Scala/Spark

Wikipedia :

Apache Spark is an open-source cluster computing framework originally developed in the AMPLab at UC Berkeley. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications.[1] By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms.[2]
Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos.[3] For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS),[4] Cassandra,[5] OpenStack Swift, and Amazon S3. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in this scenario, Spark is running on a single machine with one executor per CPU core.

Project Components

1. Spark Core and Resilient Distributed Datasets (RDDs)
2. Spark SQL
3. Spark Streaming
4. MLlib Machine Learning Library
5. GraphX

Features

- Java, Scala, Python, and R APIs.
- Scalability to over 8000 nodes in production.[11]
- Ability to cache datasets in memory for interactive data analysis: extract a working set, cache it, query it repeatedly.
- Interactive command line interface (in Scala or Python) for low-latency horizontally scalable data exploration.
- Higher level library for stream processing[12], through Spark Streaming.
- Support for structured and relational query processing (SQL), through Spark SQL.
- Higher level libraries for machine learning and graph processing.

Examples

- **Declare a list of integers as a variable called “myNumbers”:**

```
scala> val myNumbers = List(1, 2, 5, 4, 7, 3)
myNumbers: List[Int] = List(1, 2, 5, 4, 7, 3)
```

- **Declare a function, `cube`, that computes the cube (third power) of an Int.**

```
scala> def cube(a: Int): Int = a * a * a
cube: (a: Int)Int
```

- **Apply the function to `myNumbers` using the `map` function.**

```
scala> myNumbers.map(x => cube(x))
       res: List[Int] = List(1, 8, 125, 64, 343, 27)
       // Scala also provides some shorthand ways of writing this:
       // myNumbers.map(cube(_))
       // myNumbers.map(cube)
```

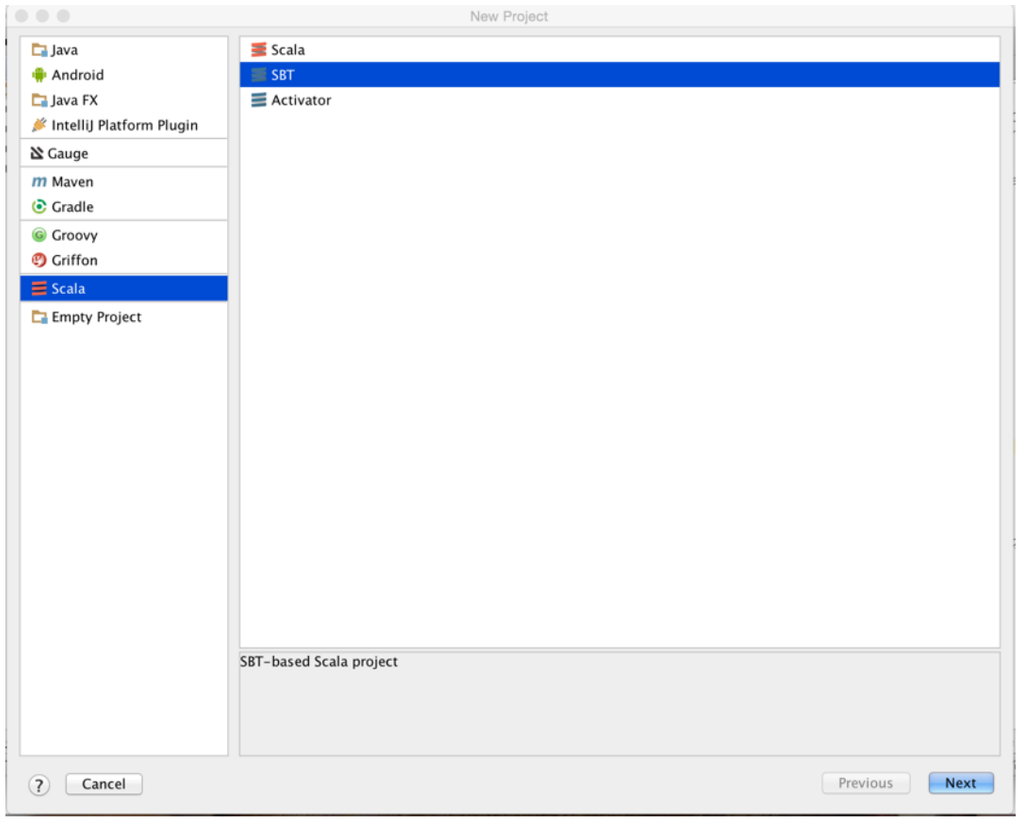
- **Then also try writing the function inline in a `map` call, using closure notation.**

