Progress OpenEdge REST

Deploying, Managing, and Troubleshooting your REST Web Application

Kumar Navneet
Principal Software Engineer
Progress Software

David Cleary
Principal Software Engineer
Progress Software

October 8, 2013

PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Agenda

- Architecture of REST Applications and REST Management Agent
- Securing Your REST Application
- Deploying REST Applications in Production
- Troubleshooting a Production REST Application

Architecture or Protocol?

- HTTP
  - HyperText Transfer Protocol
  - Defines text-based message format and exchange protocol
  - Foundation for communications in the cloud
- SOAP
  - Simple Object Access Protocol
  - Defines an XML-based message format and exchange protocol
  - Specifies bindings to HTTP and other transports
- REST
  - Representational State Transfer
  - Style of software architecture for distributed systems
  - Describes the architecture of HTTP
  - Predominate web API design model

REST Architecture

- There is no REST Specification
- REST does not define a message format
- REST does not define a message exchange protocol
- REST is stateless
  - Each client request contains all the information to service the request
- REST is resource-based
  - Resources are identified by URI (Uniform Resource Identifier)
  - Server provides a representation of resource to client
  - Client can manipulate resource on server through the representation
- CRUD Model
  - Create, Read, Update and Delete

Creating REST Applications

- Progress Developer's Studio for OpenEdge (a.k.a. PDS OE)
  - On Premise development IDE (Integrated Development Environment) to develop REST Services
  - Define Service Interface using annotations
  - Map ABL parameters to HTTP artifacts
  - Publish on OE Webserver (Development Tomcat Server). Test, make changes, Re-publish
  - Export REST Service (s) as a fully deployable WAR
  - Export REST Service as a .paar file that can be re-published in an already deployed WAR
- OpenEdge REST Application
  - Web application (.war file) to access ABL business logic RESTfully
  - Contains one or more REST service
  - Each REST service is contained in a Progress Archive file with a .paar file extension
  - Supported Webserver is Tomcat 7.0 or later

REST Management Agent (a.k.a. OERM)

