Runa WFE Graphical Process Designer. User guide.

Table of Contents

Introduction

RUNA GPD 2.0 is a graphical process designer for Runa WFE 2.0. This document describes how to develop business processes with the help of RUNA GPD 2.0[1].

Installation guide

RUNA GPD is distributed as a set of binary files – one file for each of the supported platforms.

Required software

- JRE or JDK 5.0 or higher, can be downloaded from http://java.sun.com/j2se/1.5.0/download.jsp

Install RUNA GPD

1. Install JDK (http://java.sun.com/j2se/1.5.0/install.html)
2. Unpack runa-gpd-*.zip archive and go to the `gpd-x.x.x` directory
3. Run `runa-gpd`

1 RUNA GPD generates .par archive files. Each .par file contains business process definition written in jPdl language. Reference documentation for jPdl language can be found at http://www.jboss.com/products/jbpm/docs/jPdl.

Building from Source

Required software

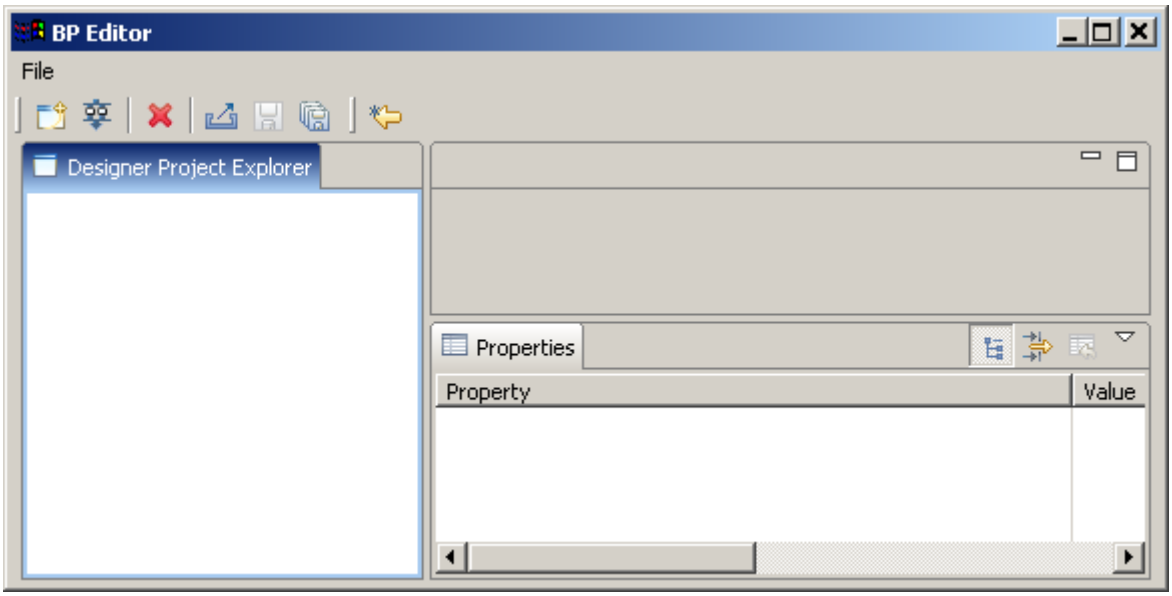
- JDK 5.0 or higher, can be downloaded from http://java.sun.com/j2se/1.5.0/download.jsp
- Eclipse IDE 3.2.2, can be downloaded from http://www.eclipse.org/downloads/
- Eclipse WTP 1.5.4 with all required plug-ins, can be downloaded from http://download.eclipse.org/webtools/downloads/drops/M-0.7.1-200509270720/

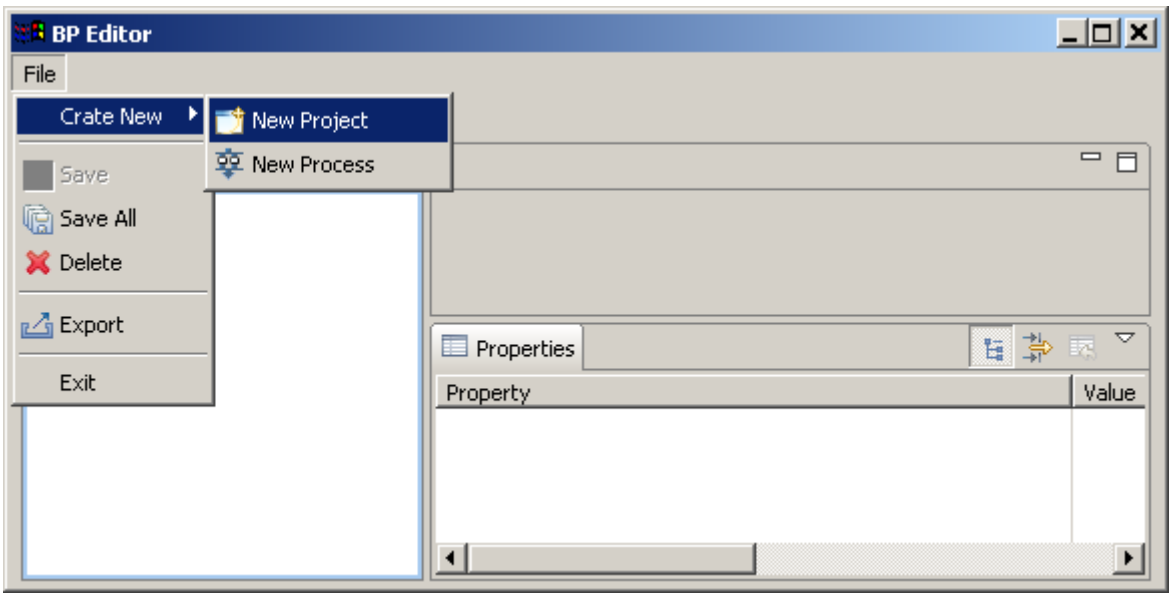
Install RUNA GPD

- Install JRE or JDK (http://java.sun.com/j2se/1.5.0/install.html)
- Install Eclipse IDE and WTP plug-ins
- Download the required version of GPD sources from the SVN repository
- Run Eclipse
- Import unpacked RUNA GPD plug-ins into you Eclipse workspace
- Run RUNA GPD plug-in using gpd.product file from org.jbpm.ui plugin.

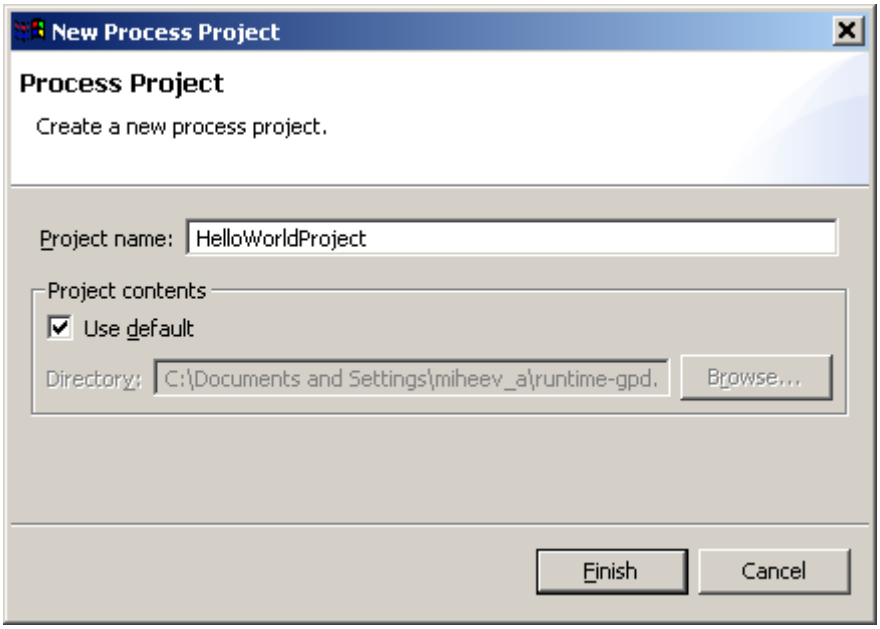
Creating new RUNA GPD Project

1. Inside RUNA GPD select the menu item **File > Create New >New Project....** to open the New Project wizard.

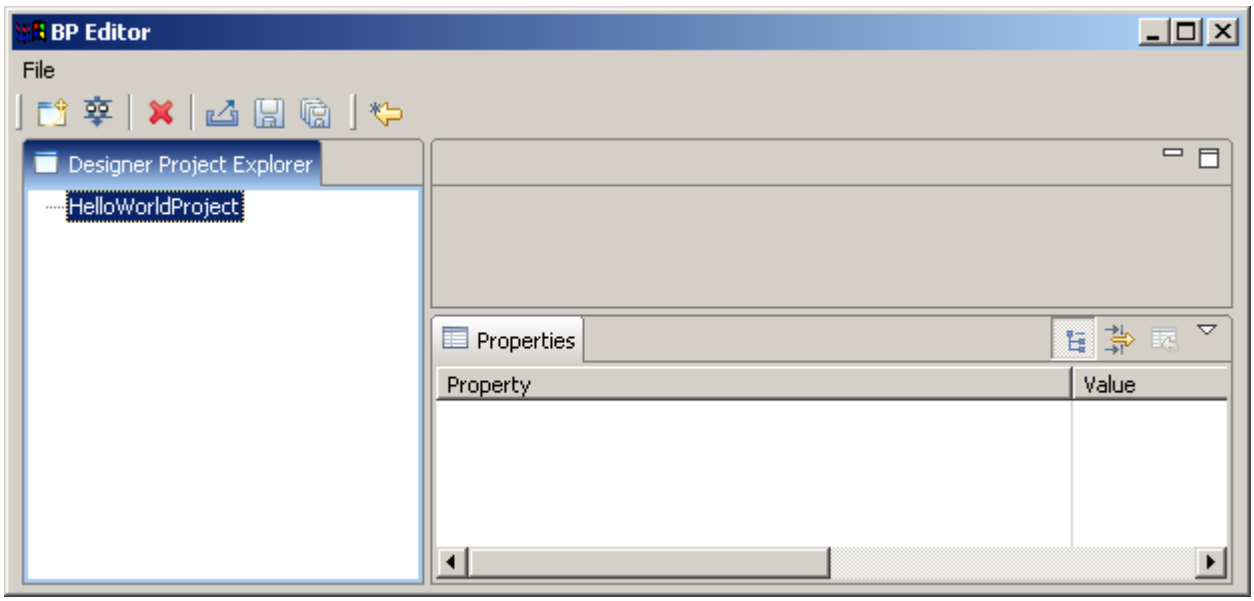




2. Enter the project name “HelloWorldProject”.



The HelloWorldProject project will be created.



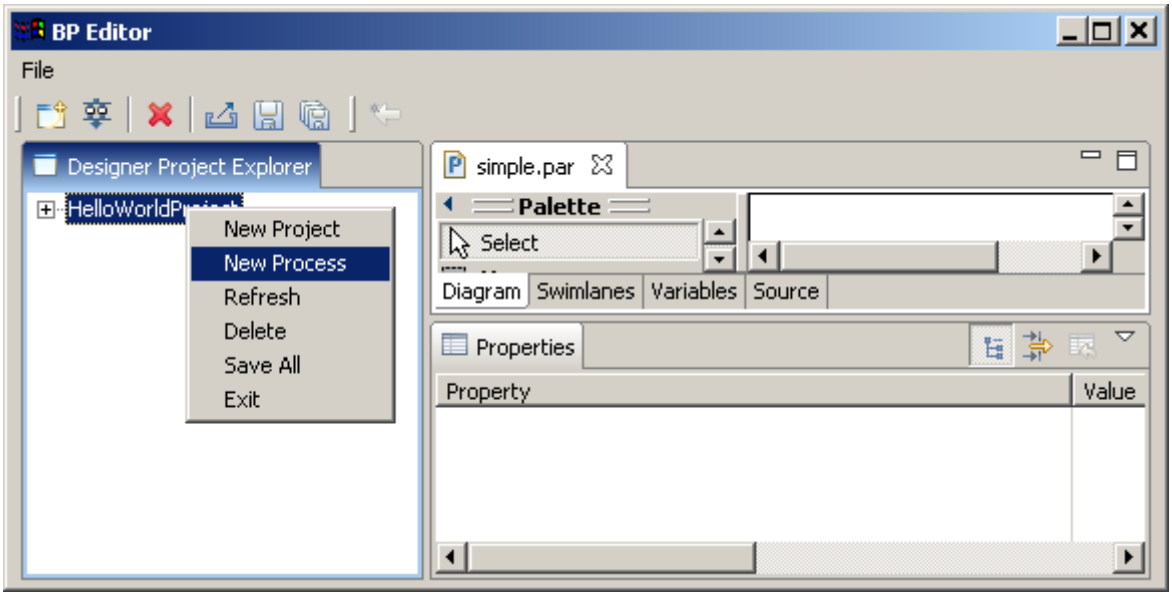
Creating HelloWorld process[2]

The process scenario

- On process start HelloWorld starting form appears.
- When "complete" button on the form is pressed process immediately ends.

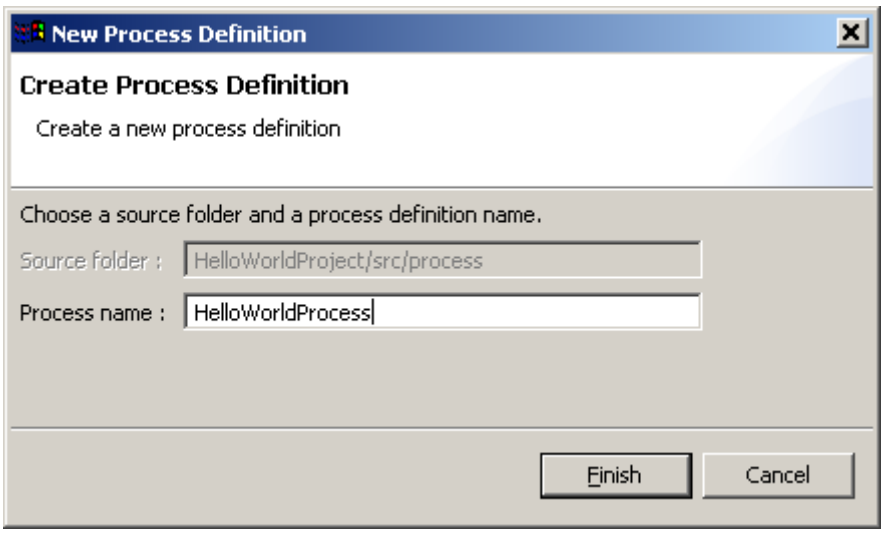
Process graph creation

Open context menu by right mouse button clicking on HelloWorldProject, then click on New Process menu item.

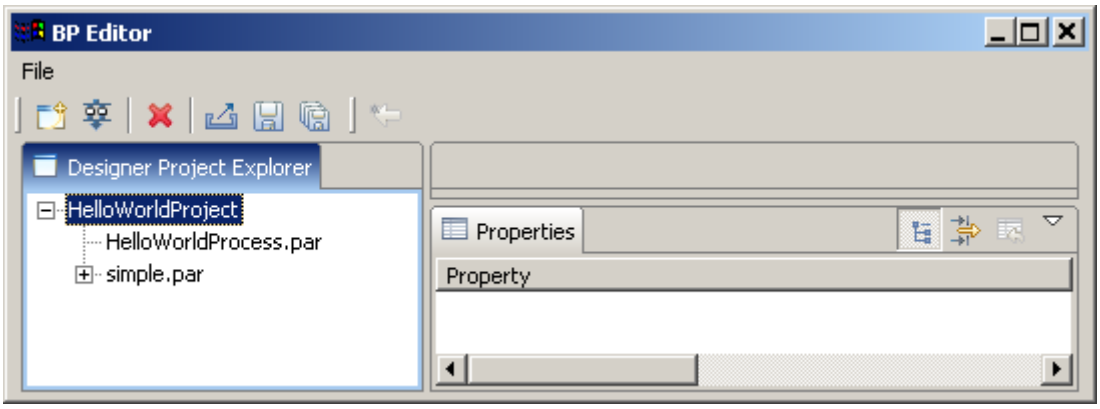


Enter HelloWorldProcess as the process name:

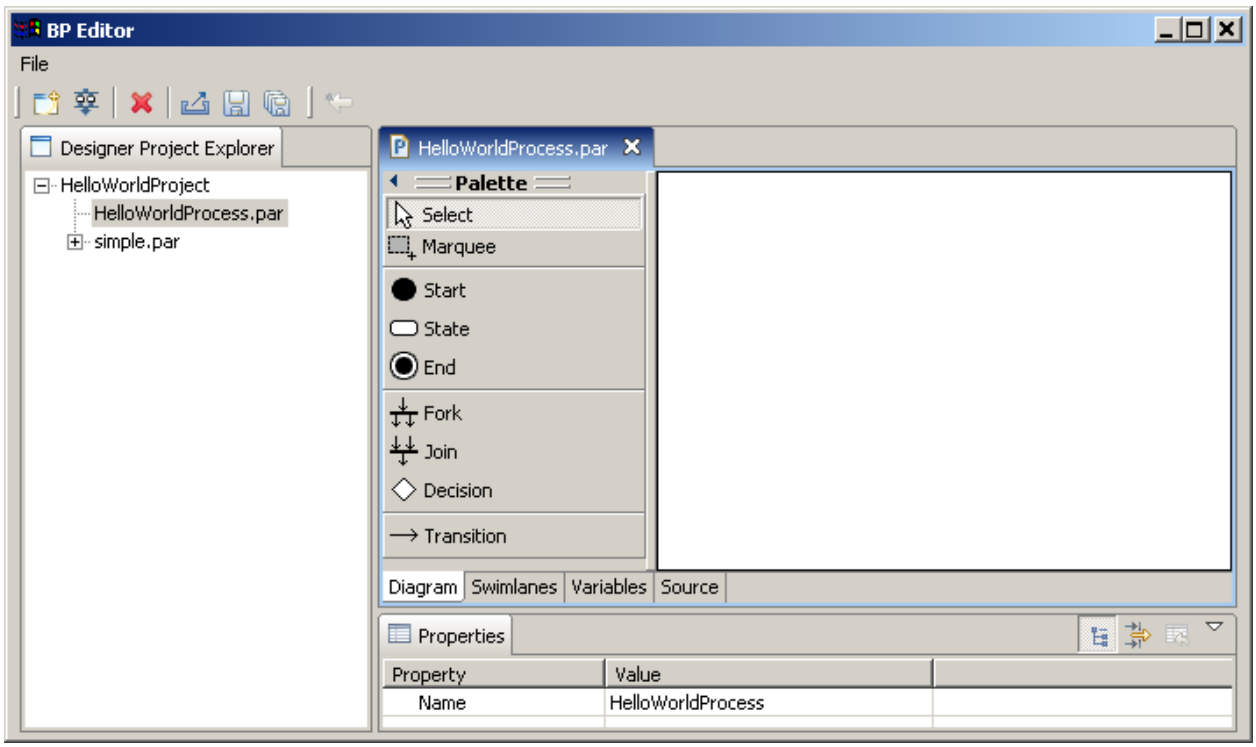
2 The process has two nodes: Start-state and Stop-state



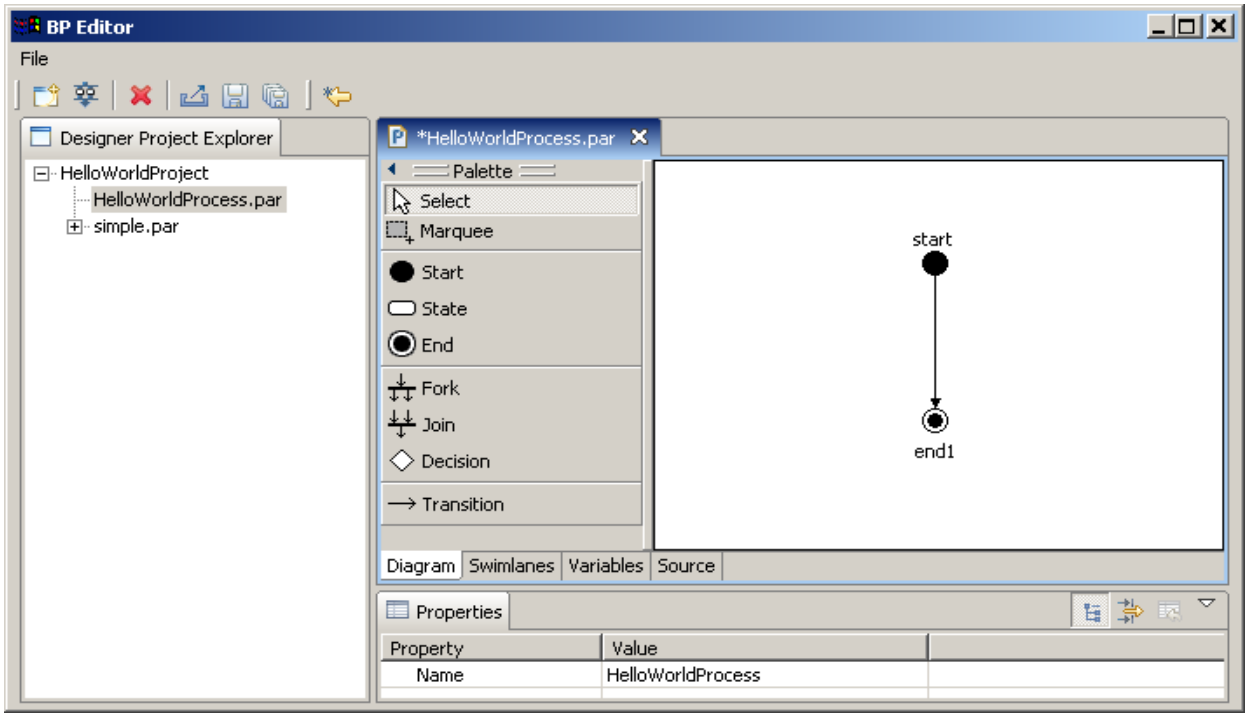
HelloWorldProcess will be created.



