Documentación de integración de pruebas Web con Selenium

Contents

[Pasos para ejecutar pruebas con Selenium en una aplicación nueva 2](#_Toc327372956)

[Supuestos 2](#_Toc327372957)

[Requisitos 2](#_Toc327372958)

[Pasos 2](#_Toc327372959)

[Generación de reportes HTML 5](#_Toc327372960)

[Correr pruebas de Selenium usando archivos HTML 5](#_Toc327372961)

[Agregar ejecución automática del servidor Jetty 7](#_Toc327372962)

[Extender la clase SeleneseTestBase 8](#_Toc327372963)

[Errores comunes 8](#_Toc327372964)

[El servidor Selenium está corriendo 8](#_Toc327372965)

# Pasos para ejecutar pruebas con Selenium en una aplicación nueva

## Supuestos

* La nueva aplicación es un proyecto mavenizado, es decir, con capacidades Maven.
* Los pasos funcionan con cualquier Framework Web, siempre y cuando sea un proyecto Maven siguiendo la estructura convencional.

## Requisitos

* Java 1.5 o superior, preferentemente el 6
* Tomcat 5.5 o superior, preferentemente el 6
* Maven 2.2.x o superior, preferentemente el 3.x.x
* Un cliente de SVN, preferentemente si es un plugin para el IDE, como Subclipse para Eclipse.

## Pasos

1. Descargué el proyecto tutorial **petclinic** del repositorio público de Spring. <https://src.springframework.org/svn/spring-samples/petclinic/trunk>
2. Compilé y generé el war de la aplicación con el plugin maven-war-plugin. El war generado es **petclinic.war** de acuerdo a la configuración ya definida en el pom.xml.

>mvn war:war

1. Desplegué la aplicación en Tomcat.
2. Comprobé que funcione entrando a <http://localhost:8082/petclinic>
3. Agregué las siguientes dependencias:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.8</version>

</dependency>

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>2.4.0</version>

<scope>test</scope>

</dependency>

1. Agregué el plugin selenium-maven-plugin que se encarga de descargar, iniciar y parar el servidor Selenium:

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>selenium-maven-plugin</artifactId>

<version>2.3</version>

<executions>

<execution>

<id>start</id>

<phase>pre-integration-test</phase>

<goals>

<goal>start-server</goal>

</goals>

<configuration>

<background>true</background>

<logOutput>true</logOutput>

<multiWindow>true</multiWindow>

</configuration>

</execution>

<execution>

<id>stop</id>

<phase>post-integration-test</phase>

<goals>

<goal>stop-server</goal>

</goals>

</execution>

</executions>

</plugin>

1. Agregué la ejecución integration-tests al plugin de Surefire. Aquí se configuran las pruebas que se deberán ejecutar. Incluí pruebas en fase de pruebas de integración (integration-test) que están en el paquete \*\*/selenium/ y son nombrados con el postfijo Tests.java.

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<configuration>

<includes>

<include>\*\*/samples/\*Tests.java</include>

</includes>

<excludes>

<exclude>\*\*/selenium/\*Tests.java</exclude>

</excludes>

</configuration>

<executions>

<execution>

<id>integration-tests</id>

<phase>integration-test</phase>

<goals>

<goal>test</goal>

</goals>

<configuration>

<skip>false</skip>

<excludes> <exclude>\*\*/samples/\*Tests.java</exclude>

</excludes>

<includes> <include>\*\*/selenium/\*Tests.java</include>

</includes>

</configuration>

</execution>

</executions>

</plugin>

1. Agregué al código fuente la siguiente clase, que es una prueba generada desde el IDE de Selenium usando la exportación del test suite con **JUnit4 (Remote Control)**.

**package** com.dextratech.test.selenium;

**import** com.thoughtworks.selenium.\*;

**import** org.junit.After;

**import** org.junit.Before;

**import** org.junit.Test;

**import** java.util.regex.Pattern;

**public** **class** Prueba1Tests **extends** ~~SeleneseTestCase~~ {

@Before

**public** **void** setUp() **throws** Exception {

selenium = **new** DefaultSelenium("localhost", 4444, "\*chrome", "https://www.google.com.mx/");

selenium.start();