Shipped as oerm.war

Installed in Tomcat that comes with OE installation

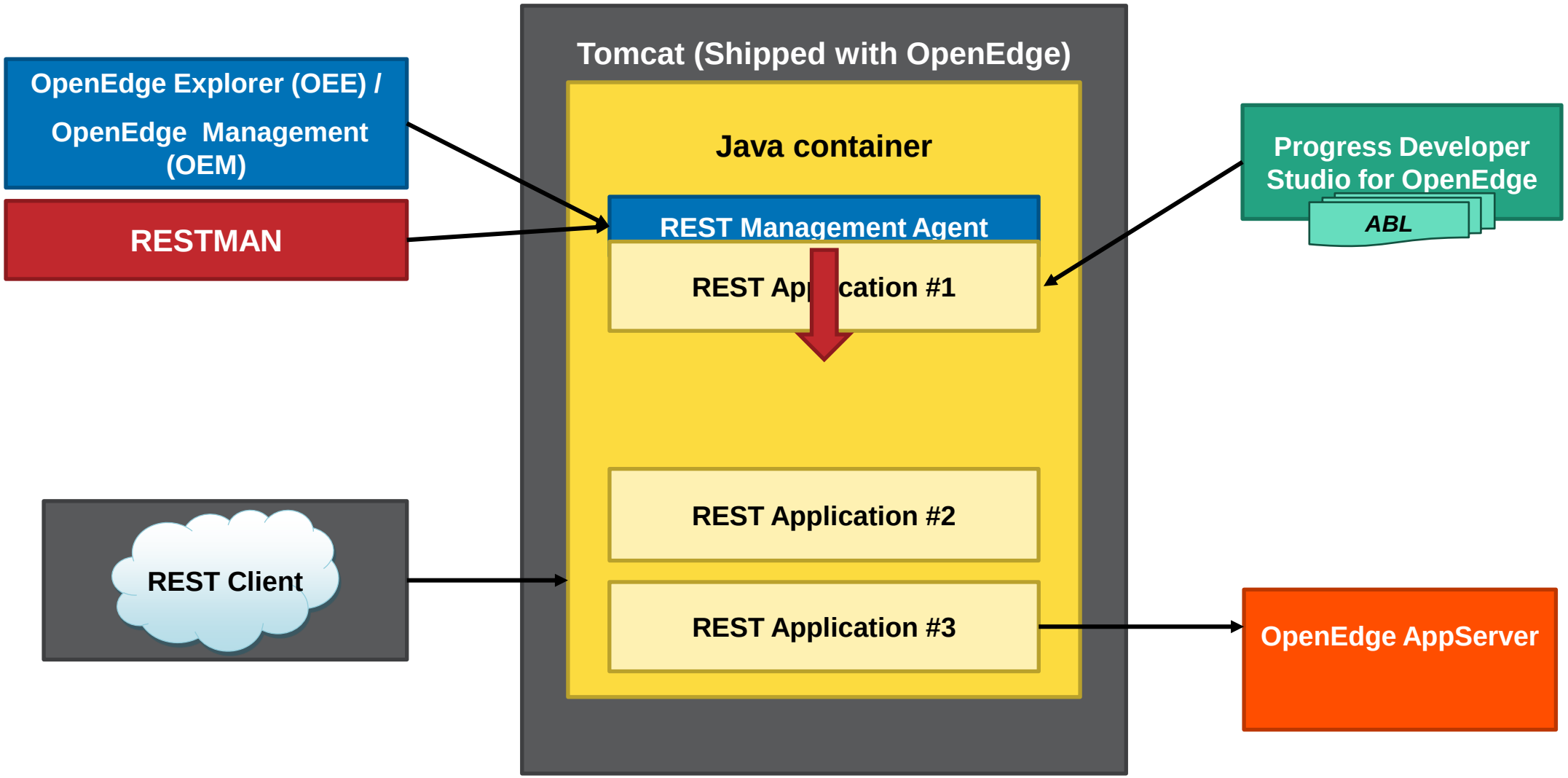
On a Production Webserver, it must be manually deployed

Is a Web application that deploys and manages other OpenEdge REST web applications

Does management on behalf of PDS OE, OEE/OEM and RESTMAN

- Publish, republish and un-publish a REST Service
- View/Edit/reset application properties and statistics
- View and manage logging

REST Management Agent and REST Application Development



Agenda

- Architecture of REST Management Agent and REST Applications
- Securing Your REST Application
- Deploying REST Applications in a Production Environment
- Troubleshooting a Production REST Application

Securing Your REST Application

REST application & REST Management Agent are full fledged Java web application

They are subject to the same risks & security requirements as any other web application.

Accepted standard for web application security is published by the OWASP (Open Web Application Security Policies)

