Process Tool Plugin Example - Simple Workflow Variable Assignment

Let's say we want to change workflow variables directly inside a system tool activity without having to create a complex beanshell script. The **Process Tool Plugin** assigns a simple expression to a workflow variable. The expression is evaluated using a beanshell interpreter, and the resulting value is converted into a string. The workflow variable is then assigned to that string value.

All workflow variables are directly accessible inside the expression, but note that all Joget Workflow variables are of the string type.

Usage Scenario:

1. Set a workflow variable to a constant value. Use **AssignVariablePlugin** as a tool. Use the parameters *workflow variable name : A, expression : 0*.
2. To increment a workflow variable value, use **AssignVariablePlugin** as a tool. Use the parameters *workflow variable name : A, expression : Integer.parseInt(A) + 1.*
3. To decrement a workflow variable value : use **AssignVariablePlugin** as a tool. Use the parameters *workflow variable name : A, expression : Integer.parseInt(A) - 1*.

Steps in Creating a Plugin:

Create an empty project folder. Inside this folder, create subdirectory src\main\java\myplugin.

Create the following text files :

src\main\java\myplugin\AssignVariablePlugin.java

```
package myplugin;

import org.joget.plugin.base.ApplicationPlugin;
import org.joget.plugin.base.DefaultPlugin;
import org.joget.plugin.base.PluginManager;
import org.joget.plugin.base.PluginProperty;
import org.joget.workflow.model.WorkflowAssignment;
import org.joget.workflow.model.WorkflowVariable;
import org.joget.workflow.model.service.WorkflowManager;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.List;
import java.util.Map;

public class AssignVariablePlugin extends DefaultPlugin implements  ApplicationPlugin
{

    public String getName() {
        return "AssignVariablePlugin";
    }

    public String getDescription() {
        return "Store Simple Expressions into a Workflow Variable ";
    }
    @Override
    public void stop(org.osgi.framework.BundleContext context) {
        super.stop(context);
    };
    public String getVersion() {
        return "1.0.0";
    }
```

```
    public PluginProperty[] getPluginProperties() {
        PluginProperty[] properties = new PluginProperty[]{
            new PluginProperty("variableName", "Workflow Variable Name",
PluginProperty.TYPE_TEXTFIELD, null, ""),
            new PluginProperty("expr", "Expression (note: all joget workflow variables
are of type String)", PluginProperty.TYPE_TEXTFIELD, null, "1"),
        };
        return properties;
    }
    String currentUrl;

    public Object execute(Map properties) {
        Object result = null;
        try {
            PluginManager pluginManager = (PluginManager)
properties.get("pluginManager");
            WorkflowAssignment wfAssignment = (WorkflowAssignment)
properties.get("workflowAssignment");
            WorkflowManager workflowManager = (WorkflowManager)
pluginManager.getBean("workflowManager");
            String newValue = "";
            String script = (String)properties.get("expr");
            List<WorkflowVariable> vars = wfAssignment.getProcessVariableList();
            Object scriptResult = executeScript(script,vars);
            if (scriptResult!=null) newValue = scriptResult.toString();
            String varName;
            varName = (String)properties.get("variableName");
            workflowManager.activityVariable(wfAssignment.getActivityId(), varName,
newValue);
            Logger.getLogger(getClass().getName()).log(Level.WARNING, "Setting
variable "+ varName + " to "+ newValue);
            return null;
        } catch (Exception e) {
            Logger.getLogger(getClass().getName()).log(Level.WARNING, "Error executing
report plugin", e);
            return null;
        }
    }
    protected Object executeScript(String script, List<WorkflowVariable> props) {
        Object result = null;
        try {
            bsh.Interpreter interpreter = new bsh.Interpreter();
            interpreter.setClassLoader(getClass().getClassLoader());
            for (WorkflowVariable v : props) {
                if (v.getName() != null)
                {
                    interpreter.set(v.getName(), v.getVal());
                } else if (v.getId()!=null)
                {
                    interpreter.set(v.getId(), v.getVal());
                }
            }
            Logger.getLogger(getClass().getName()).log(Level.FINE, "Executing script "
+ script);
            result = interpreter.eval(script);
            Logger.getLogger(getClass().getName()).log(Level.WARNING, "result is " +
(result==null?"NULL": result.getClass().toString()));
            return result;
        } catch (Exception e) {
```

```
            Logger.getLogger(getClass().getName()).log(Level.WARNING, "Error executing
script", e);
            return null;
        }
    }
```

```
}
```

