Maven HTML Cleaner Plug-in - Quick Start DocBook Guide

1.0.0.beta-1

Table of Contents

This *Quick Start DocBook Guide* just explains how to modify a Maven Project (and Environment) to get DocBook content generated in a Maven Project Site. There is a more detailed explanation available in the User Guide section Making DocBook content available in a Maven project site [../docbook/article-user-guide.html#section-maven-html-cleaner-plugin-for-docbook-content] .

Installation

What other software must be installed first?

- Java JDK v1.5 (or higher)
- Maven v2.1 (or higher)

Adding the Maven Repository

The `Maven HTML Cleaner Plug-in` is *not yet* available through the central Maven Repository, but can be found in the project specific Maven Repository `http://docbook-utils.sourceforge.net/maven2`.

This Maven Repository can be added in a few different ways to a Maven build environment. Below are the most common options listed.

- Add the `<pluginRepository>` lines direct to the project `pom.xml`

  Single user - Single project

- Add the `<pluginRepository>` lines to a `<profile>` in the `~/.m2/setting.xml`

  Single user - Multiple projects

- Add the repository `http://docbook-utils.sourceforge.net/maven2` to a Maven Repository Proxy, like Nexus [http://nexus.sonatype.org/], Artifactory [http://www.jfrog.org/products.php], etc.

  Multiple users - Multiple projects

```
<pluginRepositories>
  <pluginRepository>
    <id>docbook-utils</id>
    <name>DocBook Utils</name>
    <url>http://docbook-utils.sourceforge.net/maven2</url>
    <layout>default</layout>
    <releases>
      <enabled>true</enabled>
      <updatePolicy>daily</updatePolicy>
      <checksumPolicy>warn</checksumPolicy>
    </releases>
  </pluginRepository>
</pluginRepositories>
```

Update Maven Settings

The following lines are needed because the new Plug-ins, used here, are not in the *Default Plug-ins Domain* `<groupId>org.apache.maven.plugins</groupId>`. When these lines are added, calling the Plug-ins from the command line, is much easier. It also feels more natural, as they behave then similar as Plug-ins that come from the *Default Plug-ins Domain*.

```
<!--  When Plug-ins should be taken into account during the plug-in search, add
<!--  to this list, as they are not in the default groupId 'org.apache.maven.plu
<pluginGroups>
  <!--  http://docbook-utils.sourceforge.net/maven-htmlCleaner-plugin_1.0  -->
  <pluginGroup>net.sourceforge.docbook-utils.maven-plugins</pluginGroup>
  <!--  http://code.google.com/p/docbkx-tools/  -->
  <pluginGroup>com.agilejava.docbkx</pluginGroup>
</pluginGroups>
```

After adding the above lines the new Plug-ins can be used directly from the command line:

- `$ mvn docbkx:generate-html`
- `$ mvn html-cleaner:transform`

If the `settings.xml` is not updated, the Plug-ins can still be called, from the command line, but only by specifying their full name:

- `$ mvn com.agilejava.docbkx:docbkx:generate-html`
- `$ mvn net.sourceforge.docbook-utils.maven-plugins:html-cleaner:transform`

Adding the Plug-ins to the project pom

Add the following plug-ins to the project `pom.xml`. Maybe add the `<pluginManagement>` part in a parent project.

```
<build>
  <pluginManagement>
    <plugins>

      <!--  Generate (X)HTML / PDF documents from DocBook content  -->
      <plugin>
```

```
  <groupId>com.agilejava.docbkx</groupId>
  <artifactId>docbkx-maven-plugin</artifactId>
  <version>2.0.10</version>
  <executions>
    <execution>
      <phase>pre-site</phase>
      <goals>
        <goal>generate-html</goal>
        <goal>generate-xhtml</goal>
        <goal>generate-pdf</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <xincludeSupported>true</xincludeSupported>
  </configuration>
  <dependencies>
    <dependency>
      <groupId>org.docbook</groupId>
      <artifactId>docbook-xml</artifactId>
      <version>4.4</version>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
</plugin>

<!--  Clean up the, by docbkx-maven-plugin, generated HTML  -->
<plugin>
  <groupId>net.sourceforge.docbook-utils.maven-plugins</groupId>
  <artifactId>maven-html-cleaner-plugin</artifactId>
  <version>1.0.0.beta-1</version>
  <executions>
    <execution>
      <phase>pre-site</phase>
      <goals>
        <goal>transform</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <sourceDir>${project.build.directory}/docbkx/html</sourceDir>
    <destinationDir>${project.build.directory}/generated-site/xhtml/docboo
    <replaceExtensionMap>
      <html>xhtml</html>
    </replaceExtensionMap>
  </configuration>
</plugin>

<!--  Generate Project Site  -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-site-plugin</artifactId>
  <version>2.1</version>
  <dependencies>
```

```
            <dependency>
              <groupId>org.apache.maven.doxia</groupId>
              <artifactId>doxia-module-xhtml</artifactId>
              <version>1.1.2</version>
            </dependency>
          </dependencies>
        </plugin>
      </plugins>
    </pluginManagement>

    <!--  Plug-in used in this project  -->
    <plugins>

      <!--  Generate (X)HTML / PDF documents from DocBook content  -->
      <plugin>
        <groupId>com.agilejava.docbkx</groupId>
        <artifactId>docbkx-maven-plugin</artifactId>
      </plugin>

      <!--  Clean up the, by docbkx-maven-plugin, generated HTML  -->
      <plugin>
        <groupId>net.sourceforge.docbook-utils.maven-plugins</groupId>
        <artifactId>maven-html-cleaner-plugin</artifactId>
      </plugin>

      <!--  Generate Project Site  -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-site-plugin</artifactId>
      </plugin>
    </plugins>
  </build>
```

Generate a Maven Project Site

- **Generate the Maven project site**

  ```
  $  mvn  clean  site
  ```