Project Structure

> This page has been automatically generated by Cargo's build. Do not edit it directly as it'll be overwritten next time it's generated again.

Cargo's directory organization can be daunting for a newcomer. So here's some information on how the project is organized.

> **Legend**
> - **directory/** : represents a directory
> - **directory/** : represents a directory containing a Maven project

- **CARGO/**: Top level POM to build all other Cargo modules
  - **pom/**: Cargo is a thin wrapper that allows you to manipulate Java EE containers in a standard way.
  - **resources/**: Cargo Shared Resources (License files, Checkstyle configuration files, etc)
    - **build-tools/**: Cargo Build Tools
    - **testdata/**: Test data for the Sample applications
      - **empty-jar/**: Cargo Empty JAR test data for samples
      - **empty-ear/**: Cargo Empty EAR test data for samples
      - **simple-jar/**: Cargo Simple JAR test data for samples
      - **simple-war/**: Cargo Simple WAR test data for samples
      - **datasource-war/**: Cargo DataSource WAR test data for samples
      - **two-datasources-war/**: Cargo Multiple DataSource WAR test data for samples
      - **xadatasource-war/**: Cargo XADataSource WAR test data for samples
      - **mailsession-war/**: Cargo Mail Session WAR test data for samples
      - **datasource-cmt-local-war/**: Cargo Local Transaction DataSource WAR test data for samples
      - **classpath-war/**: Cargo Classpath WAR test data for samples
      - **simple-ejb/**: Cargo Simple EJB test data for samples
      - **simple-ear/**: Cargo Simple EAR test data for samples
      - **authentication-war/**: Cargo Authenticated WAR test data for samples
      - **expanded-war/**: Cargo Expanded WAR test data for samples
      - **tomcat-context/**: Cargo Tomcat context WAR test data for samples
      - **simple-bundle/**: Cargo Simple OSGi bundle test data for samples
      - **simple-har/**: Cargo Simple HAR test data for samples
      - **simple-aop/**: Cargo Simple AOP test data for samples
      - **systemproperty-war/**: Cargo System Property test WAR data for samples
    - **jetty-6-and-earlier-deployer/**: Webapp used by Cargo to remotely deploy applications of a Jetty server
    - **jetty-7-and-onwards-deployer/**: Webapp used by Cargo to remotely deploy applications of a Jetty server
  - **core/**: Cargo Core
    - **api/**: The Cargo Core API
      - **util/**: Core Utility API (logging, Ant utils, File utils, etc)
      - **module/**: Core API to parse/create J2EE Modules
      - **container/**: API to start/stop/configure containers
      - **generic/**: Core API to generically access the typed container API
    - **tools/**: Tools for the Core API
      - **jboss-deployer-api/**: JBoss remote deployer API
      - **jboss-deployer-5/**: Remote deployer for JBoss 5.0.x
      - **jboss-deployer-5.1-and-6/**: Remote deployer for JBoss 5.1.x and 6.x

- **containers/**: Core API Container Implementations
    - **geronimo/**: Core API implementation for Geronimo containers
    - **glassfish/**: Core API implementation for GlassFish containers
    - **jboss/**: Core API implementation for JBoss containers
    - **jetty/**: Core API implementation for Jetty containers
    - **jo/**: Core API implementation for jo! containers
    - **jonas/**: Core API implementation for JOnAS containers
    - **jrun/**: Core API implementation for JRun containers
    - **orion/**: Core API implementation for Orion containers
    - **resin/**: Core API implementation for Resin containers
    - **tomcat/**: Core API implementation for Tomcat containers
    - **weblogic/**: Core API implementation for WebLogic containers
    - **websphere/**: Core API implementation for WebSphere containers
- **uberjar/**: The Cargo Core Uberjar
- **samples/**: Samples that exercises the Cargo Java APIs
    - **java/**: Sample application that exercises the Cargo Java API
- **documentation/**: Cargo Website Resources

- **extensions/**: Cargo Extensions
    - **ant/**: Cargo Extensions for ANT
        - **tasks/**: Ant tasks for Cargo
        - **samples/**: Cargo Samples for ANT
            - **configurationFiles-test/**: Sample application that tests the configuration files options
            - **datasource-test/**: Sample application that exercises the Cargo ANT tasks with a container and a datasource
            - **remote-test/**: Sample application that exercises the Cargo ANT tasks with a remote container
    - **daemon/**: Cargo Extensions for Daemon
        - **client/**: Cargo Daemon Client
        - **webapp/**: Cargo Daemon Web Application
        - **itests/**: Cargo Daemon Integration Tests
            - **deployable-on-jetty/**: Cargo Daemon Client - Deployable on Jetty
            - **deployable-on-tomcat/**: Cargo Daemon Client - Deployable on Tomcat
    - **maven2/**: Cargo Extensions for Maven 2
        - **plugin/**: Cargo Maven 2 Plugin
        - **archetypes/**: Sample Maven archetypes for using Cargo
            - **webapp-single-module/**: Sample Maven archetype showing how to configure Cargo and Maven to run functional tests
              directly from a single webapp module.
            - **webapp-functional-tests-module/**: Sample Maven archetype showing how to configure Cargo and Maven to run functional tests
              for a webapp by creating a separate functional-tests module.
            - **webapp-with-datasource/**: Sample Maven archetype showing how to configure Cargo and Maven to run functional tests
              directly from a webapp with datasource. Cargo will be used to configure the datasource on the
              container.
            - **remote-deployment/**: Sample Maven archetype showing how to configure Cargo and Maven to generate a webapp and deploy
              it to a remote container.
        - **samples/**: Cargo Samples for Maven 2
            - **artifactInstaller-test/**: Sample application that tests the artifact installer
            - **autoDeployable-test/**: Sample application that exercises the Cargo Maven2

plugin's AutoDeployable capabilities
    - **configurationFiles-test/**: Sample application that tests the configuration files options
    - **inPlaceDevelopment-test/**: Sample application that exercises the Cargo Maven2 plugin's in-place development capabilities. Note that this sample currently works only for Jetty and Tomcat as it requires that container implementations support deploying an expanded WAR from a directory without copying it to the container deploy directory.
    - **jetty7x-embedded-test/**: Sample application that exercises the Cargo Maven2 plugin with an embedded Jetty 7.x container
    - **jetty7x-root-context-test/**: Sample application that exercises the Cargo Maven2 plugin with a Jetty 7.x container and the root context
    - **ping-test/**: Sample application that exercises the Cargo Maven2 plugin's deployable pinging capabilities
    - **multi-datasource-test/**: Sample application that exercises the Cargo Maven2 plugin with a Tomcat 6.x container and multiple datasources
    - **remoteDeploy-test/**: Sample application that tries remote deployment
    - **runMojo-test/**: Sample application that tries the cargo:run MOJO
    - **uberwar-test/**: Sample application that exercises the Cargo Maven2 plugin's uberwar feature
    - **websphere-test/**: Sample application that will be used to test the WebSphere container
- **sandbox/**: Base directory for cargo plugin projects
  - **intellijidea/** : IntelliJ IDEA plugin
  - **netbeans/** : Netbeans plugin