Double click on the HelloWorldProcess.par. The process diagram window will appear:



Click on “Start” element on the Palette, then click on diagram window. The Start state will appear on diagram window. Similarly place the End state on diagram, then click on “transition” element on the palette and connect Start and End states.



The process graph is ready.

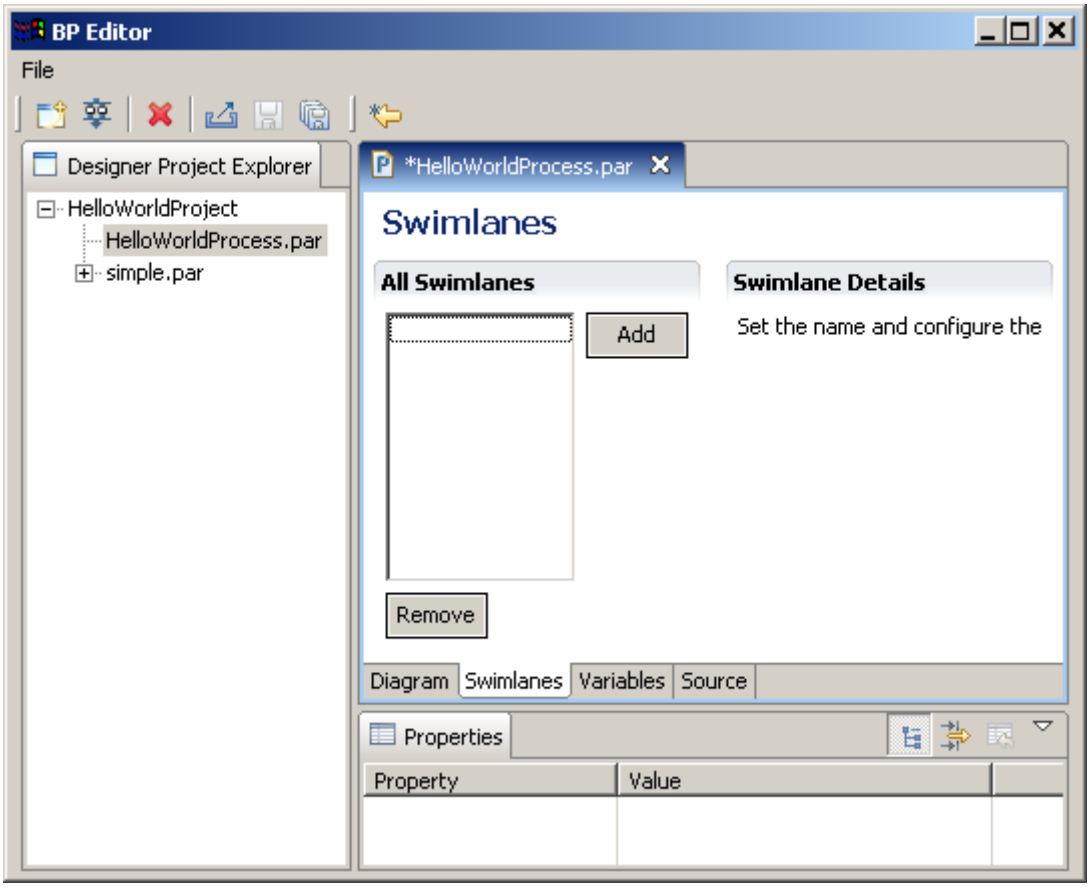
Swimlanes creation

Introduction to swimlanes

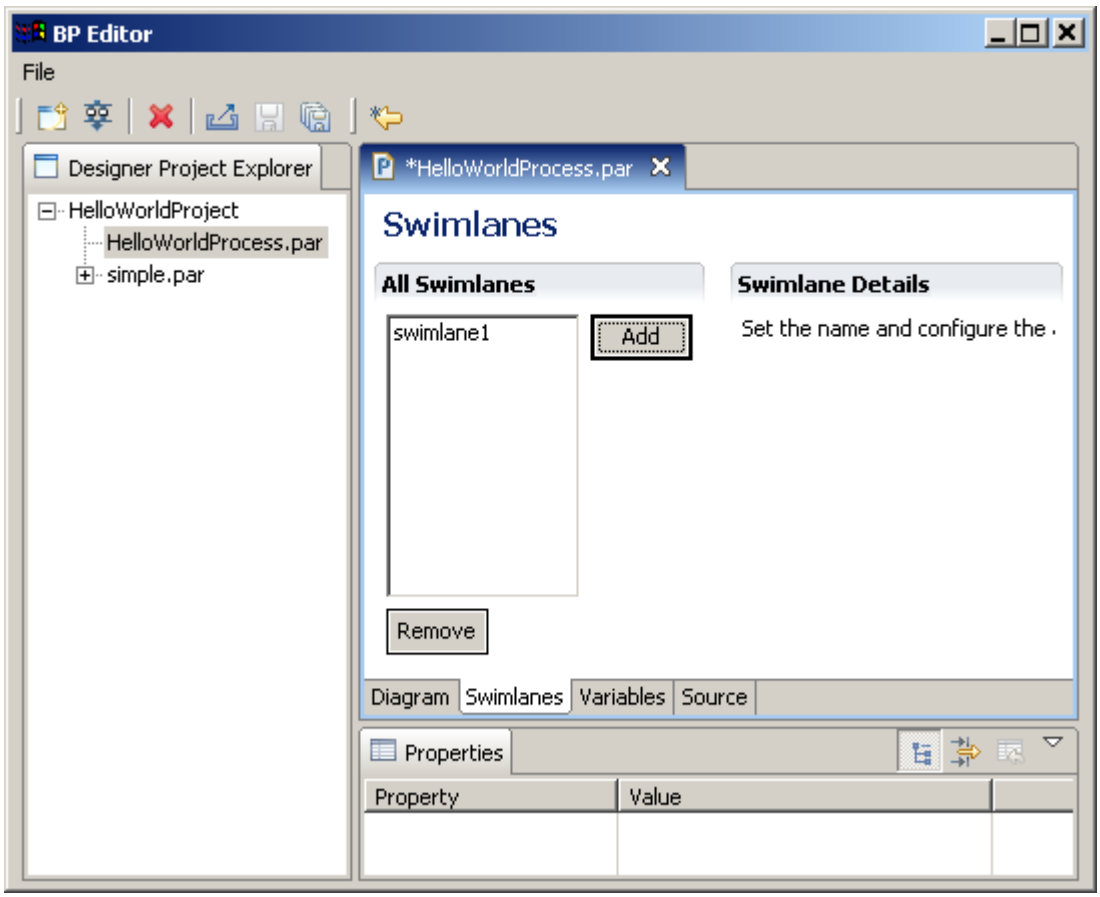
Swimlanes corresponds to business process roles. In JBoss jBPM swimlane is a special business process variable. Every state has swimlane associated with it. The start state behavior regarding swimlane differs the other state behavior. The start state fills start state swimlane with user ID. The end state has no swimlane associated. Other states use associated swimlanes to determine who can execute this state.

Swimlanes for HelloWorld process

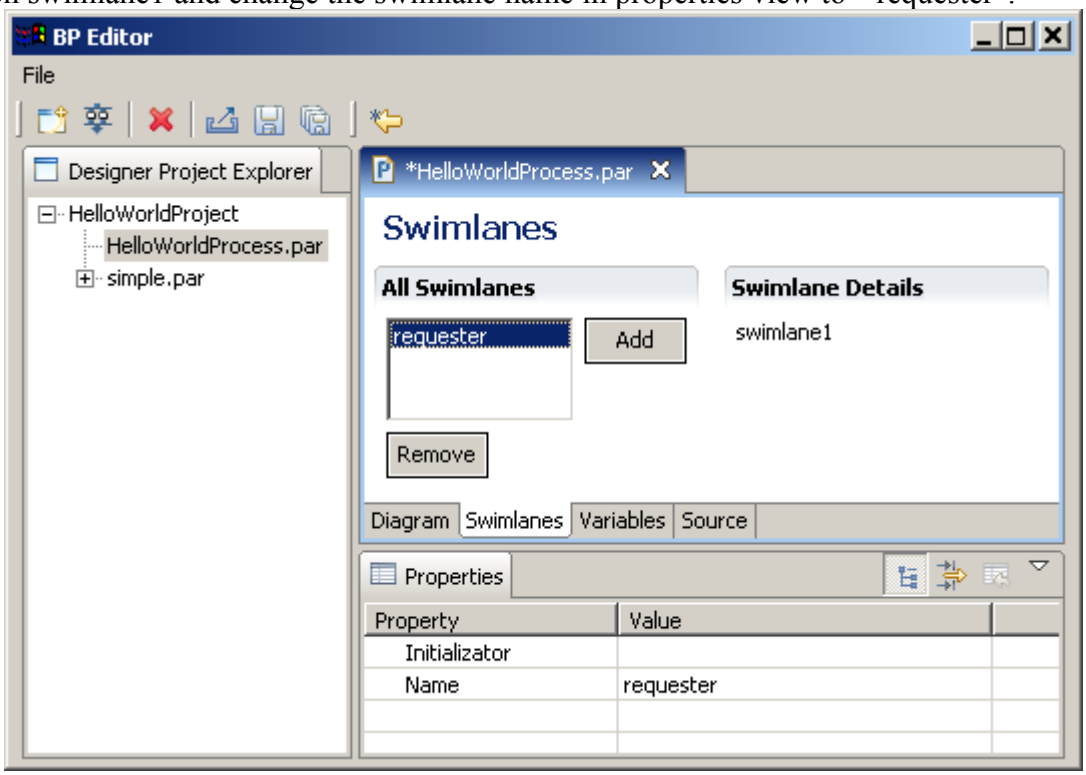
The process has only start and end states, so there is single swimlane in process.
Click Swimlanes tab. You'll see the following:



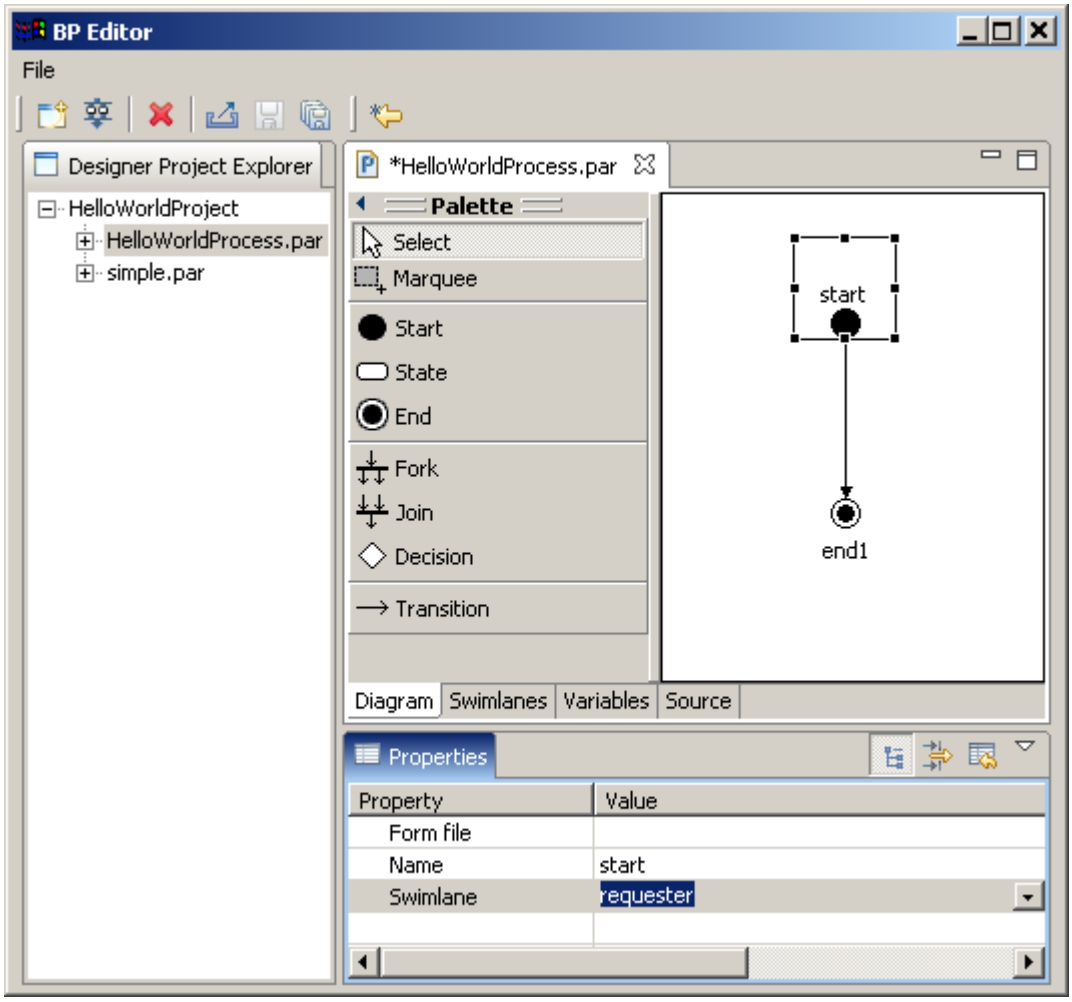
Then click add button. The “swimlane1” swimlane will appear.



Click on swimlane1 and change the swimlane name in properties view to "requester".



Click Diagram tab, click start state and choose "requester" as Swimlane value in properties view.



Graphical form creation

Form files

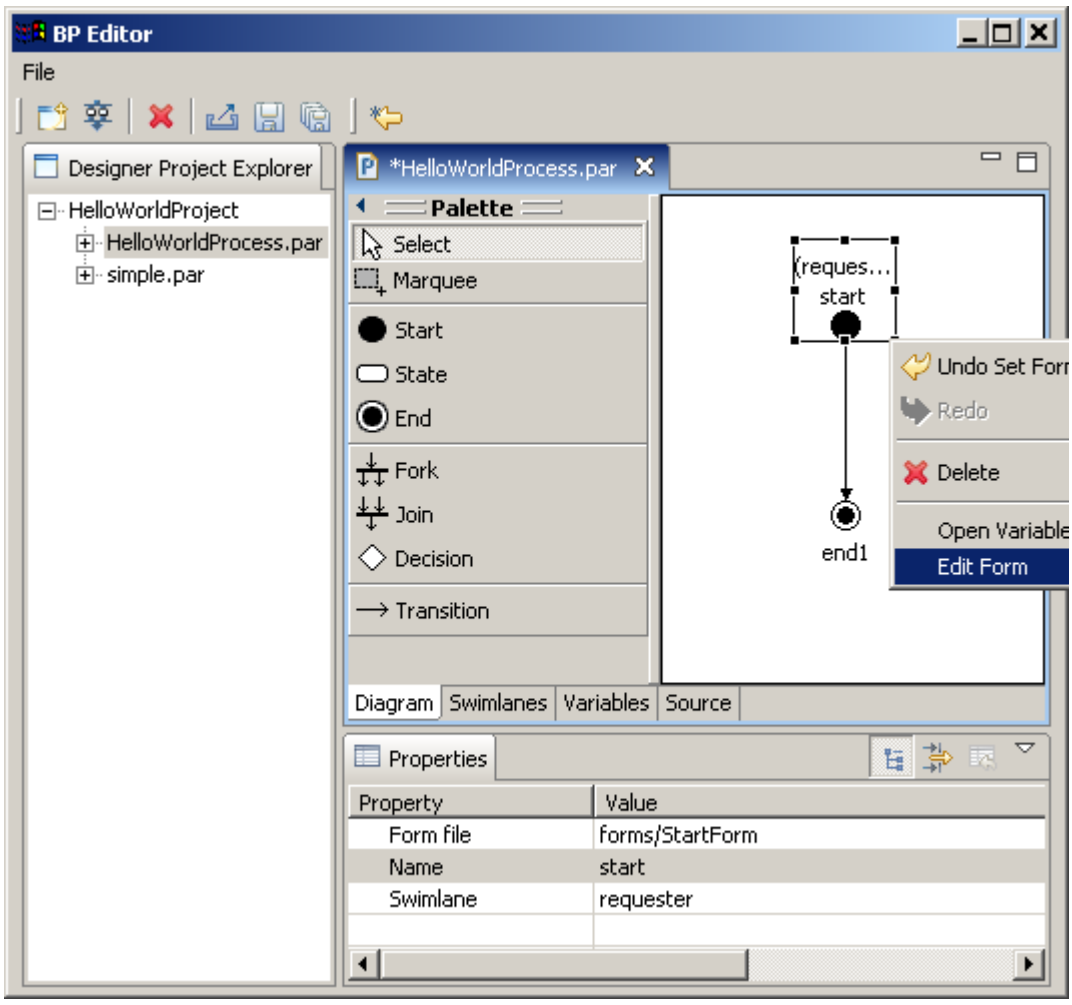
Every .form file contains the state form description. The reference form parsing mechanism uses HTML with additional tags `<customtag>`. These tags are used to display process variable value in the form.

The `< customtag>` tag have the following attributes:

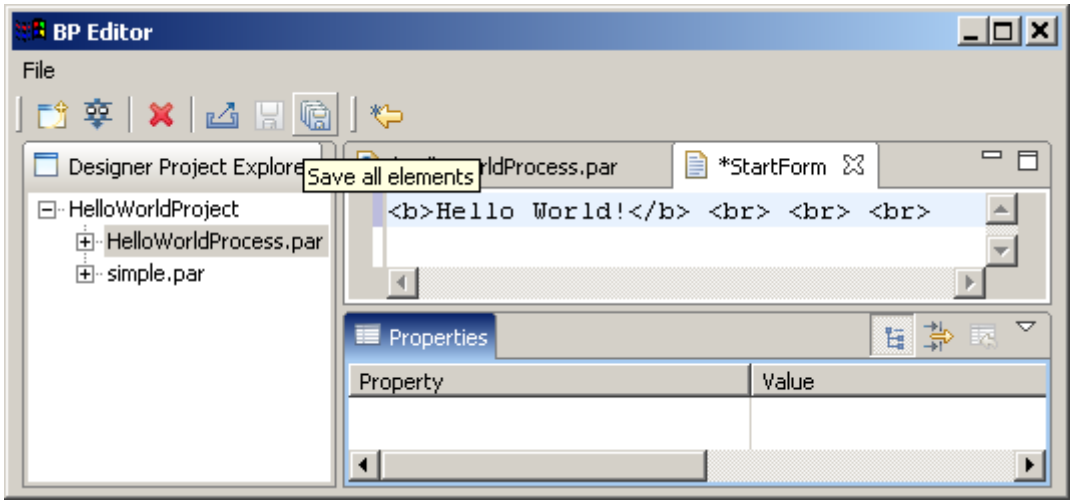
- `var` – process variable name
- `delegation` – name of Java class, used for the variable value rendering

Form file creation

HelloWorld process has no variables and only one form – start form. Click start state and set “forms/StartForm” as Form file value in properties view. Click start state by right mouse button, then click on Edit form command.

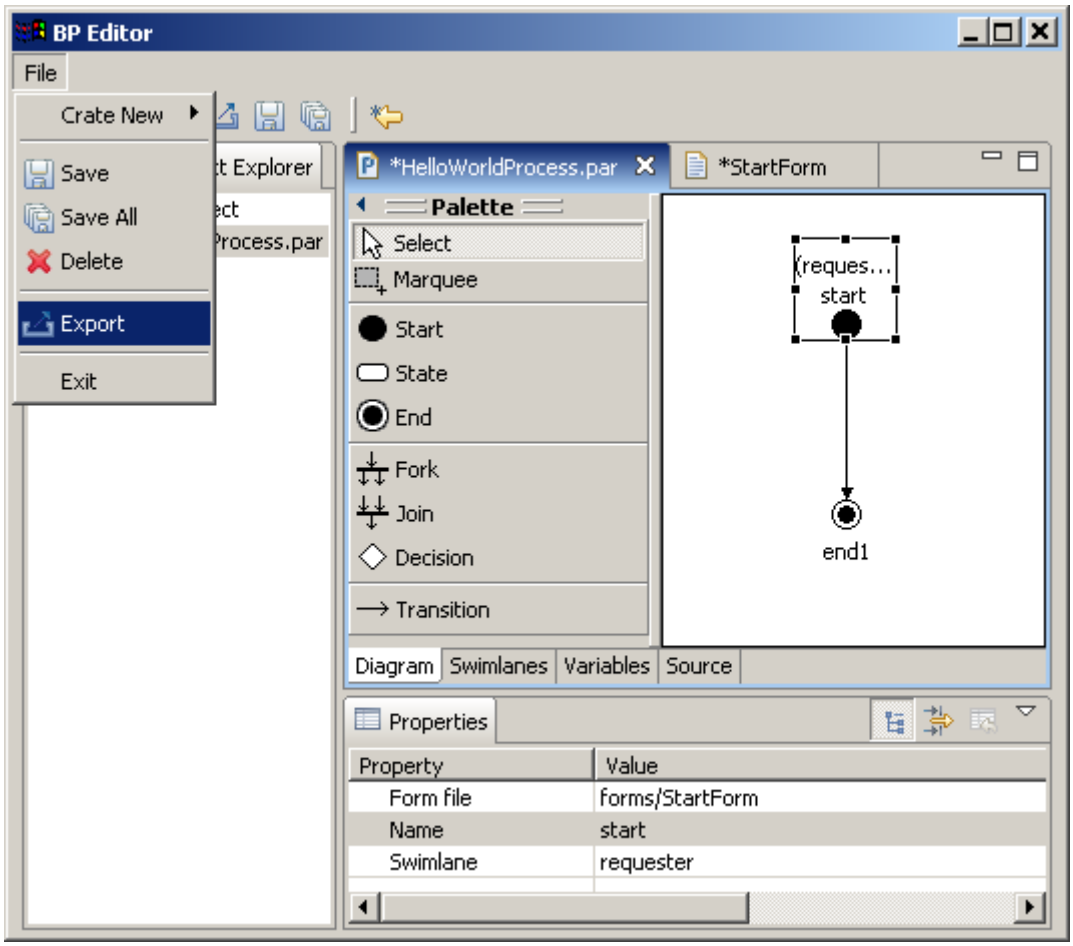


Enter text: `<b>Hello World!</b> <br> <br> <br>` into StartForm editor window. Then click “save all elements” button on toolbar.

