

sbt Reference Manual

Contents

Preface	3
sbt	3
sbt	3
.	3
Mac sbt	4
.	4
.	4
Windows sbt	4
.	4
Windows	4
Linux sbt	4
.	4
Ubuntu Debian	4
Linux RPM	5
Gentoo	6
Hello, World	6
.	6
.	6
sbt	7
.	7
.	7
.	7
sbt	8
.	8
.	8
.	8
.	8
.	9
.	9
.	9
.	9
Tab	9
.	10
.sbt	10

.	10
.	10
build.sbt	11
Keys	12
tasks settings	13
sbt Keys	13
build.sbt	13
.	14
Scope	14
Key	14
Scope	15
Scope	16
.	16
sbt scope key	16
scoped key	16
scope	17
scope	18
scope	19
.	19
.	19
+= +=	20
key	20
+= +=	22
.	22
.	22
.	22
.	25
.	25
.	26
root	27
.	27
.	28
.	28
.	28
.	28
.	28
.	29
.	30
.	30
.	30
.	30
.	31
.	34
.	35
sbt	35
.	35

.scala	36
	36
	36
sbt:	37
	37
Bare .sbt	37
bare .sbt	37
(0.13.7)	38
.scala	38
build.sbt Build.scala	38
	39
	40

Preface

sbt

sbt sbt

sbt

.sbt scopes

sbt

sbt

sbt

- sbt
- hello world
-
-
- sbt sbt
- .sbt
- Jar Shell

Mac Windows Linux

sbt terminal encoding HTTP JVM

Mac **sbt**

ZIP TGZ

Homebrew

```
$ brew install sbt
```

Macports

```
$ port install sbt
```

Windows **sbt**

ZIP TGZ

Windows

msi

Linux **sbt**

ZIP TGZ

Ubuntu **Debian**

DEB sbt

Ubuntu	Debian	DEB	DEB	apt-get aptitude
Synaptic		sbt	sudo	

```

echo "deb https://dl.bintray.com/sbt/debian /" | sudo tee -a /etc/apt/sources.list.d/sbt.list
sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 2EE0EA64E40A89B84B2DF73499E8
sudo apt-get update
sudo apt-get install sbt

sbt          Bintray Bintray      APT
sbt  aptitude Synaptic          System  Settings  ->
Software & Updates -> Other Software

```



Figure 1: Ubuntu Software & Updates Screenshot

Linux RPM

```

RPM  sbt

Linux  RPM  RPM          sbt          sudo

curl https://bintray.com/sbt/rpm/rpm > bintray-sbt-rpm.repo
sudo mv bintray-sbt-rpm.repo /etc/yum.repos.d/
sudo yum install sbt

sbt          Bintray Bintray      RPM
          sbt-launcher-package

```

Gentoo

```
sbt          ebuild          sbt ebuilds          ebuilds  sbt
mkdir -p /usr/local/portage && cd /usr/local/portage
git clone git://github.com/whiter4bbit/overlays.git
echo "PORTDIR_OVERLAY=$PORTDIR_OVERLAY /usr/local/portage/overlays" >> /etc/make.conf
emerge sbt-bin
          ebuild
```

Hello, World

```
sbt

sbt          hello          hw.scala
object Hi {
  def main(args: Array[String]) = println("Hi!")
}
hello      sbt   run      sbt      Linux   OS X

$ mkdir hello
$ cd hello
$ echo 'object Hi { def main(args: Array[String]) = println("Hi!") }' > hw.scala
$ sbt
...
> run
...
Hi!

sbt      sbt

•
• src/main/scala src/main/java
• src/test/scala src/test/java
• src/main/resources src/test/resources
• lib jar

sbt      Scala      sbt run      sbt console  Scala REPL sbt
console      classpath      Scala

build.sbt          hello  hello/build.sbt
```

```

lazy val root = (project in file("."))
  .settings(
    name := "hello",
    version := "1.0",
    scalaVersion := "2.12.1"
  )
.sbt
      build.sbt
      jar    build.sbt    name version

sbt

      hello/project/build.properties      sbt      0.13.15
sbt.version=0.13.15
sbt  release  99%      project/build.properties      sbt

      sbt      Hello, World

sbt  " "      Hello, World      hello      hello/build.sbt
hello/hw.scala hello

      hello/hw.scala      sbt Maven
src/
main/
  resources/
    <files to include in main jar here>
  scala/
    <main Scala sources>
  java/
    <main Java sources>
test/
  resources
    <files to include in test jar here>
  scala/
    <test Scala sources>
  java/

