CHAPTER 12

■ ■ ■

Custom Runtime Images

QUESTIONS AND EXERCISES

1. What is a custom Java runtime image?

   Answer:

   A custom Java runtime image is a Java runtime image that contains an application's modules and only those Java modules that are used by the application.

   263-8470

2. What is the JIMAGE format?

   Answer:

   The JIMAGE format is a JDK-internal file format to package a Java runtime image. It is optimized for speed and space.

3. What is the `jlink` tool?

   Answer:

   `jlink` is a command-line tool to create a custom Java runtime image.

4. Why do you use the `--launcher` option with the `jlink` tool?

   Answer:

When you use the `jlink` tool to create a custom Java runtime image, specifying the --launcher option creates a script/batch file. You can use the script/batch file to launch your application. The `--launcher` option has the following two forms:

- `--launcher <command>=<module>`: Specifies the launcher command for the module. `<command>` is the name of the command you want to generate to launch your application, for example, `runmyapp`. The tool will create a scrip/batch file named `<command>` to run the main class recorded in <module>.
- `--launcher <command>=<module>/<main-class>`: Specifies the launcher command for the module and the main class. `<command>` is the name of the command you want to generate to launch your application, for example, `runmyapp`. The tool will create a script/batch file named `<command>` to run the `<main-class>` in `<module>`.

5. What is the effect of using or not using the `--bind-services` option with the `jlink` tool?

Answer:

In the custom runtime image, by default, the `jlink` tool includes the specified application modules with their dependence and only the needed platform modules. Services – the `uses` statements in module declaration – are not resolved. If a service provider module is resolved by the jlink tool as part of its `--add-module` option or as a dependency for another module, the service provider module will be included in the image.

When the `--bind-services` option is used, the `jlink` tool performs full service binding during the linking process. If the added modules contain uses statements, `jlink` will scan all modules on the module path to include all service provider modules in the runtime image for the service specified in the `uses` statement.

6. What are the plugins for the `jlink` tool?

Answer:

The plugins for the `jlink` tools are transformer. The `jlink` tool collects all classes, native libraries, and configuration files into a set of resources. It builds a pipeline of transformers, which are plugins specified as command-line options. Resources are fed into the pipeline. Each transformer in the pipeline applies some kind of transformation to resources and the transformed resources are fed to the next transformer. At the end, `jlink` feeds the transformed resources to an image builder.

7. How do you list the plugins available for `jlink`?

Answer:

Use the `--list-plugins` option to print the list of all available plugins:

```
jlink --list-plugins
```

8. Name two `jlink` plugins.

Answer:

Two `jlink` plugins are compress and strip-debug.

9. Can you use a custom plugin with `jlink`?

No

10. What is the `jimage` tool? Describe the use of the following four sub-commands for the `jimage` tool: `extract`, `info`, `list`, and `verify`.

Answer:

The `jimage` tool is used to explore the contents of Java runtime images - files in JIMAGE format.

Sub-Command	Description
`extract`	Extracts all entries from the specified JIMAGE files to the current directory. Use the `--dir` option to specify another directory for extracted entries.
`info`	Prints the detailed information contained in the header of the specified JIMAGE file.
`list`	Prints the list of all modules and their entries in the specified JIMAGE file. Use the `--verbose` option to include the details of the entries such as its size, offset, and whether the entry is compressed.
`verify`	Prints a list of `.class` entries in the specified JIMAGE files that do not verify as classes.