REST Support for B2B Access to Your OpenEdge AppServer

Kumar Navneet
Principal Software Engineer
Progress Software
knavneet@progress.com

David Cleary
Principal Software Engineer
Progress Software
davec@progress.com

PROGRESS EXCHANGE 2014

Developing applications for
B2B or Business to Business,
is about empowering
your users

Progress OpenEdge REST

Agenda

- REST is the Hardest Easy Thing I’ve Done
- Creating RESTful APIs
- Using the Tools
- Moving to Production
- When Things Go Sideways
- Pacific Application Server for OpenEdge

REST Is the Hardest Easy Thing I've Done

PROGRESS EXCHANGE 2014

When REST Can Be the Right Choice When…

- Converting existing SOAP services to REST
- Need to call a pre-11.2 AppServer
- The AppServer requires access to HTTP request headers, cookies, and URL fields
- The REST client requires control over HTTP response headers, and cookies
- You need to support ANY type of HTTP compliant client (not limited to JavaScript)

REST Is Easy

- Client simply uses standard HTTP messages, responses, verbs, etc.
- Uses standard web servers – so no firewall issues
- The REST service is just a web app and can be consumed by clients written in any language
- Use any HTTP 1.1 enabled client
- Each REST resource is an object that has data and actions
- Each REST resource is identified using the triplicate: URL, verb, and media type
- **There are NO rules**

Developing a RESTful Client API Is **HARD**…

- **There are no rules**
  - No formal API contract for client developer to use
    - What is the list of supported REST resources?
    - What verbs and media type is supported for each resource?
    - What variable parameter(s) go where in the HTTP messages and responses?
- A starting point may be to think of **CRUD** operations and supporting functions
- Follow the same requirements as you use for your application
  - Multiple simultaneous Versions
  - Extensibility
  - **Intuitively** organized sets of objects and related operations (i.e. REST services)
  - Deploy as incrementally added web application(s) and REST service(s)
  - Secure (when it needs to be)

Creating RESTful APIs

PROGRESS EXCHANGE 2014

A Good RESTful API Design is Essential

- Choose between one monolithic REST service versus multiple REST services
  - Divide API [URL] space into web applications hosting related REST services (example: application administration services versus application data services)
  - Each REST service's URL path is a hierarchy of related resources
  - Each resource's URL path can have one or more instance qualifiers
  - Each resource's URL path has one or more [action] verbs (and media type)

RESTful Web Application Design

- Each deployed (OpenEdge REST) web application has
  - A web application name
  - One or more REST services
  - A security configuration (user authentication and [URL] authorization)
  - A connection to ONE State-free model AppServer [service]
- Example (for OpenEdge REST web application):