```

```

    <test Java sources>
src/

sbt

    build.sbt  sbt  project  project  .scala  .sbt

build.sbt
project/
  Build.scala
  project/  .sbt  .sbt

    classes  jars  caches  target

.gitignore
target/
  /  /  target/  project/target/

    sbt  sbt  Hello, World

sbt
$ sbt
sbt  tab
sbt  compile
> compile
  compile  run  exit  Ctrl+D  Unix  Ctrl+Z  Win-
dows

```



```

sbt          sbt          sbt
$ sbt clean compile "testOnly TestA TestB"
testOnly    TestA TestB      clean compile testOnly

```

```

- - sbt ~
> ~ compile
~

```

```

sbt
<td><tt>clean</tt></td>
<td>      <tt>target</tt>      </td>
<td><tt>compile</tt></td>
<td>      <tt>src/main/scala</tt>
src/main/java
<td><tt>test</tt></td>
<td>      </td>
<td><tt>console</tt></td>
<td>      classpath Scala      <tt>:quit</tt>
Ctrl+D Unix Ctrl+Z Windows sbt
<td><nobr><tt>run &lt; &gt;*</tt></nobr></td>
<td> sbt      main class </td>
<td><tt>package</tt></td>
<td> <tt>src/main/resources</tt>      <tt>src/main/scala</tt>      <tt>src/main/java</tt>      cla
<td><tt>help &lt; &gt;</tt></td>
<td>      </td>
<td><tt>reload</tt></td>
<td>      <tt>build.sbt</tt> <tt>project/*.scala</tt> <tt>project/*.sbt</tt>      )

```

Tab

```

tab sbt tab

```

```

sbt
<td><tt>!/td>
<td>
</td>
<td><tt>!!/td>
<td>
</td>
<td><tt>!:/td>
<td>
</td>
<td><tt>!:n/td>
<td>
<tt>n/td>
</td>
<td><tt>!n/td>
<td>
<tt>!:/td>
<tt>n/td>
</td>
<td><tt>!-n/td>
<td>
n
</td>
<td><tt>!string/td>
<td>
string
</td>
<td><tt>!?string/td>
<td>
string
</td>

```

.sbt

```

sbt " " build.sbt sbt

```

1. .sbt
2. bare .sbt
3. .scala

```

.sbt [bare .sbt ][Bare-Def] .scala

```

```

.scala project/

```

```

sbt Project
build.sbt Project

```

```

lazy val root = (project in file("."))
    immutable map
    name key
    sbt map
        Setting[T]      T      value      Setting      map
        value          map ——— map
        Setting[String]
lazy val root = (project in file("."))
    .settings(
        name := "hello"
    )
    Setting[String]      name      "hello"      map      map      sbt      map
    map sbt              key      value      key      key      sbt
Settings              map
    Project      Setting[T]      Setting[T]      sbt      map      T
value

    build.sbt

build.sbt      Project      settings scala

lazy val commonSettings = Seq(
    organization := "com.example",
    version := "0.1.0",
    scalaVersion := "2.12.1"
)

lazy val root = (project in file("."))
    .settings(
        commonSettings,
        name := "hello"
    )
    Setting      Scala      settings      Scala
    val lazy val def      build.sbt      object      class      project/
Scala
    name version scalaVersion      keys      key      SettingKey[T] TaskKey[T]
    InputKey[T]      T      value      key
    Keys      Setting[T]      :=      Java

```

```

lazy val root = (project in file("."))
  .settings(
    name.:=("hello")
  )

Scala  name := "hello"      Scala

key name      :=      Setting      Setting[String] String      name
SettingKey[String]      Setting[String]      sbt      map      name
  "hello"

      value

lazy val root = (project in file("."))
  .settings(
    name := 42 //
  )

```

Keys

Types

key

- SettingKey[T] key value
- TaskKey[T] key task value
- InputKey[T] key task Input Tasks

Keys

```

keys      Keys      build.sbt      import      sbt.Keys._      name
sbt.Keys.name

```

Keys

```

      settingKey taskKey inputKey      keys      key value      key
val      task hello      key

lazy val hello = taskKey[Unit](" task ")

.sbt      settings      vals defs      settings      vals
defs      settings

      lazy val      val

```

Task vs Setting keys

```

TaskKey[T]      task Tasks      compile      package      Unit Unit Scala
void      task      package      TaskKey[File]      task      jar
      task sbt      compile sbt      task

```

```

sbt map setting name task compile-
    key task setting "taskiness" ( key property value

```

tasks settings

```

:= setting task setting value task task

```

```

hello task

```

```

lazy val hello = taskKey[Unit]("An example task")

