Everything you always wanted to know about OpenEdge/Java integration but were afraid to ask

Frank Hilhorst
President Progressive Consulting Inc.
What is my goal today?

- Have you walk out of here with new ideas about what you can do with a business application
- Show you how you can implement these ideas
- Give you a set of examples and tools
How am I going to go about achieving this goal

- Making the case
- Why should you care
- Current state of integration OERA/JAVA
- Walk through sample OpenClient project
  - Showing some strategies for OpenEdge/JAVA integration
- Go over the tools and sample implementations in the toolkit
Making the case why you should care
A look at our solar system in the software universe

Planet Microsoft

Planet OpenEdge

Planet JAVA
I have a dream ...

that one day the OpenEdge business application will sit together with the JAVA applications at the table of Open Source brotherhood and the JAVA applications will endow them with all kinds of cool features.
Demo

- OpenEdge Call Center application
- Integrates with Google Voice through a JAVA Servlet
- Google Voice calls the customer from the Agent’s phone
- Agent screen is automatically populated with customer data
Architecture of Call Center Application

- Servlet
- WebClient App
- Amazon Cloud Computer
- Google Voice Server
- Cloud
- Phone Network
- Agent
- Customer
### The cost of the Call Center application

<table>
<thead>
<tr>
<th>Component</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jetty servlet engine</td>
<td>Free</td>
</tr>
<tr>
<td>ActiveMQ JMS server</td>
<td>Free</td>
</tr>
<tr>
<td>OpenEdge Call Center app</td>
<td>Free</td>
</tr>
<tr>
<td>4GL STOMP Adapter</td>
<td>Free</td>
</tr>
</tbody>
</table>
What is in the Toolkit

- Jetty Servlet engine
- ActiveMQ JMS Server
- Servlet with Google Voice API
- 4GL Agent Screen based upon the Sports database
- 4GL STOMP JMS Adapter
Current state of integration OERA/JAVA
Current implementations of JAVA in the OERA

- Web Services Adapter (Wsa Servlet)
- Internet Adapter (Aia servlet)
- SonicMq JMS server
  - Sonic 4GL Adapter
- Savvion
- OpenEdge architect (Eclipse)
Reasons why Planet OpenEdge and Planet Java are converging

- ABL OO
- Eclipse (OpenEdge architect)
- Mobile applications
### Java terminologies and their Progress equivalents

<table>
<thead>
<tr>
<th>Java Terminology</th>
<th>Progress Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE (Java 2 Enterprise Edition)</td>
<td>OERA (OpenEdge Reference Architecture)</td>
</tr>
<tr>
<td>Servlet Engine (Tomcat)</td>
<td>WebSpeed</td>
</tr>
<tr>
<td>Servlet</td>
<td>Cgi Wrapper</td>
</tr>
<tr>
<td>• <code>web.xml</code></td>
<td>• <code>progress.ini</code></td>
</tr>
<tr>
<td>JSP (Java Server Page)</td>
<td>Speedscript</td>
</tr>
<tr>
<td>Java Bean</td>
<td>Appserver procedure</td>
</tr>
<tr>
<td>Jar library</td>
<td>Shared procedure library</td>
</tr>
</tbody>
</table>
Options for integrating JAVA functionality into an OpenEdge application

- Servlet
  - With OpenClient connection to Progress
- Sockets
- JMS (Soniq, ActiveMq)
- JNI
  - Wrap JVM in a .dll or ActiveX control
Approaching JAVA from a PROGRESS perspective

• What you will look for ....
  • The temp-table
• What you will find .....  
  • Hash tables, vectors, arrays, collections etc. 
• What you will be unfamiliar with ....
  • multithreading
Walk through sample Project

Servlet with JAVA OpenClient integration
What you need to know about Servlets

- Development starts with an object that extend the `httpServlet` class
- End product of a JAVA servlet project is a `.war` file
  - `.war` file gets deployed in the `webapps` directory of the servlet engine
- Each servlet comes with a `web.xml` configuration file
- Acts like the `progress.ini` for that servlet
Steps involved in a JAVA OpenClient project

