UserInputPanel

UserInputPanel

(by Elmar GROM)

This panel allows you to prompt the user for data. What the user is prompted for is specified using an XML file which is included as a resource to the installer. Most of the panels that come with IzPack take user input in some form. In some panels this is through a simple user acknowledgment in others the user can enter text or select a directory through a file open dialog. In all of those cases the user input is used for the specific purpose needed by the panel that takes the input. However, if you need user input during installation that will later on be available to your application then you need to use the user input panel.

To use this panel, list it in the install file with the class name `UserInputPanel`. In addition, you must write a XML specification and add it to the install resources. The name of this resource must be `userInputSpec.xml`.

```
<resources>
    <res id="userInputSpec.xml"
src="user_input_spec.xml" parse="yes"
type="xml"/>
</resources>
...
<panels>
...
    <panel classname="UserInputPanel"
id="userinputpanel.order"/>
...
</panels>
```

A UserInputPanel can be highly dynamic from IzPack 4.3 on, as it will be refreshed every time the user input changes and will be rendered based on conditions. For instance, it would be possible to enable or disable some more options by clicking a checkbox.

Here's an example userInputSpec.xml (http://%7b%7bservername%7d%7d/sample-userInputSpec.html) showing 3 panels. There are some advanced features in this example, but the general flow should be looked at first.

The user input panel is a blank panel that can be populated with UI elements through a XML specification file. The specification supports text labels, input elements, explanatory text and some minor formatting options.

The following types of user input elements are supported:

- Text

- Combo Box
- Radio Buttons
- Check Box
- Password
- File
- Multiple files
- Directory
- Rule Input Field
- Search Field

Additionally visual elements can be added using the following types:

- Static Text
- Title
- Space
- Divider

The way in which this panel conveys the user input to your application is through the variable substitution system. User input is not directly inserted into your configuration files but the variables that you specify for this panel are set in the variable substitution system. After this operation has taken place the variables and associated values are available for all substitutions made. This way of operation has a number of implications that you should be aware of.

First, not only can you set additional variables in this way but you can also modify variables that are defined elsewhere -even built in variables. For this reason you should be careful to avoid overlaps when choosing variable names. Although there might be cases when it seems useful to modify the value of other variables, it is generally not a good idea to do so. Because you might not exactly know when other variables are set and when and where they are used throughout the installation process, there might be unintended side effects.

Second, the panel must be shown at a point during the installation process before the variables are used. In most cases you will use the values to substitute variables in launch and configuration files that you supply with your installation. For this to work you place this panel before the install panel, because the install panel uses the variable substitutor to replace all such variables. Although using this panel any later in the process will correctly set the variables internally, there won't be any affect on the files written to disk. You can also use variables set in this way in other panels that you have written yourself. There is a section in the chapter on writing your own panel that explains how to do this. Also in this case it is important to place the associated input panel in the process before the variables are used.

At this point I would also like to mention that it is possible to hide every field element based on conditions.

It would also be possible to hide select elements on the panel or the panel altogether if certain packs are not selected. For this to work you must place this panel after the packs panel. One side effect of using this feature is that it is not possible to step back once the user input panel is displayed. This is because the user might make changes in the packs selection that would require a complete rebuild of the UI. Unfortunately, building the UI is an irreversible process, therefore the user can not be allowed to go back to the packs panel.

The Basic XML Structure

The top level XML section is called `<userInput>`. For most panels it does not make sense to present them more than once, however you might want to present multiple user input panels -with different content of course. Therefore the `<userInput>` section can contain multiple tags that each specify the details for one panel instance. The tag name for this is `<panel>`.

