Process Panel

Process Panel

This panel allows you to execute arbitrary files after installation. The details for the compilation are specified using the resource `ProcessPanel.Spec.xml`.

The XML file has the following format:

```
<processing>
  <job name="do xyz">
    <os family="windows" />
    <executefile
name="$INSTALL_PATH/scripts/xyz.bat"
workingDir="$INSTALL_PATH">
      <arg>doit</arg><arg>$variable</arg>
    </executefile>
  </job>
  <job name="do xyz">
    <os family="unix" />
    <executefile
name="$INSTALL_PATH/scripts/xyz.sh">
      <arg>doit</arg><arg>$variable</arg>
    </executefile>
  </job>
</processing>
```

Each <job> may have the following attributes an `<os>` attribute.

Attribute	Default	Description	Mandatory
`name`	(not set)	IzPack condition ID for running the job	yes
`condition`	(not set)	IzPack condition ID for running the job	no

Nested elements to <job>

<os> - Limit Job Execution Environment

See the IzPack OS Restrictions tag.

<executeFile>- Execute Shell Scripts

In addition to `<arg>` elements, the `<executefile>` element also accepts `<env>` elements to set variables in the environment of the target process. This can be useful if this process requires some environment variables, such as its installation directory, to work properly. An `<env>` element has the following syntax: `<env>variable=value</env>`. Note the value supports variable substitution, for example: `<env>MY_PRODUCT_HOME=$INSTALL_PATH</env>`. The **workingDir** attribute for the <executefile> element adds the ability to set the working directory of the process spawned by the ProcessBuilder object, much as <env> elements refer to the environment object of ProcessBuilder.

The ProcessPanel now also supports configurable behaviour for the panel's "Previous" and "Next" buttons. By adding `<onFail>` and `<onSuccess>` childs to the `<processing>` element, you define which buttons you want unlocked in case of failure and in case of success, respectively.

```
<processing>
  <job name="do xyz">
    <executefile
name="$INSTALL_PATH/scripts/xyz.bat">
      <arg>doit</arg><arg>$variable</arg>
    </executefile>
  </job>
  <onFail previous="true" next="false" />
  <onSuccess previous="true" next="true"
condition="mycondition" />
  <onSuccess previous="false" next="true"
condition="!mycondition" />
</processing>
```

In the above example the job *do xyz* would be run, and if it returned with an error, the *next* button would be greyed out, and the button to the *previous* page would be enabled. If it returned without an error, the conditions of the `<onSuccess>` elements would be checked and the respective action would be taken.

`<executeclass>` - Execute Java Classes

It is also possible to execute Java classes from this panel. Here's what this feature author (Alex Bradley) says:

> i've been able to work around my requirements by extending the `ProcessPanelWorker` functionality to run user-specified classes. i've extended the DTD of the `ProcessPanel.Spec.xml` to include a new element:

```
<executeclass name="classname">
<args..../>
</executeclass>
```

> i've also added a new sub-class of `Processable` called `executeclass`. This will run a user-specified class in the context of the installer JVM with a single method :

```
run( AbstractUIProcessHandler handler,
String[] args]);
```

> It can do everything i need and more. In particular, it allows me to write a process extension and still be able to provide feedback to the user through the feedback panel, and to add new functionality to the installer, after its been built.

To use the executeclass facility, you will need to create a jar file with your class and then add it to the installer with the `The Jar Merging Element`.

`<executeForPack>` - Only execute the job for certain packs

This can be be used to run the job only if the pack is enabled. For example, the batch file will if either the `Sources` o r `Executables` packs are selected at install time.

```
<processing>
 <job name="List files">
  <executeForPack name="Sources"/>
  <executeForPack name="Executables"/>
  <os family="windows" />
  <executefile
name="$INSTALL_PATH\batch\ListFiles.bat" />
 </job>
</processing>
```

`<logfiledir>` - Output of the processPanel saved to a log

New with version 3.7 is the possibility to tee output that is written to the ProcessPanel's textarea into an optional logfile. Using this feature is pretty much straightforward, you only have to add a line in `ProcessPanel.Spec.xml` t hat will tell IzPack the location, where the logfile should be stored.

Variable substitution is performed, so you can use `$INSTALL_PATH` as example.

The name of the logfile is not (yet) configurable but should fit in most cases. It will be namedInstall_V<$APP_VER><*YYYY*>~~<*MM*>~~<*DD*><hh>~~<mm>~~<ss>_<RandomId>.log
Here's an example:

```
<processing>
  <logfiledir>$INSTALL_PATH</logfiledir>
  <job name="do xyz">
    <os family="windows" />
    <executefile
name="$INSTALL_PATH/scripts/xyz.bat">
      <arg>doit</arg><arg>$variable</arg>
    </executefile>
</processing>
```

This will generate a logfile named e.g. `Install_V1.3_2004-11-08_19-22-20_43423.log` located in `$INSTAL L_PATH`.

`ProcessPanelWorker` will write all output that is directed to `stdout` and `stderr` to this file if `ProcessPanel.Sp ec.xml` contains the `logfiledir` entry.

Please note that this one file is used for storing the complete output of all jobs and not a file for each job that is run.