OpenKM
Workflow guide

Workflow Guide

JBoss jBPM is a flexible, extensible framework for process languages. jPDL is one process language that is build on top of that common framework. It is an intuitive process language to express business processes graphically in terms of tasks, wait states for asynchronous communication, timers, automated actions,... To bind these operations together, jPDL has the most powerful and extensible control flow mechanism.

jPDL has minimal dependencies and can be used as easy as using a java library. But it can also be used in environments where extreme throughput is crucial by deploying it on a J2EE clustered application server.

jPDL can be configured with any database and it can be deployed on any application server.

Let's see a example execution of the medium workflow at Sample Workflow Execution.

Here you have some links with documentation and tutorials:

- jBPM User Guide [1]
- jBPM Tutorial [2]
- jBPM Simulation Tutorial [3]
- JBoss Tools News and Noteworthy [4]

Also there are some interesting books:

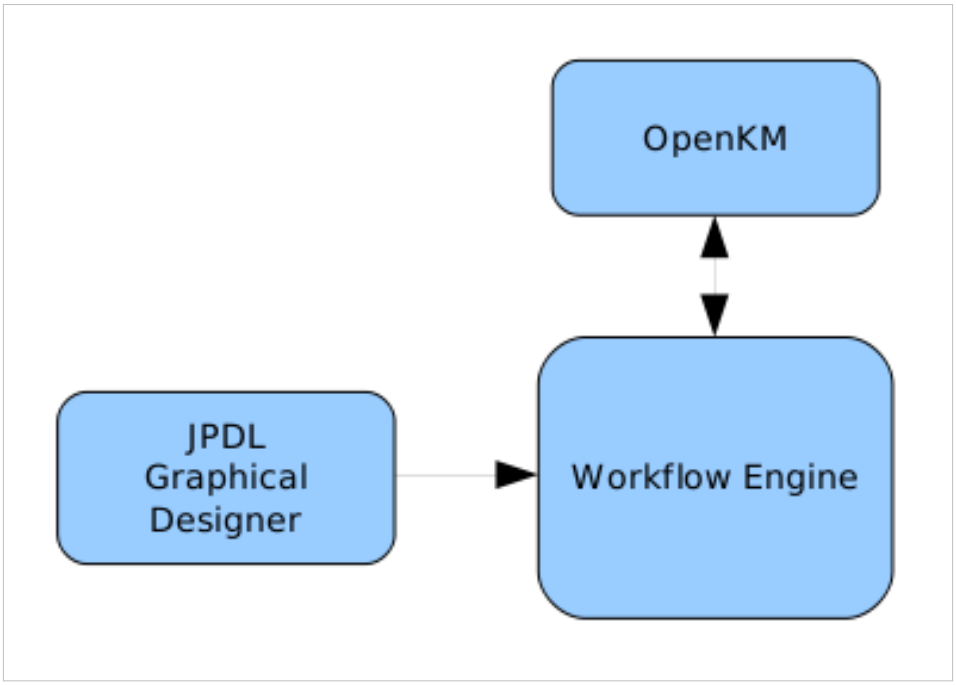
- Business Process Management with JBoss jBPM
- jBPM Developer Guide

References

[1] http://docs.jboss.com/jbpm/v3.2/userguide/html/
[2] http://www.mastertheboss.com/en/jbpm/51-jbpm-tutorial-part.html
[3] http://www.bpm-guide.de/open-source-bpm/jbpm-simulation-tutorial/
[4] http://docs.jboss.org/tools/whatsnew/

Overview

The core workflow and BPM functionality is packaged as a simple java library. This library includes a service to manage and execute processes in the jPDL database.



You can see a working example at jBPM Overview [1].

References

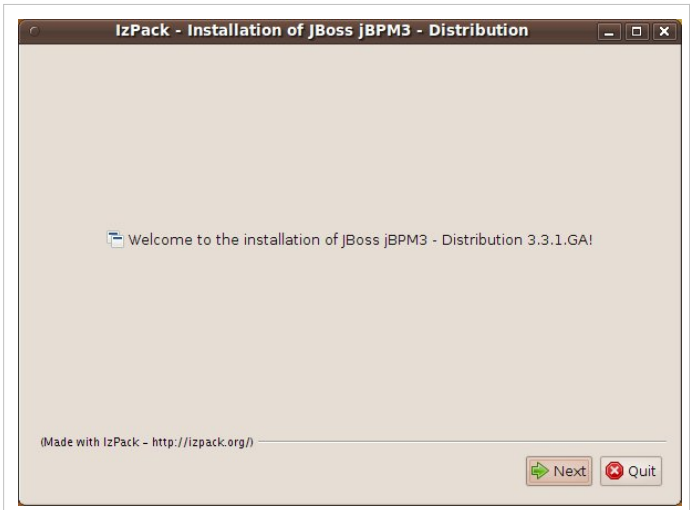
[1] http://docs.jboss.com/jbpm/v3/demos/movies/jbpm-overview.htm

JBPM installation

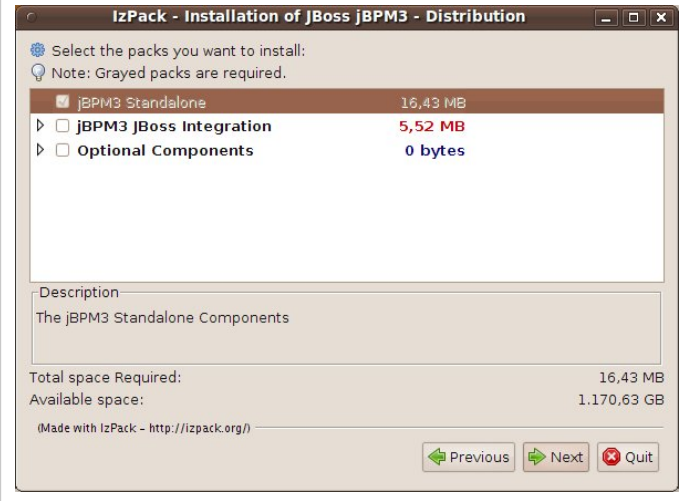
First of all, you need the jBPM libraries installed in your system. There is an installer available at http://sourceforge.net/projects/jbpm/files/. You should download the jBPM-3.3.1.GA version. After that, you can run the install process:

```
$ java -jar jbpm-installer-3.3.1.GA.jar
```

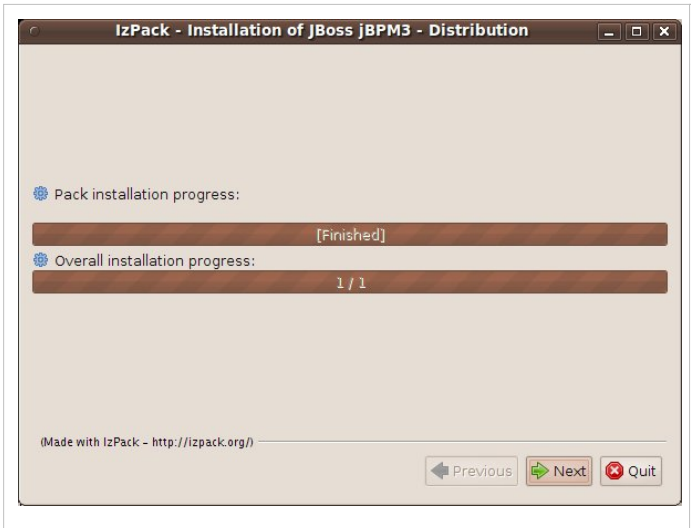
You have to select a destination path where the library will be installed:

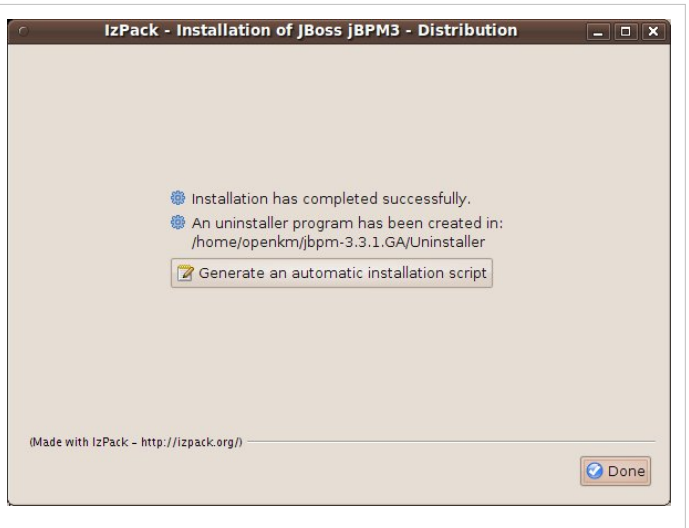


And can only select the minimun requirements to install, because we only need the jBPM library:



When finished, the libraries needed to implement workflows are copied at */home/openkm/jbpm-3.3.1.GA*. Remember this path because you need to tell to Eclipse where these libraries are located.





JBPM configuration

jBPM configuration file is located at $JBOSS_HOME/server/default/deploy/OpenKM.war/WEB-INF/classes/jbpm.cfg.xml. If you edit this file, you will see something like this:

```
<string name="jbpm.mail.smtp.host" value="smtp.your-domain.com " />
<string name="jbpm.mail.from.address" value="noreply@your-domain.com" />
<string name="resource.mail.templates" value="jbpm.mail.templates.xml"/>
<bean name="jbpm.mail.address.resolver" class="com.openkm.workflow.OKMAddressResolver" singleton="true"/>
```

As in JBoss mail configuration, you have to edit a couple of properties:

- **jbpm.mail.smtp.host**: this is the host where is located your mail server. Can be localhost if you have a local mail server installed (like Postfix) Typically the same value of mail.smtp.host.
- **jbpm.mail.from.address**: all the mails send by OpenKM will be from this mail. Can be in the form of noreply@your-domain.com. Typically the same value of mail.from.

jBPM notify you when a task has been assigned to you or when he wants to send you a reminder. These mail templates are defined in the file jbpm.mail.templates.xml which is located at the same place of the previous one. Here you can define a more elegant mail message. Also you have to configure a parameter here:

- **BaseTaskListURL**: a value in the form of http://your-domain.com:8080/OpenKM.