http://host:port/webAppName/rest/

Deployment site defined part

OE defined part

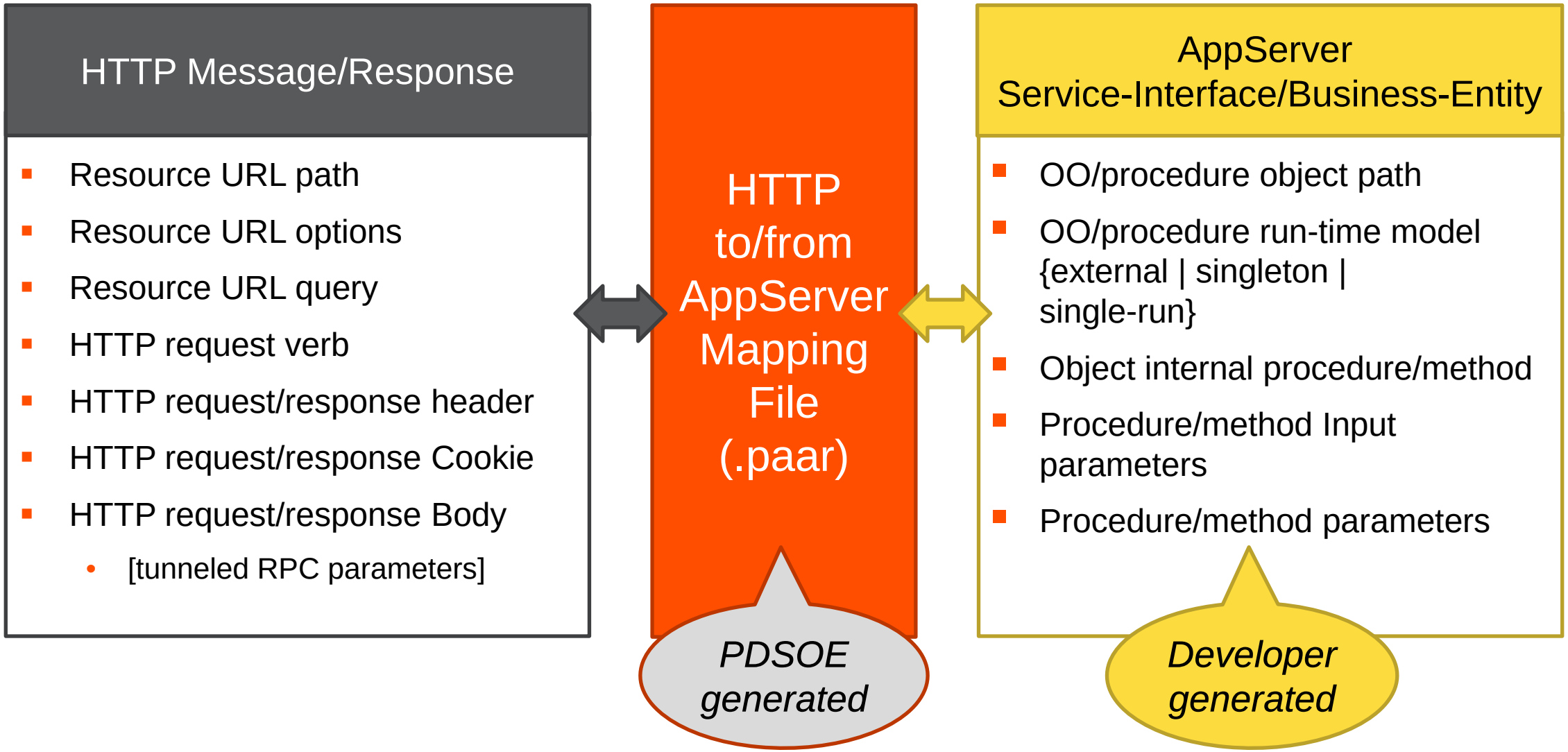
RESTful Service and URL Design

- What is a REST Service
  - Has a service-name that appears in the URL
  - A service-name contains one or more REST resources
  - Each resource has a unique URL path within the service
  - Each resource URL path can have
    - Optional input parameters and/or options
- Example

http://host:port/webAppName/rest/<service-name>/<resource-path>

Developer designed part

You Choose What GOES Where



Tomcat Reality

- A web server has a max number of web applications before memory is exhausted
- The more web applications – the longer to start the server
- Deleting Mobile/REST web application does not necessarily recover memory
- Low memory symptoms: hung requests, does not start, process crash, no errors logged
- Tips:
  - ☑ Fewer web applications
    - Combine Mobile App & Service in a single WAR using PDSOE **Export**…
    - Combine multiple REST services into a single WAR using PDSOE **Export**…
  - ☑ Turn off PDSOE auto-publish
  - ☑ Restart web server periodically after n publishes

Using the Tools

PROGRESS EXCHANGE 2014

Progress Developer Studio
for OpenEdge is a.k.a.
PDS OE

Developing REST Services Using PDS OE

Design	✓ The RESTful APIs web services and resource URL space ✓ The URLs and verbs and map them to AppServer OO classes /procedures ✓ The HTTP message and response for each URL & verb combination
Code ABL	✓ Create OO classes/procedures in PDS OE project ✓ Turn class/procedure into a ‘service interface’
Create REST Service	✓ Create Relative Service URLs ✓ Use REST Expose Editor for Mapping HTTP artifacts to ABL parameters
Configure	✓ AppServer connection properties, logging (WEB-INF/adapters/runtime.props) ✓ Security template (appSecurity-xxx.xml)
Test & Publish	✓ Publish, Test, modify, Re-test, Re-publish till you are happy ✓ Export as .war or .zip (containing only REST Service’s .paar file)

Special Considerations for 10.2B & 11.1

- Cannot use dataset/temp-table as parameters (no automatic JSON export/import)
  - Can send/receive JSON or XML as Character parameter
- Cannot use Singleton Classes in AppServer
  - Can use remote, Single-Run, Singleton procedures
- No AppServer SSO by REST service

The proof of the pudding
is in eating it!

Demo with PDS OE

Moving to Production

PROGRESS EXCHANGE 2014

Moving Your REST Service to Deployment

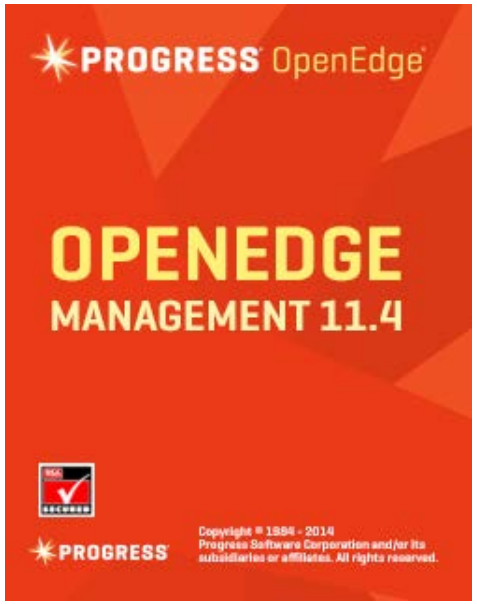
1	Export REST Service as .war (New Deployment) / .zip (Incremental Deployment).
2	Use production configured Tomcat. Deploy .war using a different Name
3	Edit default logging, AppServer connection properties, security configuration
4	Consider enabling AppServer SSO & CORS filter (for JavaScript clients)
5	Enable/Disable REST Services as per administration need

