Progress Application Server

Where does my WebSpeed fit in?

- November 16, 2017
- Chad R. Thomson, Sr. Principal Consultant
- Progress Software, Inc.
Speaker Bio

- Over 20 years of industry experience favoring reality over formality.
- Over 16 years in services focus with Progress Software
- Specializing in vendor-neutral, cross-platform application and service integration
- Passionate technology advocate
Agenda

- Primary Goals
  - Points to consider, what to expect, and questions to ask when migrating your WebSpeed application to PASOE

- Progress Application Server for OpenEdge (PASOE) Platform Intro (brief)

- Key platform differences compared to Classic WebSpeed

- Deployment Landscape and Tiering

- Requests and Responses

- Side-by-Side Code Comparison

- Other considerations
PASOE Platform
Background
PASOE Server Platform

- A single delivery platform for all Progress web-based products
  - Web services (SOAP)
  - REST services (JSON)
  - WebSpeed (HTML, JSON, other)
- Secure, proven, production-ready platform
  - Tomcat
  - Spring Security
- Provides compatibility for running existing WebSpeed code
PASOE Platform

Request (HTTP)

Web Server (Apache)

Messenger (CGIIP)

OE NameServer (optional)

Classic WebSpeed Broker (Java)

Agent/Session

Agent/Session

Agent/Session

PAS Tomcat (J2EE Container)

Multi-Session Agent
- Session/Thread
- Session/Thread

Multi-Session Agent
- Session/Thread
- Session/Thread

Request (HTTP)

Tomcat (PAS)
Question to Ask: Why should I care?

- Industry Standard
  - Consider the search for third-party support
  - Vast availability of tools, add-ons, plugins, and documentation

- Efficiency
  - Fewer resources consumed
    - Executable process count (span v exec)
    - Memory footprint
  - Simplified Communication channel
    - No CGIIP network communication

- Scalable
  - Native and extensible load balancing and clustering
Key Platform Differences
## Key Platform Differences

<table>
<thead>
<tr>
<th></th>
<th>Classic WebSpeed</th>
<th>PASOE WebSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workshop</td>
<td>Mainstay dev tool</td>
<td>Not formally supported*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>*still works (mostly)</td>
</tr>
<tr>
<td>Custom web-disp.p</td>
<td>There are better ways</td>
<td>Migrate to AppServer-style agent procedures</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mapped web-objects</td>
<td>Re-consider page architecture and design Consider client-side API-style design</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLI utils</td>
<td>wtbman, mergeprop</td>
<td>pasman, tcman, oeprops</td>
</tr>
</tbody>
</table>
The biggest difference is:
Platform and technology considerations force a mindset evolution.
Deployment Landscape and Tiering
Classic WebSpeed Landscape
PASOE WebSpeed Landscape: Simple
PASOE WebSpeed Landscape: Reverse Proxy
What you should consider:

- Where should you place the static web assets (.jpg, .js, .css) vs 'code'?

- Evaluate how your session/state management works
  - How does clustering and/or load-balancing affect your application's state-management model?

- Using PAS session-free model, continue what you were doing
  - It should just work
PASOE WebSpeed Landscape: Advanced
Requests and Responses
Requests and Responses: URL Patterns

- Typical URL Request Pattern for PASOE
  - `<scheme>://<server>:<port>/[<webapp>/]<transport>[/<service>][/<resource>]`
  - Scheme: http/https
  - WebApp: (optional) relative to Tomcat webapps folder (usually)
  - Transport: Web for WebSpeed (also REST, SOAP, or APSV)
  - Service: grouping of resources
  - Resource: pathing and optional tokens for handler mapping

- Examples
  - https://myserver.net:8443/web/this/that
Requests and Responses: URL Mapping

- PASOE - WebHandlers
  - map to URL "slugs" (openedge.properties)
    - Handler1= webhandlerclass2:/thisurlpath/{token}/{id}
    - Handler2= webhandlerclass2:/thisurlpath/{token}
    - Handler3= webhandlerclass2:/thisurlpath
    - Handler4= webhandlerclass1:/thisotherpath
    - defaultHandler= OpenEdge.Web.CompatibilityHandler

- URL handlers should be configured from MOST to LEAST specific URL
- CompatibilityHandler allows existing WebSpeed code to "just work"

- OpenEdge.Web, OpenEdge.Net, OpenEdge.Core Documentation
  - https://documentation.progress.com/output/oehttpclient/117/
class OpenEdge.Web.CompatibilityHandler implements Progress.Web.IwebHandler:  
define protected property AllowedMethods as char no-undo init "POST,GET":U get. set.

/* use private var for destructor */
define private variable mProcHandler as handle no-undo.

/* Keep track of web-handler.p */
define protected property ProcedureHandle as handle no-undo
get ()
    if not valid-handle(ProcedureHandle) then
        ProcedureHandle = StartProcedure().
    return ProcedureHandle.
end.  
set.

constructor public CompatibilityHandler():
end constructor.