}

@Test

**public** **void** testSuiteJunit() **throws** Exception {

selenium.open("/#hl=es-419&gs\_nf=1&cp=7&gs\_id=39&xhr=t&q=selenium&pf=p&output=search&sclient=psy-ab&oq=seleniu&aq=0&aqi=g4&aql=&gs\_l=&pbx=1&bav=on.2,or.r\_gc.r\_pw.r\_qf.,cf.osb&fp=fd13c0df7b475bac&biw=982&bih=632");

selenium.type("id=gbqfq", "selenium");

selenium.click("link=Selenium IDE Plugins");

selenium.waitForPageToLoad("30000");

selenium.click("link=Documentation");

selenium.waitForPageToLoad("30000");

}

@After

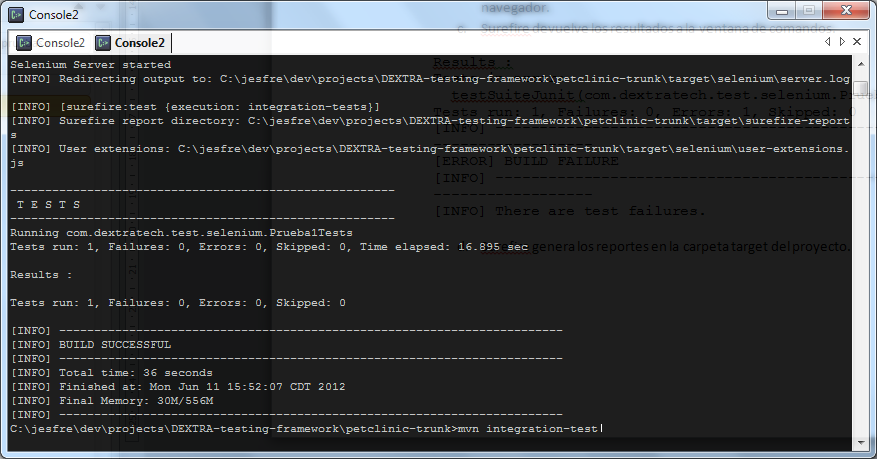
**public** **void** tearDown() **throws** Exception {

selenium.stop();

}

}

1. Para ejecutar la prueba automática con Selenium corrí la siguiente tarea: >mvn integration-test el cual se encarga de ejecutar las pruebas de Selenum.
   1. Levanta el servidor Selenium
   2. Abre una ventana del navegador (Firefox) y ejecuta las pruebas de Selenium en el navegador.
   3. Surefire devuelve los resultados a la ventana de comandos.



* 1. Surefire genera los **reportes** en la carpeta target del proyecto. **target/selenium/** y **target/surefire-reports/**.

## Generación de reportes HTML

Para generar los reportes en formato HTML de las pruebas se agregó el siguiente plugin:

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-report-plugin</artifactId>

<version>2.12</version>

<configuration>

<showSuccess>true</showSuccess>

</configuration>

</plugin>

Basta con ejecutar el comando **>mvn surefire-report:report** y éste se encargará de *parsear* los resultados individuales de las pruebas en XML que se encuentran en **target\surefire-reports** y generará el reporte HTML **target\site\ surefire-report.html**.

# Correr pruebas de Selenium usando archivos HTML

Dadas las configuraciones del primer documento. Se siguen los siguientes pasos:

1. Agregar la siguiente ejecución al plugin selenium-maven-plugin

<properties>

…

<selenium.browser>\*firefox</selenium.browser>

<selenium.suite>src/test/selenium/testSuite1.html</selenium.suite>

<selenium.startURL>http://google.com.mx</selenium.startURL>

</properties>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>selenium-maven-plugin</artifactId>

<version>2.3</version>

**<configuration>**

**<browser>${selenium.browser}</browser>**

**<suite>${selenium.suite}</suite>**

**<startURL>${selenium.startURL}</startURL>**

**</configuration>**

<executions>

<execution>

<id>start</id>

<phase>pre-integration-test</phase>

<goals>

<goal>start-server</goal>

</goals>

<configuration>

**<skip>true</skip>**

<background>true</background>

<logOutput>true</logOutput>

<multiWindow>true</multiWindow>

</configuration>

</execution>

<execution>

<id>stop</id>

<phase>post-integration-test</phase>

<goals>

<goal>stop-server</goal>

</goals>

<configuration>

<skip>true</skip>

</configuration>

</execution>

**<execution>**

**<id>test-html</id>**

**<phase>integration-test</phase>**

**<goals>**

**<goal>selenese</goal>**

**</goals>**

**<configuration>**

**<skip>false</skip>**

**</configuration>**

**</execution>**

</executions>

</plugin>

**Esta nueva ejecución espera encontrar el archivo de suite de pruebas definido en src/test/selenium/testSuite1.html.**

1. Correr **>mvn integration-test** o **>mvn selenium:selenese**. El primero causa el problema mencionado en la sección de Errores comunes.

