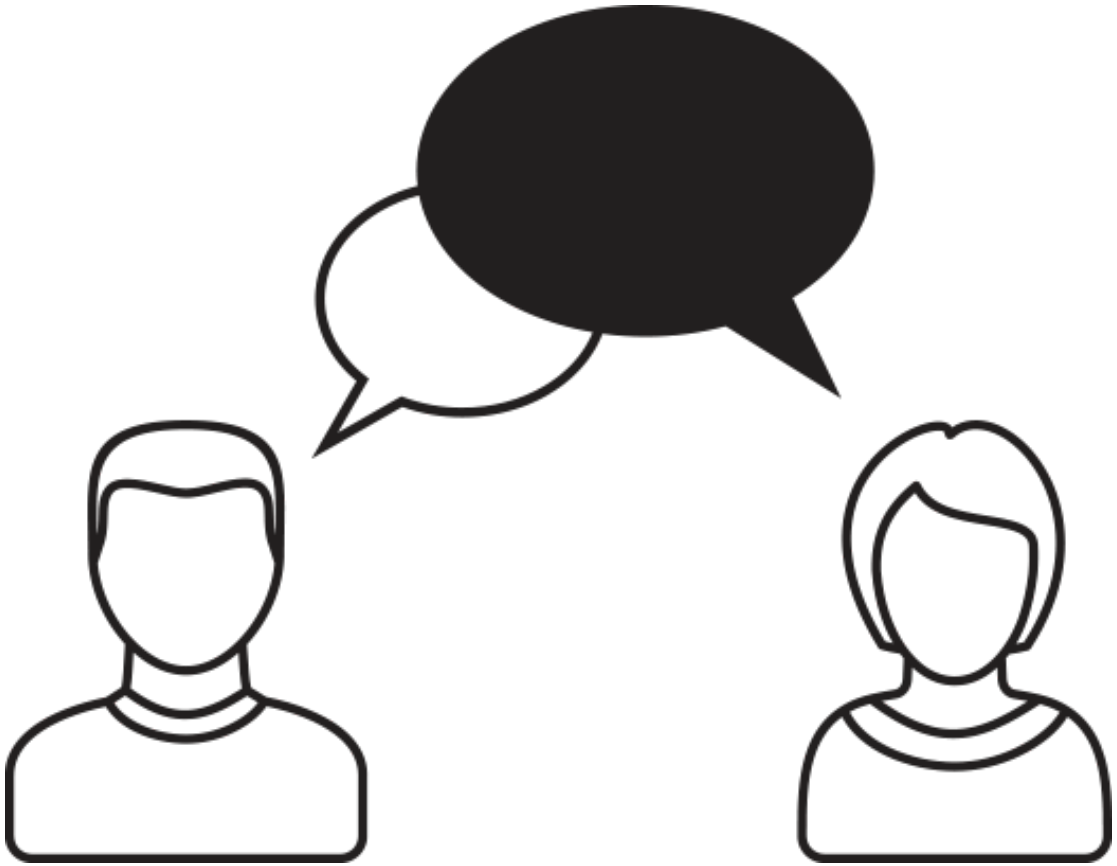

regoUniversity 2017
GEL Scripts | Advanced
Your Guides: Ben Rimmasch, Yogesh Renapure

Introductions

- Take 5 Minutes
- Turn to a Person Near You
- Introduce Yourself



Agenda

- Accessing JAVA Classes and Methods
- SOAP Web Services (XOG)
- REST Web Services
- File Handling
- Integrations

Accessing JAVA Classes and Methods

- Instantiate Java Classes and Call Java Methods
  - core:new
    - Use this GEL tag to instantiate Java classes
    - For Ex:

```
<core:new className="java.net.URL" var="myUrl">
    <core:arg type="java.lang.String" value="${myUrlString}" />
</core:new>
```

  - core:invoke
    - Use this GEL tag to call a method on an instantiated object
    - For Ex:

```
<core:invoke method="openConnection" on="${myUrl}" var="myConnection"/>
```

Accessing JAVA Classes and Methods Cont'd

- core:expr
  - Use this GEL tag to call a method on an instantiated Java object where you do not require access to the result of the operation.
  - In the REST calls, this tag can be used to set request headers
  - For Ex:

```
<core:expr value='${connection.setRequestMethod("POST")}'/>
```