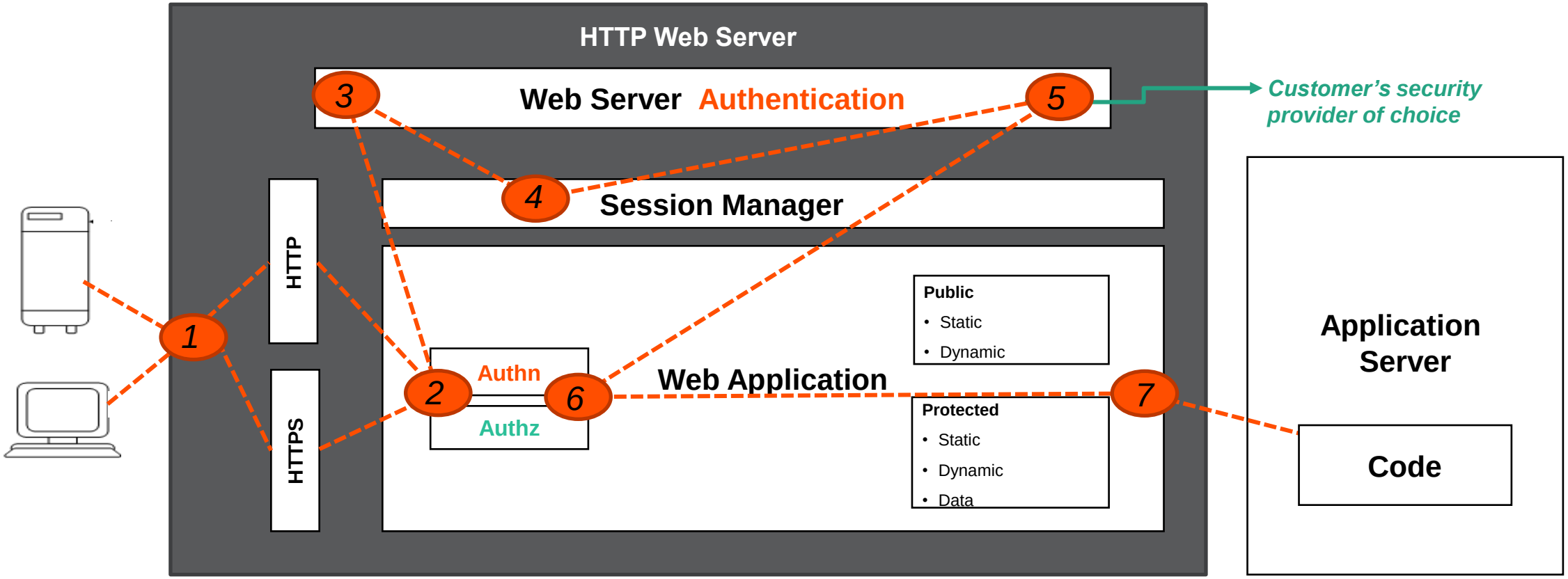
The java container security services are supported by OpenEdge REST applications and Management Agents.

OpenEdge REST applications come with the Spring Security service embedded into them and is recommended as it provides greater level of control

NOTE : The Spring Security framework extends the java container's security services, not replaces them.

General Web Application Security

- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge; it addresses the industry standards that customers will already be using

