Maven2 plugin A Maven 2 / Maven 3 plugin that wraps the Cargo Java API

Functional tests

The usage of Cargo for executing functional tests on a container does not mandate this m2 plugin. You could also directly use the Cargo Java API from your Java unit test classes (JUnit, TestNG, etc), as described on the <u>Functional testing</u> page. The choice is yours, thought the Maven2 plugin is generally more straightforward to use and integrates better with the whole build process (with profiles, easier to use deployer, proxy server support, etc.)

Table of Contents

The documentatation for this Maven2 plugin includes:

- Installation: explains how to install the plugin
- · Getting started: explains how to use the plugin on several use cases
- <u>Reference Guide</u>: provide reference documentation for all configuration options
- Tips: tips for using the plugin

Getting Started

Very quick start

CARGO can be directly run on any existing Maven2 Java EE project (WAR, EAR or other) by running:

mvn clean verify org.codehaus.cargo:cargo-maven2-plugin:run

This will create a default <u>Jetty 7.x installed local container</u> and start it using the Cargo Maven2 plugin with your Maven2 project's deployable (a WAR, for example) deployed to it; so you can run manual tests (as a first introduction).

What is magic is that if you now want to run the same tests with <u>Tomcat 7.x</u> you simply need to run (in one line):



That command will automatically download Tomcat 7.0.16 from the specified URL (taking into account any proxy server setting you would have in Maven2/Maven3), instantiate the container, create a local configuration with your application and run it. It will also save the downloaded container in the default directory (see the <u>Maven2 Plugin</u> <u>Reference Guide</u> for details), so it won't get downloaded when you run the same command twice.

Now, if you want to run this time on <u>Glassfish 3.x</u> with with the HTTP port set to 9000, run:

mvn clean verify org.codehaus.cargo:cargo-maven2-plugin:run -Dcargo.maven.containerId=glassfish3x -Dcargo.maven.containerUrl=http://download.j ava.net/glassfish/3.1.1/release/glassfish-3. 1.1.zip -Dcargo.servlet.port=9000

CARGO's main advantage is that the commands and configuration remains the same for any version of any container supported by CARGO -be it Tomcat, Jetty, JBoss, JOnAS, GlassFish, WebLogic, etc.

Like it? Well, keep on reading, then!

More examples

As usual the best way to learn to use a tool is through examples.

We have several <u>Maven2 Archetypes</u> that contain sample Maven2/Maven3 projects with different use cases for the CARGO plugin, we would really recommend that you check them out. For more details, read here: <u>Maven2</u> <u>Archetypes</u>.

In addition here are the typical uses cases covered by the plugin:

- Deploying to a running container
- Generating a container configuration deployment structure
- Merging WAR files
- <u>Starting and stopping a container</u>

The Cargo Maven plugin in detail

Here are the different goals available to call on this plugin:

Goals	Description
cargo:start	 Start a container. That goal will: If the plugin configuration requires so, installs the container. If the plugin configuration defines a container with a standalone local configuration, it will create the configuration. If the plugin configuration contains one or more deployables, it will deploy these to the container automatically. If the plugin configuration contains no deploya bles but the project's packaging is Java EE (WAR, EAR, etc.), it will deploy the project's deployable to to the container automatically. And, of course, start the container. Note: A container that's started with cargo:start will automatically shut down as soon as the parent Maven instance quits (i.e., you see a BUILD_SUCCESSFUL or BUILD_FAILED message). If you want to start a container and perform manual testing, see our next goal cargo:run.

cargo:run	Start a container and wait for the user to press $\tt CTRL + C$ to stop. That goal will:
	 If the plugin configuration requires so, installs the container. If the plugin configuration defines a container with a standalone local configuration, it will create the configuration. If the plugin configuration contains one or more deployables, it will deploy these to the container automatically. If the plugin configuration contains no deploya bles but the project's packaging is Java EE (WAR, EAR, etc.), it will deploy the project's deployable to to the container automatically. And, of course, start the container and wait for the user to press CTRL + C to stop.
cargo:stop	Stop a container.
cargo:restart	Stop and start again a container. If the container was not running before calling cargo:restart, it will simply be started.
cargo:configure	Create the configuration for a <u>local container</u> , without starting it. Note that the cargo:start and cargo:ru n goals will also install the container automatically (<u>but</u> <u>will not call cargo:install</u>).
cargo:package	Package the local container.
cargo:daemon-start	Start a container via the daemon. Read more on: <u>Carg</u> <u>o Daemon</u>
cargo:daemon-stop	Stop a container via the daemon. Read more on: <u>Cargo</u> <u>Daemon</u>
<pre>cargo:deployer-deploy (aliased to cargo:deplo y)</pre>	Deploy a deployable to a running container.
cargo:deployer-undeploy (aliased to cargo:und eploy)	Undeploy a deployable from a running container.
cargo:deployer-start	Start a deployable already installed in a running container.
cargo:deployer-stop	Stop a deployed deployable without undeploying it.

cargo:deployer-redeploy (aliased to cargo:red eploy)	Undeploy and deploy again a deployable. If the deployable was not deployed before calling cargo:de ployer-redeploy (or its alias cargo:redeploy) it will simply be deployed.
cargo:uberwar	Merge several WAR files into one.
cargo:install	Installs a container distribution on the file system. Note that the cargo:start goal will also install the container automatically (<u>but will not call cargo:insta</u> <u>11</u>).
cargo:help	Get help (list of available goals, available options, etc.).

The configuration elements are described in the <u>Reference Guide</u> section.