Boot Documentation Library
Boot User Guide

Boot User Guide (DRAFT)

abstract

draft intro...

Contents

Guide Abstract

The content of this UG is drawn from the *Boot Wiki* and the *Boot Github repo*.

Summary

Notice

Notice

Topics:

This information was developed for products and services offered in the U.S.A.

This product is meant for educational purposes only. Some of the trademarks mentioned in this product appear for identification purposes only. Not responsible for direct, indirect, incidental or consequential damages resulting from any defect, error or failure to perform. Any resemblance to real persons, living or dead is purely coincidental. Void where prohibited. Some assembly required. Batteries not included. Use only as directed. Do not use while operating a motor vehicle or heavy equipment. Do not fold, spindle or mutilate. Do not stamp. No user-serviceable parts inside. Subject to change without notice. Drop in any mailbox. No postage necessary if mailed in the United States. Postage will be paid by addressee. Post office will not deliver without postage. Some equipment shown is optional. Objects in mirror may be closer than they appear. Not recommended for children. Your mileage may vary.

No other warranty expressed or implied. This supersedes all previous notices.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. Retro Tools, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of the Retro Tools in the United States, other countries, or both:

RetroWrench®

The following terms are trademarks of other companies:

Red, Orange, Yellow, Green, Blue, Indigo, and Violot are registered trademarks of Rainbow Corporation and/or its affiliates.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

Preface

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed finibus rutrum pharetra. Etiam porttitor purus non felis semper, vel lobortis sem ullamcorper. Donec ut interdum turpis, non placerat lacus. Quisque placerat lacus id magna rhoncus, nec lacinia massa blandit. Pellentesque faucibus, dolor vitae accumsan pretium, arcu mauris eleifend felis, a iaculis justo nisl vel quam. Fusce laoreet turpis et finibus molestie. Suspendisse maximus scelerisque dui, vel vestibulum libero porttitor id. In et libero erat. Integer et dolor eget tellus dictum fermentum. Nunc velit elit, eleifend et placerat non, convallis eget mauris. Fusce congue ipsum ac commodo tincidunt. Mauris varius vulputate ante sit amet consequat. Cras et finibus est, fringilla vestibulum turpis. Quisque consectetur felis at nibh vulputate, id bibendum sapien venenatis. Mauris dapibus accumsan ornare.

About this Document

This document ...

Revision: 0.1

Acknowledgements

acks here...

Part
I

Part 1: Using Boot

Topics:

This is the overview of Part I ...

Fusce porta leo sem, non luctus lectus fermentum et. Nunc lacus mi, ultricies et ex sit amet, porttitor convallis ligula. Praesent convallis nibh id lectus pellentesque, et rutrum nibh molestie. Nunc ac convallis lectus. Aliquam eu sem eget est mollis iaculis quis eu mauris. Vestibulum iaculis turpis a urna sagittis, vitae pellentesque augue venenatis. Fusce ut efficitur arcu. Mauris volutpat velit quis purus consequat, sit amet molestie mauris aliquam. Curabitur dolor eros, congue eget erat sed, suscipit auctor erat. Cras convallis ex in sem placerat ornare non at felis.

Chapter 1

Getting Started

Topics:

concepts overview...

Installation

overview...

Configuration

There are four levels of configuration involved:

- JVM
- Clojure
- Boot
- Task

Configuring Boot means controlling the JVM environment before your project is loaded. Configuring your Clojure project means declaring dependencies, specifying which tasks to run, etc. Environment variables and boot.properties are used in the first case, profile.boot and build.boot in the latter.

Related information

Configuring Boot (wiki page)

JVM Configuration

Configuring Boot means controlling the JVM environment before your project is loaded.

Configuring the Java environment that bootstraps Clojure is done via:

- JVM system properties
- Environment variables
- Boot properties files
- • `BOOT_HOME/boot.properties` (the global configuration file)
  - `./boot.properties` (the local project configuration)
- a `.bootignore` file (analagous to `.gitignore`)

Boot being a self-documented toolchain, you can query the environment variables and properties files that Boot understands by invoking `boot -h` on the command line.