For more info, read http://docs.jboss.com/jbpm/v3.2/userguide/html/mail.html.

Eclipse plugin

jPDL also includes a graphical designer tool. The designer is a graphical tool for authoring business processes. It's an eclipse plugin.

The most important feature of the graphical designer tool is that it includes support for both the business analyst as well as the technical developer. This enables a smooth transition from business process modeling to the practical implementation.

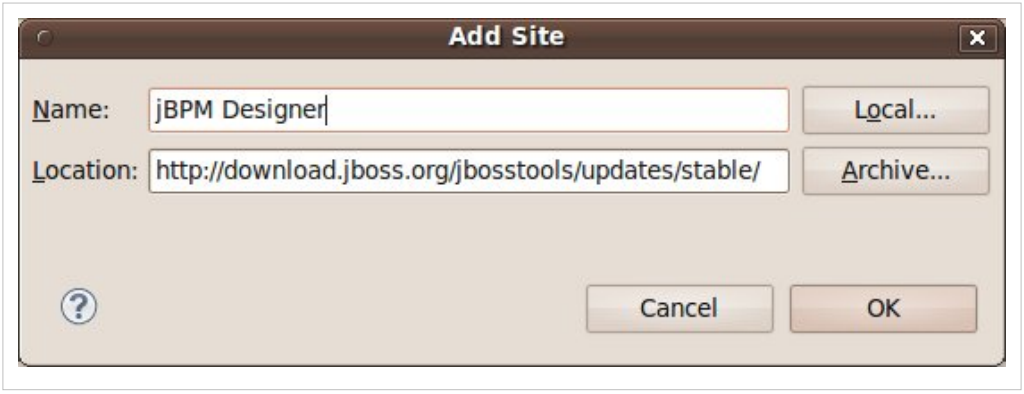
- Eclipse plugin: Installation
- Eclipse plugin: Usage

Eclipse plugin: Installation

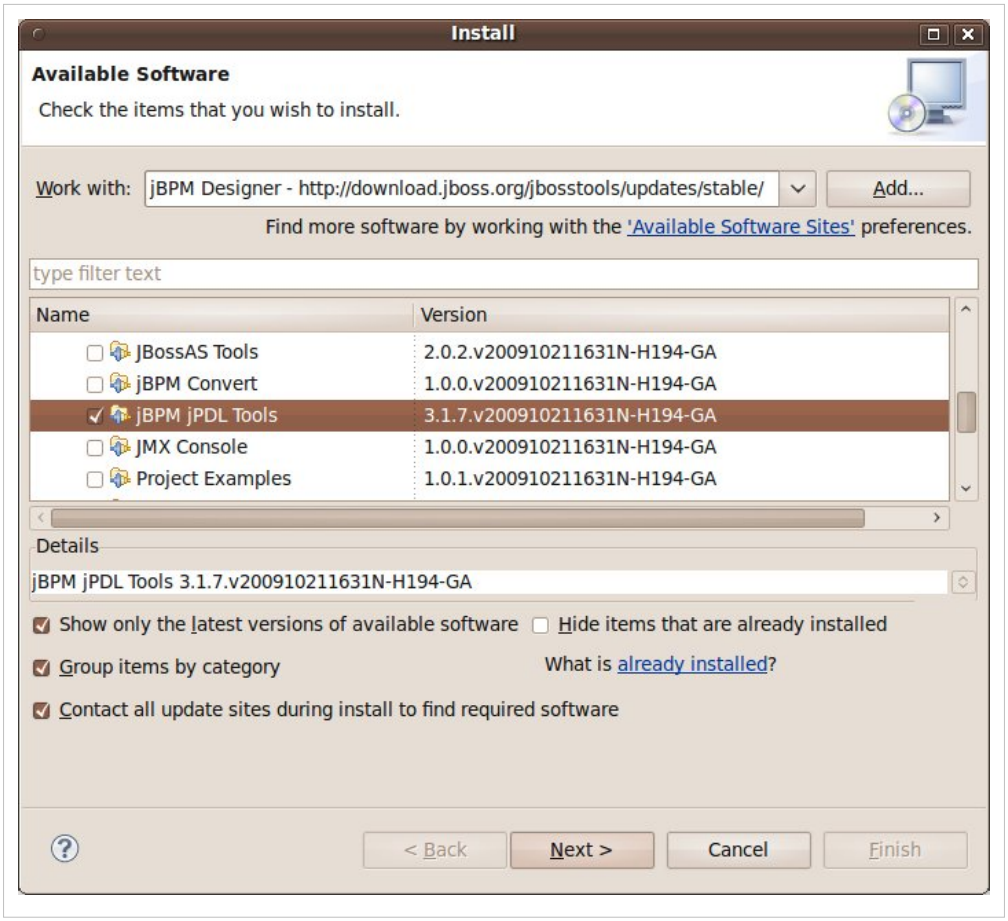
First of all, download Eclipse Galileo from http://www.eclipse.org/galileo/. Get the **Eclipse IDE for Java Developers** version. Once you have installed the application, start it and will install the jBPM designed.

A more recent Eclipse called Helios has been released. You can download it from [1]

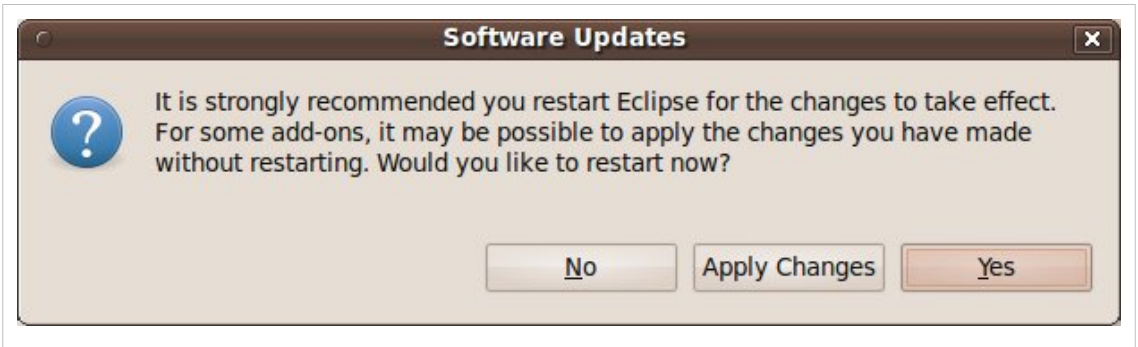
Go to Help → Install New Software... Add a new site with this location http://download.jboss.org/jbosstools/updates/stable/:



Now select **jBPM jPDL Tools** from the list and click on Next button.



Eclipse should be restarted to finalize the plugin installation.



Eclipse Helios & JBoss Tools 3.2

Recently the packages have been refactorized and is located at **All JBoss Tools 3.2.0 > jBPM 3 Tools Runtime**.

Eclipse Indigo & JBoss Tools 3.3

If you use Eclipse Indigo, follow instructions from JBoss Tools 3.3 Installation From Update Site [2]. You have two available software sites:

- http://download.jboss.org/jbosstools/updates/development/indigo/
- http://download.jboss.org/jbosstools/updates/development/indigo/soa-tooling/

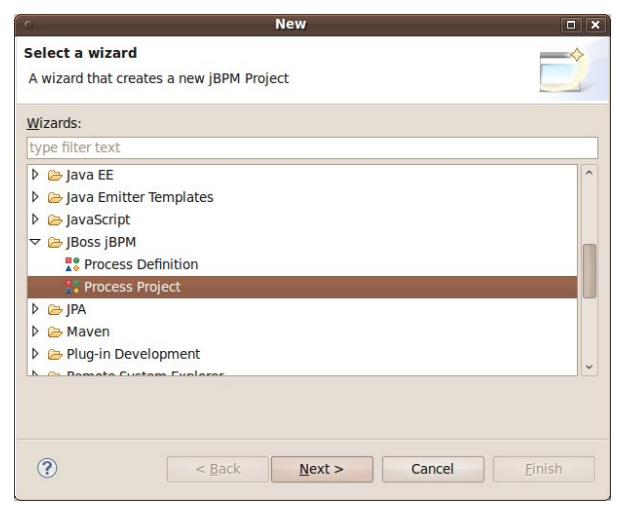
The jBPM Graphical Editor is located in the **SOA Tooling** source. Once registered the site, check the option at **All JBoss Tools - SOA Tooling 3.3 > jBPM 3 Tools Runtime**.