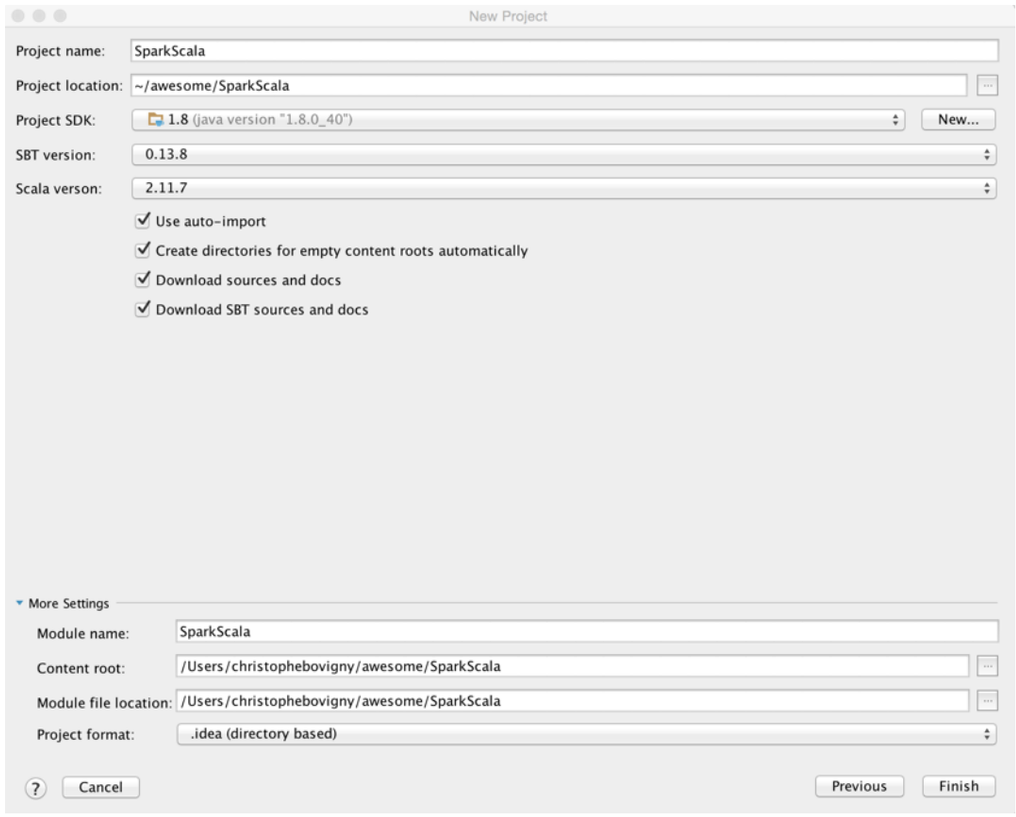
```
scala> myNumbers.map{x => x * x * x}
res: List[Int] = List(1, 8, 125, 64, 343, 27)
```

- **Define a `factorial` function that computes n! = 1 * 2 * … * n given input n. You can use either a loop or recursion, in our solution we use recursion (see steps 5-7 of First Steps to Scala). Then compute the sum of factorials in `myNumbers`.**

