Space Details

Key:	CARGO
Name:	Cargo
Description:	Uniform J2EE Container Control System
Creator (Creation Date):	bwalding (Aug 14, 2004)
Last Modifier (Mod. Date):	bwalding (Aug 14, 2004)

Available Pages

    - Jetty 4.x
    - Oc4J 9.x
    - Orion 1.x
    - Orion 2.x
    - Resin 2.x
    - Resin 3.x
    - Tomcat 3.x
    - Tomcat 4.x
    - Tomcat 5.x
    - WebLogic 8.x
- Release procedure

This page last changed on Oct 30, 2004 by vmassol.

Mission

Cargo provides a Java API to start/stop and configure Java containers

Possible use cases for Cargo:

- To start containers for integration and functional tests
- To start containers for applications that require a container to be started (Plugins for IDEs, etc)

Status

Version status (click in the status column to get release notes):

Version	Status	Comments
0.1	✓	Released on 11/09/04
0.2	✓	Released on 03/10/04
0.3	✓	Released on 30/10/04
0.4	✗	Around November 2004

As glitches may happen even after a container is released for the first time, e.g. if a new feature is added to the framework, but not supported by all containers, we encourage you to report your success/failures in the Tested on section.

Feature list

- Ant support — Cargo provides Ant tasks to perform all the operations available from the Java API
- Classpath configuration — How to configure Cargo's classpath
- Configuration — Specifies how the container is configured
- Configuration properties — Properties to configure a container (request port, shutdown port, logging level, threads, etc)
- Container Factory — Instantiate a container by name
- Container instance creation — Create a container instance
- Debugging — Explain how to perform debugging when something doesn't work in Cargo
- Deployable — Deployables are archives (WAR, EAR, etc) that can be deployed in the

container
- Deployment descriptor API — API to manipulate J2EE descriptors (currently `web.xml` and `application.xml`)
- Embedded mode
- Existing configuration — Not yet implemented
- Installer — Installs a container
- Maven support — Not yet implemented
- Passing system properties — How to pass system properties that will be available to the container while executing
- Standalone configuration — Configures your container in a specific directory
- Standalone mode
- Start — Start a container that is not already running
- Static deployment of EAR — Deploy an EAR that will be started when the container starts
- Static deployment of expanded WAR
- Static deployment of WAR — Deploy a WAR that will be started when the container starts
- Stop — Stop a running container

Container support

Container	Java API/version	Ant API/version	Maven API/version
JBoss 3.x	✗ ???	✗ ???	✗ N/A
Jetty 4.x	✓ 0.1	✗ ???	✗ N/A
OC4J 9.x	✓ 0.3	✓ 0.3	✗ N/A
Orion 1.x	✓ 0.1	✓ 0.1	✗ N/A
Orion 2.x	✓ 0.1	✓ 0.1	✗ N/A
Resin 2.x	✓ 0.1	✓ 0.1	✗ N/A
Resin 3.x	✓ 0.1	✓ 0.1	✗ N/A
Tomcat 3.x	✓ 0.1	✓ 0.1	✗ N/A
Tomcat 4.x	✓ 0.1	✓ 0.1	✗ N/A
Tomcat 5.x	✓ 0.1	✓ 0.1	✗ N/A
WebLogic 8.x	✓ 0.3	✓ 0.3	✗ N/A

Quick Start

The following piece of code demonstrates how to configure Resin 3.0.8 to start in `target/resin3x` and deploy a WAR located in `src/testinput/simple.war`. The default port is 8080. Please note that the `container.start()` and `container.stop()` methods wait until the container is fully started and fully stopped before continuing. Thus, for any action you are executing after, you are assured the container is completely operational.

```
Container container = new Resin3xContainer();
container.setHomeDir("c:/apps/resin-3.0.8");

Deployable deployable =
    container.getDeployableFactory().createWAR("path/to/simple.war");
container.addDeployable(deployable);

container.start();

// Here you are assured the container is started.

container.stop();

// Here you are assured the container is stopped.
```

Navigation

This page last changed on Oct 30, 2004 by vmassol.