# Agregar ejecución automática del servidor Jetty

Para complementar aún más las pruebas de integración, se puede automatizar la inicialización del servidor y el despliegue de la aplicación de forma automática. La forma más sencilla y rápida es hacerlo usando el servidor Jetty. Para hacerlo se agrega el plugin correspondiente.

<plugin>

<groupId>org.mortbay.jetty</groupId>

<artifactId>maven-jetty-plugin</artifactId>

<version>6.1.26</version>

<configuration>

<contextPath>/</contextPath>

<scanIntervalSeconds>3</scanIntervalSeconds>

<webDefaultXml>src/main/resources/jetty-webdefault.xml</webDefaultXml>

<scanTargetPatterns>

<scanTargetPattern>

<directory>src/main/webapp/WEB-INF</directory>

<excludes>

<exclude>\*\*/\*.jsp</exclude>

<exclude>src/test/\*.\*</exclude>

</excludes>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

</scanTargetPattern>

</scanTargetPatterns>

</configuration>

<executions>

<execution>

<id>start-jetty</id>

<phase>pre-integration-test</phase>

<goals>

<goal>run</goal>

</goals>

<configuration>

<scanIntervalSeconds>0</scanIntervalSeconds>

<daemon>true</daemon>

</configuration>

</execution>

<execution>

<id>stop-jetty</id>

<phase>post-integration-test</phase>

<goals>

<goal>stop</goal>

</goals>

</execution>

</executions>

</plugin>

La configuración webDefaultXml permite agregar ciertas configuraciones para prevenir que, durante la depuración de código en el desarrollo, Jetty bloquee los archivos de Web como \*.css, \*.js, \*.jsp, etc. El contenido del archivo src/main/resources/jetty-webdefault.xml se muestra en [jetty-webdefault.xml](#_jetty-webdefault.xml).

# Extender la clase SeleneseTestBase

Actualmente, el IDE de Selenium genera las clases extendiendo de la clase SeleneseTestCase, pero esta clase está deprecada. En lugar de extender de esa clase se sugiere extenderé de SeleneseTestBase, esto es para los casos de prueba solamente, ya que **para los suites de pruebas es necesario que extienda de SeleneseTestCase**.

Además de extender de la clase SeleneseTestBase, lo que hice fue extender (mejorar) la implementación construyendo una superclase para las pruebas que extiende de esa clase, y agregando las implementaciones de mejora. Desde ahí se pueden agregar más mejoras.

# Errores comunes

## El servidor Selenium está corriendo

Entrar a la dirección: <http://localhost:4444/selenium-server/driver/?cmd=shutDownSeleniumServer>

La respuesta debe ser **OKOK**, en caso de que el servidor realmente no esté levantado entonces marcará que no se puede conectar.

**Causa**

Este error ocurrió porque se estaba ejecutando la tarea definida en la ejecución start-server porque se está ejecutando directamente la fase **integration-test**. Para evitar que la tarea inicie el servidor Selenium, basta con agregar la configuración **<skip>true</skip>** a la ejecución.

# jetty-webdefault.xml

<?xml version=*"1.0"* encoding=*"ISO-8859-1"*?>

<!-- ===================================================================== -->

<!-- This file contains the default descriptor for web applications. -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!-- The intent of this descriptor is to include jetty specific or common -->

<!-- configuration for all webapps. If a context has a webdefault.xml -->

<!-- descriptor, it is applied before the contexts own web.xml file -->

<!-- -->

<!-- A context may be assigned a default descriptor by: -->

<!-- + Calling WebApplicationContext.setDefaultsDescriptor -->

<!-- + Passed an arg to addWebApplications -->

<!-- -->

<!-- This file is used both as the resource within the jetty.jar (which is -->

<!-- used as the default if no explicit defaults descriptor is set) and it -->

<!-- is copied to the etc directory of the Jetty distro and explicitly -->

<!-- by the jetty.xml file. -->

<!-- -->

<!-- ===================================================================== -->

<web-app

xmlns=*"http://java.sun.com/xml/ns/javaee"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*

metadata-complete=*"true"*

version=*"2.5"*>

<description>

Default web.xml file.