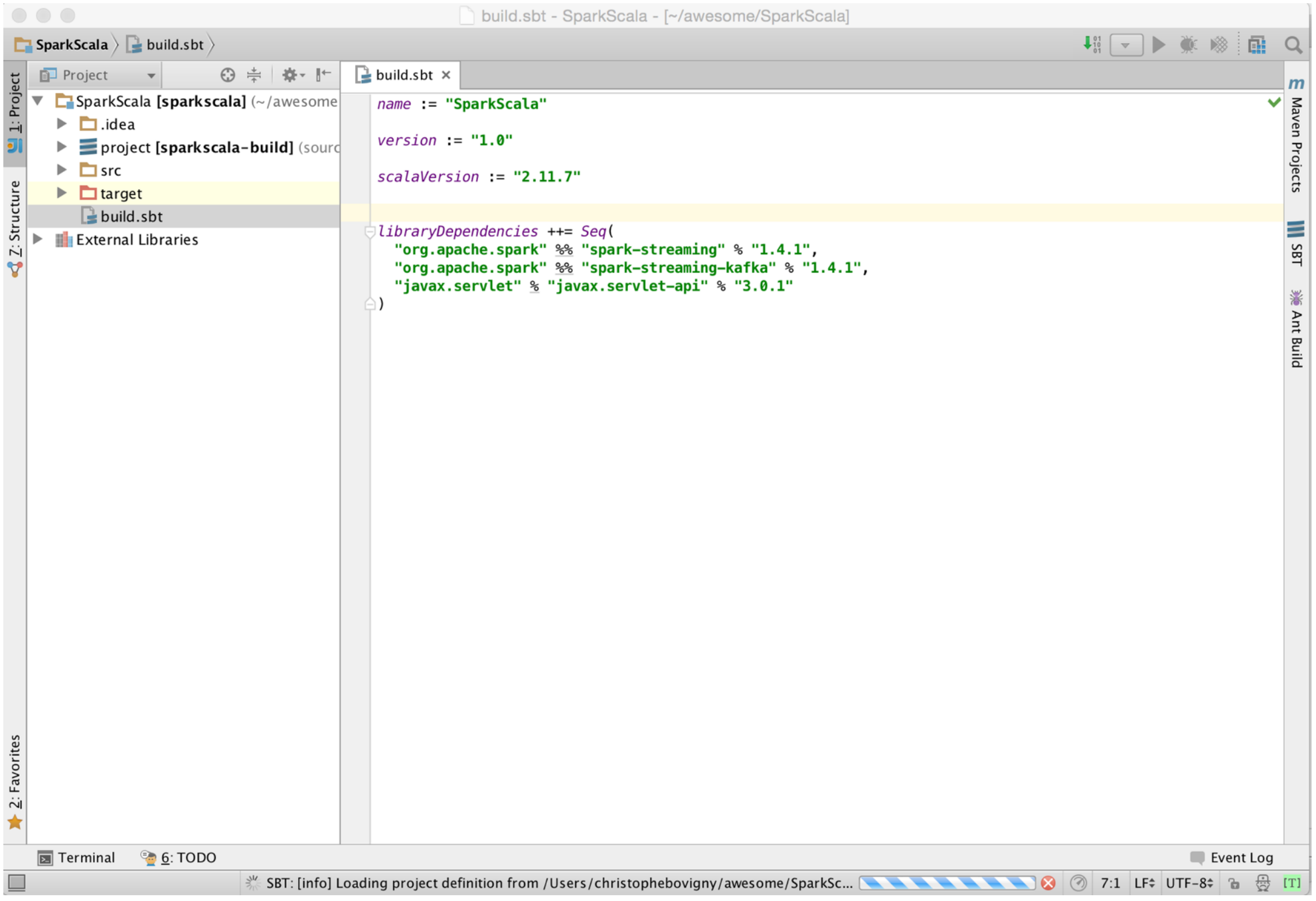
```
scala> def factorial(n:Int):Int = if (n==0) 1 else n * factorial(n-1) // From http://bit.ly/b2sVKI
factorial: (Int)Int
scala> myNumbers.map(factorial).sum
res: Int = 5193
```