/****
 * Handle a web request.
 */
method public integer HandleRequest():
define variable cMethod as character no-undo.

    if lookup(cMethod,AllowedMethods) = 0 then
        /* throwing errors to the client is not supported */
        /* undo, throw an approriate "Method "+ cMethod + " is not supported by webs */
        return error "Method " + cMethod + " is not supported by webspeed compatabil
run process-web-request in ProcedureHandler.
    return 0.
Requests and Responses: web-handler.p

- Any of this look familiar?
  - Closely resembles classic web-disp.p
Requests and Responses: WebRequest Object

Purpose: Default handler for the HTTP GET method. The request being serviced and an optional status code is returned. A zero or null value means this method will deal with all errors.

```
method override protected integer HandleGet(input poRequest as OpenEdge.web.IWebRequest):
```

```
poRequest:
```

```java
// Class Browser

define variable poRequest as OpenEdge.web.IWebRequest
```

© 2017 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.
Requests and Responses: WebResponse Object

/* 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/ohttpclient */ 

assign oResponse = new OpenEdge.Web.WebResponse()
            oResponse:StatusCode = integer(StatusCodesEnum:OK).

/* This body object can be a string or something else (JsonObject for instance) */ 

assign oBody = new OpenEdge.Core.String(
            'Hello bravepoint'
            + '\r\n\r\n' + CRLF +
            + 'This message was returned by HandleGet in OtherWebHandler.
            ).

assign oResponse:Entity = oBody
            /* HTTP messages require a content type */
            oResponse:ContentType = 'text/plain';
            /* ContentLength is good too */
            oResponse:ContentLength = oBody:Size

            /* 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().
            /* Finish writing the response message */
            oWriter:Close().
Side-by-Side Comparison
## Side-by-Side: Requests

<table>
<thead>
<tr>
<th>Classic WebSpeed</th>
<th>PASOE WebSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>val = get-value('field')</code></td>
<td><code>val = poRequest:GetContextValue(&quot;thisvar&quot;)</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><code>cookieval = get-cookie(&quot;cookiename&quot;)</code></td>
<td><code>def var ocookie as OpenEdge.Net.HTTP.Cookie</code></td>
</tr>
<tr>
<td></td>
<td><code>occookie = poRequest:GetCookie(&quot;cookie&quot;)</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Assuming a Handler mapping of:</strong></td>
<td><strong>Accessing URL params</strong></td>
</tr>
<tr>
<td><code>Handler1= webhandlerclass2:/thisurlpath/{token}/{id}</code></td>
<td><code>oReq:PathParameterNames /* &quot;token,id&quot; */</code></td>
</tr>
<tr>
<td><strong>URL request of:</strong></td>
<td><code>oReq:GetPathParameter(&quot;token&quot;) /* &quot;hello&quot; */</code></td>
</tr>
<tr>
<td><code>http://pca2017.thomson.net/web/thisurlpath/hello/1</code></td>
<td><code>oReq:GetPathParameter(&quot;id&quot;) /* &quot;1&quot; */</code></td>
</tr>
</tbody>
</table>
## Side-by-Side: Responses

<table>
<thead>
<tr>
<th>Classic WebSpeed</th>
<th>PASOE WebSpeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>{&amp;OUT}, put stream webstream ...</td>
<td>WebResponse and WebResponseWriter</td>
</tr>
<tr>
<td>{&amp;out}</td>
<td></td>
</tr>
<tr>
<td>&quot;&lt;h1&gt;Hello World!&lt;/h1&gt;&quot;</td>
<td></td>
</tr>
<tr>
<td>{&amp;end}</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>oResponse = new OpenEdge.Web.WebResponse()</td>
<td></td>
</tr>
<tr>
<td>oResponse:ContentType = 'text/html'</td>
<td></td>
</tr>
<tr>
<td>oResponse:Entity = new String(&quot;&lt;h1&gt;Hello World! &lt;/h1&gt;&quot;)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>oWriter = new WebResponseWriter(oResponse)</td>
<td></td>
</tr>
<tr>
<td>oWriter:Open()</td>
<td></td>
</tr>
<tr>
<td>oWriter:Close()</td>
<td></td>
</tr>
</tbody>
</table>
Thinking of migrating to PASOE? You need to **change your mind**!
What to consider: Deployment and Design

- Separate "handlers" from "code"
  - think: API-entry-points vs logic
  - handlers address requests/responses
  - application code provides logic

- Traditional WebSpeed r-code
  - ESS or wrap-cgi
  - Position with logic

- Namespace organization
  - Pay attention to PROPATH
Other Considerations

- A Lot of Planning and [re-]evaluation
  - Substantial departure from traditional WebSpeed development and deployment
  - Classic web development patterns vs "micro-services" style (REST, JSON)
    - Web de-evolution: Static => Dynamic generation => Static w/REST
  - Authentication, security model, TLS termination points

- Time Cost for infrastructure planning, ramp-up, and implementation
  - Direct investments to devOps or internal assets
  - Do not under estimate the need to prepare for production deployment

- IDE Consideration
  - For classic WebSpeed, use any IDE
  - PAS Tooling Built-in to PDSOE
  - PAS development is much more OOP-related
Questions?