Cargo 0.3 doc

- Home
- News
- Features
- Javadoc
- License

Archives

- Doc Archives

Download

- Cargo 0.3
- [Cargo 0.3 doc

|Documentation Archives^cargo-0.3.pdf]

Containers

- JBoss 3.x
- Jetty 4.x
- Orion 1.x
- Orion 2.x
- Resin 2.x
- Resin 3.x
- Tomcat 3.x
- Tomcat 4.x
- Tomcat 5.x
- Weblogic 8.x
- Oc4j 9.x

Support

- Issues
- Roadmap
- Change log

Community

- Mailing Lists
- Who we are

Developers

- Credits
- SVN
- Wiki
- Maven site
- Release procedure
- DC status

Maven support

This page last changed on Oct 24, 2004 by vmassol.

Definition

Not yet implemented

Tested on

This page last changed on Oct 08, 2004 by vmassol.

In this section you can find the test status of the different containers for the different Cargo releases.
This page will then contain results of testing the framework in real world configurations.

Add your own experiences to the section matching your framework version, using the following format:

- Tomcat
  - 4.1.27 (J2EE 1.2 and J2EE 1.3) – Vincent Massol
  - 4.1.28 (J2EE 1.3) **failed** – jerome@coffeebreaks.org

Cargo 0.2

- Resin
  - 3.0.8 (J2EE 1.3) – Vincent Massol
  - 3.0.9 (J2EE 1.3) **failed** – Vincent Massol
    - It fails because Resin 3.0.9 no longer supports the <cache> directive (it's now only support for the professional versions). See CARGO-44. Support added in Cargo 0.3.
- Tomcat
  - 3.3.2 (J2EE 1.3) – Vincent Massol
  - 4.1.30 (J2EE 1.3) – Vincent Massol
  - 5.0.25 (J2EE 1.3) – Vincent Massol
  - 5.0.28 (J2EE 1.3) – Vincent Massol
  - 5.5.2 (J2EE 1.3) – Vincent Massol
  - 5.5.3-alpha (J2EE 1.3) – Vincent Massol
- Orion
- Jetty

Cargo 0.1

- Resin
  - 3.0.8 (J2EE 1.3) – Vincent Massol
- Tomcat
  - 3.3.2 (J2EE 1.3) – Vincent Massol
  - 4.1.30 (J2EE 1.3) – Vincent Massol
  - 5.0.25 (J2EE 1.3) – Vincent Massol

    - 5.0.28 (J2EE 1.3) – Vincent Massol
- Orion
    - 1.6.0b (J2EE 1.3) – Vincent Massol
    - 2.0.3 (J2EE 1.3) – Vincent Massol
- Jetty
    - 4.1.20 (J2EE 1.3) – Vincent Massol
    - 4.2.17 (J2EE 1.3) – Vincent Massol

SVN

This page last changed on Aug 20, 2004 by vmassol.

For general information see the SVN page on Codehaus.

Web Access

http://svn.cargo.codehaus.org

Anonymous SVN Access

```
svn co svn://svn.cargo.codehaus.org/cargo/scm/cargo/trunk
```

Developer SVN Access via SSH

```
svn co svn+ssh://svn.cargo.codehaus.org/home/projects/cargo/scm/cargo/trunk
```

SVN Access behind a firewall

Currently Codehaus does not support WebDAV access.

Credits

This page last changed on Oct 22, 2004 by vmassol.

The following persons deserve credit for Cargo:

- **Apache and The Jakarta cactus project**: Cargo started as a refactoring of the Cactus Ant integration subproject
- Vincent Massol: Lead developer of Cargo (and of Cactus)
- **Christopher Lenz**: Has developed most of the Cactus Ant integration code that spawned the Cargo project
- Desire ATANGA: Implementation of Tomcat support, Implementation of WebLogic support
- Jerome Lacoste: General ideas and discussions about Cargo
- Tim Shadel: Implementation of OC4J support

A special mention to the following Cargo users who have helped make Cargo better and helped promote it:

- Mike-Cannon Brookes: Asked for expanded war support
- Matt Raible: Asked for improvements to the Tomcat support so that Cargo can support nicely AppFuse

If we have forgotten anyone, please accept our apologies and feel free to mention on the list so that we can correct the error.

Documentation Archives

This page last changed on Oct 03, 2004 by vmassol.

This web site contains the documentation for the next version of Cargo.

