Embedding Jetty

Embedding Jetty

Quick Start

Quick Start - Handlers

The following code implements an anonymous hello handler and starts a jetty server with it:

```
Handler handler=new AbstractHandler()
{
    public void handle(String target,
HttpServletRequest request,
HttpServletResponse response, int dispatch)
        throws IOException, ServletException
    {

response.setContentType("text/html");

response.setStatus(HttpServletResponse.SC_OK
);

response.getWriter().println("<h1>Hello</h1>
");
        ((Request)request).setHandled(true);
    }
};

Server server = new Server(8080);
server.setHandler(handler);
server.start();
```

Quick Start - Servlets

Here is a simple example of embedding Jetty using a Hello world servlet:

```
Server server = new Server(8080);
Context root = new
Context(server,"/",Context.SESSIONS);
root.addServlet(new ServletHolder(new
HelloServlet("Ciao")), "/*");
server.start();
```

Quick Start - jetty.xml

You can programatically create a jetty Server, and still use a jetty.xml file to configure it:

```
Server server = new Server();
XmlConfiguration configuration = new
XmlConfiguration(new
File("myJetty.xml").toURL()); //or use new
XmlConfiguration(new
FileInputStream("myJetty.xml"));
configuration.configure(server);
server.start();
```

Quick Start - Servlets and .jsp pages (hosted within the same .jar as the embedder)

Here is a simple example of embedding Jetty using a Hello world servlet:

```
// assumes that this directory contains
.html and .jsp files
// This is just a directory within your
source tree, and can be exported as part of
your normal .jar
```

```
final String WEBAPPDIR =
"com/xxx/yyy/webapp";

final Server server = new
Server(httpServerPort);


final String CONTEXTPATH = "/admin";

// for localhost:port/admin/index.html and
whatever else is in the webapp directory
final URL warUrl =
this.class.getClassLoader().getResource(WEBA
PPDIR);
final String warUrlString =
warUrl.toExternalForm();
server.setHandler(new
WebAppContext(warUrlString, CONTEXTPATH));

// for localhost:port/servlets/cust, etc.
final Context context = new Context(server,
"/servlets", Context.SESSIONS);
context.addServlet(new ServletHolder(new
CustomerServlet(whatever)), "/cust");
context.addServlet(new ServletHolder(new
```

```
UserServlet(whatever)), "/user");

server.start();
```

More Examples

These examples are all included as part of the standard Jetty distribution in the $JETTY_HOME/examples/embedded sub project.

- FileServer - simple HTTP file server
- OneHandler - Embed a single handler - useful when there is only a single simple source of content.
- ManyHandlers - Embed multiple handlers, one called after the other on each request.
- OneContext - An example of a handler within a context, which allows a contextPath, resourceBase and class loader to be set.
- MinimalServlets - An example of a ServletHandler without a context.
- OneServletContext - A servletHandler within a context, which has been constructed with a session handler.
- ManyServletContexts - Multiple servlet contexts using context path to select the context.
- LikeJettyXml - a code example that mimics the configuration of the standard jetty.xml file
- FromXmlConfiguration - a example that shows how XML fragments may be used to build a server and/or context.
- OneWebApp - an example showing a single webapp being added programmatically.

Setting up the classpath

To run Jetty you need only the following jars on the classpath:

- servlet-api-2.5-6.x.jar
- jetty-util-6.x.jar
- jetty-6.x.jar

This gives you the capability to handle HTTP with handlers, servlets and webapplications.
Additional features such as JSP and AJP require additional jars (normally found in subdirectories of $JETTY_HOME/lib).

Java Server Pages

JSP2.1

> **Remember!**
>
> You need to use J2SE5 (aka jdk 1.5) if you wish to use JSP2.1.

This is the jsp version mandated by servlet specification 2.5. You will need these jars:

- ant-1.6.5.jar
- core-3.1.1.jar
- jsp-2.1.jar
- jsp-api-2.1.jar

JSP2.0

JSP 2.0 is used with version 2.4 of the servlet specification. Depending on the functionality you require, you may be able to use this version rather than the newer JSP2.1
These are the dependencies when JSP 2.0 is used:

- ant-1.6.4.jar
- jasper-compiler-5.5.15.jar
- jasper-runtime-5.5.15.jar
- jsp-api-2.0.jar
- commons-el-1.0.jar
- jcl104-over-slf4j-1.0-rc5.jar
- slf4j-simple-1.0-rc5.jar
- xmlParserAPIs-2.6.2.jar
- xercesImpl-2.6.2.jar