## 3.4 Servlet Packaging

In a production environment, multiple programmers can be developing servlets for the same server. So, placing all the servlets in the same directory results in a massive, hard-to-manage collection of classes and risks name conflicts when two developers inadvertently choose the same name for a servlet or a utility class. Now, Web applications (see Section 2.11) help with this problem by dividing things up into separate directories, each with its own set of servlets, utility classes, JSP pages, and HTML files. However, since even a single Web application can be large, you still need the standard Java solution for avoiding name conflicts: packages. Besides, as you will see later, custom classes used by JSP pages should *always* be in packages. You might as well get in the habit early.

When you put your servlets in packages, you need to perform the following two additional steps.

1. **Place the files in a subdirectory that matches the intended package name.** For example, we'll use the `coreservlets` package for most of the rest of the servlets in this book. So, the class files need to go in a subdirectory called `coreservlets`. Remember that case matters for both package names and directory names, regardless of what operating system you are using.

2. **Insert a package statement in the class file.** For instance, for a class to be in a package called `somePackage`, the class should be in the `somePackage` directory and the *first* non-comment line of the file should read

   ```
   package somePackage;
   ```

   For example, Listing 3.4 presents a variation of the `HelloServlet` class that is in the `coreservlets` package and thus the `coreservlets` directory. As discussed in Section 2.8 (Test Your Setup), the class file should be placed in `*install_dir*/webapps/ROOT/WEB-INF/classes/coreservlets` for Tomcat, `*install_dir*/servers/default/default-ear/default-war/WEB-INF/classes/coreservlets` for JRun, and `*install_dir*/doc/WEB-INF/classes/coreservlets` for Resin. Other servers have similar installation locations.

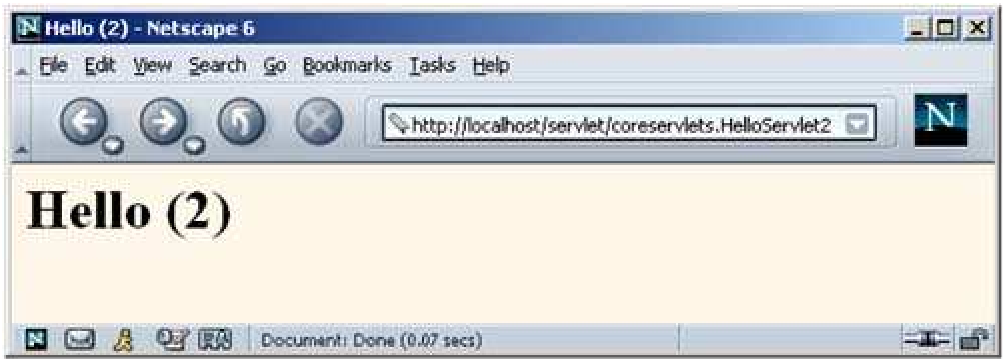
Figure 3-4 shows the servlet accessed by means of the default URL.

### Listing 3.4 coreservlets/HelloServlet2.java

```
package coreservlets;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

/** Simple servlet for testing the use of packages. */

public class HelloServlet2 extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String docType =
      "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.0 " +
      "Transitional//EN\">\n";
```

```
    out.println(docType +
                "<HTML>\n" +
                "<HEAD><TITLE>Hello (2)</TITLE></HEAD>\n" +
                "<BODY BGCOLOR=\"#FDF5E6\">\n" +
                "<H1>Hello (2)</H1>\n" +
                "</BODY></HTML>");
  }
}
```

**Figure 3-4. Result of `http://localhost/servlet/coreservlets.HelloServlet2`.**


Hello (2) - Netscape 6
File Edit View Search Go Bookmarks Tasks Help
http://localhost/servlet/coreservlets.HelloServlet2
Document: Done (0.07 secs)

Hello (2)