Available documentation PDFs:

- Cargo 0.1 documentation
- Cargo 0.2 documentation

Resin 3.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✗	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PORT	✓	
	GeneralPropertySet.HOSTNAME	✓	
	GeneralPropertySet.LOGGING	✓	

Instantiating in Java

```
Container container = new Resin3xContainer();
[...]
```

Instantiating in Ant

```
<cargo-resin3x [...]
</cargo-resin3x>
```

Resin 2.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✗	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PORT	✓	
	GeneralPropertySet.HOSTNAME	✓	
	GeneralPropertySet.LOGGING	✓	

Instantiating in Java

```
Container container = new Resin2xContainer();
[...]
```

Instantiating in Ant

```
<cargo-resin2x [...]
</cargo-resin2x>
```

Orion 2.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✓	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PO T	✓	
	GeneralPropertySet.H TNAME	✗	
	GeneralPropertySet.L GING	✗	

Custom configuration properties:

Property name	Java constant	Valid values	Description	Example

	to use			
cargo.orion.rmi.port	OrionPropertySet.RMI_PORT	Integer	Port for the Orion RMI server	"25791"

Instantiating in Java

```
Container container = new Orion2xContainer();
[...]
```

Instantiating in Ant

```
<cargo-orion2x [...]
</cargo-orion2x>
```

Orion 1.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✓	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PORT	✓	
	GeneralPropertySet.HOSTNAME	✗	
	GeneralPropertySet.LOGGING	✗	

Custom configuration properties:

Property name	Java constant	Valid values	Description	Example

	to use			
cargo.orion.rmi.port	OrionPropertySet.RMI_PORT	Integer	Port for the Orion RMI server	"25791"

Instantiating in Java

```
Container container = new Orion1xContainer();
[...]
```

Instantiating in Ant

```
<cargo-orion1x [...]
</cargo-orion1x>
```

Tomcat 3.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✗	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PO	✓ T	
	GeneralPropertySet.H	✗ TNAME	
	GeneralPropertySet.L	✓ GING	

Instantiating in Java

```
Container container = new Tomcat3xContainer();
[...]
```

Instantiating in Ant

```
<cargo-tomcat3x [...]
</cargo-tomcat3x>
```

Tomcat 4.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	Does not support `META-INF/context.xml` files yet
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✗	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PORT	✓	
	GeneralPropertySet.HOSTNAME	✓	
	GeneralPropertySet.LOGGING	✓	

Custom configuration properties:

Property name	Java constant to use	Valid values	Description	Example
cargo.tomcat.shutdown.port	TomcatPropertySet.SHUTDOWN_PORT	integer	TCP/IP port number on which this server waits for a shutdown command	"8205"

Instantiating in Java

```
Container container = new Tomcat4xContainer();
[...]
```

Instantiating in Ant

```
<cargo-tomcat4x [...]
</cargo-tomcat4x>
```

Tomcat 5.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Note: Tomcat 5.5.x is supported (Requires JDK 1.5+)

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✗	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PO[illegible]T	✓	
	GeneralPropertySet.H[illegible]TNAME	✓	
	GeneralPropertySet.L[illegible]GING	✓	

Custom configuration properties:

Property name	Java constant to use	Valid values	Description	Example
cargo.tomcat.shutdown.port	TomcatPropertySet.SHUTDOWN_PORT	integer	TCP/IP port number on which this server waits for a shutdown command	"8205"

Instantiating in Java

```
Container container = new Tomcat5xContainer();
[...]
```

Instantiating in Ant

```
<cargo-tomcat5x [...]
</cargo-tomcat5x>
```

Jetty 4.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✗	
	Standalone mode	✗	
	Embedded mode	✓	
Ant API	Ant support	✗	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PO	✓	
	GeneralPropertySet.H	✗ TNAME	
	GeneralPropertySet.L	✗ GING	

Instantiating in Java

```
Container container = new Jetty4xEmbeddedContainer();
[...]
```

JBoss 3.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Note: Not implemented yet

Feature category	Feature name	Supported	Comments
Java API	Start	❌	
	Stop	❌	
	Extra classpath passing	❌	
	Container factory	❌	
	Debugging	❌	
	Passing system properties	❌	
	Standalone configuration	❌	
	Existing configuration	❌	
	Static deployment of WAR	❌	
	Static deployment of expanded WAR	❌	
	Static deployment of EAR	❌	
	Standalone mode	❌	
	Embedded mode	❌	
Ant API	Ant support	❌	
Maven API	Maven support	❌	
Properties	ServletPropertySet.PORT	❌	
	GeneralPropertySet.HOSTNAME	❌	
	GeneralPropertySet.LOGGING	❌	

Instantiating in Java

```
Container container = new JBoss3x();
[...]
```

Instantiating in Ant

```
<cargo-jboss3x [...]
</cargo-jboss3x>
```

This page last changed on Oct 23, 2004 by vmassol.

