-the-box uses `OpenEdge.Web.DefaultWebHandler` or `CompatibilityHandler`

Which handler is used?

Client

http://localhost:8810/SportsApp/web/PayrollSvc/catalog/ — 1

http://localhost:8810/SportsApp/web/PayrollSvc/data/Employee — 2

http://localhost:8810/SportsApp/web/PayrollSvc/ui — default

Config

```
[mediaresource.SportsApp.WEB]
    defaultHandler = OpenEdge.Web.DefaultWebHandler
    handler1       = Sports.SchemaHandler        : /{resources}/catalog
    handler2       = Sports.DataHandler          : /{resources}/data/{service}
    handler3       = Sports.DataHandler          : /{resources}/data/
    handler4       = App.DocumentHandler         : /forms/{form-name}
    handler5       = Sports.SportsHandler        : /CustomerSvc/catalog
```

Where does my WebHandler live?

- We consider it a Service Interface
  - Should be in the webapp `WEB-INF/openedge` B
- But it's Just ABL so can be anywhere on PROPATH

B
A

```
[AppServer.Agent.mediaresource]
PROPATH=
${CATALINA_BASE}/webapps/ROOT/WEB-INF/openedge,

${CATALINA_BASE}/openedge,

/application/path,
${DLC}/tty,
${DLC}/tty/netlib/OpenEdge.Net.pl
```

```
${CATALINA_BASE}
+---bin
+---common
+---conf
        openedge.properties
+---logs
+---openedge                                    A
|   \---sports
|           EmployeeBE.cls
+---temp
+---webapps
|   +---ROOT                                    B
|   |   +---META-INF
|   |   +---static
|   |   |   +---auth
|   |   |   +---error
|   |   |   +---images
|   |   \---WEB-INF
|   |       +---openedge                        B
|   |       |   \---sports
|   |       |           ImageWebHandler.cls
|   |       \---tlr
\---workA
```

Coding a web handler (recommended approach)

1. Create a new class that inherits from `OpenEdge.Web.WebHandler`
2. You must implement 3 methods …
   i. `HandleGet`
   ii. `HandleMethodNotImplemented`
   iii. `HandleNotAllowed`
3. … and you'll want to implement more
   iv. `Handle<http-method> (eg. HandlePost)`
   v. `HandleException`

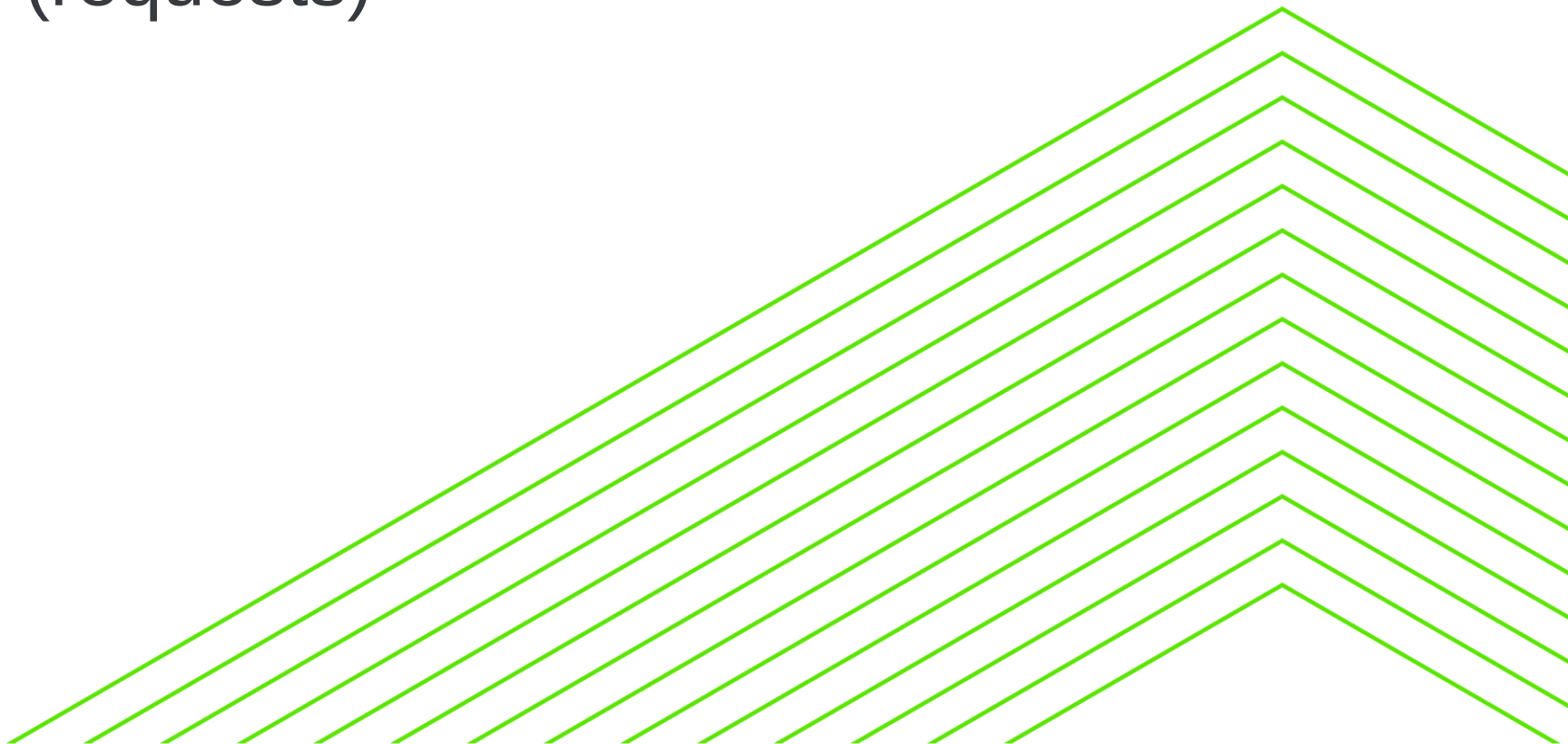
Coding a web handler (generated code)

```
METHOD OVERRIDE PROTECTED INTEGER HandleGet(INPUT poRequest AS OpenEdge.Web.IWebRequest ):
  DEFINE VARIABLE oResponse AS OpenEdge.Net.HTTP.IHttpResponse NO-UNDO.
  DEFINE VARIABLE oWriter   AS OpenEdge.Web.WebResponseWriter  NO-UNDO.
  DEFINE VARIABLE oBody     AS OpenEdge.Core.String            NO-UNDO.

  /* The WebResponse body is a wrapper around an entire HTTP response message. It contains a status code and
     reason; headers;  cookies and a message body. API-level doc for th                    at
     https://documentation.progress.com/output/oehttpclient/ */
  ASSIGN oResponse             = NEW OpenEdge.Web.WebResponse()
         oResponse:StatusCode = INTEGER(StatusCodeEnum:OK)
         /* This body object can be a string or something else
         oBody = NEW OpenEdge.Core.String('Hello Truman')
         oResponse:Entity        = oBody
         oResponse:ContentType   = 'text/plain':u /* HTTP messages requ
         oResponse:ContentLength = oBody:Size     /* ContentLength is g
  /* The WebResponseWriter ensures that the status line and all headers
     body/entity. */
  ASSIGN oWriter = NEW WebResponseWriter(oResponse).
  oWriter:Open().
  /* The Progress.IO.OutputStream Write() methods take multiple overloads, for a variety of data types. See
     the doc for more information. */
  oWriter:Write(oBody:Value).
  /* Finish writing the response message */
  oWriter:Close().
  /* A response of 0 means that this handler will build the entire response; a non-zero value is mapped to a
     static handler in the webapp's /static/error folder. The mappings are maintained in the webapps's
     WEB-INF/web.xml
     A predefined set of HTTP status codes is provided in the OpenEdge.Net.HTTP.StatusCodeEnum enumeration */
  RETURN 0.
END METHOD.
```