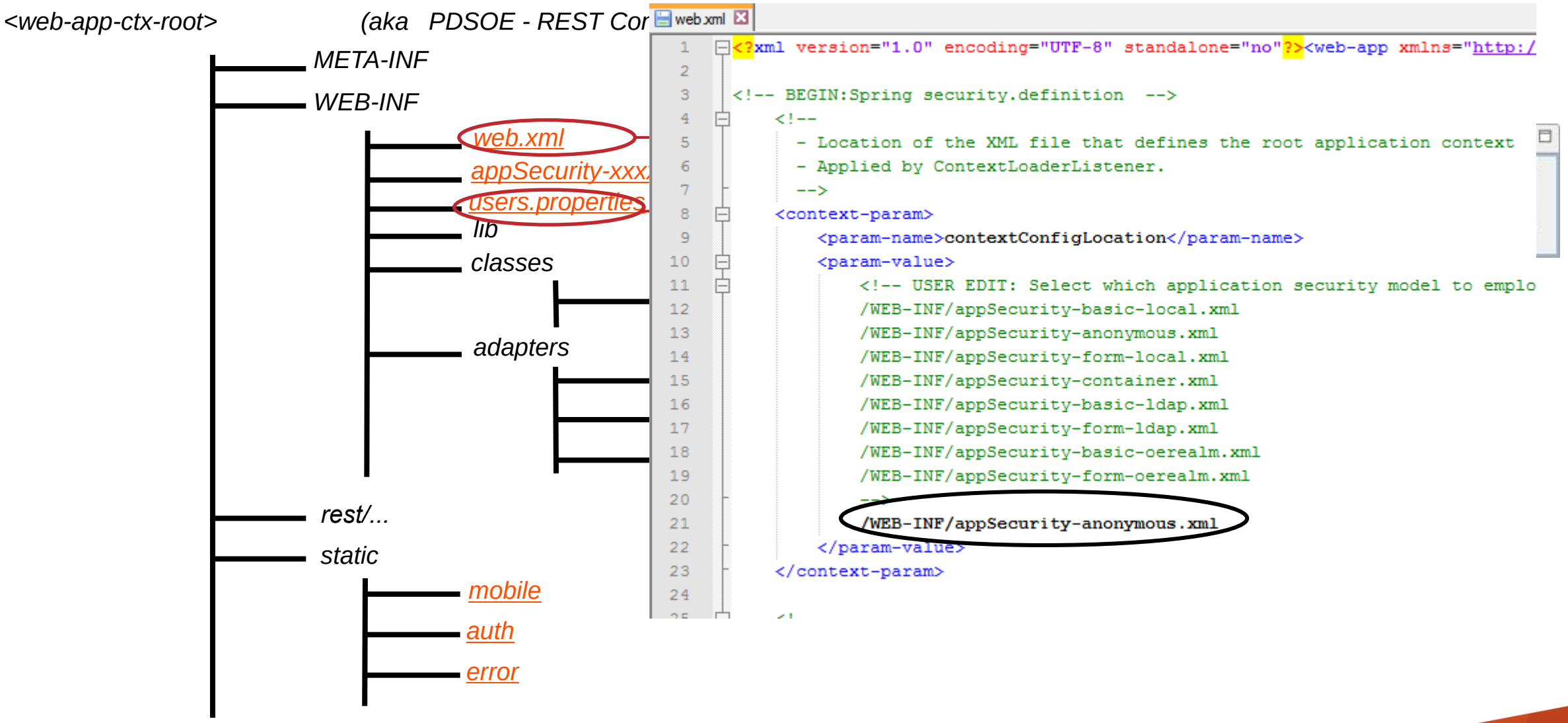


Web Application Authentication Models

- Anonymous : The 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*

** Other authentication models available - not certified*

Configuring OpenEdge Web Application



External Providers

- LDAP
  - Lightweight Directory Access Protocol
  - Enterprise standard for user authentication and authorization
  - Supported via Spring Security
- OpenEdge SPA (Single Point of Authentication)
  - Based on custom realm support in OpenEdge BPM
  - Hooks into custom authentication systems
  - Provides authentication support only! No authz
  - Can use a digest authentication methodology – No passwords over wire
  - Implement built-in OOABL Interface
    - Progress.Security.Realm.IHybridRealm
    - Reference implementation using _user table
  - $DLC/src/samples/security (11.3 or later)

AppServer Single Sign-On

- ClientPrincipal authentication token created from Spring authentication token
- ClientPrincipal passed with each request to Agent
- Request context information available via
  - session:current-request-info:GetClientPrincipal()
  - session:current-request-info:clientContextID.
  - session:current-request-info:procedureName.
- Client-Principal SESSION-ID equals clientContextID attribute
- Client-Principal STATE attribute is SSO

Agenda

- Architecture of REST Management Agent and REST Applications
- Securing Your REST Application
- Deploying REST Applications in Production
- Troubleshooting a Production REST Application

On-premise Deployment

- Legacy Deployment
  - Physical system with Tomcat, REST Management Agent, and REST Applications
  - OpenEdge AppServer on same system, different system, or multiple systems (NameServer Load Balancing)
- Tomcat configuration
  - Request threads (default 200)
  - Spare threads (default 4)
- REST Application Configuration
  - Session-free
    - Stateless pipelines request
    - Performs a Connect-Run-Disconnect
  - *Sessions (min, max, initial, idle timeout)
    - Tomcat request thread = 1 session