```

```

lazy val root = (project in file("."))
  .settings(
    hello := { println("Hello!") }
  )

```

```

settings

```

```

lazy val root = (project in file("."))
  .settings(
    name := "hello"
  )

```

Tasks Settings

```

task key Setting setting key Setting taskKey := 42
Setting[Task[T]] settingKey := 42 Setting[T] task key
T value
T Task[T] setting task setting

```

sbt Keys

```

sbt task name task compile compile task compile
task key
    setting key name task key name setting key value task
key name task value show <task name> <task name>
task key name camelCase name Scala
    key sbt inspect <keyname> inspect setting
value setting

```

build.sbt

```

import build.sbt

```

```

import sbt._
import Process._
import Keys._

    .scala      Build      Plugin      .scala

        jar      lib/      build.sbt
val derby = "org.apache.derby" % "derby" % "10.4.1.3"

lazy val commonSettings = Seq(
  organization := "com.example",
  version := "0.1.0",
  scalaVersion := "2.12.1"
)

lazy val root = (project in file("."))
  .settings(
    commonSettings,
    name := "hello",
    libraryDependencies += derby
  )

10.4.1.3 Apache Derby
key libraryDependencies += := % += key
%      Ivy      ID

```

Scope

```
scope      .sbt
```

Key

```

name      key      sbt      map
key              "scope"

```

- key
- key compile main test

- Key packageOptions jar class packageBin packageSrc

key *name* scope

scoped key

sbt map settings map key *scope* key set-
 ting build.sbt scope key
 scope build.sbt scope

Scope

Scope scope key

scope

- Projects
- Configurations
- Tasks

Project Scope

settings keys

Project setting setting setting

Configuration Scope

configuration classpath Configuration Ivy

MavenScopes

sbt configurations

- Compile src/main/scala
- Test src/test/scala
- Runtime task run classpath

key configuration configuration task

key compile package run key key sourceDirectories scalacOptions

fullClasspath configuration

Task Scope

Settings task task packageSrc setting packageOptions

task key packageSrc key packageOptions scope

task packageSrc packageBin packageDoc key artifactName

packageOptions key task

Scope

```
scope task task Global
Global setting task Global setting task

scope key key
scope sbt scope key scope sbt scope Global
scope scope
scope scope
inspect key " "
```

sbt scope key

```
sbt scope keys
{<build-uri><project-id>/config:intask::key
• {<build-uri><project-id> project project scope
  <project-id>
• config configuration
• intask task
• key scope key
“*” Global scope
scoped key
• project project
• configuration task key configuration
Configuration
```

scoped key

- fullClasspath key scope project key configuration task scope
- test:fullClasspath configuration fullClasspath test configuration scope scope
- *:fullClasspath configuration Global configuration
- doc::fullClasspath key fullClasspath doc task project configuration

- {file:/home/hp/checkout/hello/}default-aea33a/test:fullClasspath
project {file:/home/hp/checkout/hello/}default-aea33a
{file:/home/hp/checkout/hello/} project project id
default-aea33a configuration test task
- {file:/home/hp/checkout/hello/}/test:fullClasspath {file:/home/hp/checkout/hello/}
project
- {./}/test:fullClasspath {./} project {./} Scala
ThisBuild
- {file:/home/hp/checkout/hello/}/compile:doc::fullClasspath
scope

scope

```
sbt          inspect      key      scope  inspect test:fullClasspath

$ sbt
> inspect test:fullClasspath
[info] Task: scala.collection.Seq[sbt.Attributed[java.io.File]]
[info] Description:
[info] The exported classpath, consisting of build products and unmanaged and managed, internal
[info] Provided by:
[info] {file:/home/hp/checkout/hello/}default-aea33a/test:fullClasspath
[info] Dependencies:
[info]   test:exportedProducts
[info]   test:dependencyClasspath
[info] Reverse dependencies:
[info]   test:runMain
[info]   test:run
[info]   test:testLoader
[info]   test:console
[info] Delegates:
[info]   test:fullClasspath
[info]   runtime:fullClasspath
[info]   compile:fullClasspath
[info]   *:fullClasspath
[info]   {./}/test:fullClasspath
[info]   {./}/runtime:fullClasspath
[info]   {./}/compile:fullClasspath
[info]   {./}/*:fullClasspath
[info]   */test:fullClasspath
[info]   */runtime:fullClasspath
[info]   */compile:fullClasspath
[info]   */*:fullClasspath
[info] Related:
[info]   compile:fullClasspath
[info]   compile:fullClasspath(for doc)
```

```

[info] test:fullClasspath(for doc)
[info] runtime:fullClasspath
      task .sbt      setting      task      scala.collection.Seq[sbt.Attributed[java.io.File]]
“Provided by”      scoped key      {file:/home/hp/checkout/hello/}default-aea33a/test:fullClasspa
      test configuration      {file:/home/hp/checkout/hello/}default-aea33a
project
“Dependencies”