Do something useful here

- Run a .P
- Read a file from disk
- Call your mother

Mapping gazinters (requests)



gazinters: goes-inters,
goes into's,
inputs

Making sense of HTTP requests: raw HTTP

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (WIN32/64) OpenEdge/11.6.1.0.1293 Lib-ABLSockets/0.3.0
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsgtu14M6qn2xSrgEXXTo=
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

Making sense of HTTP requests: request line

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                poRequest as IWebRequest):


 message oRequest:Method
         /* POST */

        oRequest:Version.
        /* HTTP/1.1 */
```

Making sense of HTTP requests: URI

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                poRequest as IWebRequest):


message poRequest:URI:Host
        /* sstwsuat.taxware.net */
        poRequest:URI:Port
        /* 80 */
        poRequest:URI:Path
        /* Twe/api/rest/calcTax/doc */
        poRequest:URI:GetQueryNames()
        /* ['filter'] */
        poRequest:URI
                  :GetQueryValue('filter')
        /* {ablFilter:""} */
```

Making sense of HTTP requests: headers

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oAcceptHdr as HttpHeader.

oAcceptHdr = poRequest
                :GetHeader('Accept').

message
    oAcceptHdr:Name
    /* Accept */
    oAcceptHdr :Value
    /* application/json */
    oAcceptHdr
        :GetParameterValue('charset')
     /* utf-8 */
```

Making sense of HTTP requests: content

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oData as JsonObject.

message poRequest:ContentType
        /* application/json */
        poRequest:ContentLength
        /* 1805 */
```

Making sense of HTTP requests: content

```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                 poRequest as IWebRequest):
def var oData as JsonObject.
def var oWriter as MessageWriter no-undo.

oWriter = EntityWriterBuilder:Build(poRequest)
                              :Writer.
oWriter:Open().
oWriter:Write(poRequest:Entity).
oWriter:Close().

if type-of(oWriter:Entity, JsonObject) then
  oData = cast(oWriter:Entity, JsonObject).

message oData:GetLogical('isAudit')
        /* false */
```

Entity Writers - `OE.Net.HTTP.Filter.Writer.EntityWriterRegistry`

Name	Writer type	Content types
JSON	JsonEntityWriter	application/json ; application/vnd.progress+json
XML	XmlEntityWriter	text/xml ; application/xml ; text/xml-external-parsed-entity ; application/xml-external-parsed-entity ; application/xml-dtd
Text	StringEntityWriter	text/*
Binary	BinaryEntityWriter	application/octet-stream ; application/pdf ; application/zip ; application/gzip ; audio/* ; image/* ; video/*
Multipart data	MultipartEntityWriter	multipart/*

Default writers in OpenEdge.Net.HTTP.Filter.Payload.*

Inherit from MessageWriter

Making sense of HTTP requests: calling business logic

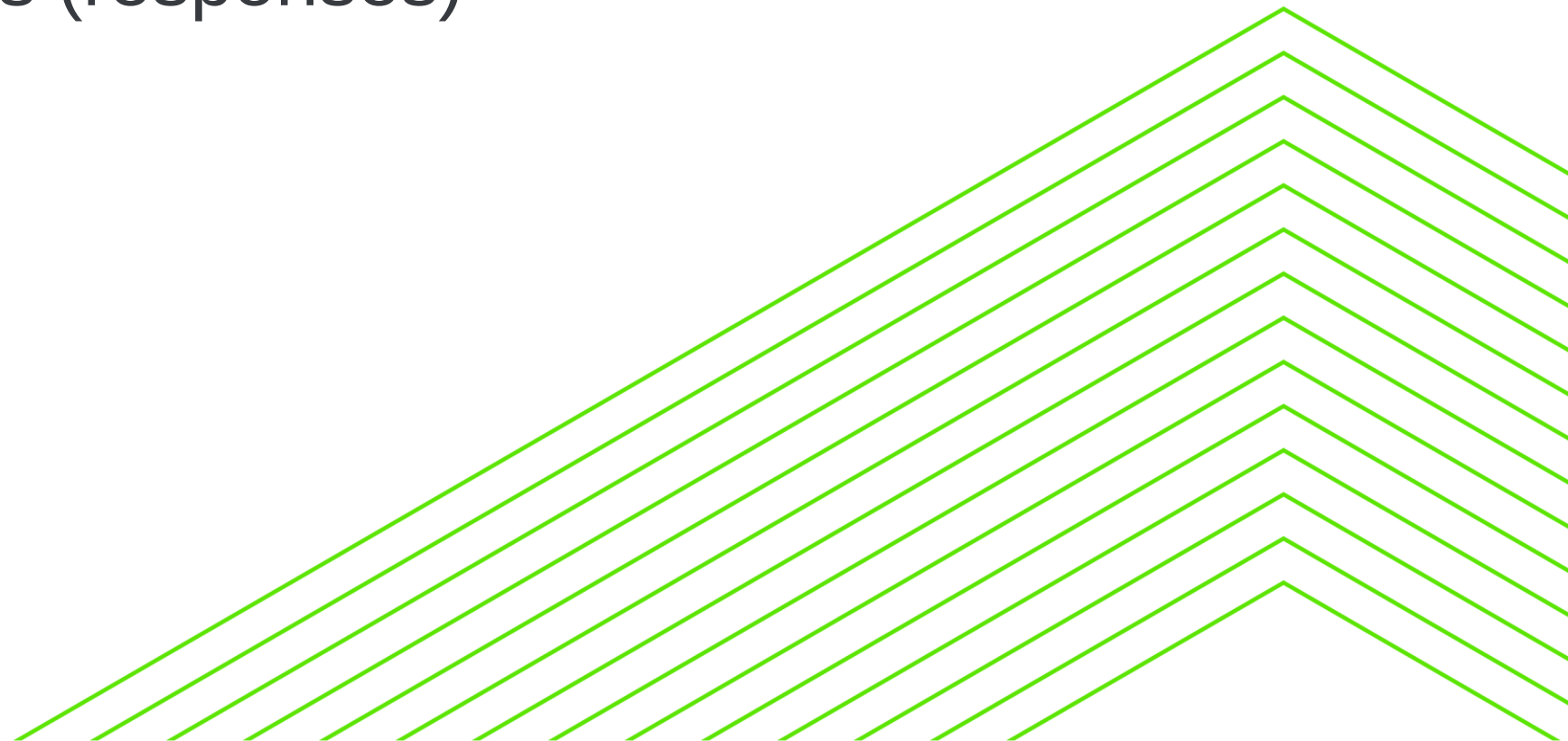
```
POST /Twe/api/rest/calcTax/doc HTTP/1.1
User-Agent: OpenEdge-HttpClient/0.3.0 (W
Host: sstwsuat.taxware.net
Date: 2016-03-04T12:43:18.547-05:00
Content-Type: application/json
Content-Length: 1805
Authorization: TAX restuat@IGS:NXDFGebsg
Accept: application/json; charset=utf-8

{ "rsltLvl": "1",
  "isAudit": false,
  "currn": "USD",
  "txCalcTp": 1,
  "trnDocNum": "277-0",
  "docDt": "2016-02-29T11:58:00",
  "lines": [
    { "debCredIndr": 1,
      "grossAmt": 858.8,
      "custAttrbs": {
        "COMPANY": "578",
        "CUSTOMER-NUMBER": "101",
        "DIVISION": "1",
        "PRODUCT": "1-101",
        "PRODUCT-CATEGORY": "4",
        "WAREHOUSE": "main"
} } ] }
```

```
method override protected int HandlePost(
                  poRequest as IWebRequest):

def var oData as JsonObject.
def var hCalcData as handle.


case poRequest:ContentType:
  when 'application/json' then
    hCalcData:read-json(
                  'JsonObject', oData).
end case.

run Do_Tax_Calc in hTaxPProc (
   input-output dataset-handle hCalcData).
```

