524 Progress Application Server: Where does my WebSpeed fit in?

June 5, 2017
Chad R. Thomson
Senior Principal Consultant
Progress Software
Speaker BIO

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

- Primary Goals:
  - Describe the PAS Platform as applies specifically to WebSpeed applications
  - Points to consider, what to expect, and questions to ask when migrating your WebSpeed application to the PAS Platform

- Progress Application Server (PAS) Platform Background (brief)
- Key platform differences compared to Classic WebSpeed
- Deployment Landscape and Tiering
- Requests and Responses
- Side-by-Side Code Comparison
- Other Considerations
Progress Application Server (PAS)
Platform Background
Progress Application 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
Progress Application Server 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

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

- **Industry Standard**
  - Consider searching for support
    - "OpenEdge" or "WebSpeed" vs "Tomcat", "Spring Security"
  - Vast availability of Tools, add-ons, plug-ins, and documentation

- **Efficiency**
  - Fewer resources
    - Executable processes (spawn, exec)
    - Memory
  - Simplified Communication channel
    - No CGIIP network communication
    - No OE Nameserver

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

**Classic WebSpeed**
- Workshop tools
- Customized web-disp.p
  - *there are better ways*
- Mapped web-objects
- Status and configuration utilities
  - wtbman
  - mergeprop

**PAS WebSpeed**
- Workshop not formally supported
- Web-disp.p modifications need to be migrated
- No Mapped-object support
- New configuration utilities
  - pasman
  - tcman
  - oeprops

Key Platform Differences: How do they affect me?

Without this...

- Workshop Tools
  - *hint: still accessible*
- Web-disp.p modifications
- Mapped web-objects
- Classic CLI utilities

You may need to...

- Look at tool use and evaluate new toolchain components
- Use agent and session init procedures
  - *think: AppServer srvStartupProcedure*
- Re-consider page architecture design and approach
  - eg. Client-side Ajax web-handlers or custom template objects
- Adopt new CLI tools
  - Review and migrate current deployment and configuration scripts
The biggest difference:

Platform and technology considerations force a mindset evolution.
Deployment Landscape and Tiering
Classic WebSpeed Landscape
PAS WebSpeed Landscape: Simple

- Where do I deploy PAS/Tomcat?
  - It's a web server, right? So, it should go into the DMZ, right?
PAS WebSpeed Landscape: Reverse Proxy
What you should consider

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

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

- Using PAS session-free model, continue what you were doing
  - It should *just work*
PAS WebSpeed Landscape: Advanced

http://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/pasoe-admin/load-balancing-options-for-pas-for-openedge.html
Requests and Responses
Requests and Responses: URL Patterns

- Typical URL Request Pattern for WebSpeed
  - `<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
  - http://pca2017.thomson.net/web/test/this/that
Requests and Responses: URL Mapping

- PAS 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
Requests and Responses: OpenEdge.Web.CompatibilityHandler

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 mProcHandle 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 new AppError("Method " + cMethod + ": is not supported by WebSpeed compatibility"
            run process-web-request in ProcedureHandle.
        return 0.
Requests and Responses: web-handler.p

```pascal
procedure process-web-request:
    define variable cMimeCharset as character no-undo.
    define variable cProCharset as character no-undo.
    define variable iTest as integer no-undo.

    output {&WEBSTREAM} TO "WEB":U.

    /* This MUST be the first thing written */
    output-http-header("", "HTTP/1.1 200 OK":U).

    /* Parse the request/CGI from the web server. */
    run init-cgi in web-utilities-hdl.

    /* Initialize for web-request. */
    run init-request in web-utilities-hdl.

    run run-web-object in web-utilities-hdl (AppProgram) no-error.

    /* Run clean up and maintenance code */
    run end-request in web-utilities-hdl no-error.

finally:
    output {&WEBSTREAM} CLOSE.
end.
end procedure. /* process-web-request */
```