```

```

      sbt
      • configuration runtime:fullClasspath compile:fullClasspath
        scoped key project      “ project” task      Global
      • project      “ project” task      Global configuration
        Global *:fullClasspath
      • project project      {..} ThisBuild
      • project      Global */test:fullClasspath      project      current
        Global      * “ project” project      */test:fullClasspath
        test:fullClasspath
      • project configuration      Global */*:fullClasspath      task
        Global */*:fullClasspath      Global
      inspect fullClasspath      inspect test:fullClasspath      con-
figuration sbt      compile      inspect compile:fullClasspath
inspect fullClasspath
      inspect *:fullClasspath      fullClasspath      Global configuration

```

Configuration

scope

```

      build.sbt      bare key      project configuration task      Global
      lazy val root = (project in file("."))
      .settings(
        name := "hello"
      )
      sbt      inspect name      {file:/home/hp/checkout/hello/}default-aea33a/*:name
        project      {file:/home/hp/checkout/hello/}default-aea33a configu-
ration *      task
Keys      in      scope in      scope      name      Compile configuration

```

```

name in Compile := "hello"
      name      packageBin task

```

```

name in packageBin := "hello"

  name    scope    Compile configuration packageBin task
name in (Compile, packageBin) := "hello"

  Global
name in Global := "hello"

  name in Global    scope    Global    scope    Global task
configuration    Global    project    Global    */*:name
{file:/home/hp/checkout/hello/}default-aea33a/*:name

  Scala in := Scala Java
name.in(Compile).:=("hello")

```

scope

```

key          scope compile task    Compile Test configuration scope
scope

key compile    compile in Compile compile in Test compile
project scope task configuration scope compile task
  " "          scope          scope key          scope sbt
    " compile:compile "

  name key          key name scope scope packageOptions
in (Compile, packageBin) key name packageOptions key
name in key          scope project global config global task

```

```

:= .sbt scope

```

```

.sbt Setting Setting sbt map Setting
sbt map map map sbt

setting map .sbt :=

:= Setting map name := "hello" map map
key name "hello"

```

```

+= +=

:=      key      SettingKey[T]      T      key      se-
quence
• +=
• +=

key sourceDirectories in Compile Seq[File] key
src/main/scala source
sourceDirectories in Compile += new File("source")
sbt file()
sourceDirectories in Compile += file("source")
file() File
+=

sourceDirectories in Compile += Seq(file("sources1"), file("sources2"))
Seq(a, b, c, ...) Scala
source :=
sourceDirectories in Compile := Seq(file("sources1"), file("sources2"))

key

task setting value value := += +=
project organization
// name our organization after our project (both are SettingKey[String])
organization := name.value

// name is a Key[String], baseDirectory is a Key[File]
// name the project after the directory it's inside
name := baseDirectory.value.getName
java.io.File getName baseDirectory

name := "project " + name.value + " from " + organization.value + " version " + version.valu
name organization version name

```

```

    name := baseDirectory.value.getName    name    baseDirectory
build.sbt    sbt    inspect name
[info] Dependencies:
[info] *:baseDirectory

    sbt    setting    setting    setting    task    task
        inspect compile    key compileInputs    inspect compileInputs
        key    compile    sbt    update    compile    sbt
    update
    sbt    key    key

    := +=    +=    key    sbt    "    "    key
scope
sbt    sbt

    key    task
    task    setting    task    task    Def.task    taskValue    := +=
+=

        classpath    source generator
sourceGenerators in Compile += Def.task {
    myGenerator(baseDirectory.value, (managedClasspath in Compile).value)
}.taskValue

    task

.sbt    :=    task key    Setting[Task[T]]    Setting[T] Set-
ting    Task    Task    Setting
    key    Keys
val scalacOptions = taskKey[Seq[String]]("Options for the Scala compiler.")
val checksums = settingKey[Seq[String]]("The list of checksums to generate and to verify for
scalacOptions    checksums    key    task
    build.sbt    scalacOptions    checksums
// scalacOptions task    checksums setting
scalacOptions := checksums.value
    setting    key    task    key    setting    key    task
    task

