

Cargo Daemon

Introduction

The Cargo Daemon is a Web-based application that uses the Cargo API to configure, start and stop containers on a remote machine.

The daemon is meant to be listening 24/7, to allow users to deploy new containers and web applications at their command.

It can be accessed using a browser-based UI, via Java API or Maven2 plugin.

Why use the Cargo Daemon?

Most web containers (e.g. Tomcat, Jetty) provide built-in remote deployment facilities already, also many of them already have daemon integrations; so why use the Cargo Daemon?

- **During intense redeployment** (i.e., testing and even sometimes QA or production hot deployments): All of the remote deployment facilities that keep the JVM alive will eventually suffer from the dreaded `java.lang.OutOfMemoryError: PermGen space` exception if something in the web application is leaking memory. Most web containers try their best to track down these 'dead' objects and forcefully remove them, but it does not always succeed to reclaim the memory. With a leaking web application, the available memory starts to shrink after each redeploy, and eventually the memory is exhausted. The only solution to this is to kill the JVM, and restart it. And that is exactly what the Cargo Daemon tries to manage. It will try to shutdown the web application cleanly, but if that fails it will forcefully kill the JVM. It is the only way to guarantee that a new version of your web application always starts when you want it to.
- **In heterogeneous environments**: With Cargo, the way you configure the container is independent from the underlying server -you can set the different [configuration properties](#), define [datasources](#), add [deployables](#), etc. transparently. You can therefore use the Cargo Daemon as a container-independent daemon, with support for the generation of the proper configuration on all supported containers.
- **During upgrades and/or application server product evaluations**: As Cargo is not dependent on the application server nor on its version, you can easily reuse an existing Cargo Daemon setup to use it for another version of a container, or another container altogether; without having to worry about understanding how to configure it.

Table of Contents

The documentatation for the Cargo Daemon includes:

- [Installation](#): explains how to install and run the daemon
- [Getting started](#): very quick guide on using the daemon

Installation

i Java versions for running the Daemon

Cargo Daemon requires Java 6 or greater in order to run in standalone mode.

If you deploy the Cargo Daemon WAR file on an existing container, the minimum requirement is Java version 5.

To install and run the Cargo Daemon:

1. Download the Cargo Daemon from the [Downloads](#) page
2. Execute by typing:

```
java -jar  
target/cargo-daemon-webapp-<version>.war
```

where <version> is the version number of the daemon that you have downloaded.

By default, the Cargo Daemon will run on HTTP port 18000. To change it, use the `-p` option:

```
java -jar  
target/cargo-daemon-webapp-<version>.war -p  
18001
```

Additionally, Cargo Daemon will save log files in the cargo home directory unless the `-nologging` option is used.

Note that the Cargo Daemon is a WAR file; you can actually also deploy it as a WAR on any existing container. This can be useful if you want to, for example, reuse a certain security configuration.

The daemon also accepts other parameters, in the form of system properties:

Property name	Description	Mandatory?	Default value
---------------	-------------	------------	---------------

<code>daemon.home</code>	<p>Directory in which the standalone daemon server stores its files. These include the temporary files (such as its own WAR and server temporary files) as well as the server log files (<code>AWS-xxxxxxxxxxxxxxxxx.log</code>, where <code>xxxxxxxxxxxx</code> is the timestamp at which the daemon was started).</p> <p>This property is not used and completely ignored if the daemon WAR file is deployed on an existing container.</p>		<code>\${user.home}/.cargo</code>
<code>cargo.home</code>	<p>Directory in which the daemon (be it standalone or deployed on an existing container) stores the list of containers, downloaded container archives, container logs, etc.</p>		<code>\${user.home}/.cargo</code> <p>Note that the standalone daemon by default sets this to <code>\${daemon.home}</code></p>

Getting started using the browser UI

To use the Cargo Daemon via the browser UI, simply open `http://<machine>:<port>/` -where `<machine>` is the machine host name or IP address and `<port>` is the port number used (default is 18000):

- To start a container, fill in the form and press **Start**:

Please specify below the various options for your container and select **Start**.