Definition

Explain how to perform debugging when something doesn't work in Cargo. Indeed it can happen that the container does not start or stop as expected. Or that some deployable does not deploy fine. Or whatever else! Here is a short list of things you can do to try debugging the problem.

Redirecting container output to a file

The `container.setOutput(File)` API allows you redirect the container console (stdout) to a file. This is the first file you should check in case of problem.

Example using the Java API

Starting Tomcat 4.x specifying an output console log file:

```
Container container = new Tomcat4xContainer();
container.setHomeDir("c:/apps/jakarta-tomcat-4.1.30");

container.setOutput("target/output.log");

container.start();
```

Use the `container.setAppend(true|false)` method to decide whether the log file is recreated or whether it is appended to, keeping the previous execution logs.

Example using the Ant API

Starting Tomcat 4.x specifying an output console log file:

```
<cargo-tomcat4x homeDir="c:/apps/jakarta-tomcat-4.1.30" action="start"
    output="target/output.log"/>
```

Use the `append="true|false"` attribute to decide whether the log file is recreated or whether it is appended to, keeping the previous execution logs.

Generating Cargo logs

Some Cargo classes support generation of logs. This is implemented through the notion of Monitor.

For example to turn on logging monitoring on a `Container` class, you can use:

```
Monitor fileMonitor = new FileMonitor(new File("c:/tmp/cargo.log"), true);
container.setMonitor(fileMonitor);
```

There are several Monitors that are readily available in the Cargo distribution:

- FileMonitor: logs messages to a file
- SimpleMonitor: logs messages to the console (stdout)

Turning on container logs

Cargo is able to configure containers to generate various levels logs. There are 3 levels defined: "low", "medium" and "high". They represent the quantity of information you wish in the generated log file. You can turn on container logging by using the following API:

```
container.setProperty(GeneralPropertySet.LOGGING, "medium");
```

The generated log files will then be found in the Working directory you have specified on the container (through the `container.setWorkingDir()` call).

When using the Ant tasks, you can specify the log file by using the `log` attribute. For example:

```
<cargo-resin3x [...] log="target/cargo.log"/>
```

News

This page last changed on Oct 03, 2004 by vmassol.

Add Cargo news here using the blog feature.

Oc4J 9.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✓	
	Static deployment of EAR	✓	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PO T	✓	
	GeneralPropertySet.H TNAME	✗	
	GeneralPropertySet.L GING	✗	

Custom configuration properties:

Property name	Java constant	Valid values	Description	Example

	to use			
cargo.orion.rmi.port	OrionPropertySet.RMI_PORT	Integer	Port for the Orion RMI server	"25791"

Instantiating in Java

```
Container container = new Oc4j9xContainer();
[...]
```

Instantiating in Ant

```
<cargo-oc4j9x [...]
</cargo-oc4j9x>
```

Test Results

This page last changed on Oct 20, 2004 by vmassol.

Core Java API unit tests:

Ant API unit tests:

Samples/Java functional tests:

Samples/Ant functional tests:

Features

This page last changed on Oct 22, 2004 by vmassol.

- Ant support — Cargo provides Ant tasks to perform all the operations available from the Java API
- Classpath configuration — How to configure Cargo's classpath
- Configuration — Specifies how the container is configured
- Configuration properties — Properties to configure a container (request port, shutdown port, logging level, threads, etc)
- Container Factory — Instantiate a container by name
- Container instance creation — Create a container instance
- Debugging — Explain how to perform debugging when something doesn't work in Cargo
- Deployable — Deployables are archives (WAR, EAR, etc) that can be deployed in the container
- Deployment descriptor API — API to manipulate J2EE descriptors (currently `web.xml` and `application.xml`)
- Embedded mode
- Existing configuration — Not yet implemented
- Installer — Installs a container
- Maven support — Not yet implemented
- Passing system properties — How to pass system properties that will be available to the container while executing
- Standalone configuration — Configures your container in a specific directory
- Standalone mode
- Start — Start a container that is not already running
- Static deployment of EAR — Deploy an EAR that will be started when the container starts
- Static deployment of expanded WAR
- Static deployment of WAR — Deploy a WAR that will be started when the container starts
- Stop — Stop a running container

Start

This page last changed on Oct 23, 2004 by vmassol.

