Moving On Up to PASOE/Web

Migrating WebSpeed Applications to Progress Application Server
Overview
Progress Application Server Platform

- A single delivery platform for all Progress Web-based products
- Not only the application but also the web server to support it
  - Created from Apache Tomcat 7.0.55 distribution
- Designed for secure operation
  - Spring Security Framework included
  - Realms and roles defined to implement access control
High-level features

- **Secure**
- **Simple**
  - Administration, scalability, application migration, deployment
  - AppServer connection and operating states
- **Customer Extensible**
  - Open REST APIs for customer developed metrics, monitoring, and administration
  - Installation tailoring
- **Better analysis tools**
  - Built-in metrics gathering, current state queries
- **Faster and optimizes resources**
  - Runs same ABL application and client load with less memory and CPU consumption
The return of WebSpeed

(hooray)
Why hooray?

All the methods (verbs)

<table>
<thead>
<tr>
<th>GET</th>
<th>HEAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT</td>
<td>PATCH</td>
</tr>
<tr>
<td>POST</td>
<td>OPTIONS</td>
</tr>
<tr>
<td>DELETE</td>
<td>TRACE</td>
</tr>
</tbody>
</table>
Why hooray?

All the methods (verbs)

All the message

```plaintext
POST /rest/Employeesvc/Employee HTTP/1.1
Authorization: Basic dG9tY2F0OnRvbWNhdA==
Host: oelxdev06:8881
User-Agent: OpenEdge-HttpClient/0.3.0
Content-Length: 18

{"request": "data"}
```
Why hooray?

All the methods (verbs)

All the message

All the control

```plaintext
define variable webRequest as IWebRequest
define variable httpStatus as integer
assign webRequest = new WebRequest()
httpStatus = integer(StatusCodes.OK)
case MethodEnum:GetEnum(webRequest:Method):
    when MethodEnum:DELETE then httpStatus = integer(StatusCodes.OK)
    when MethodEnum:GET then httpStatus = integer(StatusCodes.OK)
    when MethodEnum:HEAD then httpStatus = integer(StatusCodes.OK)
    when MethodEnum:POST then httpStatus = integer(StatusCodes.OK)
    when MethodEnum:PUT then httpStatus = integer(StatusCodes.OK)
    otherwise httpStatus = integer(StatusCodes.NOT_IMPLEMENTED)
end case.
Assert:NotNull(httpStatus, 'Status return code') /* good */
return httpStatus.
catch oError as Progress.Lang.Error:
    assign httpStatus = HandleException(oError, webRequest)
if httpStatus eq ? then
    assign httpStatus = 0.
return httpStatus.
end catch.
```

```plaintext
GET
PUT
HEAD
PATCH
```

```plaintext
oHeader = poRequest:GetHeader('Content-Type').
oHeader:ParamDelimiter = ';':u.
oHeader:ExtractParameters().

/* URL is /web/img/Employee/{EmpNum} */
assign iEmpNum = integer(poRequest:GetPathParameter('EmpNum'):u)
oEntityWriter = EntityWriterBuilder:Build(poRequest):Option('multipartBoundary':u,
oHeader:GetParameterValue('boundary':u)) :Writer.
oEntityWriter:Open().
oEntityWriter:Write(poRequest:Entity).
oEntityWriter:Close().
assign oEntity = cast(oEntityWriter:Entity, MultipartEntity)
oPart = oEntity:GetPart(1)
oHeader = oPart:Headers:Get('Content-Disposition':u)

/* Content-Disposition: form-data; name="myphoto.png";
filename="emp_21.png" */
cImageFileName = oHeader:GetParameterValue('filename':u).
moBE:WriteEmployeePic(iEmpNum,
cImageFileName,
cast(oPart:Body, ByteBucket):GetBytes()).
```
But Wait! There's More!