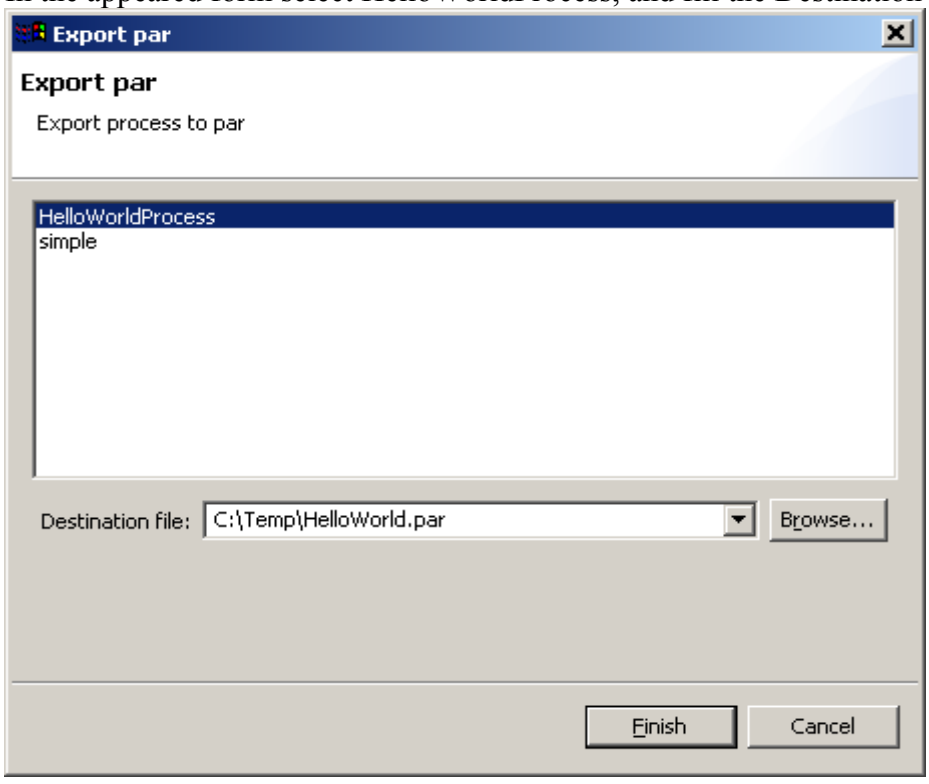


Process definition archive creation

Select HelloWorldProcess.par, then open menu item File/Export to create .par process archive.



In the appeared form select HelloWorldProcess, and fill the Destination file field.

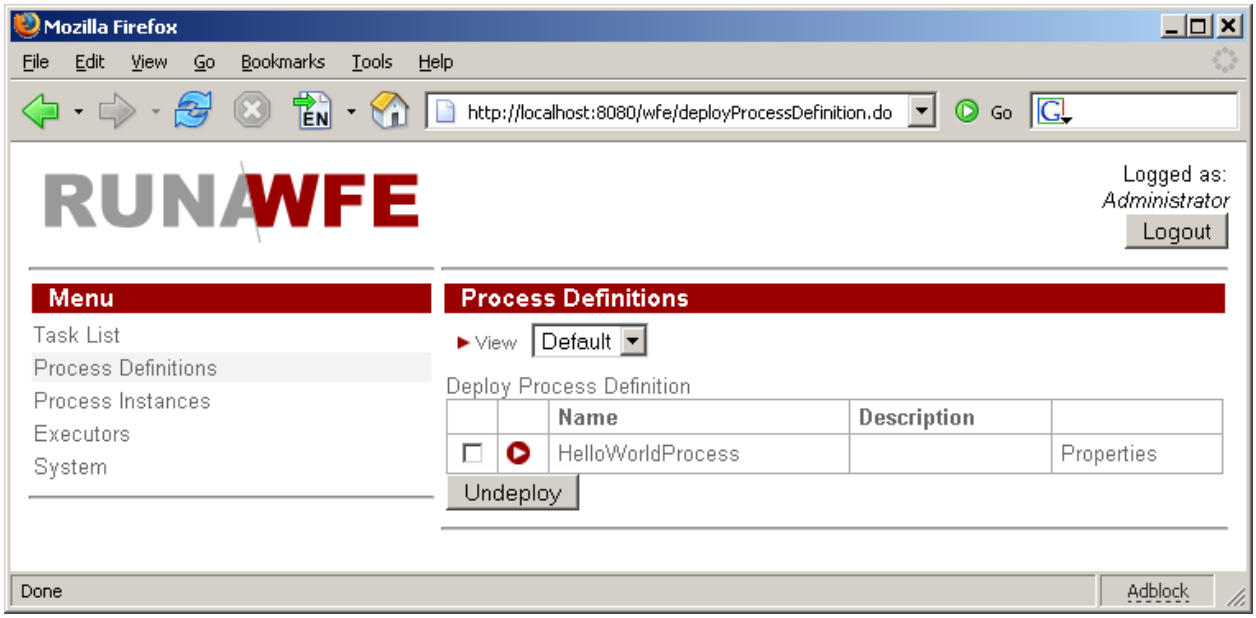


Then click “finish” button. The HelloWorld process file will be generated.

Deployment into RUNA WFE environment

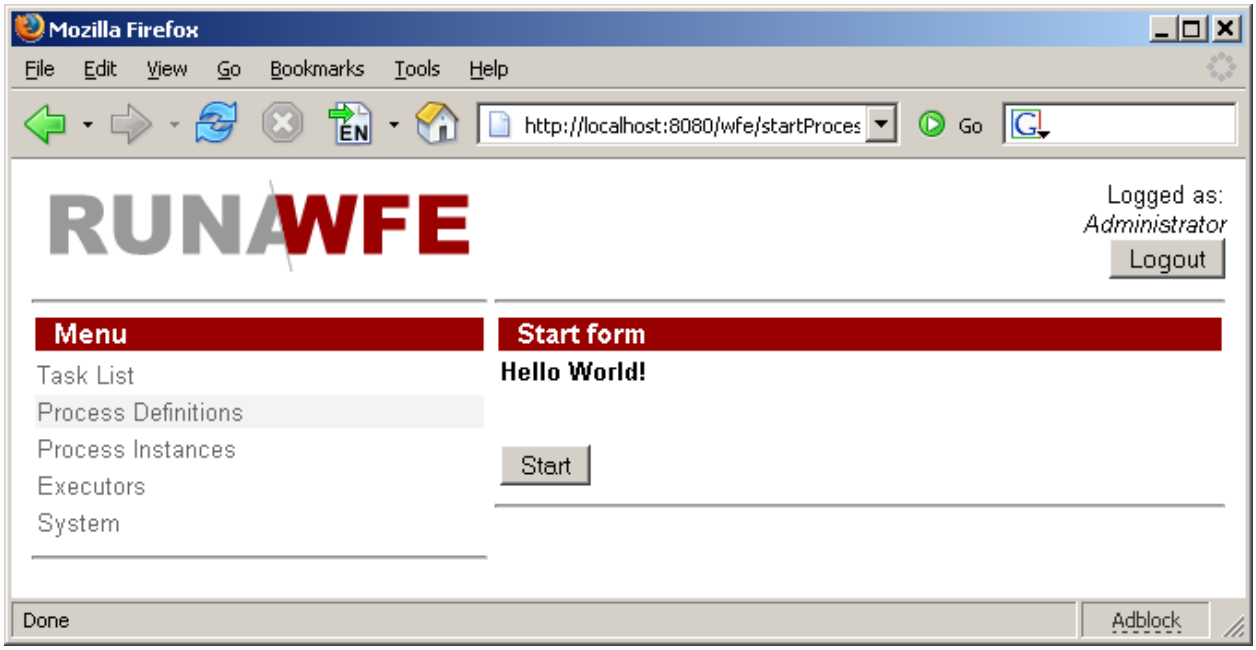
Login into RUNA WFE web interface as Administrator (The default Administrators password is wf, see RUNA WFE 2.0 manual for details).

Go to “process definition” menu. Press deploy process definition[3].
Press browse button to select process definition archive.
Press Ok button.



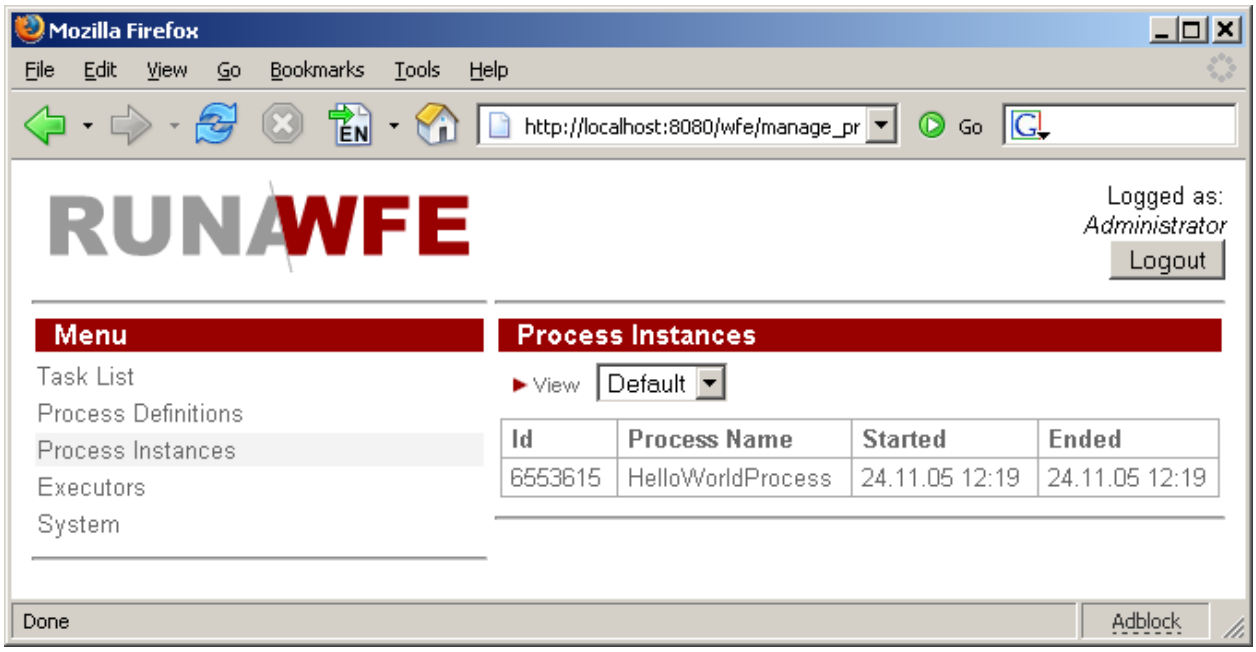
Running Hello World process

Click on process name in “process definition” menu. You'll see the start form:



3*Note:In order to deploy a process you must have Process Definition permission on System (can be granted via system menu).*

Click on “Start” button. The “Hello World” process starts and immediately ends. You can see the process instance in the “Process Instances” menu:



Creating OverTime demo process

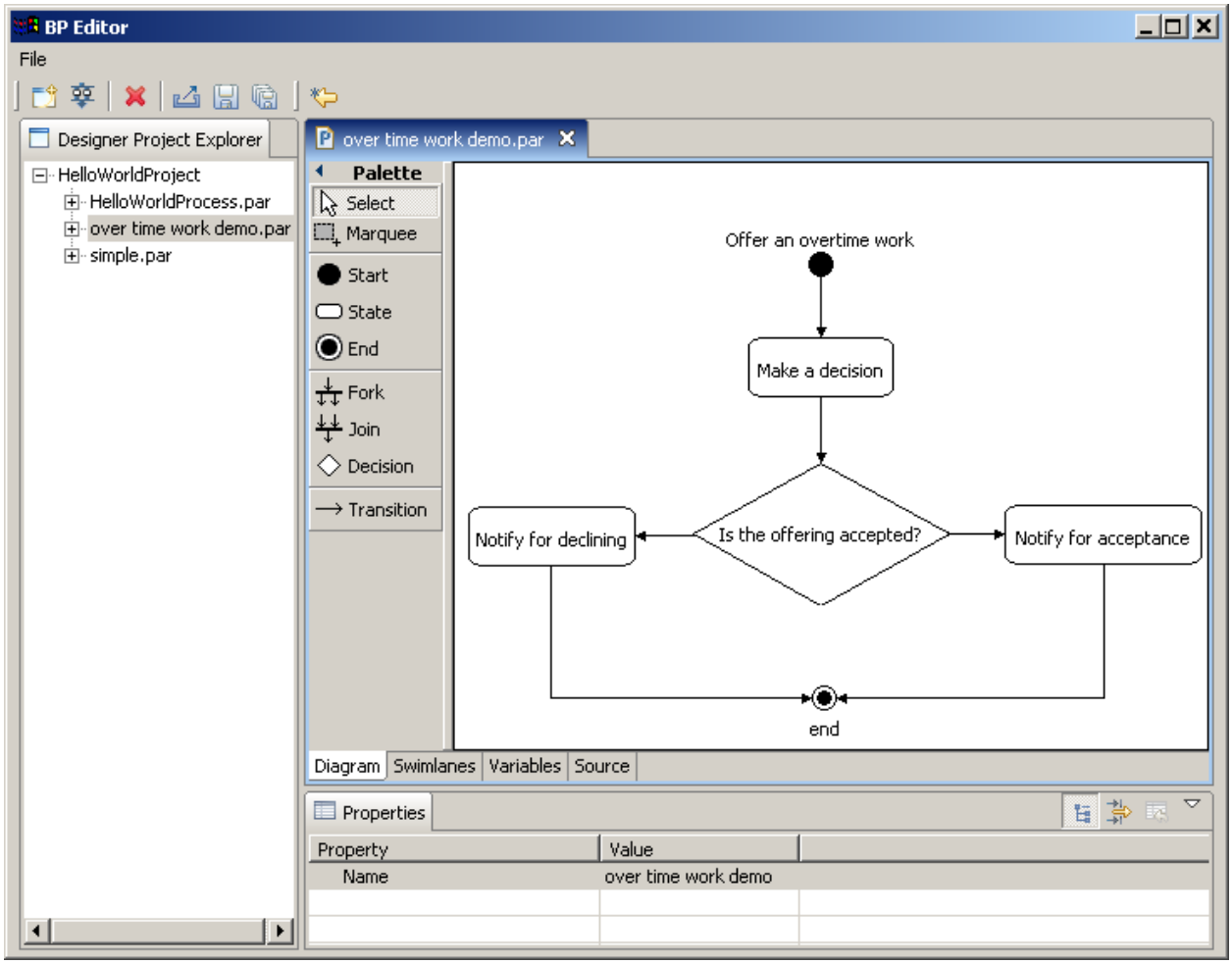
The process scenario

Manager offers an over time work to employee. Employee accepts or rejects the offer. Then manager receives the corresponding notification.

It is supposed that all managers are members of group “manager” and all employees are members of group “staff”.

Process graph creation

Open context menu by right mouse button clicking on HelloWorldProject, then click on New Process command. Enter “over time work demo” as the process name. Double click on the “over time work demo.par”. The process diagram window will appear. By clicking at palette elements and process diagram window create the following graph for business process (also it is necessary to rename states (in properties view)):



Swimlanes creation

Swimlanes overview

Swimlanes behavior in StartState and EndState were described in “Creating Hello World process” section. Other state use associated swimlanes to determine who can execute the state.

Swimlane is a special process variable. It may be initialized at any moment of process execution. If the current state's swimlane is initialized, then only one executor receives task corresponding to the state.

If the current state's swimlane is not initialized, then the special mechanism is used to decide who can perform the task. This mechanism is called orgfunction. Orgfunction is defined by special kind Java class and parameters. This class on the basis of parameter values generates list of executors who can execute state. If executor is in this list, the associated with state task is shown as available in his/her tasklist. The first user, who execute state, initialize swimlane with his/her ID.

There is another way to initialize swimlane. User ID can be assigned to process variable with name of swimlane. In this case orgfunction is not needed and swimlane is initialized by variable setting.

Orgfunction must be defined using jPdl assignment delegation configuration with following format: `<Initializer Java class>(<parameter>,<parameter>,…)`

RUNA WFE distribution contains following sample implementations of orgfunction that can be used as point of reference:

- `ru.runa.af.organizationfunction.ExecutorByNameFunction`. Parameter is name of executor. The initializator returns user or group of users.
- `ru.runa.af.organizationfunction.DemoChiefFunction`. Parameter is user id. The initializator returns the chief of this user (demo specific class)

Parameters can be either string constants or process variable value. Variable values must be enclosed in special braces -- ${} (e.g. ${<variable name>}). In this case during execution orgfunction class receives corresponding variable value.
Examples:
`ru.runa.af.organizationfunction.DemoChiefFunction(${requester})` returns boss of requestor.
In this case `${requester}` is a swimlane/variable.
`ru.runa.af.organizationfunction.ExecutorByNameFunction(manager))` returns executor(actor or group[4]) with name `manager`. In this case `manager` is constant.

Swimlanes for OverTime demo process

Business process has two swimlanes:

- manager
- staff

Swimlanes initialization description:

Swimlane	**Initialization description**
manager	Person, who started the process
staff	Person, whom choose manager in the Start form

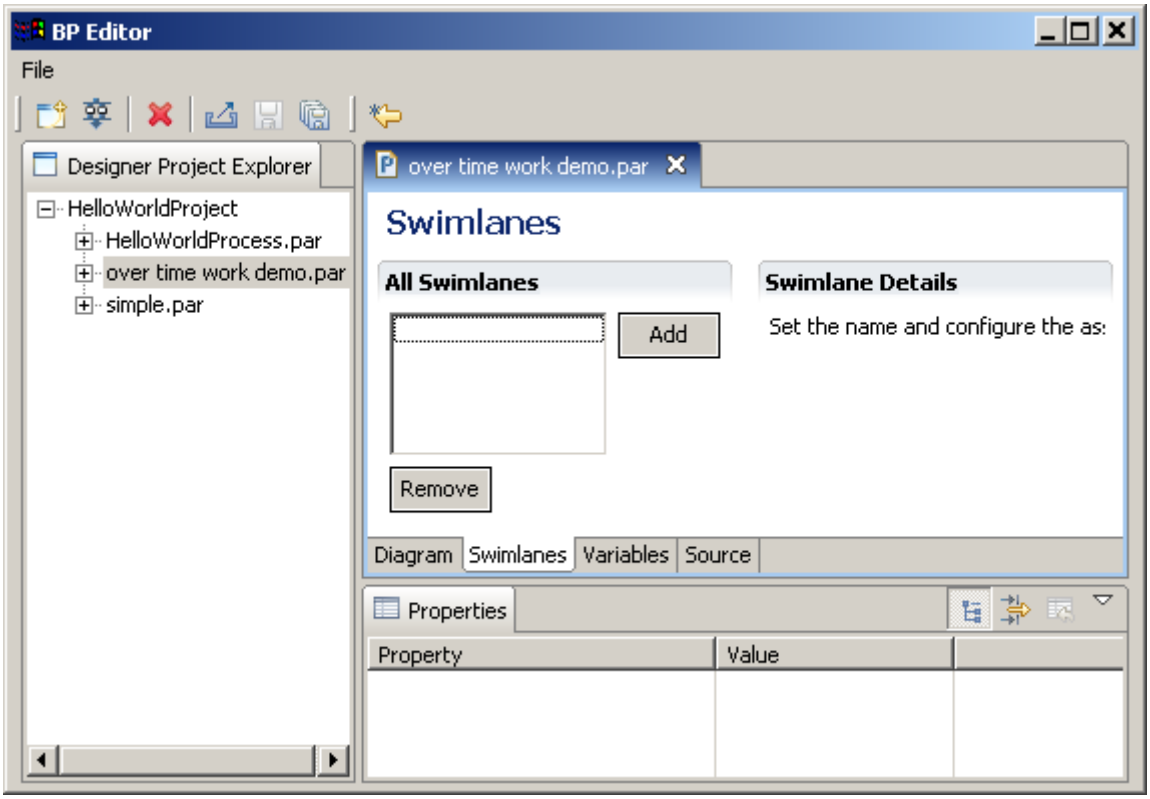
State – swimlane mappings:

State	**Associated swimlane**
Offer an overtime work	manager
Make a decision	staff
Notify for declining	manager
Notify for acceptance	manager

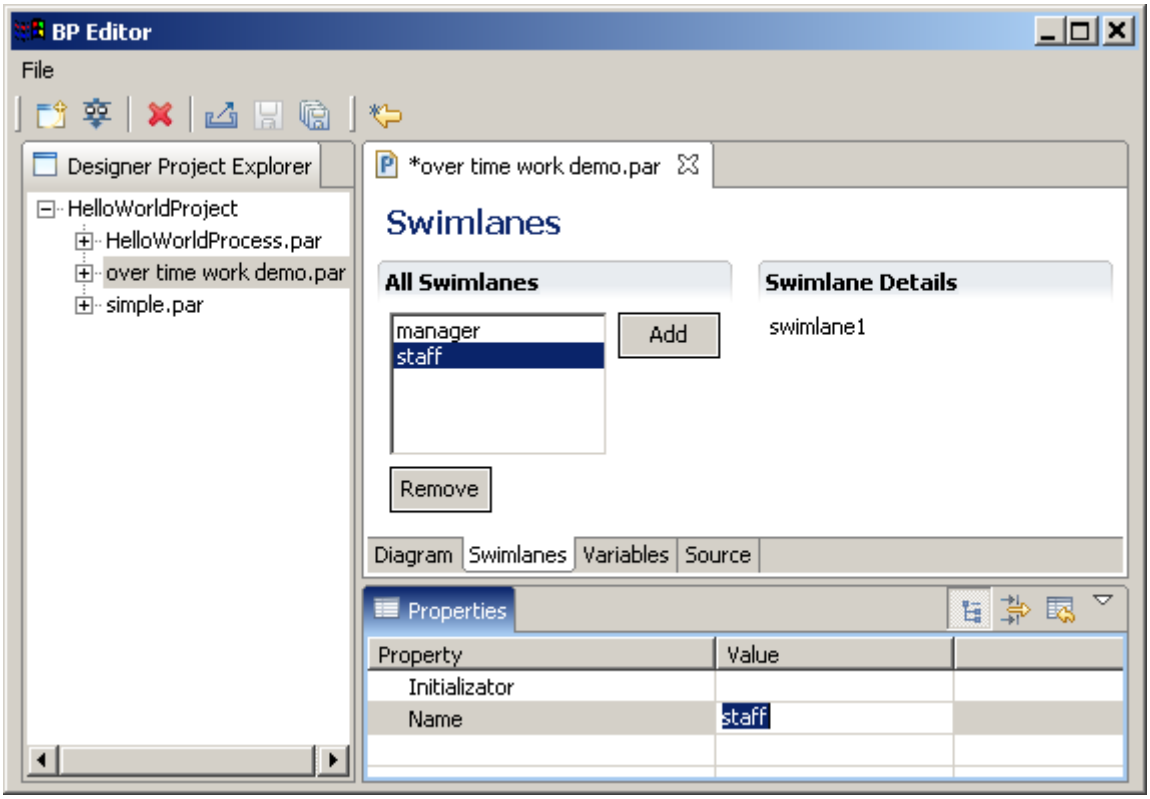
4 If orgfunction returns group all group members can be assigned to swimlane.

