Scala

Felix Dietrich

Agenda

- Introduction
- A First Example DEMO
- Object Oriented Programming DEMO
- Functional Programming DEMO
- Concurrent Programming
- Actors WORKSHOP

Introduction

- OOP & FP at the same level
- Runs on JVM
- Focus on concurrent programming
- “Scala By Example”, Martin Odersky, 2009

Introduction

- Scala is a **general purpose** programming language designed to express **common programming patterns** in a concise, elegant, and **type-safe** way.

http://www.scala-lang.org/

Introduction

- It smoothly integrates features of **object-oriented** and **functional** languages, enabling Java and other programmers to be more productive.

http://www.scala-lang.org/

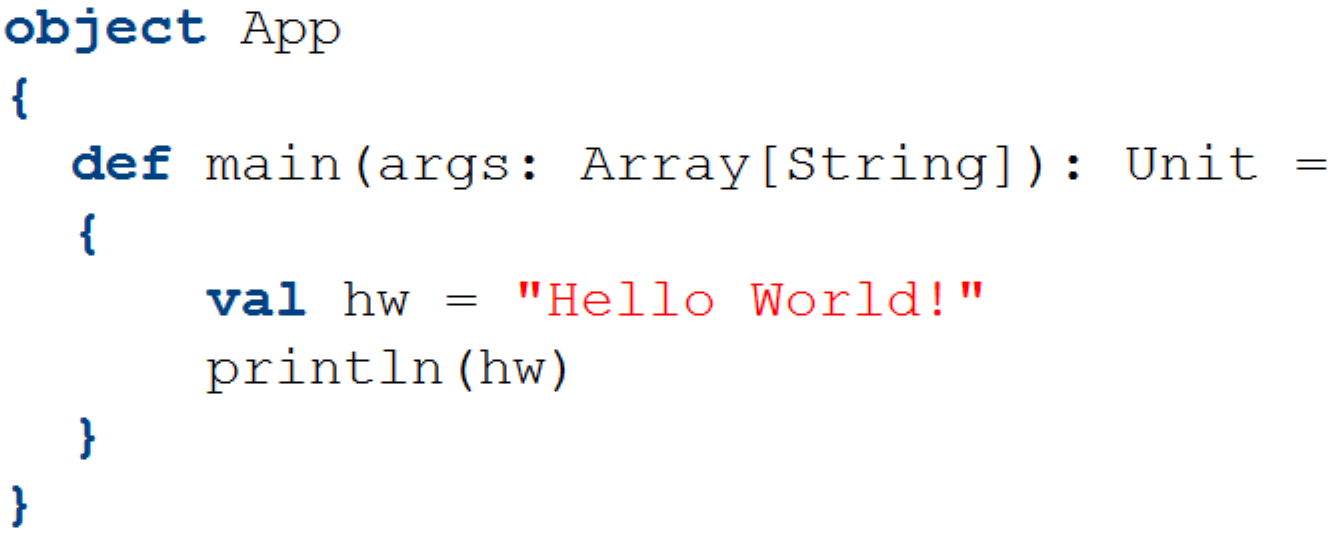
Introduction

- **Code sizes are typically reduced by a factor of two to three** when compared to an equivalent Java application.

http://www.scala-lang.org/

A First Example

DEMO



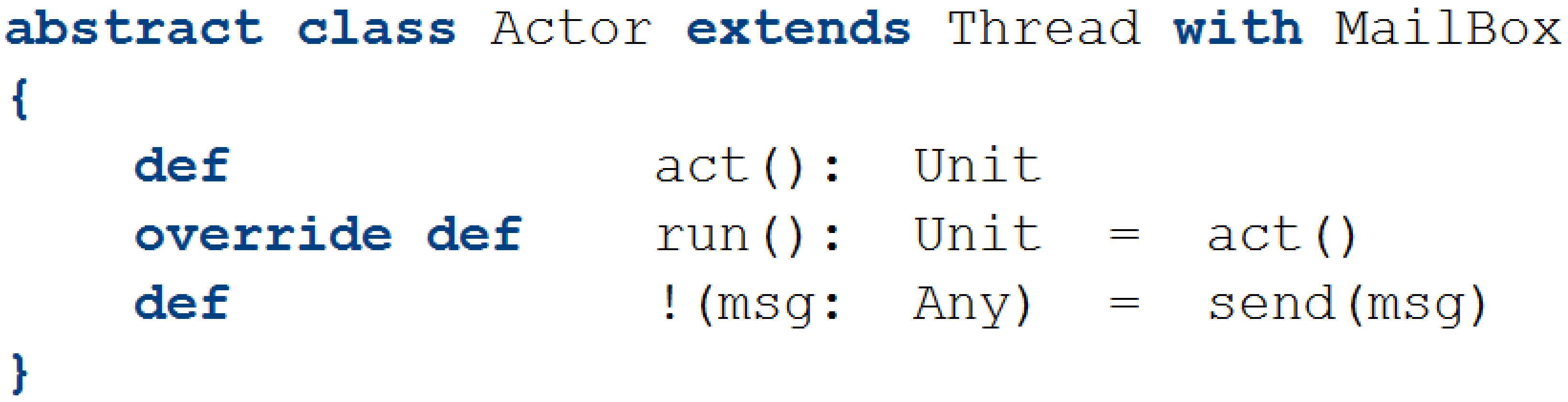
```
object App
{
  def main(args: Array[String]): Unit =
  {
      val hw = "Hello World!"
      println(hw)
  }
}
```

ScalaIDE

- Available on github: https://github.com/felix11/MSP-Workshops
- Java must be installed
- Scala path must be set

- No whitespace in paths!

A (more advanced) First Example



```
abstract class Actor extends Thread with MailBox
{
    def                 act():  Unit
    override def        run():  Unit  =  act()
    def                 !(msg:  Any)  =  send(msg)
}
```

Object Oriented Programming

DEMO

- **`class`**, **`extends`**, **`abstract`**, see Java
- **`def`**, see Java: member
- **`trait`**, see Java: interface, implemented

Functional Programming

DEMO

```
range(1,10) |> (map(square,_)) |> (reduce((a,b)=>a+b, 0, _))
```

Concurrent Programming

- **SyncVars** – synchronized variables
- **Futures** – future results
- **Mailboxes** – receiving messages
- **Actors** – concurrent communication

Actors

- Scala's **primary concurrency construct** is actors.
- Actors are basically concurrent processes that **communicate by exchanging messages**.
- Actors can also be seen as a form of **active objects** where invoking a method corresponds to sending a message.

http://www.scala-lang.org/

Actors

WORKSHOP