def f : Java=>Scala

—— 在Java codebase 中引入Scala的正确姿势

前言

- 自我介绍
- 当我们喷Java的时候，我们都在喷一些什么？
- 可是也许，Java也是令人喜爱的？
- 所以，我选Scala！
- 平衡与克制
- 共生与不强

当我们喷Java的时候，我们都在喷一些什么？

- 啰嗦冗余
- (?)僵化死板（大量的固定模式/规则/约束）
- 割裂
- 残废（泛型，lambda, stream api）
- 历史包袱重

可是也许，Java也是令人喜爱的?

- 超级强大的JVM系统，优秀的类加载机制
- 多年积累的生态
- 多年积累的最佳实践经验与模式
- 诸多大厂背书，诸多大型开源项目实践背书

所以，我选Scala!

- 站在Java巨人的肩膀上
- 改善诸多Java留下的包袱和痛点，减少错误
- （非常）强大的语法特性组合，丰富的武器库
- Scala作者在设计上的苦心孤诣
- 少打字，省键盘，代码量直线下降，令人懒惰🐶
- 简练的同时，可读性（可以）非常好

所以，我选Scala!

Everytime when I code with Scala，
full powerful armor and weapons

Back to Java

前言

- 自我介绍
- 当我们喷Java的时候，我们都在喷一些什么？
- 可是也许，Java也是令人喜爱的？
- 所以，我选Scala！
- 平衡与克制
- 共生与不强

先决条件

- （？）**大数据相关团队**
- (团队规模||项目人工)<=5人
- 团队的Tech lead对**Scala**持有**Open**的态度
- 团队中除你以外至少有一个人愿意学习并使用**Scala**
- 对**Scala**语言本身有一定的知识储备
- （optional）有新项目/新模块等待开发
- （optional）找到一个能证明“好十倍”的场景

override val f : Java=>Scala =Scala collection

——强大的集合设计让处理集合变得简单而享受

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Scala collection

- 集合字面量
- map/flatMap/filter/foreach
- 集合交并补
- fold/group by/forAll/exists/(*)zip/mkString
- 集合类的高度统一性与集合互转
- Array in Scala
- (*)可变集合
- (*)For comprehension

对Scala collection的评估

- 推荐指数： 
- 实用指数： 
- 效率提升：Java8+：  （before8）： 
- 错误避免： 
- 推荐理由：Scala集合提供了一套逻辑自洽功能强大的API，利用他处理集合相关的非常简单容易，强大的类型推理帮助我们提早发现错误。集合用于串联其他操作也很方便
- 注意事项：Scala集合提供的Api非常多，建议大家循序渐进地学习和使用feature（Collection的设计其实非常的精妙和自洽）。同时建议不要刻意追求缩短代码行数

override val f : Java=>Scala = Immutable

——默认且广泛使用的不可变减少code的bad smell

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Immutable

- 从Java的Final出发
- 并发编程的安全性
- 状态“消除”
- 可读性与简洁性
- Scala，make immutable as default
- Use Immutable as possible(平衡心态)

对Immutable的评估

- 推荐指数：★★★★
- 实用指数：★★★★
- 效率提升：★★★
- 错误避免：★★★☆
- 推荐理由：不可变已经事实成为最佳实践的推荐做法，Scala使之成为默认行为，同时组合Scala的强大语法，可以使在Java中不易val的情况在Scala变得可能
- 注意事项：虽然不可变很好，但是可变在很多场景是好用甚至是必要的，我们应该在适当的场合选择使用可变，同时利用作用域封闭等情况使可变尽量安全

override val f : Java=>Scala = Try & Option

——应对null值和异常的另一种优雅手段

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Try & Option

- Eliminate null with Option
- Compare with java.Optional
- Handle error with Try
- (*)Talk about Either
- (*)Work with for comprehension

