NAME

jmod − create JMOD files and list the content of existing JMOD files

SYNOPSIS

`jmod` (**`create`**|**`extract`**|**`list`**|**`describe`**|**`hash`**) [*options*] *jmod−file*

Includes the following:

Main operation modes

`create`
Creates a new JMOD archive file.

`extract`
Extracts all the files from the JMOD archive file.

`list` Prints the names of all the entries.

`describe`
Prints the module details.

`hash` Determines leaf modules and records the hashes of the dependencies that directly and indirectly require them.

Options

options See **Options for jmod**.

Required

jmod−file
Specifies the name of the JMOD file to create or from which to retrieve information.

DESCRIPTION

Note: For most development tasks, including deploying modules on the module path or publishing them to a Maven repository, continue to package modules in modular JAR files. The **`jmod`** tool is intended for modules that have native libraries or other configuration files or for modules that you intend to link, with the **jlink** tool, to a runtime image.

The JMOD file format lets you aggregate files other than **`.class`** files, metadata, and resources. This format is transportable but not executable, which means that you can use it during compile time or link time but not at run time.

Many **`jmod`** options involve specifying a path whose contents are copied into the resulting JMOD files. These options copy all the contents of the specified path, including subdirectories and their contents, but exclude files whose names match the pattern specified by the **`--exclude`** option.

With the **`--hash-modules`** option or the **`jmod hash`** command, you can, in each module's descriptor, record hashes of the content of the modules that are allowed to depend upon it, thus "tying" together these modules. This enables a package to be exported to one or more specifically−named modules and to no others through qualified exports. The runtime verifies if the recorded hash of a module matches the one resolved at run time; if not, the runtime returns an error.

OPTIONS FOR JMOD

`--class-path` *path*
Specifies the location of application JAR files or a directory containing classes to copy into the resulting JMOD file.

`--cmds` *path*
Specifies the location of native commands to copy into the resulting JMOD file.

`--config` *path*
Specifies the location of user−editable configuration files to copy into the resulting JMOD file.

`--dir` *path*
Specifies the location where **`jmod`** puts extracted files from the specified JMOD archive.

`--dry-run`
Performs a dry run of hash mode. It identifies leaf modules and their required modules without recording any hash values.

`--exclude` *pattern-list*
Excludes files matching the supplied comma-separated pattern list, each element using one the following forms:

- *glob-pattern*
- **`glob:`***glob-pattern*
- **`regex:`***regex-pattern*

See the **FileSystem.getPathMatcher** method for the syntax of *glob-pattern*. See the**P attern** class for the syntax of *regex-pattern*, which represents a regular expression.

`--hash-modules` *regex-pattern*
Determines the leaf modules and records the hashes of the dependencies directly and indirectly requiring them, based on the module graph of the modules matching the given *regex-pattern*. The hashes are recorded in the JMOD archive file being created, or a JMOD archive or modular JAR on the module path specified by the **`jmod hash`** command.

`--header-files` *path*
Specifies the location of header files to copy into the resulting JMOD file.

`--help` or **`-h`**
Prints a usage message.

`--help-extra`
Prints help for extra options.

`--legal-notices` *path*
Specifies the location of legal notices to copy into the resulting JMOD file.

`--libs` *path*
Specifies location of native libraries to copy into the resulting JMOD file.

`--main-class` *class-name*
Specifies main class to record in the module-info.class file.

`--man-pages` *path*
Specifies the location of man pages to copy into the resulting JMOD file.

`--module-version` *module-version*
Specifies the module version to record in the module-info.class file.

`--module-path` *path* or **`-p`** *path*
Specifies the module path. This option is required if you also specify **`--hash-modules`**.

`--target-platform` *platform*
Specifies the target platform.

`--version`
Prints version information of the **`jmod`** tool.

@filename
Reads options from the specified file.

An options file is a text file that contains the options and values that you would ordinarily enter in a command prompt. Options may appear on one line or on several lines. You may not specify environment variables for path names. You may comment out lines by prefixinga hash symbol (**#**) to the beginning of the line.