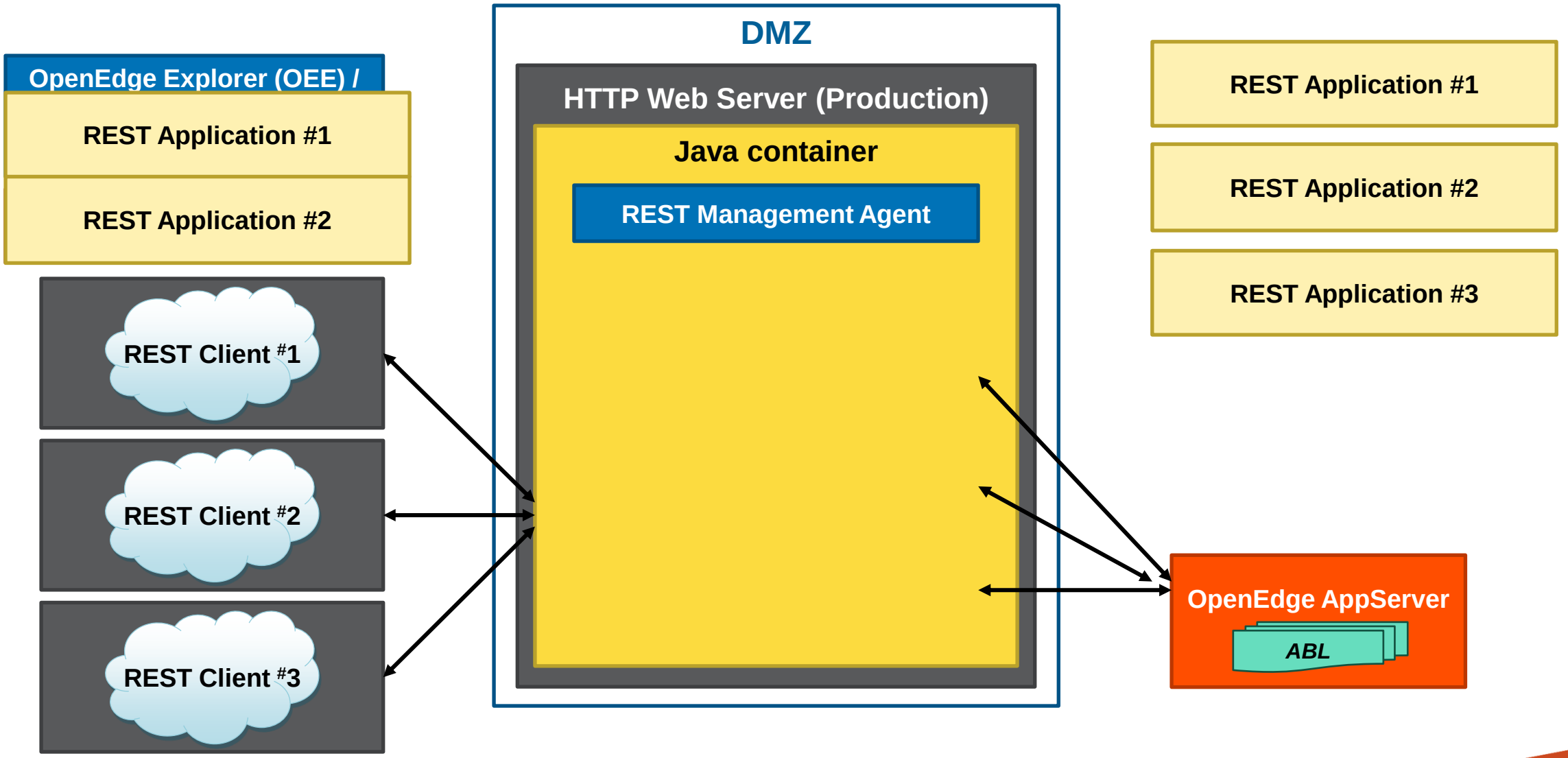
Private Cloud Deployment

- Public Cloud Infrastructure within firewall
  - Applications and data secured by corporate firewall
  - Management agent protected from outside threats
- VM Deployment
  - One or more virtual machines
  - New VM’s spin up to meet demand
- NameServer load balancing to add new resources
  - AppServer clones spin up during times of demand
  - Session-free mode routes individual requests to different VMs

