CREACIÓN Y PRUEBA DE PROYECTOS SCALA CON SBT

En este laboratorio verás dos herramientas que se usan comúnmente en proyectos de Scala:

- La herramienta de construcción sbt
- ScalaTest , un marco de prueba de código fuente

Comenzaremos mostrando cómo usar sbt para construir sus proyectos de Scala y luego mostraremos cómo usar sbt y ScalaTest juntos para probar sus proyectos de Scala.

Construyendo proyectos Scala con sbt

Puede utilizar varias herramientas diferentes para crear sus proyectos de Scala, incluidas Ant, Maven, Gradle, Mill y más. Pero una herramienta llamada sbt fue la primera herramienta de compilación creada específicamente para Scala.

Asegúrate de tener Java 8 JDK (también conocido como 1.8)

Ejecute javac -versionen la línea de comando y asegúrese de ver javac 1.8.___

Si no tienes la versión 1.8 o superior, instala el JDK

Instalar sbt y VSCode

Creando un proyecto "Hola, mundo"

Puedes crear un proyecto sbt “Hola, mundo” en solo unos pocos pasos. Primero, cree un directorio para trabajar y vaya a ese directorio:

$ mkdir hello

$ cd hello

En el directorio hello, cree un subdirectorio project:

$ mkdir project

Cree un archivo llamado build.properties en el directorio project, con el siguiente contenido:

sbt.version=1.6.1

Luego cree un archivo llamado build.sbt en el directorio raíz del proyecto que contiene esta línea:

scalaVersion := "3.4.0"

Ahora cree un archivo llamado algo así como Hello.scala (la primera parte del nombre no importa) con esta línea:

@main def helloWorld = println("Hello, world")

Eso es todo lo que tienes que hacer.

Debería tener una estructura de proyecto como la siguiente:

$ tree

.

├── build.sbt

├── Hello.scala

└── project

  └── build.properties

Ahora ejecute el proyecto con este sbtcomando:

$ sbt run

Debería ver un resultado similar a este, incluido el "Hello, world" de su programa:

$ sbt run

[info] welcome to sbt 1.5.4 (AdoptOpenJDK Java 11.x)

[info] loading project definition from project ...

[info] loading settings for project from build.sbt ...

[info] compiling 1 Scala source to target/scala-3.0.0/classes ...

[info] running helloWorld

Hello, world

[success] Total time: 2 s

El iniciador de sbt, la sbtherramienta de línea de comandos, carga la versión de sbt configurada en el archivo project/build.properties , que carga la versión del compilador de Scala configurada en el archivo build.sbt y compila el código en el archivo Hello.scala. y ejecuta el código de bytes resultante.

Cuando mires tu directorio, verás que sbt tiene un directorio llamado target . Estos son directorios de trabajo que utiliza sbt.

Como puede ver, crear y ejecutar un pequeño proyecto Scala con sbt requiere sólo unos sencillos pasos.

Usando sbt con proyectos más grandes

Para un proyecto pequeño, eso es todo lo que sbt necesita para ejecutarse. Para proyectos más grandes que requieren muchos archivos de código fuente, dependencias o complementos sbt, querrás crear una estructura de directorio organizada. El resto de esta sección demuestra la estructura que utiliza sbt.

La estructura del directorio sbt

Al igual que Maven, sbt utiliza una estructura de directorio de proyecto estándar. Un buen beneficio de esto es que una vez que se sienta cómodo con su estructura, le resultará más fácil trabajar en otros proyectos de Scala/sbt.

Lo primero que debe saber es que debajo del directorio raíz de su proyecto, sbt espera una estructura de directorios similar a esta:

```
.
├── build.sbt
├── project/
│   └── build.properties
├── src/
│   ├── main/
│   │   ├── java/
│   │   ├── resources/
│   │   └── scala/
│   └── test/
│       ├── java/
│       ├── resources/
│       └── scala/
└── target/
```

También puede agregar un directorio lib en el directorio raíz si desea agregar dependencias no administradas (archivos JAR) a su proyecto.

Si va a crear un proyecto que tiene pruebas y archivos de código fuente de Scala, pero no utilizará ningún archivo de código fuente de Java y no necesita ningún "recurso", como imágenes incrustadas, archivos de configuración, etc. —Esto es todo lo que realmente necesitas en el directorio src :

.