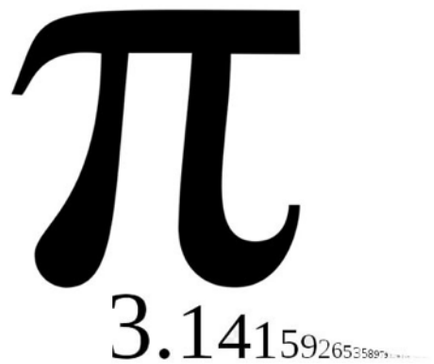
Setup a new Project



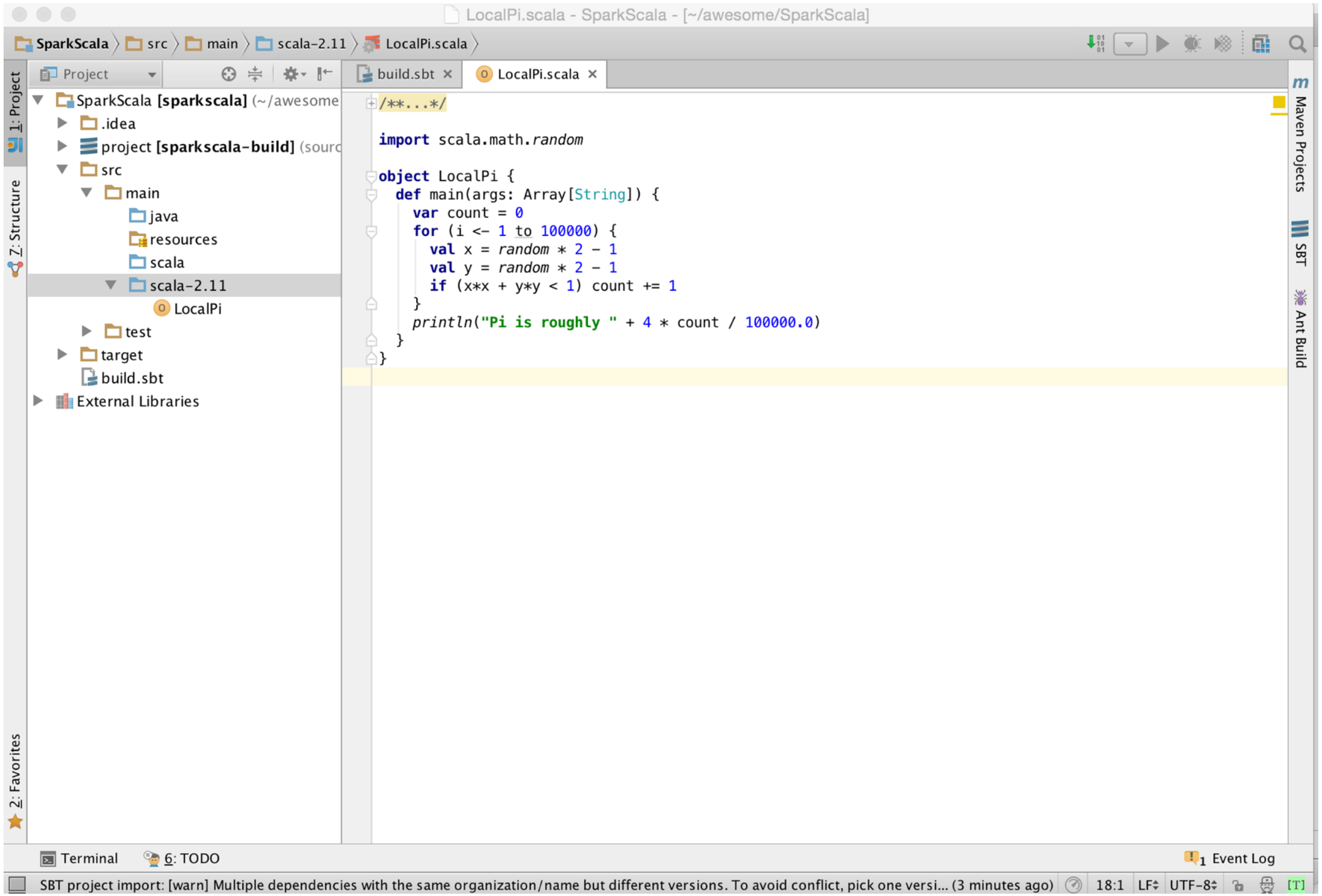


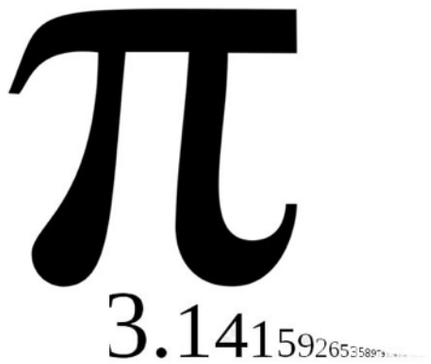
Add dependencies



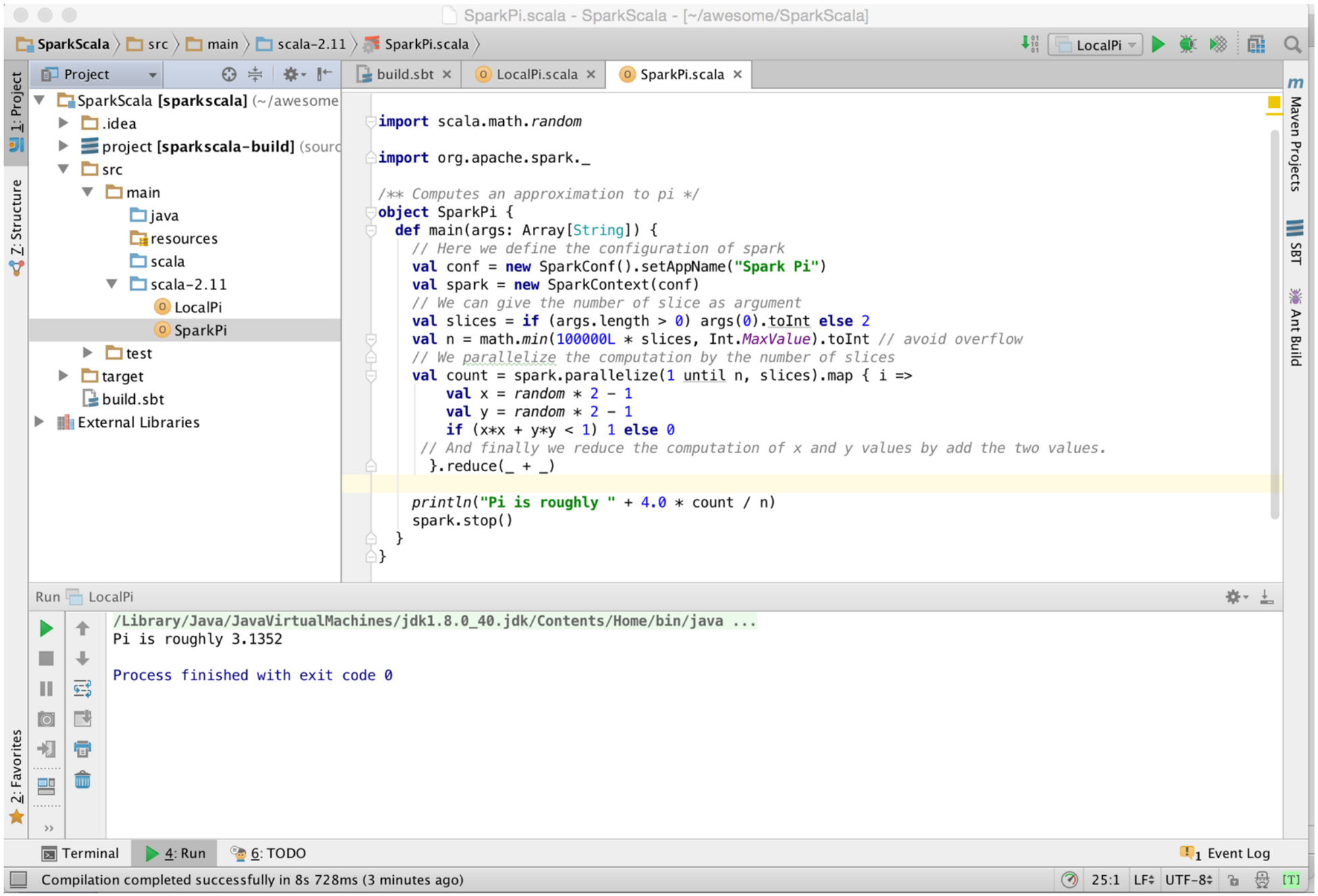

3.14159265

Pi / Local

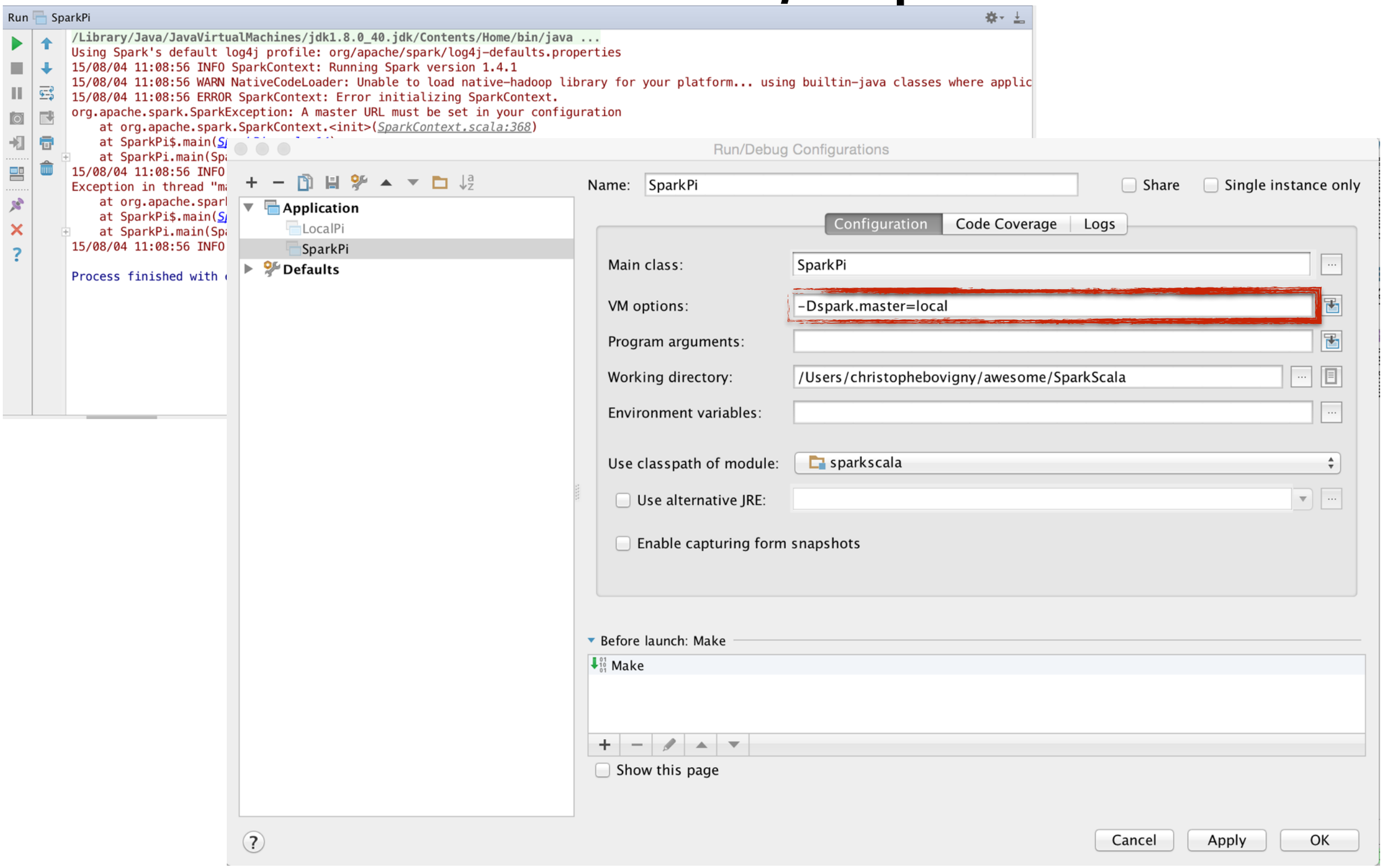

LocalPi.scala - SparkScala - [~/awesome/SparkScala]
SparkScala
src
main
scala-2.11
LocalPi.scala
Project
build.sbt
LocalPi.scala
SparkScala [sparkscala] (~/awesome
.idea
project [sparkscala-build] (sourc
src
main
java
resources
scala
scala-2.11
LocalPi
test
target
build.sbt
External Libraries
/**...*/
import scala.math.random
object LocalPi {
def main(args: Array[String]) {
var count = 0
for (i <- 1 to 100000) {
val x = random * 2 - 1
val y = random * 2 - 1
if (x*x + y*y < 1) count += 1
}
println("Pi is roughly " + 4 * count / 100000.0)
}
}
1: Project
7: Structure
2: Favorites
Maven Projects
SBT