Swimlanes creation

Click on “Swimlanes” tab.

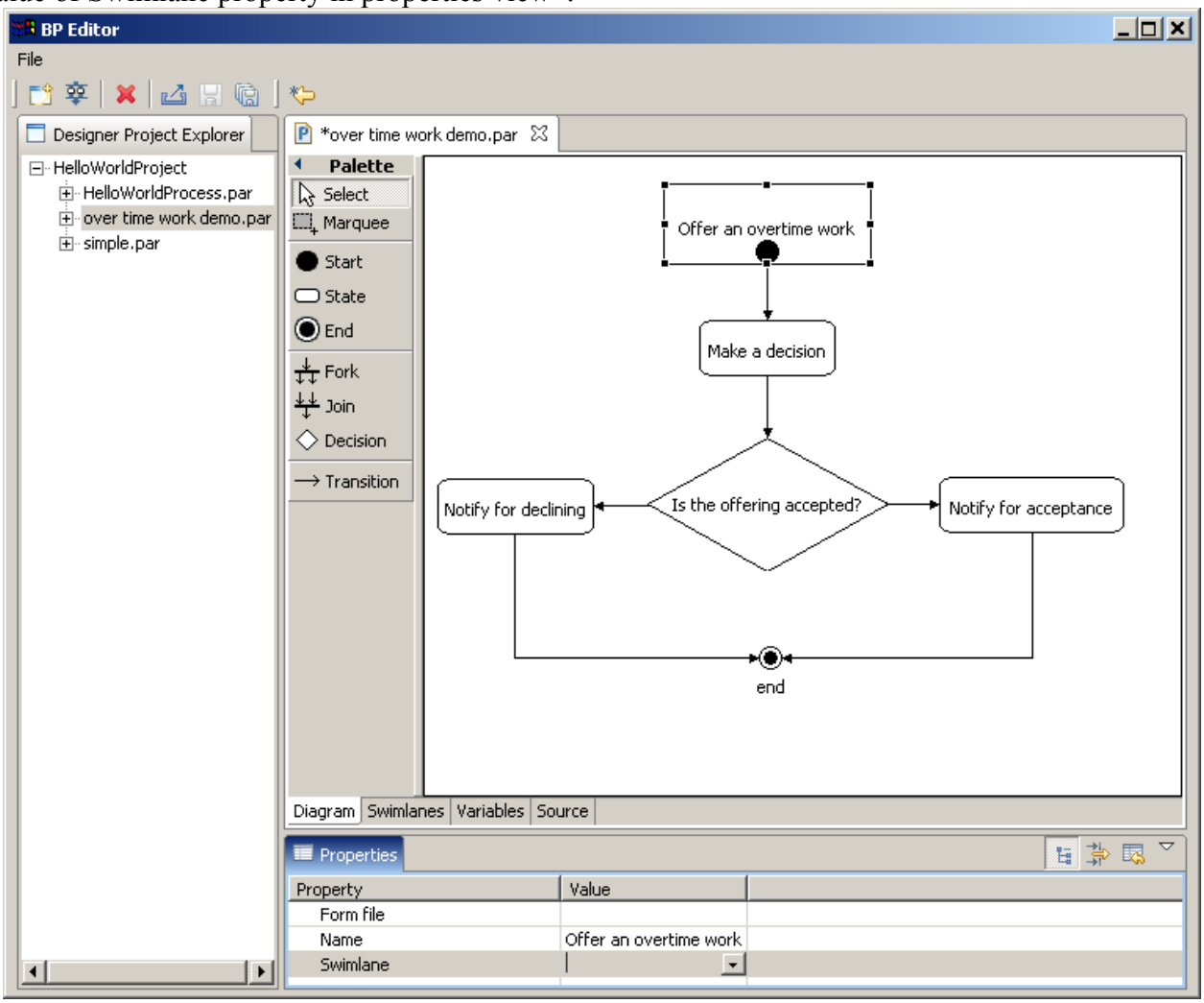


Click “Add” button. Select appeared “swimlane1” swimlane and rename it to manager in properties view. Then add “staff” swimlane:



Open “Diagram” tab, select “Offer an overtime work” StartState, then select “manager”in

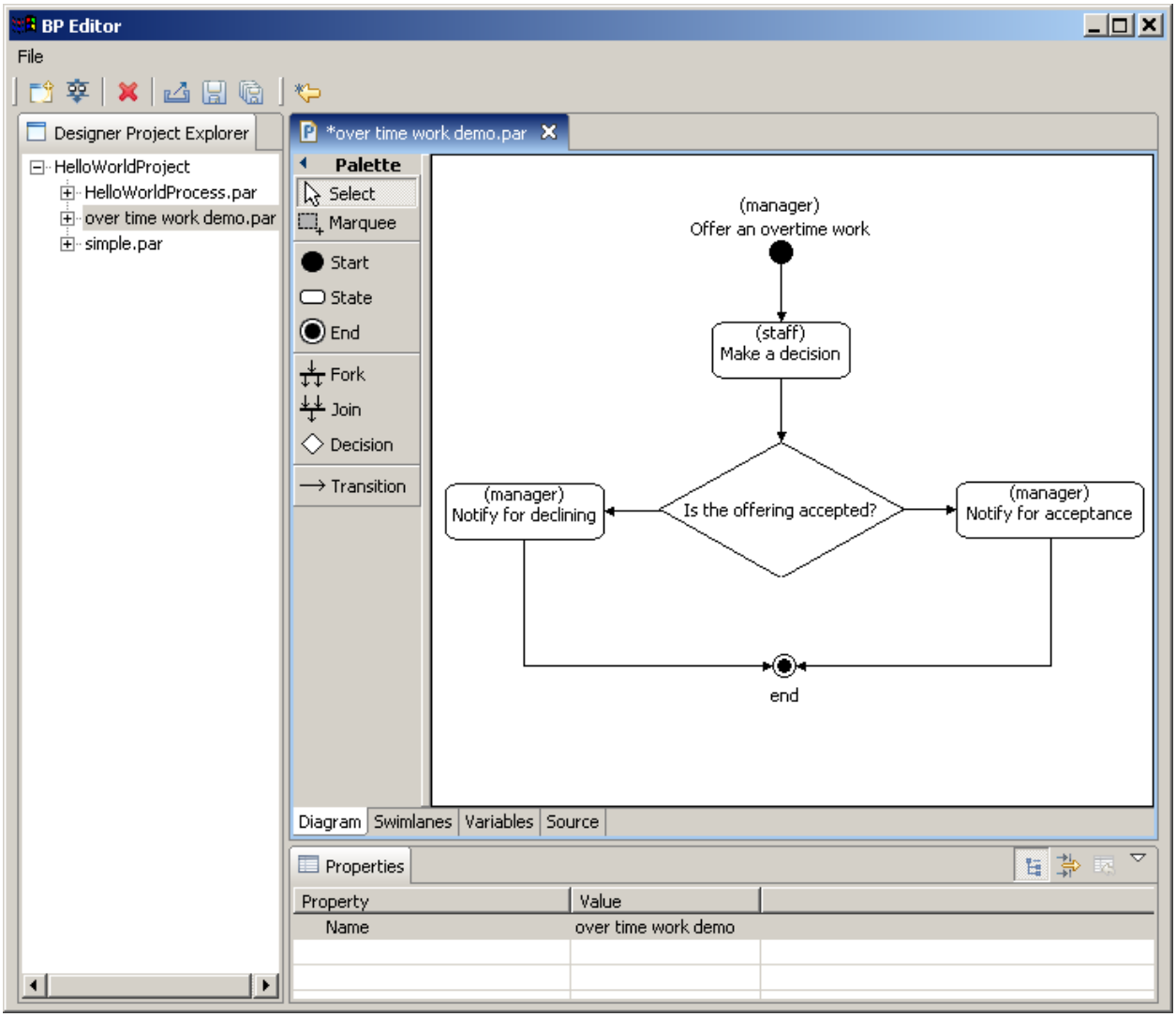
the value of Swimlane property in properties view[5].



Select “Make a decision” State, then select *staff* in the value of Swimlane property in properties view.

Then associate “Notify for declining” and “Notify for acceptance” states with *manager* swimlane. You'll see swimlane name in round brackets at upper part of each state:

5 The Initializator field is empty in both cases, because *manager* swimlane is initialized as swimlane, associated with StartState, *staff* swimlane will be initialized in form as process variable.



Creating and linking variables with states

Variables description and variables initialization

The following variables are used in OverTime demo process:

Variable	Type
since	DateTime
till	DateTime
reason	String
comment	Text
staff person decision	Boolean
staff person comment	Text

Variables

- since
- till

- reason
- comment
- staff[6]

are initialized in StartState “Offer an overtime work”.

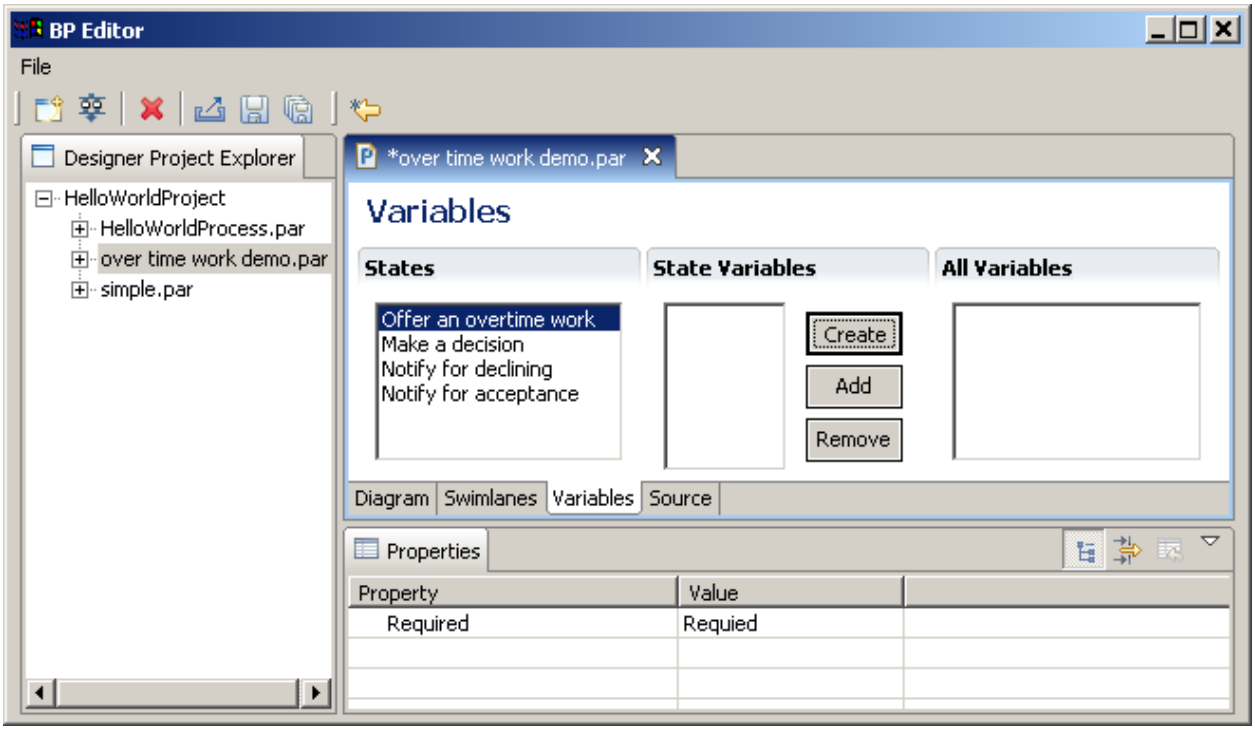
Variables

- staff person decision
- staff person comment

are initialized in State “Make a decision”

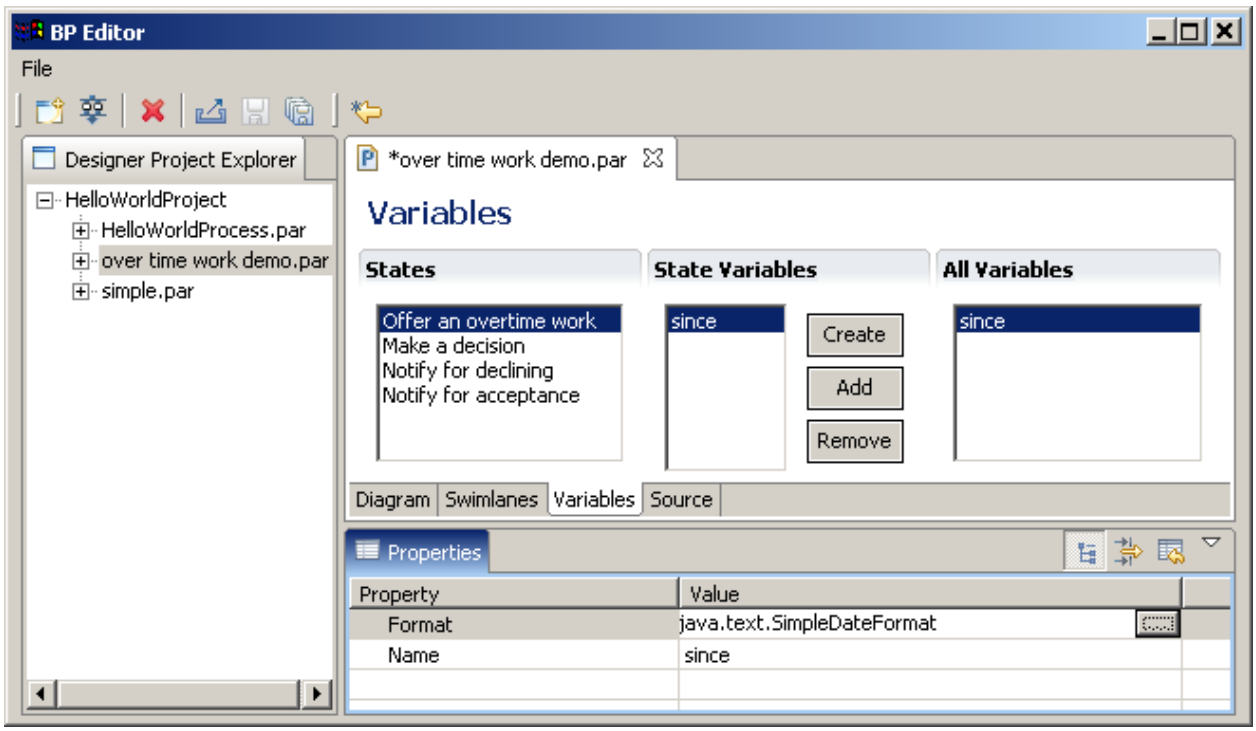
Variables creation

Click on “Variables” tab. Select “Offer an overtime work” state:

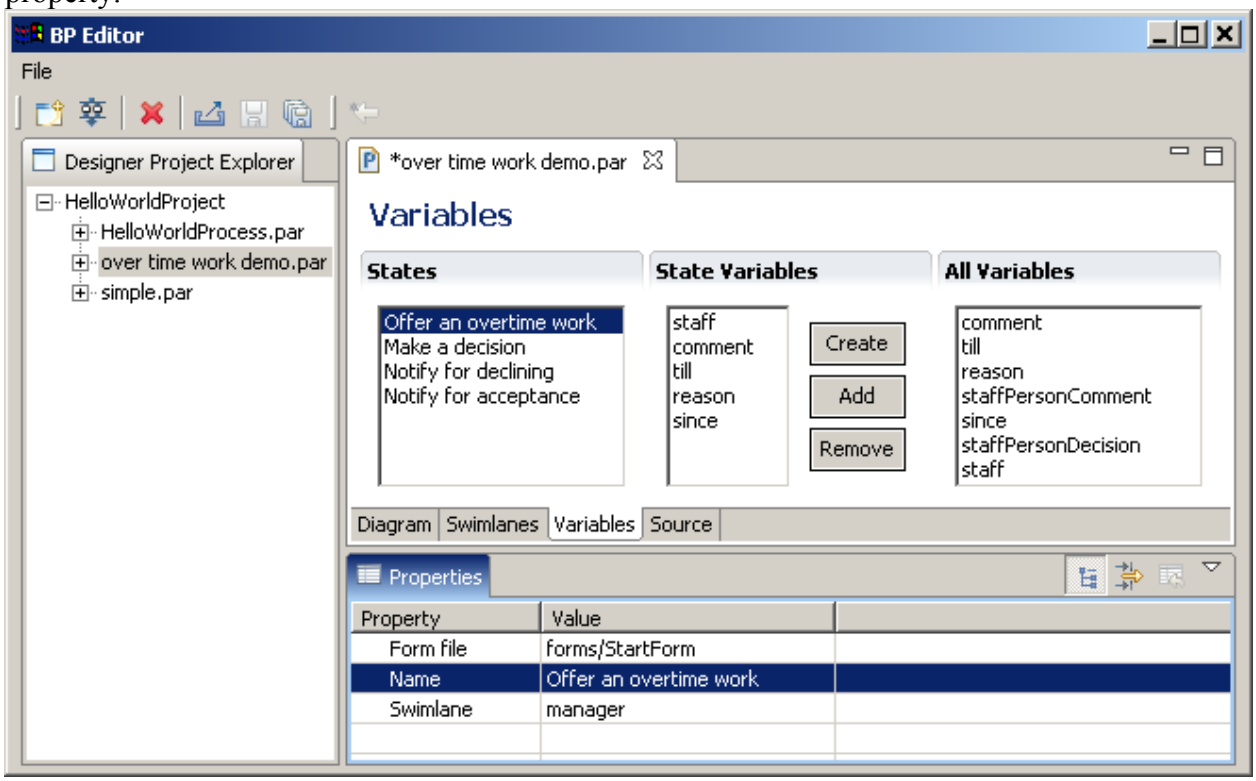


Click “Create” button, rename default variable name to “since”, choose DateTime format in the Format field:

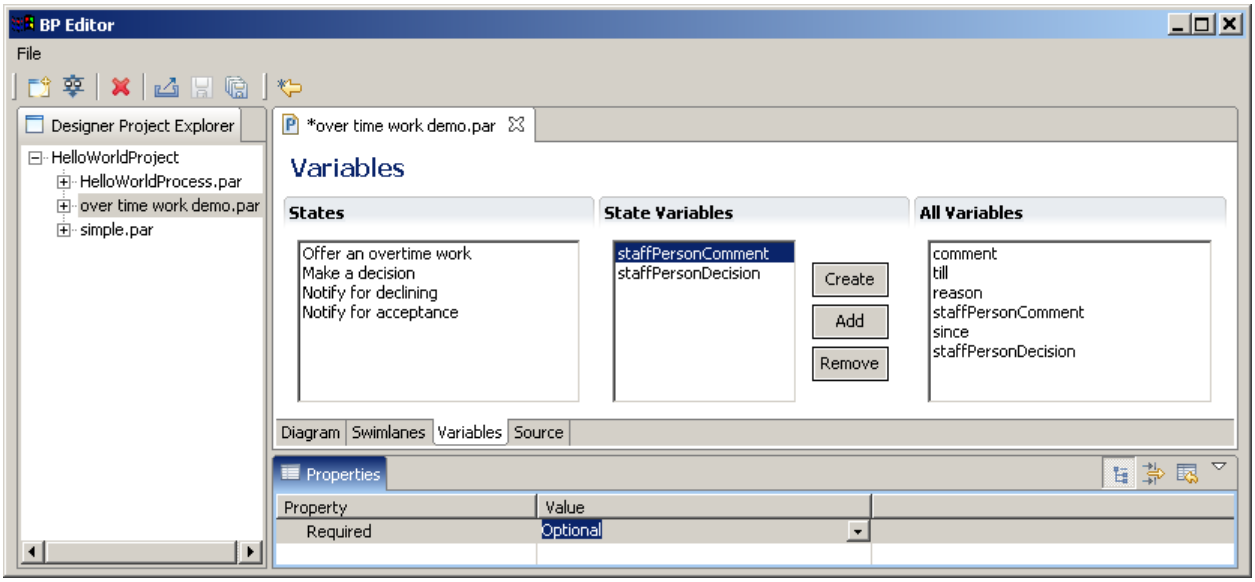
6 Staff is swimlane but it is initalized as regular variable.