└── src/

  ├── main/

  │ └── scala/

  └── test/

    └── scala/

“Hola mundo” con una estructura de directorio sbt

Crear esta estructura de directorios es simple. Existen herramientas para hacer esto por usted, pero suponiendo que esté usando un sistema Unix/Linux, puede usar estos comandos para crear la estructura de directorios de su primer proyecto sbt:

$ mkdir HelloWorld

$ cd HelloWorld

$ mkdir -p src/{main,test}/scala

$ mkdir project target

Cuando ejecuta un find .comando después de ejecutar esos comandos, debería ver este resultado:

$ find .

.

./project

./src

./src/main

./src/main/scala

./src/test

./src/test/scala

./target

Si ves eso, estás en excelente forma para el siguiente paso.

Hay otras formas de crear archivos y directorios para un proyecto sbt. Una forma es utilizar el sbt new comando, que está documentado aquí en scala-sbt.org . Ese enfoque no se muestra aquí porque algunos de los archivos que crea son más complicados de lo necesario para una introducción como esta.

Creando un primer archivo build.sbt

En este punto, sólo necesitas dos cosas más para ejecutar un proyecto "Hola, mundo":

- Un archivo build.sbt
- Un archivo Hola.scala

Para un proyecto pequeño como este, el archivo build.sbt solo necesita una scalaVersion entrada, pero agregaremos tres líneas que se ven comúnmente:

```
name := "HelloWorld"

version := "0.1"

scalaVersion := "3.4.0"
```

Debido a que los proyectos de sbt utilizan una estructura de directorio estándar, sbt puede encontrar todo lo que necesita.

Ahora sólo necesitas agregar un pequeño programa de “Hola, mundo”.

Un programa “Hola, mundo”

En proyectos grandes, todos sus archivos de código fuente de Scala estarán en los directorios src/main/scala y src/test/scala , pero para un pequeño proyecto de muestra como este, puede colocar su archivo de código fuente en el directorio raíz de su proyecto. Por lo tanto, cree un archivo llamado HelloWorld.scala en el directorio raíz con este contenido:

```
@main def helloWorld = println("Hello, world")
```

Ese código define un método "principal" de Scala 3 que imprime "Hello, world"cuando se ejecuta.

Ahora, usa el sbt run comando para compilar y ejecutar tu proyecto:

```
$ sbt run

[info] welcome to sbt

[info] loading settings for project ...

[info] loading project definition

[info] loading settings for project root from build.sbt ...

[info] Compiling 1 Scala source ...

[info] running helloWorld

Hello, world

[success] Total time: 4 s
```

La primera vez que sbt lo ejecuta, descarga todo lo que necesita y puede tardar unos minutos en ejecutarse, pero después se vuelve mucho más rápido.

Además, una vez que este primer paso funcione, descubrirá que es mucho más rápido ejecutar sbt de forma interactiva. Para hacer eso, primero ejecute el sbtcomando por sí solo:

```
$ sbt

[info] welcome to sbt

[info] loading settings for project ...

[info] loading project definition ...

[info] loading settings for project root from build.sbt ...

[info] sbt server started at

    local:///${HOME}/.sbt/1.0/server/7d26bae822c36a31071c/sock

sbt:hello-world> _
```

Luego, dentro de este shell sbt, ejecuta su runcomando:

```
sbt:hello-world> run

[info] running helloWorld

Hello, world

[success] Total time: 0 s
```

Ahí es mucho más rápido.

Usar plantillas de proyecto

Crear manualmente la estructura del proyecto puede resultar tedioso. Afortunadamente, sbt puede crearlo por ti, basándose en una plantilla.

Para crear un proyecto de Scala 3 a partir de una plantilla, ejecute el siguiente comando en un shell:

```
$ sbt new scala/scala3.g8
```

Sbt cargará la plantilla, hará algunas preguntas y creará los archivos del proyecto en un subdirectorio:

```
$ tree scala-3-project-template

scala-3-project-template

├── build.sbt

├── project

│   └── build.properties
```

```
├── README.md
└── src
    ├── main
    │   └── scala
    │       └── Main.scala
    └── test
        └── scala
            └── Test1.scala
```

Usando sbt con ScalaTest

ScalaTest es una de las principales bibliotecas de prueba para proyectos Scala. En esta sección verá los pasos necesarios para crear un proyecto Scala/sbt que utilice ScalaTest.

