Scala 安装及环境配置 Scala 数据类型

Scala 基础语法

如果你之前是一名 Java 程序员，并了解 Java 语言的基础知识，那么你能很快学会 Scala 的基础语法。

Scala 与 Java 的最大区别是：Scala 语句末尾的分号 ; 是可选的。

我们可以认为 Scala 程序是对象的集合，通过调用彼此的方法来实现消息传递。接下来我们来理解下，类，对象，方法，实例变量的概念：

- **对象 -** 对象有属性和行为。例如：一只狗的状属性有：颜色，名字，行为有：叫、跑、吃等。对象是一个类的实例。
- **类 -** 类是对象的抽象，而对象是类的具体实例。
- **方法 -** 方法描述的基本的行为，一个类可以包含多个方法。
- **字段 -** 每个对象都有它唯一的实例变量集合，即字段。对象的属性通过给字段赋值来创建。

第一个 Scala 程序

交互式编程

交互式编程不需要创建脚本文件，可以通过以下命令调用：

```
$ scala
Welcome to Scala version 2.11.7 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_31).
Type in expressions to have them evaluated.
Type :help for more information.

scala> 1 + 1
res0: Int = 2

scala> println("Hello World!")
Hello World!

scala>
```

脚本形式

我们也可以通过创建一个 HelloWorld.scala 的文件来执行代码，HelloWorld.scala 代码如下所示：

```
object HelloWorld {
   /* 这是我的第一个 Scala 程序
    * 以下程序将输出'Hello World!'
    */
   def main(args: Array[String]) {
```

```
        println("Hello, world!") // 输出 Hello World
    }
}
```

接下来我们使用 scalac 命令编译它：

```
$ scalac HelloWorld.scala
$ ls
HelloWorld$.class    HelloWorld.scala
HelloWorld.class
```

编译后我们可以看到目录下生成了 HelloWorld.class 文件，该文件可以在Java Virtual Machine (JVM)上运行。

编译后，我们可以使用以下命令来执行程序：

```
$ scala HelloWorld
Hello, world!
```

在线实例 »

基本语法

Scala 基本语法需要注意以下几点：

- **区分大小写** - Scala是大小写敏感的，这意味着标识Hello 和 hello在Scala中会有不同的含义。
- **类名** - 对于所有的类名的第一个字母要大写。
  如果需要使用几个单词来构成一个类的名称，每个单词的第一个字母要大写。

  示例：*class MyFirstScalaClass*
- **方法名称** - 所有的方法名称的第一个字母用小写。
  如果若干单词被用于构成方法的名称，则每个单词的第一个字母应大写。

  示例：*def myMethodName()*
- **程序文件名** - 程序文件的名称应该与对象名称完全匹配(新版本不需要了，但建议保留这种习惯)。
  保存文件时，应该保存它使用的对象名称（记住Scala是区分大小写），并追加".scala"为文件扩展名。（如果文件名和对象名称不匹配，程序将无法编译）。

  示例: 假设"HelloWorld"是对象的名称。那么该文件应保存为'HelloWorld.scala"
- **def main(args: Array[String])** - Scala程序从main()方法开始处理，这是每一个Scala程序的强制程序入口部分。

标识符

Scala 可以使用两种形式的标志符，字符数字和符号。

字符数字使用字母或是下划线开头，后面可以接字母或是数字，符号"$"在 Scala 中也看作为字母。然而以"$"开头的标识符为保留的 Scala 编译器产生的标志符使用，应用程序应该避免使用"$"开始的标识符，以免造成冲突。

Scala 的命名规则采用和 Java 类似的 camel 命名规则，首字符小写，比如 toString。类名的首字符还是使用大写。此外也应该避免使用以下划线结尾的标志符以避免冲突。符号标志符包含一个或多个符号，如+，:，? 等，比如:

```
+ ++ ::: < ?> :->
```

Scala 内部实现时会使用转义的标志符，比如:-> 使用 $colon$minus$greater 来表示这个符号。因此如果你需要在 Java 代码中访问:->方法，你需要使用 Scala 的内部名称 $colon$minus$greater。

混合标志符由字符数字标志符后面跟着一个或多个符号组成，比如 unary_+ 为 Scala 对+方法的内部实现时的名称。字面量标志符为使用"定义的字符串，比如 `x` `yield`。

你可以在"之间使用任何有效的 Scala 标志符，Scala 将它们解释为一个 Scala 标志符，一个典型的使用为 Thread 的 yield 方法， 在 Scala 中你不能使用 Thread.yield()是因为 yield 为 Scala 中的关键字， 你必须使用 Thread.`yield`()来使用这个方法。

Scala 关键字

下表列出了 scala 保留关键字，我们不能使用以下关键字作为变量：

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	return	sealed	super
this	throw	trait	try
true	type	val	var
while	with	yield	
-	:	=	=>
<-	<:	<%	>:

#	@		

Scala 注释

Scala 类似 Java 支持单行和多行注释。多行注释可以嵌套，但必须正确嵌套，一个注释开始符号对应一个结束符号。注释在 Scala 编译中会被忽略，实例如下：

```
object HelloWorld {
   /* 这是一个 Scala 程序
    * 这是一行注释
    * 这里演示了多行注释
    */
   def main(args: Array[String]) {
      // 输出 Hello World
      // 这是一个单行注释
      println("Hello, world!")
   }
}
```

空行和空格

一行中只有空格或者带有注释，Scala 会认为其是空行，会忽略它。标记可以被空格或者注释来分割。

换行符

Scala是面向行的语言，语句可以用分号（;）结束或换行符。Scala 程序里,语句末尾的分号通常是可选的。如果你愿意可以输入一个,但若一行里仅 有一个语句也可不写。另一方面,如果一行里写多个语句那么分号是需要的。例如

```
val s = "菜鸟教程"; println(s)
```

Scala 包

定义包

Scala 使用 package 关键字定义包，在Scala将代码定义到某个包中有两种方式：

第一种方法和 Java 一样，在文件的头定义包名，这种方法就后续所有代码都放在该包中。 比如：

```
package com.runoob
class HelloWorld
```

第二种方法有些类似 C#，如：

```
package com.runoob {
  class HelloWorld
```

```
}
```

第二种方法，可以在一个文件中定义多个包。

引用

Scala 使用 import 关键字引用包。

```
import java.awt.Color  // 引入Color

import java.awt._  // 引入包内所有成员

def handler(evt: event.ActionEvent) { // java.awt.event.ActionEvent
  ...  // 因为引入了java.awt，所以可以省去前面的部分
}
```

import语句可以出现在任何地方，而不是只能在文件顶部。import的效果从开始延伸到语句块的结束。这可以大幅减少名称冲突的可能性。

如果想要引入包中的几个成员，可以使用selector（选取器）：

```
import java.awt.{Color, Font}

// 重命名成员
import java.util.{HashMap => JavaHashMap}

// 隐藏成员
import java.util.{HashMap => _, _} // 引入了util包的所有成员，但是HashMap被隐藏了
```

> ***注意：****默认情况下，Scala 总会引入 java.lang._ 、 scala._ 和 Predef._，这里也能解释，为什么以scala开头的包，在使用时都是省去scala.的。*

← Scala 安装及环境配置 Scala 数据类型 →