The following is an example of an options file for the **jmod** command:

```
#Wed Dec 07 00:40:19 EST 2016
create --class-path mods/com.greetings --module-path mlib
  --cmds commands --config configfiles --header-files src/h
  --libs lib --main-class com.greetings.Main
  --man-pages man --module-version 1.0
  --os-arch "x86_x64" --os-name "Mac OS X"
  --os-version "10.10.5" greetingsmod
```

EXTRA OPTIONS FOR JMOD

In addition to the options described in **Options for jmod**, the following are extra options that can be used with the command.

--do-not-resolve-by-default
: Exclude from the default root set of modules

--warn-if-resolved
: Hint for a tool to issue a warning if the module is resolved. One of deprecated, deprecated−for−removal, or incubating.

JMOD CREATE EXAMPLE

The following is an example of creating a JMOD file:

```
jmod create --class-path mods/com.greetings --cmds commands
  --config configfiles --header-files src/h --libs lib
  --main-class com.greetings.Main --man-pages man --module-version 1.0
  --os-arch "x86_x64" --os-name "Mac OS X"
  --os-version "10.10.5" greetingsmod
```

JMOD HASH EXAMPLE

The following example demonstrates what happens when you try to link a leaf module (in this example, **ma**) with a required module (**mb**), and the hash value recorded in the required module doesn't match that of the leaf module.

1. Create and compile the following **.java** files:
   - **jmodhashex/src/ma/module-info.java**

     ```
     module ma {
       requires mb;
     }
     ```

   - **jmodhashex/src/mb/module-info.java**

     ```
     module mb {
     }
     ```

   - **jmodhashex2/src/ma/module-info.java**

     ```
     module ma {
       requires mb;
     }
     ```

   - **jmodhashex2/src/mb/module-info.java**

     ```
     module mb {
     }
     ```

2. Create a JMOD archive for each module. Create the directories **jmodhashex/jmods** and **jmodhashex2/jmods**, and then run the following commands from the **jmodhashex** directory, then from the **jmodhashex2** directory:
   - **jmod create --class-path mods/ma jmods/ma.jmod**

- **jmod create --class-path mods/mb jmods/mb.jmod**

3. Optionally preview the **jmod hash** command. Run the following command from the **jmodhashex** directory:

   jmod hash --dry-run -module-path jmods --hash-modules .*

   The command prints the following:

   ```
   Dry run:
   mb
     hashes ma SHA-256 07667d5032004b37b42ec2bb81b46df380cf29e66962a16481ac
   ```

   This indicates that the **jmod hash** command (without the **--dry-run** option) will record the hash value of the leaf module **ma** in the module **mb**.

4. Record hash values in the JMOD archive files contained in the **jmodhashex** directory. Run the following command from the **jmodhashex** directory:

   ```
   jmod hash --module-path jmods --hash-modules .*
   ```

   The command prints the following:

   ```
   Hashes are recorded in module mb
   ```

5. Print information about each JMOD archive contained in the **jmodhashex** directory. Run the highlighted commands from the **jmodhashex** directory:

   ```
   jmod describe jmods/ma.jmod

   ma
     requires mandated java.base
     requires mb

   jmod describe jmods/mb.jmod

   mb
     requires mandated java.base
     hashes ma SHA-256 07667d5032004b37b42ec2bb81b46df380cf29e66962a16481ac
   ```

6. Attempt to create a runtime image that contains the module **ma** from the directory **jmodhashex2** but the module **mb** from the directory **jmodhashex**. Run the following command from the **jmodhashex2** directory:

   - **Linux and OS X:**

     ```
     jlink --module-path $JAVA_HOME/jmods:jmods/ma.jmod:../jmod-
     hashex/jmods/mb.jmod --add-modules ma --output ma-app
     ```

   - **Windows:**

     ```
     jlink --module-path %JAVA_HOME%/jmods;jmods/ma.jmod;../jmod-
     hashex/jmods/mb.jmod --add-modules ma --output ma-app
     ```

   The command prints an error message similar to the following:

   ```
   Error: Hash of ma (a2d77889b0cb067df02a3abc39b01ac1151966157a68dc4241562
   expected hash (07667d5032004b37b42ec2bb81b46df380cf29e66962a16481ace2e71
   ```