JavaAgent/Backing Bean

Execute Converted Chart from Button

JFreeChart

System Requirements:
Download Domino Designer 8.5.2 Environment (DDE)
http://www.ibm.com/developerworks/downloads/ls/dominodesigner/

Download JFreeChart
http://www.jfree.org/

Download jcommon jar file
http://www.jfree.org/jcommon/

Introduction:
Convert a Lotus Notes Domino Database JavaAgent chart code to be used by an Xpages form via a Backing Bean. Remove Static data and grab live data from a view to fill the chart. Lastly, run the chart using button on form.

Disclaimer:
Information contained in the following is presented as is. This tutorial assumes you have basic programming knowledge. All tutorials are based on an Eclipse/Eclipse-based software. Should you need to familiarize yourself with a certain Eclipse environment, prior to continuing this tutorial, please stop now and see our Tutorials page...

Convert JavaAgent to Backing Bean
At this point we assume Domino Designer 8.5.2, JFreeChart, and JCommon are downloaded/installed and a new JavaAgent is already created. Jump ahead to Backing Bean creation.

JavaAgent.java

[CODE]

```
public void NotesMain()
{
	try
	{
		Session session = getSession();
		AgentContext agentContext = session.getAgentContext();

		HashMap map = new HashMap();
		map.put("Pierre", new Integer(178));
		map.put("Dick", new Integer(87));
```

```
            map.put("Ola", new Double(200));
            map.put("Random", new Double( (Math.random() * 200)));

            writeChartToDisk("Diagram", "c:\\test.jpg", map);

        } catch (Exception e)
        {
            e.printStackTrace();
        }
    }
```

Steps to converting:

(1) Replace the AgentContext code and add FacesContext code

- from this

```
Session session = getSession();
      AgentContext agentContext = session.getAgentContext();
```

- to this

```
Database database = (Database) FacesContext.getCurrentInstance()
            .getApplication().getVariableResolver().resolveVariable(
                        FacesContext.getCurrentInstance(), "database");
```

(2) Add Params to HashMap

- from this

```
HashMap map = new HashMap();
```

- to this

```
HashMap<String, Number> map = new HashMap<String, Number>();
```

(3) Lastly, remove 'extends AgentBase' in the class definition, and comment out 'e.printStackTrace(getAgentOutput());' at the end of the file

Steps to Loading data:

(1) Call a view, example 'By Category', grab Office Departments
(2) Build Document collections for 3 Departments
(3) Replace Static data with Departments data

TIP: Alternatively, one can build an array of values and feed that to the chart. This is merely a sample to show exiting code modified with little effort.

Backing Bean it!

Your JavaAgent works and we assume it compiles okay... You now need to add code under a button to fire up the chart via Xpages. You are going to convert this JavaAgent to a class that can be accessed

through your Java Source folder; you will then create an Xpage to load the chart. Copy and paste below to your environment, areas of interest have been highlighted for your convenience.