Handle id:	<input type="text" value="tomcat6-test"/>												
Container identifier:	<input type="text" value="tomcat6x"/>												
Container autostart:	<input type="checkbox"/>												
Container home directory:	<input type="text" value="/Applications/Java/apache-tomcat-6.0.37/binaries"/>												
... or installer zip URL:	<input type="text"/>												
... or installer zip file:	<input type="button" value="Choose File"/> no file selected <input type="text" value="Save as:"/>												
Configuration type:	<input type="text" value="EXISTING"/>												
Configuration home directory (optional):	<input type="text" value="/Applications/Java/apache-tomcat-6.0.37/configuration-with-oracle-db"/>												
Container log output file name (optional):	<input type="text"/>												
Container append output:	<input type="checkbox"/>												
Container system properties:	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td colspan="2"><input type="button" value="Add property"/></td></tr></tbody></table>	Name	Value	<input type="button" value="Add property"/>									
Name	Value												
<input type="button" value="Add property"/>													
Configuration properties:	<table><thead><tr><th>Name</th><th>Value</th></tr></thead><tbody><tr><td><input type="text" value="cargo.servlet.port"/></td><td><input type="text" value="8081"/></td></tr><tr><td colspan="2"><input type="button" value="Add property"/></td></tr></tbody></table>	Name	Value	<input type="text" value="cargo.servlet.port"/>	<input type="text" value="8081"/>	<input type="button" value="Add property"/>							
Name	Value												
<input type="text" value="cargo.servlet.port"/>	<input type="text" value="8081"/>												
<input type="button" value="Add property"/>													
Configuration files:	<table><thead><tr><th>File</th><th>Filename</th><th>Directory</th><th>Overwrite</th><th>Parse</th><th>Encoding</th></tr></thead><tbody><tr><td colspan="6"><input type="button" value="Add file"/></td></tr></tbody></table>	File	Filename	Directory	Overwrite	Parse	Encoding	<input type="button" value="Add file"/>					
File	Filename	Directory	Overwrite	Parse	Encoding								
<input type="button" value="Add file"/>													
Deployable files:	<table><thead><tr><th>File</th><th>Filename</th><th>Type</th></tr></thead><tbody><tr><td><input type="button" value="Choose File"/>  datasource-war-...1-SNAPSHOT.war</td><td><input type="text" value="datasource-war.war"/></td><td><input type="text" value="war"/></td></tr><tr><td colspan="3"><input type="button" value="Add file"/></td></tr></tbody></table>	File	Filename	Type	<input type="button" value="Choose File"/>  datasource-war-...1-SNAPSHOT.war	<input type="text" value="datasource-war.war"/>	<input type="text" value="war"/>	<input type="button" value="Add file"/>					
File	Filename	Type											
<input type="button" value="Choose File"/>  datasource-war-...1-SNAPSHOT.war	<input type="text" value="datasource-war.war"/>	<input type="text" value="war"/>											
<input type="button" value="Add file"/>													

- To **stop**, **restart**, **delete** or **view logs** of a container, use the actions on the containers list:

Running containers:

Handle id	Status	Actions			
tomcat6-test	started	<input type="button" value="Stop"/>	<input type="button" value="Start"/>	<input type="button" value="View logs"/>	<input type="button" value="Delete"/>

- The Cargo Daemon keeps a persistent record on disk of all the containers that have been submitted. Containers that have been submitted will stay in the list, even when they are stopped. This allows you to manually restart them, or view the logs even after the container is stopped.
- If you want the container to be removed from the list, simply press the **delete** button.
- Containers can also be submitted with the `autostart` property, this will automatically restart the container if the daemon notices it is stopped.

Getting started with the Java API / Maven2/Maven3 plugin

As stated before, the Cargo Daemon is also available programmatically:

- The details of the Java API can be seen on the [Javadoc for o.c.c.tools.daemon.DaemonClient](#)
- The details of the configuration for the Maven2/Maven3 plugin is documented in the [Daemon configuration section of the Maven2/Maven3 plugin reference guide](#) and the example can be seen via the [Daemon archetype](#).

 Java versions for connecting to the Daemon via Java API or Maven2/Maven3 plugin

In order to connect to the Daemon via Java API or Maven2/Maven3 plugin, the minimum requirement is Java version 5.

To get the required libraries for using the Daemon via Java API, please check the [Downloads](#) page.