目录

目录

[Scala 学习笔记 1](#_Toc468975960)

[概述 1](#_Toc468975961)

[Intellij IDEA 2](#_Toc468975962)

[入门 6](#_Toc468975963)

[语法 6](#_Toc468975964)

[基本类型 7](#_Toc468975965)

[语句 7](#_Toc468975966)

[面向对象化编程 7](#_Toc468975967)

[XML 解析 8](#_Toc468975968)

[scala-swing 工具包 8](#_Toc468975969)

[异常处理 try-catch 11](#_Toc468975970)

[scala语言包 11](#_Toc468975971)

[ISSUE 12](#_Toc468975972)

[\* 错误: 找不到或无法加载主类 12](#_Toc468975973)

[应用场景 12](#_Toc468975974)

# Scala 学习笔记

**Joshua Conero**

## 概述

日期： 2016年10月17日 星期一

参考网站：<http://www.scala-lang.org/>

语言含义/特性：[WHATE IS SCALA](http://scala-lang.org/what-is-scala.html) ?

**scalability**-可扩展性，

**Scalable Language**-可伸缩语言

**Object-Oriented** - 面向对象

every value is an object and every operation is a method-call

The language supports advanced component architectures through classes and traits: **Class-Trait-Object**

**Functional**- 函数式

Scala is also a full-blown functional language

**Seamless Java Interop**- 与Java无缝对接

Java libraries, frameworks and tools are all available. Build tools like ant or **maven**, IDEs like **Eclipse**, **IntelliJ**, or **Netbeans**, frameworks like **Spring** or **Hibernate** all work seamlessly with Scala. Scala runs on all common **JVMs** and also on **Android**.

**Functions are Objects**- 函数即对象

The function type is just a regular class. The algebraic data types found in languages such as **Haskell**, **F#** or **ML** are modelled in Scala as class hierarchies. Pattern matching is possible over arbitrary classes

To some, Scala feels like a scripting language - 像脚本语言

can play with it by typing one-line expressions and observing the results

can also rely on it for large mission critical systems, as many companies, including Twitter, LinkedIn, or Intel do.

**Future-Proof** - 未来

makes use of concurrent and synchronous processing, parallel utilization of multiple cores, and distributed processing in the cloud.

Its functional nature makes it easier to write safe and performant multi-threaded code.