- core:invokeStatic
  - Use this GEL tag to call a static method of a Java class
  - For Ex:

```
<core:invokeStatic className="com.niku.union.utility.Base64" method="encode" var="encodedString">
    <core:arg value="${valueToEncode}"/>
</core:invokeStatic>
```

SOAP Web Services (XOG): Namespaces

- Think of XML namespaces as a way to organize tag names (and attribute names) into groups
- The namespaces below are required to call CA PPM SOAP Web Services (XOG)

```
<gel:script xmlns:core="jelly:core"
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
    xmlns:xog="http://www.niku.com/xog"
    xmlns:soap="jelly:com.niku.union.gel.SOAPTagLibrary"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
```

SOAP Web Services (XOG): Obtain Session ID

- The following snippet is used to generate a Session ID. This approach cannot be used for cross-environment SOAP calls

```
<!-- Get sessionId by username -->
<gel:parameter var="username" default="admin"/>
<core:new className="com.niku.union.security.DefaultSecurityIdentifier" var="secId" />
<core:invokeStatic var="userSessionCtrl" className="com.niku.union.security.UserSessionControllerFactory" method="getInstance" />
<core:set var="secId" value="${userSessionCtrl.init(username, secId)}"/>
<core:set var="sessionID" value="${secId.getSessionId()}"/>

<core:choose>
    <core:when test="${sessionID == null}">
        <gel:log level="ERROR"> Unable to obtain a Session ID. </gel:log>
    </core:when>
    <core:otherwise>
        <!-- Execute XOG -->
    </core:otherwise>
</core:choose>
```

SOAP Web Services (XOG): Build XML

- Use the <gel:parse> tag to build a XOG XML
  - For Ex:

```
<gel:parse var="userXML">
  <NikuDataBus xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../xsd/nikuxog_user.xsd">
  <Header action="write" externalSource="NIKU" objectType="user" version="13.3.0.286"/>
    <Users>
      <User externalId="" isLDAP="false" uiThemeDefaultPartitionCode="" userLanguage="English"
        userLocale="en_US" userName="${row.userName}" userStatus="${row.userStatus}"
        userTimezone="America/Los_Angeles" userType="INTERNAL">
        <PersonalInformation emailAddress="${row.email}" firstName="${row.firstName}" lastName="${row.lastName}"/>
        <Resource resourceId="${row.resourceId}"/>
        <Groups>
      <Group id="${row.groupId}">
            <nls description="${row.groupDescr}" languageCode="en" name="${row.groupName}"/>
       </Group>
    </Groups>
      </User>
    </Users>
  </NikuDataBus>
</gel:parse>
```

SOAP Web Services (XOG): Read, Write, Delete Nodes

- Read Node
  - Use the below snippet to read the description of a group from userXML

```
<gel:set asString="true"
    select="$userXML//Users/User/Groups/Group/nls/@description"
    var="groupDescr"/>
```

- Write Node
  - Use the below snippet to create a new group node in the userXML

```
<gel:parse var="insertNode">
   <Group id="pm">
      <nls description="Project Managers Group" languageCode="en" name="Project Managers"/>
   </Group>
</gel:parse>
<gel:set insert="true" select="$userXML//Users/User/Groups/Group" value="${insertNode}"/>
```

SOAP Web Services (XOG): Read, Write, Delete Nodes

- Delete Node
  - Use the below snippet to delete a group node from userXML

```
<gel:set select="$userXML//Users/User/Groups/Group[@id='pm']" var="groupToDelete" />
<core:set value="${groupToDelete.getParentNode()}" var="parent" />
<core:set value="${parent.removeChild(groupToDelete)}" var="void" />
```

SOAP Web Services (XOG): Execute XOG

- Use the below snippet to execute a XOG SOAP web service call

```
<!-- Execute XOG -->
<soap:invoke endpoint="internal" var="result">
  <soap:message>
    <soapenv:Envelope>
      <soapenv:Header>
        <xog:Auth>
          <xog:SessionID>${sessionID}</xog:SessionID>
        </xog:Auth>
      </soapenv:Header>
      <soapenv:Body>
        <gel:include select="$userXML"/>
      </soapenv:Body>
    </soapenv:Envelope>
  </soap:message>
</soap:invoke>
```

SOAP Web Services (XOG): Execute XOG Cont'd

- Obtaining XOG endpoint
  - endpoint = "Internal"
    - Quick way to setup SOAP call to the same instance

```
<!-- Execute XOG -->
<soap:invoke endpoint="internal" var="result">
    …
    …
</soap:invoke>
```

