# 课堂笔记

## 包引入

**import** scala.collection.mutable.ListBuffer

**import** scala.collection.mutable.\_ 其中\_同Java里的\*,表示全部

**import** scala.collection.mutable.{ListBuffer,HashMap} 多个类

import语句可以写到任何地方，Scala特性，但感觉没必要。

**隐式引入**

Java里默认引入了java.lang.\* , 所以不需要显式import就可以使用该包下的类。

Scala默认引入了如下几个包，依次顺序为：

import java.lang.\_

import scala.\_

import Predef.\_

首先引入java.lang.\_ ，当引入scala包时会覆盖之前的引入，比如scala.StringBuilder 会覆盖java.lang.StringBuilder，而不会冲突。

比如 **var** a:*StringBuilder* = **new** *StringBuilder*

如何用Java的：

**var** a:java.lang.StringBuilder = **new** java.lang.StringBuilder()

对于其他scala开头的包，不需要写成如下完整路径：

scala.collection.mutable.ListBuffer

简写为collection.mutable.ListBuffer 即可。

## 泛型

作用和Java里一样，这个概念是模仿Java的。

举例说明泛型作用：

def fun(a:String) ，只能传入String类型参数，但如果预期变了需要传入其他类型时该函数就不支持了。

如果改为def fun(a:Any) ，则可支持传入任何类型的参数，但采用Any进行定义时无法使用一些功能，比如无法使用Iterable的map和filter函数。

简单案例如下：

**import** scala.collection.mutable.Iterable

**class** Test1[A,B](a:A,b:B) { //泛型类

**def** fun[T](i:T):Unit= //泛型函数

{

println(i.toString()) //正确，因为任何类型都有toString()函数

// println(i.map(\_ + 1)) //提示有误，原因是无法确保T类型存在map()函数，如何解决？

}

}

**object** Test1

{

**def** main(args: Array[*String*]): Unit = {

**val** t1 = **new** Test1(1,"hello") //可以传入任意类型，编译器会自动推断类型

**val** t2 = **new** Test1("aa",Array(12,334,566)) //试想，如果类型A和B是明确类型，是无法同时满足t1和t2的入参格式的。

//泛型就是提供这种便利：

//1、定义class或函数时，使用泛型，不明确指定参数的类型。

//2、引用的时候根据入参，编译器会自动推断参数的类型。

t2.fun("123")

t2.fun(233)

t2.fun(Array(12,334,566))

}

}

## 注解 Annotation

和泛型一样是继承（模仿）Java的特性，但青出于蓝胜于蓝，比Java的强大。

之前有讲过@BeanProperty，给属性字段自动加上getter和setter方法。

Scala里可以给类、方法、属性字段、参数、表达式及其他任何类型都可以添加注解。