- Any of this look familiar?
  - Closely resembles web-disp.p
Requests and Responses: WebRequest Object
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/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) */
assign  
  oBody = new OpenEdge.Core.String(  
    'Hello bravepoint'  
    + '
':u /*CRLF */  
    + 'This message was returned by HandleGet in OtherWebHandler.'  
  )
.

assign  
  oResponse:Entity = oBody  
  /* HTTP messages require a content type */
  oResponse:ContentType = 'text/plain':u  
  /* 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 Code Comparison
Side-by-Side Code Comparison: Requests

### Classic WebSpeed

- get-value
  - \( \text{val} = \text{get-value}("\text{thisvar}\") \)

- get-cookie
  - \( \text{cookieval} = \text{get-cookie}("\text{cookiename}\") \)

### PAS WebSpeed

- WebRequest:GetContextValue()
  - \( \text{val} = \text{poRequest:GetContextValue}("\text{thisvar}\") \)

- WebRequest:GetCookie()
  - \( \text{def \ var \ ocookie \ as \ OpenEdge.Net.HTTP.Cookie} \)
  - \( \text{ocookie} = \text{poRequest:GetCookie}("\text{cookie}\") \)

- Accessing URL params
  - \( \text{oReq:PathParameterNames} /* "token.id" */ \)
  - \( \text{oReq:GetPathParameter}("\text{token}\") /* "hello" */ \)
  - \( \text{oReq:GetPathParameter}("\text{id}\") /* "1" */ \)

Assuming a Handler mapping of:

**Handler1= webhandlerclass2:/thisurlpath/{token}/{id}**

URL request of:

http://pca2017.thomson.net/web/thisurlpath/hello/1
Side-by-Side Code Comparison: Response

**Classic WebSpeed**

- `{&OUT}`
  - `{&out}`
    - "<h1>Hello World!</h1>"
  - `{&end}`

**PAS WebSpeed**

- **WebResponse and WebResponseWriter**
  - `oResponse = new OpenEdge.Web.WebResponse()`
    - `oResponse:ContentType = 'text/html'
      oResponse:Entity = new String("<h1>Hello World!</h1>")`
  - `oWriter = new WebResponseWriter(oResponse)
    oWriter:Open()
    oWriter:Close()`
What you should consider: Code Deployment

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

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

- Adopt a namespace organization pattern even if you're not using OOP
  - Pay attention to PROPATH
Other Considerations

- A Lot of Planning and [re-]evaluation
  - Platform and technology requirements may represent substantial departure from traditional WebSpeed development and deployment
  - Classic web development patterns vs "micro-services" style (REST, JSON)
    - Evolution of web development has come full-circle
    - Static web => Dynamic generation => Static w/REST
  - Authentication, security model, TLS termination points
    - Intrusion or exploitation sensitivity
- Time Cost for infrastructure planning, ramp-up, and implementation
- IDE Consideration
  - prior WebSpeed development, any IDE would do
    - Still use preferred tools for presentation
  - PAS development is much more OOP-related
    - seriously consider using PDSOE
Questions?
References

- 503: **OpenEdge REST Backends, JSDO, Kendo UI and Angular 2**, Michael Fechner
- 284: **PAS for OpenEdge Web Handlers: A Deep Dive**, Peter Judge
  - (2016) 612: *Moving On Up: Migrating Your Webspeed Application to PASOE*
- (2016) 392: **Why Progress Application Server for OpenEdge?**, Roy Elis
- TCman quick reference card
  - [https://community.progress.com/community_groups/openedge_general/m/documents/2968](https://community.progress.com/community_groups/openedge_general/m/documents/2968)
- Edsel Garcia, Generic WebSpeed Data objects
  - [https://community.progress.com/community_groups/mobile/m/documents/2677](https://community.progress.com/community_groups/mobile/m/documents/2677)
- OpenEdge Corelib and Netlib Documentation
  - [https://documentation.progress.com/output/oehttpclient/oe116/index.html](https://documentation.progress.com/output/oehttpclient/oe116/index.html)
- OpenEdge PAS load balancing options