The following properties can be set as system properties, environmental variables, or in a `boot.properties` file:

- BOOT_AS_ROOT Set to 'yes' to allow boot to run as root.
- BOOT_CLOJURE_VERSION The version of Clojure boot will provide (1.8.0).
- BOOT_HOME Directory where boot stores global state (~/.boot).
- BOOT_FILE Build script name (build.boot).
- BOOT_JVM_OPTIONS Specify JVM options (Unix/Linux/OSX only). (*note)
- BOOT_LOCAL_REPO The local Maven repo path (~/.m2/repository).
- BOOT_VERSION Specify the version of boot core to use.
- BOOT_COLOR Turn colorized output on or off

Warning: Since BOOT_JVM_OPTIONS is used to launch the initial java process, it must be specified as a system environment variable (e.g. `export BOOT_JVM_OPTIONS=-client)`, not in a boot.properties file.

Related information

Configuring Boot (wiki page)

boot.properties
Usually, DITA builds require setting a number of parameters that do not change frequently. You can reference a set of build parameters defined in a `boot.properties` file when building output with the `dita` command. If needed, you can override any parameter by specifying it explicitly as an argument to the `dita` command.

About `.properties` files

A `.properties` file is a text file that enumerates one or more name-value pairs, one per line, in the format `name = value`. The `.properties` filename extension is customarily used, but is not required.

- Lines beginning with the `#` character are comments.
- Properties specified as arguments of the `dita` command override those set in `.properties` files.

  Restriction: For this reason, args.input and transtype can't be set in the `.properties` file.
- If you specify the same property more than once, the last instance is used.
- Properties not used by the selected transformation type are ignored.
- Properties can reference other property values defined elsewhere in the `.properties` file or passed by the `dita` command. Use the Ant `${property.name}` syntax.
- You can set properties not only for the default DITA-OT transformation types, but also for custom plugins.

1. Create your `boot.properties` file.

   For example:

   ```
   #http://boot-clj.com
   #Mon Jan 16 06:26:54 CST 2017
   BOOT_EMIT_TARGET=no
   BOOT_CLOJURE_NAME=org.clojure/clojure
   BOOT_VERSION=2.7.1
   BOOT_CLOJURE_VERSION=1.9.0-alpha14
   ```

2. Reference your `.properties` file with the `dita` command when building your output.

   ```
   dita --input=my.ditamap --format=html5 --propertyfile=my.properties
   ```

3. If needed, pass additional arguments to the `dita` command to override specific build parameters.

   For example, to build output once with `<draft>` and `<required-cleanup>` content:

   ```
   dita --input=my.ditamap --format=html5 --propertyfile=my.properties \
        --args.draft=yes
   ```

Configuration Variables in the Environment

The following env variables control boot's configuration:

- BOOT_AS_ROOT Set to 'yes' to allow boot to run as root.
- BOOT_CLOJURE_VERSION The version of Clojure boot will provide (1.8.0).
- BOOT_HOME Directory where boot stores global state (~/.boot).
- BOOT_FILE Build script name (build.boot).
- BOOT_JVM_OPTIONS Specify JVM options (Unix/Linux/OSX only). (*note)
- BOOT_LOCAL_REPO The local Maven repo path (~/.m2/repository).
- BOOT_VERSION Specify the version of boot core to use.
- BOOT_COLOR Turn colorized output on or off

Related information
Configuring Boot (wiki page)

Clojure Configuration

The profile.boot and build.boot scripts are programs that configure your Clojure project. They impact the program that runs once clojure is bootstrapped.