This file is applied to a Web application before it's own WEB\_INF/web.xml file

</description>

<!-- ==================================================================== -->

<!-- Context params to control Session Cookies -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!-- UNCOMMENT TO ACTIVATE

<context-param>

<param-name>org.mortbay.jetty.servlet.SessionDomain</param-name>

<param-value>127.0.0.1</param-value>

</context-param>

<context-param>

<param-name>org.mortbay.jetty.servlet.SessionPath</param-name>

<param-value>/</param-value>

</context-param>

<context-param>

<param-name>org.mortbay.jetty.servlet.MaxAge</param-name>

<param-value>-1</param-value>

</context-param>

-->

<context-param>

<param-name>org.mortbay.jetty.webapp.NoTLDJarPattern</param-name>

<param-value>start.jar|ant-.\*\.jar|dojo-.\*\.jar|jetty-.\*\.jar|jsp-api-.\*\.jar|junit-.\*\.jar|servlet-api-.\*\.jar|dnsns\.jar|rt\.jar|jsse\.jar|tools\.jar|sunpkcs11\.jar|sunjce\_provider\.jar|xerces.\*\.jar</param-value>

</context-param>

<!-- ==================================================================== -->

<!-- The default servlet. -->

<!-- This servlet, normally mapped to /, provides the handling for static -->

<!-- content, OPTIONS and TRACE methods for the context. -->

<!-- The following initParameters are supported: -->

<!-- -->

<!-- acceptRanges If true, range requests and responses are -->

<!-- supported -->

<!-- -->

<!-- dirAllowed If true, directory listings are returned if no -->

<!-- welcome file is found. Else 403 Forbidden. -->

<!-- -->

<!-- welcomeServlets If true, attempt to dispatch to welcome files -->

<!-- that are servlets, if no matching static -->

<!-- resources can be found. -->

<!-- -->

<!-- redirectWelcome If true, redirect welcome file requests -->

<!-- else use request dispatcher forwards -->

<!-- -->

<!-- gzip If set to true, then static content will be served-->

<!-- as gzip content encoded if a matching resource is -->

<!-- found ending with ".gz" -->

<!-- -->

<!-- resoureBase Can be set to replace the context resource base -->

<!-- -->

<!-- relativeResourceBase -->

<!-- Set with a pathname relative to the base of the -->

<!-- servlet context root. Useful for only serving -->

<!-- static content from only specific subdirectories. -->

<!-- -->

<!-- useFileMappedBuffer -->

<!-- If set to true (the default), a memory mapped -->

<!-- file buffer will be used to serve static content -->

<!-- when using an NIO connector. Setting this value -->

<!-- to false means that a direct buffer will be used -->

<!-- instead. If you are having trouble with Windows -->

<!-- file locking, set this to false. -->

<!-- -->

<!-- cacheControl If set, all static content will have this value -->

<!-- set as the cache-control header. -->

<!-- -->

<!-- maxCacheSize Maximum size of the static resource cache -->

<!-- -->

<!-- maxCachedFileSize Maximum size of any single file in the cache -->

<!-- -->

<!-- maxCachedFiles Maximum number of files in the cache -->

<!-- -->

<!-- cacheType "nio", "bio" or "both" to determine the type(s) -->

<!-- of resource cache. A bio cached buffer may be used-->

<!-- by nio but is not as efficient as a nio buffer. -->

<!-- An nio cached buffer may not be used by bio. -->

<!-- -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<servlet>

<servlet-name>default</servlet-name>

<servlet-class>org.mortbay.jetty.servlet.DefaultServlet</servlet-class>

<init-param>

<param-name>acceptRanges</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>dirAllowed</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>welcomeServlets</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>redirectWelcome</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>maxCacheSize</param-name>

<param-value>256000000</param-value>

</init-param>

<init-param>

<param-name>maxCachedFileSize</param-name>

<param-value>10000000</param-value>

</init-param>

<init-param>

<param-name>maxCachedFiles</param-name>

<param-value>1000</param-value>

</init-param>

<init-param>

<param-name>cacheType</param-name>

<param-value>both</param-value>

</init-param>

<init-param>

<param-name>gzip</param-name>

<param-value>true</param-value>

</init-param>

<init-param>

<param-name>useFileMappedBuffer</param-name>

<param-value>false</param-value>

</init-param>

<!--

<init-param>

<param-name>cacheControl</param-name>

<param-value>max-age=3600,public</param-value>

</init-param>

-->

<load-on-startup>0</load-on-startup>

</servlet>

<servlet-mapping> <servlet-name>default</servlet-name> <url-pattern>/</url-pattern> </servlet-mapping>