All the methods (verbs)
All the message
All the control

define variable webRequest as IWebRequest
define variable httpStatus as integer
assign webRequest = new WebRequest()
httpStatus = integer(StatusEnum:None)
case MethodEnum:GetEnum(webRequest):
  when MethodEnum:DELETE then httpStatus = HandleDelete(webRequest)
  when MethodEnum:GET then httpStatus = HandleGet(webRequest)
  when MethodEnum:HEAD then httpStatus = HandleHead(webRequest)
  when MethodEnum:POST then httpStatus = HandlePost(webRequest)
  when MethodEnum:PUT then httpStatus = HandlePut(webRequest)
  otherwise httpStatus = HandleNotImplemented(webRequest)
end case.
Assert:NotNull(httpStatus, 'Status return code')
/* good */
return httpStatus.
catch oError as Progress.Lang.Error:
  assign httpStatus = HandleException(oError, webRequest)
  no error
if httpStatus eq ? then
  assign httpStatus = 0.
return httpStatus.
end catch.
Moving parts
Request flow … incoming

1. Instance receives the request
   
   POST /CorpHR/web/Employees/pjudge HTTP/1.1

2. ABL application config maps the URI to an ABL handler
   
   handler1=Sports.Web.EmployeeHandler: /Employees/{EmpName}

3. Handler calls a HTTP method-appropriate method
   
   method override protected integer HandlePost(
       poRequest as IWebRequest)

4. HandlePost() reads the request and calls appropriate business logic
   
   oHeader = poRequest:GetHeader('Content-Type').
   moBE:WriteEmployeePic(iEmpNum, cImageFileName, oPic).
   moBE:GetEmployeeInfo(iEmpNum, output oEmployeeJsonData).
Request flow … incoming

1. Instance receives the request
   
   POST /CorpHR/web/Employees/pjudge HTTP/1.1

2. ABL application config maps the URI to an ABL handler
   
   handler1=Sports.Web.EmployeeHandler: /Employees/{EmpName}

3. Handler calls a HTTP method-appropriate method
   
   method override protected integer HandlePost(
     poRequest as IWebRequest)

4. HandlePost() reads the request and calls appropriate business logic
   
   oHeader = poRequest:GetHeader('Content-Type').
   moBE:WriteEmployeePic(iEmpNum, cImageFileName, oPic).
   moBE:GetEmployeeInfo(iEmpNum, output oEmployeeJsonData).
Request flow ... incoming

1. Instance receives the request
   POST /CorpHR/web/Employees/pjudge HTTP/1.1

2. ABL application config maps the URI to an ABL handler
   handler1=Sports.Web.EmployeeHandler: /Employees/{EmpName}

3. Handler calls a HTTP method-appropriate method
   
   method override protected integer HandlePost(
       poRequest as IWebRequest)

4. HandlePost() reads the request and calls appropriate business logic
   oHeader = poRequest:GetHeader('Content-Type').
   moBE:WriteEmployeePic(iEmpNum, cImageFileName, oPic).
   moBE:GetEmployeeInfo(iEmpNum, output oEmployeeJsonData).

Concepts

- Web handler
- Handle* method
Request flow ... incoming

1. Instance receives the request
   
   POST /CorpHR/web/Employees/pjudge HTTP/1.1

2. ABL application config maps the URI to an ABL handler
   
   handler1=Sports.Web.EmployeeHandler: /Employees/{EmpName}

3. Handler calls a HTTP method-appropriate method
   
   method override protected integer HandlePost(
       poRequest as IWebRequest)

4. HandlePost() reads the request and calls appropriate business logic
   
   oHeader = poRequest:GetHeader('Content-Type').
   moBE:WriteEmployeePic(iEmpNum, cImageFileName, oPic).
   moBE:GetEmployeeInfo(iEmpNum, output oEmployeeJsonData).

Concepts

- Web handler
- Handle*
  method
- IWebRequest
Request flow ... outgoing

5. HandlePost() creates a response object
   
   ```
   assign oResp = new OpenEdge.Web.WebResponse()
   oResp:Entity = oEmployeeJsonData
   oResp:ContentType = 'application/json':u.
   ```

6. HandlePost() writes the response data to the output stream
   
   ```
   oWriter = new WebResponseWriter(oResp).
   oWriter:Open().
   cast(oResp:Entity, JsonObject):Write(mEntity).
   oResp:ContentLength = get-size(mEntity).
   oWriter:Write(mEntity).
   return integer(StatusCodeEnum:None). // 0 - zero
   ```

7. Alternative approach for errors/status codes
   
   ```
   return integer(StatusCodeEnum:InternalServerError). // 500
   ```
Request flow … outgoing

5. HandlePost() creates a response object
   
   ```javascript
   assign oResp = new OpenEdge.Web.WebResponse()
   oResp:Entity = oEmployeeJsonData
   oResp:ContentType = 'application/json':u.
   ```

