

2002/01 mkJPS

Holger Zahnleiter
holger@zahnleiter.org

Date: 07.04.03

Abstract

Some code generators generate Java classes, including a package declaration, but simply put all files into a single destination folder, without paying attention to the package definition. When compiling these generated classes, the resulting class files get distributed to subfolders, in accordance to their package definition. In case, one also wants to distribute the generated Java source files according to their package definition, he might use this program.

This tool is meant for moving Java source code files into folders, corresponding to the package the file belongs to. One can use it as a batchfile, shell script or an Ant task.

1 Project characteristics

Project code	2002/01
Project name	mkJPS
Started	02.09.02
Ended	03.09.02
Version/release	1/1
Used tools and libraries	JDK 1.3.1
Runs on platform	Java 1.3.1 (or newer), Ant 1.5 (or newer)

2 Introduction

A typical situation in which code generators are used is when one describes a (remote) service as an IDL or WSDL file. On server and client side, stubs can be created from this service description. One example for such a code generator is `wsdl2java`. This tool belongs to GLUE, a SOAP toolkit from The Mind Electric, and exactly behaves as stated in the abstract. `wsdl2java` generates Java source files, including the package declaration, but it does not copy the files to the appropriate folder. Since I usually use Ant to compile, test and deploy my applications, writing a new task for Ant seemed to be natural.

On my various Web Service projects, the combination of `wsdl2java` and `mkJPS` both served well.

3 Installation

In case you only downloaded `mkjps.jar`, the only thing you need to do is to copy that Java archive to any folder and make it part of your class path. In case you downloaded the complete archive, including source files and documentation, you need to unpack all files to any folder. The sub-folder `/lib` contains the Java archive `mkjps.jar`. Make it part of your class path. In case you do not want `mkjps.jar` to be part of your class path, use the `-classpath` option of your Java Virtual Machine.

4 The source classes

mkJPS consists of three Java classes, all part of the same package `org.zahnleiter.tools.mkjps`.

- The class `JavaPackageStructureGenerator` implements the functionality of creating a directory tree and distributing Java source files to that directory structure according to their package declaration.
- The class `mkJPS` contains a `main()` method, allowing mkJPS to run as a batch or shell script.
- The class `mkJPS4Ant` enables mkJPS to be used from within Ant as a new task, by using the `<taskdef>`-tag.

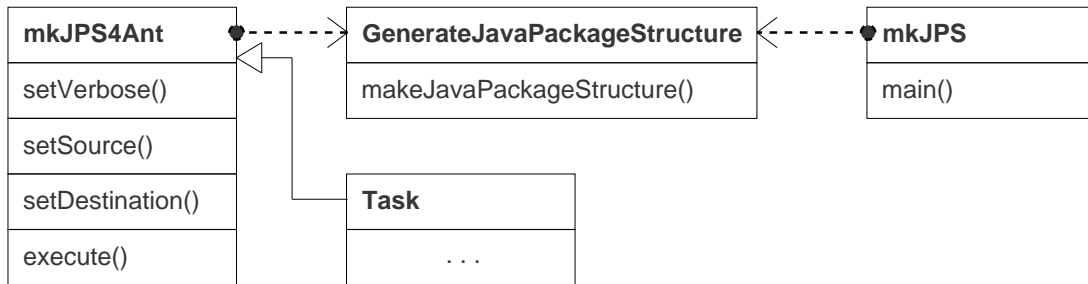


Figure 1: Class diagram

4.1 The main class (`JavaPackageStructureGenerator`)

Besides several protected methods, `JavaPackageStructureGenerator` contains a public method `int makeJavaPackageStructure(File pSrc, File pDest, boolean pVerbose)`, which does some assertions and then calls the recursive method `protected int processFolder(File pSrc, File pDest)`. These methods together implement the functionality mentioned above. In broad outline, the overall algorithm looks like this:

- Make sure `pSrc` and `pDest` are folders and do not denote the same folder.
- For all files in the given folder (`pSrc`) do:
 - Parse the file by using the `java.io.StreamTokenizer` class.
 - Search for the first appearance of the keyword `package`. Replace all dots in the package name by slashes to gain the folder name. Copy the file relative to `pDest` by concatenating the sub-folder name generated before. If no package is given, copy the file to the root folder, which is `pDest`. Please note, that the stream tokenizer is configured to ignore Java style comments and strings to ensure that the `package` keyword is not read from a comment or string.
 - If user wants additional information (`pVerbose == true`), write name of copied file to standard output.
- For all directories in the given folder (`pSrc`) do:
 - Call this method recursively, where the current folder will be passed to the method as the new source folder (`pDest`).

4.2 The wrapper class for scripting (`mkJPS`)

The class `mkJPS` introduces a `main()`-method. This allows `mkJPS` to be called from a command line or shell. In script mode, the names of all files copied are printed to the standard output.