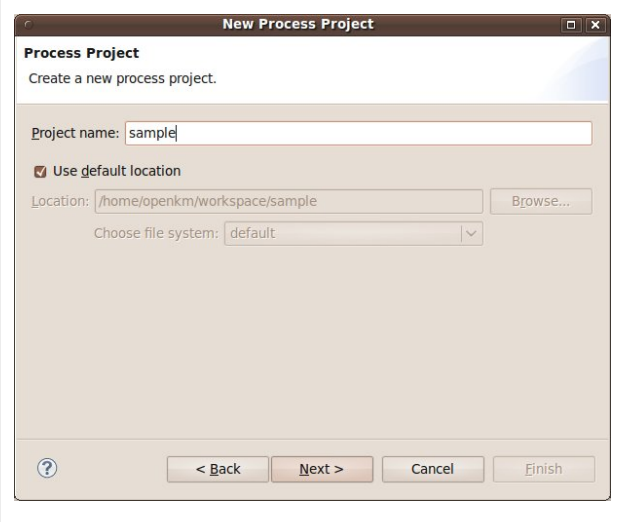
References

[1] http://www.eclipse.org/downloads/.
[2] http://www.jboss.org/tools/download/installation/update_3_3

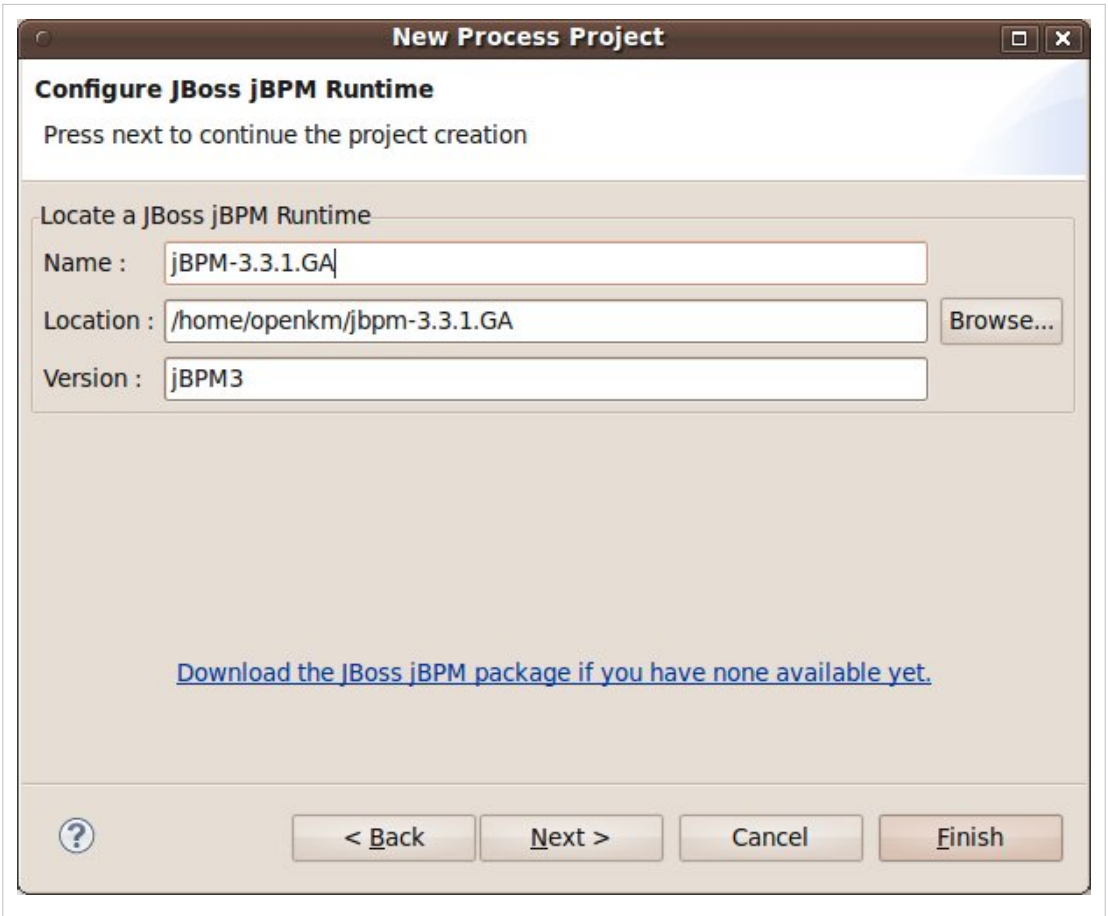
Eclipse plugin: Usage

To create a new process definition, go to File → New → Other and select "Process Definition", but before you need to create a "Process Project". Will appear another form where you need to specify the source folder and the process name. Lets call it simply "hello".

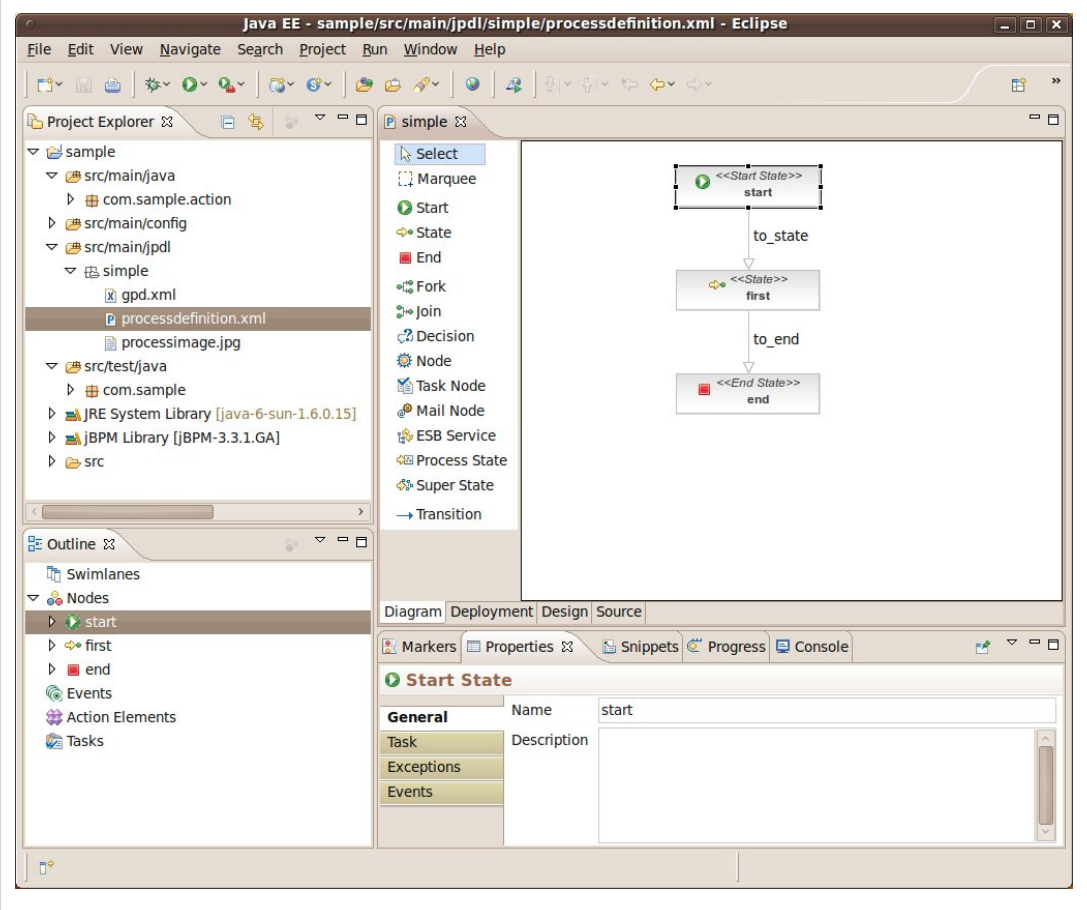




If is the first time you create a Process Definition, you have to configure the jBPM runtime. Remember the path of the jBPM library installation (See JBPM installation)



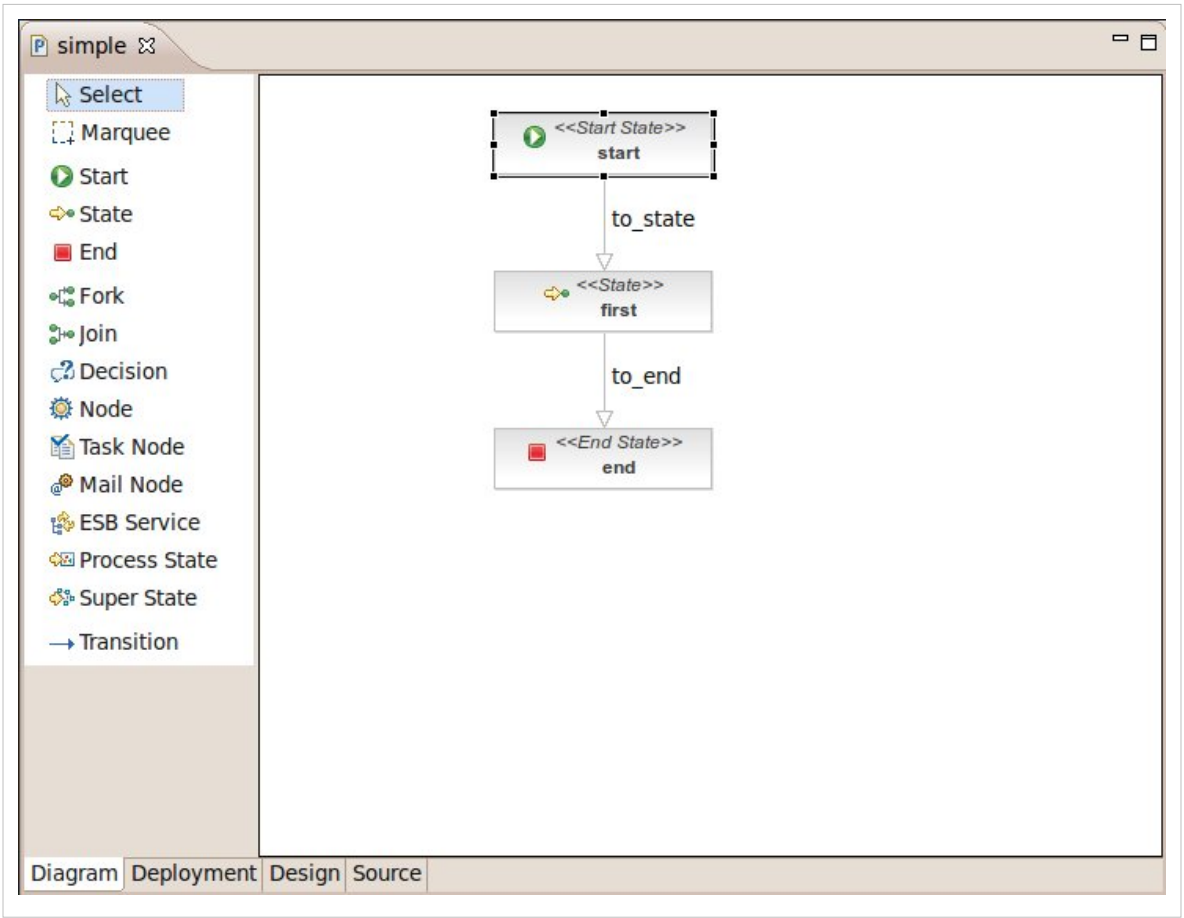
Once the wizard has finished, you can see an Eclipse environment with a central panel divided in four tabs:



