Scala and Akka Actors

Scala

- Facile interazione con Java
- Completamente orientato agli oggetti e funzionale
- Qualsiasi cosa è un oggetto, compresi numeri e funzioni
- Le funzioni si possono passare come argomenti e salvare in variabili

```
object HelloWorld
{
    def main(args: Array[String])
    {
        println("Hello World!")
    }
}
```

Classi

Molto simili alle classi Java, con un'importante differenza:

possono avere dei parametri

```
class Complex (real: Double, imaginary: Double){
    def re() = real
    def im() = imaginary
}

new Complex(1.5, 2.4)

// tipo di ritorno dedotto automaticamente dal compilatore
```

Object e Case Classes

Oltre alle classi, Scala offre gli **Object:**

- Sono classi con un'unica istanza
- Istanziati una sola volta
- Visibilità globale
- Non esistono membri statici, si usano gli Object

e le **Case Classes:**

- Setter e getter definiti automaticamente
- Costruttore definito automaticamente
- Metodi *toString* e *equals* definiti automaticamente

Pattern Matching

```
object Test {
   def main(args: Array[String]) {
      println(matchTest("two"))
      println(matchTest("test"))
      println(matchTest(1))

   }
   def matchTest(x: Any){
      x match {
         case 1 => "one"
         case "two" => 2
         case y: Int => "scala.Int"
         case _ => "many"
      }
   }
}
```

Vantaggi di Scala

- **NON CI SONO I PUNTI E VIRGOLA**
- Supporto nativo per il pattern matching
- I programmi scritti in Scala sono molto più piccoli
- Moltissime strutture dati e funzioni
- Object
- Case classes

Attori Akka

- Modello per la programmazione concorrente
- Attori come primitiva universale
- Realizzazione sistemi concorrenti e distribuiti
- Un Attore può inviare e ricevere messaggi
- Permette di eseguire chiamate asincrone

Attori Akka esempio

```
import akka.actor.Actor
import akka.actor.ActorSystem
import akka.actor.Props

class HelloActor extends Actor {
  def receive = {
    case "hello" => println("hello back at you")
    case _       => println("huh?")
  }
}

object Main extends App {
  val system = ActorSystem("HelloSystem")
  // default Actor constructor
  val helloActor = system.actorOf(Props[HelloActor], name =
      "helloactor")
  helloActor ! "hello"
  helloActor ! "buenos dias"
}
```

Configurare l'ambiente

Tutto quello che vi serve:

- **Scala:** `http://www.scala-lang.org/downloads`
- **Akka Actors:** `http://akka.io/downloads/`
- **SBT (Simple Build Tool):** build tool per Scala

  `https://github.com/harrah/xsbt/wiki/Setup`
- **Tutorial:** `http://doc.akka.io/docs/akka/2.0.2/intro/getting-started-first-scala.html`

Simple Build Tool

Aggiungete le seguenti righe di codice al file *Build.scala*

```
name := "My Project"

version := "1.0"

scalaVersion := "2.9.1"

resolvers += "Typesafe Repository" at
    "http://repo.typesafe.com/typesafe/releases/"

libraryDependencies += "com.typesafe.akka" % "akka-actor" %
    "2.0.2"
```

Imparare Scala

http://my.safaribooksonline.com/book/programming/scala/9780132761772

Imparare Scala

- **Esercizi:** `http://aperiodic.net/phil/scala/s-99/`
- **Tutorial:** `www.scala-lang.org/docu/files/ScalaTutorial.pdf`
- **Documentazione:** `www.scala-lang.org/docu/files/ScalaReference.pdf`