6. HandlePost() writes the response data to the output stream
   
   ```javascript
   oWriter = new WebResponseWriter(oResp).
   oWriter:Open().
   cast(oResp:Entity, JsonObject):Write(mEntity).
   oResp:ContentLength = get-size(mEntity).
   oWriter:Write(mEntity).
   return integer(StatusCodeEnum:None). // 0 - zero
   ```

7. Alternative approach for errors/status codes
   
   ```javascript
   return integer(StatusCodeEnum:InternalServer>Error). // 500
   ```

Concepts

- Web handler
- Handle* method
- IWebRequest
- Web Response object
Request flow ... outgoing

5. HandlePost() creates a response object
   ```
   assign oResp = new OpenEdge.Web.WebResponse()
   oResp:Entity = oEmployeeJsonData
   oResp:ContentType = 'application/json':u.
   ```

6. HandlePost() writes the response data to the output stream
   ```
   oWriter = new WebResponseWriter(oResp).
   oWriter:Open().
   cast(oResp:Entity, JsonObject):Write(mEntity).
   oResp:ContentLength = get-size(mEntity).
   oWriter:Write(mEntity).
   return integer(StatusCodeEnum:None). // 0 - zero
   ```

7. Alternative approach for errors/status codes
   ```
   return integer(StatusCodeEnum:InternalServerError). // 500
5. HandlePost() creates a response object
   ```
   assign oResp = new OpenEdge.Web.WebResponse()
   oResp:Entity = oEmployeeJsonData
   oResp:ContentType = 'application/json':u.
   ```

6. HandlePost() writes the response data to the output stream
   ```
   oWriter = new WebResponseWriter(oResp).
   oWriter:Open().
   cast(oResp:Entity, JsonObject):Write(mEntity).
   oResp:ContentLength = get-size(mEntity).
   oWriter:Write(mEntity).
   return integer(StatusCodeEnum:None). // 0 - zero
   ```

7. Alternative approach for errors/status codes
   ```
   return integer(StatusCodeEnum:InternalServerError). // 500
   ```

Concepts
- Web handler
- Handle* method
- IWebRequest
- Web Response object
- Web Response Writer
- Status Codes
What is a web handler?

- Performs request routing & parameter mapping
  - Figures out what to run
  - Converts HTTP into ABL
- OOABL implementation of Progress.Web.IWebRequest ❯ for advanced use only
- In-the-box versions include
  - OpenEdge.Web.WebHandler ❯ use this for new code
    - Abstract class with some default behaviour
  - OpenEdge.Web.CompatibilityHandler ❯ what 'classic' WebSpeed uses
  - OpenEdge.Web.DefaultHandler ❯ locked-out version

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

- `defaultHandler=OpenEdge.Web.DefaultHandler`
- `handler1=Sports.Web.SportsHandler: /{resources}/catalog/{service}

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

**<relative-uri>**
- Relative to `/web`
- Needs leading `/`
- Text "customer"
- Tokens supported: `{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**
Where does my WebHandler live?

- We consider it a Service Interface
  - So in the webapp - WEB-INF/openedge

- But it's Just ABL so can be anywhere on PROPATH
  - Packages
  - Propath
Coding a web handler

1. Create a new class that inherits from OpenEdge.Web.WebHandler

2. You must implement 3 methods …
   1. HandleGet
   2. HandleMethodNotImplemented
   3. HandleNotAllowed

3. … but you'll want to implement more
   • Handle<http-method>
   • HandleException