相关词汇其他： REPL and IDE

Scala是纯面向对象的，每个值都是一个对象

Scala的设计目的是要和两种主流面向对象编程语言Java和C#实现无缝互操作，这两种主流语言都非纯面向对象。

Scala始于2001年，由**洛桑联邦理工学院**(EPFL)的编程方法实验室研发。

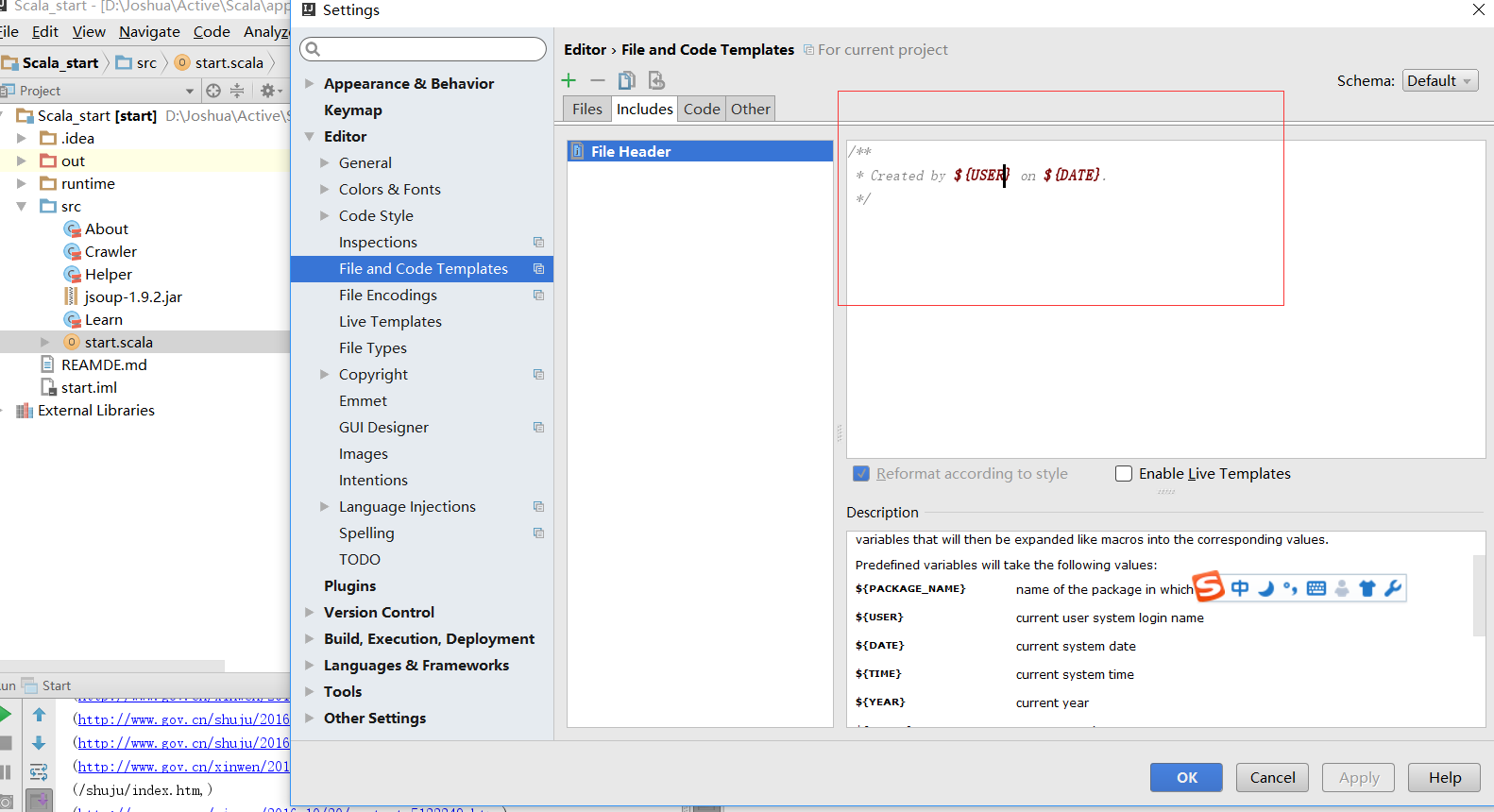
可运行JVM/.NET

### Intellij IDEA

网站: <http://www.jetbrains.com/idea/>

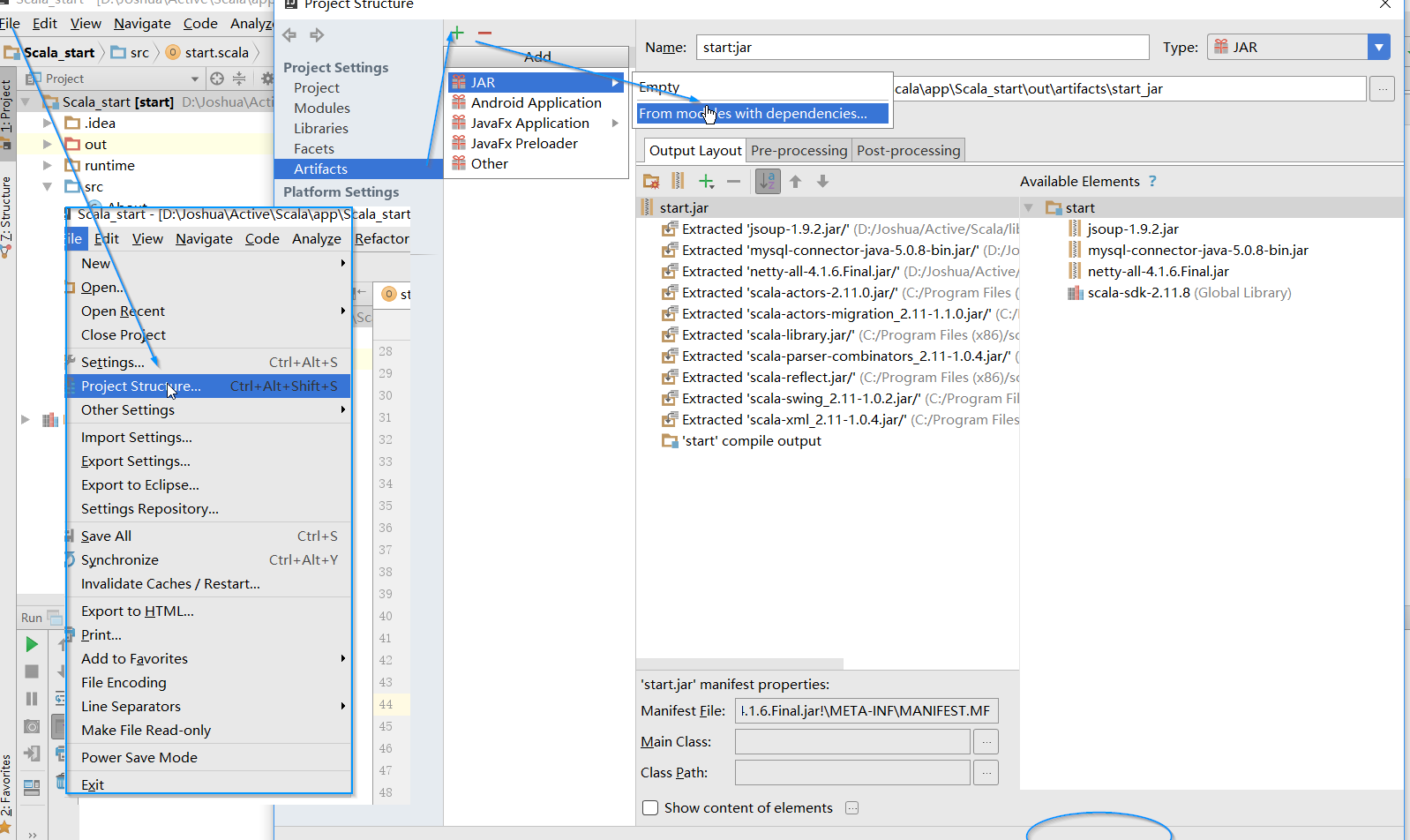
**设置代码头注释格式**:

1. 在IDEA环境中去设置，File->Settings->File and Code Templates(IDEA14)->Include->File Header:修改${USER}为自定义名字

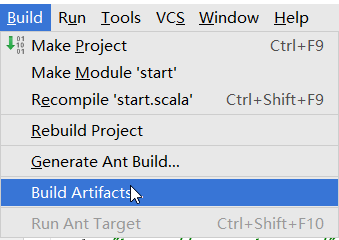


\*来源: [http://www.tuicool.com/articles/QRfYZrB](%20http:/www.tuicool.com/articles/QRfYZrB)

**intellij IDEA 打包-项目为jar**:



如图执行1



如图执行2

\*来源: [http://blog.csdn.net/baolibin528/article/details/50315863](%20http:/www.tuicool.com/articles/QRfYZrB)

自开发jar包：

**脚本：**

**def** point(x:Int,y:Int): Int ={

}

// 1. conero. scala

**package** conero

**class** Helper{}

// 2. Crypto.scala

**package** conero  
**class** Crypto {}

// n...