They are evaluated in the following order:

1. `BOOT_HOME/profile.boot`
2. `./profile.boot`
3. `./build.boot`

They are evaluated in the same namespace and environment, so things you define in say `BOOT_HOME/profile.boot` are visible to expressions in `./profile.boot` and `./build.boot`, for example. Expressions that are evaluated later can override, redef, etc. anything done in scripts that were evaluated earlier.

The project-local `profile.boot` script can be useful when you have project-specific configuration that you don't want to keep in version control. Credentials, configuration that is not shared with the team, etc.

Related information

Configuring Boot (wiki page)

`build.boot`

Example: `build.boot` file

The following `build.boot` files ...

Sample `build.boot` file:

```
(def +project+ 'foo/bar)
(def +version+ "0.1.0-SNAPSHOT")

(set-env!
 :asset-paths #{"resources"}
 :source-paths #{"src/java"}
 :resource-paths #{"src/clj"}

 :repositories
 #(conj % ["maven-central" {:url "http://mvnrepository.com"}]
        ["central" "http://repo1.maven.org/maven2/"])

 :dependencies '[[org.clojure/clojure RELEASE]
                 [org.clojure/core.async "0.2.395"]
                 [slingshot "0.12.2"]])

(task-options!
 pom  {:project     +project+
       :version     +version+
       :description "My project"
       :url         "https://github.com/..."
       :scm         {:url "https://github.com/..."}
       :license {"EPL" "http://www.eclipse.org/legal/epl-
v10.html"}}
 push {:repo "clojars"})

(deftask monitor
  "watch etc."
  []
  (comp (watch)
        (notify :audible true)
        (pom)
        (jar)
```

```
          (install)))
```

`profile.boot`

Example: `profile.boot` file

The following `profile.boot` files ...

Sample `profile.boot` file:

```
;; profile.boot is a script file; it can include any clojure

;; for example, this configures stack trace printing
(alter-var-root
 #'boot.from.io.aviso.exception/*fonts*
 assoc :message boot.from.io.aviso.ansi/white-font)
```

Boot Configuration

Boot's configuration is expressed in the environment map, which is usually configured in `build.boot`.

Related information

Boot Environment (wiki page)
Filesets (wiki page)

Boot Environment Map

The keys in boot's environment map are:

:resource-paths	A set of path strings. These paths will supply the content that will be on the classpath of the initial Fileset, and the files contained will marked with roles +INPUT, +OUTPUT and so be emitted as final artifacts.
:source-paths	A set of path strings. These paths will supply the content that will be on the classpath of the initial Fileset, and the files contained will be marked with roles +INPUT,-OUTPUT, so they may be used to generate output but will not themselves be emitted as final artifacts.
... etc...	...

Task Configuration

Mechanisms:

- Task options: command line, build.boot.
- Boot env map `(get-env)`
- Env variables
- EDN files

Related information

Boot Environment (wiki page)

Task Options

Task options:

Related information

Task Options DSL (wiki page)

EDN Configuration Files

A more advanced technique is to read configuration parameters from a file. EDN files are commonly used in this way, although any data format could be used.

See the Boot Task Developer's Guide for details.

Related information

>Extensible Data Notation (EDN)

Chapter 2

Concepts

Topics:

concepts overview...

Tasks

A boot *task* is ...

Related information

Tasks (Wiki page)

Middleware Concepts

The concept of *middleware* is central to boot's concept of a task.

Boot borrows the notion of middleware from *Ring*.

Important: Boot's notion of "middleware" is a little bit peculiar, and should not be confused with other *common uses of the term*.

Anatomy of a Boot Task

A boot task has the following structure:

Pipelines

A boot *pipeline* is ...

Pipeline Concepts

Pipeline processing...

Pipelines are actually two-way, or more accurately, there are two pipelines, connected at the extremities, running in opposite directions. So it's really a circuit rather than a pipeline. Electrical flow is probably a better metaphor than water flow.

Filesets

A boot *fileset* is ...

Related information

Filesets (wiki page)

Fileset Roles

There are three roles ...

Workspaces

A boot *workspace* is ...