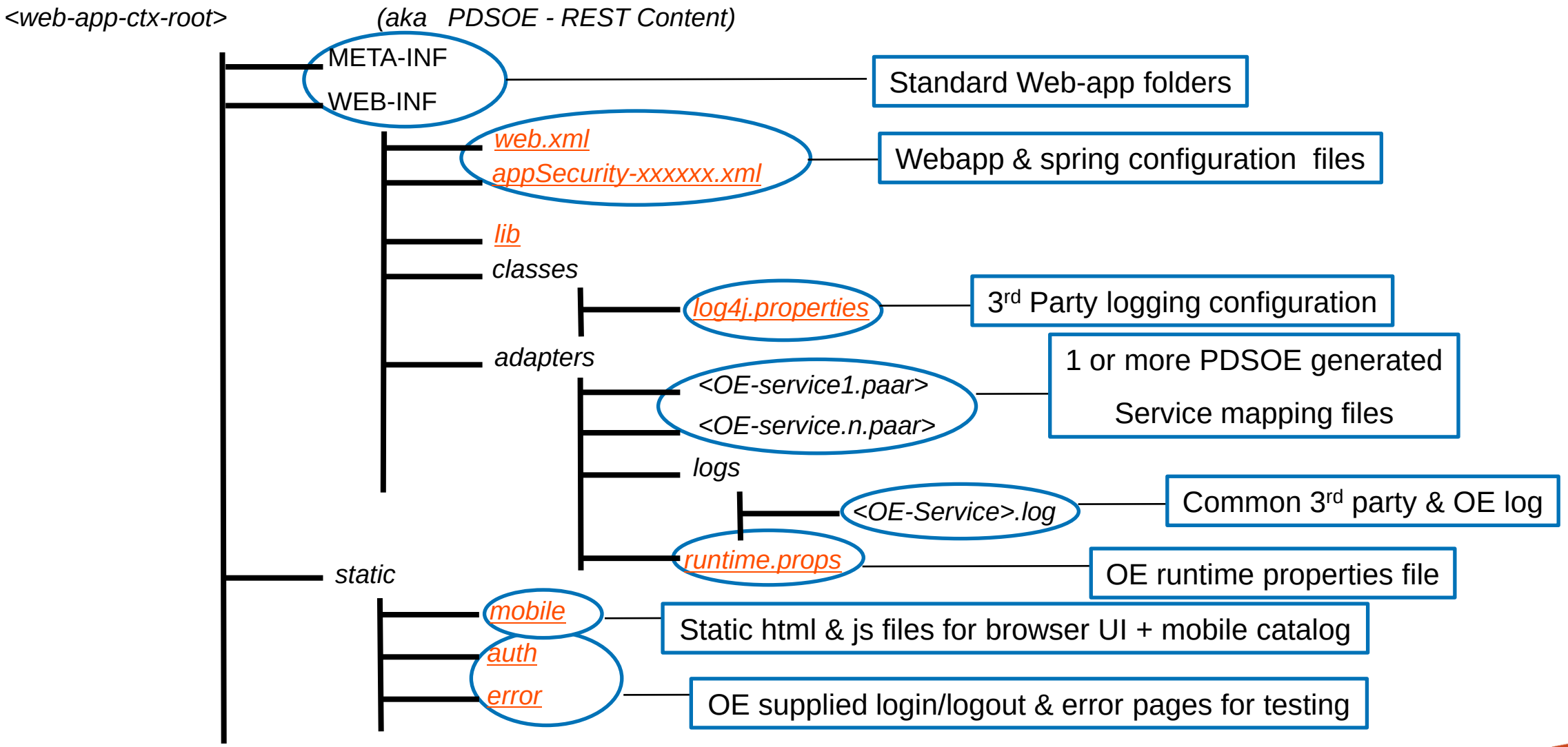
Public Cloud Deployment

- Provides highly scalable and reliable architecture
  - Just pay for what you use
- Hosted outside of corporate firewall
  - Infrastructure managed by others
  - Application and data need to be locked down
- No REST Management Agent
  - No HTTP/S access to admin and config application
  - Applications manually deployed as .war
  - Services deployed as .paar files
  - Properties configured with text editor

