The Java™ Tutorials

Trail: Deployment
Lesson: Packaging Programs in JAR Files
Section: Using JAR-related APIs

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases.

The JarRunner Class

The JarRunner application is launched with a command of this form:

```
java JarRunner url [arguments]
```

In the previous section, we've seen how `JarClassLoader` is able to identify and load the main class of a JAR-bundled application from a given URL. To complete the JarRunner application, therefore, we need to be able to take a URL and any arguments from the command line, and pass them to an instance of `JarClassLoader`. These tasks belong to the `JarRunner` class, the entry point of the JarRunner application.

It begins by creating a `java.net.URL` object from the URL specified on the command line:

```
public static void main(String[] args) {
    if (args.length < 1) {
        usage();
    }
    URL url = null;
    try {
        url = new URL(args[0]);
    } catch (MalformedURLException e) {
        fatal("Invalid URL: " + args[0]);
    }
```

If `args.length < 1`, that means no URL was specified on the command line, so a usage message is printed. If the first command-line argument is a good URL, a new `URL` object is created to represent it.

Next, JarRunner creates a new instance of `JarClassLoader`, passing to the constructor the URL that was specified on the command-line:

```
JarClassLoader cl = new JarClassLoader(url);
```

As we saw in the previous section, it's through `JarClassLoader` that JarRunner taps into the JAR-handling APIs.

The URL that's passed to the `JarClassLoader` constructor is the URL of the JAR-bundled application that you want to run. JarRunner next calls the class loader's `getMainClassName` method to identify the entry-point class for the application:

```
String name = null;
try {
    name = cl.getMainClassName();
} catch (IOException e) {
    System.err.println("I/O error while loading JAR file:");
    e.printStackTrace();
    System.exit(1);
}
if (name == null) {
    fatal("Specified jar file does not contain a 'Main-Class'" +
          " manifest attribute");
}
```

The key statement is highlighted in bold. The other statements are for error handling.

Once `JarRunner` has identified the application's entry-point class, only two steps remain: passing any arguments to the application and actually launching the application. `JarRunner` performs these steps with this code:

```
// Get arguments for the application
String[] newArgs = new String[args.length - 1];
System.arraycopy(args, 1, newArgs, 0, newArgs.length);
// Invoke application's main class
try {
```

```
            cl.invokeClass(name, newArgs);
        } catch (ClassNotFoundException e) {
            fatal("Class not found: " + name);
        } catch (NoSuchMethodException e) {
            fatal("Class does not define a 'main' method: " + name);
        } catch (InvocationTargetException e) {
            e.getTargetException().printStackTrace();
            System.exit(1);
        }
```

Recall that the first command-line argument was the URL of the JAR-bundled application. Any arguments to be passed to that application are therefore in element `1` and beyond in the `args` array. `JarRunner` takes those elements, and creates a new array called `newArgs` to pass to the application (bold line above). `JarRunner` then passes the entry-point's class name and the new argument list to the `invokeClass` method of `JarClassLoader`. As we saw in the previous section, `invokeClass` will load the application's entry-point class, pass it any arguments, and launch the application.

Previous page: The JarClassLoader Class
Next page: Questions and Exercises: JAR