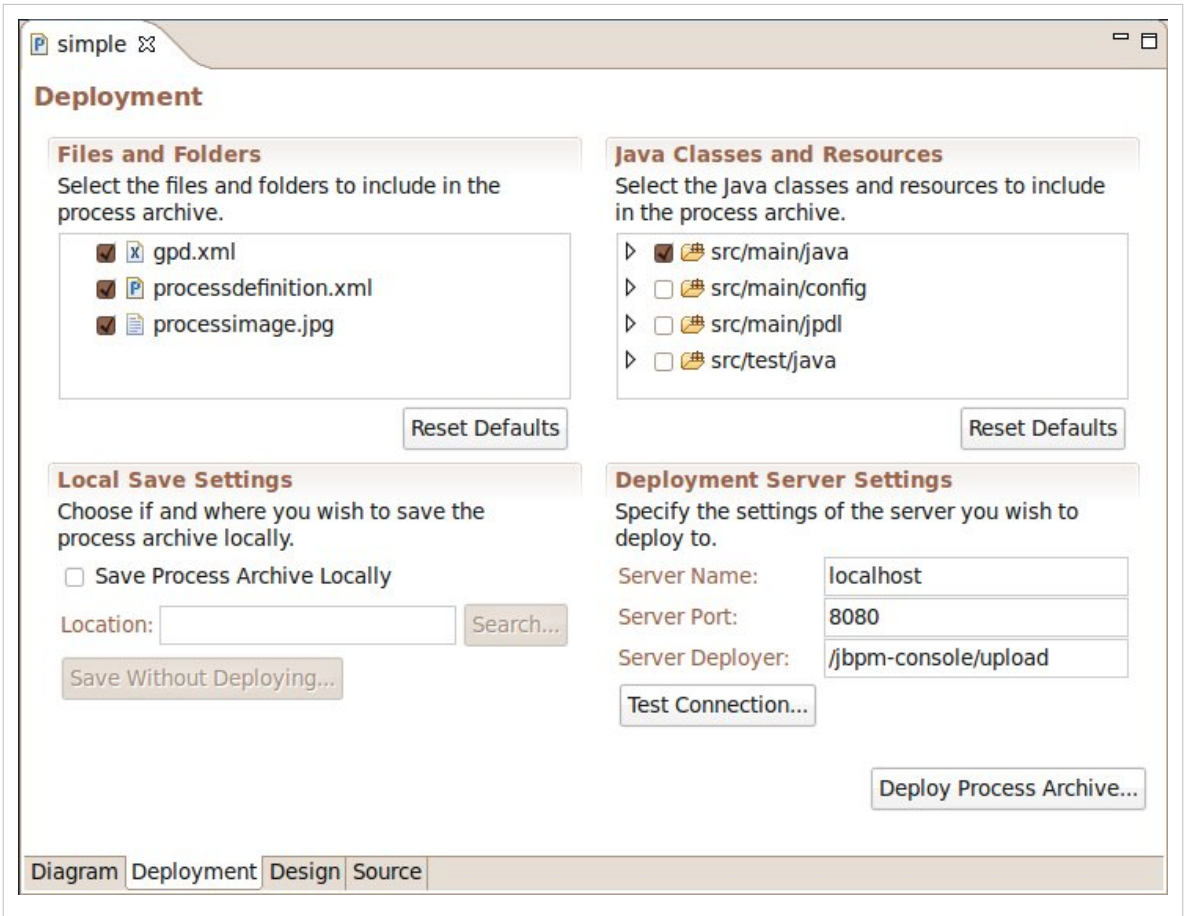
There is a new revision of the jBPM Eclipse plugin with some changes. They are described at:

- jBPM Tools 3.2.0.M2 What's New [1]
- jBPM Tools 3.2.0.M1 What's New [2]

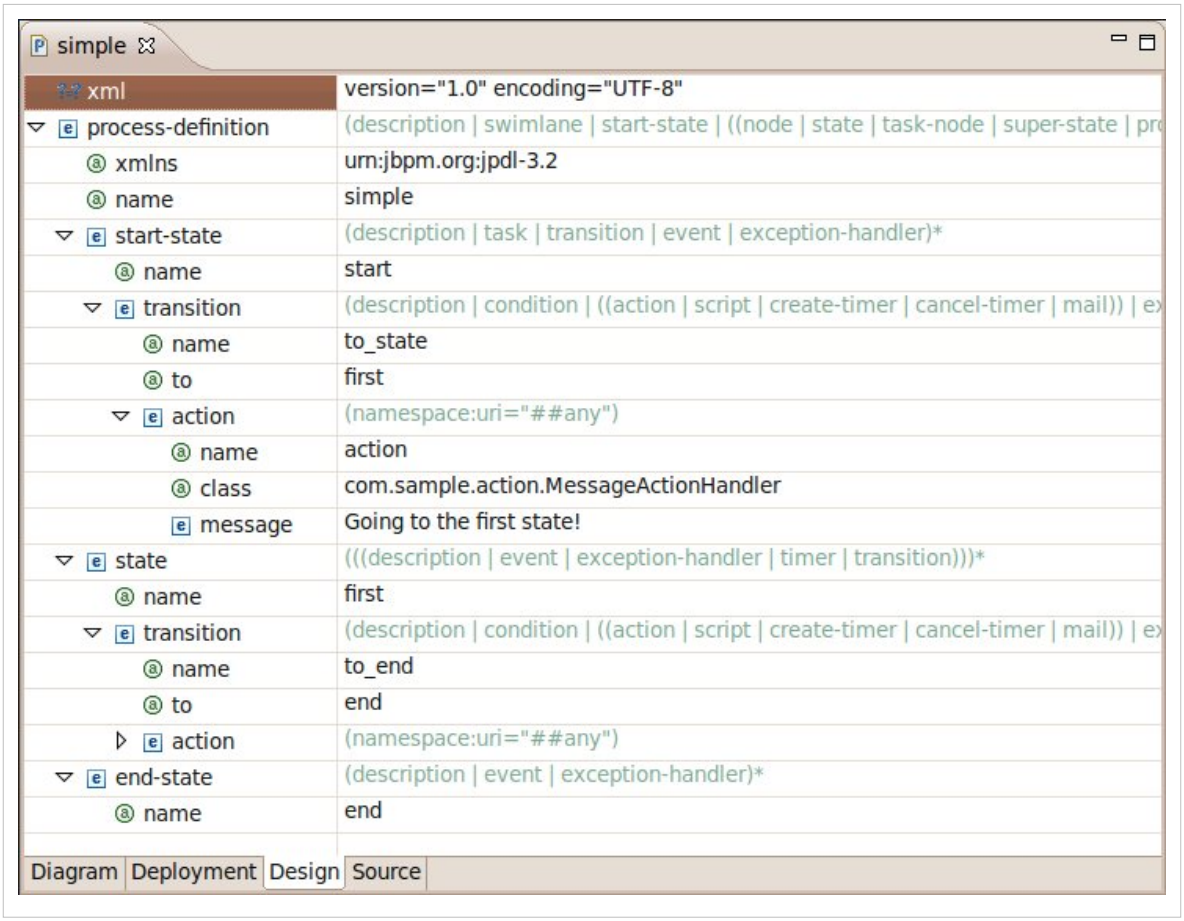
- **Diagram**: This is the main work zone. Here you can construct your process definitions on a visual way.



- **Deployment**: Here you can deploy the created process definition directly to the server or save it to deploy manually. You can create a deployable archive going to "Local Save Settings" section, marking the "Save Process Archive Locally" option, selecting a location and clicking in the "Save Without Deploying..." button. The file extension should be ".par".



- **Design**: Is the source code of the process definition. It is an hierarchical representation of the XML which describe the nodes and transitions. See jPDL xml schema chapter in jBPM jPDL User Guide for more info.



- **Source**: Is the source code of the process definition. It is an XML which describe the nodes and transitions. See jPDL xml schema chapter in jBPM jPDL User Guide for more info.

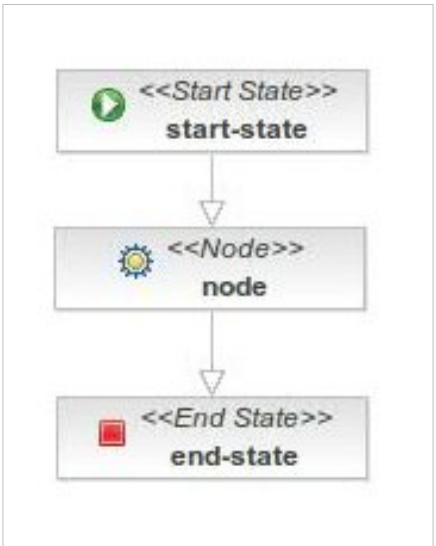
References

[1] http://docs.jboss.org/tools/whatsnew/jbpm/jbpm-news-3.2.0.M2.html

[2] http://docs.jboss.org/tools/whatsnew/jbpm/jbpm-news-3.2.0.M1.html

Hello world!

