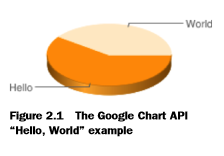
## 2.2 使用Google图表工具

Chart API是一种REST式的web服务，同时也是谷歌公布的众多API之一。就是下面那个图的API。它的文档在<https://developers.google.com/chart/image/>。这个图表工具为JavaScript使用者提供了大量的API，但是在输入端最终却是带有查询参数的URL。



开发者发送请求到URL <https://chart.apis.google.com/chart> 同时还得附加一些请求参数，以获得确定的图表种类、型号、数据等。这个API也需要一个“Hello World”例子，这有一个饼图的URL（结果你刚刚已经看到了）

<https://chart.apis.google.com/chart?cht=p3&chs=250x100&chd=t:60,40>

&chl=Hello|World

上面那段URL是一个（但是一行里放不下）。输入这个URL以后一个被分为60%、40%；250x100像素的；分别写着“Hello”和“World”的3D饼图就呈现在你眼前了。

刚刚展现的URL是写死的，为了更有通用性，我将向你展示一种从“string、lists、maps、closures、buiders”中获取URL的方式。

目标：用Groovy写一个段脚本来生成这个3D饼图桌面应用。

接下来，我们将讨论：

字符串的操作

Lists 和 maps

用closures获取数据

Groovy的构建者类

一开始我将会写一个简单的小脚本，一点点将它改成最终版本。

### 2.2.1 用查询字符串构建URL

一开始，我们需要一个变量去表达基础URL。在Groovy脚本中实际上你不必定义任何类型。一旦你定义一个类型，这个变量就会变成局部变量。如果没有，这个变量就会是“局部绑定”的（什么是“局部绑定”我们下章再讲）。因为我知道这个URL在我改变它之前一定包含一个字符串，所以我将宣称这个变量为java.lang.String类型

String base = ‘http://chart.apis.google.com/chart?’

Groovy的类型是任选的。这意味着你可以确定一个类型，或者在你不知道或不关心变量类型时可以用一个del关键字去宣称变量。Dierk Koening Groovy in action的作者曾说过：

**用def：** 如果你知道这个类型，就去定义它吧。话句话讲，如果你知道这个变量是String或者Date，或者是一个Employee，就去定义这个变量吧。

以我的经验来看，最开始我用def很多，但随着时间的推移，我越来越少用它。我十分同意Dierk，现在在我用def之前我都会停下来想一会，这个变量是不是有一个确定的类型。但也有很多跟我意见相左的开发者。可变换类型的语言魅力就在于此：它可以容下任何人。

现在我需要为URL构建查询参数了。我用将要用一种这类问题的典型用法来取代将查询参数直接写死在URL中，这种方法就是构建一个map，然后从这个map中生成查询字符串。带着这种思像，这有些map参数。

def params = [cht:’p3’,chs:’250x100’,chd:’t:60,40’,chl:’hello|World’]

在Groovy中，能用一个方括号来建立一个map，每个键值对的键和值分别在冒号的左右两边。键的类型默认为字符串，值可以为任意类型。默认的，变量’params’为java.util.LinkedHashMap类的实例。

容器：Groovy有自带的lists和maps，maps的键默认为string类型的。

每一个对应的值在单引号之间。在Groovy中，在单引号中的字符串是java.lang.String的实例。双引号中的字符串是“被修改过的”字符串，叫做GString。我将给你展现一个例子。

为了将map变成一个查询字符串，我首先要将map中的键值对变成”key=value”形式的字符串，并且随后我要用分隔符将它们串联在一起。第一步就是用collect方法变换（这是一个特别的、在Groovy全部容器类都有的方法）。Collection方法将一个闭包作为参数，这适用于容器中的每一个元素，能返回一个包含结果的新容器。

闭包将会在下一个补充块中被介绍，并且对他的讨论将会贯穿整部书，但目前认为他们的代码块代表一个函数体，这可能需要虚拟参数。当被用作map这个闭包可能会有一两个参数，当只有一个参数的时候，它代表着map中的键值对，当有两个参数时，一个代表key另一个代表value。

为了将map变成key=value的list，我们在collect方法中需要两个参数。

Params.collect{k,v->”$k=$v”}

在Groovy中，如果任何一个方法最后一个参数是闭包，你就可以将这个闭包拿到小括号外面。如果集合的唯一参数是闭包，这样的话圆括号可以省略。

**什么是闭包？**

闭包是一段被括号分隔的代码块，它能被视作一个对象。这个箭头符号用于指定虚参。在当前的例子中，闭包被适用于map，两个虚参分别是k和v（代表着键值对中的key和value）。在箭头右边的表达式代表，将每对key和value带入被‘=’分隔的GString中。Collect方法将每个键值对变成目标形式的字符串，最后产生一个包含结果的list。

