

Daisy Maven Plugin

WARNING

This documentation refers to an alpha version of the Daisy Java Adapter. This means that any interfaces or contracts can change without a preceding deprecation period.

WARNING

The Daisy Maven Plugin requires a Daisy 1.5.1 repository. It won't work with any higher or lower version.

The Daisy Maven Plugin creates a raw Maven 2 documentation site which can be used by the Maven site plugin to create HTML documents. Additionally it has to provide support for link resolution across several of such documentations.

Features

- export a Daisy site, based on a navigation document and a “home collection” as starting point
- link resolution, also across different sites
- create XHTML that is understood by the Maven 2 Site plugin
- provide your own export strategies for custom document types
- configure the number of parallel connections that are used to retrieve documents
- add the site.xml as configuration. This also makes is easy to set the Maven Site skin for a multi-module documentation.
- an already existing site.xml can be enhanced by sections coming from a Daisy CMS

Usage

Setup the Maven Daisy plugin

Note: If you want to see the plugin in action, checkout the Cocoon project which makes use of it to create its docs.

First, add your new plugin to your POM:

```
<project>
  <build>
    <plugins>
      <plugin>
        <groupId>org.daisycms</groupId>
        <artifactId>daisy-maven-plugin</artifactId>
        <version>1.0.0-alpha-2</version>
        <configuration>...</configuration>
      </plugin>
    </plugins>
  </build>
  <repositories>
    <repository>
      <id>cocoon.dev</id>
      <url>http://cocoondev.org/repository2</url>
      <releases>
        <enabled>true</enabled>
      </releases>
    </repository>
  </repositories>
</project>
```

```

    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>
<pluginRepositories>
  <pluginRepository>
    <id>cocoon-dev.org</id>
    <url>http://cocoondev.org/repository2</url>
    <releases>
      <enabled>true</enabled>
    </releases>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>
</project>

```

Then you have to provide a configuration containing information about where to find documents that should be exported by the plugin, how it can access the Daisy CMS server and what it should do in the case that the current navigation document doesn't contain a referenced document:

```

<configuration>
  <repoUrl>http://localhost:9263</repoUrl>
  <daisyServerId>[the-server-id-as-reference-to-settings.xml-entry</daisyServerId>
  <author>Your name</author>
  <navDocId>4711</navDocId>
  <collection>[collection-of-valid-documents]</collection>
  <collections>
    <collection>
      <name>[other-collection]</name>
      <path>[relative-path-to-other-collection]</path>
    </collection>
  </collections>
</configuration>

```

From this description you might suspect that the central configuration option is a navigation document. This navigation document is used to create a list of all documents that will be retrieved from the repository and it is also used to create the menu of the Maven site.

While the documents are retrieved from the repository, it is checked if all linked documents are members of a particular collection. If not, they won't be exported. That's the reason, why the configuration also contains a collection element. In the case that they are members of this collection but not listed in the navigation document, they will be exported and will finally be available as one of the result documents.

The question now is what happens with linked Daisy documents that are not in the configured collection. For this purpose you can provide additional configurations that contain the relative paths to the locations of those documents. The idea behind this solution is that a linked document will be exported from a different Maven project but it will be published to the same location.

And finally, you have to configure where the Daisy repository can be found and you have to provide user credentials. Unfortunately you have to use a user with administration rights. The repository URL is set as part of the plugin configuration as shown above and the credentials have to be added to the Maven settings. Usually they are added to `~/.m2/settings.xml`:

```

<settings>
  <servers>
    <server>
      <id>[repo-id-as-used-in-plugin-configuraiton]</id>
      <username>[user-with-admin-rights]</username>
      <password>[secret-password]</password>
    </server>
  </servers>
</settings>

```

```
</server>
</servers>
</settings>
```

Configure the Maven Site site.xml (including the Maven skin)

You can add the site.xml that should be created as configuration option. Note that you have to escape the XML:

```
<configuration>
  <siteXmlContent>
    &lt;project&gt;
      &lt;skin&gt;
        &lt;groupId&gt;org.apache.cocoon&lt;/groupId&gt;
        &lt;artifactId&gt;cocoon-maven-site-skin&lt;/artifactId&gt;
        &lt;version&gt;1.0.0-SNAPSHOT&lt;/version&gt;
      &lt;/skin&gt;
      &lt;body&gt;
        &lt;!-- @daisy-start@ --&gt;&lt;!-- @daisy-end@ --&gt;
        &lt;menu ref="reports" /&gt;
      &lt;/body&gt;
    &lt;/project&gt;
  </siteXmlContent>
  <exportStrategyClass>org.daisycms.clientapp.maven.export.CocoonExportStrategy</
exportStrategyClass>
</configuration>
```

Run the plugin

If you have configured your plugin to run automatically with the Maven Site plugin by

```
<plugin>
  <groupId>org.daisycms</groupId>
  <artifactId>daisy-maven-plugin</artifactId>
  <version>1.0.0-SNAPSHOT</version>
  <executions>
    <execution>
      <phase>pre-site</phase>
      <goals>
        <goal>export</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

then you only have to enter

```
mvn site
```

Fields

Name	Value
Category	Miscellaneous