

Fast WebApps with Hightide

Fast Web Development with the Hightide and Maven

Hightide Convenience Dependencies

JavaEE APIs

When developing a new JavaEE webapp, you need to include a number of dependencies in your pom.xml which represent the APIs your webapp will reference (think JNDI, JTA etc). Hightide makes this fast and easy by providing a single dependency you can include in your pom.xml which will automatically transitively include all the usual apis. Use it thus:

```
<dependencies>
  . . .
  <dependency>

<groupId>com.webtide.hightide</groupId>

<artifactId>hightide-provided-apis</artifact
Id>
    <version>6.1H.4-beta</version>
    <type>pom</type>
    <scope>provided</scope>
  </dependency>
  . . .
</dependencies>
```

Hightide Dependencies

To make development with hightide even easier, we've provided a single dependency which you can include in your project's pom.xml which will automatically transitively include all the jars that are provided by hightide at runtime. Unlike the [Hightide APIs Dependency](#), it will put the implementation jars for those APIs on the classpath. Use this when your code explicitly access the implementation code rather than the JavaEE api.

```
<dependency>

<groupId>com.webtide.hightide</groupId>

<artifactId>hightide-server-dependencies</ar
tifactId>
    <version>6.1H.4-beta</version>
    <type>pom</type>
    <scope>provided</scope>
</dependency>
```

Hightide Plugin

mvn hightide:run

This goal is used in-situ on a Maven project without first requiring that the project is assembled into a war, saving time during the development cycle. The plugin forks a parallel lifecycle to ensure that the "compile" phase has been completed before invoking Hightide. This means that you do not need to explicitly execute a "mvn compile" first. It also means that a "mvn clean hightide:run" will ensure that a full fresh compile is done before invoking Hightide. This goal is similar to the mvn jetty:run goal, however, all services found in standalone hightide - such as JTA (Atomikos), JDBC (Derby) and JMS (ActiveMQ) - are automatically started and available to the webapp, significantly easing testing and promoting rapid development.

As with the mvn jetty:run target, the plugin can be configured to run continuously, scanning for changes in the project and automatically performing a hot redeploy when necessary. This allows the developer to concentrate on coding changes to the project using their IDE of choice and have those changes immediately and transparently reflected in the running web container, eliminating development time that is wasted on rebuilding, reassembling and redeploying.

Required Parameters

Name	Type	Description
classesDirectory	File	The directory containing generated classes
contextPath	String	The context path for the webapp. Defaults to the name of the webapp's artifact

scanIntervalSeconds	int	The interval in seconds to scan the webapp for changes and restart the context if necessary. Disabled by default
testClassesDirectory	File	The directory containing generated test classes
tmpDirectory	File	The temporary directory to use for the webapp. Defaults to target/work
webAppSourceDirectory	File	Root directory for all html/jsp etc files

Optional Parameters

Name	Type	Description
connectors	Connector[]	List of connectors to use. If none are configured then we use a single SelectChannelConnector at port 8080
contextHandlers	ContextHandler[]	List of other contexts to set up. Optional
jettyConfig	String	Location of a jetty xml configuration file whose contents will be applied before any plugin configuration. Optional
jettyEnvXml	String	The location of a jetty-env.xml file. Optional
overrideWebXml	File	A web.xml file to be applied AFTER the webapp's web.xml file. Useful for applying different build profiles, eg test,production etc. Optional
requestLog	RequestLog	A RequestLog implementation to use for the webapp at runtime. Optional
scanTargets	File[]	List of files or directories to additionally periodically scan for changes. Optional

systemProperties	SystemProperty[]	System properties to set before execution. Note that these properties will NOT override System properties that have been set on the command line or by the JVM. Optional
transaction	UserTransaction	Transaction resource. Optional
useTestClasspath	boolean	If true, the <testOutputDirectory> and the dependencies of <scope>test</scope> will be put first on the runtime classpath. Default value is false
userRealms	UserRealm[]	List of security realms to set up. Optional
webDefaultXml	File	A webdefault.xml file to use instead of the default for the webapp. Optional
webXml	String	The location of the web.xml file. If not set then it is assumed it is in <code>\${basedir}/src/main/webapp/WEB-INF</code>

mvn hightide:run-war

This goal is used to assemble your webapp into a war and automatically deploy it to Hightide.

Once invoked, the plugin can be configured to run continuously, scanning for changes in the project and to the war file and automatically performing a hot redeploy when necessary.

You may also specify the location of a jetty.xml file whose contents will be applied before any plugin configuration. This can be used, for example, to deploy a static webapp that is not part of your maven build.

Required Parameters

Name	Type	Description
contextPath	String	The context path for the webapp. Defaults to the name of the webapp's artifact
scanIntervalSeconds	int	The interval in seconds to scan the webapp for changes and restart the context if necessary. Disabled by default

tmpDirectory	File	The temporary directory to use for the webapp. Defaults to target/work
webApp	File	The location of the war file

Optional Parameters

Name	Type	Description
connectors	Connector[]	List of connectors to use. If none are configured then we use a single SelectChannelConnector at port 8080
jettyConfig	String	Location of a jetty xml configuration file whose contents will be applied before any plugin configuration. Optional
overrideWebXml	File	A web.xml file to be applied AFTER the webapp's web.xml file. Useful for applying different build profiles, eg test, production etc. Optional
requestLog	RequestLog	A RequestLog implementation to use for the webapp at runtime. Optional
systemProperties	SystemProperty[]	System properties to set before execution. Note that these properties will NOT override System properties that have been set on the command line or by the JVM. Optional
transaction	UserTransaction	Transaction resource. Optional
userRealms	UserRealm[]	List of security realms to set up. Optional
webDefaultXml	File	A webdefault.xml file to use instead of the default for the webapp. Optional

mvn hightide:run-exploded

This goal is used to assemble your webapp into an exploded war and automatically deploy it to Hightide.

Once invoked, the plugin can be configured to run continuously, scanning for changes in the pom.xml and to

WEB-INF/web.xml, WEB-INF/classes or WEB-INF/lib and hot redeploy when a change is detected.

You may also specify the location of a jetty.xml file whose contents will be applied before any plugin configuration. This can be used, for example, to deploy a static webapp that is not part of your maven build.

Required Parameters

Name	Type	Description
contextPath	String	The context path for the webapp. Defaults to the name of the webapp's artifact
scanIntervalSeconds	int	The interval in seconds to scan the webapp for changes and restart the context if necessary. Disabled by default
tmpDirectory	File	The temporary directory to use for the webapp. Defaults to target/work
webApp	File	The location of the war file

Optional Parameters

Name	Type	Description
connectors	Connector[]	List of connectors to use. If none are configured then we use a single SelectChannelConnector at port 8080
jettyConfig	String	Location of a jetty xml configuration file whose contents will be applied before any plugin configuration. Optional
overrideWebXml	File	A web.xml file to be applied AFTER the webapp's web.xml file. Useful for applying different build profiles, eg test, production etc. Optional
requestLog	RequestLog	A RequestLog implementation to use for the webapp at runtime. Optional

systemProperties	SystemProperty[]	System properties to set before execution. Note that these properties will NOT override System properties that have been set on the command line or by the JVM. Optional
transaction	UserTransaction	Transaction resource. Optional
userRealms	UserRealm[]	List of security realms to set up. Optional
webDefaultXml	File	A webdefault.xml file to use instead of the default for the webapp. Optional