```

```

// checksums setting scalacOptions task
checksums := scalacOptions.value

+= +=

setting task key :=
cleanFiles += file("coverage-report-" + name.value + ".txt")

.sbt Scopes

• lib jar
• repository

jar lib classpath
jar lib ScalaCheck Specs2 ScalaTest
lib classpaths compile test run console classpath
dependencyClasspath in Compile dependencyClasspath in
Runtime
build.sbt unmanagedBase key lib
custom_lib lib
unmanagedBase := baseDirectory.value / "custom_lib"
baseDirectory baseDirectory unmanagedBase
value
unmanagedBase jar task unmanagedJars
task unmanagedJars task Compile configuration lib
unmanagedJars in Compile := Seq.empty[sbt.Attributed[java.io.File]]

sbt Apache Ivy Ivy Maven

```

```

libraryDependencies Key
      libraryDependencies      Maven POM      Ivy      sbt

      groupId artifactId revision
libraryDependencies += groupId % artifactID % revision
      Configuration val configuration
libraryDependencies += groupId % artifactID % revision % configuration
libraryDependencies Keys
val libraryDependencies = settingKey[Seq[ModuleID]]("Declares managed dependencies.")
%      ModuleID      ModuleID      libraryDependencies
      sbt Ivy      sbt      Apache Derby      Maven2
libraryDependencies += "org.apache.derby" % "derby" % "10.4.1.3"
      build.sbt      update sbt Derby      ~/.ivy2/cache/org.apache.derby/
compile update      update
      +=
libraryDependencies += Seq(
  groupId % artifactID % revision,
  groupId % otherID % otherRevision
)

libraryDependencies :=

%%      Scala

      groupId %% artifactID % revision      groupId % artifactID %
revision      groupId      %% sbt      Scala      %%
libraryDependencies += "org.scala-tools" % "scala-stm_2.11.1" % "0.3"
      scalaVersion 2.11.1      "org.scala-tools"      %%
libraryDependencies += "org.scala-tools" %% "scala-stm" % "0.3"
      Scala      jar

      Scala      %%      2.10.1      scalaVersion
:= "2.10.4"      %%      2.10.1      %%      Scala

```

Ivy

```
groupID % artifactID % revision  revision      Ivy
"latest.integration" "2.9.+"    "[1.0,)"    "1.6.1"  Ivy
```

```
sbt      Maven2      resolver  Ivy
```

```
resolvers += name at location
            at
```

```
resolvers += "Sonatype OSS Snapshots" at "https://oss.sonatype.org/content/repositories/snapshots"
```

```
resolvers key  Keys
```

```
val resolvers = settingKey[Seq[Resolver]]("resolvers")
```

```
at      Resolver
```

```
sbt      Maven
```

```
resolvers += "Local Maven Repository" at "file://" + Path.userHome.absolutePath + "/.m2/repository"
```

```
resolvers += Resolver.mavenLocal
```

```
resolvers
```

```
sbt resolvers      externalResolvers
```

```
      externalResolvers  resolvers
```

Per-configuration dependencies

```
src/test/scala  Test configuration
```

```
Test configuration classpath  Compile configuration  % "test"
```

```
libraryDependencies += "org.apache.derby" % "derby" % "10.4.1.3" % "test"
```

```
Test configuration
```

```
libraryDependencies += "org.apache.derby" % "derby" % "10.4.1.3" % Test
```



```

sbt          show compile:dependencyClasspath    derby jar      show
test:dependencyClasspath    derby jar
      ScalaCheck Specs2  ScalaTest    % "test"

```

```

.sbt

```

```

      jar
Project lazy val
lazy val util = project

lazy val core = project
val      ID      ID      in
lazy val util = project.in(file("util"))

lazy val core = project in file("core")

```

To factor out common settings across multiple projects, create a sequence named `commonSettings` and call `settings` method on each project.

```

      commonSettings      settings

lazy val commonSettings = Seq(
  organization := "com.example",
  version := "0.1.0",
  scalaVersion := "2.12.1"
)

lazy val core = (project in file("core"))
  .settings(
    commonSettings,
    // other settings
  )

lazy val util = (project in file("util"))

```

```

.settings(
  commonSettings,
  // other settings
)

version

```

aggregate classpath

Aggregation

Aggregation aggregate task aggregated

```

lazy val root = (project in file(".")).aggregate(util, core)

```

```

lazy val util = project

```

```

lazy val core = project

```

```

    root    util    core          sbt

```

```

        root    task    update task

```

```

lazy val root = (project in file("."))
  .aggregate(util, core)
  .settings(
    aggregate in update := false
  )