  - From properties.xml using JAVA Classes

```
<core:invokeStatic className="com.niku.union.config.ConfigurationManager" method="getInstance" var="config" />
    <core:set var="URL" value="${config.getProperties().getWebServer().getWebServerInstance(0).getSslEntryUrl()}" />
<gel:log level="INFO">Environment is: ${URL}</gel:log>
```

SOAP Web Services (XOG): Execute XOG Cont'd

- From properties.xml using GEL tags
  - If the entryUrl doesn't work, the sslEntryUrl or schedulerUrl can be pulled

```
<core:invokeStatic className="java.lang.System" method="getenv" var="nikuHome">
    <core:arg value="NIKU_HOME"/>
</core:invokeStatic>
<gel:parse file="${nikuHome}/config/properties.xml" var="propertiesXML"/>
<gel:set asString="true"
    select="$propertiesXML/properties/webServer/webServerInstance[@id='app']/@entryUrl"
    var="host_url"/>
<gel:log level="INFO">Host URL: ${hostUrl}</gel:log>
<gel:log>${hostUrl}/niku/xog</gel:log>
```

SOAP Web Services (XOG): Parse Output

- To ensure success and as part of error handling the following snippet demonstrates how to parse through the XML response returned by the SOAP call to the XOG interface

```
<!-- Parse Results -->
<gel:set asString="true" select="$result//XOGOutput/Status/@state" var="XOGState"/>
<gel:set asString="true" select="$result//XOGOutput/Statistics" var="xogStats"/>
<gel:set asString="true" select="$result//XOGOutput/Statistics/@failureRecords" var="XOGFailures"/>

<core:choose>
  <!-- Success -->
  <core:when test="${XOGState == 'SUCCESS' and XOGFailures == 0}">
    <gel:log level="INFO">User XOG Stats: ${xogStats} </gel:log>
  </core:when>
  <!-- Failure -->
  <core:otherwise>
    <gel:log level="INFO">User XOG Stats: ${xogStats} </gel:log>
    <gel:log level="WARN"><gel:expr select="$userXML/"/></gel:log>
    <gel:log level="ERROR"><gel:expr select="$result/"/></gel:log>
  </core:otherwise>
</core:choose>
```

SOAP Web Services (XOG) : XOG Logout

- Use below snippet to end the XOG session

```
<soap:invoke endpoint="internal" var="xogLogOut">
  <soap:message>
    <soapenv:Envelope>
      <soapenv:Header>
        <xog:Auth>
          <xog:SessionID><![CDATA[${cSessionID}]]></xog:SessionID>
        </xog:Auth>
      </soapenv:Header>
      <soapenv:Body>
        <xog:Logout />
        <gel:log level="INFO">Logged Out of XOG</gel:log>
      </soapenv:Body>
    </soapenv:Envelope>
  </soap:message>
</soap:invoke>
```

REST Web Services

- A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.
- REST technology is generally preferred to the more robust SOAP technology because REST leverages less bandwidth, making it more suitable for internet usage.
- Results typically JSON (JavaScript Object Notation) response format
- Currently CA PPM REST APIs are not supported for customer or partner use
- The steps described here can be used to call REST APIs of any application from CA PPM

REST Web Services – Authentication

- Below are the most commonly used authentication methods for REST web services
  - Basic Authentication
    - Authenticate using Base64 encoded string of Username and Password

```
<core:invokeStatic className="com.niku.union.utility.Base64" method="encode" var="encodedString">
    <core:arg value="${userName}:${password}" />
</core:invokeStatic>
<core:set value="Basic ${encodedString}" var="authKey" />
```

  - Token Based Authentication
    - Token from Login and Logout API
    - OAuth 2.0 tokens
    - Application generated tokens

REST Web Services – REST Call

- Maintain Session
  - CA PPM REST API requires to maintain the session. Use the below snippet to maintain the CA PPM REST session

```
<core:new className="java.net.CookieManager" var="cManager" />
<core:getStatic var="cPolicy" className="java.net.CookiePolicy" field="ACCEPT_ALL"/>
<core:invoke var="setCPolicy" method="setCookiePolicy" on="${cManager}" >
    <core:arg value="${cPolicy}"></core:arg>
</core:invoke>
<core:invokeStatic className="java.net.CookieHandler" method="setDefault">
    <core:arg type="java.net.CookieHandler" value="${cManager}" />
</core:invokeStatic>
```