Incoming data – `OpenEdge.Web.IWebRequest`

Message element	oRequest = new WebRequest()	
HTTP method ("verb")	:Method	"POST"
URL	:URI	http://localhost:8810/SportsSvc/web/Customer/catalog?filter={"ablWhere"...}
Query parameters	:URI:GetQueryNames() :URI:GetQueryValue	["filter"] Filter => " {'ablWhere':'custnum eq 42"} "
Headers	:GetHeaders() :GetHeader(<name>):Value	[HttpHeader, HttpHeader, HttpHeader] Accept => "application/json"
Cookies	:GetCookie(<name>)	
Path parameters	* URITemplate * PathParameterNames * GetPathParameter(<name>)	{resources}/catalog/{service} "resources,service" "Customer"
Entity / message body	ContentType / ContentLength Entity	application/json
Path information	* TransportPath * PathInfo * WebAppPath	/web /Customer/catalog /SportsSvc

Mapping gazouters (responses)



gazouters: goes-outers,
goes out of's,
outputs

Returning data (good or bad)

- `OpenEdge.Web.WebResponseWriter` a built-in class
- Writes the HTTP 'preamble' nicely
  - Status line `HTTP/1.1 200 OK`
  - Headers
  - Cookies
- You choose how the body is written

Single Write(<data>) call	**Multiple Write(<data>) calls**
Gather entire body before sending	Enables HTTP chunking
Choose your error-page strategy	Cannot use PASOE static error pages
Set all the meta-data before you need writing	Preamble written on first subsequent Write()

- Very similar in many senses to `{&OUT}` approach for 'cgi-wrapper' WebSpeed

Building HTTP Responses – status line

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oHdr as HttpHeader.
def var oResponse as IHttpResponse.
def var hCalcData as handle.


oResponse = new WebResponse().

run Do_Tax_Calc in hTaxPProc (
          input-output dataset-handle hCalcData).

/* default is 200/ OK */
oResponse:StatusCode = 201. //Created


/* these are the same */
oResponse:StatusCode =
     integer(StatusCodeEnum:Created).
```

Building HTTP Responses – headers & cookies

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oAcceptHdr as HttpHeader.
def var oResponse as IHttpResponse.

oResponse:SetHeader('Location',
                    'http://www.example.com/resource').



oResponse:SetCookie(
      new OpenEdge.Net.HTTP.Cookie(
                  'GeoIP' ,
                  '.example.com' ,
                  '/' ,
                  'US:MA:Bedford:42.49:-71.28:v4'  )).
```

Building HTTP Responses – content

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oAcceptHdr as HttpHeader.
def var oResponse as IHttpResponse.
def var hCalcData as handle.


/* Decide what format to send the response */
assign oAcceptHdr = poRequest:GetHeader('Accept').
case oAcceptHdr:Value:
  when 'application/json' then
    assign oResponse:ContentType = 'application/json'.

  when 'text/xml' then
    assign oResponse:ContentType = 'text/xml'

end case.

oResponse:Entity = new OpenEdge.Core.WidgetHandle(hCalcData).
```

Building HTTP Responses – writing the response

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oWriter as OpenEdge.Web.WebResponseWriter.
def var oResponse as IHttpResponse.

/* This Writer does the right thing */
oWriter = new WebResponseWriter(oResponse).

oWriter:Open().

//oWriter:Write([LONGCHAR|MEMPTR]).

oWriter:Close().


return 0.  // Signal to PASOE that we've taken care of it
```

Building HTTP Responses – OE.Web.WebResponseWriter

```
/* Dumps a complete response to the output stream. */
method protected void WriteBody():
  define variable oCTWriter as MessageWriter no-undo.
  define variable oBytes as ByteBucket no-undo.
  define variable mEntity as memptr no-undo.
  /* Convert from an body to bytes */
  if valid-object(this-object:Response:Entity) then
  do on error undo, throw:
    if this-object:preamble_written = false then WriteHttpPreamble().
    /* turn the 'real' data into bytes */
    assign oCTWriter = BodyWriterBuilder:Build(this-object:Response)
                       :Writer.
    oCTWriter:Open().
    oCTWriter:Write(this-object:Response:Entity).
    oCTWriter:Close().
    /* write the bytes to the stream */
    assign oBytes  = cast(oCTWriter:Entity, ByteBucket)
           mEntity = oBytes:GetBytes():Value
           /* may not be written to the stream */
           this-object:Response:ContentLength = get-size(mEntity).
    // write the bytes out
    export stream-handle WebStream mEntity.
  end.
```

Content Type Writers - `OE.Net.HTTP.Filter.Writer.BodyWriterRegistry`

Name	Writer type	Content types
JSON	JsonBodyWriter	application/json ; application/vnd.progress+json
XML	XmlBodyWriter	text/xml ; application/xml ; text/xml-external-parsed-entity ; application/xml-external-parsed-entity ; application/xml-dtd
HTML	HtmlBodyWriter	text/html
Text	StringBodyWriter	text/*
Binary	BinaryBodyWriter	application/octet-stream ; application/pdf ; application/zip ; application/gzip ; audio/* ; image/* ; video/*
Multipart data	MultipartBodyWriter	multipart/*
Form data	FormDataBodyWriter	application/x-www-form-urlencoded

Default writers in OpenEdge.Net.HTTP.Filter.Payload.*

Inherit from MessageWriter

The search alogrithm is

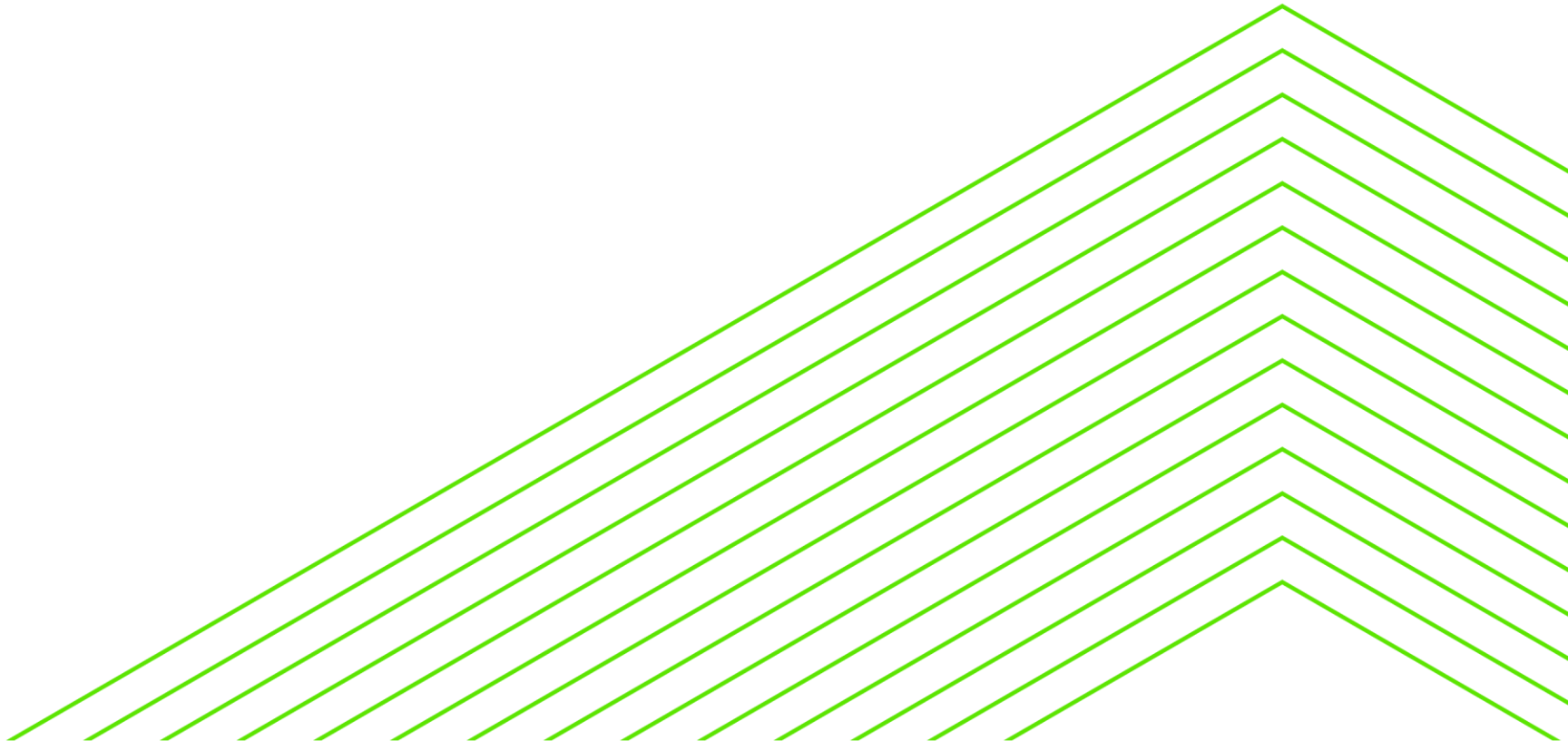
1. Exact match - type / sub-type
2. Vendor types - type / vendor-prefix-removed-sub-type
3. General type - type / *
4. Fallback to HTTP default - application/octet-stream