Coding a web handler (generated code)

METHOD OVERRIDE PROTECTED INT 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 this and related classes can be found 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 (JsonObject for instance) */
    oBody = NEW OpenEdge.Core.String('Hello Administrator')
    oResponse:Entity = oBody
    oResponse:ContentType = 'text/plain':u /* HTTP messages require a content type */
    oResponse:ContentLength = oBody:Size /* ContentLength is good too */
    /* The WebResponseWriter ensures that the status line and all headers are written out before the message 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.
Coding a web handler (cont'd)

METHOD OVERRIDE PROTECTED INTEGER HandleNotImplemented(  
    INPUT poRequest AS OpenEdge.Web.IWebRequest):

/* Throwing an error from this method results in a 500/Internal Server Error response. The  
web handler will attempt to log this exception. See the HandleGet method's comments on  
choosing a value to return from this method. */

    UNDO, THROW NEW Progress.Lang.AppError("METHOD NOT IMPLEMENTED").

END METHOD.
Incoming data – OpenEdge.Web.IWebRequest

<table>
<thead>
<tr>
<th>Message element</th>
<th>oRequest = new WebRequest()</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP method (&quot;verb&quot;)</td>
<td>:Method &quot;POST&quot;</td>
</tr>
<tr>
<td>Query parameters</td>
<td>:URI:GetQueryNames() :URI:GetQueryValue [&quot;filter&quot;] Filter =&gt; &quot; {'ablWhere': 'custnum eq 42'} &quot;</td>
</tr>
<tr>
<td>Headers</td>
<td>:GetHeaders() :GetHeader(&lt;name&gt;):Value [HttpHeader, HttpHeader, HttpHeader] Accept =&gt; &quot;application/json&quot;</td>
</tr>
<tr>
<td>Cookies</td>
<td>:GetCookie(&lt;name&gt;)</td>
</tr>
<tr>
<td>Path parameters</td>
<td>PathParameterNames &quot;Resources,service&quot; GetPathParameter(&lt;name&gt;) &quot;Customer&quot;</td>
</tr>
<tr>
<td>Entity / message body</td>
<td>ContentType / ContentLength application/json Entity</td>
</tr>
<tr>
<td>Path information</td>
<td>TransportPath /web PathInfo /Customer/catalog WebAppPath /SportsSvc</td>
</tr>
</tbody>
</table>
Outgoing data – IWebResponse

- If a request asks the question "please do something for me", a response is the answer
- You have TOTAL control over …
  - Status codes: 200/OK, 201/Created, 418/I'm a teapot, 501/Not implemented
  - Headers & cookies
  - Entities (payload/body) – can be anything (almost)
    - HTML, JSON, text, XML, multipart, binary, …
  - Response chunking
Return to sender

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

<table>
<thead>
<tr>
<th>Single Write(&lt;data&gt;) call</th>
<th>Multiple Write(&lt;data&gt;) calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather entire body before sending</td>
<td>Enables HTTP chunking</td>
</tr>
<tr>
<td>Choose your error-page strategy</td>
<td>Cannot use PASOE static error pages</td>
</tr>
<tr>
<td>Set all the meta-data before you need writing</td>
<td>Preamble written on first subsequent Write()</td>
</tr>
</tbody>
</table>

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

- 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

- 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/