```

[...]

```

aggregate in update  update task  scope    key    scopes
task task

```

Classpath

```

    dependsOn    core classpath    util    core

```

```

lazy val core = project.dependsOn(util)

```

```

    core    util          core    util

```

```

    dependsOn(bar, baz) dependsOn

```

configuration classpath

```
foo dependsOn(bar) foo compile configuration bar compile configuration
      dependsOn(bar % "compile->compile")

"compile->compile" -> "depends on" "test->compile" foo test
configuration bar compile configuration

->config ->compile dependsOn(bar % "test") foo test configuration
bar Compile configuration

"test->test" test test bar/src/test/scala
foo/src/test/scala

configuration dependsOn(bar % "test->test;compile->compile")
```

root

```
sbt

hello-foo base = file("foo") foo foo
foo/Foo.scala foo/src/main/scala sbt foo

foo .sbt foo/build.sbt hello-foo scope

hello hello/build.sbt hello/bar/build.sbt hello/foo/build.sbt
version := "0.6" sbt show version

> show version
[info] hello-foo/*:version
[info] 0.7
[info] hello-bar/*:version
[info] 0.9
[info] hello/*:version
[info] 0.5

hello-foo/*:version hello/foo/build.sbt hello-bar/*:version
hello/bar/build.sbt hello/*:version hello/build.sbt scoped
keys version key scope build.sbt build.sbt

.sbt .scala .scala
.scala

project/*.scala foo/project/Build.scala
```

```
sbt projects project <projectname> task
compile root

ID task subProjectID/compile
```

.sbt .sbt .sbt project/ Scala

build.sbt

task codeCoverage task

```
hello sbt-site hello/project/site.sbt Ivy ID
addSbtPlugin
addSbtPlugin("com.typesafe.sbt" % "sbt-site" % "0.7.0")
sbt-assembly hello/project/assembly.sbt
addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.11.2")

resolvers += Resolver.sonatypeRepo("public")
```

0.13.5 sbt

```
build.sbt
lazy val util = (project in file("util"))
.enablePlugins(FooPlugin, BarPlugin)
.settings(
  name := "hello-util"
)
enablePlugins
disablePlugins util IvyPlugin build.sbt
```

```

lazy val util = (project in file("util"))
  .enablePlugins(FooPlugin, BarPlugin)
  .disablePlugins(plugins.IvyPlugin)
  .settings(
    name := "hello-util"
  )

```

```

sbt      plugins

```

```

> plugins
In file:/home/jsuereth/projects/sbt/test-ivy-issues/
  sbt.plugins.IvyPlugin: enabled in scala-sbt-org
  sbt.plugins.JvmPlugin: enabled in scala-sbt-org
  sbt.plugins.CorePlugin: enabled in scala-sbt-org
  sbt.plugins.JUnitXmlReportPlugin: enabled in scala-sbt-org

plugins      sbt      sbt      3

1. CorePlugin:  task
2. IvyPlugin:
3. JvmPlugin:   Java/Scala

JUnitXmlReportPlugin  junit-xml

```

```

sbt-site      site.sbt
site.settings

```

```

// `util`      site
lazy val util = (project in file("util"))

// `core`      site
lazy val core = (project in file("core"))
  .settings(site.settings)

```

```

~/.sbt/0.13/plugins/  ~/.sbt/0.13/plugins/  classpath
sbt      ~/.sbt/0.13/plugins/  .sbt  .scala  project/

~/.sbt/0.13/plugins//build.sbt      addSbtPlugin()

```

- IDE sbt IDE
- web xsbt-web-plugin

sbt .sbt

```

SettingKey TaskKey .sbt InputKey
Keys
val scalaVersion = settingKey[String]("scala ")
val clean = taskKey[Unit]("source ")
    "scalaVersion" " scala "
.sbt T SettingKey[T] T TaskKey [T] .sbt
    " " batch
.sbt .scala autoImport val .sbt

:=

val sampleStringTask = taskKey[String]("A sample string task.")
val sampleIntTask = taskKey[Int]("A sample int task.")

lazy val commonSettings = Seq(
  organization := "com.example",
  version := "0.1.0-SNAPSHOT"
)

lazy val library = (project in file("library"))
  .settings(
    commonSettings,

```

```

sampleStringTask := System.getProperty("user.home"),
sampleIntTask := {
  val sum = 1 + 2
  println("sum: " + sum)
  sum
}
)

value
sbt      Scala      HTML      HTML
HTML
sbt      API  IO