Building HTTP Responses – success

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 217
Location: http://www.example.com
                          /resource
Server: example.com
Set-cookie: GeoIP=
  US:MA:Bedford:42.49:-71.28:v4;
  domain=.example.com; path=/

{ "dsCalcData": {
    "ttData": [
        {"field1": 0},
        {"field1": 1}
] } }
```

```
method override protected int HandlePost(
                poRequest as IWebRequest):

oResponse:StatusCode = int(StatusCodeEnum:Created)

oResponse:SetHeader('Location',
'http://www.example.com/resource')

oResponse:SetCookie(new OpenEdge.Net.HTTP.Cookie(
       'GeoIP' ,
       '.example.com' ,
       '/' ,
       'US:MA:Bedford:42.49:-71.28:v4'  ))

oResponse:ContentType   = 'application/json'
oResponse:ContentLength = get-size(mEntity)
oResponse:Entity =
         new OpenEdge.Core.WidgetHandle(hCalcData)
```

Errors



Returning errors - application

- An error can be Just Another Response
  - Set the StatusCode to 5xx or 4xx and add an entity (or not)
  - `RETURN 0`
  - Not always suited for standardized responses

```
{ "_retVal": "",
  "_errors": [
    { "_errorMsg": "no param:FirstParam",
      "_errorNum": -1}
    ],
  "_type": "Progress.Lang.AppError",
  "_stack": [
    "GetPathParameter doh.WebRequestMock at line 101 ...",
    "GetInputValue OE...OperationHandler at line 185 ...",
    "Execute OE…ClassOperationHandler at line 1276 ...",
] }
```

Returning errors - PASOE

- You can return a static error page
  - Handler returns `INTEGER` value
  - Set up in web.xml
    per webapp in WEB-INF/
  - Static page is returned
    by default in WEB-INF/jsp/

```
Accept: text/html
```

```
<error-page>
  <location>/WEB-INF/jsp/errorPage.jsp</location>
</error-page>
```

Progress

Progress Application Server

An error occurred while executing your request!

405 - Method Not Allowed: the method type was not allowed for requested URI - GET /web/BadResource

Error details

Remote user:	null
User principal:	null
Scheme:	http
Remote address:	0:0:0:0:0:0:0:1
Server name:	localhost
PASOE product type:	dev
HTTP status:	405
Error detail:	

Debug details

HTTP method:	GET
Web application:	/ROOT
Transport:	OEWebServlet
Request URL:	/web/BadResource
Path info:	null
Servlet name:	OEWebServlet
URI:	/web/BadResource
Exception class:	
Exception message:	
Exception stack trace:	

Returning errors - PASOE

- You can return a static error page
  - Handler returns `INTEGER` value
  - Set up in web.xml
    per webapp in WEB-INF/
  - Static page is returned
    by default in WEB-INF/jsp/

```
Accept: application/json
```

```
<error-page>
  <location>/WEB-INF/jsp/errorPage.jsp</location>
</error-page>
```

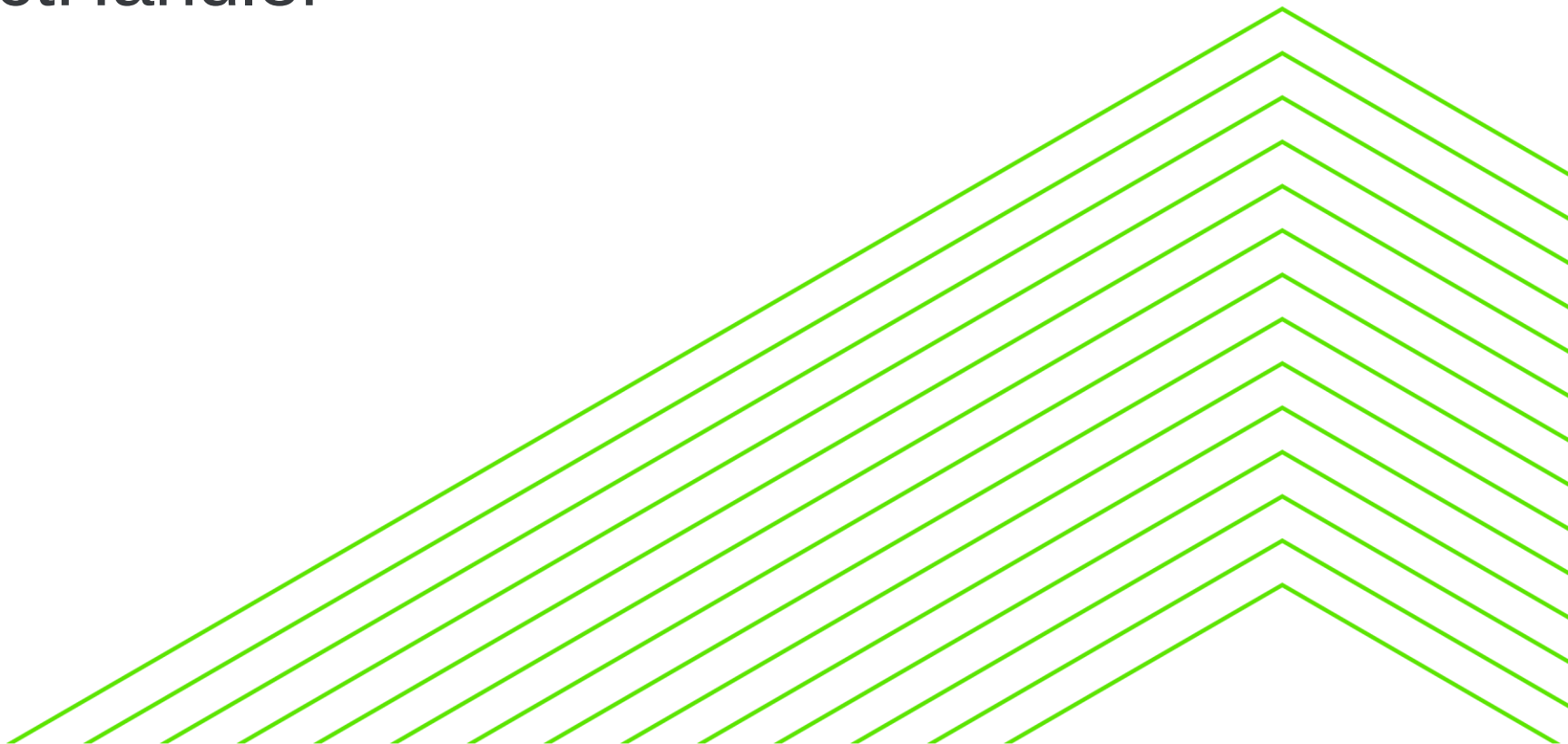