<!-- ==================================================================== -->

<!-- JSP Servlet -->

<!-- This is the jasper JSP servlet from the jakarta project -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!-- The JSP page compiler and execution servlet, which is the mechanism -->

<!-- used by Glassfish to support JSP pages. Traditionally, this servlet -->

<!-- is mapped to URL patterh "\*.jsp". This servlet supports the -->

<!-- following initialization parameters (default values are in square -->

<!-- brackets): -->

<!-- -->

<!-- checkInterval If development is false and reloading is true, -->

<!-- background compiles are enabled. checkInterval -->

<!-- is the time in seconds between checks to see -->

<!-- if a JSP page needs to be recompiled. [300] -->

<!-- -->

<!-- compiler Which compiler Ant should use to compile JSP -->

<!-- pages. See the Ant documenation for more -->

<!-- information. [javac] -->

<!-- -->

<!-- classdebuginfo Should the class file be compiled with -->

<!-- debugging information? [true] -->

<!-- -->

<!-- classpath What class path should I use while compiling -->

<!-- generated servlets? [Created dynamically -->

<!-- based on the current web application] -->

<!-- Set to ? to make the container explicitly set -->

<!-- this parameter. -->

<!-- -->

<!-- development Is Jasper used in development mode (will check -->

<!-- for JSP modification on every access)? [true] -->

<!-- -->

<!-- enablePooling Determines whether tag handler pooling is -->

<!-- enabled [true] -->

<!-- -->

<!-- fork Tell Ant to fork compiles of JSP pages so that -->

<!-- a separate JVM is used for JSP page compiles -->

<!-- from the one Tomcat is running in. [true] -->

<!-- -->

<!-- ieClassId The class-id value to be sent to Internet -->

<!-- Explorer when using <jsp:plugin> tags. -->

<!-- [clsid:8AD9C840-044E-11D1-B3E9-00805F499D93] -->

<!-- -->

<!-- javaEncoding Java file encoding to use for generating java -->

<!-- source files. [UTF-8] -->

<!-- -->

<!-- keepgenerated Should we keep the generated Java source code -->

<!-- for each page instead of deleting it? [true] -->

<!-- -->

<!-- logVerbosityLevel The level of detailed messages to be produced -->

<!-- by this servlet. Increasing levels cause the -->

<!-- generation of more messages. Valid values are -->

<!-- FATAL, ERROR, WARNING, INFORMATION, and DEBUG. -->

<!-- [WARNING] -->

<!-- -->

<!-- mappedfile Should we generate static content with one -->

<!-- print statement per input line, to ease -->

<!-- debugging? [false] -->

<!-- -->

<!-- -->

<!-- reloading Should Jasper check for modified JSPs? [true] -->

<!-- -->

<!-- suppressSmap Should the generation of SMAP info for JSR45 -->

<!-- debugging be suppressed? [false] -->

<!-- -->

<!-- dumpSmap Should the SMAP info for JSR45 debugging be -->

<!-- dumped to a file? [false] -->

<!-- False if suppressSmap is true -->

<!-- -->

<!-- scratchdir What scratch directory should we use when -->

<!-- compiling JSP pages? [default work directory -->

<!-- for the current web application] -->

<!-- -->

<!-- tagpoolMaxSize The maximum tag handler pool size [5] -->

<!-- -->

<!-- xpoweredBy Determines whether X-Powered-By response -->

<!-- header is added by generated servlet [false] -->

<!-- -->

<!-- If you wish to use Jikes to compile JSP pages: -->

<!-- Set the init parameter "compiler" to "jikes". Define -->

<!-- the property "-Dbuild.compiler.emacs=true" when starting Jetty -->

<!-- to cause Jikes to emit error messages in a format compatible with -->

<!-- Jasper. -->

<!-- If you get an error reporting that jikes can't use UTF-8 encoding, -->

<!-- try setting the init parameter "javaEncoding" to "ISO-8859-1". -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<servlet id=*"jsp"*>

<servlet-name>jsp</servlet-name>

<servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>

<init-param>

<param-name>logVerbosityLevel</param-name>

<param-value>DEBUG</param-value>

</init-param>

<init-param>

<param-name>fork</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>xpoweredBy</param-name>