```

```

value
sampeIntTask
sampleIntTask := {
  val sum = 1 + 2      // first
  println("sum: " + sum) // second
  sum                  // third
}

JVM  sum 3

startServer stopServer  sampeIntTask

val startServer = taskKey[Unit]("start server")
val stopServer = taskKey[Unit]("stop server")
val sampleIntTask = taskKey[Int]("A sample int task.")
val sampleStringTask = taskKey[String]("A sample string task.")

lazy val commonSettings = Seq(
  organization := "com.example",
  version := "0.1.0-SNAPSHOT"
)

lazy val library = (project in file("library"))
.settings(
  commonSettings,
  startServer := {
    println("starting...")
    Thread.sleep(500)
  },
  stopServer := {
    println("stopping...")
  }
)

```

```

    Thread.sleep(500)
  },
  sampleIntTask := {
    startServer.value
    val sum = 1 + 2
    println("sum: " + sum)
    stopServer.value // THIS WON'T WORK
    sum
  },
  sampleStringTask := {
    startServer.value
    val s = sampleIntTask.value.toString
    println("s: " + s)
    s
  }
}
)

sbt      sampleIntTask
> sampleIntTask
stopping...
starting...
sum: 3
[success] Total time: 1 s, completed Dec 22, 2014 5:00:00 PM
      sampleIntTask

```



Figure 2: task-dependency

Scala	value	sampleIntTask	startServer	stopServer	sampleIntTask	sbt
•	sampleIntTask					
•						
•						


```

sbt      sampleStringTask
> sampleStringTask
stopping...
starting...
sum: 3
s: 3
[success] Total time: 1 s, completed Dec 22, 2014 5:30:00 PM

sampleStringTask startServer sampleIntTask  sampleIntTask  startServer
Scala             value          sampeStringTask

```

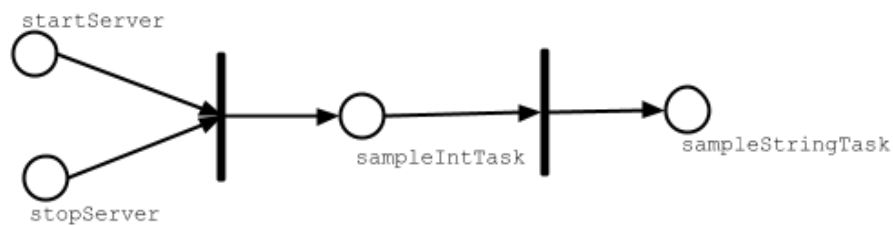


Figure 3: task-dependency

```

test      compile in Test test in Test

```

```

stopServer                                     stopServer sampleStringTask stopServer
sampleStringTask
lazy val library = (project in file("library"))
.settings(
  commonSettings,
  startServer := {
    println("starting...")
    Thread.sleep(500)
  },
  sampleIntTask := {
    startServer.value
    val sum = 1 + 2
    println("sum: " + sum)
    sum
  },
  sampleStringTask := {
    startServer.value
    val s = sampleIntTask.value.toString
    println("s: " + s)
    s
  }
)

```

```

    },
    sampleStringTask := {
      val old = sampleStringTask.value
      println("stopping...")
      Thread.sleep(500)
      old
    }
  )

  sampleStringTask

> sampleStringTask
starting...
sum: 3
s: 3
stopping...
[success] Total time: 1 s, completed Dec 22, 2014 6:00:00 PM

```



Figure 4: task-dependency

Scala

```

Scala    project/ServerUtil.scala

sampleIntTask := {
  ServerUtil.startServer
  try {
    val sum = 1 + 2
    println("sum: " + sum)
  } finally {
    ServerUtil.stopServer
  }
  sum
}

```

build.sbt

sbt

```
build.sbt      sbt      sbt      Scala      sbt
project
  sbt
  ,      project/project/
```

```
hello/      #
  Hello.scala      #      src/main/scala
  build.sbt      # build.sbt project/
  project/      #
    Build.scala      #
    build.sbt      #      --project/project
    project/      #
      Build.scala # project/project/
      project/project/
      .scala .sbt      build.sbt Build.scala
```

```
project .scala      project/Dependencies.scala
import sbt._

object Dependencies {
  // Versions
  lazy val akkaVersion = "2.3.8"
```

```

// Libraries
val akkaActor = "com.typesafe.akka" %% "akka-actor" % akkaVersion
val akkaCluster = "com.typesafe.akka" %% "akka-cluster" % akkaVersion
val specs2core = "org.specs2" %% "specs2-core" % "2.4.17"

// Projects
val backendDeps =
  Seq(akkaActor, specs2core % Test)
}

Dependencies build.sbt      val      Dependencies._
import Dependencies._

lazy val commonSettings = Seq(
  version := "0.1.0",
  scalaVersion := "2.12.1"
)

lazy val backend = (project in file("backend"))
  .settings(
    commonSettings,
    libraryDependencies ++= backendDeps
  )

.scala

.scala      Scala
           build.sbt      project/*.scala      .scala      scala

           project/*.scala

sbt          sbt      sbt