```xml
<error-page>
  <location>/WEB-INF/jsp/errorPage.jsp</location>
</error-page>
```
Migration
Application server components

Classic AppServer Components

- AdminServer
- NameServer

State-Aware
- AppServer
- 50 Agents

Stateless
- AppServer
- 50 Agents

State-Free
- Rest/Mobile
- AppServer
- 50 Agents

PASOE Components

- PASOE
  - APSV (AIA)
  - SOAP (WSA)
  - REST/Mobile
  - WEB
- Session Manager
- MSAgent
  - 150 ABL Sessions

Client

AIA WSA
- WebSpeed
- REST/Mobile
Instances, web-apps and ABL applications

**PASOE Instance oepas1**

**ABL Application** sports-demo

<table>
<thead>
<tr>
<th>ABL Service</th>
<th>Services</th>
<th>Transports</th>
</tr>
</thead>
<tbody>
<tr>
<td>/apsv</td>
<td>APSV</td>
<td></td>
</tr>
<tr>
<td>/rest/*</td>
<td>REST</td>
<td></td>
</tr>
<tr>
<td>/wsa/*</td>
<td>SOAP</td>
<td></td>
</tr>
<tr>
<td>/web/*</td>
<td>WEB</td>
<td></td>
</tr>
</tbody>
</table>

**WEB-INF/rcode, …**

Webapp Security

**MS Agent**

- AVMs
- ABL Session Map
- Admin Session
- Agent sessions
- rcode cfg, img

**Application security context**

**Clients**

**ABL Request**
Instances, web-apps and ABL applications

**PASOE Instance oepas1**

**ABL Application** **sports-demo**

**/CorpSports**

- **Services**
  - /apsv: APSV
  - /rest/*: REST
  - /wsa/*: SOAP
  - /web/*: WEB

- **WEB-INF/rcode, …**

**/PubSports**

- **Services**
  - /apsv: APSV
  - /rest/*: REST
  - /wsa/*: SOAP
  - /web/*: WEB

- **WEB-INF/rcode, …**

**Clients**

**MS Agent**

**AVMs**

- Agent sessions
- rcode
- cfg, img

**Application security context**
Instances, web-apps and ABL applications

**PASOE Instance oepas1**

**ABL Application sports-demo**

- **ABL**:
  - /SportsSvc
    - Webapp Security
    - WEB-INF/rcode, ...
  - Services:
    - /apsv: APSV
    - /rest/*: REST
    - /wsa/*: SOAP
    - /web/*: WEB

**ABL Application tax-erp**

- **ABL**:
  - /TaxCalcSvc
    - Webapp Security
    - WEB-INF/rcode, ...
  - Services:
    - /apsv: APSV
    - /rest/*: REST
    - /wsa/*: SOAP
    - /web/*: WEB

Clients

http://

MS Agent

Session Manager

Application security context

ABL request

Agent sessions

rcode cfg, img

HTTP Requests
Migration: architecture

- Classic
  - One call from www -> cgi -> web-disp.p
  - Process, session, request all same thing
  - Wait-for web-notify

- PASOE WEB
  - Agent and Session and request totally separate
  - Sessions/processes persist beyond request
  - Classic request/response
  - SESSION:REMOTE, ~CLIENT-TYPE, ~SERVER-OPERATING-MODE checks
  - SESSION:CURRENT-REQUEST-INFO:AdapterType
Migration: Session state

- **Classic**
  - Can only be stateless

- **PASOE**
  - Has to be session-free
  - Means requests aren't guaranteed to go to the same session
  - WebSpeed global variables may need (re)setting in different places
  - Beware of cleanup code being excessively enthusiastic
## PAS for OpenEdge Production versus Development Products

<table>
<thead>
<tr>
<th>PAS for OE Development</th>
<th>PAS for OE Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can compile code</td>
<td>Cannot compile code</td>
</tr>
<tr>
<td>Non-secure configuration</td>
<td>Secure configuration</td>
</tr>
<tr>
<td>Test server instance in $WRKDIR</td>
<td>No test server instances</td>
</tr>
<tr>
<td>Remote administration included</td>
<td>Remote administration optional</td>
</tr>
<tr>
<td>Tomcat remote admin enabled</td>
<td>Tomcat remote admin optional</td>
</tr>
<tr>
<td>OpenEdge remote admin enabled</td>
<td>OpenEdge remote admin optional</td>
</tr>
<tr>
<td>Built-in oeadl web application (ROOT)</td>
<td>Built-in oeadl web application (ROOT)</td>
</tr>
<tr>
<td>All transport deployed and enabled</td>
<td>All transports deployed but disabled</td>
</tr>
<tr>
<td>Restricted</td>
<td>Unrestricted</td>
</tr>
<tr>
<td>5 concurrent requests</td>
<td>concurrent requests</td>
</tr>
<tr>
<td>1 agent</td>
<td>number of agents</td>
</tr>
</tbody>
</table>
Summary

- WEB transport gives you the capabilities to handle requests from any client
  - Complete control over the 'gazinters' and 'gazouters'

- In parallel to your APSV clients
  - With the same or different security models

- PASOE gives you the ability to tailor the deployment of your ABL applications
  - Depending on client sizes
  - Depending on you scalability needs
  - Depending on the developer-deployer-model
Questions?

Peter Judge pjudge@progress.com
Security
Spring Security

- OpenEdge supplements the Java container’s security with the industry-recognized Spring security framework
- Spring Security is a customizable authentication and access control framework
  http://projects.spring.io/spring-security/
- It is one of the industry standards for securing Spring-based applications
Web Application Authentication Models

- **Anonymous**: No user authentication or login session

- **HTTP Basic Authentication**: Client sends base64 encoded user name/password to web application in each http request
  - HTTP header: Authorization

- **HTTP Form Authentication**: Client logs in and out the web application once per session
  - **Login**: The client obtains user credentials and POSTs them to the web application
    - URI: `/static/auth/j_spring_security_check`
    - Body: `j_username=xxxx&j_password=yyyy&submit=Submit+Query`
    - Cookie: JSESSIONID
  - **Logout**: The client uses a GET request to log out
    - URI: `/static/auth/j_spring_security_logout`