- Create Appserver procedure you want to call from JAVA
- Create a Java Proxy for the appserver procedure with ProxyGen
- Create .jar file for JAVA classes
- Create JAVA project
- Import proxy and OpenClient .jars
- Code JAVA Project
Diagram of sample project

HTTP Request

Servlet

Progress Dispatch Procedure

Appserver Procedure (Cgi Wrapper)

OpenClient

HTML or XML
What the dispatch procedure looks like

```plaintext
DEFINE INPUT PARAMETER icOperation AS CHARACTER NO-UNDO.
DEFINE INPUT PARAMETER iliInputParameters AS LONGCHAR NO-UNDO.
DEFINE OUTPUT PARAMETER ocContentType AS CHARACTER NO-UNDO.
DEFINE OUTPUT PARAMETER cloOutput AS LONGCHAR NO-UNDO.

RUN VALUE(icOperation) {
    INPUT iliInputParameters.
    OUTPUT ocContentType.
    OUTPUT cloOutput) NO-ERROR.

    IF ERROR-STATUS.ERROR THEN
        DO:
            ASSIGN ocContentType = "text/html"
            cloOutput = "<html><head>" + 
            "<title>ProCog Dispatch Error</title>" + 
            "</head><body>" + 
            SUBST("$1":$2") + 
            ERROR-STATUS.GET-MESSAGE(1). + 
            ERROR-STATUS.GET-MESSAGE(1). + 
            "</body></html>"
        END.
    END.
```
What the Cgi Wrapper include file looks like

DEFINE INPUT PARAMETER ilcInputParameters AS LONGCHAR NO-UNDO.
DEFINE INPUT PARAMETER ilcFileUpload AS LONGCHAR NO-UNDO.
DEFINE OUTPUT PARAMETER ocContentType AS CHARACTER. NO-UNDO.
DEFINE OUTPUT PARAMETER oloOutput AS LONGCHAR NO-UNDO.

&GLOBAL-DEFINE OUT oloOutput = oloOutput +
&GLOBAL-DEFINE OUT-LONG oloOutput = oloOutput +

FUNCTION get-value RETURNS CHARACTER
( /* parameter-definitions */
   icParName AS CHARACTER ) FORWARD.

FUNCTION output-content-type RETURNS LOGICAL
( /* parameter-definitions */
   icContentType AS CHAR ) FORWARD.
Creating a JAVA Proxy for the dispatch procedure
Creating a JAVA Proxy for the dispatch procedure (continued)
The JAVA files created by ProxyGen
Create a .jar library for the JAVA proxy classes

- `jar cvf ProxyProCgi.jar ./ProxyProCgi/*.*.class`
Import the Proxy jar and the JAVA OpenClient jar’s into JAVA project
3 Step process of Executing a Proxy method

- Create Appserver Connection object
- Instantiate proxy object passing connection object as input parameter
- Execute method in proxy object
Java code for Executing a Proxy method

```java
public String ProcCgiDispatch(HttpServlet servlet,
        String cOperation,
        String cQueryParameters,
        StringBuffer ocContentType,
        StringHolder ocOutput)
    throws OpenXMLException, IOException{
    String cAppserverUrl = servlet.getInitParameter("AppserverUrl");
    // Step 1 create Appserver Connection OBJECT
    Connection appServerConnection = new Connection(cAppserverUrl, "");
    // Step 2 Instantiate proxy object passing connection object as INPUT
    ProCgi ghAppserver = new ProCgi(appServerConnection);
    // Step 3 Execute method in proxy object
    ghAppserver.ProCgiDispatch(cOperation,
            cQueryParameters,
            ocContentType,
            ocOutput);
}
```
The next Strategy for Integration

- Object to Object mapping
  - Create 4GL objects with same attributes and methods as Java objects
  - Serialize 4GL objects to XML
    - Serialize XML back to Java Objects
So what if we could create Graphs directly from a CGI wrapper?
So let’s stick our hand in the cooky jar
Basic steps for creating a Graph with JFreeChart