The `<panel>` tag uses the following attributes:

Name	Required	Description	Values
id	yes	This is the id of the user input panel for which this specification should be used. This id links to the panel specification in the install.xml file.	String Value
layout	no	Sets the alignment of fields used in the panelThere are three general layout rules this panel uses, they are `left` , `center` and `right`. While i think left is most commonly used, you might want to experiment with this attribute and see which you like best. The default is `left`.	left, center, right **default:** l eft
border	no	Normally the user input is shown with a small border. To prevent this border set this attribute to `false`.	true, false **default: true**
column_width	no	This can be used to set the column width of the two column layout. This value is in percent of the whole size. If it is set to `0`, which is the default, the with will be set automatically. It it is set to `50` the gap between the left and right column is in the middle of the panel. This makes it possible to make a centered layout.	0-100 percent **default: 0**

Concepts and XML Elements Common to All Fields

Before we dive into the details of defining the various UI elements I would like to present XML elements and general concepts that apply throughout. This saves me a lot of work in writing and you a lot of repetitive reading and maybe a tree or two.

The UI elements are generally laid out top to bottom in the order they appear in the XML file. The only exception to this rule is the title, which always appears at the very top. The layout pattern for the input fields is as follows: If a description is defined, it appears first, using the full available layout width. The input field is placed beneath the description. With fields such as the text field or the combo box, the label is placed to the left and the input field to the right. Fields such as radio buttons and check boxes are somewhat indented and have the label text appear to their right.

Each UI element is specified with a `<field>` tag. The <field> tag has some common attributes:

Name	Required	Description	Value
type	yes	Used to specify what kind of field you want to place.	Any valid field type
`label_position`	no	Used to change the default position of the label.	west, westonly, both, eastonly, east **defatul:** we st
label_align	no	Used to change the default alignment of the label.	left, center, right **default:** l eft
`label_indent`	no	Controls whether or not the label is indented.	true, false **default:** false
`control_position`	no	Used to change the default position of the field's control.	west, westonly, both, eastonly, east **default:** ea st
control_align	no	Used to change the default alignment of the control.	left, center, right **default:** l eft
control_indent	no	Controls whether or not the control is indented.	true, false **default:** false
variable	yes, if input field	Specified the variable that should be substituted with the user input.	A variable name.
conditionid	no	Specifies the id of a condition in install.xml which must evaluate to true in order to display the field.	A valid condition id in install.xml.

Examples

In the following example, the the label and the control will be shown on it's own row.

```
<field type="text" variable="value1">
  <spec txt="The label" id="" size="20"
set="default value" label_both="both"
control_position="both" />
</field>
```

In the following example, the label is right aligned to the control.

```
<field type="text" variable="value1">
  <spec txt="The label" id="" size="20"
set="default value" label_align="right" />
</field>
```

Almost all fields allow a description element. The description is part of the data within the field element. There can only be one description per field. If you add more than one description, the first description is used and the others ignored. There are three attributes used with this tag. The text is specified through the `txt` or the `id` attribute. The details on using them are described below. The attributes are all optional but you must specify text to use, either directly or through the `id` attribute. In addition, you can set the text justification to `left`, `center` and `right` with the `align` attribute.

Name	Required	Description	Value
txt	yes, if no id	Specified the default text to use in absence of a language package.	Any text
id	yes, if no txt	Specifies text description through use of a language package.	A valid id
align	no	Sets the justification of the description.	left, center, right **default:** l eft

The following example illustrates the general pattern for field specification:

```
<field type="text"
variable="myFirstVariable">
  <description align="left" txt="A
description" id="description1"/>
   .
   .
   .
</field>
```

A very frequently used pattern is for the definition of text. Where ever text is needed (labels, descriptions, static text, choices etc.) it can be specified in place using the `txt` attribute. This is convenient if you are only supporting a single language. However, if you would like to separate your text definitions from the panel specification or if you need to support multiple languages you might want to use the `id` attribute instead to only specify an identifier. You can then add multiple XML files with the same name as this spec file (userInputSpec.xml) appended with an underscore '_' and the the appropriate three letter ISO3 language code. The content of those files must conform to the specification for IzPack language packages. For more details on this topic see the chapter on language packages under advanced features. `id` defines an identifier that is also defined in the language package, together with the localized text to use. It is possible to use both the `txt` and the `id` attribute. In this case the text from the language package is used. If for some reason the language package is not available or the `id` is not defined there, the text specified with `txt` is used as default.