对Try&Option的评估

- 推荐指数： 
- 实用指数： 
- 效率提升： 
- 错误避免： 
- 推荐理由：利用Option去**处理**和**组合**可能含有null的逻辑。利用Try去**处理**和**组合**可能出现异常的逻辑
- 注意事项：站在jvm上，就算使用Option也还是要有时与null打交道，同事避免出现Some(null)的情况出现。try-catch仍然是有用的

override val f : Java=>Scala = Scala Function

——源自函数式的强大function概念功能助力开发

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Scala Function

- 函数一等公民与引用透明
- (*)**Function** or **Method**
- 匿名函数
- 在function中定义function
- Function with default value

对Scala Function的评估

- 推荐指数：  （你没机会拒绝它）
- 实用指数： 
- 效率提升： 
- 错误避免： 
- 推荐理由：Scala在函数这块借鉴了大量函数式编程的优秀概念和范式（而非直接使语言是函数式语言），确实很强大好用
- 注意事项：可读性和简洁优先，尽可能使你的函数引用透明的

override val f : Java=>Scala = Future

——并发和异步原来可以更简单

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Future

- Map/FlatMap
- (*)Promise
- foreach/onComplete/andThen
- Future.sequence/Future.traverse
- Await

对Future的评估

- 推荐指数：★★★ （确实有需求再用）
- 实用指数：★★★☆
- 效率提升：★★★
- 错误避免：★★☆
- 推荐理由：Scala Future通过强大的功能使得异步执行逻辑可以向同步逻辑的方向简化，同时在异步逻辑的组合和结果的获取上变得容易和灵活
- 注意事项:敬畏！并发编程的固有复杂度就在那里，写的时候一定要小心。对于异步编程的失败边界一定要划分清楚！

override val f : Java=>Scala = Pattern Match

——绝对强大的模式匹配大幅提升编程质量

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Pattern match

- Not only switch case
- Not only if else
- Pattern match with case class/Try/Option/Future/Either/etc.
- (*)Pattern match with Extractor
- (*)Pattern with Regex

对Pattern Match的评估

- 推荐指数：  （真的好用)
- 实用指数： 
- 效率提升： 
- 错误避免： 
- 推荐理由：找不到不推荐的理由，全面的提升编程体验和可读性
- 注意事项：注意匹配项要写全，或者使用通配符兜底（as default）

override val f : Java=>Scala =Work with Java

Work with Java

- 轻松一步开启Scala/Java混编之路
- 善待legacy code
- 利用Scala替换Java不方便完成的逻辑/方法/模块
- 选择“朴实”与“谨慎”的方式与Java交互
- Equality in Java & Equality in Scala
- 值类型
- Java与Scala泛型约束的差别
- Scala与Java集合间转换的考量

override val f : Java=>Scala = Other tips

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Other tips

- String operations/regex
- Tuples
- Define empty
- Alias when import
- (*)for comprehension
- (*)scala.process.sys
- (*)scala.parsing.combinator
- (*)Implicit value/class

override val f : Java=>Scala =Advanced topic

作者：刘首维 Mail:455211062@qq.com，非商业用途随意转载

Advanced topic

- Real meaning of Map/FlatMap
- Deep dive into Scala
- Work with Akka
- Work with Cats
- Work with Spark

后记

- Scala编译很慢吗?
- Scala很复杂吗?
- Scala的定位是?
- 系统复杂性，业务复杂性与语言复杂性
- 吐槽Scala?
- 付出必要学习成本
- 期望

附录：Scala的高瞻远瞩

Java新特性	对应Scala特性	Java实现版本	在Scala与Java的对比
Lambda 表达式	Scala Function的小子集	1.8	Scala>>>Java
方法引用	一等公民+（*）eta转换	1.8	Scala>>Java
接口默认方法	Trait功能子集	1.8	Scala == Java
Stream Api	Scala collection子集	1.8+	Scala>>Java
Optional	Option	1.8+	Scala>>Java
CompletableFuture	Future	1.8+	Scala>Java
创建部分不可变集合	Scala collection子集	1.9+	Scala>Java
局部变量使用var	基础语法特性下位版本	1.10+	Scala>>Java
String增强方法系列	Scala String基础语法	1.11+	Scala >Java
Swtich case增强	Pattern match 子集	1.12+	Scala>>>>>>Java
Raw String支持	Scala Raw String子集	1.13+	Scala>Java
Record class	Scala case class	1.14	Scala==Java