

A look at the start.jar mechanism

A Look at the start.jar mechanism

You're probably familiar with the handy start.jar way of [running jetty](#):

```
java -jar start.jar [config file list]
```

Let's take a look at the way start.jar works and how you can customize it (if you need to).

The start.config file

The key to the mechanism is the `start.config` file. This file does a number of housekeeping chores such as setting up classpaths and System properties. The default `start.config` file is found in `start.jar/org/mortbay/start/start.config`.

When invoked from the command line as `-jar start.jar`, the jvm looks inside the `META-INF/MANIFEST.MF` file of the jar for the entry:

```
Main-Class: org.mortbay.start.Main
```

It then invokes the `main()` method of the named class which in this instance is `org.mortbay.start.Main`, which in turn parses the `start.config` file and invokes the `main()` method of the class named therein. It's a little clearer when we look at an [#example](#), but first let's check out the syntax of `start.config`.

Syntax

Each line contains an entry of the form:

```
SUBJECT [ [!] CONDITION [AND|OR] ]*
```

where SUBJECT:

- ending with ".class" is the Main class to run.
- ending with ".xml" is a configuration file for the command line
- ending with "/" is a directory from which to add all jar and zip files.
- ending with "/*" is a directory from which to add all unconsidered jar and zip files.
- ending with "**" is a directory from which to recursively add all unconsidered jar and zip files.
- containing an "=" are used to assign system properties.
- all other subjects are treated as files to be added to the classpath.
- SUBJECT may include system properties using `$(propertyname)` syntax.

Files starting with "/" are considered absolute, all others are relative to the home directory.

where `CONDITION` is one of:

- "always" (always **true**)
- "never" (always **false**)
- "available" classname (**true** iff classname is on the classpath)
- property name (**true** if property is set)
- "java" OPERATOR version (java version compared to literal)
- nargs OPERATOR number (number of command line args compared to literal)

where OPERATOR:

- is one of "<", ">", "<=", ">=", "==", "!="

`CONDITION` can be combined with "AND", "OR" or "!", with "AND" being the assumed operator for a list of `CONDITION`.

Classpath operations are evaluated on the fly, so once a class or jar is added to the classpath, subsequent available conditions will see that class.

Example

Here's an example `start.config` file that is, in fact, taken from a recent Jetty distribution.

```
$(jetty.class.path)
always

# Try different settings of jetty.home until
the jetty.jar is found.
jetty.home=.
! exists $(jetty.home)/lib/jetty.jar
jetty.home=..
! exists $(jetty.home)/lib/jetty.jar
jetty.home=/home/jetty
! exists $(jetty.home)/lib/jetty.jar
jetty.home=/C:/jetty
! exists $(jetty.home)/lib/jetty.jar
jetty.home=.
! exists $(jetty.home)/lib/jetty.jar
```

```
# The main class to run after this
org.mortbay.xml.XmlConfiguration.class

# The default configuration files if none
are specified on the command line
$(jetty.home)/etc/jetty.xml
nargs == 0

# javax.servlet classes
$(jetty.home)/lib/servlet-api-2.5.jar
! available javax.servlet.Servlet

# Set the jetty classpath
$(jetty.home)/lib/*
always

# Set the classpath for the supporting cast
$(jetty.home)/lib/jsp-2.1/*
java >= 1.5
$(jetty.home)/lib/jsp-2.0/*
! available
org.apache.jasper.servlet.JspServlet
$(jetty.home)/lib/management/*
$(jetty.home)/lib/naming/*
$(jetty.home)/lib/plus/*
$(jetty.home)/lib/xbean/*

# Recursively add all jars and zips from the
ext lib
$(jetty.home)/lib/ext/**
always
```

```
# Try some standard locations for anything  
missing.
```

```
/usr/share/java/ant.jar
```

```
! available org.apache.tools.ant.Main
```

```
# Add a resources directory if it is there
$(jetty.home)/resources/
```

Customizing start.config

You have two choices. You can unjar the `start.jar` file and extract the `start.config` file, make your changes and then re-jar it. Or, you can create your own `start.config` file and then reference it on the command line like so:

```
java -DSTART=/my/dir/start.config -jar
start.jar
```

If you only wish to change the class whose `main()` method gets invoked after `start.config` has been parsed, then you can just set the `jetty.server` property instead:

```
java -Djetty.server=com.acme.my.Main
```

Warning

Think carefully before changing the main class. The default setup ensures that any configuration files passed on the command line are parsed and Jetty is started correctly.