REST Web Services – REST Call Cont'd

- Below are the steps to Call CA PPM REST API
  - Form Input JSON

```
<core:set var="requestJSON" escapeText="false">
    {
      "code": "REST01",
      "isOpenForTimeEntry": "true",
      "description": "Project Created via REST",
      "isActive": "true",
      "name": "REST Project"

    }
</core:set>
```

  - Open Connection to CA PPM REST URI

```
<core:set var="restEndPoint" value="http://ppm.example.com/ppm/rest/v1/projects"/>
<core:new var="restUrl" className="java.net.URL">
    <core:arg type="java.lang.String" value="${restEndPoint}"/>
</core:new>
<core:invoke var="connection" on="${restUrl}" method="openConnection"/>
```

REST Web Services – REST Call Cont'd

- Set Request Headers

```
<core:expr value='${connection.setRequestMethod("POST")}'/>
<core:expr value='${connection.setRequestProperty("Authorization", authKey)}'/>
<core:expr value='${connection.setRequestProperty("Content-type", "application/json")}'/>
<core:expr value='${connection.setRequestProperty("Accept", "application/json")}'/>
<core:expr value='${connection.setRequestProperty("Connection", "keep-alive")}'/>
<core:expr value="${connection.setDoOutput(true)}"/>
```

- Write Input JSON to REST API

```
<core:invoke var="outputStream" on="${connection}" method="getOutputStream" />
<core:new var="outputStreamWriter" className="java.io.OutputStreamWriter">
    <core:arg type="java.io.OutputStream" value="${outputStream}" />
</core:new>
<core:expr value="${outputStreamWriter.write(requestJSON)}" />
<core:expr value="${outputStreamWriter.close()}" />
```

REST Web Services – REST Call Cont'd

- Process REST Output

```
<core:invoke var="restOutput" on="${connection}" method="getInputStream" />
<core:invoke var="restResponseCode" on="${connection}" method="getResponseCode" />
<core:choose>
    <core:when test="${restResponseCode == '200'}">
        <gel:log>Successfully created CA PPM Project</gel:log>
        <!-- Convert REST output stream to String -->
        <core:invokeStatic className="org.apache.cxf.helpers.IOUtils" method="toString" var="projectOutputString">
            <core:arg type="java.io.InputStream" value="${restOutput}"/>
            <core:arg value="UTF-8"/>
        </core:invokeStatic>
        <core:expr value="${restOutput.close()}"/>
        <!-- Convert output string to JSON Object -->
        <core:new className="org.json.JSONObject" var="projectJsonObject">
            <core:arg type="java.lang.String" value='${projectOutputString}'/>
        </core:new>
        <core:set var="prjInternalID" value="${projectJsonObject.get('_internalId')}"/>
        <gel:log>Project Id: ${prjInternalID}</gel:log>
    </core:when>
    <core:otherwise>
        <gel:log>Failed to create CA PPM Project</gel:log>
    </core:otherwise>
</core:choose>
</core:catch>
```

File Handling: Overview

- The CA provided GEL tag libraries can
  - Read files and write to a file
  - Perform a set of FTP operations
  - Parse nodes and attributes of XML or comma-delimited files with some limitations
- The CA provided GEL tag libraries cannot
  - Create a directory to put files in
  - Move files around
  - Delete files after it is done with them
- Java can almost always fill in the gaps

File Handling: Read File

- Below is the example script to read a pipe delimited file

```
<gel:script xmlns:core="jelly:core"
    xmlns:gel="jelly:com.niku.union.gel.GELTagLibrary"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:files="jelly:com.niku.union.gel.FileTagLibrary">

    <gel:parameter var="vFileName" default="/fs0/CA PPM1/share/RESOURCES.CSV"/>

    <files:readFile fileName="${vFileName}" delimiter="\|" var="vResourceData" embedded="false"/>

    <core:forEach items="${vResourceData.rows}" var="row" begin="1" end="10">
        <gel:log level="INFO"> Resource Last Name: ${row[0]} </gel:log>
        <gel:log level="INFO"> Resource First Name: ${row[1]} </gel:log>
    </core:forEach>

</gel:script>
```

  - ‘embedded’ attribute determines if data is surrounded by quotes or not. Default value is false.