A process definition is a directed graph, made up of nodes and transitions. The hello world process has 3 nodes. To see how the pieces fit together, we're going to start with a simple process without the use of the designer tool. The following picture shows the graphical representation of the hello world process:



And this is the XML which generates this process definition graph:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="" name="Hello world">
  <start-state name="start-state">
    <transition to="node"></transition>
  </start-state>
  <node name="node">
    <transition to="end-state"></transition>
  </node>
  <end-state name="end-state"></end-state>
</process-definition>
```

This is a very basic example of a process definition. Is useless unless you make some improvements like adding an action.

Once the process definition is created, you can register it in the workflow engine. Log as okmAdmin and go to Administration → Workflow. The process definition list is actually empty. To register this new process definition go to Eclipse and to the Deployment tab. The Server Deployer input should be set to /OpenKM/upload. Click on "Deploy Process Archive..." button.

If you go to the process definition list in the OpenKM administration, you can click on the "Reload" link and the new process will appear on the list. Here you can also upload a process definition archive, if you want.

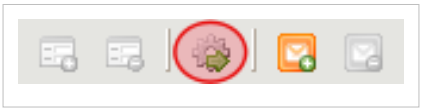
If you register a process definition several times, the old definition persist until removed. This mean that will be several versions of a process definition. This is because you can update a process definition but the old one may be executing actually.

When you start a workflow related to a document, it always will use the last version of the process definition.

Starting a workflow

You have created a process definition and registered in the engine. Now will see how to start a workflow.

From the OpenKM web interface, access to the Desktop (the default tab when you log into OpenKM). Here you can see the documents and folders stored in the repository. To start an workflow from OpenKM you must select a document and push on the button in the toolbar.

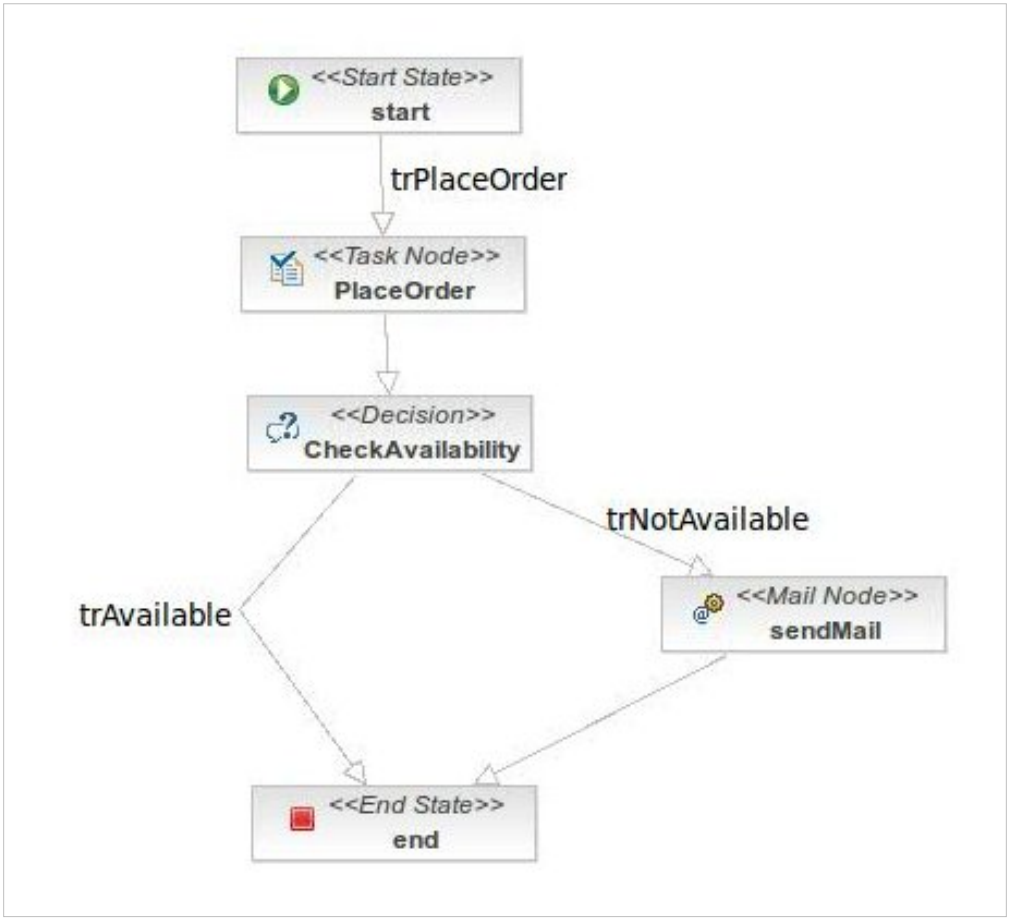


	In OpenKM all the workflows are related to a document.

In this simple case, the workflow will begin and end almost instantly because there is no request for user to input data. In the next sample will request some data to the user.

User input request

Now we will create a workflow which need some data from user. This is the workflow graph:



And this is the workflow definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition  xmlns="urn:jbpm.org:jpdl-3.2"  name="UserInput">
```

```
  <start-state name="start">
    <transition to="PlaceOrder" name="trPlaceOrder"></transition>
  </start-state>
  <task-node name="PlaceOrder">
    <task name="MyTask">
      <assignment actor-id="tazplat"></assignment>
      <controller></controller>
    </task>
    <transition to="CheckAvailability"></transition>
  </task-node>
  <decision name="CheckAvailability" expression="#{(amount&gt;100)?'trNotAvailable':'trAvailable'}">
    <transition to="end" name="trAvailable"></transition>
    <transition to="sendMail" name="trNotAvailable"></transition>
  </decision>
  <mail-node name="sendMail" to="stock@your-domain.com">
    <subject>We need more product</subject>
    <text>There is no product, so we must to buy more units!</text>
    <transition to="end"></transition>
  </mail-node>
  <end-state name="end"></end-state>
</process-definition>
```

And this is the form for the user input value (for more info read Workflow Forms description):

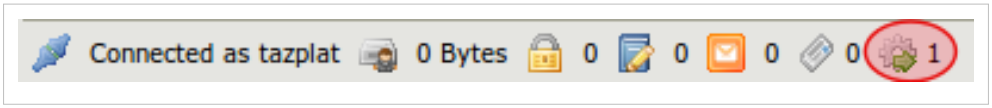
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.0//EN"

"http://www.openkm.com/dtd/workflow-forms-2.0.dtd">
<workflow-forms>
  <workflow-form task="MyTask">
    <input label="Amount" name="amount" />
    <button label="Submit" />
  </workflow-form>
</workflow-forms>
```

This process definition has a task node (with a task called "MyTask") which is assigned to the user "tazplat". You should change this user to meet an existing user in you OpenKM installation. The user have to enter a value for the amount variable. This value is evaluated in the CheckAvailability decision node by this expression:

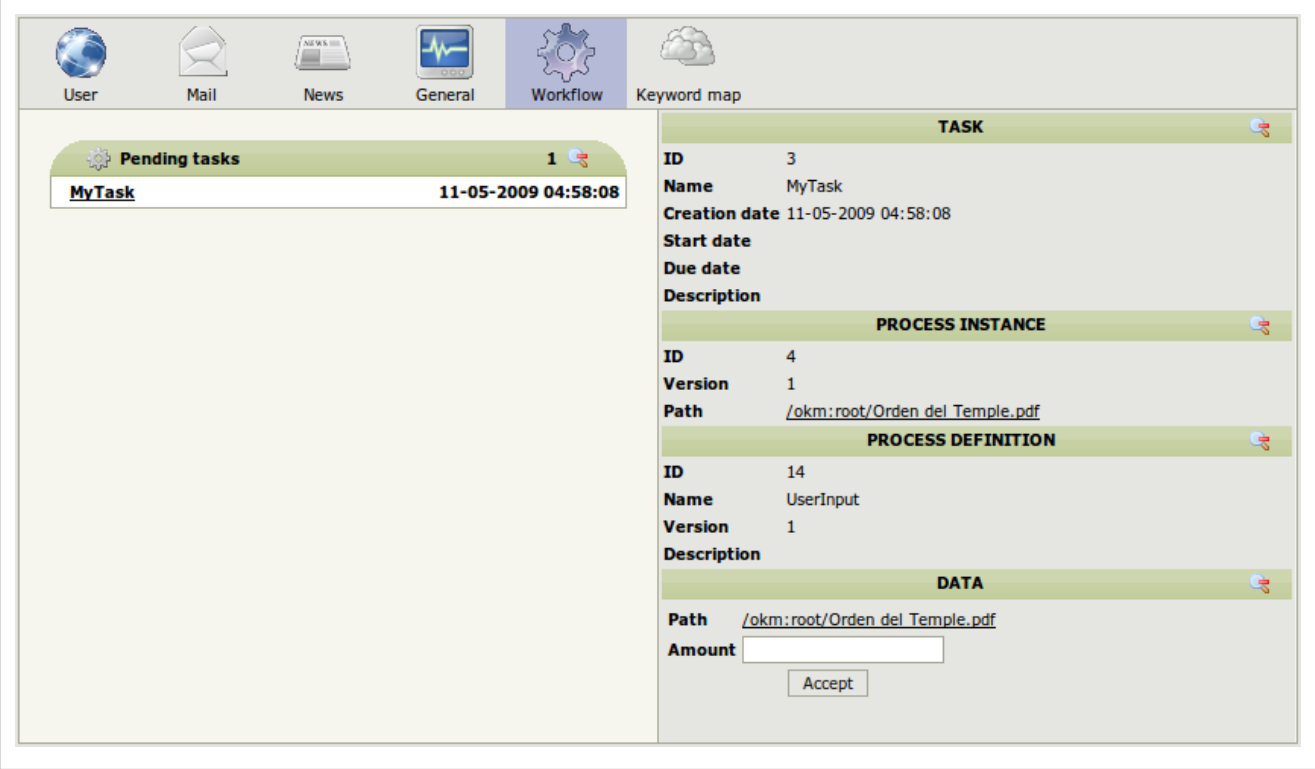
```
#{(amount.value>100)?'trNotAvailable':'trAvailable'}
```

And depending on the requested value, the flow will go across the sendMail mail node or directly to the end state.

Once the process definition is deployed in OpenKM, any user can start the workflow. When the the user input is required, a mail is sent to the user "tazplat". He should log into OpenKM web interface and will see something like this notification:



If he click on this icon, the view will switch to the dashboard. Select the task listed at **Pending** tasks and he will see the details. In the data section are displayed the variables attached to this process instance: an unmodifiable one called **Path** which describe the associated document, and an input called **Amount** where the user can enter a quantity.



Workflow Forms definition

- For **OpenKM 5.1.8** the workflow forms DTD is **workflow-forms-2.1.dtd**.
- For **OpenKM 5.1** the workflow forms DTD is **workflow-forms-2.0.dtd**.
- For **OpenKM 5.0** the workflow forms DTD is **workflow-forms-1.1.dtd**.
- For **OpenKM 4.1** is **workflow-forms-1.0.dtd**.

When you want to retrieve data from an user in the workflow, you have to define a way to do so. OpenKM uses the forms.xml file to draw the components in the form. See this sample:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.1//EN"

"http://www.openkm.com/dtd/workflow-forms-2.1.dtd">
<workflow-forms>
  <workflow-form task="MyTask">
    <input label="Amount" name="amount" />
    <button label="Submit" />
  </workflow-form>
</workflow-forms>
```

Here we can see an XML which describes the components related to a task called "MyTask". This means that when when workflow is executing the "MyTask" task, it need the user who is assigned this task to enter some information to continue. In the description, we can see an INPUT where the user should type an amount.