• Create dataset object
• Populate dataset object with graph data
• Create Graph object passing dataset as input parameter
• Stream Graph as .png file to the browser
What the Java Code looks like

```java
protected void processRequest(HttpServletRequest request,
                          HttpServletResponse response)
{
    OutputStream out = response.getOutputStream();
    /* ... Step 1 Create dataset OBJECT. */
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    /* ... Step 2 Populate dataset with data. */
    dataset.addValue(10.0, "S1", "CA1");
    dataset.addValue(40.0, "S1", "CA2");
    /* ... Step 3 Create graph object with dataset as input. */
    JFreeChart chart = ChartFactory.createBarChart(
        "Bar Chart",
        "Category",
        "Value",
        dataset,
        PlotOrientation.VERTICAL,
        true, true, false);
    response.setContentType("image/png");
    /* ... Step 4 Stream Graph as a PNG file to the browser. */
    ChartUtilities.writeChartAsPNG(out, chart, 400, 300);
}
```
4GL Cgi Wrapper code for a simple Pie Chart

```plaintext
output-content-type ("text/xml")
pieDataSet = NEW TurboGraph.DefaultPieDataset()
viCountryCnt = 0.
FOR EACH Customer BREAK BY Customer.Country:
    viCountryCnt = viCountryCnt + 1.
    IF LAST-OF(Customer.Country) THEN
        DO:
            pieDataSet.setValue(Customer.Country + SUBST(" \&1",viCountryCnt). DEC(viCountryCnt)).
        END.
    END.
pieChart = NEW TurboGraph.PieChart(pieDataSet).
pieChart.setWidth(800).
pieChart.setHeight(600).
pieChart.setDisplayIn3D(FALSE).
pieChart.setTitle("Customers by Country").
lcSerializedXml = pieChart.Serialize()
(\&OUT) lcSerializedXml.
```
What the serialized XML looks like
So let's Demo
What is in the Toolkit

- Jetty Servlet engine deployed with
  - Call center servlet
  - ProCgi servlet
  - ProGraph servlet
- Sample CGI Wrappers
  - For Pie Chart and Bar Chart
- Call Center 4GL application
What is in the Toolkit (Continued)

- ActiveMQ JMS server
  - OpenSource JMS server
- 4GL Stomp adapter
  - Connects 4GL client with ActiveMq
- Eclipse for J2EE (Optional)
  - Comes ProCgi project configured and loaded
- 4GL JNI example
A look into the kitchen at Progressive Consulting

- Hot off the stove
  - TurboCgi
    - WebSpeed alternative
  - TurboComet
    - Comet Server with OpenClient integration
A look into the kitchen at Progressive Consulting (continued)

- On the stove
  - TurboGraph
    - Allows you to create Graphs directly from a Cgi Wrapper
  - TurboVoice
    - Allows you to integrate VOIP into Progres application
  - TurboLocation
    - Building location aware business applications
How to install the toolkit

- Unzip JavaIntegration.zip to C: drive
- Should create directory structure C: \JavaIntegration
- You need to have JAVA installed
- JAVA_HOME must be defined
How to install the toolkit (continued)

• Define appserver
  • Connects to database
    • C:/JavaIntegration/Sports2000.db
  • Propath should include
    • C:\JavaIntegration\GoogleVoice
    • C:\JavaIntegration\ProCgi\CgiWrappers
How to install the toolkit (continued)

- In C:\JavaIntegration\GoogleVoice\Ini\progress.ini
  - Point DLC variable to your Progress install directory
  - Point the following parameter to your appserver

    [AspectSimulator]
    PollingUrl=-DirectConnect -H 127.0.0.1 -S 3390
How to install the toolkit (continued)

- In C:\JavaIntegration\Shortcuts start the following shortcuts in the following order
  - Start ActiveMQ
  - Start Jetty Servlet Engine
  - In the browser type the following url
    - http://localhost:8080/AspectSvr/Aspect
  - Google Voice Demo
  - Eclipse JAVA J2EE
I would like to hear from you

- If you have a problem installing the toolkit
- If some toolkit component(s) do not work for you
- If you could use any of the tools but something is missing
- If you have ideas about features you want to add to your business application that may be available in JAVA
  - Let’s make business applications exciting
  - *Drop me a line*
Questions?