操作结果在这里：

[“cht=p3”,”chs=250x100”,”chd=t:60,40”,”chl=Hello|World”]

图2.2说明了这一过程。

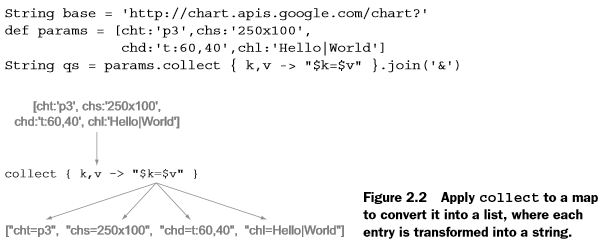
为了构造查询字符串，用一个Groovy加入容器类的方法—---join。这个join方法有一个参数，用来加到相邻两组字符串之间。为了构造查询字符串，调用join伴着一个分隔符作为参数。

[“cht=p3”,”chs=250x100”,”chd=t:60,40”,”chl=Hello|World”].join(‘&’)

我们需要的字符串就出来了：

“cht=p3&chs=250x100&chd=t:60,40&chl=Hello|World”

我们总结一下获得这个URL的过程，用基础的URL，和参数map构建：



现在我们得到的结果其实是一个字符串，而不是URL。在将它变成URL之前，让我们来验证这个程序能否正常工作。正常情况下我们需要测试，在第6章中我们再谈测试。这我们只需要用Groovy中assert这个关键字，他会用到一个布尔表达式参数。如果最后结果是true，什么都不会返回。但如果错了，就会在打印错误报告。如果我用Map中的contains方法，检查每个键值对是否被修改成了合理的形式：

params.each{ k,v ->

assert qs.contains(“$k=$v”)

}

这个assert 关键字： Groovy的断言，是一个简单的判断正确性的方式。如果最后结果是true，什么都不会返回。但如果错了，就会在打印错误报告。

这个join方法好处之一就是你不用担心分隔符被加到了字符串的开始或者结尾。它只会加到字符串中间。

注意：有一种情况括号（在join方法中）是必要的。在Groovy中，如果你调用一个函数，它没有参数，而且你没加括号，编译器就会假设你正在调用getter或setter方法。如果你想用join方法（而不是调用getJoin()，它也不存在），你需要这个括号。

### 2.2.2 发送URL

这个Groovy的JDK在String类中加入了toURL()的方法。和你想象的一样，这个方法可以将java.lang.String的实例对象转化成java.net.URL的实例对象。

**Groovy的JDK**

Groovy在现有的java类库中加了很多有用的方法。一次次，当我发现这些被添加在String、Date或Collection中的方法是，我多希望它也被加入到了原生java中。Groovy的JDK就是这些被添加的方法集合，它们有他们自己的JavaDoc。Groovy的JDK文档在Groovy的官网上可以下载。

Groovy的JDK更多细节将在第3章被讨论。

为向这个URL发送HTTP GET请求，并且获取结果，将字符串变换成URL。就得调用Groovy的JDK中领一个方法----------getText()方法（被加入到了java.net.URL之中）。换句话说，一个网页的数据能通过url.toURL().text方式获取。

在这里我故意去用.text这种写法，因为在Groovy中它会去调用getText()方法。在这里调用getText()方法没有任何错误。但是之前那种是更加地道的Groovy写法。

我们代码如我们所料的运行了。在本章中我通过一些例子，向大家展示了一些web服务的技术。但有些特殊的时候，如果我们通过Google图表获取的信息是个图片，那么将它变成文本就是没意义的。

**Groovy的属性**

访问一个类的属性时，Groovy中自动会调用相关的getter、setter方法。

接下来我将要构建一个图形用户接口，它将包括在javax.swing.ImageIcon的图片。我将要给大家举一个“构建者”的例子，这是一个使你加深对于Groovy中元编程理解极好的例子。

### 2.2.3 用SwingBuilder创建一个UI界面

在Groovy中每一个类都有一个元类。一个元类是实际调用进程的类。如果你调用一个类不存在的方法，这个调用最终会被在元类中叫methodMissing的方法拦截。类似的，如果你访问了一个不存在的属性，同样会被一个叫propertyMissing的方法拦截。订制methodMissing和propertyMissing的行为是Groovy元编程的核心。

Groovy元编程是一个很大的课题，但在这我就将展示其中一个有用的方面：创建一个构建者类。在构建者类中，如果调用methodMissing将会对这种类型的构建者，做出特别的行为。