Production Deployment



Anatomy of an OpenEdge Web Application



Deploying OpenEdge REST Applications

Remotely (using REST Management Agent)	Manually (w/o REST Management Agent)
▪ Create an instance of OERM on remote machine where OEE/OEM/restman runs	▪ N/A
▪ Deploy and undeploy can be done remotely	▪ Need to manually deploy the REST applications
▪ Incremental publish of REST Service can be done in one step remotely	▪ Incremental publish of REST Service needs manual steps viz. (a) copy the .paar file in adapters folder and, (b) modifying web.xml file to add the .paar file name to a context parameter called archiveFiles
▪ One or all deployed REST Applications can be disabled in one step remotely	▪ In order to disable a REST Application you need to change serviceAvailable value by editing runtime.props file, and you need to reload the application
▪ You can change log levels dynamically	▪ Changing log level requires reload of the application
▪ On OEM console, you can view of Application logs, add alerts based on regular expressions in the log.	▪ N/A

Logging for Remote Deployment and Administrative Ttasks

- admserv.log
  - If you deploy/ edit or view a property using OEE/OEM or restman, the request goes through AdminServer.
  - You can increase the logging level by adding –DLogLevel=5 in AdminServerPlugin.properties file under AdminServer plugin section.
- oerm.log
  - The REST Management Agent does the deployment in the web container
  - You can dynamically increase the loggingLevel of the REST Managment agent instance using OEM or restman to get more logging
- Tomcat logs
  - Catalina.log, localhost.log, host-manager.log, localhost_access_log.txt

Agenda

- Architecture of REST Management Agent and REST Applications
- Securing Your REST Application
- Deploying REST Applications in Production
- Troubleshooting a Production REST Application

Checklist Before You Start Accessing Your REST Service

- ✓ The Tomcat server is started with no errors in catalina log
- ✓ The Appserver is started and the ABL business code is deployed in the OE Work Directory
- ✓ The database required for the service is up and running
- ✓ Type http(s)//<host-name>:port/<application> and hit go. This must returns 200 OK response
- ✓ The application is ENABLED for access by admin and other users.
  - ✓ serviceAvailable property defined in application's runtime.props must be set to 1 (ENABLED)
  - ✓ If application is deployed under REST Management agent, adminEnabled and webAppEnabled properties of the REST Manager instance must be set to true.

Trouble Accessing REST Service

- Check if the Application is up and running
  - Type http(s)://<host-name>/<application>/rest in a browser and you must get WADL (Web Application Description Language) as response
- 401 Unauthorized or 403 Forbidden
  - Check user id and password which should match the app-security mode
  - Turn on DEBUG for spring and security related packages (org.springframework and com.progress.rest.security) in log4j.properties
- 415 Unsupported Media Type
  - Check the Media types for HTTP request and response (application/json)
- 501 service unavailable
  - Application needs to be enabled
- 500 Internal Server Error
  - Check application log, appserver broker and server logs

Troubleshooting Production REST Application

- Log files
  - <application>/WEB-INF/adapters/<application>.log
- Increase the logging level of the application (4 is highest, 2 is default)
  - Using OEM or restman (dynamic, doesn't need application context reload)
  - Manually setting serviceLoggingLevel in <application>/WEB-INF/adapters/runtime.props
- Increase the fault level of the application (4 is highest, 2 is default)
  - Using OEM or restman (dynamic, doesn't need application context reload)
  - Manually setting serviceFaultLevel in <application>/WEB-INF/adapters/runtime.props
- Adding OpenClient specific entry type like BrokerClient to the application (REST is default)*
  - When the problem is during interaction with AppServer.
  - Using OEM or restman (dynamic, doesn't need application context reload)
  - Manually setting serviceLogEntryType in <application>/WEB-INF/adapters/runtime.props

** Refer to Java OpenClient documentation for complete set of LogEntryTypes*

JSON for ABL Dataset or Temptable

- Verify that JSON representation of an ABL Temptable or Dataset is proper
- Example of an ABL Temptable mapped to a node named custRecord in HTTP request body