Tuning REST Service Security

- Edit the REST service's security defaults in PDSOE
- Spring Security always performs an
  - Authentication process [even for anonymous]
  - Authorization process [even for anonymous]
- Do not run production systems with the anonymous security model
- Recommendation: unit test with at least one restricted access security policy to verify your clients handle error conditions
- The Spring Security authorization uses roles [format: ROLE_<role-name>]
  - Roles name are obtained from where the user accounts are authenticated
  - Tip: Group all of your public access information into one REST service
  - Tip: The REST urls access controls are evaluated in the order found in the appSecurity file
  - Tip: Put the exception cases first, and general cases later
  - Tip: DO NOT REMOVE THE DENY ALL FOUND AT THE END

Remote Management Tools

- Needs REST Management Agent (oerm.war) installed in Tomcat
- Helps in Deploying & Managing REST Applications remotely
- Can be used by Production Administrations



Graphical Management Center

```
proenv>restman -help

Command Tool Usage for AppServer REST Adapter
=============================================================================
-help or -h                                 Display command line help.
-name or -i (Name)  (Required)              Name of AppServer REST Adapter
-user or -u (UserName)                      User name
-host or -r                                 Host name where AdminServer is running
-port                                       Port number of running AdminsServer
-webserverauth                              User name for Web Server Authentication
-war                                        location of WAR file
-appname                                    friendly name of a Web Service
-prop                                       name of property to be set
-value                                      new value of property
-query or -q                                query named WSA or Web Service
-deploy                                     deploy a Web Service
-undeploy                                   undeploy a named Web Service
-list                                       lists all deployed Web Services
-getdefaults                                displays the default properties
-setdefaults                                sets a default properties
-resetdefaults                              resets the default properties
-enable                                     enables a Web Service
-disable                                    disables a Web Service
-getstats                                   displays statistics
-resetstats                                 resets statistics
-getprops                                   displays the run-time properties
-setprops                                   sets a run-time properties
-resetprops                                 reset the run-time properties
-unpublish                                  unpublish a named REST Service
-republish                                  republish change-set to a deployed app
```

Command Line Utility (restman)

When Things Go Sideways

PROGRESS EXCHANGE 2014

Debug Guide

- Tools
  - A good proxy debug tool
  - Turning on HTTP message tracking in the web application
- Logs
  - Web server logs
  - Web application logs
  - Web application logging configuration
- Flow
  - Ping the web application
  - Get the RESTful API description
  - Ping the AppServer
  - Access the AppServer’s RESTful API

Debugging Available REST Services

- You can identify the information required to invoke an OpenEdge REST Web service by sending a GET request from a REST client in the following URI format:

```
http[s]://<host_name>:<port>/<rest_application_name>/rest
```

```
{
    "AppServerStatus":{
        "PingStatus":"[true|False]",
        "ABLReturnVal":"[return string from ABL code]"
        "Error": {
                    _errorMsg:"[Error Message (if any) while running the
                               ABL code]"
                    _errorNum:"[Error Number associated with the error
                               message]"
                }
    },
    "RESTServices":[
        {[REST Service1 JSON]},
        {[REST Service2 JSON]},
        ...
        {[REST ServiceN JSON]}
    ]
}
```

Frequently Used Operations

1	Identifying the AppServer and ABL code execution status
2	Constructing the URL to invoke an internal procedure
3	Setting media types for your REST request
4	Sending a value to the ABL input parameter
5	Receiving a value from the ABL output parameter

Pacific AppServer for OpenEdge

PROGRESS EXCHANGE 2014

Pacific AppServer for OpenEdge

- Based on Tomcat 7.0.55
- Configured for production by default
- Built-in adapters
  - APSV (AIA Replacement)
  - REST
  - SOAP
- Supports existing .paar and .wsm files

Want to Learn More about OpenEdge 11?

- **Role-based learning paths** are available for OpenEdge 11
- Each course is available as **Instructor-led training or eLearning**
- **Instructor-led training:**
  - $500 per student per day
  - https://www.progress.com/support-and-services/education/instructor-led-training
- eLearning:
  - Via the Progress Education Community (https://wbt.progress.com):
  - OpenEdge Developer Catalog: $1500 per user per year
  - OpenEdge Administrator Catalog: $900 per user per year
- User Assistance videos:
  - https://www.progress.com/products/pacific/help/openedge

PROGRESS EXCHANGE 2014

Visit the Resource Portal

- Get **session details** & presentation **downloads**
- Complete a **survey**
- Access the latest Progress **product literature**

www.progress.com/exchange2014

PROGRESS