Definition

Start a container that is not already running

Explanation

First you need to create a `Container` instance. This can be done using the container factory or directly by instating a container implementation class.

Once you have this container instance, starting the container is as simple as calling the `start()` method. Before doing this though you'll need to ensure you have defined the container's `homeDir` (if you're using a container in standalone mode - It's not required for containers in embedded mode).

Of course it you wish to statically deploy archives, you'll need to add deployables to the container.

It is important to note that the `Container.start()` method will wait until the container is **fully started** before returning.

Example using the Java API

Starting Resin 3.x with no deployable:

```
Container container = new Resin3xContainer();
container.setHomeDir("c:/apps/resin-3.0.8");

container.start();
```

Example using the Ant API

Before being able to use the Cargo Ant tasks you need to register them against Ant. This is done by using the Ant `<taskdef>` element. See the Ant support page. The action to start the container is specified using the `action="start"` attribute as shown below.

Starting Resin 3.x with no deployable:

```
<cargo-resin3x homeDir="c:/apps/resin-3.0.8" action="start"/>
```

Stop

This page last changed on Oct 23, 2004 by vmassol.

Definition

Stop a running container

Note: The stop action waits till the container is fully stopped before returning.

Example using the Java API

Stopping Orion 1.x:

```
Container container = new Orion1xContainer();
container.setHomeDir("c:/apps/orion-1.6.0b");

container.stop();
```

Example using the Ant API

Stopping Orion 1.x:

```
<cargo-orion1x homeDir="c:/apps/orion-1.6.0b" action="stop"/>
```

Standalone configuration

This page last changed on Oct 24, 2004 by vmassol.

Definition

Configures your container in a specific directory

Explanation

The standalone configuration allows configuring your container so that it is setup to start in a directory you choose (see the configuration page for more general explanations).

You use the standalone configuration by creating an instance of the `StandaloneConfiguration` class. The constructors of `StandaloneConfiguration` are:

```
public StandaloneConfiguration(String configurationId)
public StandaloneConfiguration(File dir)
```

With the first constructor you specify a configuration id (any id will do) and Cargo will setup your container to start in a `configurationId` directory in your tmp directory (pointed to by the `java.io.tmpdir` System property). The second constructor allows to specify exactly where you wish the configuration directory to be.

Note that if you don't specify any configuration, Cargo will use a standalone configuration by default with a `configurationId` named after your container type (e.g. "tomcat5x" for a Tomcat 5.x container).

Example using the Java API

Start a Tomcat 5.x container and set it up to start in a `target/tomcat5x` directory:

```
StandaloneConfiguration configuration =
    new StandaloneConfiguration(new File("target/tomcat5x"));
Container container = new Tomcat5xContainer();
[...]
container.setConfiguration(configuration);
[...]
container.start();
```

Example using the Ant API

Start a Tomcat 5.x container and set it up to start in a `target/tomcat5x` directory:

```
<cargo-tomcat5x workingDir="target/tomcat5x"[...]>
[...]
</cargo-tomcat5x>
```

Note: The Ant task is still using the old `workingDir` attribute that was there in Cargo 0.2 (which has been deprecated for the Java API). In the future it is planned to be removed and replaced by a nested `<configuration>` (or `<standaloneConfiguration>`) element.

Existing configuration

This page last changed on Oct 24, 2004 by vmassol.

Definition

Not yet implemented

Installer

This page last changed on Oct 23, 2004 by vmassol.

Definition

Installs a container

Explanation

An Installer is meant to install a container. There is currently only a single Installer implementation: `ZipURLInstaller` which downloads a zipped container distribution from a URL and which installs it (i.e. unpacks it) in a specified directory. This is useful if you wish to fully automate a container installation without having to ask the user to manually install a container on their machine.

Example

```
Installer installer = new ZipURLInstaller(
    "http://www.caucho.com/download/resin-3.0.9.zip",
    "target/installs");
installer.install();

Container container = new Resin3xContainer();
container.setHomeDir(installer.getHomeDir());
[...]
```

Static deployment of WAR

This page last changed on Oct 22, 2004 by vmassol.

Definition

Deploy a WAR that will be started when the container starts

Example

Let's see how to Jetty 4.x (in embedded mode) with a WAR to deploy in it.