File Handling: Write File

- Writing files is a way to export data, perhaps to be used in an integration by another application

```
<file:writeFile delimiter="," embedded="false" fileName="Resources.csv">
    <sql:query dataSource="${CA_PPMDS}" escapeText="false" var="result">
        <![CDATA[
        SELECT  u.first_name firstName,
                u.last_name lastName,
                u.user_name userName
        FROM    cmn_sec_users u
        WHERE  u.user_status_id = 200
        ]]>
    </sql:query>
    <core:forEach items="${result.rows}" trim="true" var="row">
        <file:line>
            <file:column value="${row.userName}"/>
            <file:column value="${row.lastName}"/>
            <file:column value="${row.firstName}"/>
        </file:line>
    </core:forEach>
</file:writeFile>
```

  - Use serialize tag to write XML content to an XML file

```
<gel:serialize fileName="C:\XOG\output\lookup_mapping.xml" var="${loadContent}"/>
```

File Handling: FTP

- By including the namespace for the FTPTagLibrary, GEL has the ability to read and write files on an FTP Server
  - This library does not support SFTP, FTPS or passive mode
  - Namespace
    - <gel:script **xmlns:gel="jelly:com.niku.union.gel.FTPTagLibrary"**>
  - Tags
    - ftp:open
    - ftp:put
    - ftp:get

File Operations: FTP Cont'd

- Below is the snippet to read a file from FTP location

```
<ftp:open hostName="myclarityserver" user="niku" password="clarity">
    <ftp:get localDir="c:/temp" fileName="app-ca.log" remoteDir="/niku/clarity/logs"/>
</ftp:open>
```

- Below is the snippet to write a file to FTP location

```
<ftp:open hostName="localhost" user="niku" password="clarity">
    <ftp:put localDir="/home/niku/xog/bin" fileName="gel.bat" remoteDir="/tmp"/>
</ftp:open>
```

Integrations: Overview

- Integration Triggers
  - Manual (User initiated)
  - Scheduled
  - Event Based

- Integration Methods
  - Flat Files
  - Web Services
  - Database Links
  - Third Party Tools

Integrations: Methods

- Flat File Based Integration
  - Based on extract from source system
  - Not a real time integration but a scheduled job
  - Needs coordination between both application teams
  - Complexity of the development is medium
- Web Service Based Integration
  - Applications should be able to access web services
  - Can provide near real time integration
  - Both application teams needs involvement in case of bi-directional integration
  - Complexity of the development is high

Integrations: Methods Cont'd

- Database Links Based Integrations
  - Applications should provide direct database connections
  - High risk, high performance
  - Less complex to develop
- Third Party Tools
  - Integration can be a real time or a scheduled job
  - More dependency on the third party tools
  - Less complex to implement
  - Can be difficult to troubleshoot issues

Integrations: Details

- Staging areas should often be utilized to load and validate the data prior to pushing the data into the final destination inside CA PPM
  - This helps reduce errors during the data load and can allow user interaction to correct the values prior to processing
- A combination of GEL tag libraries, Java, and SQL can be utilized in integration code
  - A GEL script inside a process
  - A SQL stored procedure inside a job definition
  - A Java executable inside a job definition
  - Note that stored procedures and Java executables are unsupported in on-demand environments for security and stability reasons

Integrations: Details Cont'd

- A typical flat file approach for exporting data from a GEL script:
  - A query is ran to gather data to be sent to the other application
  - The data is formatted into a file using the writeFile tag
  - The file is uploaded to a FTP server using the FTPTagLibrary open and put tags
  - The process containing the GEL script is scheduled to run at an agreed time
- The same process, instead of being scheduled, can be invoked by the external application.

Questions?

regoUniversity 2017

Let Rego be your guide.

Thank You For Attending regoUniversity

Instructions for PMI credits

- Access your account at pmi.org
- Click on **Certification**
- Click on **Maintain My Certification**
- Scroll down to **Report PDU's**
- Click on Course Training (or other appropriate category)
- Enter **Rego Consulting**
- Enter Activity- **Enter Name of Course**
- Enter **Description**
- Enter **Date Started**
- Enter **Date Completed**
- Provide Contact Person **Name of Person to Contact**
- Provide Contact E-Mail **E-Mail of Person to Contact**
- Enter Number of **PDU's Claimed** (1 PDU per course hour)
- Click on the **I agree this claim is accurate box**
- Click **Submit** button

Let us know how we can improve!
Don't forget to fill out the class survey.

Phone
888.813.0444

Email
info@regouniversity.com

Website
www.regouniversity.com