<param-value>false</param-value>

</init-param>

<!--

<init-param>

<param-name>classpath</param-name>

<param-value>?</param-value>

</init-param>

-->

<load-on-startup>0</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>jsp</servlet-name>

<url-pattern>\*.jsp</url-pattern>

<url-pattern>\*.jspf</url-pattern>

<url-pattern>\*.jspx</url-pattern>

<url-pattern>\*.xsp</url-pattern>

<url-pattern>\*.JSP</url-pattern>

<url-pattern>\*.JSPF</url-pattern>

<url-pattern>\*.JSPX</url-pattern>

<url-pattern>\*.XSP</url-pattern>

</servlet-mapping>

<!-- ==================================================================== -->

<!-- Dynamic Servlet Invoker. -->

<!-- This servlet invokes anonymous servlets that have not been defined -->

<!-- in the web.xml or by other means. The first element of the pathInfo -->

<!-- of a request passed to the envoker is treated as a servlet name for -->

<!-- an existing servlet, or as a class name of a new servlet. -->

<!-- This servlet is normally mapped to /servlet/\* -->

<!-- This servlet support the following initParams: -->

<!-- -->

<!-- nonContextServlets If false, the invoker can only load -->

<!-- servlets from the contexts classloader. -->

<!-- This is false by default and setting this -->

<!-- to true may have security implications. -->

<!-- -->

<!-- verbose If true, log dynamic loads -->

<!-- -->

<!-- \* All other parameters are copied to the -->

<!-- each dynamic servlet as init parameters -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!-- Uncomment for dynamic invocation

<servlet>

<servlet-name>invoker</servlet-name>

<servlet-class>org.mortbay.jetty.servlet.Invoker</servlet-class>

<init-param>

<param-name>verbose</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>nonContextServlets</param-name>

<param-value>false</param-value>

</init-param>

<init-param>

<param-name>dynamicParam</param-name>

<param-value>anyValue</param-value>

</init-param>

<load-on-startup>0</load-on-startup>

</servlet>

<servlet-mapping> <servlet-name>invoker</servlet-name> <url-pattern>/servlet/\*</url-pattern> </servlet-mapping>

-->

<!-- ==================================================================== -->

<session-config>

<session-timeout>30</session-timeout>

</session-config>

<!-- ==================================================================== -->

<!-- Default MIME mappings -->

<!-- The default MIME mappings are provided by the mime.properties -->

<!-- resource in the org.mortbay.jetty.jar file. Additional or modified -->

<!-- mappings may be specified here -->

<!-- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -->

<!-- UNCOMMENT TO ACTIVATE

<mime-mapping>

<extension>mysuffix</extension>

<mime-type>mymime/type</mime-type>

</mime-mapping>

-->

<!-- ==================================================================== -->

<welcome-file-list>

<welcome-file>index.html</welcome-file>

<welcome-file>index.htm</welcome-file>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

<!-- ==================================================================== -->

<locale-encoding-mapping-list>

<locale-encoding-mapping><locale>ar</locale><encoding>ISO-8859-6</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>be</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>bg</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>ca</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>cs</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>da</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>de</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>el</locale><encoding>ISO-8859-7</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>en</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>es</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>et</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>fi</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>fr</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>hr</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>hu</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>is</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>it</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>iw</locale><encoding>ISO-8859-8</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>ja</locale><encoding>Shift\_JIS</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>ko</locale><encoding>EUC-KR</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>lt</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>lv</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>mk</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>nl</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>no</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>pl</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>pt</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>ro</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>ru</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>sh</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>sk</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>sl</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>sq</locale><encoding>ISO-8859-2</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>sr</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>sv</locale><encoding>ISO-8859-1</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>tr</locale><encoding>ISO-8859-9</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>uk</locale><encoding>ISO-8859-5</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>zh</locale><encoding>GB2312</encoding></locale-encoding-mapping>

<locale-encoding-mapping><locale>zh\_TW</locale><encoding>Big5</encoding></locale-encoding-mapping>

</locale-encoding-mapping-list>

<security-constraint>

<web-resource-collection>

<web-resource-name>Disable TRACE</web-resource-name>

<url-pattern>/</url-pattern>

<http-method>TRACE</http-method>

</web-resource-collection>

<auth-constraint/>

</security-constraint>

</web-app>