打包: conero.jar

使用>>

import conero.Helper

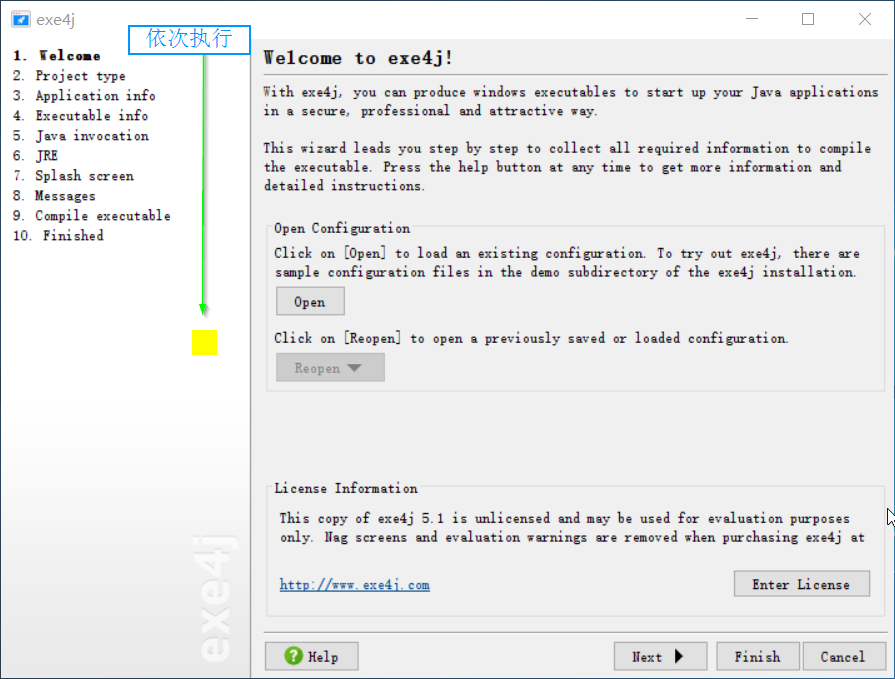
jar 转 exe 的方式工具：

1. exe4j

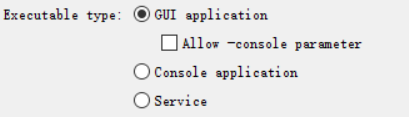
2. maven

**exe4j** :

下载地址如: <http://www.softpedia.com/dyn-postdownload.php/7b90a13450eaee0b17941a5bf5a997ce/5822d0a4/2a16c/0/3?tsf=0>。



如图运行



选择程序类型： GUI/Console/Server 脚本类型

\* 图形化程序/窗口应用程序/服务器程序

\* exej 商业软件， 若未注册会有“this executable was created an evalution version of exe4j”。[注册码](http://conkeyn.iteye.com/blog/1743199)。

用户名和公司名可随便填

A-XVK258563F-1p4lv7mg7sav

A-XVK209982F-1y0i3h4ywx2h1

A-XVK267351F-dpurrhnyarva

A-XVK204432F-1kkoilo1jy2h3r

A-XVK246130F-1l7msieqiwqnq

A-XVK249554F-pllh351kcke50

A-XVK238729F-25yn13iea25i

A-XVK222711F-134h5ta8yxbm0

A-XVK275016F-15wjjcbn4tpj

A-XVK275016F-15wjjcbn4tpj

\*

## 入门

1. 支持";"或者“空”结尾
2. 数组下标与其他语言不同的 Array(1) 第一个元素，Array(2),Array(n),其他语言：Array[1],Array[2],Array[n]