All input fields can be pre-set with a value of your choice. Although the details vary a bit from field type to field type, the `set` attribute is always used to accomplish this. The `set` attribute is of course optional. Please note that if you use the `set` attribute, you would have to keep in mind, that the UserInputPanel will be rendered after each user input. Thus it will be set back to the default value if you don't use a condition to handle that. IzPack generates builtin conditions for every variable used as target for a field to be able to check if there's any input.

All fields that take user input use a `<spec>` tag to define the details of the input field. In the some cases the content of this tag is rather simple. Input fields with a more complex nature tend to have accordingly complex content in this tag. Since the details vary widely, they are explained with each input field.

Any number of `<createForPack name=`*`a pack name`*` />` tags can be added to the `<panel>` and `<field>` secti ons. This tag has only one attribute and no data. The attribute is `name` and specifies the name of one of the installation packs that you have defined. Here is how it works: if no `<createForPack ...>` tag exists in a section, the entity is always created. However, if the tag exists, the entity is only created if one or more of the listed packs are selected for installation. As mentioned before, if you are using this feature, make sure the user input panel shows up after the packs panel.

Also, any number of `<createForUnselectedPack name=`*`a pack name`*` />` tags can be added to the `<panel>` and `<field>` sections. This tag has only one attribute and no data. It works exactly like createForPack except that once added user input panel will appear for only NOT Selected packs. As mentioned earlier, you need to make sure that the user input panel shows up after the packs panel for this feature to work.

There is a possibility to use variables in those elements where the text is supplied via `txt` attribute. This includes static fields and input fields (spec, description). The text can contain unlimited number of variables that will be substituted. Variable substitution also works with language packs, just use variables in your language pack, and they will be still substituted properly.

Example

In the following example, the variables: name1, name2, name3 will be substituted.

```
<field type="text" variable="value1">
  <description align="left" txt="Configuration for $name1 and $name2" id=""/>
  <spec txt="The value for $name3:" id="" size="20" set="default value" />
</field>
```

Internationalization

To provide internationalization you can create a file named `userInputLang.xml_xyz` where `xyz` is the ISO3 code of the language in lowercase. Please be aware that case is significant. This file has to be inserted in the resources section of `install.xml` with the `id` and `src` attributes set at the name of the file.

Example:

If you have the following userInputSpec.xml and you want to internationalize `input.comment`, `input.proxy`, `input.port` for English and French you have to create two files named userInputLang.xml_eng and userInputLang.xml_fra:

```
<userInput>
  <panel id="panel1">
    <field type="staticText" align="left"
txt="My comment is here."
id="input.comment"/>
    <field type="text"
variable="proxyaddress">
      <spec txt="Proxy Host:"
id="input.proxy" size="25" set=""/>
    </field>
    <field type="text" variable="proxyPort">
      <spec txt="Proxy Port:"
id="input.port" size="6" set=""/>
    </field>
  </panel>
</userInput>
```

userInputLang.xml_eng

```
<langpack>
  <str id="input.comment" txt="English:My
comment is here."/>
  <str id="input.proxy" txt="English:Proxy
Host:"/>
  <str id="input.port" txt="English:Proxy
Port:"/>
</langpack>
```

userInputLang.xml_fra

```
<langpack>
  <str id="input.comment" txt="French:My
comment is here."/>
  <str id="input.proxy" txt="French:Proxy
Host:"/>
  <str id="input.port" txt="French:Proxy
Port:"/>
</langpack>
```

you will also have to add the following to the install.xml file

install.xml

```
<resources>
  ...
  <res id="userInputSpec.xml"
src="userInputSpec.xml"/>
  <res id="userInputLang.xml_eng"
src="userInputLang.xml_eng" />
  <res id="userInputLang.xml_fra"
src="userInputLang.xml_fra" />
  ...
</resources>
```