Note: Unlike the other containers, the Jetty integration does not require the Jetty container to be installed. You simply need to add the Jetty jar (`org.mortbay.jetty.jar`), the Servlet API jar (`servletapi.jar`), and the Tomcat Jasper jars (`jasper-compiler.jar`, `jasper-runtime.jar`) to your classpath. Thus the `homeDir` property has not effect.

```
Container container = new Jetty4xEmbeddedContainer();

Deployable war = container.getDeployableFactory().createWAR("src/data/some.war");
container.addDeployable(war);

container.start();
```

Static deployment of expanded WAR

This page last changed on Oct 22, 2004 by vmassol.

Static deployment of EAR

This page last changed on Oct 23, 2004 by vmassol.

Definition

Deploy an EAR that will be started when the container starts

Example using the Java API

Starting Orion 2.x with an EAR to deploy:

```
Container container = new Orion2xContainer();
container.setHomeDir("c:/apps/orion-2.0.3");

Deployable ear = container.getDeployableFactory().createEAR("src/data/some.ear");
container.addDeployable(ear);

container.start();
```

Example using the Ant API

Starting Orion 2.x with an EAR to deploy:

```
<cargo-orion2x homeDir="c:/apps/orion-2.0.3" action="start">
  <ear earFile="src/data/some.ear"/>
</cargo-orion2x>
```

Standalone mode

This page last changed on Oct 22, 2004 by vmassol.

Embedded mode

This page last changed on Oct 22, 2004 by vmassol.

Cargo provides different container implementations. A Container implementation can be either standalone or embedded. The embedded mode means that Cargo is using directly the container's Java API to control it. If you're using one of the embedded implementation you'll need to ensure that you have the container's jars in your classpath.

Advantages of embbeded mode:

- Faster. There's no need to start a new JVM nor new threads.
- Simpler. There's no need to install the container in a directory

Here is the list of container implementations that support the embedded mode:

- `Jetty4xEmbeddedContainer`: Jetty 4.x implementation

Configuration properties

This page last changed on Oct 23, 2004 by vmassol.

Definition

Properties to configure a container (request port, shutdown port, logging level, threads, etc)

Explanations

It is possible to set container configuration properties using the Cargo API. These properties are applied to a Configuration.

There are 2 kinds of properties:

- General properties
- Container-specific properties. See each container's page for a list of the custom properties it supports.

General properties:

Property name	Java constant to use	Valid values	Description	Example
cargo.servlet.port	ServletPropertySet.PORT	integer	Port on which the Servlet/JSP container will listen to	"8280"
cargo.hostname	GeneralPropertySet.HOSTNAME	string	Host name on which the container will listen to	"myserver"
cargo.logging	GeneralPropertySet.LOGGING	"low", "medium" or "high"	Level representing the quantity of information we wish to log	"medium"

Example using the Java API

Starting Tomcat 5.x on a specific port:

```
Container container = new Tomcat5xContainer();
container.setHomeDir("c:/apps/jakarta-tomcat-5.0.29");
Configuration configuration = new StandaloneConfiguration("tomcat5x");

configuration.setProperty(ServletPropertySet.PORT, "8081");

container.setConfiguration(configuration);
[...]
```

Example using the Ant API

Starting Tomcat 5.x on a specific port:

```
<cargo-tomcat5x homeDir="c:/apps/jakarta-tomcat-5.0.29" action="start">
  <property name="cargo.servlet.port" value="8081"/>
</cargo-tomcat5x>
```

Container Factory

This page last changed on Oct 22, 2004 by vmassol.

Definition

Instantiate a container by name

Explanation

There are 2 solutions to instantiate a container:

- by explicitely creating a new instance of the container itself. For example to instantiate a Resin 3.x container:

```
Container container = new Resin3xContainer();
```

- by using the `ContainerFactory` class. The advantage is then that you can instantiate by name and thus your code can be generic which is nice if you plan to run the same code with different containers. For example, to instantiate a Resin 3.x container:

```
ContainerFactory factory = new ContainerFactory();
Container container = factory.createContainer("resin3x");
```

Note: You can also pass the full container class name (that's useful if you wish to instantiate a custom container you have developed):

```
Container container =
factory.createContainer("org.codehaus.cargo.container.resin.Resin3xContainer");
```

Containers

This page last changed on Oct 22, 2004 by vmassol.

Here is the list of supported containers.

- JBoss 3.x
- Jetty 4.x
- Oc4J 9.x
- Orion 1.x
- Orion 2.x
- Resin 2.x
- Resin 3.x
- Tomcat 3.x
- Tomcat 4.x
- Tomcat 5.x
- WebLogic 8.x

Configuration

This page last changed on Oct 23, 2004 by vmassol.