1) Crear la estructura del directorio del proyecto.

Al igual que en la lección anterior, cree una estructura de directorio de proyecto sbt para un proyecto llamado HelloScalaTest con los siguientes comandos:

$ mkdir HelloScalaTest

$ cd HelloScalaTest

$ mkdir -p src/{main,test}/scala

$ mkdir project

2) Cree los archivos build.properties y build.sbt

A continuación, cree un archivo build.properties en el subdirectorio proyecto/ de su proyecto con esta línea:

sbt.version=1.5.4

A continuación, cree un archivo build.sbt en el directorio raíz de su proyecto con este contenido:

name := "HelloScalaTest"

version := "0.1"

scalaVersion := "3.4.0"

```
libraryDependencies ++= Seq(
  "org.scalatest" %% "scalatest" % "3.2.9" % Test
)
```

Las primeras tres líneas de este archivo son esencialmente las mismas que las del primer ejemplo. Las libraryDependencieslíneas le indican a sbt que incluya las dependencias (archivos JAR) necesarias para incluir ScalaTest.

La documentación de ScalaTest siempre ha sido buena y siempre puede encontrar información actualizada sobre cómo deberían verse esas líneas en la página Instalación de ScalaTest .

3) Cree un archivo de código fuente de Scala

A continuación, cree un programa Scala que pueda utilizar para demostrar ScalaTest. Primero, cree un directorio en src/main/scala llamado math :

```
$ mkdir src/main/scala/math
```

Luego, dentro de ese directorio, cree un archivo llamado MathUtils.scala con este contenido:

```
package math

object MathUtils:
  def double(i: Int) = i * 2
```

Ese método proporciona una forma sencilla de demostrar ScalaTest.

4) Crea tus primeras pruebas de ScalaTest

ScalaTest es muy flexible y ofrece varias formas diferentes de escribir pruebas. Una forma sencilla de comenzar es escribir pruebas utilizando ScalaTest AnyFunSuite. Para comenzar, cree un directorio llamado math en el directorio src/test/scala :

```
$ mkdir src/test/scala/math
```

A continuación, cree un archivo llamado MathUtilsTests.scala en ese directorio con el siguiente contenido:

```
package math
```

```
import org.scalatest.funsuite.AnyFunSuite

class MathUtilsTests extends AnyFunSuite:

 // test 1
 test("'double' should handle 0") {
  val result = MathUtils.double(0)
  assert(result == 0)
 }

 // test 2
 test("'double' should handle 1") {
  val result = MathUtils.double(1)
  assert(result == 2)
 }

 test("test with Int.MaxValue") (pending)

end MathUtilsTests
```

Este código demuestra el AnyFunSuite enfoque ScalaTest. Algunos puntos importantes:

- Tu clase de prueba debería extenderse AnyFunSuite
- Las pruebas se crean como se muestra, dándole a cada una test un nombre único.
- Al final de cada prueba, debe llamar assert para comprobar que se ha cumplido una condición.
- Cuando sepa que desea escribir una prueba, pero no quiere escribirla ahora, cree la prueba como "pendiente", con la sintaxis que se muestra.

Usar ScalaTest de esta manera es similar a JUnit, por lo que si llega a Scala desde Java, con suerte esto será similar.

Ahora puedes ejecutar estas pruebas con el sbt test comando. Saltándose las primeras líneas de salida, el resultado se ve así:

```
sbt:HelloScalaTest> test


[info] Compiling 1 Scala source ...
[info] MathUtilsTests:
[info] - 'double' should handle 0
[info] - 'double' should handle 1
[info] - test with Int.MaxValue (pending)
[info] Total number of tests run: 2
[info] Suites: completed 1, aborted 0
[info] Tests: succeeded 2, failed 0, canceled 0, ignored 0, pending 1
[info] All tests passed.
[success] Total time: 1 s
```

Si todo funciona bien, verá un resultado similar a este. Bienvenido al mundo de probar aplicaciones Scala con sbt y ScalaTest.