Java servlets: Extending your Domino applications

by Julie Forgo

[Editor's note: This article resides in "Iris Today", the technical Webzine located on the http://www.notes.net Web site produced by Iris Associates, the developers of Domino/Notes.]

Applets took the Web world by storm with an easy way to bring Web pages to life with spinning globes, blinking text, and a whole wave of flashy effects. Now, move over applets, and make room for servlets, the behind-the-scenes cousins that are delivering less flash, but plenty of power on the server-side of the show. This article will show you how to add servlets to extend the power of your Domino applications.

Servlets are server-side Java components that offer an alternative to CGI and PERL programming for building Web applications. Servlets generally perform better and are more easily extensible than CGI programs. You can use servlets for a wide range of applications -- for example, for creating Web pages that are dynamically updated, or for connecting to and exchanging data between applications. For more information on the benefits of using servlets, visit the Sun Microsystems site at **http://www.sun.com/java**.

A key difference between Java server agents and Java servlets is the scope of each. An agent loads when requested, runs, then unloads. A servlet loads when the server starts or when it is requested, and the code stays resident until the server shuts down. So, for example, for a task such as connecting to a database, you might choose a servlet to maintain a persistent connection that allows for ongoing data transfer without reloading code. For information on when and how to use a Java agent, see Bob Balaban's article "**Tips on Debugging Java Agents**."

Note: For Domino 4.6, servlets are not supported on OS/2 or NT/Alpha. They are supported on all other server platforms.

Where do servlets come from?

You can think of a servlet as a Java applet that runs on the server rather than on a Web page or in a document. As with Java applets, you can create your own servlet, refer to an existing servlet by its URL, or use a pre-built one. Because servlets are a fairly new technology, there are not very many pre-built servlets to choose from yet. Also, because they are commonly used for connecting two applications -- for example, Notes and a legacy database -- they are more likely to be custom-built.

To examine or try out existing servlets, use the samples included with the Java Software Development Kit (JSDK) or visit the Web sites **http://jserv.javasoft.com** and **http://javashareware.com**. Remember that you must restart the server to load a new servlet or to see changes to a modified servlet.

Writing a servlet from scratch

Domino 4.6 supports the JavaSoft servlet classes and interfaces. To write a servlet, you extend the Servlet class and define Java methods for establishing and managing connections. Servlets offer valuable features such as thread-safe code, automatic memory management, and built-in networking support.

To write a servlet, you extend the base Servlet class:

Javax.servlet.http.HttpServlet
Javax.servlet.GenericServlet

Then you define methods to describe what the servlet does. Some of the more common methods are:

- doGet() and doPost() to handle GET and POST requests.
- getServiceInfo() to provide servlet descriptions

- getLastModified() for conditional GETS
- init() for servlet initialization code
- service() to handle raw requests

For complete documentation on servlet methods and properties, refer to the Java servlet documentation at: **http://jserv.javasoft.com/products/java-server/servlets**

Before you compile the servlet code, the servlet classes need to be added to the CLASSPATH environment variable. (Domino does this for you automatically when you enable Java servlet support on the server -- which you'll learn about later.) Then, you can compile the servlet code using your favorite Java compiler.

Example: the HelloWorld Servlet

The following code illustrates the structure of a simple servlet -- the obligatory HelloWorld example.

```
// HelloWorldServlet -- A servlet that just says hi
public class HelloWorldServlet extends HttpServlet {

    public void doGet (HttpServletRequest req, HttpServletResponse res) throws ServletException,
    IOException
    {
        res.setContentType("text/html");
        ServletOutputStream out = res.getOutputStream();
        out.println("<html>");
        out.println("<head><title>Hello World</title></head>");
        out.println("<body>");
        out.println("<h1>Hello World</h1>");
        out.println("</body></html>");
    }

    public String getServletInfo() {
        return "Create a page that says <i>Hello World</i> and send it back";
    }
}
```

Using Java servlets with the Domino server

To use a Java servlet with Domino, you need to:

1. Enable Java servlet support.
2. Install the servlet.
3. Register the servlet.
4. Access the servlet via the Web.

Step 1: Enabling Java servlet support

In Domino, support for servlets is disabled by default. To enable servlet support, edit the server's Notes.ini file and add the following line:

```
DominoEnableJavaServlets=1
```

When you restart the server, it loads the Java Virtual Machine and locates the ServletManager Java class, adding the file icsclass.jar to the CLASSPATH environment variable automatically.

As the servlet support is loading, the following three lines appear on the server console:

```
01/01/98 11:23:36 AM HTTP Web Server started
01/01/98 11:23:38 AM JVM: Java Virtual Machine initialized
01/01/98 11:23:38 AM Java Servlet Manager initialized
```

Step 2: Installing the servlet
To install a servlet, whether it is one you wrote or a pre-built one, you create a Servlets subdirectory in the Domino server data directory and copy the compiled servlets into this directory. Then edit the server's Notes.ini file and add a setting for JavaUserClasses to include a path for this subdirectory. For example:

```
JavaUserClasses=c:\lotus\notes\data\domino\servlets
```

Step 3: Registering the servlet
In order to access a servlet from the Web, it must be registered in a Domino servlet configuration file, Servlet.cnf, that you create in the Domino data directory. When the server starts up, it checks this configuration file to see if there are any servlets to load. In the servlet configuration file, you must specify each servlet by name. Optionally, you can also specify initialization parameters, a servlet class for URL mapping, and instructions for loading the servlet at startup time.

The following example registers the Hello World servlet with no parameters and maps the servlet to a URL:

```
# -- Servlet.cnf
# Register a simple servlet
    Servlet HelloWorld Servlet {
    }
#Map the servlet to a specific URL
    Service HelloWorld /Servlet/Hello
```

The next example registers a servlet called DBAccess that does include initialization parameters and that loads at startup.

```
# -- Servlet.cnf
# Register a simple servlet
    Servlet DBAccessServlet {
    Server=w3.internat.net
    Path=/corp2/ordbsh
    Acct=JoeUser
    GO_LOAD_AT_STARTUP=Yes
    }
#Map the servlet to a specific URL
    Service DBAccess /Servlet/DBServe
```

Step 4: Accessing a servlet from the Web
Once the servlet is registered and mapped to a specific URL, you can access the servlet with that URL, using full URL syntax. For example, entering the URL http://myserver/Servlet/Hello runs the HelloWorld servlet registered in the previous step.

Trying it for yourself

To try out the HelloWorld servlet for yourself:

1. Follow step 1 to enable Java servlet support.
2. Create a servlets subdirectory where Domino is installed and copy the HelloWorldServlet.class file below into it.
3. Copy the SERVLET.CNF file below into your Domino data directory.
4. Startup the Web Server and try the following URL:

   http://127.0.0.1/Servlet/HelloWorldServlet

5. If you see "HelloWorld" in a glorious large font in your browser window, it works!


HelloWorldServlet.clas


servlet.cnf

Summary

Java servlets are an emerging technology that can add power to your Domino applications. Although they may not be as flashy as applets, servlets allow you to do more on the server-side of things, such as maintaining connections for exchanging data between applications. Plus, servlets feature thread-safe code, automatic memory management and built-in networking support. Servlets and Domino make a powerful combination -- the proof is in the programming.

ABOUT THE AUTHOR

Julie Forgo, a technical writer with more than ten years' experience, has documented various aspects of Notes for the past four years. Her area of expertise is Domino application development, and she recently co-authored a revision of the *Best Practices: Application Developer's Guide*. When not busy extracting feature details from developers, Julie chases after her two children and performs miracles with a hot-glue gun.