Definition

Specifies how the container is configured

Explanation

The notion of Configuration is different from the notion of Installation. Indeed a container is installed in the place where you have installed it (or where the Cargo Installer has installed it).

However, it is possible to configure this container to tell it deploy archives to some other directories and consider this other directory as the new home directory of the container. All containers support this feature. This allows to leave an installed container directory intact (no modifications) and to compartiment all modifications to some directory that you are controlling. Say you wish to automate some functional tests. You can set the new container home to point to your temporary build directory. This is the configuration we call Standalone configuration.

By opposition if you choose to use the default configuration that your container has set up when you installed it, it is called an Existing configuration.

Please note that at the moment Cargo only support Standalone configurations.

Deployable

This page last changed on Oct 23, 2004 by vmassol.

Definition

Deployables are archives (WAR, EAR, etc) that can be deployed in the container

Explanation

A Deployable class is a wrapper class around a physical archive. Deployable}}s are constructed using a {{DeployableFactory provided by your container. The reason for this factory is to support container extensions to archives (for example, Tomcat supports context.xml files located in your WAR's META-INF directory, JBoss allows for a jboss-web.xml located in your WAR, etc).

The DeployableFactory interface offers different methods for creating Deployable}}s (e.g. {{DeployableFactory.createEAR(String), DeployableFactory.createWAR(String), etc).

Once you have a Deployable instance wrapping your archive, you can tell the Container to deploy it when the container starts. This is achieved by calling the Container.addDeployable(Deployable) API. You can also read how to statically deploy a WAR or how to statically deploy an EAR.

In the near future we'll add support to deploy {{Deployable}}s in a running container (a.k.a. dynamic deployments).

Example using the Java API

Deploying a WAR in Tomcat 5.x:

```
Container container = new Tomcat5xContainer();
container.setHomeDir("c:/apps/tomcat-5.0.29");

DeployableFactory factory = container.getDeployableFactory();
WAR war = factory.createWAR("path/to/my.war");

container.addDeployable(war);

container.start();
```

Example using the Ant API

Depploying a WAR in Tomcat 5.x:

```
<cargo-tomcat5x homeDir="c:/apps/tomcat-5.0.29" action="start">
  <war warfile="path/to/my.war"/>
</cargo-tomcat5x>
```

Ant support

This page last changed on Oct 23, 2004 by vmassol.

Definition

Cargo provides Ant tasks to perform all the operations available from the Java API

Explanation

Before using the Ant API you need to register the Cargo Ant tasks into Ant. This is done in the following manner:

```
<taskdef resource="cargo.tasks">
  <classpath>
    <pathelement location="${cargo.jar}"/>
  </classpath>
</taskdef>
```

Example

Here's a full example showing how to deploy a WAR, and expanded WAR and an EAR in an Orion 2.x container. Please note that the `output` and `log` attribute are optional. The `property` elements allow you to tune how the container is configured. Here we're telling it to start on port 8180 and to generate the maximum amount of logs in the container `output` file.

```
<taskdef resource="cargo.tasks">
  <classpath>
    <pathelement location="path/to/cargo.jar"/>
  </classpath>
</taskdef>

<cargo-orion2x homeDir="c:/apps/orion-2.0.3" output="target/output.log"
    log="target/cargo.log" action="start">
  <war warfile="path/to/my/simple.war"/>
  <war warfile="path/to/my/expandedwar/simple"/>
  <ear earfile="path/to/my/simple.ear"/>
  <property name="cargo.servlet.port" value="8180"/>
  <property name="cargo.logging" value="high"/>
</cargo-orion2x>
```

Deployment descriptor API

This page last changed on Oct 22, 2004 by vmassol.

Definition

API to manipulate J2EE descriptors (currently `web.xml` and `application.xml`)

Explanation

TODO. Most notably the API allows merging two `web.xml` files.

Example

TODO

Container instance creation

This page last changed on Oct 22, 2004 by vmassol.

Definition

Create a container instance

Explanation

A container instance is created by simply instantiating the Java object implementing the container. Each container implementation offers a main Java object wrapping its container and which allows to manipulate the container (start, stop, configure, deploy archives, etc).