```

sbt:

- Scala Scala Programming in Scala Scala
- .sbt
- Setting sbt Setting task
- Setting key := += ++=
- Setting sbt
- key
- tasks key value task Non-task
- Scopes
- key value scope
- scope configuration project task
- scope task configuration
- configuration Compile Test
- project “ ” scope
- scopes scope
- build.sbt .scala task
- sbt
-
- addSbtPlugin project/plugins.sbt build.sbt
- sbt

sbt

Bare .sbt

.sbt .sbt

bare .sbt

```
.sbt .scala bare .sbt
bare .sbt Setting[_] Project
name := "hello"

version := "1.0"

scalaVersion := "2.12.1"
```

```

( 0.13.7 )

0.13.7
bare build.sbt
//
name := "hello"
version := "1.0"
scalaVersion := "2.10.3"
sbt

.scala

.scala          sbt .scala          sbt 0.13      .sbt
               .sbt

build.sbt  Build.scala

.sbt .scala
           hello      hello/project/Build.scala
import sbt._
import Keys._

object HelloBuild extends Build {
  val sampleKeyA = settingKey[String]("demo key A")
  val sampleKeyB = settingKey[String]("demo key B")
  val sampleKeyC = settingKey[String]("demo key C")
  val sampleKeyD = settingKey[String]("demo key D")

  override lazy val settings = super.settings ++
    Seq(
      sampleKeyA := "A: in Build.settings in Build.scala",
      resolvers := Seq()
    )

  lazy val root = Project(id = "hello",
    base = file("."),
    settings = Seq(
      sampleKeyB := "B: in the root project settings in Build.scala"
    ))
}

hello/build.sbt

```

```

sampleKeyC in ThisBuild := "C: in build.sbt scoped to ThisBuild"

sampleKeyD := "D: in build.sbt"

sbt      inspect sampleKeyA
[info] Setting: java.lang.String = A: in Build.settings in Build.scala
[info] Provided by:
[info] {file:/home/hp/checkout/hello/}/*:sampleKeyA

      inspect sampleKeyC
[info] Setting: java.lang.String = C: in build.sbt scoped to ThisBuild
[info] Provided by:
[info] {file:/home/hp/checkout/hello/}/*:sampleKeyC

      "Provided by"      value      .sbt      sampleKeyC in ThisBuild
.scala Build.settings      sbt

      inspect sampleKeyB
[info] Setting: java.lang.String = B: in the root project settings in Build.scala
[info] Provided by:
[info] {file:/home/hp/checkout/hello/}hello/*:sampleKeyB

      sampleKeyB      ({file:/home/hp/checkout/hello/}hello)
({file:/home/hp/checkout/hello/})

      inspect sampleKeyD      sampleKeyB
[info] Setting: java.lang.String = D: in build.sbt
[info] Provided by:
[info] {file:/home/hp/checkout/hello/}hello/*:sampleKeyD

sbt .sbt      Build.settings      Project.setting      .sbt
Build.scala      sampleC      sampleD      build.sbt      build.sbt      " "
Build.sbt

      sampleKeyC sampleKeyD      build.sbt      sbt      Build      .sbt
import HelloBuild._      build.sbt

• .scala      Build.settings
• .scala      Project.settings
• .scala      Build      .sbt
• .sbt      .scala
• .sbt

```

```

sbt      project/      reload plugins

```

```

> reload plugins
[info] Set current project to default-a0e8e4 (in build file:/home/hp/checkout/hello/project/)
> show sources
[info] ArrayBuffer(/home/hp/checkout/hello/project/Build.scala)
> reload return
[info] Loading project definition from /home/hp/checkout/hello/project
[info] Set current project to hello (in build file:/home/hp/checkout/hello/)
> show sources
[info] ArrayBuffer(/home/hp/checkout/hello/hw.scala)
>

```

```

    reload return

```

```

build.sbt      Build Project settings      Build Project
settings      build.sbt          sbt        Build Project
“      ”

```

- .scala Build.settings Project.settings
- ~/.sbt/0.13/global.sbt
-
- .sbt
- project ~/.sbt/0.13/plugins/