Create the other variables, initialized in StartState, set for “comment” field “optional” property:



Click “Make a decision” state. Create variables “staffPersonDecision” and “staffPersonComment”. Set staffPersonComment to “optional”:



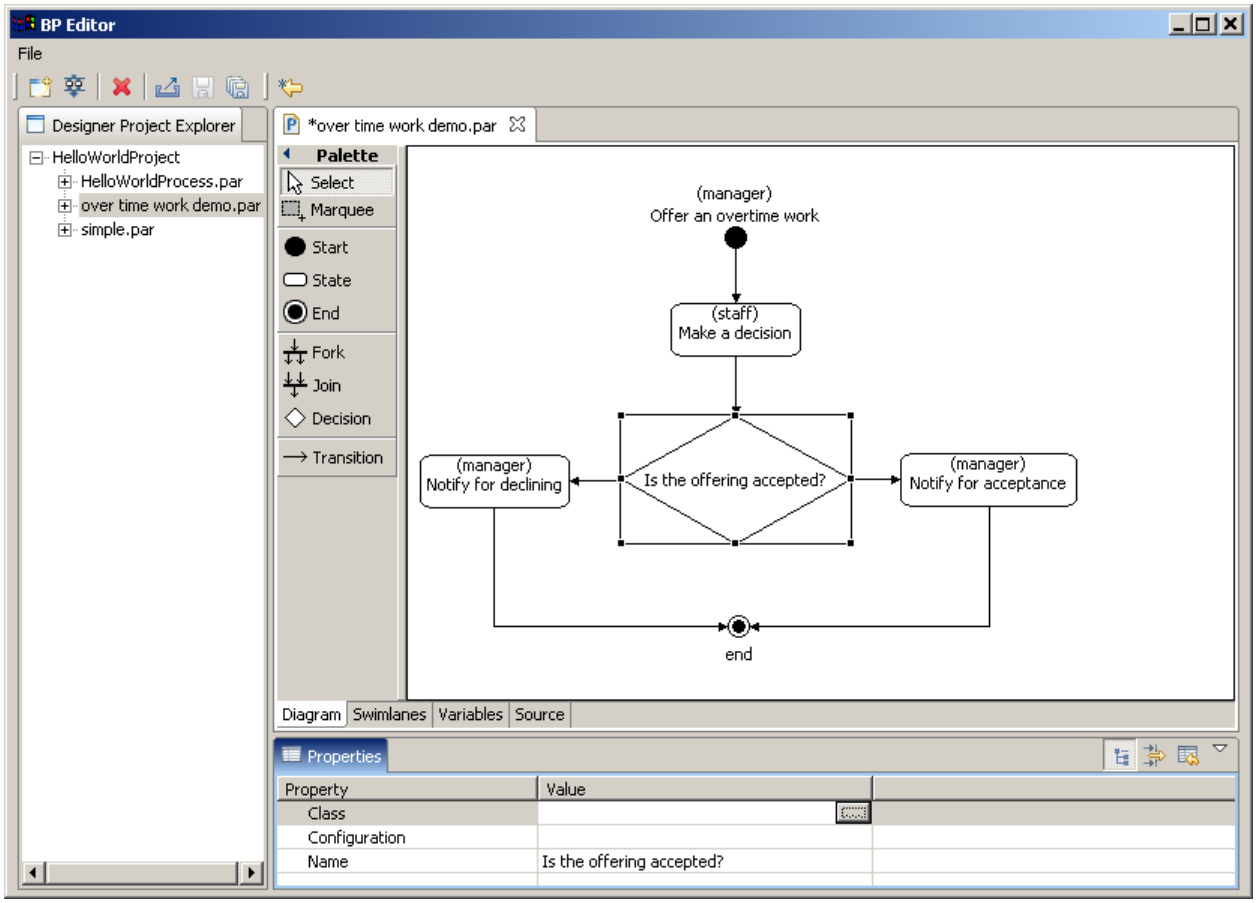
Decision formula creation

Decision formulas description

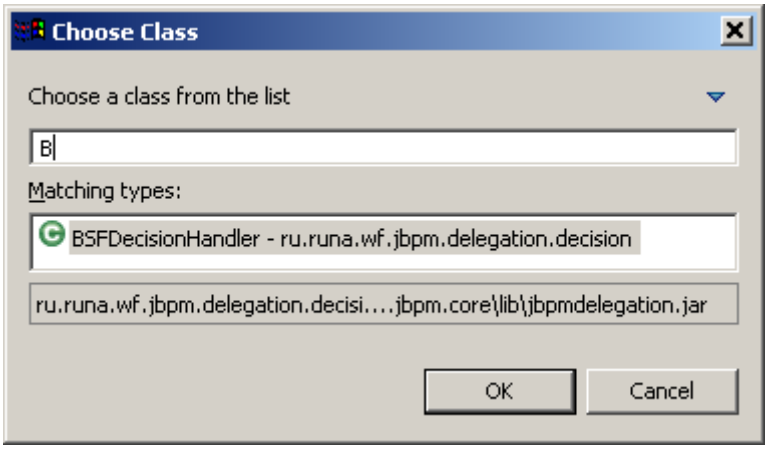
Default class for formula processing (`BSFDecisionHandler`) choose transition on the base of BeanShell script in the "Configuration" field (see. http://www.beanshell.org/intro.html). It is possible to use process variables in the script body. The type of script result value must be String. Result value must be equal to one of leaving transition names.

Decision formula creation in RUNA GPD

Click "diagram" tab. Select "Is the offering accepted?" decision, then click on right hand end of property class:

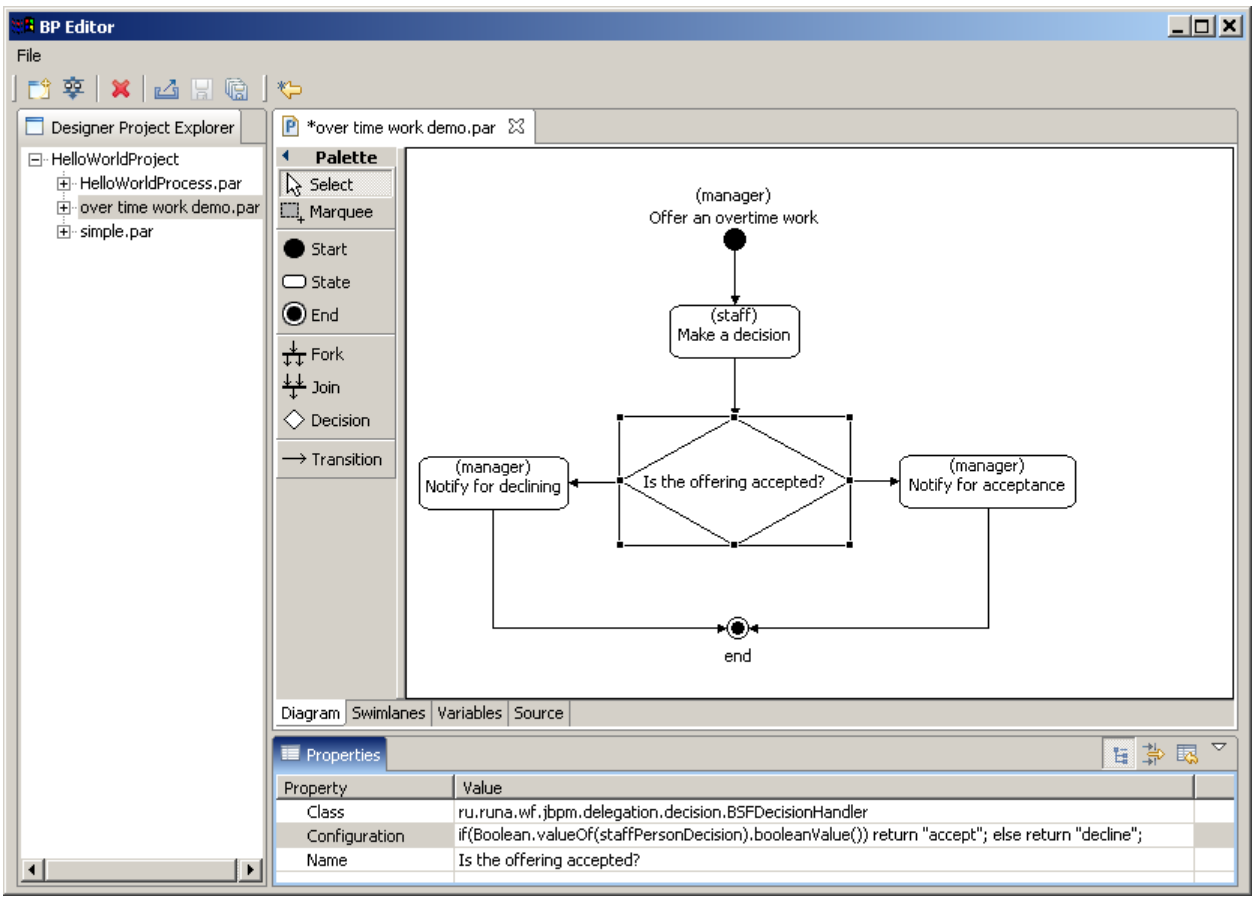


In the appeared form enter “b” in the top field, then choose class `BSFDecisionHandler`

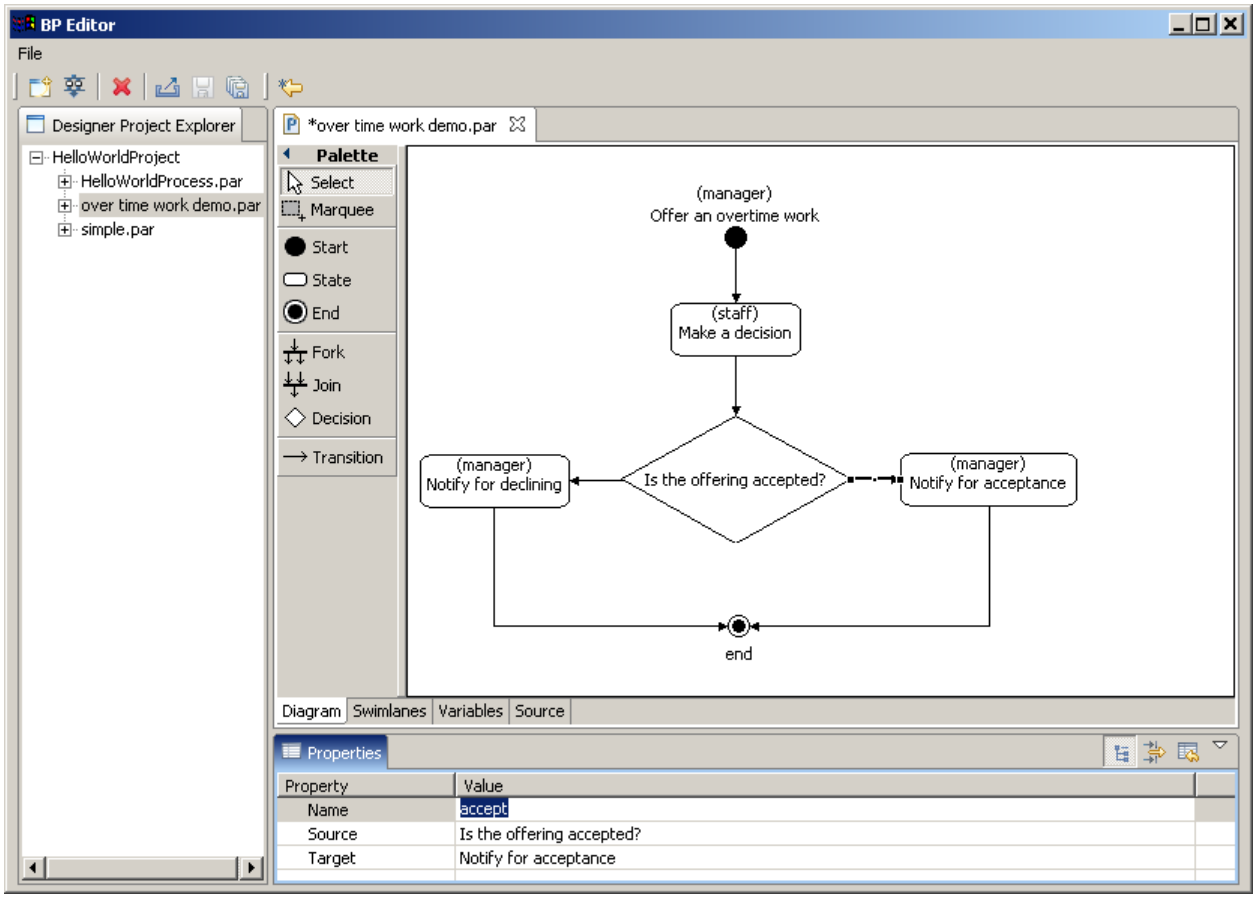


Then click “OK”.
In the “Configuration”field enter following BeanShell script :

```
if(Boolean.valueOf(staffPersonDecision).booleanValue())
return "accept"; else return "decline";
```



Click on the left leaving decision transition, rename it to “decline”. Click on the right leaving decision transition, rename it to “accept”.



Graphical forms creation

RUNA WFE forms description

State can be used for human and/or bot interaction[7].

If state is used for human interaction, then a form must be associated with it. Every form is described in a .form file. The form parsing mechanism uses HTML with additional tags `<customtag>`. These tags are used to display process variable value in the form.

The `< customtag>` tag have the following attributes:

- `var` – process variable name
- `delegation` – name of Java class, used for the variable value rendering (must implements `ru.runa.wf.web.html.VarTag` interface)

The following delegation-classes are used in "customtag" tags in forms of OverTimeWorkDemo process:

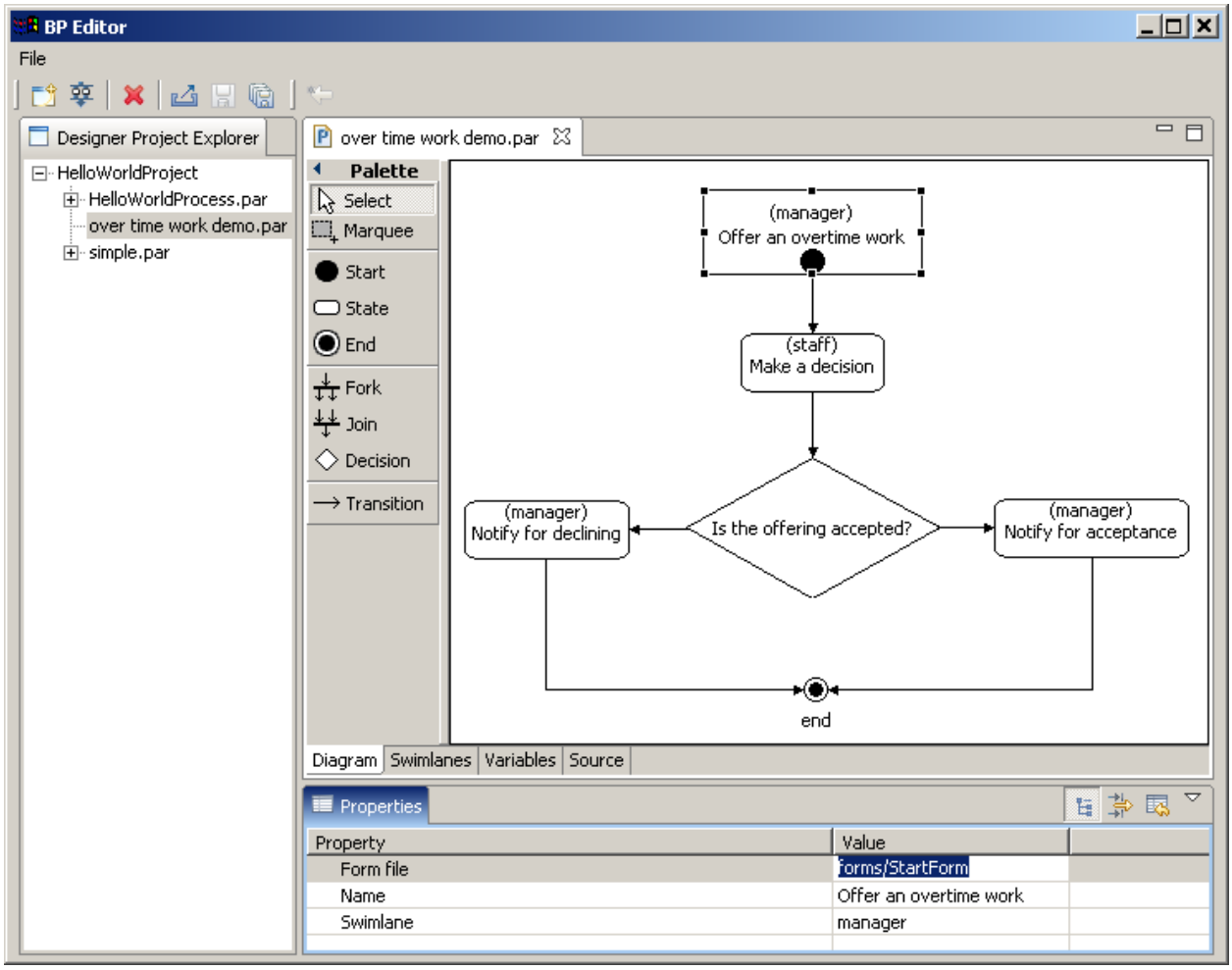
Class	***Description***
GroupMembersComboboxVarTag	Generates choice of group members. The group name equals the «var» parameter of the tag. Returns selected actor ID (In OverTimeWorkDemo process "staff"group is used)
DateTimeInputVarTag	Generates field for DateTime input
ActorFullNameDisplayVarTag	Generates Actor name. «var» parameter of the tag refers to variable, containing actor ID.

7 Bot is a special kind of workflow application for automation/integration purpose.

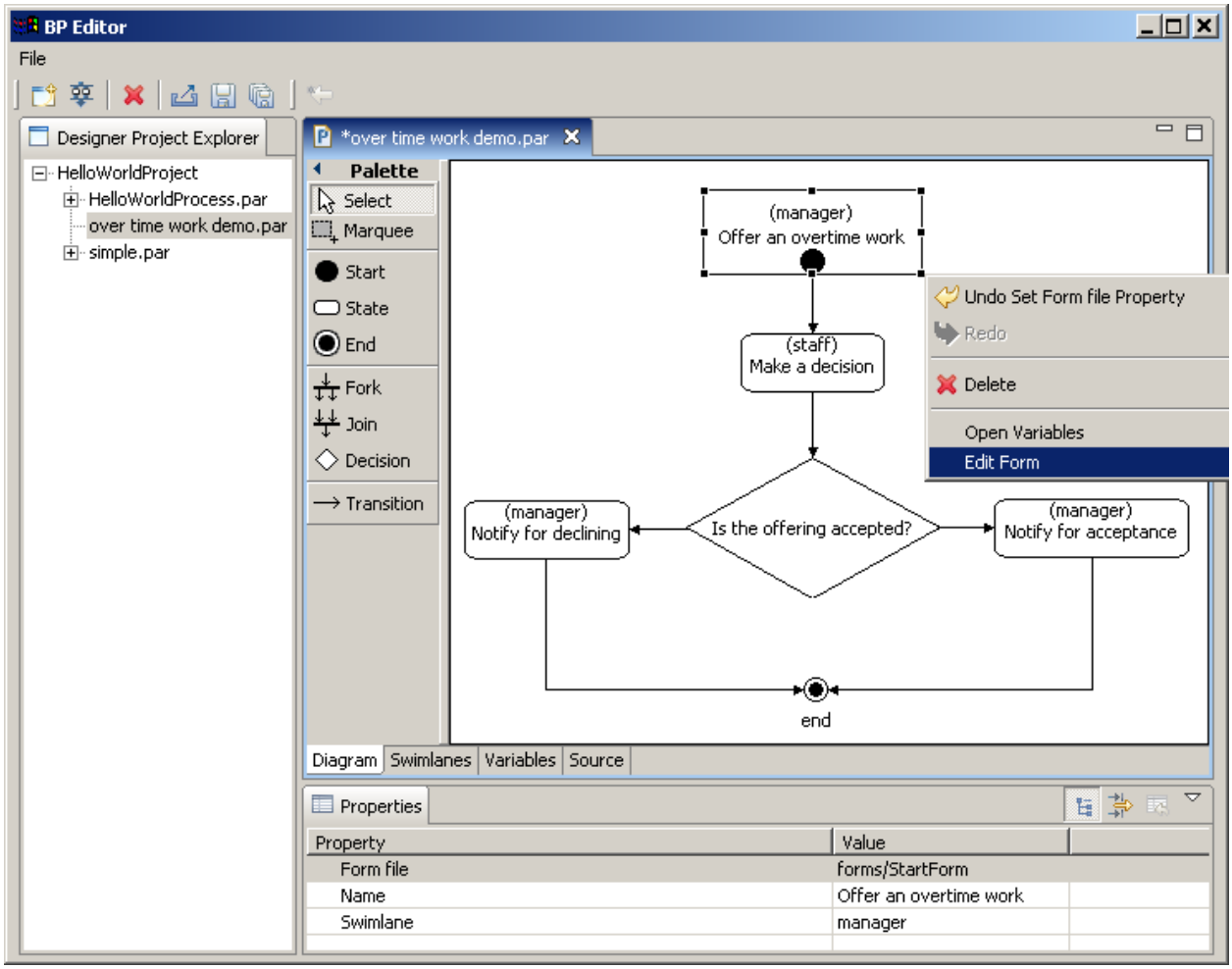
Class	*Description*
DateTimeValueDisplayVarTag	Generates field for DateTime, showing in read only mode.
VariableValueDisplayVarTag	Generates field for String, showing in read only mode.

Creating forms with the help of RUNA GPD

Click on “Offer an overtime work” state, then enter “forms/StartForm” in the “Form file” field.