Log File – Check Points

- Verify if the Catalina log shows any SEVERE (e.g. permGen space) & if yes rectify it.
- Verify if the REST Adapter receives proper parameter values from the REST client

```
2013-09-16 11:35:55.797 [DEBUG][REST] Procedure name: clientInformation.p
2013-09-16 11:35:55.797 [TRACE][REST] Populating CafParamArray
2013-09-16 11:35:55.797 [DEBUG][REST] Param name: custId
2013-09-16 11:35:55.798 [DEBUG][REST]    Param type: 4
2013-09-16 11:35:55.798 [DEBUG][REST]    Param ordinal: 0
2013-09-16 11:35:55.798 [DEBUG][REST]    Param mode: 1
2013-09-16 11:35:55.798 [DEBUG][REST]    Param value: 1
2013-09-16 11:35:55.798 [DEBUG][REST] Converted input paramter: Type = java.lang.Integer Value = 1
```

Information about Parameters passed to REST Adapter

- Verify if the REST Adapter calls proper ABL procedures on the AppServer

```
2013-09-16 11:35:55.883 [INFO][REST] Running an internal procedure: clientInformation.p->GetAllCustomerOrders
```

- Verify if the REST Adapter reads proper data from the AppServer & Exits successfully

```
2013-09-16 11:35:55.904 [TRACE][REST] _REST [uBroker-Client  ] readMsgbuf[1268]
2013-09-16 11:35:55.904 [TRACE][REST] 22 43 75 73 74 6f 6d 65 72 4f 72 64 65 72 73 22 "CustomerOrders"

2013-09-16 11:35:55.914 [INFO][REST] Successfully exiting AppServerProducer
2013-09-16 11:35:55.916 [DEBUG][REST] Entering route :: ExecuteOUT
```

- Verify that Appserver and database logs don't contain any error

Using Tomcat Container Provided Request Dumper Filter for Debugging

- Logs information from the HTTP request and response in a log file of your choice
- You need to add this filter in application's web.xml

```
<filter>
    <filter-name>requestdumper</filter-name>
    <filter-class>
        org.apache.catalina.filters.RequestDumperFilter
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>requestdumper</filter-name>
    <url-pattern>/rest/CustomerService/*</url-pattern>
</filter-mapping>
```

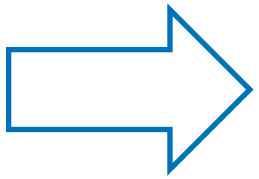
- You need to make changes in $CATALINA_BASE/conf/logging.properties to redirect logs to a dedicated file

** For more information on using this filter refer Tomcat 7 documentation*

WADL Output

- WADL stands for Web application Description Language. It is not a standard like WSDL for SOAP.
- The URI information needed to access a REST service is determined at Development Time.
- OpenEdge **does not recommend** relying on WADL output for URI construction.
- However, in production environment the WADL output can be used to cross check if the URL and HTTP verbs are valid or not

```
- <application>
    <grammars/>
  - <resources base="http://localhost:8980/Test1Service/rest/Test1Service">
    - <resource path="/test1">
      - <method name="GET">
        - <request>
            <param name="custId" style="query" default="" type="xs:string"/>
          </request>
          <response status="204"/>
        </method>
      - <method name="POST">
          <response status="204"/>
        </method>
      </resource>
    </resources>
  </application>
```



http://localhost:8980/Test1Service/rest/Test1Service/test1

☑ GET

☒ DELETE

☑ POST

** The WADL output will be enhanced to give more debugging information in upcoming OE releases*

In Conclusion…

- Understanding of REST Application, Adapter and Management Agent Architecture
- Methods and techniques for securing your application
- Deploying your application both manually and through the Management Agent
- Checklist of troubleshooting tips for your production application

Questions?

Additional Resources

- PUG Challenge Americas – http://pugchallenge.org
  - Server Access – REST of the Story: http://pugchallenge.org/downloads/413_REST_of_Story.pptx
  - REST Security - http://pugchallenge.org/downloads/344_REST_Security.pdf
- OE Mobile Community on PSDN - http://communities.progress.com/pcom/community/psdn/openedge/oemobile
- Mobile Debugging Tips - http://communities.progress.com/pcom/docs/DOC-107884

PROGRESS