```
{
    "error_code": 405,
    "status_txt": "405 - Method Not Allowed: the method type was not
allowed for requested URI - GET /web/BadResource",
    "error_details": {
        "remote_user": "null",
        "user_principal": "null",
        "url_scheme": "http",
        "remote_addr": "0:0:0:0:0:0:0:1",
        "server_name": "localhost",
        "product_type": "dev",
        "http_status": 405,
        "error_detail": ""
    },
    "debug_details": {
        "http_method": "GET",
        "web_application": "/ROOT",
        "transport": "OEWebServlet",
        "request_url": "/web/BadResource",
        "path_info": "null",
        "servlet": "OEWebServlet",
        "uri": "/web/BadResource",
        "exception_class": "",
        "exception_message": "",
        "exception_stack_trace": ""
    }
}
```

Building HTTP Responses – errors

```
method override protected int HandlePost(
                poRequest as IWebRequest):

def var oAcceptHdr as HttpHeader.
def var oResponse as IHttpResponse.

/* Decide what format to send the response */
catch oErr as Progress.Lang.Error:
  if oAcceptHeader:Value eq 'application/json' then
  assign
    oResponse:StatusCode  = 500
    oResponse:ContentType = 'application/json'
    oResponse:Entity      = oErr.
  else
    return 500.  // Internal Server Error
end catch.
```

Built in: DataObjectHandler



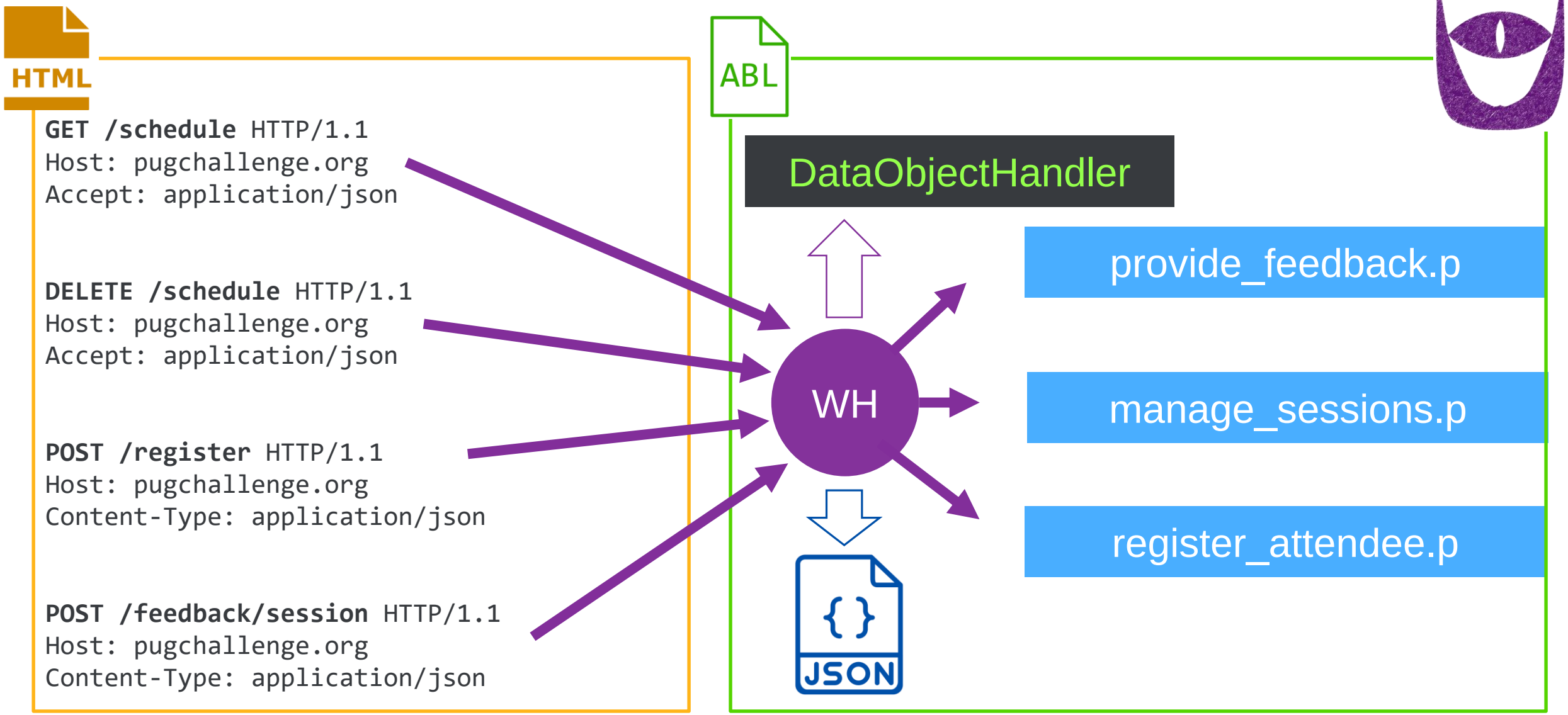
DataObjectHandler

- ABL WebHandler - `OpenEdge.Web.DataObject.DataObjectHandler`
  - Initially for JSDO use-cases …
    … but “invoke” operations means it can do almost anything
  - Why an ABL web handler?
    - Promote and encourage the use of **/web** (aka dogfooding)
    - Add (customer-requested) flexibility to responses (headers, status codes, etc.)

So what?

1. You don't have to write any gazinter or gazouter code
   … no need to understand HTTP semantics
2. Easily expose your AppServer code on the Web
3. Use the URI you'd like

One Handler To Rule Them All



What can I map?

```
"/invoke/1": {
  "PUT": {
    "contentType": "application/json",
    "statusCode": 202,
    "options": {
      "requestEnvelope":  true,
      "responseEnvelope": true
    },
    "entity": {
    "name": "doh.FieldInvokeEntity",
    "function": "in_1",
    "arg": [ {
      "ablName": "",
      "ioMode" : "RETURN",
      "ablType": "character extent",
      "msgElem": { "type": "body",
                   "name": null }
    }, {
      "ablName": "pcChar",
      "ioMode" : "input",
      "ablType": "character",
      "msgElem": { "type": "field",
                   "name": "pcChar" }
} } ] } } },
```

URI mapping

```
/invoke/1
/{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 (plus array variants)