pom.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<project  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd"
xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <modelVersion>4.0.0</modelVersion>
  <groupId>myplugin</groupId>
  <artifactId>AssignVariablePlugin</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>bundle</packaging>
  <name>AssignVariablePlugin</name>
  <url>http://inventorsparadox.blogspot.com</url>
  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>2.0.2</version>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.4.3</version>
        <executions>
          <execution>
            <id>integration-test</id>
            <phase>integration-test</phase>
            <goals>
              <goal>test</goal>
            </goals>
            <configuration>
              <skipTests>false</skipTests>
            </configuration>
          </execution>
        </executions>
        <configuration>
          <skipTests>true</skipTests>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
        <extensions>true</extensions>
        <configuration>
          <instructions>
            <Export-Package>myplugin</Export-Package>
            <Private-Package>myplugin.*</Private-Package>
            <Bundle-Activator>myplugin.AssignVariablePlugin</Bundle-Activator>
```

```
<Import-Package>!*,org.joget.report.dao,org.joget.report.model,org.joget.report.servic
e,org.joget.commons.util,org.joget.plugin.base,org.joget.plugin.property.model,org.jog
et.plugin.property.service,org.joget.directory.model,org.joget.directory.model.service
,org.joget.directory.dao,org.joget.workflow.model,org.joget.workflow.model.dao,org.jog
et.workflow.model.service,org.joget.workflow.util,org.joget.apps.app.dao,org.joget.app
s.app.lib,org.joget.apps.app.model,org.joget.apps.app.service,org.joget.apps.datalist.
lib,org.joget.apps.datalist.model,org.joget.apps.datalist.service,org.joget.apps.form.
lib,org.joget.apps.form.dao,org.joget.apps.form.model,org.joget.apps.form.service,org.
joget.apps.list.service,org.joget.apps.userview.lib,org.joget.apps.userview.model,org.
joget.apps.userview.service,org.joget.apps.workflow.lib,javax.servlet,javax.servlet.ht
tp,org.osgi.framework;version="1.3.0"</Import-Package>
<Embed-Dependency>scope=compile|runtime;inline=false;artifactId=!commons-logging</Embe
d-Dependency>
                        <Embed-Transitive>true</Embed-Transitive>
                        <Embed-Directory>dependency</Embed-Directory>
                        <Embed-StripGroup>true</Embed-StripGroup>
                        <DynamicImport-Package>*</DynamicImport-Package>
</instructions>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <distributionManagement>
    <repository>
      <id>internal</id>
      <url>http://dev.joget.org/archiva/repository/internal</url>
    </repository>
    <snapshotRepository>
      <id>snapshots</id>
      <url>http://dev.joget.org/archiva/repository/snapshots</url>
    </snapshotRepository>
  </distributionManagement>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.4</version>
      <scope>test</scope>
    </dependency>
<dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring</artifactId>
            <version>2.5.4</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>2.5.4</version>
            <scope>test</scope>
        </dependency>     <dependency>
            <groupId>org.beanshell</groupId>
            <artifactId>bsh</artifactId>
            <version>2.0b4</version>
        </dependency>
  <dependency>
            <groupId>org.joget</groupId>
            <artifactId>wflow-core</artifactId>
            <version>3.0-SNAPSHOT</version>
```

```
        <scope>provided</scope>
    </dependency>
```

```
  </dependencies>
</project>
```

Build Step:

```
mvn install
```

Installation :

Login to Joget Workflow as admin, choose System Settings>Manage Plugins>Upload Plugin, then browse to the target directory of the project folder. Upload the project snapshot jar that was generated by the previous maven execution.