CompilePanel

Usage

The CompilePanel can be used to compile Java source code that is being installed.

This panel must appears after xxxx panel to be sure that the code is available for compilation.

The details for the compilation are specified using the resource **`CompilePanel.Spec.xml`**. Need explanation and example showing how the panel is linked to the Compile.Spec.xml resource or is it done magically

Example of a **`CompilePanel.Spec.xml`** file that offers a choice of 2 compilers and includes 2 compilation tasks.

:

CompilePanel.Spec.xml resource example

```
<izpack:compilation version="5.0" xmlns:izpack="http://izpack.org/schema/compilation"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xsi:schemaLocation="http://izpack.org/schema/compilation
http://izpack.org/schema/5.0/izpack-compilation-5.0.xsd">

    <global>
        <compiler>
            <choice value="$JAVA_HOME/bin/javac"/>
            <choice value="jikes"/>
        </compiler>
        <arguments>
            <choice value="-O -g:none"/>
            <choice value="-O"/>
            <choice value="-g"/>
            <choice value=""/>
        </arguments>
    </global>
    <jobs>
        <classpath add="$INSTALL_PATH/src/classes/"/>
        <job name="optional name">
            <directory name="$INSTALL_PATH/src/classes/xyz"/>
        </job>
        <job name="another job">
            <packdependency name="some package name"/>
            <classpath sub="$INSTALL_PATH/"/>
            <directory name="$INSTALL_PATH/src/classes/abc"/>
            <file name="$INSTALL_PATH/some/file.java"/>
        </job>
    </jobs>
</izpack:compilation>
```

Actually, jobs can be nested. Why do you want to do that? How do you nest them?

A change to the classpath within a job only affects this job and nested jobs. The classpath should be specified before any files or directories.

Compilation elements

Global

Element	Description	Required
`compiler`	This section lists the choices of compilers that can be used to compile the sources	
`arguments`	????	

Jobs

Jobs element

```
<jobs>
    <classpath add="$INSTALL_PATH/src/classes/" />
    <job name="optional name">
      <directory name="$INSTALL_PATH/src/classes/xyz" />
    </job>
    <job name="another job">
      <packdepency name="some package name" />
      <classpath sub="$INSTALL_PATH/" />
      <directory name="$INSTALL_PATH/src/classes/abc" />
      <file name="$INSTALL_PATH/some/file.java" />
    </job>
  </jobs>
```

The **`<jobs>`** element contains a `<classpath>` element and one or more **`<job>`** elements.

The **`<classpath>`** element at the **`<jobs>`** level applies to nested **`<job>`** tasks.

Job

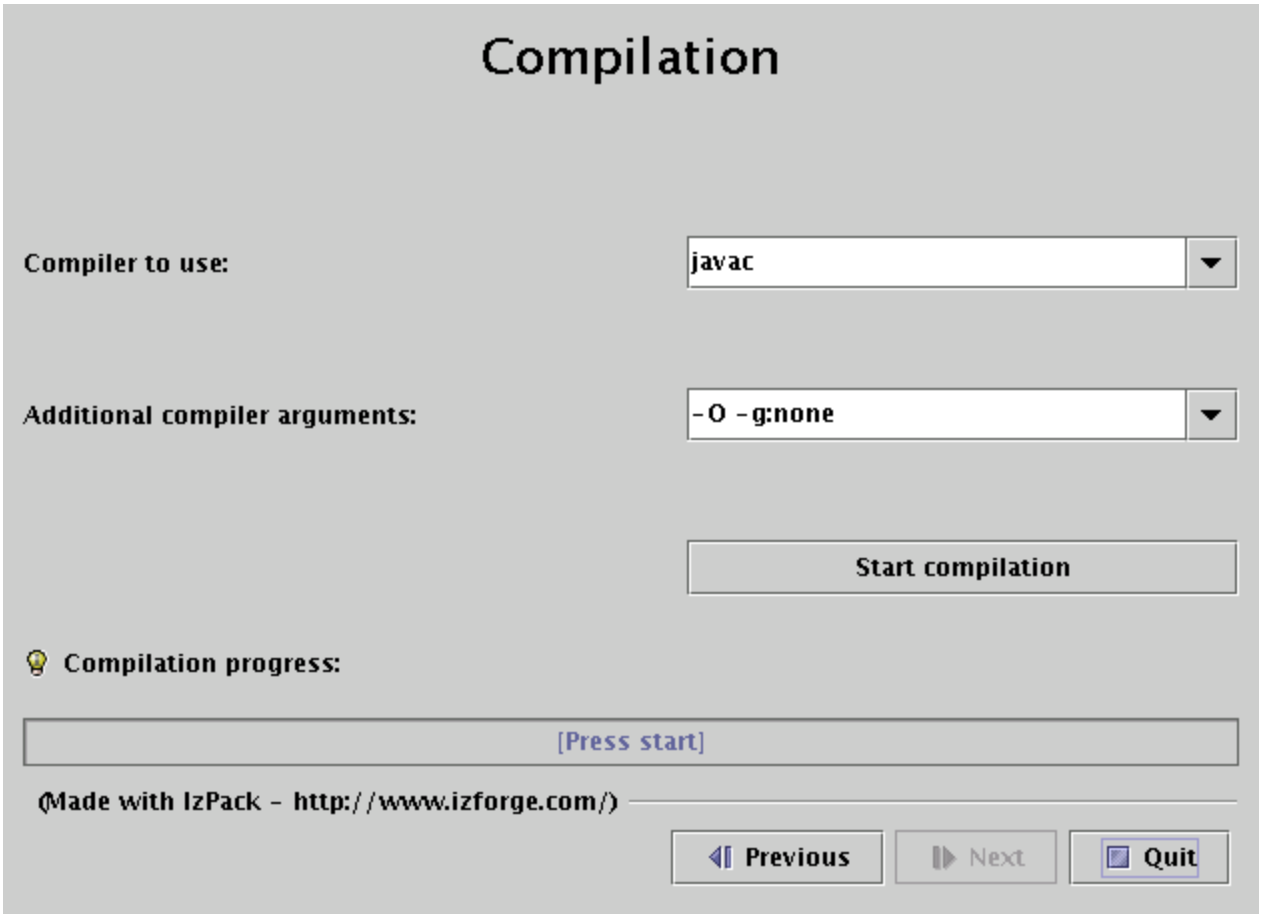
Jobs can have an optional **name** attribute which is used to document the task's role.

Jobs can have the following nested elements.

Element	Description	Required
`packdepency`	???	No
`classpath`	???	No
`directory`	???	No
`file`	???	No

User Interaction

The user can change the compiler to use and choose from some default compilation options before the compilation is started.



Customization

Resource Name	Description

CompilePanel.Spec.xml	This code appears above. This description should be about setting the resource up to point to the file. <compilation> <global> <compiler> <choice value="..." /> ... </compiler> <arguments> <choice value="..." /> ... </arguments> </global> <jobs> <classpath add="..."/> <job name="optional name"> <directory name="..." /> <packdependency name="some package name"/> <classpath sub="..."/> <file name="..." /> </job> … ... </jobs> </compilation>