An important principle of the boot build process is that tasks do not refer to named places in the filesystem. Tasks may only create files in managed temp directories provided by boot. These temp directories are:

- Anonymous – tasks do not specify the location of the temp dir.
- Local – tasks do not pass references to temp dirs to other tasks.
- Managed – temp dirs are cleaned up by boot as necessary.

In order to communicate files in these temp directories to the rest of the build process they must be added to the fileset object, described below.

Related information

Tasks (Wiki page)

Workspace Concepts

"Workspace" is a metaphor for ...

Pods

A boot *pod* is ...

Related information

Pods (wiki page)

Pod Concepts

Pod concept short desc ...

Chapter 3

Developing Boot Tasks

Topics:

overview...

Boot APIs

overview...

Tasklib Development Workflow

overview...

Chapter 4

Troubleshooting

Topics:

Tips

The verbosity of the task logging can be increased by using the `-v` flag. For even more verbosity -vv can be used. Eg. `$ boot -v build`

In the REPL you use it like this (1-3, increasing verbosity):

```
(reset! boot.util/*verbosity* 2) (boot (build))
```

Suppose you do `boot -v foobar` and you get a stack trace. You can then do `boot -vb |cat -n` and see the matching line numbers. Actually you should do `boot -vb foobar` to get the exact same generated code; just add `-b` to the thing that caused the error.

`boot show` is your friend. remember: it's a task. that means you can insert it in anywhere in a pipeline in order to dump useful info to stdout. For example, `boot show -f my-task show -f` will print the before and after filesets for my-task.

Related information

Troubleshooting (wiki page)

No reader function

No reader function for tag object

Condition

You get this message when you run boot.

Cause

May happen if you try to access something external from within a pod. Only forms that can be printed with pr-str and read via read-string can pass between pods. That's why you see that exception.

This is due to the way Clojure works; interfaces and classes are created dynamically at runtime, so two clojure runtimes can't understand each other's clojure things. Remember that pods are runtimes.

Remedy

You want to pass in the string path to the java.io.File object, not the object itself; i.e. pass `~(.getPath tgt)` instead of `~tgt`. The test it must pass is `(= tgt (read-string (pr-str tgt)))`.

No output files

After running a boot task pipeline, no output is found.

Condition

Cause

Boot tasks work on filesets, which are not written to the user filesystem. Well, they are, but in hidden, private areas. To get the output written to disk you have to tell boot explicitly to do this.

Remedy

The `target` task writes files in the resources and assets filesets to a target directory on disk. By default the directory is `target`.

For example, to write output to `out`:

```
$ boot mytask target -d out
```

Part
II

Part 2: Reference Manual

Topics:

This is a reference manual for Boot.

Fusce porta leo sem, non luctus lectus fermentum et. Nunc lacus mi, ultricies et ex sit amet, porttitor convallis ligula. Praesent convallis nibh id lectus pellentesque, et rutrum nibh molestie. Nunc ac convallis lectus. Aliquam eu sem eget est mollis iaculis quis eu mauris. Vestibulum iaculis turpis a urna sagittis, vitae pellentesque augue venenatis. Fusce ut efficitur arcu. Mauris volutpat velit quis purus consequat, sit amet molestie mauris aliquam. Curabitur dolor eros, congue eget erat sed, suscipit auctor erat. Cras convallis ex in sem placerat ornare non at felis.

Chapter 5

Boot Standard Task Library

Topics:

Boot comes with a set of built-in tasks.

`add-repo`

```
$ boot add-repo --arg ...
```

```
(add-repo :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`add-repo` wiki page.

`aot`

```
$ boot aot --arg ...
```

```
(aot :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`aot` wiki page.

`install`

```
$ boot install --arg ...
```

```
(install :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`aot` wiki page.

help

```
$ boot help --arg ...
```

```
(help :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`help` wiki page.

jar

```
$ boot jar --arg ...
```

```
(jar :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`jar` wiki page.

`javac`

```
$ boot javac --arg ...
```

```
(javac :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`javac` wiki page.