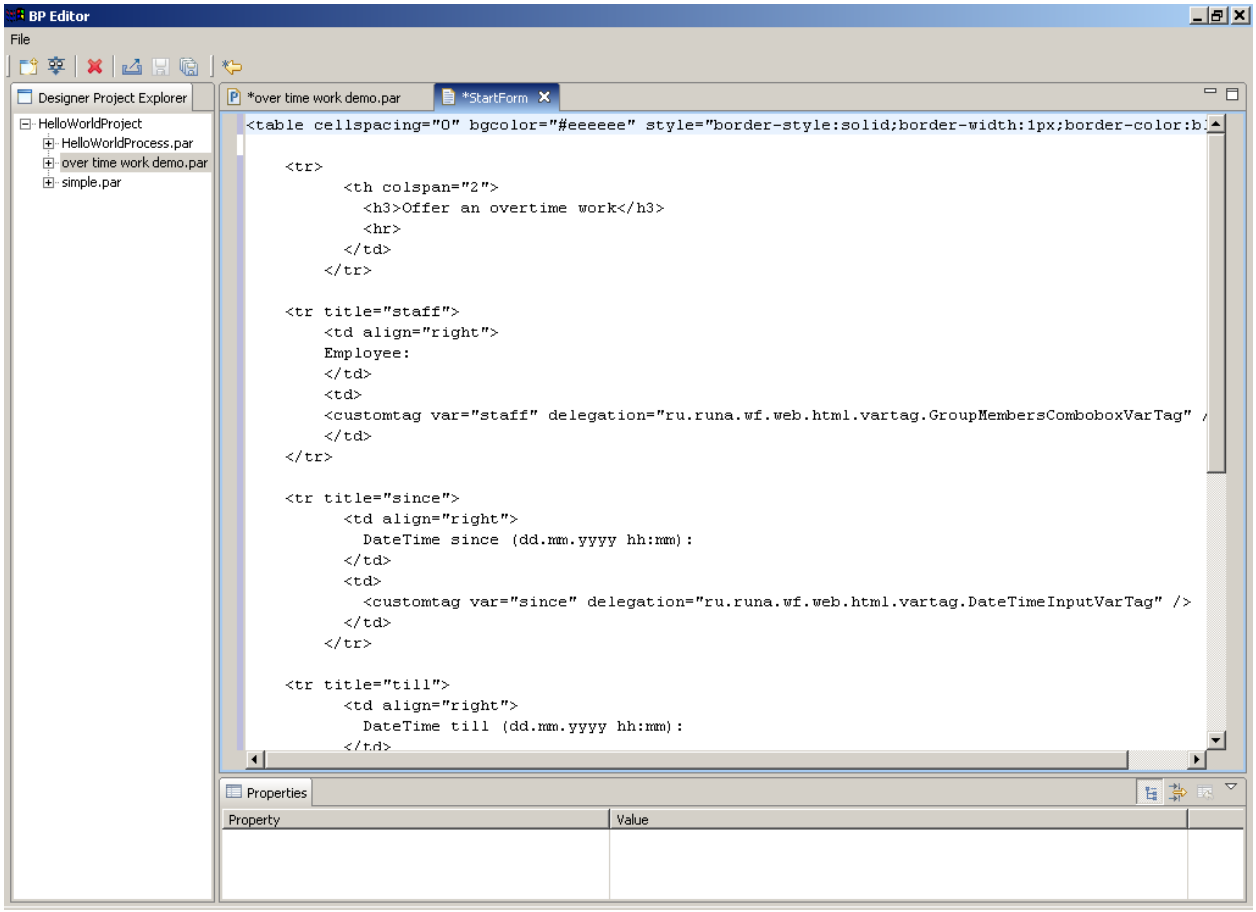
Click on “Offer an overtime work” state by right mouse button, then choose “Edit form” menu item.



Enter in the appeared window the following:

```
<table cellspacing="0" bgcolor="#eeeeee" style="border-style:solid;border-width:1px;border-color:black;">
  <tr>
  <th colspan="2">
          <h3>Offer an overtime work</h3>
          <hr>
  </th>
  </tr>

  <tr title="staff">
  <td align="right">
          Employee:
  </td>
  <td>
          <customtag var="staff" delegation="ru.runa.wf.web.html.vartag.GroupMembersComboboxVarTag" />
  </td>
  </tr>

  <tr title="since">
  <td align="right">
          DateTime since (dd.mm.yyyy hh:mm):
  </td>
  <td>
          <customtag var="since" delegation="ru.runa.wf.web.html.vartag.DateTimeInputVarTag" />
  </td>
  </tr>

  <tr title="till">
  <td align="right">
          DateTime till (dd.mm.yyyy hh:mm):
  </td>
  <td>
          <customtag var="till" delegation="ru.runa.wf.web.html.vartag.DateTimeInputVarTag" />
  </td>
  </tr>

  <tr title="reason">
  <td align="right">
          Reason :
  </td>
```

```
        <td>
        <INPUT TYPE="text" NAME="reason">
        </td>
        </tr>

        <tr title="comment">
        <td align="right">
                Comment :
        </td>
        <td>
                <textarea name="comment"></textarea>
        </td>
        </tr>
      </table>
```



Click “Save” icon on the toolbar.

Create forms for all other process states:

Process state	***Form name***
Offer an overtime work	forms/StartForm
Make a decision	forms/MakeaDecision
Notify for declining	forms/NotifyForDeclining
Notify for acceptance	forms/NotifyForAcceptance

The content of “MakeaDecision” form is the following:

```
<table cellspacing="0" bgcolor="#eeeeee" style="border-style:solid;border-width:1px;border-color:black;">

        <tr>
        <th nowrap="true" colspan="2">
```

```
                <h3> Accept or decline the offering of over time work</h3>
                <hr>
        </th>
        </tr>

        <tr title="manager">
        <td nowrap="true" align="right">
                manager:
        </td>
        <td nowrap="true" >
                <customtag var="manager" delegation="ru.runa.wf.web.html.vartag.ActorFullNameDisplayVarTag" />
        </td>
        </tr>

        <tr title="staff">
        <td nowrap="true" align="right">
                staff persone (employee):
        </td>
        <td nowrap="true" >
                <customtag var="staff" delegation="ru.runa.wf.web.html.vartag.ActorFullNameDisplayVarTag" />
        </td>
        </tr>

        <tr title="since">
        <td nowrap="true" align="right">
                DateTime since:
        </td>
        <td nowrap="true">
                <customtag var="since" delegation="ru.runa.wf.web.html.vartag.DateTimeValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="till">
        <td nowrap="true" align="right">
                DateTime till:
        </td>
        <td nowrap="true">
                <customtag var="till" delegation="ru.runa.wf.web.html.vartag.DateTimeValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="reason">
        <td nowrap="true" align="right">
                Reason:
        </td>
        <td nowrap="true">
                <customtag var="reason" delegation="ru.runa.wf.web.html.vartag.VariableValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="comment">
        <td nowrap="true" align="right">
                Comment:
        </td>
        <td nowrap="true">
                <customtag var="comment" delegation="ru.runa.wf.web.html.vartag.VariableValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="staff person comment">
        <td nowrap="true" align="right">
                staff person comment:
        </td>
        <td nowrap="true">
                <textarea name="staff person comment">
                <customtag var="staff person comment"
                   delegation="ru.runa.wf.web.html.vartag.VariableValueDisplayVarTag" />
                </textarea>
        </td>
        </tr>

        <tr>
        <td nowrap="true" align="right" colspan="2">
                <hr>
                <input type="radio" name="staffPersonDecision" value="true" checked/> Accept
                <input type="radio" name="staffPersonDecision" value="false"/> Decline
        </tr>
</table>
```

The content of “NotifyForDeclining” form is the following:

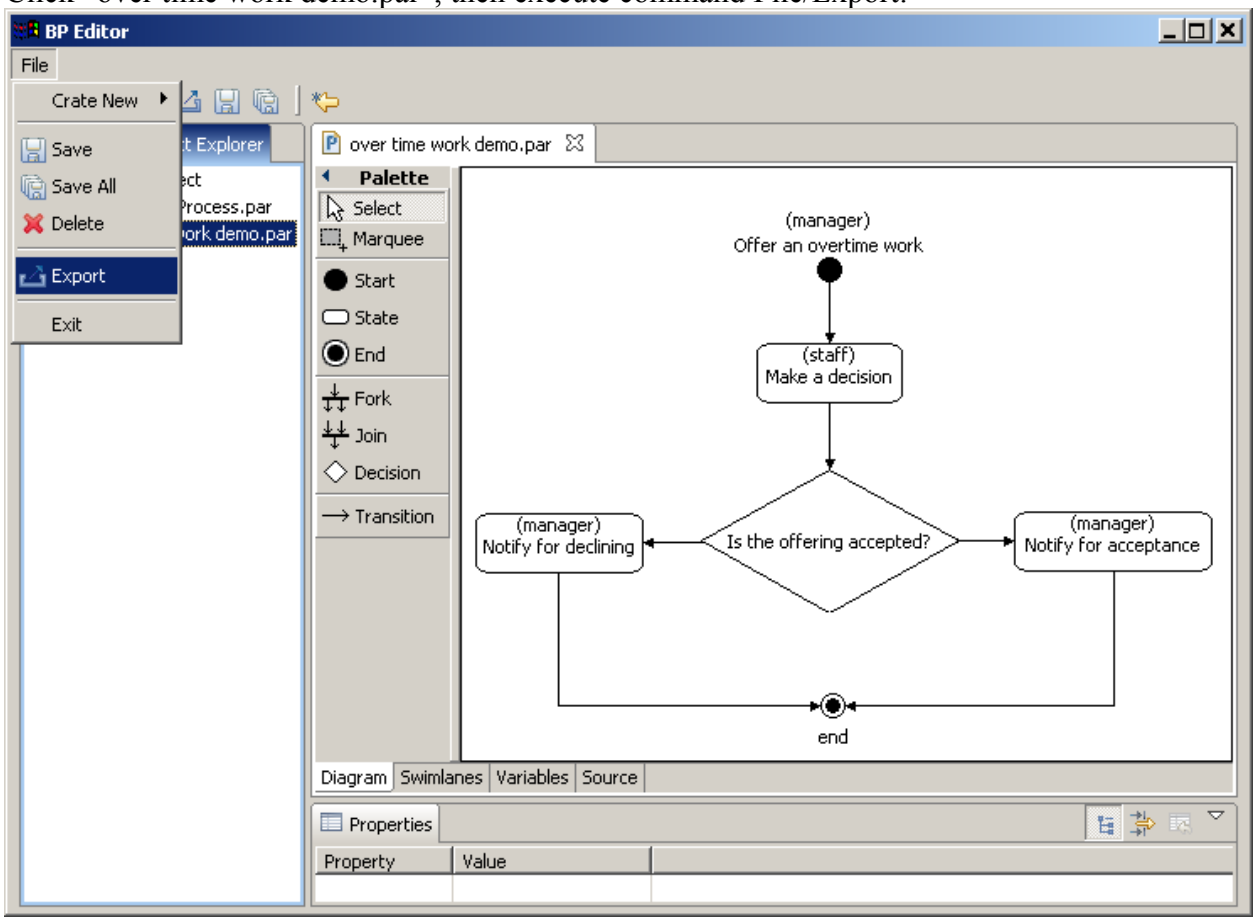
```
<table cellspacing="0" bgcolor="#eeeeee" style="border-style:solid;border-width:1px;border-color:black;">

        <tr>
        <th nowrap="true" colspan="2">
                <h3>OverTime work - declined</h3>
                <hr>
        </th>
        </tr>
```

```
        <tr title="manager">
        <td nowrap="true" align="right">
                manager:
        </td>
        <td nowrap="true" >
                <customtag var="manager" delegation="ru.runa.wf.web.html.vartag.ActorFullNameDisplayVarTag" />
        </td>
        </tr>

        <tr title="staff">
        <td nowrap="true" align="right">
                staff persone (employee):
        </td>
        <td nowrap="true" >
                <customtag var="staff" delegation="ru.runa.wf.web.html.vartag.ActorFullNameDisplayVarTag" />
        </td>
        </tr>

        <tr title="since">
        <td nowrap="true" align="right">
                DateTime since:
        </td>
        <td nowrap="true">
                <customtag var="since" delegation="ru.runa.wf.web.html.vartag.DateTimeValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="till">
        <td nowrap="true" align="right">
                DateTime till:
        </td>
        <td nowrap="true">
                <customtag var="till" delegation="ru.runa.wf.web.html.vartag.DateTimeValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="reason">
        <td nowrap="true" align="right">
                Reason:
        </td>
        <td nowrap="true">
                <customtag var="reason" delegation="ru.runa.wf.web.html.vartag.VariableValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="comment">
        <td nowrap="true" align="right">
                Comment:
        </td>
        <td nowrap="true">
                <customtag var="comment" delegation="ru.runa.wf.web.html.vartag.VariableValueDisplayVarTag" />
        </td>
        </tr>

        <tr title="staff person comment">
        <td nowrap="true" align="right">
                staff person comment:
        </td>
        <td nowrap="true">
                <customtag                    var="staff                    person                    comment"
delegation="ru.runa.wf.web.html.vartag.VariableValueDisplayVarTag" />
        </td>
        </tr>

</table>
```

Form “NotifyForAcceptance” differs from the form “NotifyForDeclining” only by the header: “OverTime work – accepted” instead of “OverTime work – declined”.

Process definition archive file creation

Click “over time work demo.par”, then execute command File/Export:

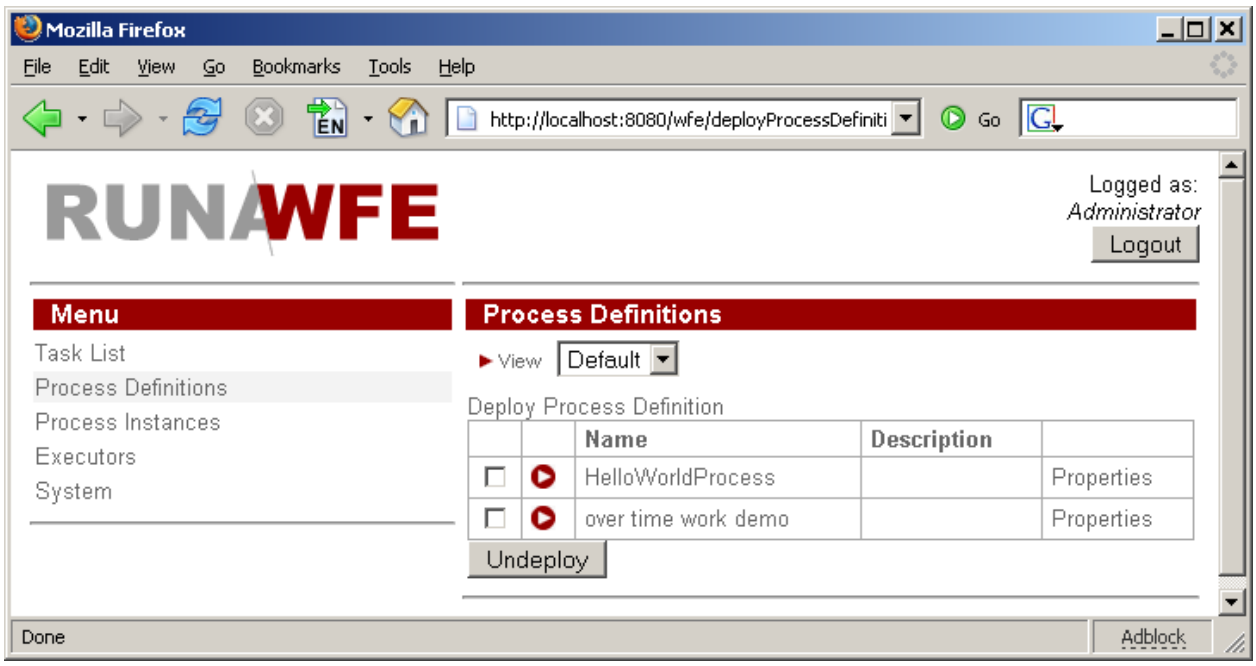


In the appeared form click on over time work demo, then enter “C:\Temp\OverTimeWorkDemo.par” in “Destination file” field and click “finish” button. The OverTimeWorkDemo process file will be generated.

Deployment in JBoss jBPM engine via RUNA WFE environment

Login into RUNA WFE web interface as Administrator (The default Administrators password is wf, see RUNA WFE 2.0 manual for details).
Go to “process definition” menu.
Press deploy process definition[8].
Press browse button to select process definition archive.
Press Ok button.
After deployment process appears in the process list:

8*Note:In order to deploy a process you must have Process Definition permission on System (can be granted via system menu).*



Running OverTime Work Demo process

Groups and actors creation

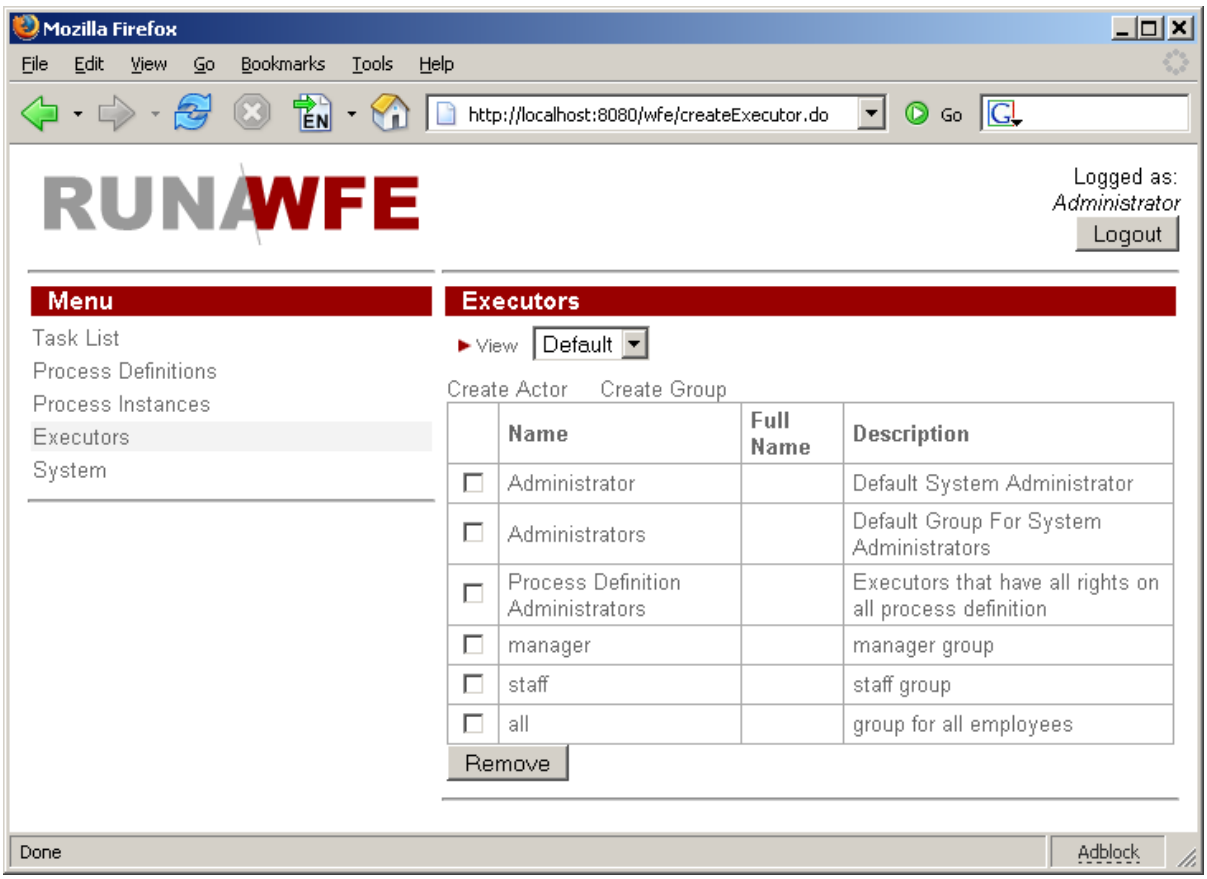
Before running the process create actors and groups of actors:

Group	Group members
manager	nero
staff	attila
all	nero attila

Click on the menu item “Executors”

Using “Create Group” command create the following groups:

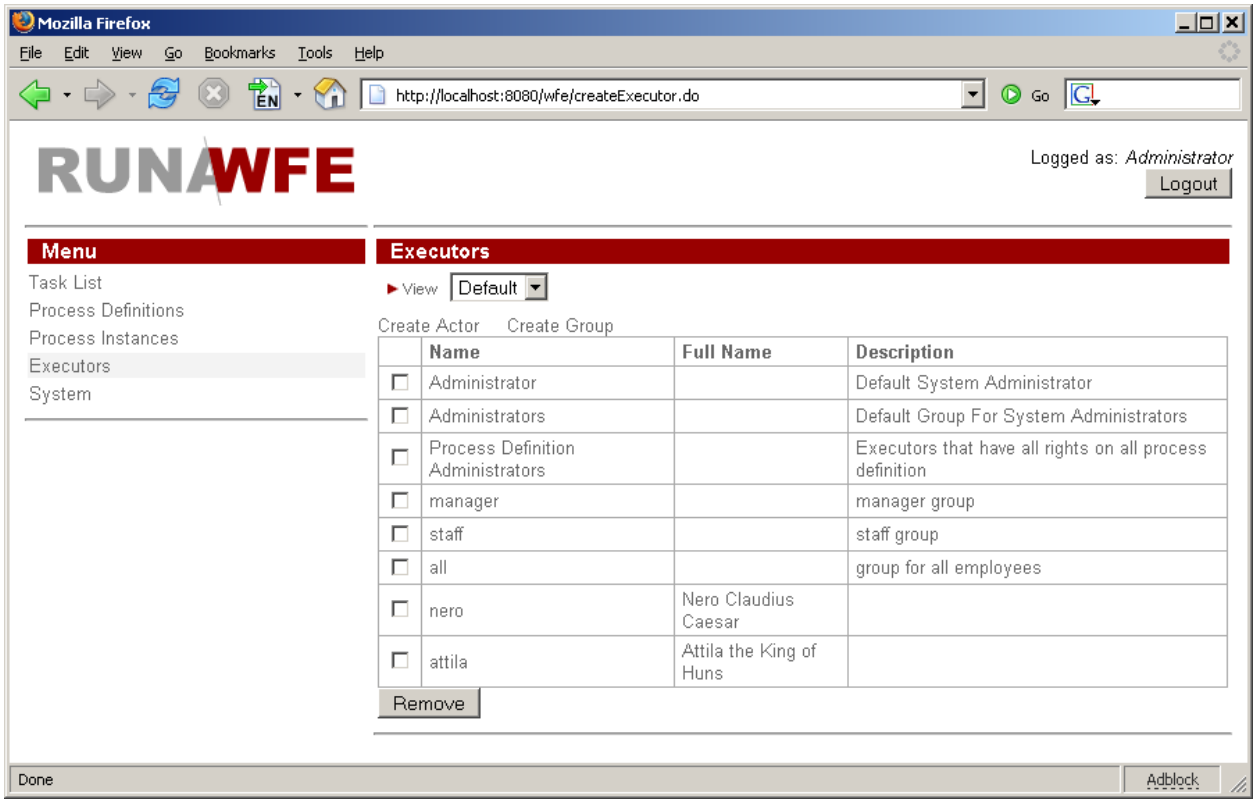
- manager
- staff
- all



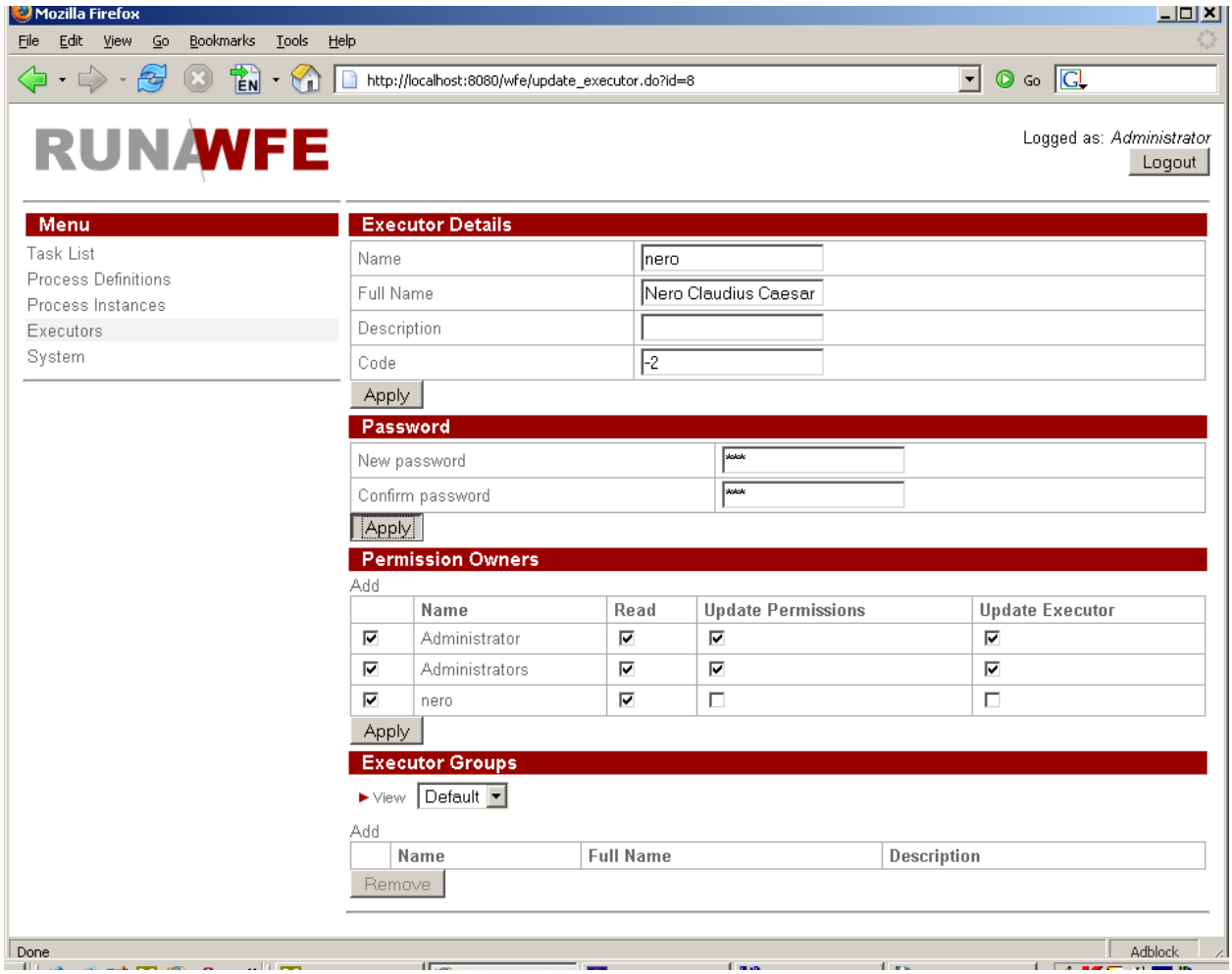
Using “Create Actor” command create the following actors:

- nero
- attila

You’ll se the following:



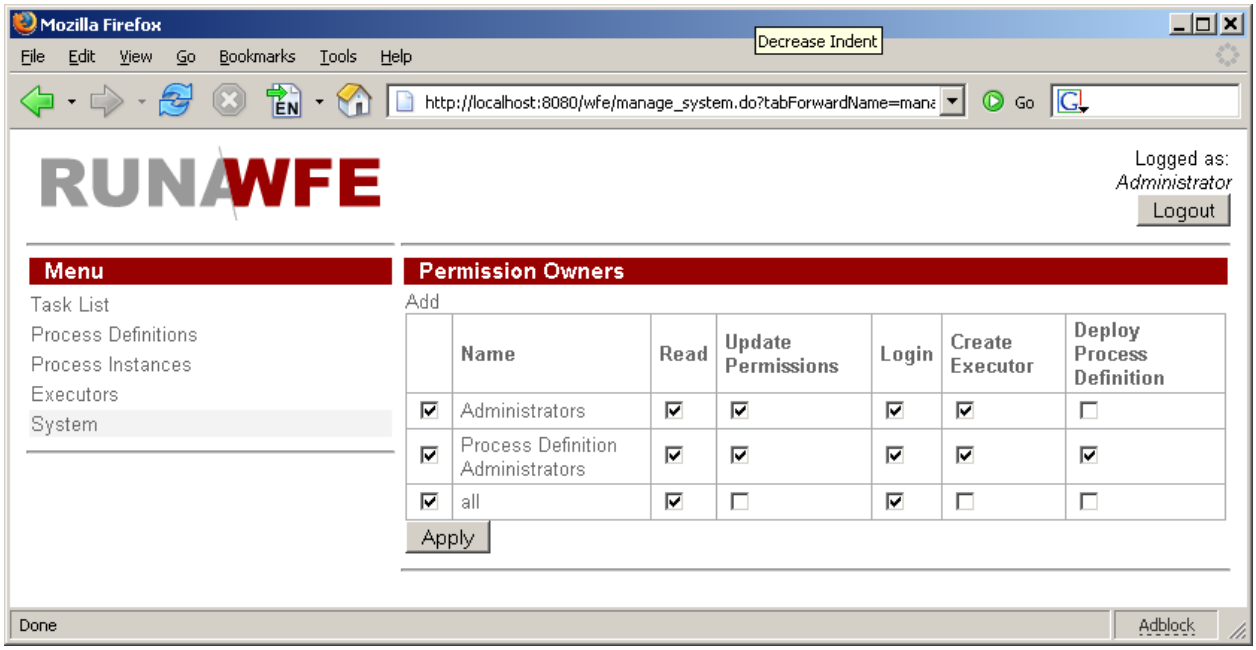
Set password 123 for users nero and attila



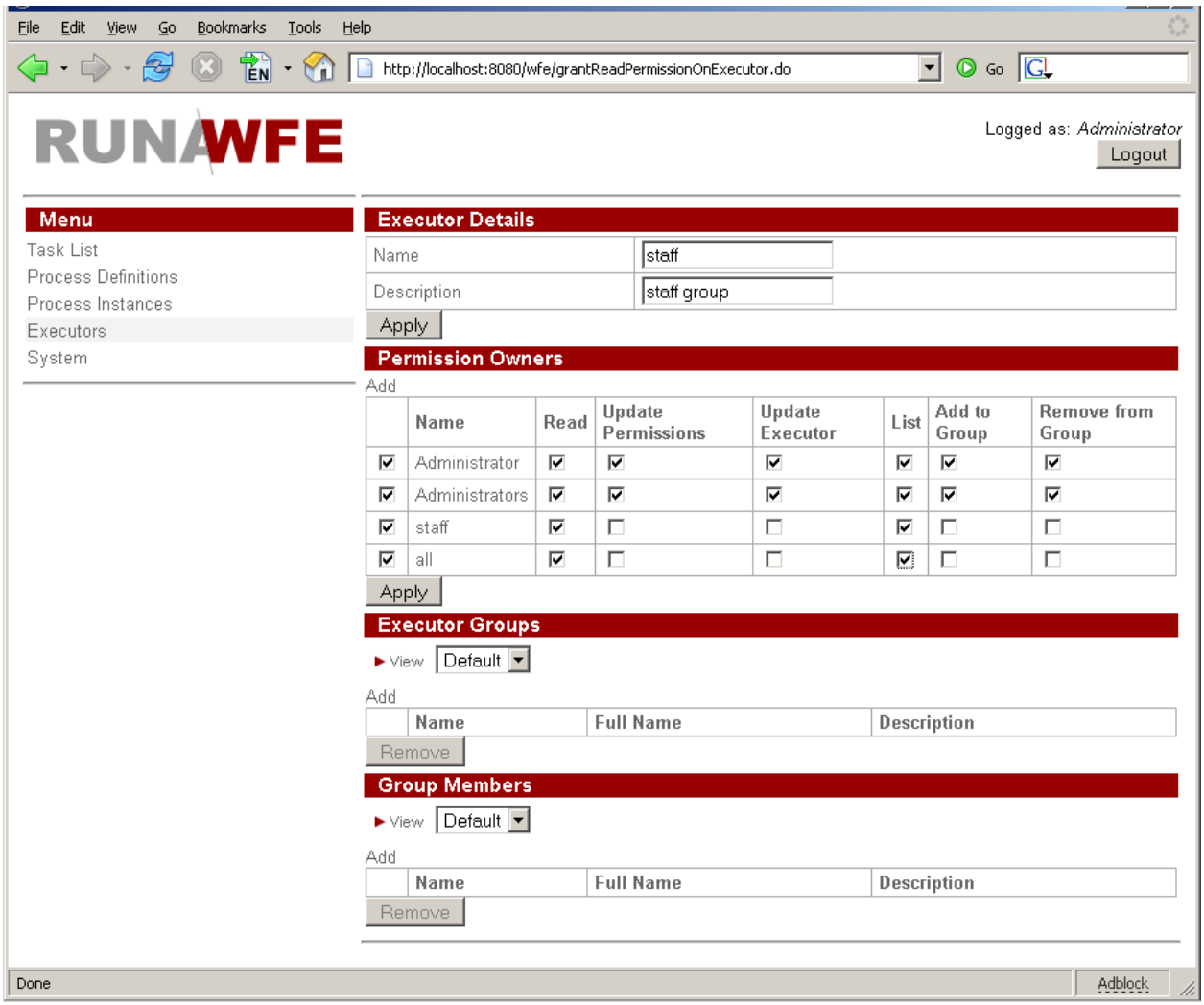
Add actors in groups

Groups	Group members
manager	nero
staff	attila
all	nero attila

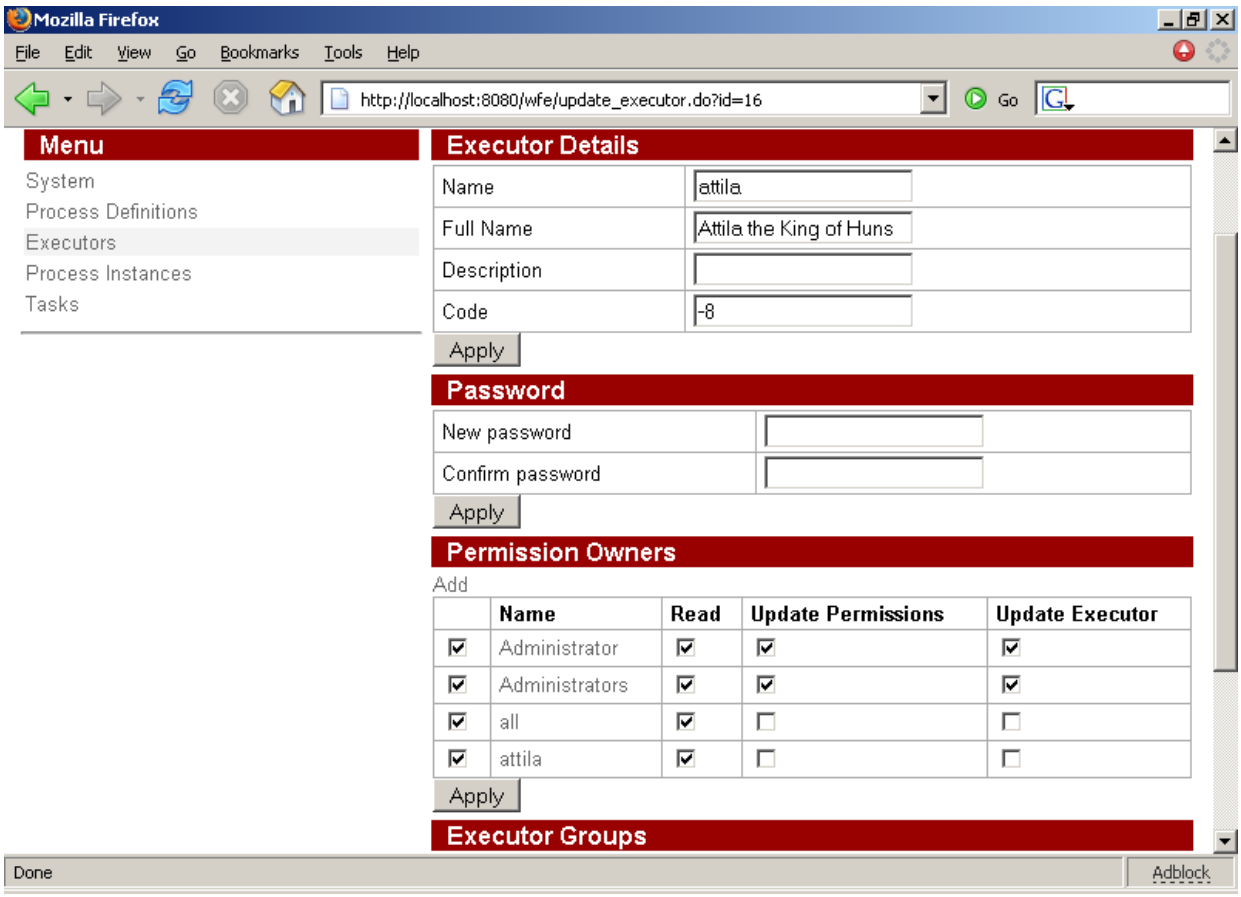
Click the menu item “System”, and give the group *all* login and read permissions.



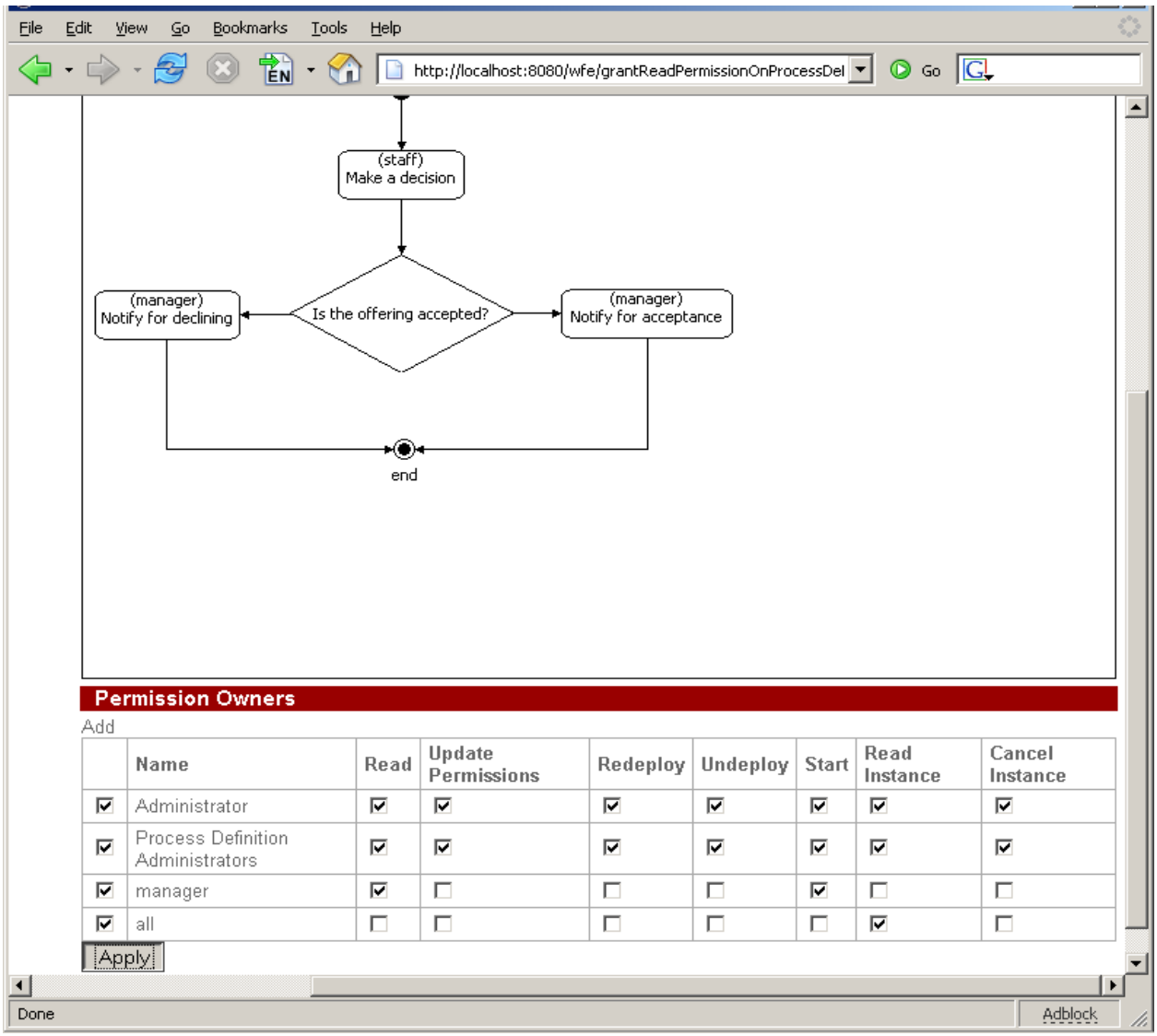
Edit the *staff* group: Give permission to read and list to the *all* group.



Give permission to read the *all* group for every actor. Give permission to read and list the *all* group for «staff» group.

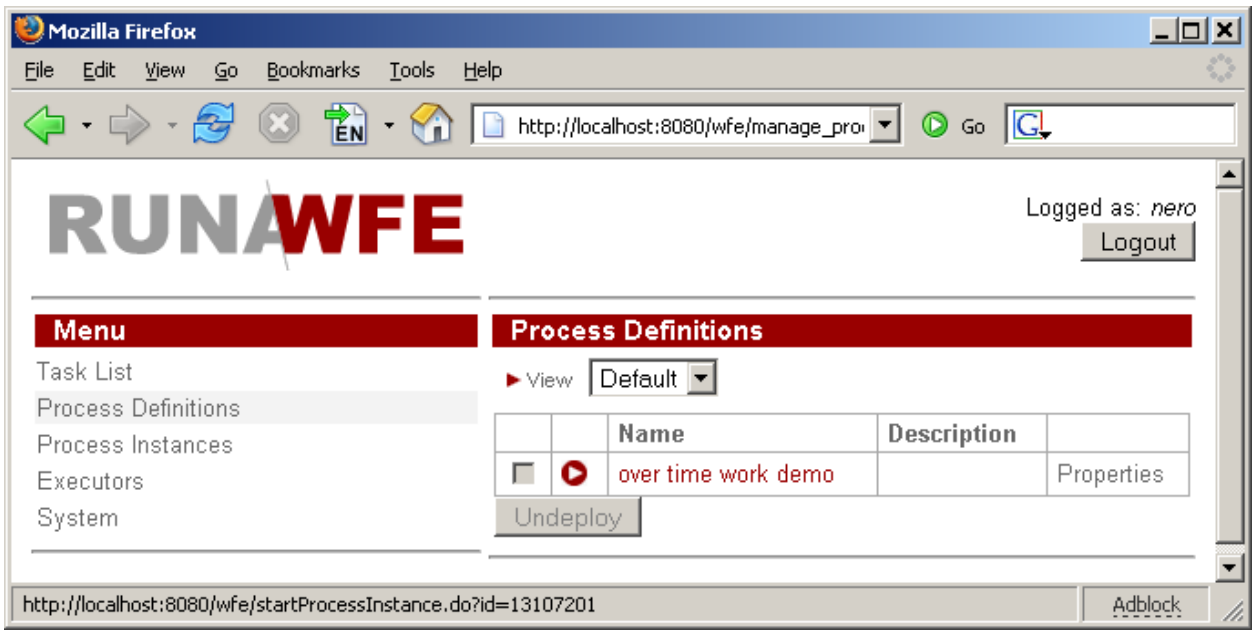


Click on processes “over time work demo” properties. Set the permission “read instance” for *all* group and permissions “read” and “start” for *manager* group.