JavaAgentChartBean.java

```
package com.dokoll.solutions.inc.runCharts;
/**
 * Created from copy: 2011.06.07.3.48.PM
 * Static data for Charts, trial code to convert and add view data
 */
//load imports for Java IO, Util classes
import java.io.FileOutputStream;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
//load imports for Faces classes
import javax.faces.context.FacesContext;
//load imports for Domino classes
import lotus.domino.AgentBase;
import lotus.domino.Document;
import lotus.domino.DocumentCollection;
import lotus.domino.View;
import lotus.domino.local.Database;
//load imports for JFreeChart classes
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartUtilities;
import org.jfree.chart.JFreeChart;
import org.jfree.data.general.DefaultPieDataset;
/**
 * @author Dököll Solutions, Inc.
 * @version 2011.06.07.3.48.PM
 *
 */
public class JavaAgentChartBean
{
            //Run the Program
            public void NotesMain() {

            try
            {
                  //get the current database being used
                  Database database= (Database) FacesContext.getCurrentInstance()
                  .getApplication().getVariableResolver()
                  .resolveVariable(FacesContext.getCurrentInstance(), "database");
                  System.out.println("Database Obtained..." + database);
                  //Call database view
                  View view = database.getView("By Category");

                  //Build a collection for each Departement slice in chart
                  //2012.03.26.10.PM
                  DocumentCollection reColl = (DocumentCollection)
database.getAllDocuments();
                  //Declare/initialize DepartmentOne String
                  String DepartMentOneName = "Research Team";
                  //Search the String
```

```
                    reColl.FTSearch(DepartMentOneName);
                    //get number a count of times DepartMentOneName loaded
                    int Doc1Counts = reColl.getCount();
                    //Build a collection for each Departement slice in chart
                    //2012.03.26.10.PM
                    DocumentCollection devColl = (DocumentCollection)
database.getAllDocuments();
                    //Declare/initialize DepartmentTwo String
                    String DepartMentTwoName = "Development Team";
                    //Search the String
                    devColl.FTSearch(DepartMentTwoName);
                    //get number a count of times DepartMentTwoName loaded
                    int Doc2Counts = devColl.getCount();
                    //Build a collection for each Departement slice in chart
                    //2012.03.26.10.PM
                    DocumentCollection depColl = (DocumentCollection)
database.getAllDocuments();
                    //Declare/initialize DepartmentThree String
                    String DepartMentThreeName = "Deployment and Testing";
                    //Search the String
                    depColl.FTSearch(DepartMentThreeName);
                    //get number a count of times DepartMentThreeName loaded
                    int Doc3Counts = depColl.getCount();
                    // turn off auto-update so that if we make a change to a document
                    // and re-saves, it won't affect the sort order in the view
                    view.setAutoUpdate(false);
                    // gets the first document in the view
                    Document doc = view.getFirstDocument();
                    //TO DO:  Loop through once collection and feed to Map
                    //Add a statement to ensure this specific collection is not null
                    //or to ensure data is available to fill specific slice
                    HashMap<String, Number> map = new HashMap<String, Number>();
                    map.put(DepartMentOneName, new Integer(Doc1Counts));
                    map.put(DepartMentTwoName, new Integer(Doc2Counts));
                    map.put(DepartMentThreeName, new Double(Doc3Counts));
                    Document temp = null;
                    //begin looping through document list(s)
                    while (doc != null) {
                    //TO DO: Get UserInfo into file Name, add a date for uniqueness
                          //format date in this fashion YYYY.MM.DD.HH.MM.SS.
                          //Append AM or PM
                          String UserInfo = "411User";
                          //2011.06.14.2.05.PM
                          writeChartToDisk("Diagram",
"C:\\temp\\XML_DATA\\"+UserInfo+"pietest.jpg", map);
                          // do something with the document here...
                          // whatever, just don't delete it (yet)!
                          temp = view.getNextDocument(doc); // get the next one
                          doc.recycle(); // recycle the one we're done with
                          doc = temp;
                    }

              } catch (Exception e)
              {
```

```
                e.printStackTrace();
            }
      }
      private void writeChartToDisk(String title, String fileName, Map<String,
Number> map)
      {
            //put map values in to a DefaultPieDataset
            Iterator<String> iterator = map.keySet().iterator();
            DefaultPieDataset pieDataset = new DefaultPieDataset();
            while (iterator.hasNext())
            {
                  Object o = iterator.next();
                  Object o2 = map.get(o);
                  pieDataset.setValue((String) o,  (Number) o2);
            }
            //Create the actual chart
            JFreeChart chart = ChartFactory.createPieChart(title, pieDataset, true,
true, true);
            //Write chart to disk as JPG file, and ask explorer.exe to show it.
            //You could extract the file to the html directory of the domino server
or attach to a notes-document
            try
            {
                  FileOutputStream fos = new FileOutputStream( fileName);
                  ChartUtilities.writeChartAsJPEG(fos, 1, chart, 750, 400);
                  fos.flush();
                  fos.close();

                  Runtime run = Runtime.getRuntime();
                  run.exec("explorer.exe  " + fileName );

            } catch (Exception e)
            {

            }
      }

}
```

Build Xpages file

Full code added below, area of interest have been highlighted for your convenience.

xploadcharts.xsp

```
<?xml version="1.0" encoding="UTF-8"?>
<xp:view xmlns:xp="http://www.ibm.com/xsp/core">
      <xp:table>
            <xp:tr>
                  <xp:td>
                        <xp:viewTitle xp:key="viewTitle" id="viewTitle1"
value="Launch Charts"
                              style="background-color:rgb(255,255,255);font-
size:14pt;width:138.0px" />
                  </xp:td>
            </xp:tr>
```

```
            <xp:tr>
                  <xp:td>
                        <xp:button value="Fetch" id="button1"
                              action="#{JavaAgentChartBean.NotesMain}"
type="submit" save="true" />
                  </xp:td>
            </xp:tr>
      </xp:table>
</xp:view>
```

Conclusion:
You can now have users click on an button from the Xpage to load specific charts or simply run them as a JavaAgent. If using as a JavaAgent and still loading from Xpages, you should use beforePageLoad in your code and have Java code run the Agent

Added info: You will need to reference `JavaAgentChartBean.java` in your faces-config.xml file.

Resources:

Original code found here:
http://www.dominoexperts.com/articles/Visualize-your-domino-data-using-Open-Source-java-

Similar & additional here:
http://www.java2s.com/Code/Java/Chart/JFreeChartPieChartDemo1.htm

Questions, comments, please post a brief message. Thank you for coming...