The class to use for instantiating a container can be found in each container's page:

- JBoss 3.x
- Jetty 4.x
- Oc4J 9.x
- Orion 1.x
- Orion 2.x
- Resin 2.x
- Resin 3.x
- Tomcat 3.x
- Tomcat 4.x
- Tomcat 5.x
- WebLogic 8.x

In addition it's possible to instantiate a container by name.

Example

```
Container container = new Orion2xContainer();
[...]
Container container = new Resin3xContainer();
[...]
Container container = new Weblogic8xContainer();
[...]
```

Classpath configuration

This page last changed on Oct 23, 2004 by vmassol.

Definition

How to configure Cargo's classpath

Explanation

TODO (explain the 2 modes: standalone and embedded and how Cargo treats both WRT classpath. Explain also the setExtraClasspath() API)

Example using the Java API

Starting Orion 1.x with Clover jar added to its classpath. For example if you have instrumented your source code with Clover you'll need to add the Clover jar to the classpath.

```
Container container = new Orion1xContainer();
container.setHomeDir("c:/apps/orion-1.6.0b");

container.setExtraClasspath(new String[] { "libs/clover.jar" });

container.start();
```

Example using the Ant API

Starting Orion 1.x with some additional classpath entries:

```
<cargo-orion1x homeDir="c:/apps/orion-1.6.0b" action="start">
  <extraClasspath>
    <pathelement location="libs/clover.jar"/>
  </extraClasspath>
</cargo-orion1x>
```

Passing system properties

This page last changed on Oct 23, 2004 by vmassol.

Definition

How to pass system properties that will be available to the container while executing

Explanations

TODO

Example using the Java API

Starting Tomcat 3.x with some System properties set in the container JVM:

```
Container container = new Tomcat3xContainer();
container.setHomeDir("c:/apps/jakarta-tomcat-3.3.2");

Map props = new HashMap();
props.put("mypropery", "myvalue");
container.setSystemProperties(props);

container.start();
```

Example using the Ant API

Starting Tomcat 3.x with some System properties set in the container JVM:

```
<cargo-tomcat3x homeDir="c:/apps/jakarta-tomcat-3.3.2" action="start">
  <sysproperty key="myproperty" value="myvalue"/>
</cargo-tomcat3x>
```

WebLogic 8.x

This page last changed on Oct 23, 2004 by vmassol.

Supported Features

Feature category	Feature name	Supported	Comments
Java API	Start	✓	
	Stop	✓	
	Extra classpath passing	✓	
	Container factory	✓	
	Debugging	✓	
	Passing system properties	✓	
	Standalone configuration	✓	
	Existing configuration	✗	
	Static deployment of WAR	✓	
	Static deployment of expanded WAR	✗	
	Static deployment of EAR	✓	
	Standalone mode	✓	
	Embedded mode	✗	
Ant API	Ant support	✓	
Maven API	Maven support	✗	
Properties	ServletPropertySet.PO[illegible]T	✓	
	GeneralPropertySet.H[illegible]TNAME	✗	
	GeneralPropertySet.L[illegible]GING	✗	

Instantiating in Java

```
Container container = new WebLogic8xContainer();
[...]
```

Instantiating in Ant

```
<cargo-weblogic8x [...]
</cargo-weblogic8x>
```

Release procedure

This page last changed on Oct 29, 2004 by vmassol.

1. Perform a clean SVN checkout
2. Edit top level project.xml and modify `<currentVersion>` tag. Ex: from `0.3-SNAPSHOT` to `0.3`
3. Run `maven` at top level to generate the distribution
4. Run `maven cargo:site` to generate the development site
5. Zip `target/maven/docs` to docs.zip and upload it to `beaver.codehaus.org:/home/projects/cargo/public_html`
6. Log on `beaver.codehaus.org` and unzip docs.zip to overwrite the existing older docs
7. Upload the Cargo jar located in `distribution/target/maven/cargo-<version>.jar` to `beaver.codehaus.org:/home/projects/cargo/dist`. Also make sure you also run md5 to generate the checksum.
8. Log onto Cargo JIRA, release the current version and add the next version
9. Check that the Cargo wiki is up to date. Specifically, perform the following updates:
   a. modify the status on the home page about the delivery
   b. export the wiki to PDF and add the generated PDF to the docs archive page
   c. modify the navigation page to include the latest download link and docs
10. Send an announcement email to Cargo dev and Cargo user mailing lists (and to any other onsite site, magazines, etc)
11. Tag SVN by copying the HEAD to `.../tags/<version>`
12. Modify again `project.xml` and modify `<currentVersion>` for the next version