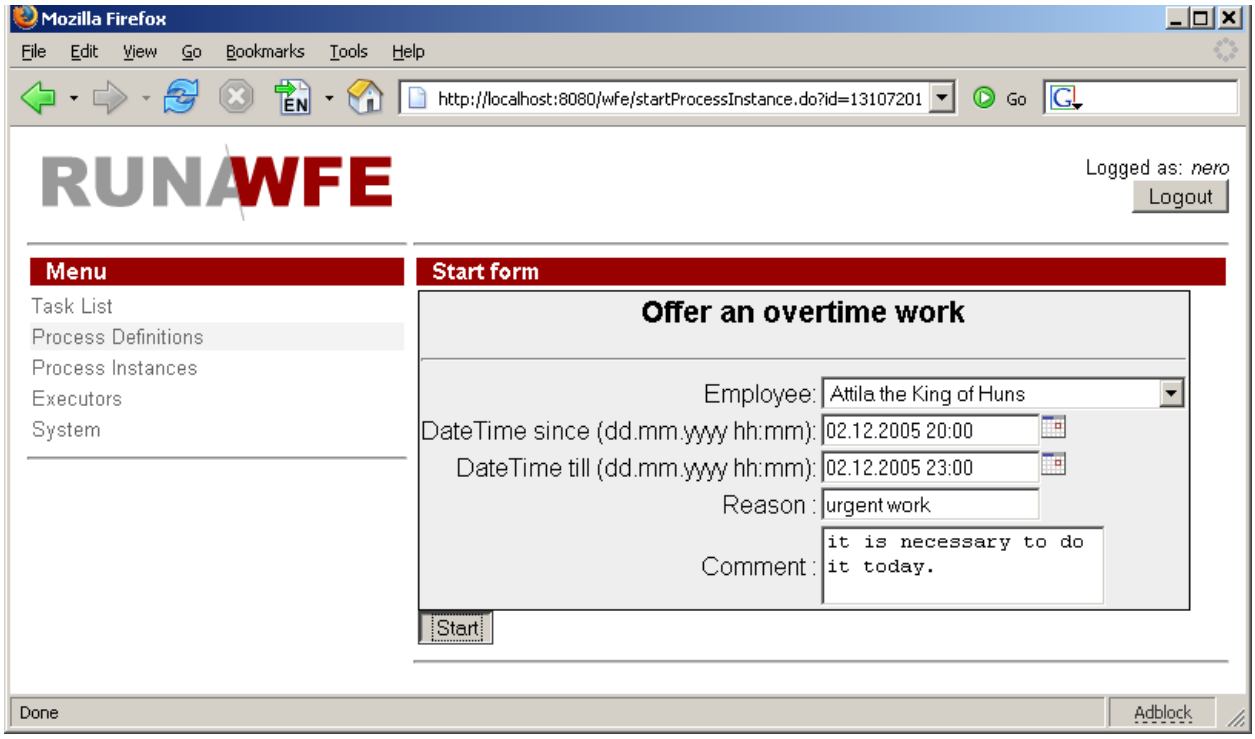


Process running

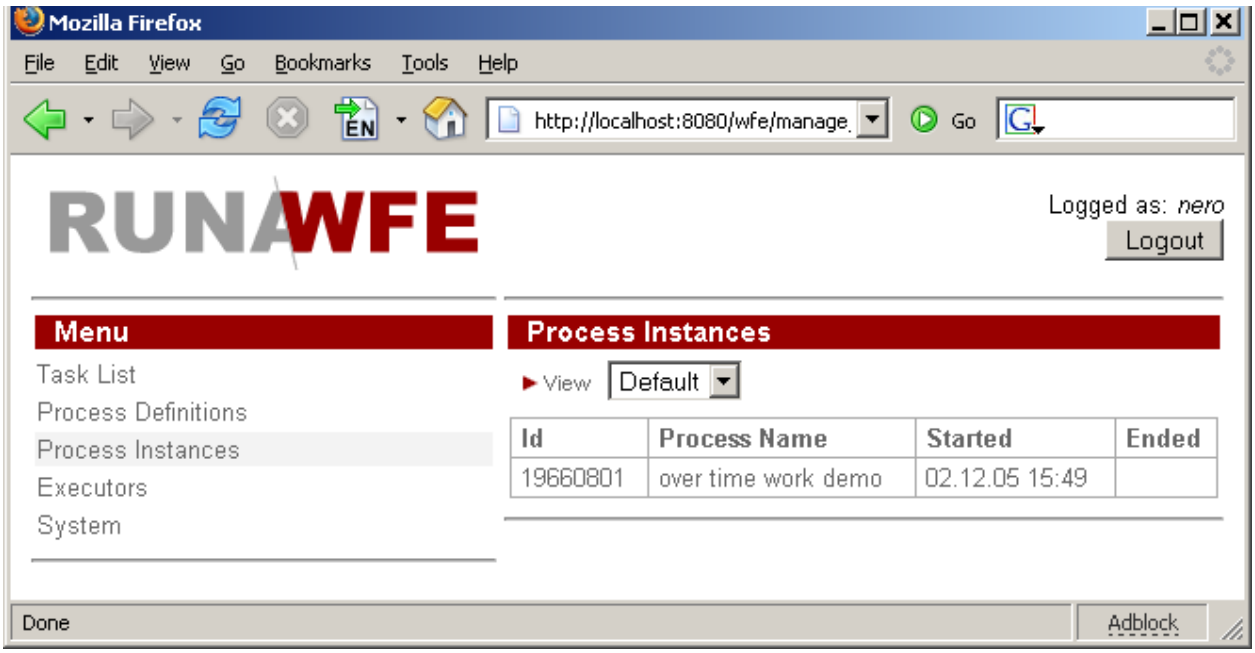
Enter the system as nero (password is “123”). Click the “Process Definitions” menu:



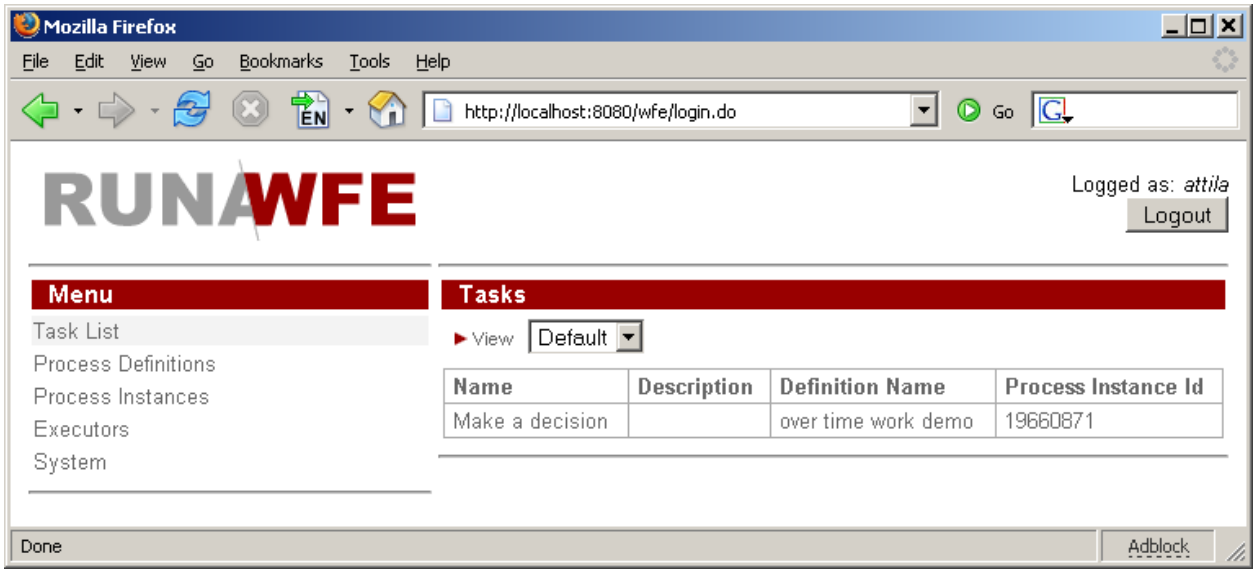
Click on “over time work demo” process. Fill the form and click “Start” button.



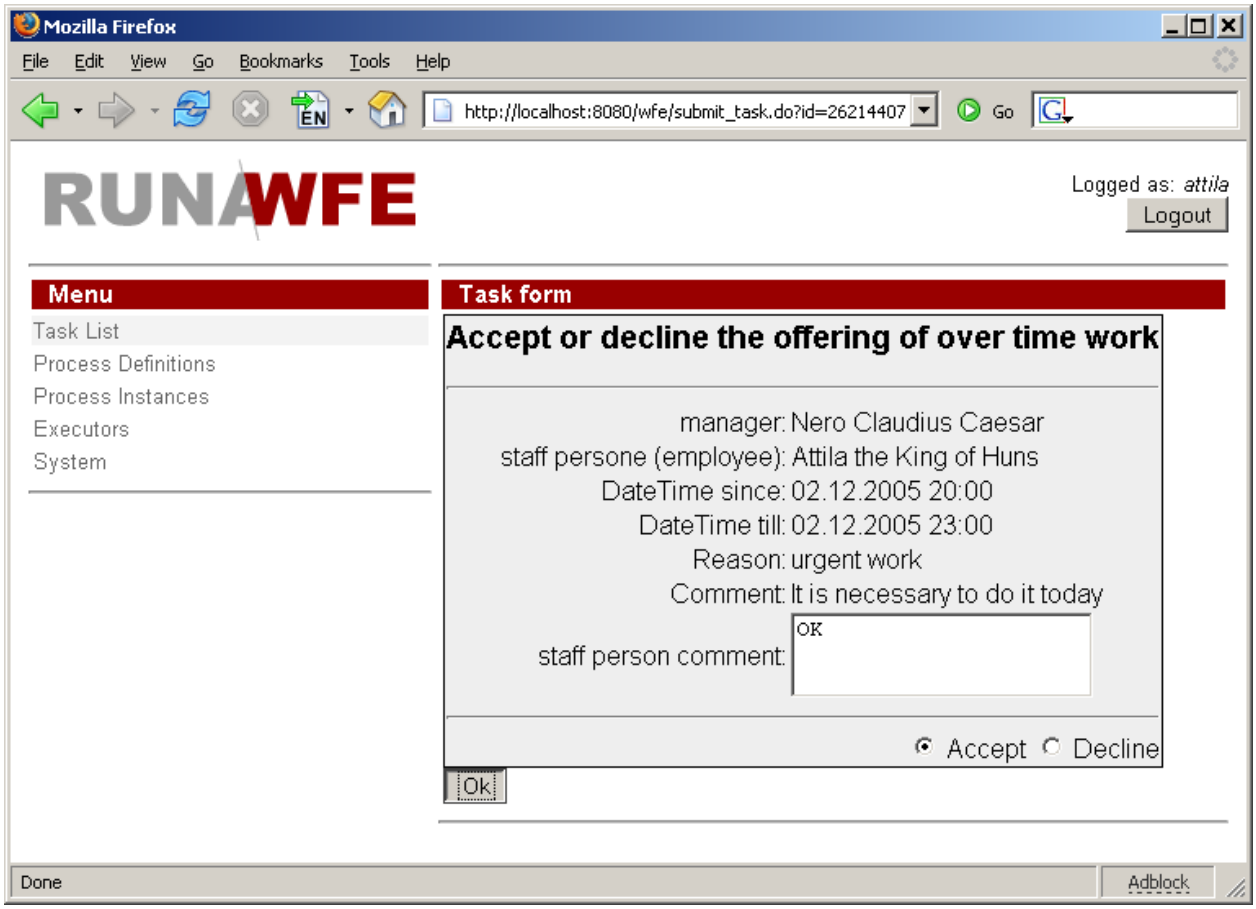
The process instance will be created. You can see it in the “Process Instances” menu.



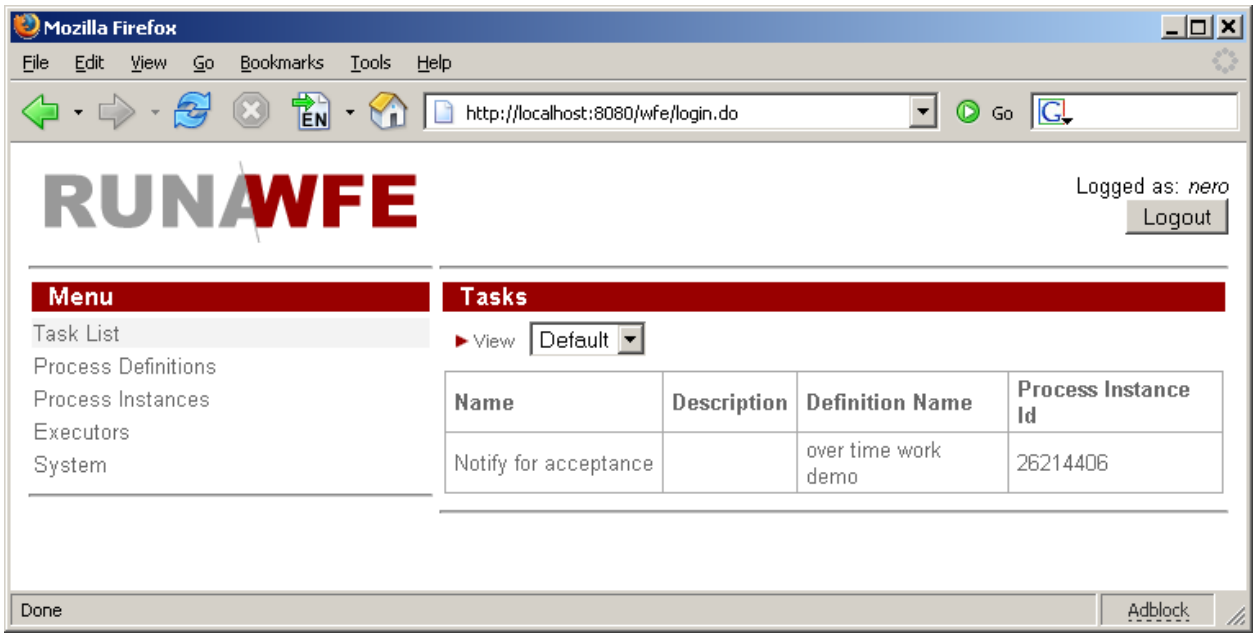
Enter the system as attila (password is “123”).
You'll see the task in the tasklist.



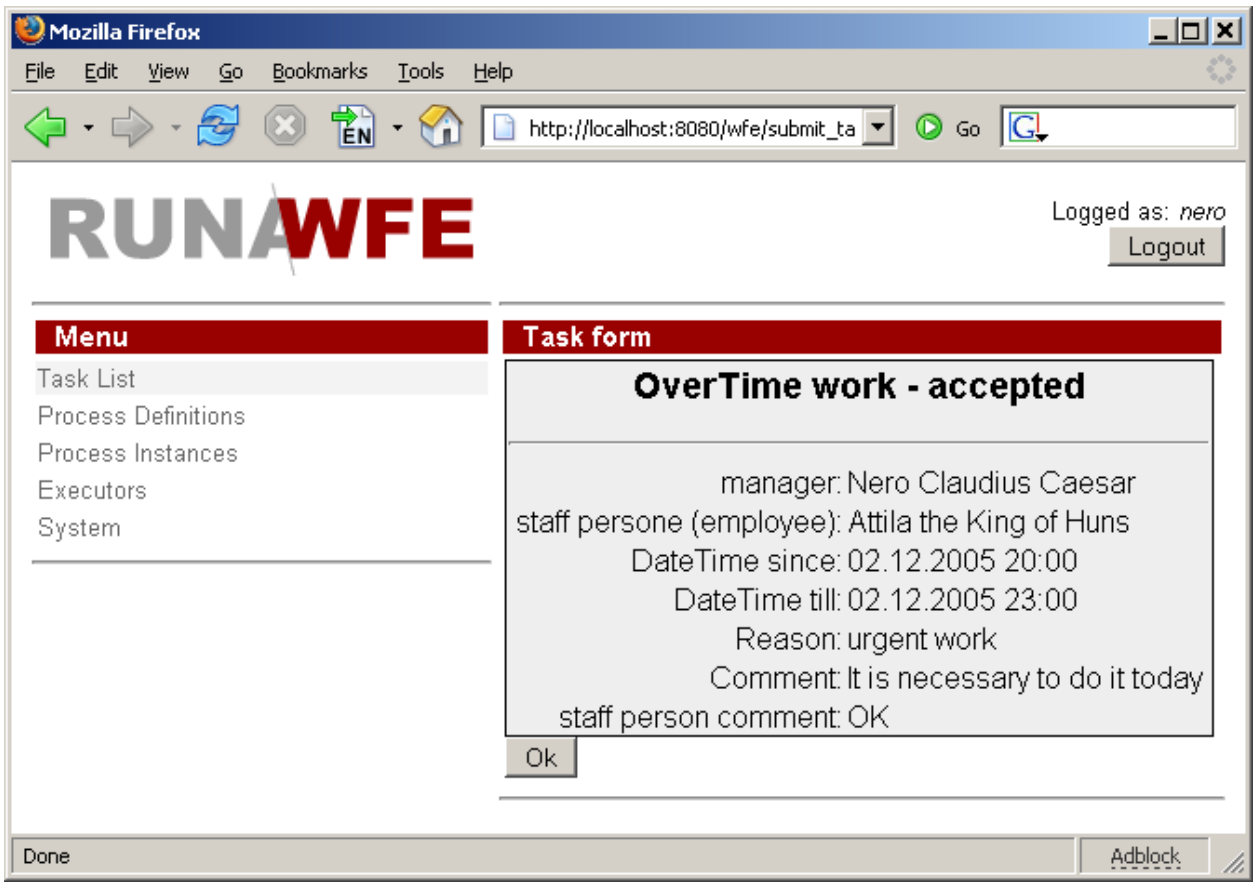
Execute the task.



Enter the system as nero. You'll see the task in the tasklist.



Execute the task.



After clicking «OK» the business process is finished.

Process definition archive file creation

Click “over time work demo.par”, then execute command File/Export:

Form validation

Using validation

On the context menu for the form in process diagram click 'Form validation' menu. It will create default validation file with attached 'required' validator for all variables of that form. Customize validation file for your needs. You can optionally use client-side validation for form based on JavaScript. Check menu 'Use JavaScript validation' in the same context menu. It will be accesible after validation file is created.

Concepts

Validators can be categorized by 2 types: Field Validator (checks is the field value is valid) and Non-Field Validator (It’s not attached to field and checks input data over all fields).

If value of the field is null validation of the Field Validator attached to that field accepts until for that field not attached ‘required’ validator.

Base validators

Validator type	Description + Example
date	Validator that checks if the date supplied is within a specific range. **min** - the min date range. If not specified will not be checked. **max** - the max date range. If not specified will not be checked. <field name="birthday"> <field-validator type="date"> <param name="min">01.01.1990</param> <param name="max">01.01.2000</param> <message>Birthday must be within ${min} and ${max}</message> </field> </field>
double	Validator that checks if the double specified is within a certain range. (if parameter is not specified, it will not be checked) **minInclusive** - the minimum inclusive value **maxInclusive** - the maximum inclusive value **minExclusive** - the minimum exclusive value **maxExclusive** - the maximum exclusive value <field name="percentage"> <field-validator type="double"> <param name="minExclusive">0.123</param> <param name="maxExclusive">99.98</param> <message> It needs to be between ${minExclusive} and ${maxExclusive}

	</message> </field-validator> </field>
email	Validator checks that a given String field is a valid email address. The regular expression used to validate that the string is an email address is: \\b(^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@([A-Za-z0-9-])+((\\.com)|(\\.net)|(\\.org)|(\\.info)|(\\.edu)|(\\.mil)|(\\.gov)|(\\.biz)|(\\.ws)|(\\.us)|(\\.tv)|(\\.cc)|(\\.aero)|(\\.arpa)|(\\.coop)|(\\.int)|(\\.jobs)|(\\.museum)|(\\.name)|(\\.pro)|(\\.travel)|(\\.nato)|(\\..{2,3})|(\\..{2,3}\\..{2,3}))$)\\b <field name="myEmail"> <field-validator type="email"> <message>Must provide a valid email</message> </field-validator> </field>
expression	A Non-Field Level validator that validates based on BSH expression supplied. **expression** - the BSH expression to be evaluated against the variables (Must evaluate to a Boolean) <validator type="expression"> <param name="expression"><![CDATA[number > number2]]></param> <message>Failed to meet BSH Expression</message> </validator>
number	Validator that checks if the integer specified is within a certain range. (if parameter is not specified, it will not be checked) **min** - the minimum value **max** - the maximum value <field name="age"> <field-validator type="int"> <param name="min">20</param> <param name="max">50</param> <message>Age needs to be between ${min} and ${max}</message> </field-validator> </field>
regex	Validates a string field using a regular expression. **expression** - The RegExp expression REQUIRED **caseSensitive** - Boolean (Optional). Sets whether the expression should be matched against in a case-sensitive way. Default is true. **trim** - Boolean (Optional). Sets whether the expression should be trimed before matching. Default is true. <field name="myStrangePostcode"> <field-validator type="regex">

	<param name="expression"><![CDATA[([aAb][123][eEfF][456])]]></param> </field-validator> </field>
required	Checks that field value is not null
requiredstring	Validator checks that a String field is non-null and has a length > 0. (i.e. it isn't ""). The "trim" parameter determines whether it will trim the String before performing the length check. If unspecified, the String will be trimmed. **trim** - trim the field name value before validating (default is true) <field name="username"> <field-validator type="requiredstring"> <param name="trim">true</param> <message>username is required</message> </field-validator> </field>
stringlength	Validator checks that a String field is of a certain length. If the "minLength" parameter is specified, it will make sure that the String has at least that many characters. If the "maxLength" parameter is specified, it will make sure that the String has at most that many characters. The "trim" parameter determines whether it will trim the String before performing the length check. If unspecified, the String will be trimmed. **maxLength** - The max length of the field value. Default ignore. **minLength** - The min length of the field value. Default ignore. **trim** - Trim the field value before evaluating its min/max length. Default true <field name="myPurchaseCode"> <field-validator type="stringlength"> <param name="minLength">10</param> <param name="maxLength>10</param> <param name="trim">true</param> <message>Code needs to be 10 characters long</message> </field-validator> </field>
url	Validator checks that a given field is a String and a valid URL <field name="myHomepage"> <field-validator type="url"> <message>Invalid homepage url</message> </field-validator> </field>

DTD for the form validation file

<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT validators (field|validator)+>

```
<!ELEMENT field (field-validator+)>
<!ATTLIST field name CDATA #REQUIRED>
<!ELEMENT field-validator (param*, message)>
<!ATTLIST field-validator type CDATA #REQUIRED short-circuit (true|false) “false">
<!ELEMENT validator (param*, message)>
<!ATTLIST validator type CDATA #REQUIRED short-circuit (true|false) "false">
<!ELEMENT param (#PCDATA)>
<!ATTLIST param name CDATA #REQUIRED>
<!ELEMENT message (#PCDATA)>
<!ATTLIST message key CDATA #IMPLIED>
```