### 语法

**基本类型定义**

**函数：**

**def** point(x:Int,y:Int): Int ={

}

说明： point函数返回 Int类型数据

**定义变量**：

var num = 1

常量：

val PI = 3.14

词法的两种模式：Scala模式与XML模式。

关键字：

**abstract case catch class def**

**do else extends false final**

**finally for forSome if implicit**

**import lazy match new null**

**object override package private protected**

**requires return sealed super this**

**throw trait try true type**

**val var while with yield**

**\*. lazy** 惰性变量，实现延迟加载(懒加载)。惰性变量只能是不可变变量，并且只有在调用惰性变量时，才会去实例化这个变量。

如:

lazy val ui = new Panel {}

def top = new MainFrame {

content += ui

}

?**柯里化的函数**

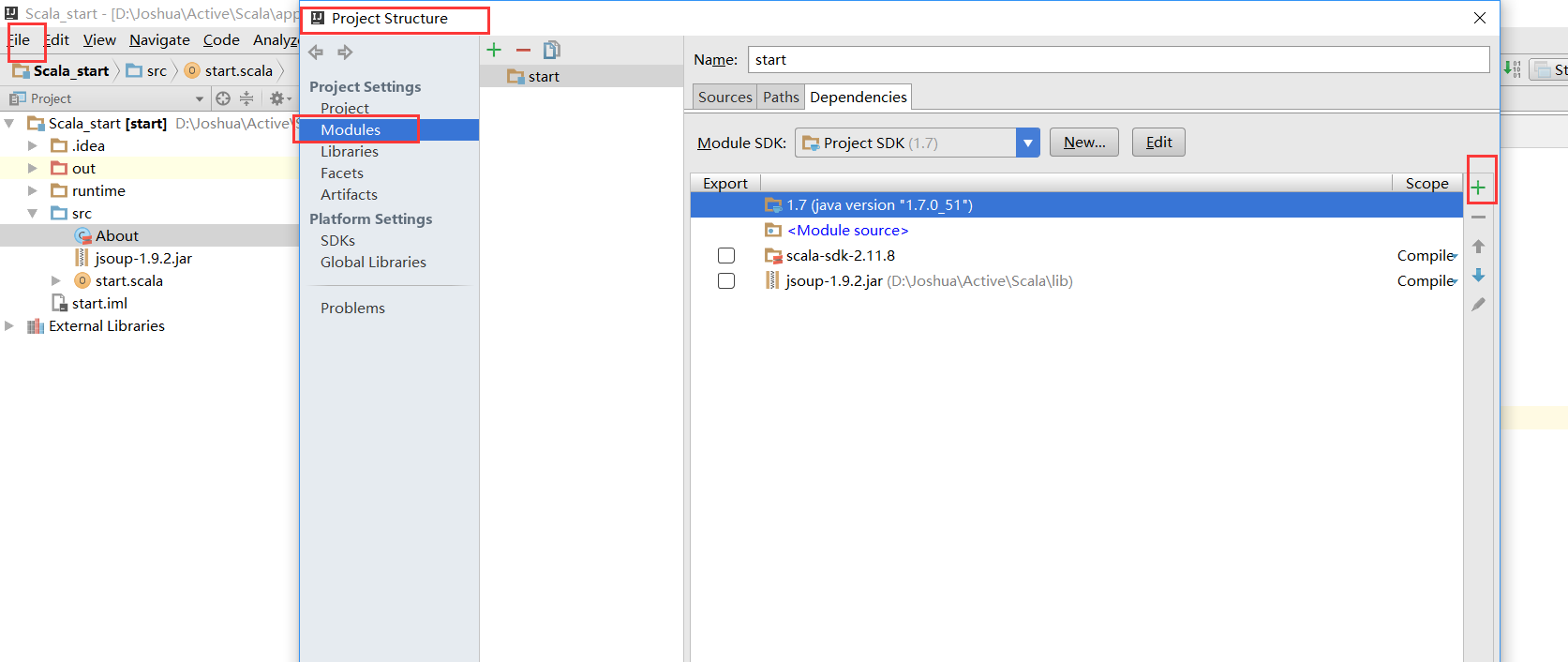
**绑定**：命名在局部定义与声明，继承，import子句，package子句中存在，这些可以统称为绑定。

**Java/Scala调用第三方Jar**:

1.**Intellij IDEA**

选择: >File>Project>Structrue>Modules>

选择:>Dependencies>左边 "+">1.Jars or ....