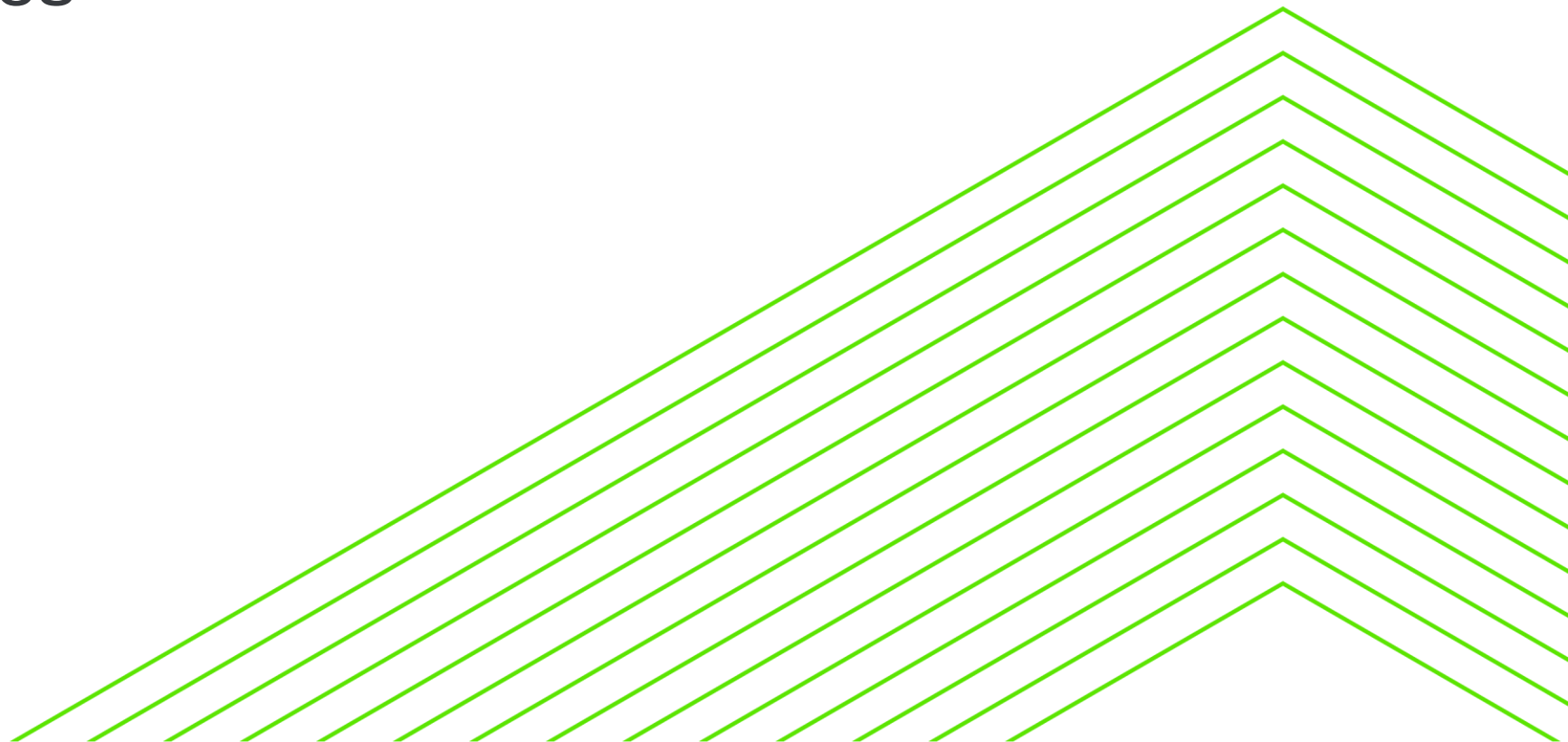
```
"character", "longchar", "integer", "int64", "decimal",
"logical", "rowid", "recid", "date", "datetime", "datetime-tz",
"raw", "memptr", "dataset", "temp-table",
"class <ooabl.type.name>"
```

HTTP Message elements

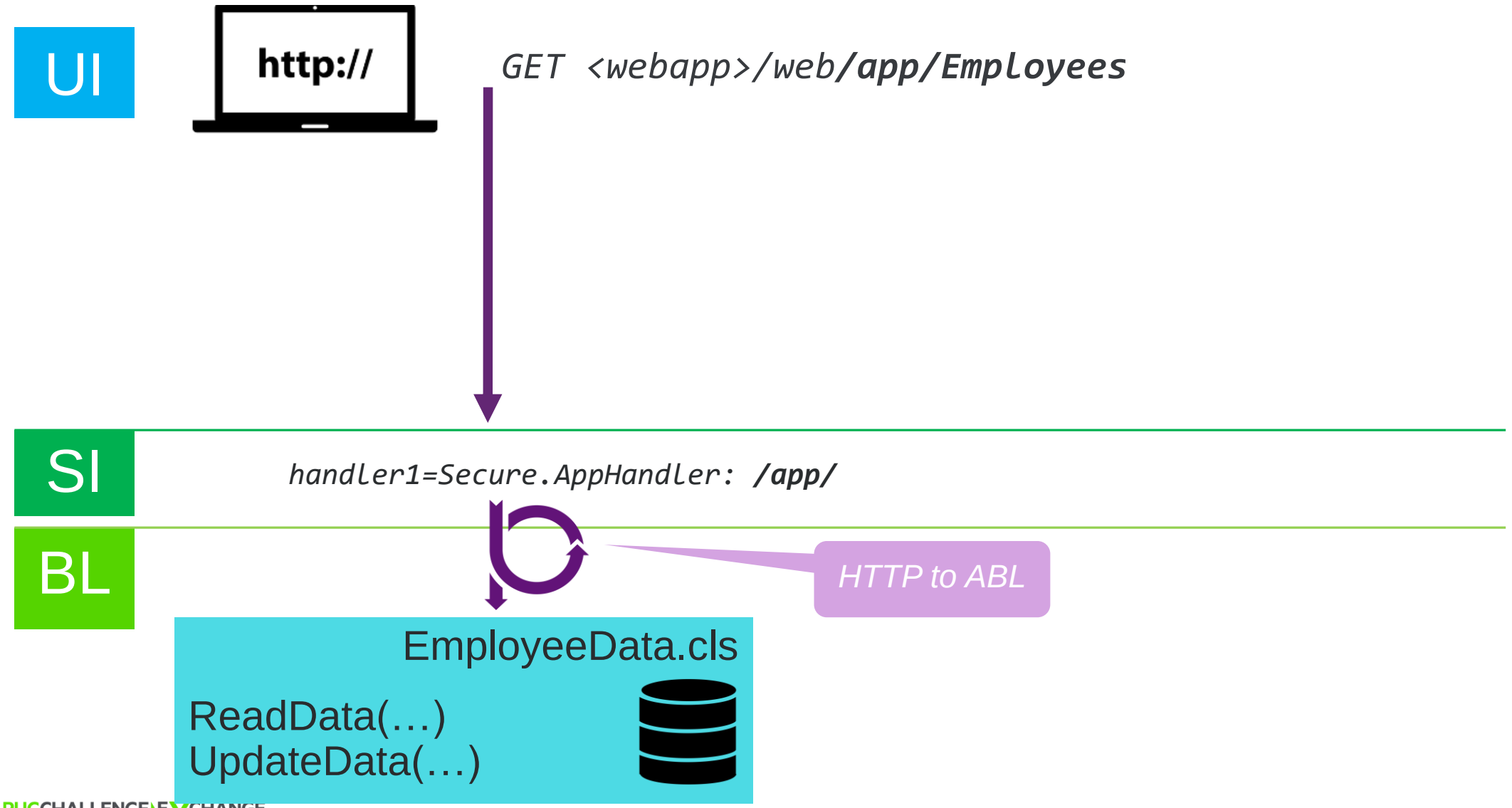
Request-only	"path", "query", "httpMethod", "request", "constant"
Response-only	"none", "statusCode", "statusReason"
Both	"cookie", "header", "field", "body"