4.3 The wrapper class for Ant integration (mkJPS4Ant)

The class `mkJPS4Ant` needs to inherit from `org.apache.tools.ant.Task`. Which parameters `mkJPS` can take can be determined by examining `mkJPS4Ant` by using reflection. Therefore, the Ant framework searches for methods beginning with `set`. The remainder of the method name will be treated as the parameter name. Ant gathers the parameter values from the Ant script by reading a targets attributes, where the attribute names reflect the names of the `set`-methods mentioned above. The type of the single argument of such `set`-methods denotes the type, which the attribute will have. This class implements the following methods:

- `public void setSource(File pValue)` - Denotes the folder from were to read the Java source files. This will introduce the attribute `source` in the Ant script.
- `public void setDestination(File pValue)` - Denotes the root folder. All Java source files get copied relative to this folder. Introduces the `destination` attribute in the Ant script.
- `public void setVerbose(boolean pValue)` - Denotes whether to print additional information or not. Introduces the `verbose` attribute in the Ant script.
- `public void execute()` - Gets called by Ant after Ant has read the parameters (attributes). This method now invokes the functionality decrived already. When Ant has read a parameter, it calls the corresponding `set`-method to set that value. When calling the method, which implements the funtionality, `execute()` can apply these values and pass them as parameters to the called method.

Please note that this class is only a wrapper, which allows integration into Ant. The "real" functionality is implemented by the `makeJavaPackageStructure()` method which belongs to the class `JavaPackageStructureGenerator`.

5 Ways of using mkJPS

There are three ways of using `mkJPS`.

1. Use it as a batch or shell skript.
2. Directly call the method of the Java class.
3. Use it as a new task of Apache Ant.

Please not that either the file `mkjps.jar` or the folder containing the corresponding class files `org/zahnleiter/tools/mkjps/` must be part of the class path in oder to be able to execute `mkJPS`.

Generally spoken, `mkJPS` accepts three parameters: One denotes the folder, were to find the Java source files. The other denotes the folder were the Java source files will be copied to (relatively). Source and destination folder must not be equal. The third parameter tells `mkJPS` to print out the names of all files copied.

5.1 For using with scripting languages

As said above, the class `mkJPS` contains a `main()`-method, allowing usage of this program from within shell scripts (Unix) or batch files (Windows). In the sub-folder `/bin` one will find two examples for using `mkJPS` from within shell or batch scripts. The only thing important is, that all required Java classes are needed to be in the classpath. Please note, that the program always runs in verbose mode.

The usage is like this: `java org.zahnleiter.tools.mkjps.mkJPS -src source-folder -dest destination-folder`.The meaning of the parameters `source-folder` and `destination-folder` follows the things already said.

5.2 For direct use

For direct use one can call the public `main()`-method of the class `mkJPS` or call the public method `makeJavaPackageStructure()` of the class `JavaPackageStructureGenerator`.

5.3 For use in conjunction with Apache Ant

When desired, mkJPS can incorporate with Apache Ant. Therefore, place the following elements in your Ant script, where:

- `target-name` is the name of the Ant targeted,
- `source-folder` is the name of the folder were to find the Java classes,
- `destination-fplder` is the name of the folder were to store the Java classes and
- `true-or-false` can be either `true` or `false`, in case the user wants to see the names of all files copied.

```
<target name="target-name">
  <taskdef name="mkjps" classname="org.zahnleiter.tools.mkjps.mkJPS4Ant"/>
  <mkjps source="source-folder" destination="destination-folder"
        verbose="true-or-false"/>
</target>
```

A List of project files

File Name	Comments
<code>bin\mkjps.bat</code>	Demonstrates usage in batch files.
<code>bin\mkjps.sh</code>	Demonstrates usage in shell scripts.
<code>build.xml</code>	Ant scribt for building and testing mkJPS.
<code>lib\mkjps.jar</code>	Contains all classes necessary to run mkJPS as shell script, batch-file or Ant task.
<code>...ools\mkjps\JavaPackageStructureGenerator.java</code>	The class, which implements the functionality.
<code>src\org\zahnleiter\tools\mkjps\mkJPS.java</code>	A main program, which can be called by shell scripts or batch files.
<code>src\org\zahnleiter\tools\mkjps\mkJPS4Ant.java</code>	This class allows mkJPS acting as an Ant task.

Table 2: Project files

B Suppliers

[S1] **The Mind Electric**
15455 Dallas Parkway
TX 75001 Addison
Texas
USA
www.themindelectric.com/

GLUE (SOAP toolkit), EXML (XML parser)

C Related web pages

[W1] **Apache Ant**
jakarta.apache.org/ant/index.html

A XML based make tool - GREAT!!!

Disclaimer

The use of my page's content (programs, wiring diagrams, pictures, documents) is free for non-commercial purposes only.

The information in this document has been carefully reviewed and is believed to be reliable, but I do not assume any liability arising out of the application or use of any documents, programs or circuit described herein.

Furthermore I want to declare that I'm not responsible in any way for the content of other web pages, books and other sources I'm referring to.