As you can see in the XML DOCTYPE, there is a formal definition at http:/ / www. openkm. com/ dtd/ workflow-forms-2.1.dtd.

If your OpenKM server can't access to Internet, you can download this DTD and copy to a conveniente place at you server. Remember to update your forms.xml with the TDT location.

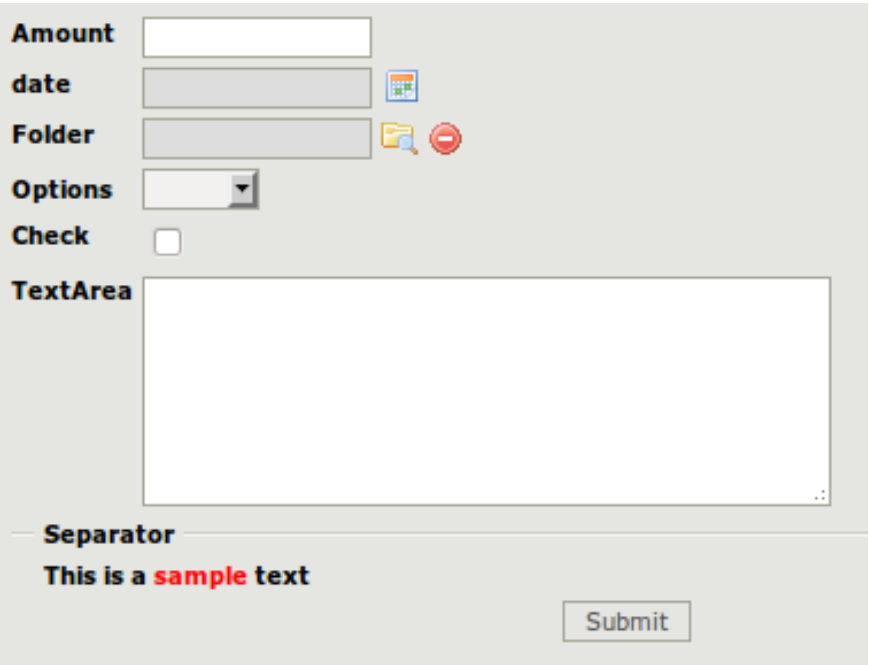
```
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.1//EN"

"file:///path/to/workflow-forms-2.1.dtd">
```

Now let's see a more complete form definition:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 2.1//EN"

"http://www.openkm.com/dtd/workflow-forms-2.1.dtd">
<workflow-forms>
  <workflow-form task="Sample">
    <input name="input" label="Input"/>
    <input name="date" label="Date" type="date"/>
    <input name="folder" label="Folder" type="folder"></input>
    <select name="options" label="Options">
      <option label="One" value="one"/>
      <option label="Two" value="two"/>
      <option label="Three" value="three"/>
    </select>
```

```
    <checkbox name="check" label="Check"/>
    <textarea name="textarea" label="TextArea"/>
    <separator name="separator" label="Separator"/>
    <text name="text" label="This is a &lt;font style='color: red'&gt;sample&lt;/font&gt; text"/>
    <button name="submit" label="Submit" />
  </workflow-form>
</workflow-forms>
```

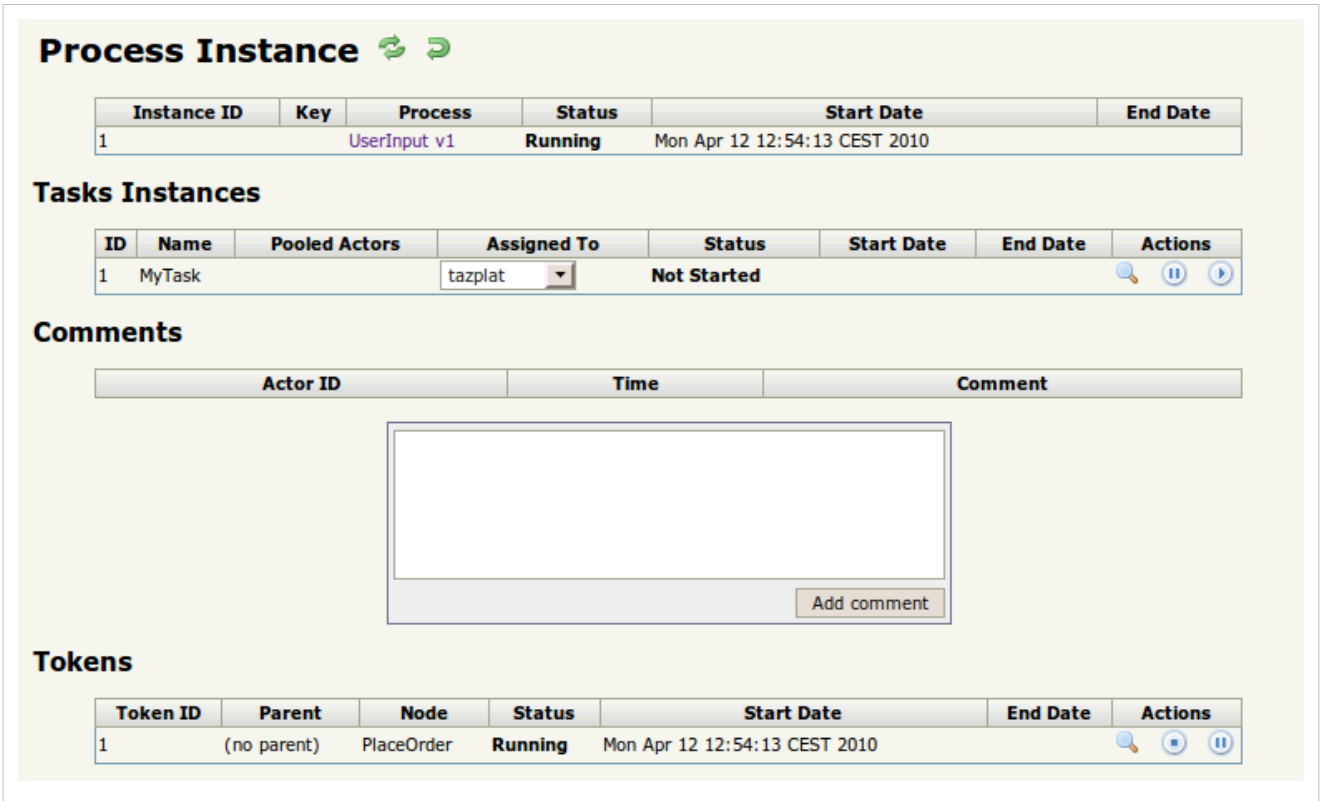
This form definition is rendered as this:



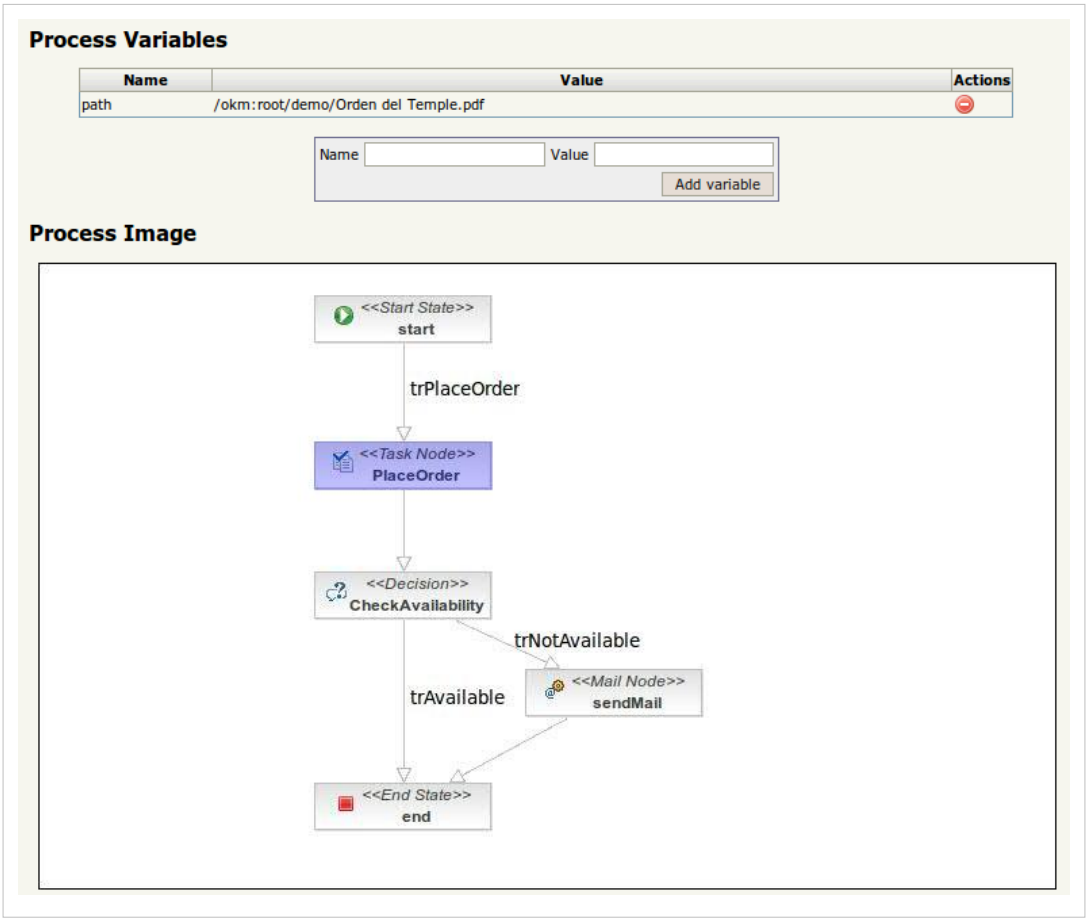
See Form Element description for a detailed definition of every form element. These form elements have also an additional property called "readonly" with can be used to make a property not modifiable by the user.

Administration interface

If you log as **okmAdmin**, you can access to the administration web interface and will see detailed info about the deployed process definitions and running process instances:



Also you can see the process definition graph with the current executed node highlighted:


Process Variables
Name
Value
Actions
path
/okm:root/demo/Orden del Temple.pdf
Name
Value
Add variable
Process Image
<<Start State>>
start
trPlaceOrder
<<Task Node>>
PlaceOrder
<<Decision>>
CheckAvailability
trNotAvailable
trAvailable
<<Mail Node>>
sendMail
<<End State>>
end

Examples

These are three sample process definitions, each one with the process image, the source xml and the associated handlers (if any). There is also a fourth element called **form definition** which is only available from OpenKM 5.0 and actually is under heavy testing. Please, contact us [1] if you want to test this new feature.

- Examples: Simple
- Examples: Medium
- Examples: Advanced

Also you can see a basic example at:

- Ejemplo Básico jBPM (I) [2] (spanish)
- Ejemplo Básico jBPM (y II) [3] (spanish)

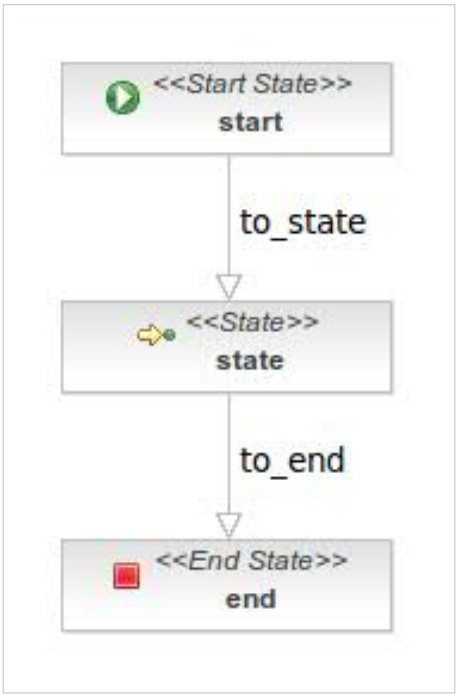
References

[1] http://www.openkm.com/Contact
[2] http://www.workflowsworld.com/2009/01/ejemplo-basico-jbpm-i/
[3] http://www.workflowsworld.com/2009/01/ejemplo-basico-jbpm-y-ii/

Examples: Simple

Download and test this process definition:  .

Process image



Process definition

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.2" name="simple">
  <start-state name="start">
    <transition name="to_state" to="state"></transition>
    <event type="node-leave">
      <script>
        print(&quot;Node start&quot;);
      </script>
    </event>
  </start-state>
  <state name="state">
    <event type="node-enter">
      <script>
        print(&quot;Node state&quot;);
        executionContext.leaveNode();
      </script>
    </event>
    <transition name="to_end" to="end">
      <action name="action" class="com.openkm.sample.MessageActionHandler">
        <message>About to finish!</message>
      </action>
    </transition>
  </state>
  <end-state name="end">
    <event type="node-enter">
      <script>
        print(&quot;Node end
(&quot;+executionContext.getVariable(&quot;message&quot;)+&quot;)&quot;);
      </script>
    </event>
  </end-state>
</process-definition>
```

