Scala基础与实践

陈超
@CrazyJvm

- IN ACTION ！ ！ ！

Scala

- 基于JVM的FP+OO
- 静态类型
- 和Java互操作

- 解释器(interpreter)
- 值与变量(val & var)
- 函数(Function)

- 条件表达式(if)
- 循环表达式(no continue, no break)
- 语句终止(; no need)

- 类(class)
  - 声明类（一个源文件中可以包含很多类，并且都是public级别）
  - getter 和 setter
  - 构造函数(primary constructor & auxiliary constructor)
  - 继承(extends)
  - 重写父类方法(override def)
  - 重写字段(override val , override var)

- 抽象类（abstract class）
  - 类的一个或者多个方法没有没完整的定义
  - 声明抽象方法不需要加abstract关键字,只需要不写方法体
  - 子类重写父类的抽象方法时不需要加override
  - 父类可以声明抽象字段(没有初始值的字段)
  - 子类重写父类的抽象字段时不需要加override

- 特质(trait) —对比下JAVA8的接口
  - 字段和行为的集合
  - 混入类中
  - 通过with关键字，一个类可以扩展多个特质

- trait 续
  - 当做接口
  - 带有具体实现的接口
  - 带有特质的对象
  - 特质从左到右被构造

- apply方法
- 单例对象

- 包(package com.xx.data)
  - 支持嵌套，下层可以访问上层作用域中的名称
  - 可串联
  - 顶部标记
  - 包对象
  - 包可见性
  - 包在任何地方都可以引入，作用域至该语句所在块的末尾
  - 重命名引入成员(xx => yy)
  - 隐藏方法(xx => _)
  - 自动引入(java.lang._　scala._　Predef._)

- 模式匹配
  - 标准用法(match)
  - 使用守卫
  - 匹配类型

- case class(多用在模式匹配中)
  - 构造器中的每一个类型都为val， 不建议用var。
  - 不用new就可以直接产生对象(为什么？apply方法)

- 高阶函数
  - 匿名函数 val double = (x : Int) => 2 * x
  - 函数作为参数
  - 参数类型推断
  - 常用高阶函数
    - map, filter, reduce等等

- 集合
  - List
  - Set
  - Tuple
  - Map

- 集合操作
  - foreach(类似于map，但是没有返回值)
  - map(迭代)
  - filter(过滤)
  - zip(聚合)
  - partition(列表分割)
  - flatten(扁平化)
  - flatMap(map + flatten)

- 泛型
  - 泛型类
    - class Pair[T, S](val first : T, val second : S)
  - 泛型方法
    - def compute[T](list : List[T]) = …

- 隐式转换
  - 位于源目标类型的伴生对象中的隐式函数
  - 位于当前作用域可以以单个标识符指代的隐式函数
- 隐式参数
- 隐式类(Scala 2.10)

- think more ! write more !

谢谢大家