Server side Java with Apache/iX

» Return to original page

(please notice that I am only *tasting* software, not *testing* it ;-)

When I recently played with a trial version of Java Web Server from Sun (on MPE/iX, of course), I learned a little bit about the concept of Servlets for implementing server side functionality in Java. I found the servlet technology and associated API quite interesting, not just because of its underlying concept and architecture, but also because it allowed me -still a Java beginner- to implement a few examples of using server side Java to generate dynamic web pages with amazingly small effort. So I thought, I'd write up this little piece to share some of the experiences with you. Last, but not least, because it also can be used with Apache/iX - but we'll cover that later.

What the h* is a servlet?**

A servlet is basically a user-supplied extension of the web server's default functionality. Like an Applet is a piece of Java bytecode that is executed inside the web browser to implement some functionality on the client side, a Servlet is a piece of Java bytecode that is executed inside the web server to implement some functionality on the server side. In typical cases it would be used to generate web pages with dynamic content on-the-fly or provide some web interface to a server side application.

The servlet technology is just another means to avoid the classic CGI child process implementation and its associated overhead for short-lived process creation as well as hassle with state management to make a sequence of inherently independent http requests form some kind of logical transaction from the user perspective.

The Java Web Server invokes a Servlet for handling selected http requests. It simply calls the appropriate method from the Servlet API, passing parameters that give access to the http request data (like `QUERY_STRING`, `PATH_INFO`, cookies, or input fields of an HTML form) as well as allow sending a response back to the client. This does not involve a child process creation, it is just a kind of subroutine call, if you a allow me this non-object oriented term here. Just like the QWEBS web server, for example, can call a user-supplied COBOL subroutine from an XL. The servlet code is loaded and initialized once and then stays alive for handling multiple requests (with the option of multi-threading for concurrent requests).

Where does Apache/iX come into play?

Unlike the Java Web Server from Sun, Apache is not written in Java and thus cannot simply call a Java "subroutine" (without invoking some kind of Java Virtual Machine first). But do not despair, we live in freeware land here! There is a project called JServ for Apache, that essentially brings Java Servlet capabilities to this popular web server.

The implementation is quite similar to FastCGI (in case you should know that other interesting technology) i.e. the Apache server is enhanced with a mod_jserv module which allows the httpd server processes to talk to a JServ servlet engine using a TCP connection. The servlet engine is written in Java and handles the servlet loading and invocation. You can run one or more JServ engines on different TCP port numbers or even different machines! So you can, for example run the Apache web server on Unix or Linux or NT (if you should feel a need to do so, for example for load balancing, independent hardware scaling, network isolation or whatever reason) and the JServ application server on MPE/iX, where the data lives.

It might even be possible to use Stronghold, the commercial "Apache with strong encryption SSL", in such a setup... (but I haven't tried).

The nice thing about Apache and JServ is that they both build on MPE/iX "out of the box"! Believe me that it was a pleasant surprise for me to download Apache 1.3.4 from www.apache.org and JServ

1.0b3 from java.apache.org and be able to compile, link and integrate both of them without having to change a single line of code in their sources (okay, except for the usual MPE specific adjustments to the GNU autoconf based `configure` script). I did have some hope that this would be true for the JServ engine, as it is written in pure Java, but it also turned out to hold true for the mod_jserv module for Apache, which is written in C. Getting FastCGI from fastcgi.idle.com built for Apache/iX was a little bit more work (but that's another story, although FastCGI also does allow implementing in Java, besides C, Perl, etc).

Gimme some examples, please!

You can find a lot of reading materials about servlet technology and the Servlet API at the java.sun.com web site. I found their servlet tutorial quite helpful (hey, it did give me enough intro to enable me creating these examples). If you download the Servlet Developer Kit, you will have materials like tutorial and API docs available for local (offline) reading. The JSDK also contains a *servletrunner* program to help you test servlets without a full-blown Java Web Server or Apache JServ at hand. Pretty much like *appletviewer* allows you to test Applets outside of a Java capable web browser.

Here are my first three examples of Java Servlets. Please be merciful with me as I am still a beginner in the Java language. The programs -oops- servlets will certainly have lots of room for improvement, but should at least be sufficient to give you an idea and a point for getting started with your own experiments. Feel free to share your favourite examples with me and other interested people!

- Hello World

  What else would you expect as a first piece of code? Okay, this one has a hit counter built in to show that the servlet indeed is loaded only once and from then on simply invoked to handle client requests. Please notice that I have not yet paid attention to making this counter thread-safe.

- CobCGI look-alike (and the associated HTML form)

  You might know my CGI example in COBOL/iX and related tips regarding use of `READX` or `PRINT` intrinsics as well as `getenv()` and similar calls to the Posix C library. Here is the same demo implemented as Java Servlet, so you can see the differences or compare them side-by-side. You might notice that the COBOL example did not yet have code to translate URL-encoded input fields back to "plain text" (i.e. replace occurences of + by space and %xx by the respective special char). The servlet does not need such code as the Servlet API already provides very handy methods for retrieving parsed http request parameters!

- JDBC to MusicDBE

  You certainly know the little Music database from "Getting Started with IMAGE/SQL". If not, then hurry to have a look at the command file `IMSQL.SAMPLEDB.SYS` to unpack your personal copy into your current group! By attaching that little TurboIMAGE database to an SQL DBE it becomes accessible for JDBC (Java DataBase Connectivity) - if you have already installed a JDBC driver for Allbase and IMAGE/SQL on your 3000 or PC or Mac or whatever platform.

  While I have never programmed with ODBC (hey, I am still not intending to become a PC freak ;-) it seemed amazingly easy to me to access an SQL database using the `java.sql` API. This servlet creates a "home page" with links to two pages that show the result of an on-the-fly SQL query. Notice the hit counters (again not thread-safe) to prove that the pages are "live" (or give you a hint to press "reload" if your browser shows a cached version only).

This whole paper (how can this web page be called a paper without having been printed? ;-) is a fairly quick write-up to spread the news and share experiences. Feedback is welcome, as usual. It might, for example be helpful to add a session log of how I unpacked and built Apache 1.3.4 and JServ 1.0b3 on my 3000. Or maybe a downloadable binary? Or maybe some snippets from the Apache and JServ config files? Just let me know. And keep in mind that I cannot promise a guaranteed response time :-) See you on HP3000-L.

Lars Appel, February 1999

» Return to original page