Ant Build
Terminal
6: TODO
1 Event Log
SBT project import: [warn] Multiple dependencies with the same organization/name but different versions. To avoid conflict, pick one versi... (3 minutes ago)
18:1
LF
UTF-8
[T]


3.14159265

Pi / Spark


SparkPi.scala - SparkScala - [~/awesome/SparkScala]
SparkScala
src
main
scala-2.11
SparkPi.scala
LocalPi
Project
build.sbt
LocalPi.scala
SparkPi.scala
SparkScala [sparkscala] (~/awesome
.idea
project [sparkscala-build] (sourc
src
main
java
resources
scala
scala-2.11
LocalPi
SparkPi
test
target
build.sbt
External Libraries
import scala.math.random
import org.apache.spark._
/** Computes an approximation to pi */
object SparkPi {
def main(args: Array[String]) {
// Here we define the configuration of spark
val conf = new SparkConf().setAppName("Spark Pi")
val spark = new SparkContext(conf)
// We can give the number of slice as argument
val slices = if (args.length > 0) args(0).toInt else 2
val n = math.min(100000L * slices, Int.MaxValue).toInt // avoid overflow
// We parallelize the computation by the number of slices
val count = spark.parallelize(1 until n, slices).map { i =>
val x = random * 2 - 1
val y = random * 2 - 1
if (x*x + y*y < 1) 1 else 0
// And finally we reduce the computation of x and y values by add the two values.
}.reduce(_ + _)
println("Pi is roughly " + 4.0 * count / n)
spark.stop()
}
}
Run LocalPi
/Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home/bin/java ...
Pi is roughly 3.1352
Process finished with exit code 0
Maven Projects
SBT
Ant Build
1: Project
Z: Structure
2: Favorites
Terminal
4: Run
6: TODO
Event Log
Compilation completed successfully in 8s 728ms (3 minutes ago)
25:1
LF
UTF-8

Run Pi /Spark


Run SparkPi
/Library/Java/JavaVirtualMachines/jdk1.8.0_40.jdk/Contents/Home/bin/java ...
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
15/08/04 11:08:56 INFO SparkContext: Running Spark version 1.4.1
15/08/04 11:08:56 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applic
15/08/04 11:08:56 ERROR SparkContext: Error initializing SparkContext.
org.apache.spark.SparkException: A master URL must be set in your configuration
at org.apache.spark.SparkContext.<init>(SparkContext.scala:368)
Process finished with
Run/Debug Configurations
Application
LocalPi
SparkPi
Defaults
Name:
SparkPi
Share
Single instance only
Configuration
Code Coverage
Logs
Main class:
SparkPi
VM options:
-Dspark.master=local
Program arguments:
Working directory:
/Users/christophebovigny/awesome/SparkScala
Environment variables:
Use classpath of module:
sparkscala
Use alternative JRE:
Enable capturing form snapshots
Before launch: Make
Make
Show this page
Cancel
Apply
OK