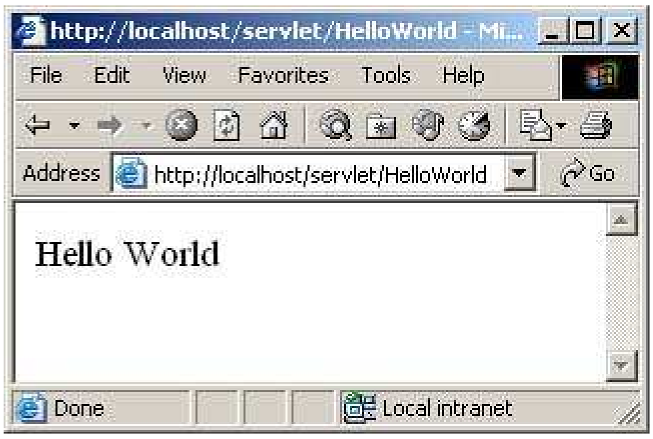
3.2 A Servlet That Generates Plain Text

Listing 3.2 shows a simple servlet that outputs plain text, with the output shown in Figure 3-2. Before we move on, it is worth spending some time reviewing the process of installing, compiling, and running this simple servlet. See Chapter 2 (Server Setup and Configuration) for a much more detailed description of the process.

Figure 3-2. Result of `http://localhost/servlet/HelloWorld`.


http://localhost/servlet/HelloWorld - Mi...
File Edit View Favorites Tools Help
Address http://localhost/servlet/HelloWorld
Go
Done
Local intranet

Hello World

First, be sure that you've already verified the basics:

- That your server is set up properly as described in Section 2.3 (Configure the Server).
- That your development `CLASSPATH` refers to the necessary three entries (the servlet JAR file, your top-level development directory, and ".") as described in Section 2.7 (Set Up Your Development Environment).
- That all of the test cases of Section 2.8 (Test Your Setup) execute successfully.

Second, type "`javac HelloWorld.java`" or tell your development environment to compile the servlet (e.g., by clicking Build in your IDE or selecting Compile from the emacs JDE menu). This step will compile your servlet to create `HelloWorld.class`.

Third, move `HelloWorld.class` to the directory that your server uses to store servlets that are in the default Web application. The exact location varies from server to server, but is typically of the form *`install_dir`*`/.../WEB-INF/classes` (see Section 2.10 for details). For Tomcat you use *`install_dir`*`/webapps/ROOT/WEB-INF/classes`, for JRun you use *`install_dir`*`/servers/default/default-ear/default-war/WEB-INF/classes`, and for Resin you use *`install_dir`*`/doc/WEB-INF/classes`. Alternatively, you can use one of the techniques of Section 2.9 (Establish a Simplified Deployment Method) to automatically place the class files in the appropriate location.

Finally, invoke your servlet. This last step involves using either the default URL of `http://`*`host`*`/servlet/`*`ServletName`* or a custom URL defined in the `web.xml` file as described in Section 2.11 (Web Applications: A Preview). During initial development, you will almost certainly find it convenient to use the default URL so that you don't have to edit the `web.xml` file each time you test a new servlet. When you deploy real applications, however, you almost always disable the default URL and assign explicit URLs in the `web.xml` file (see Section 2.11, "Web Applications: A Preview"). In fact, servers are not absolutely required to support the default URL, and a few, most notably BEA WebLogic, do not.

Figure 3-2 shows the servlet being accessed by means of the default URL, with the server running on the local machine.

Listing 3.2 HelloWorld.java

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

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");
  }
}
```