### 基本类型

类型判断: isInstanceOf

如: "string". isInstanceOf[String] => true

### 语句

for

// for to/until to >= 10 / until < 10

**for**(i <- 0 to 10) println(i)

for(v <- Array[String]) println(v)

for((k,v) <- Array[String]) println(k,v)

for(

### 面向对象化编程

1. **private变量**：在Scala声明private变量会Scala编译器会自动生成get/set,在Scala中变量是需要初始化的，如果不声明private默认是public的

2. **trait**：在java中可以通过interface实现多重继承，在Scala中可以通过特征（trait）实现多重继承，不过与java不同的是，它可以定义自己的属性和实现方法体，在没有自己的实现方法体时可以认为它时java interface是等价的，在Scala中也是一般只能继承一个父类，可以通过多个with进行多重继承。

3. **Class**中的method的参数 默认 是 val，及不可修改类型

4.**object**:

Object 在Scala里面的引入可能就是因为scala没有静态的方法和字段，没有办法 实现类似C++的singleton模型所以引入object

Object由于上述原因，同类的差别在于没有状态，只有方法。换句话说成员中仅有val没有var，而且不能给object传递参数

Object定义并不表示声明了一个新的类型，定义一个变量是obj类型是错误的。

object对象类似于Java的静态类,它的成员、它的方法都默认是静态的。

构造函数：

Class 内可将游离的代码作为构造函数，级系统自己判断。

**类/构造函数:**

**class** Learn() {

**val** *u* = 0 // 属性  
 run()  
 **def** run():Unit = {  
 test() // this.test() 可省this.  
 run\_hack()  
 }

def test():Unit = {}

def run\_hack():Unit = {}

}

### XML 解析

引入包： **import** scala.xml.XML

1. p\"test" p的直接下级text 元素
2. p\"@name" p的当前元素name属性
3. p\\"test" p的所有下级text 元素
4. p\\"@name" p的所有下属name属性

### scala-swing 工具包

***位置：***

Orientation.*Vertical 垂直布局*

Orientation. Horizontal *水平布局*

***表单基本元素***

*文本输入框*

1. TextField 输入框

2. FormattedTextField 输入框/格式检测

3. TextArea 文本域

4. PasswordField 密码输入框

5. TextComponent 为文本抽象框

*按钮* ：

1. Button 基本按钮

2. ButtonGroup 按钮组

3. ToggleButton 开关按钮

*其他* :

Label标签

CheckBox复选框

ComboBox 下拉框

RadioButton 单选框

ProgressBar 进度条

FileChooser 文件选择器

ColorChooser 颜色选择器

Slider 滑块

***菜单：***

1. CheckMenuItem 下拉菜单

2. RadioMenuItem 单选菜单

Menu 菜单

MenuBar 菜单条

MenuItem 菜单项

PopupMenu popup菜单

Separator 菜单分割符/分割线

Action 动作

***面板*** ：

Panel 面板

TabbedPane tab面板

BoxPanel 类似HTML中的div

SplitPane 可拖动面板

EditorPane 编辑面板- 可现实HTML代码

GridPanel 表格/网格面板

FlowPanel 流式面板- 会自动漂移

SplitPane 分隔栏

BorderPanel 边界面板

***其他组件***：

ListView 列表

Dialog 对话框

Frame 浮动框

MainFrame 主浮动框

LayoutContainer 布局容器

ScrollBar 滚动条

ScrollPane 滚动面板

Table 表格

***属性***设置：

preferredSize 应用的首选大小/用于布局管理器

maximumSize 最大尺寸

minimumSize 最小尺寸

size 类似觉得尺寸

border 内边距

enabled 有效/无效设置

editable 可否编辑

focusable 焦距获取

font 字体

lineWrap 行距

charWrap 字间距

visible 显示/隐藏

Cursor 鼠标形状

background 背景色

foreground 前景色

opaque 可否透明

bounds 边界/ UIElement

location 位置

preferredSize = **new** Dimension(800, 500)

border = Swing.*EmptyBorder*(5,5,5,5)