在这里我将举例说明一个Swing构建者。这个类要将组成和构造Swing用户接口的名称拦截在外。这事实上演示要比解释更容易。在我们开始之前在引入几个Google图表的脚本：

import java.awt.BorderLayout as BL

import javax.swing.WindowConstants as WC

import groovy.swing.SwingBuilder

import javax.swing.ImageIcon

**自动引入：**

你也许注意到了，我刚刚没有进行任何引入操作。Java会自动引入java.lang这个包。Groovy会自动引入java.util, java.io, java.net, groovy.lang, groovy.util, java.math.BigInteger还有java.math.BigDecimal以及java.lang。

这段脚本中，我们将从java标准库中引入三个类。前两个引入用了as操作给了各自类一个别名。就是当用到BorderLayout和WindowConstants时，可以用BL和WC取代。我也加入了ImageIcon类，它将用来捕获Google图表传回来的图片。从Groovy类库中引入的SwingBuilder将要负责Swing UI的创建。

as 关键字：这个as关键字有几种用法，其中之一就是为引用提供别名。这个as关键字相当于asType方法（它会讲java.lang.Object作为Groovy JDK的一部分加入其中）

在SwingBuilder中，有时你会调用本不存在的方法，但是Swing API 会去调用其他相应的方法。例如当你调用frame方法时，实际上你在实例化JFrame类。给它一个类似于map的参数visible：true 相当于调用setVisible方法并且传了一个true参数。

这有一个用了构建者的代码。每一个方法都不在SwingBuilder中，而是调用在Swing类库中相应的方法:

swingBuilder.edt {

frame(title:'Hello, World!', visible:true, pack: true,

defaultCloseOperation:WC.EXIT\_ON\_CLOSE) {

label(icon:new ImageIcon("$base$qs".toURL()),

constraints:BL.CENTER)

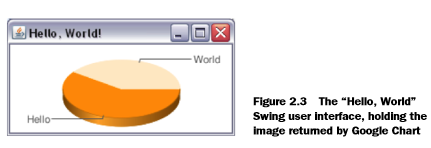
}

}

这个edt方法用分支线程创建了一个GUI。它有一个闭包作为参数，随后有意思的事就开始了。在闭包中第一句是调用frame方法，但事实上在SwingBuilder中没有这个方法。这个构建者的元类就拦截了这个调用并且将它作为javax.swing.JFrame类的实例的一个请求。这个frame方法在这里陈列了一系类map键值对，来用于JFrame的标题，可视化，关闭操作。这个构建者将它理解为JFrame实例setTitle, setVisible, 和setDefaultCloseOperation方法的调用。

在圆括号之后又有一个闭包。它们被理解为JFrame实例的组成成分。下一调用是label方法，这在这里是不存在的。这个Swing构建者知道要创建一个JLabel作为结果，调用它的setIcon方法用从Google图表返回的图片new一个ImageIcon对象，并将JLabel放在BorderLayout的中央。

最终，在这个frame闭包以后，我调用JFrame中的pack方法，来返回一个能足够装下图片的GUI。下面有完整的代码。



代码清单2.1 用Google图表构建一个3D饼图

import java.awt.BorderLayout as BL

import javax.swing.WindowConstants as WC

import groovy.swing.SwingBuilder

import javax.swing.ImageIcon

def base = 'http://chart.apis.google.com/chart?'

def params = [cht:'p3',chs:'250x100',

chd:'t:60,40',chl:'Hello|World']

String qs = params.collect { k,v -> "$k=$v" }.join('&')

SwingBuilder.edt {

frame(title:'Hello, Chart!', pack: true,

visible:true, defaultCloseOperation:WC.EXIT\_ON\_CLOSE) {

label(icon:new ImageIcon("$base$qs".toURL()),

constraints:BL.CENTER)

}

}

结果如图2.3所示。

本节学到了什么？

1. Groovy的变量是有类型的，但当你不知道或不关心它的类型时 你可用def关键字。这中def类型也能用在方法的返回值类型和参数类型。
2. Groovy对于lists和maps有原生的语法。这里有几个用Groovy map的例子；list的例子还有很多。
3. 闭包像带有参数的方法体。
4. 这个collect方法可以通过对每一个元素提供闭包和返回列表，来生成容器。
5. Groovy的JDK在标准java API中添加了很多方法。
6. Groovy的解释器和构造器可以简单的与众多API工作。

下一个例子是展示Groovy对于xml解释、生成的能力，还有数据库操作，正则表达式，groovlets等。