Process handlers

```
package com.openkm;

import org.jbpm.graph.def.ActionHandler;
import org.jbpm.graph.exe.ExecutionContext;

public class MessageActionHandler implements ActionHandler {
  private static final long serialVersionUID = 1L;

  /**
   * The message member gets its value from the configuration in the
   * processdefinition. The value is injected directly by the engine.
```

```
    */
  String message;

  /**
   * A message process variable is assigned the value of the message
   * member. The process variable is created if it doesn't exist yet.
   */
  @Override
  public void execute(ExecutionContext context) throws Exception {
    context.getContextInstance().setVariable("message", message);
    System.out.println("From MessageActionHandler...");
  }
}
```

Form definition

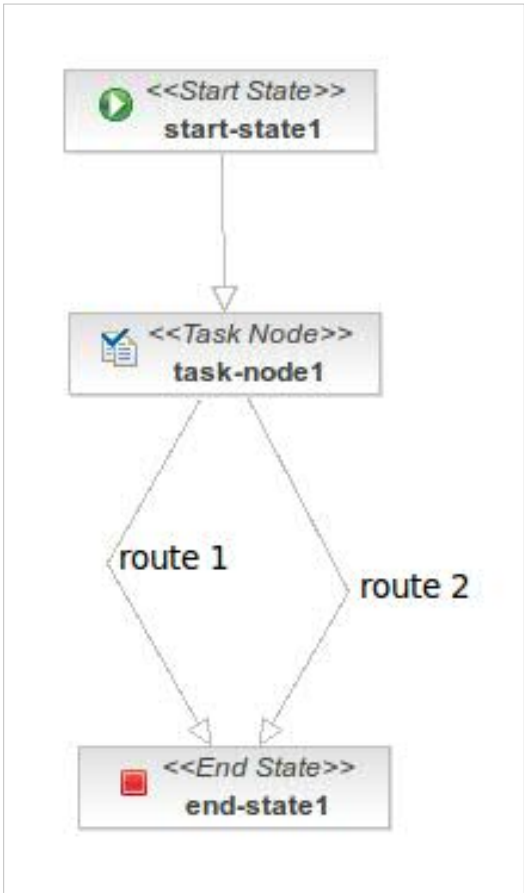
None

Examples: Medium

Download and test this process definition:  .
If you see log messages like these, don't worry because are normal:

```
INFO  [JpdlXmlReader] process xml information: no swimlane or assignment specified for task
'<task xmlns="urn:jbpm.org:jpdl-3.2" name="start" blocking="false" signalling="true" priority="normal" notify="false">
  <description>Task sample</description>
</task>'
WARN  [ProxyWarnLog] Narrowing proxy to class org.jbpm.graph.node.StartState - this operation breaks ==
WARN  [ProxyWarnLog] Narrowing proxy to class org.jbpm.graph.node.TaskNode - this operation breaks ==
WARN  [ProxyWarnLog] Narrowing proxy to class org.jbpm.graph.node.TaskNode - this operation breaks ==
```

Process image



Process definition

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition xmlns="urn:jbpm.org:jpdl-3.2" name="medium">
  <start-state name="start-state1">
    <task name="start">
      <description>Task sample</description>
    </task>
    <transition to="task-node1"></transition>
  </start-state>

  <task-node name="task-node1">
    <task name="user_info" priority="low">
      <assignment actor-id="okmAdmin"></assignment>
      <event type="task-create">
        <script>taskInstance.start();</script>
      </event>
    </task>
    <transition to="end-state1" name="route 1">
      <script>print(&quot;Going through: route 1&quot;);</script>
    </transition>
    <transition to="end-state1" name="route 2">
```

```
      <script>print(&quot;Going through: route 2&quot;);</script>
    </transition>
  </task-node>