PUGCHALLENGE EXCHANGE AMERICAS

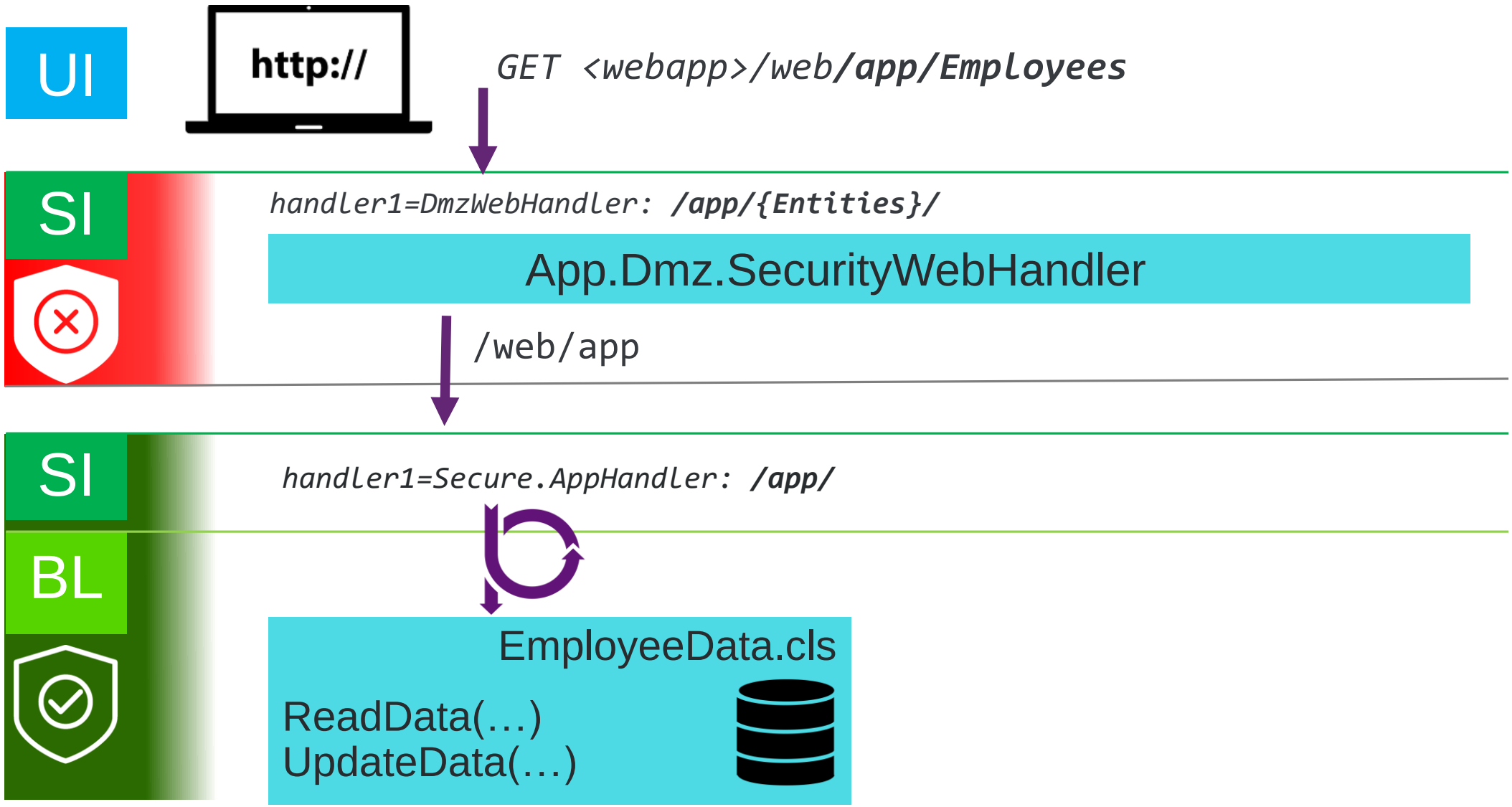
Server Architectures



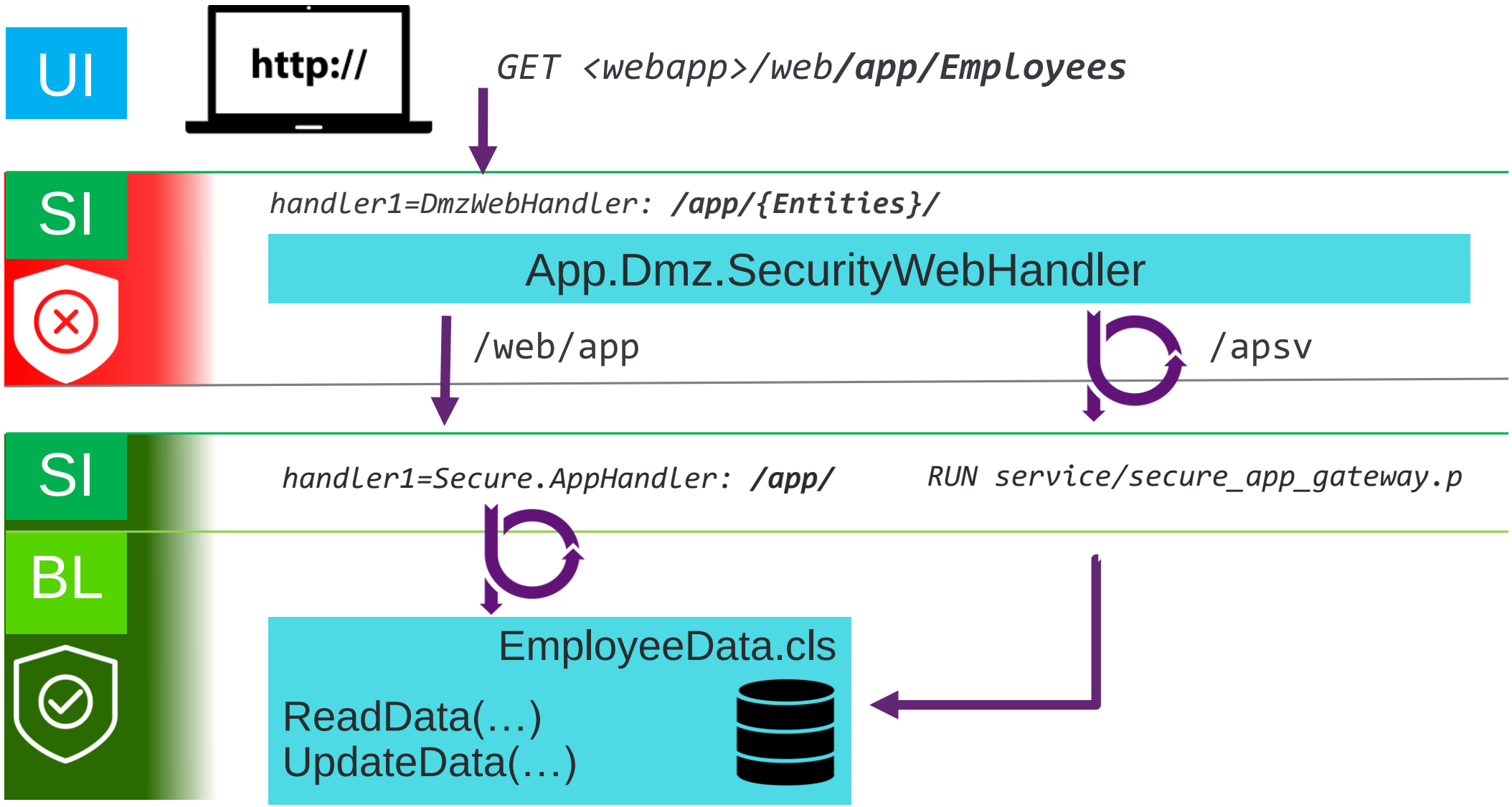
Single-tier architectures



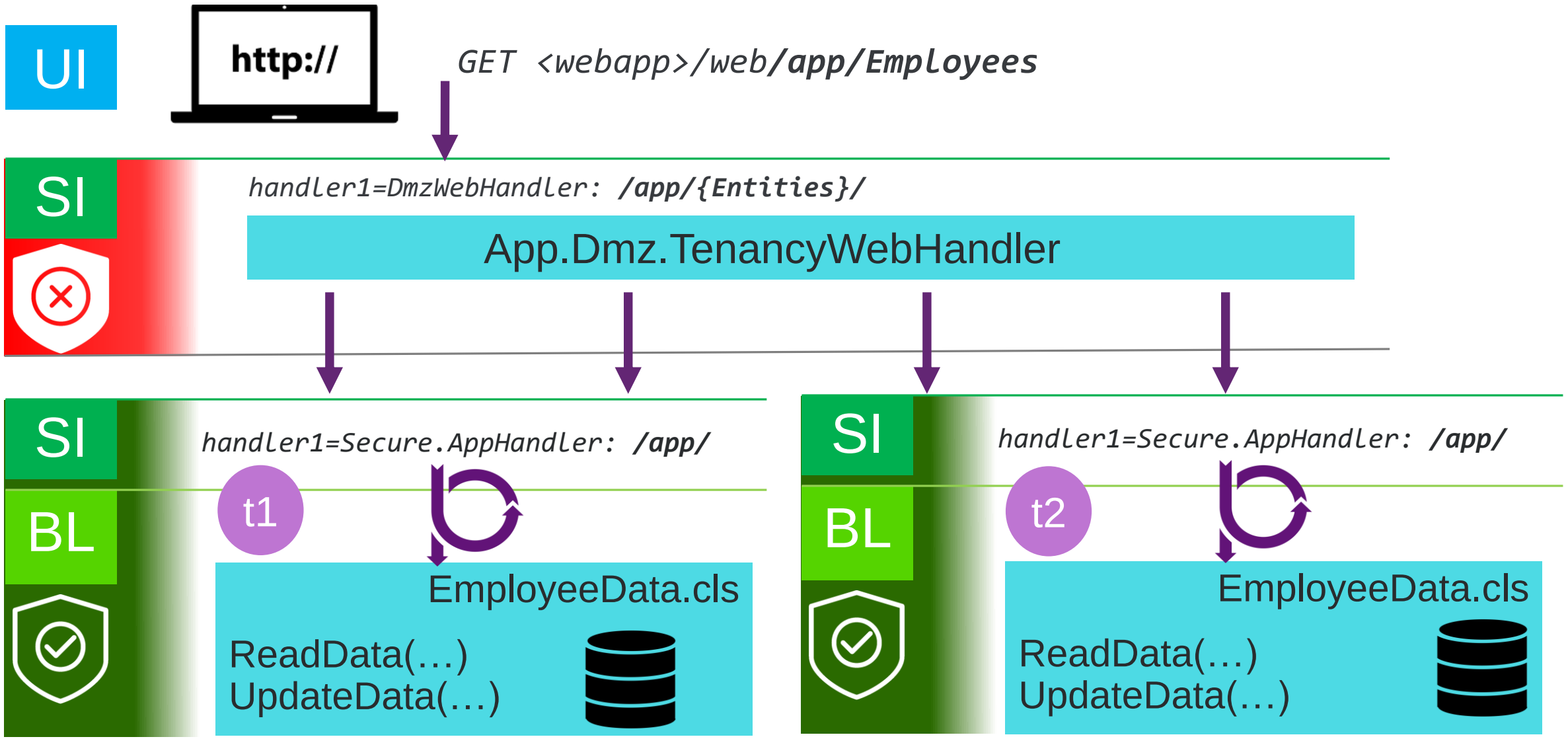
Multi-tier architectures



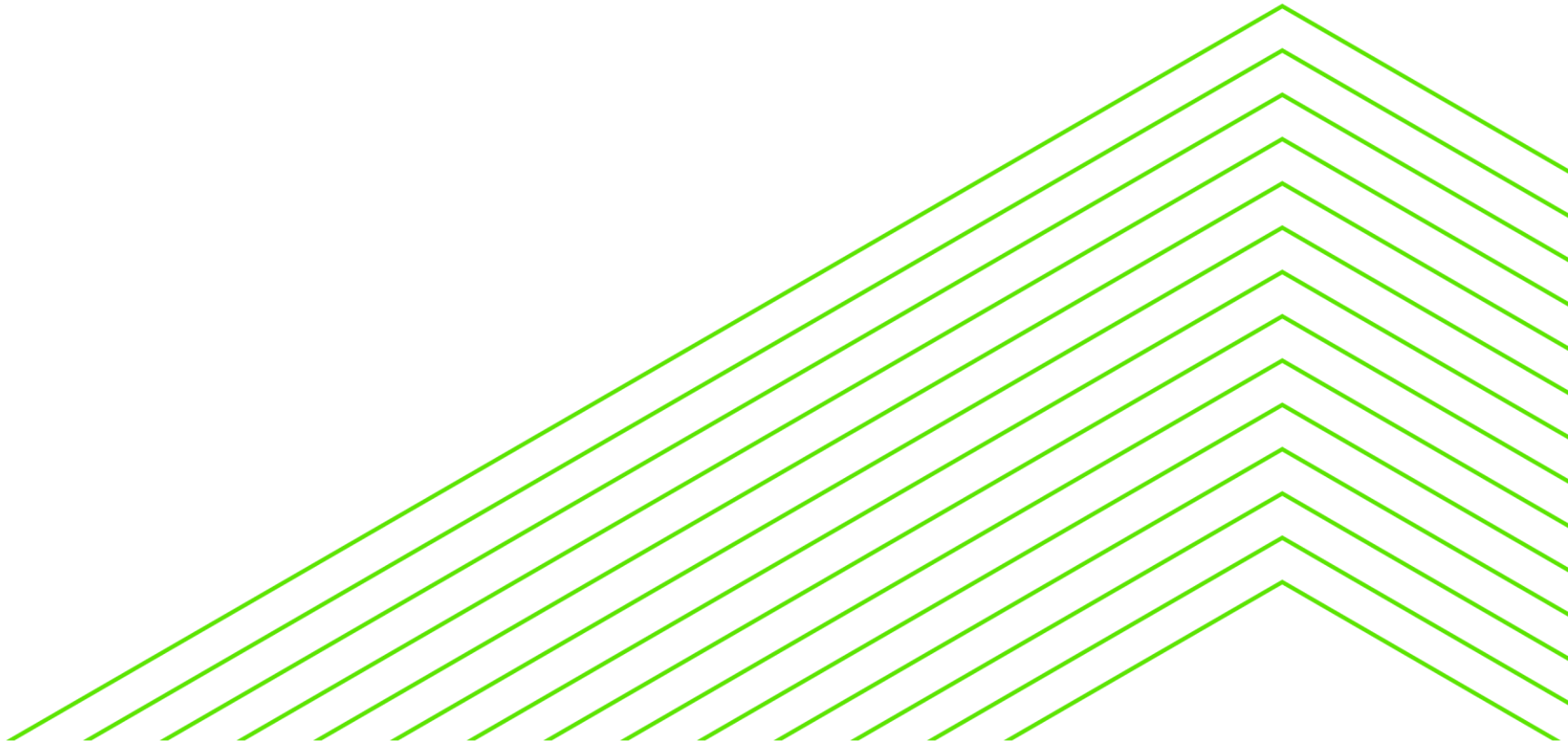
Multi-tier architectures



Multi-tier, multi-tenant architectures



Demo



Conclusion

- Overview of WebHandlers
- How and where to use them
- DataObjectHandler

- Demo code is available at https://github.com/PeterJudge-PSC/http_samples/tree/master/web_handler/img_handler
  - Check out README.md for instructions

PUGCHALLENGE EXCHANGE AMERICAS

Peter Judge pjudge@progress.com