

# J2se6HttpServerSPI

## What is this ?

The JAX-WS 2.x reference implementation (aka [Metro](#), included in J2SE 6) ships with an embedded web server that handles [Endpoint](#) publishing. This web server can be replaced by Jetty thanks to the contributed [Java HTTP Server SPI](#) implementation.

This extension has been written not only to workaround the internal Sun HttpServer problem (<http://forums.java.net/jive/message.jspa?messageID=130831>) but also to allow tight integration of JAX-WS web services and servlets.

## Prerequisites

- [svn](#)
- [mvn](#) ( preferably at least 2.0.4 )
- [jdk 1.6](#)

## Obtaining the Source

The project uses [Subversion](#) as it's source code control system.

The source can be checked out anonymously using:

```
svn co
http://svn.codehaus.org/jetty-contrib/trunk/
j2se6
```

## Building

The project uses maven2 as its build tool.

Go to the top level directory of the project and type:

```
mvn -DJ2SE6_HOME=/opt/jdk1.6.0 clean install
```

This will build jetty6 locally, copy the jars and artifacts into the correct directories and put snapshot versions of the artifacts into your local maven repository.

## Installing

Copy the freshly-built **j2se6-6.1-SNAPSHOT.jar** archive to **\$JETTY\_HOME/lib/ext**. As long as this JAR is on the classpath, the JVM should pick up the Jetty implementation instead of the default one.

## Configuring

By default, an independent Jetty server will be started no matter if there is already an instance running. This is a good choice if you just want to write a standalone application publishing web services via Jetty's web stack.

You can however tell the provider to reuse an existing server by setting the static **server** property on the provider's implementation in the **jetty.xml** configuration file:

```
<Configure id="Server"
class="org.mortbay.jetty.Server">

    <Call
class="org.mortbay.jetty.j2se6.JettyHttpServer
erProvider" name="setServer">
        <Arg><Ref id="Server"/></Arg>
    </Call>

    ...

</Configure>
```

You absolutely need to have a `ContextHandlerCollection` in your servers handlers chain. Here's an example one, extracted from the default **jetty.xml**:

```

<Set name="handler">
  <New id="Handlers"
class="org.mortbay.jetty.handler.HandlerColl
ection">
    <Set name="handlers">
      <Array
type="org.mortbay.jetty.Handler">
        <Item>
          <New id="Contexts"
class="org.mortbay.jetty.handler.ContextHand
lerCollection"/>
        </Item>
        <Item>
          <New id="DefaultHandler"
class="org.mortbay.jetty.handler.DefaultHand
ler"/>
        </Item>
        <Item>
          <New id="RequestLog"
class="org.mortbay.jetty.handler.RequestLogH
andler"/>
        </Item>
      </Array>
    </Set>
  </New>
</Set>

```

The provider will then add new contexts to the specified server eventually binding a new Connector if the port specified in the endpoint is not yet bound by Jetty.