  <end-state name="end-state1">
    <event type="node-enter">
      <script name="mensajes">
        print(&quot;End node reached: &quot;+node);
        print(&quot;Var 'quantity':
&quot;+executionContext.getVariable(&quot;quantity&quot;));
        print(&quot;Var 'name':
&quot;+executionContext.getVariable(&quot;name&quot;));
        print(&quot;Var 'type':
&quot;+executionContext.getVariable(&quot;type&quot;));
      </script>
    </event>
  </end-state>
</process-definition>
```

Process handlers

None

Form definition

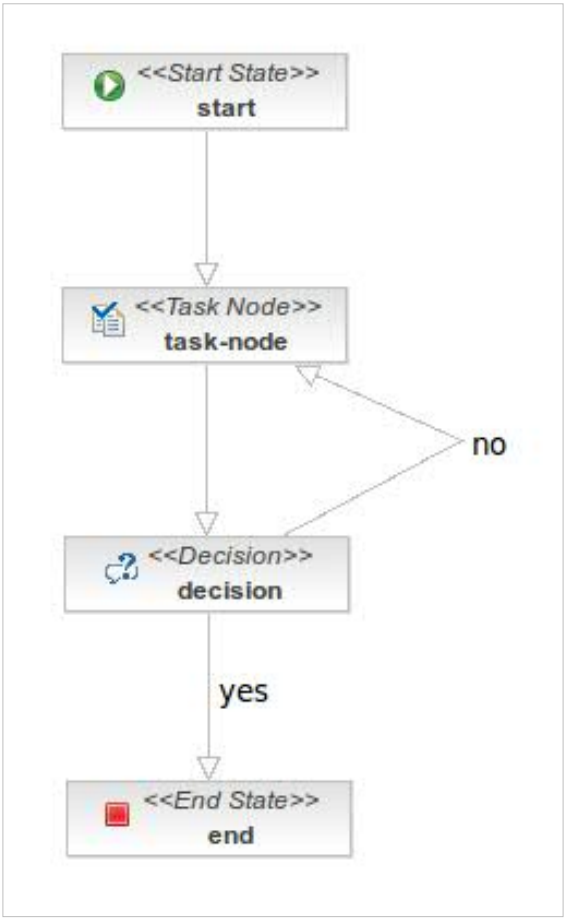
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 1.1//EN"

"http://www.openkm.com/dtd/workflow-forms-1.1.dtd">
<workflow-forms>
  <workflow-form task="start">
        <input label="Quantity" name="quantity" value="10"/>
        <button name="save" label="Save" />
  </workflow-form>
  <workflow-form task="user_info">
    <input label="Name" type="text" name="name" value="John" />
    <input label="Surname" type="text" name="surname" value="Doe" />
    <textarea label="Info" name="info" value=""/>
    <select label="Type" name="type" type="simple">
      <option label="Type 1" value="t1" />
      <option label="Type 2" value="t2" selected="true" />
      <option label="Type 3" value="t3" />
    </select>
    <button name="goto1" label="Goto 1" value="route 1" type="transition" />
    <button name="goto2" label="Goto 2" value="route 2" type="transition" />
  </workflow-form>
</workflow-forms>
```

Examples: Advanced

Download and test this process definition: .

	The taks will be assigned to an user called "monkiki" so you need to create this user and log as him to see the task assignment. Also you can assing this task to another user from the process instance workflow administration.

Process image



Process definition

```
<?xml version="1.0" encoding="UTF-8"?>
<process-definition  xmlns="urn:jbpm.org:jpdl-3.2"  name="advanced">
  <start-state name="start">
    <transition to="task-node"></transition>
  </start-state>

  <task-node name="task-node">
    <task name="guess_a_number">
```

```
      <assignment actor-id="monkiki"></assignment>
      <event type="task-create">
        <script>taskInstance.start();</script>
      </event>
    </task>
    <transition to="decision"></transition>
  </task-node>

  <decision name="decision">
    <handler class="com.openkm.sample.VerifyNumber"></handler>
      <transition to="task-node" name="no"></transition>
      <transition to="end" name="yes"></transition>
    </decision>

  <end-state name="end"></end-state>
</process-definition>
```

Process handlers

```
package com.openkm.sample;

import com.openkm.bean.form.Input;

import org.jbpm.graph.exe.Comment;
import org.jbpm.graph.exe.ExecutionContext;
import org.jbpm.graph.node.DecisionHandler;

public class VerifyNumber implements DecisionHandler {
  private static final long serialVersionUID = 1L;

  @Override
  public String decide(ExecutionContext executionContext) throws
Exception {
    Input numberStr = (Input)
executionContext.getContextInstance().getVariable("number");
    Input guessStr = (Input)
executionContext.getContextInstance().getVariable("guess");
    System.out.println("numberStr: " + numberStr);
    System.out.println("guessStr: " + numberStr);

    Integer guess = Integer.valueOf(guessStr.getValue());
    Integer number;

    if (numberStr != null && !numberStr.equals("")) {
      number = Integer.valueOf(numberStr.getValue());
    } else {
      number = 10;
    }
```

```
    if (guess > number) {
      System.out.println("Too high!");
      executionContext.getToken().addComment(new Comment("system",
guess + " is too high!"));
      return "no";
    } else if (guess < number) {
      System.out.println("Too low!");
      executionContext.getToken().addComment(new Comment("system",
guess + " is too low!"));
      return "no";
    } else {
      System.out.println("Great!");
      executionContext.getToken().addComment(new Comment("system",
guess + " is great!"));
      return "yes";
    }
  }
}
```

Form definition

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE workflow-forms PUBLIC "-//OpenKM//DTD Workflow Forms 1.1//EN"

"http://www.openkm.com/dtd/workflow-forms-1.1.dtd">
<workflow-forms>
  <workflow-form task="run_config">
    <input label="Number to guess" name="number" />
    <button name="submit" label="Submit" />
  </workflow-form>
  <workflow-form task="guess_a_number">
    <input label="Guess" name="guess" />
    <button name="submit" label="Submit" />
  </workflow-form>
</workflow-forms>
```

Article Sources and Contributors

Workflow Guide *Source*: http://wiki.openkm.com/index.php?oldid=5018 *Contributors*: Pavila

Overview *Source*: http://wiki.openkm.com/index.php?oldid=1583 *Contributors*: Pavila

JBPM installation *Source*: http://wiki.openkm.com/index.php?oldid=1551 *Contributors*: Pavila

JBPM configuration *Source*: http://wiki.openkm.com/index.php?oldid=861 *Contributors*: Pavila

Eclipse plugin *Source*: http://wiki.openkm.com/index.php?oldid=870 *Contributors*: Pavila

Eclipse plugin: Installation *Source*: http://wiki.openkm.com/index.php?oldid=4591 *Contributors*: Pavila

Eclipse plugin: Usage *Source*: http://wiki.openkm.com/index.php?oldid=4280 *Contributors*: Pavila

Hello world! *Source*: http://wiki.openkm.com/index.php?oldid=1015 *Contributors*: Pavila

Starting a workflow *Source*: http://wiki.openkm.com/index.php?oldid=1032 *Contributors*: Pavila

User input request *Source*: http://wiki.openkm.com/index.php?oldid=4651 *Contributors*: Pavila

Workflow Forms definition *Source*: http://wiki.openkm.com/index.php?oldid=5064 *Contributors*: Pavila

Administration interface *Source*: http://wiki.openkm.com/index.php?oldid=1010 *Contributors*: Pavila

Examples *Source*: http://wiki.openkm.com/index.php?oldid=4468 *Contributors*: Pavila

Examples: Simple *Source*: http://wiki.openkm.com/index.php?oldid=4485 *Contributors*: Pavila

Examples: Medium *Source*: http://wiki.openkm.com/index.php?oldid=4486 *Contributors*: Pavila

Examples: Advanced *Source*: http://wiki.openkm.com/index.php?oldid=4477 *Contributors*: Pavila

Image Sources, Licenses and Contributors

File:Padlock.gif *Source*: http://wiki.openkm.com/index.php?title=File:Padlock.gif *License*: unknown *Contributors*: Pavila

File:Nota idea.png *Source*: http://wiki.openkm.com/index.php?title=File:Nota_idea.png *License*: unknown *Contributors*: Pavila

File:Okm workflow guide 001.png *Source*: http://wiki.openkm.com/index.php?title=File:Okm_workflow_guide_001.png *License*: unknown *Contributors*: Pavila

File:Jbpm install 01.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_install_01.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm install 02.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_install_02.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm install 03.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_install_03.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm install 04.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_install_04.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm install 05.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_install_05.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm install 06.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_install_06.jpg *License*: unknown *Contributors*: Pavila

File:Nota clasica.png *Source*: http://wiki.openkm.com/index.php?title=File:Nota_clasica.png *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse install 01.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_install_01.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse install 02.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_install_02.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse install 03.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_install_03.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 01.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_01.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 02.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_02.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 03.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_03.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 04.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_04.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 05.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_05.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 06.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_06.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 07.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_07.jpg *License*: unknown *Contributors*: Pavila

File:Jbpm eclipse new 08.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Jbpm_eclipse_new_08.jpg *License*: unknown *Contributors*: Pavila

File:Hello.world.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Hello.world.jpg *License*: unknown *Contributors*: Pavila

File:Openkm.toolbar.workflow.png *Source*: http://wiki.openkm.com/index.php?title=File:Openkm.toolbar.workflow.png *License*: unknown *Contributors*: Pavila

File:Workflow.user.input.jpg *Source*: http://wiki.openkm.com/index.php?title=File:Workflow.user.input.jpg *License*: unknown *Contributors*: Pavila

File:Openkm.status.workflow.png *Source*: http://wiki.openkm.com/index.php?title=File:Openkm.status.workflow.png *License*: unknown *Contributors*: Pavila

File:Openkm.dashboard.workflow.png *Source*: http://wiki.openkm.com/index.php?title=File:Openkm.dashboard.workflow.png *License*: unknown *Contributors*: Pavila

File:Workflow Forms 1.png *Source*: http://wiki.openkm.com/index.php?title=File:Workflow_Forms_1.png *License*: unknown *Contributors*: Pavila

File:Openkm.admin.workflow.1.png *Source*: http://wiki.openkm.com/index.php?title=File:Openkm.admin.workflow.1.png *License*: unknown *Contributors*: Pavila

File:Openkm.admin.workflow.2.png *Source*: http://wiki.openkm.com/index.php?title=File:Openkm.admin.workflow.2.png *License*: unknown *Contributors*: Pavila

File:Simple.par *Source*: http://wiki.openkm.com/index.php?title=File:Simple.par *License*: unknown *Contributors*: Pavila

File:Workflow example simple.png *Source*: http://wiki.openkm.com/index.php?title=File:Workflow_example_simple.png *License*: unknown *Contributors*: Pavila

File:Medium.par *Source*: http://wiki.openkm.com/index.php?title=File:Medium.par *License*: unknown *Contributors*: Pavila

File:Workflow example medium.png *Source*: http://wiki.openkm.com/index.php?title=File:Workflow_example_medium.png *License*: unknown *Contributors*: Pavila

File:Advanced.par *Source*: http://wiki.openkm.com/index.php?title=File:Advanced.par *License*: unknown *Contributors*: Pavila

File:Workflow example advanced.png *Source*: http://wiki.openkm.com/index.php?title=File:Workflow_example_advanced.png *License*: unknown *Contributors*: Pavila