


# Java-Success.com

Prepare to fast-track, choose & go places with 800+ Java & Big Data Q&As with lots of code & diagrams.

[Home](#) [Why? ▾](#) [300+ Java FAQs ▾](#) [300+ Big Data FAQs ▾](#) [Courses ▾](#)[👤 Membership ▾](#) [Your Career ▾](#)[Home](#) › [bigdata-success.com](#) › [Tutorials - Big Data](#) › [TUT - Starting Spark & Scala](#) ›

04. Setting up &amp; getting started with sbt

## 04. Setting up & getting started with sbt

 Posted on [October 20, 2018](#)

SBT (i.e. **Simple Build Tool**) for Scala requires **Java & Scala**. This tutorial is based on Java 8 (i.e. 1.8.0) and Scala 2.12.7. Scala runs on the JVM, so Java and Scala stacks can be freely mixed. You can call Java libraries from Scala.

### Install sbt

**Step 1:** Install sbt on Windows/Mac.

To install sbt on Windows, download and run the Windows MSI from <http://www.scala->

### 300+ Java Interview FAQs

300+ Java FAQs



16+ Java Key Areas Q&amp;As



150+ Java Architect FAQs



80+ Java Code Quality Q&amp;As



150+ Java Coding Q&amp;As



### 300+ Big Data Interview FAQs

300+ Big Data FAQs



Tutorials - Big Data

TUT -  Starting Big Data

TUT - Starting Spark &amp; Scala

 TOP

[sbt.org/download.html](https://sbt.org/download.html).

To install sbt on Mac

```
1  
2 brew install sbt  
3
```

## Creating a project & sbt commands on command-line

**Step 2:** Create a new folder “sbt-tutorial” inside a folder say “projects”. Change directory “~/projects/sbt-tutorial”.

```
1  
2 $ cd ~/projects/sbt-tutorial  
3 ~/projects/sbt-tutorial]$ sbt  
4 sbt:sbt-tutorial>  
5
```

**Step 3:** On the sbt command-line you can use “show” command to display Scala and the project versions.

```
1 sbt:sbt-tutorial> show scalaVersion  
2 [info] 2.12.7  
3
```

```
1 sbt:sbt-tutorial> show version  
2 [info] 0.1.0-SNAPSHOT  
3
```

**Step 4:** You can modify the version with the “set” command.

Any changes need to be saved with “**session save**”. The values that are modified with the “**set**” commands will be saved to the “**build.sbt**” file.

TUT - Starting with Python

TUT - Kafka

TUT - Pig

TUT - Apache Storm

TUT - Spark Scala on Zeppelin

TUT - Cloudera

TUT - Cloudera on Docker

TUT - File Formats

TUT - Spark on Docker

TUT - Flume

TUT - Hadoop (HDFS)

TUT - HBase (NoSQL)

TUT - Hive (SQL)

TUT - Hadoop & Spark

TUT - MapReduce

TUT - Spark and Scala

TUT - Spark & Java

TUT - PySpark on Databricks

TUT - Zookeeper

## 800+ Java Interview Q&As

300+ Core Java Q&As



300+ Enterprise Java Q&As



150+ Java Frameworks Q&As



120+ Companion Tech Q&As



Tutorials - Enterprise Java



```
1 sbt:sbt-tutorial> session save
2
1 sbt:sbt-tutorial> set version := "0.1"
2
```

**Step 5:** You can open a Scala REPL within sbt with the “**console**” command.

```
1 sbt:sbt-tutorial> console
2 [info] Starting scala interpreter...
3 Welcome to Scala 2.12.7 (Java HotSpot(TM) 64-Bit S
4 Type in expressions for evaluation. Or try :help.
5
6 scala>
7
```

You can exit Scala REPL with “:q” or “sys.exit” commands.

**Step 6:** You can exit the sbt with the “exit” command. You should now have the directories project, target & the **build.sbt** file.

## Creating the project artefacts in Scala

**Step 7:** In “/project/sbt-tutorial”, create a new folder “src/main/scala”.

```
1 ~/projects/sbt-tutorial]$ mkdir -p "src/main/scala"
2
```

**Step 8:** Create a package inside “src/main/scala” named “com/sbt-hello”.

```
1 ~/projects/sbt-tutorial/src/main/scala]$ mkdir -p
2
```



**Step 9:** Create a new Scala file in the “com.sbthello” package.

```
1 ~/projects/sbt-tutorial/src/main/scala/com/sbt-hello  
2
```

**Step 10:** Open a text editor or an IDE like “Visual Studio Code”, and type the following basic Scala code.

```
1 package com.sbthello  
2  
3 object HelloSbt {  
4     def main(args: Array[String]) = {  
5         print("Hello Sbt")  
6     }  
7  
8 }
```

**Step 11:** Type “sbt” from a command-line from the project folder (i.e. ~/projects/sbt-tutorial).

```
1 ~/project/sbt-tutorial]$ sbt
```

**Step 12:** Type “compile” to compile the code.

```
1 sbt:sbt-tutorial> compile  
2 [success] Total time: 1 s, completed 20/10/2018 1:50:00  
3
```

**Step 13:** Type “package” to package the code as a jar file

```
1 sbt:sbt-tutorial> package  
2 [success] Total time: 0 s, completed 20/10/2018 1:50:00  
3
```



You can see the built jar file “**sbt-tutorial\_2.12-0.1.jar**” in the “~/projects/sbt-tutorial/target/scala/2.12” folder.

## Making it ready for Eclipse import

**Step 14:** Create the “**plugins.sbt**” file inside the “~/projects/sbt-tutorial/project”, add the following line.

```
1 addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse" % "0.4.0")
2
```

**Step 15:** From the command-line type “**sbt**”, and then when inside sbt type “**eclipse**” to generate the eclipse metadata files like like .project & .classpath.

```
1 ~/project/sbt-tutorial]$ sbt
```

```
1 sbt:sbt-tutorial> eclipse
```

**Step 16:** Open Eclipse with Scala IDE installed as per the previous tutorial. Within Eclipse import “sbt-tutorial” with “**File -> Import -> General -> Existing Projects into Workspace**”.

**Step 17:** Within Eclipse select the top level of the Scala project, right-click and select “**properties**” and select “**Scala compiler**”. Make sure that the “**User project settings**” is ticked.

**Step 18:** Now select the folder “sbthello” right-mouse-click and select “**Build Path**” and click on “**Include**”. Do the same for “**com**” as well. Now Eclipse recognises “com.sbthello” as a package. The symbol



of both the package and the file “HelloSbt.scala” change.

**Step 19:** Right mouse click on “HelloSbt.scala” and select “Run As” and then “Scala Application”.

## Outputs:

1	Hello Sbt

◀ 9+ CI/CD Docker compose DevOps interview Q&As

Docker DevOps Q&As with Big Data code snippets ▶

## Disclaimer

The contents in this Java-Success are copyrighted and from EmpoweringTech pty ltd. The EmpoweringTech pty ltd has the right to correct or enhance the current content without any prior notice. These are general advice only, and one needs to take his/her own circumstances into consideration. The EmpoweringTech pty ltd will not be held liable for any damages caused or alleged to be caused either directly or indirectly by these materials and resources. Any trademarked names or labels used in this blog remain the property of their respective trademark owners. Links to external sites do not imply endorsement of the linked-to sites. [Privacy Policy](#)