`pom`

```
$ boot pom --arg ...
```

```
(pom :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`pom` wiki page.

push

```
$ boot push --arg ...
```

```
(push :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

push wiki page.

repl

```
$ boot repl --arg ...
```

```
(repl :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

repl wiki page.

show

```
$ boot show --arg ...
```

```
(show :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

show wiki page.

sift

```
$ boot sift --arg ...
```

```
(sift :arg ...)
```

DESCRIPTION

Transform the fileset, matching paths against regexes.

Arguments:

:add-asset

-A

#{str} The set of directory paths to add to assets.

:add-jar

-j

{sym regex} The map of jar to path regex of entries in jar to unpack.

:add-meta

-M

{regex kw} The map of path regex to meta key to add.

:add-resource

-R

#{str} The set of directory paths to add to resources.

:add-source

-S

#{str} The set of directory paths to add to sources.

:include

-i

#{regex} The set of regexes that paths must match.

:invert

-v

bool Invert the sense of matching.

:move

-m

{regex str} The map of regex to replacement path strings.

:to-asset

-a

#{regex} The set of regexes of paths to move to assets.

:to-resource

-r

#{regex} The set of regexes of paths to move to resources.

:to-source

-s

#{regex} The set of regexes of paths to move to sources.

:with-meta

-w

#{kw} The set of metadata keys files must have.

:help

bool

Print this help info.

The `--to-asset`, `--to-resource`, and `--to-source` options move matching paths to the corresponding section of the fileset. This can be used to make source files into resource files, for example, etc. If `--invert` is also specified the transformation is done to paths that do ***NOT*** match.

AUTHOR

Written by John Doe.

Related information

`sift` wiki page.

`target`

```
$ boot target --arg ...
```

```
(target :arg ...)
```

DESCRIPTION

Writes output files to the given directory on the filesystem.

where:

:help

bool

Print this help info.

:dir

#{str}

The set of directories to write to (target). Default: `target`

:no-link

bool

Don't create hard links.

:no-clean

bool

Don't clean target before writing project files.

AUTHOR

Written by John Doe.

Related information

`target` wiki page.

`uber`

```
$ boot uber --arg ...
```

```
(uber :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`uber` wiki page.

`wait`

```
$ boot wait --arg ...
```

```
(wait :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`show` wiki page.

war

```
$ boot war --arg ...
```

```
(war :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

war wiki page.

watch

```
$ boot watch --arg ...
```

```
(watch :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

show wiki page.

web

```
$ boot web --arg ...
```

```
(web :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`web` wiki page.

`zip`

```
$ boot zip --arg ...
```

```
(zip :arg ...)
```

DESCRIPTION

Arguments:

:help

bool

Print this help info.

AUTHOR

Written by John Doe.

Related information

`zip` wiki page.

Chapter 6

Boot API

Topics:

The Boot API consists of several namespaces:

boot.core

DESCRIPTION

The boot.core namespace contains functions that ...

AUTHOR

Written by John Doe.

Related information

`boot.core` wiki page.

input-files

DESCRIPTION

AUTHOR

Written by John Doe.

Related information

`boot.core` wiki page.

tmp-dir

DESCRIPTION

AUTHOR

Written by John Doe.

Related information

`boot.core` wiki page.

boot.pod

DESCRIPTION

The boot.pod namespace contains functions that ...

AUTHOR

Written by John Doe.

Related information

`boot.pod` wiki page.

Appendix

A

Appendix

This appendix describes things that you rarely need to know.

You can consult this section when you need detailed information about a specific component.

Glossary

AVR
: AVR is a kind of microcontroller from Atmel

USB flash drive
: A small portable drive.

Arduino
: Arduino is ...

WSN
: **W**ireless **S**ensor **N**etwork. A network of sensor nodes that communicate by radio.

Index