***表单基本元素***

事件- scala.swing.event

listenTo 事件监听

deafTo 事件监听解除

**窗口关闭监听事件**

**// 窗口关闭监听事件**

**new** MainFrame {

listenTo(**this**)  
*reactions* += {  
 **case** e:event.WindowClosing => *// 窗口关闭监听*

println(**"窗口正在关闭"**)

}

// alter 框

Dialog.*showMessage*(**this**,**"内容 "**,**"标题"**)

**// tabPane 选择事件**

**// tabPane 选择事件**

[val](http://www.ne.jp/asahi/hishidama/home/tech/scala/statement.html#val) *tabPane* = new **TabbedPane** {

preferredSize = new Dimension(256, 212)

[listenTo](http://www.ne.jp/asahi/hishidama/home/tech/scala/swing/event.html)(**this.selection**)

[reactions](http://www.ne.jp/asahi/hishidama/home/tech/scala/swing/event.html) += {

case **event.SelectionChanged**(source) =>

if (**selection.index** >= 0) {

val page = **selection.page**

println(page.**title**)

}

}

}

### 异常处理 try-catch

**抛出异常**

throw new IllegalArgumentException; 抛出异常

**try-catch-finally**

try 抛出异常，catch 捕获后使用case块进行处理，并在case里面中断(如果没有finally)。使用finally子句可在错误异常处理完成以后继续执行

try{

跑错语句

}catch{

case e: Exception => 语句...

case e:xxx => ...

}finally{

... 异常完成以后

}

### scala语言包

scala.annotation 注解

scala.beans 网络程序员的资源

scala.collection 集合

scala.compat 兼容

scala.concurrent 并行

scala.io 读写

scala.math 数学

scala.ref 参考

scala.reflect 反射

scala.runtime 运行

scala.sys 系统

scala.text 文本

scala.util 工具

## ISSUE

### \* 错误: 找不到或无法加载主类

?? 在运行正常的程序包中引入 package 名称报错

**// 引入包错误**

**package** conero

**object** CrawlerGUI {

**def** main():Unit = {

println("引入包时错误！")

}

}

**// 可正常运行**

**object** CrawlerGUI {

**def** main():Unit = {

println("引入包时错误！")

}

}

### \* (swing-scala) TabbedPane 的tab标题和index

遍历 TabbedPane 的tab标题和index

*// 验证 tab是否已经存在，防止重复生中 ??***def** tabExist(tab:TabbedPane,title:String):Boolean = {  
 **val** all = tab.selection.*index* **if**( all >= 0){  
 **for**(i <- 0 to all){  
 tab.selection.*page*(i)  
 }  
 }  
 **false**}

## 应用场景

Scala是数据挖掘算法领域最有力的编程语言之一

#### 网络爬虫+数据分析+分析数据保存/ Scala\_start

概述：

1.java框架

2.Scala与java互操作

3.Scala数据分析以及保存处理

项目名称： **Scala\_start**

java爬虫框架介绍：

参考网站：

1. [给推荐几个github上优秀的java爬虫项目？](https://www.zhihu.com/question/31427895)

2.[网络爬虫的基本原理](http://www.cnblogs.com/wawlian/archive/2012/06/18/2553061.html)

[nutch](https://github.com/apache/nutch) apache下的开源爬虫程序，功能丰富，文档完整。有数据抓取解析以及存储的模块。

[Heritrix](https://github.com/internetarchive/heritrix3) 很早就有了，经历过很多次更新，使用的人比较多，功能齐全，文档完整，网上的资料也多。有自己的web管理控制台，包含了一个HTTP 服务器。操作者可以通过选择Crawler命令来操作控制台。

[crawler4j](https://github.com/yasserg/crawler4j)因为只拥有爬虫的核心功能，所以上手极为简单，几分钟就可以写一个多线程爬虫程序。

项目实际

一、引用库/jar

1. jsoup HTML 解析库(jsoup-1.9.2.jar)