iRules 101 - #4 - Switch

, 2009-02-02

在前面的文章中，我提到过每个iRules的核心几乎都是IF命令。在这些情况下，如果你需要为某个值执行条件判断，这里还有一个可以在大多数情况下使用的条件命令，它比IF执行起来更加快速并且简单易懂。本文将会讨论“Switch”语句，哪些情况你会需要使用它，以及如何对它进行使用。

语法

Switch命令的语法如下：
switch ?*options*? *string pattern body* ?*pattern body* ...?
switch ?*options*? *string* {*pattern body* ?*pattern body* ...?}
Switch命令会把它的String参数和Pattern参数逐一进行匹配，一旦它找到一个能和String匹配的Pattern，它就通过TCL解释器执行后面的Body部分参数，并返回计算值。如果最后一个Pattern参数是Default，那么它可以匹配所有值。如果没有Pattern参数可以和String进行匹配，并且语句中没有包含Default，那么这个Switch命令就返回一个空的String。

如果Switch后面的第一个参数以”-”开始，那它就会被当作选项。以下是一些当前支持的选项。
-exact
对String和Pattern进行精确匹配，这是默认的选项。
-glob
使用通配符匹配模式（和在”String Match”中使用同样的实现方法）
-regexp
使用正则表达式对字符和Pattern进行匹配（详见**re_syntax**参考页里的描述）
- -
选项结尾标志，它后面的参数都会被当成String来对待，哪怕它的起始字符是一个破折号（－）。

iRules为Pattern和Body提供了两种语法。第一种是为Pattern和命令提供不同的参数;如果某些Pattern或命令需要用到替换时，这种方式是非常方便的。第二种形式将所有的Pattern和命令全都放在同一个参数里;这个参数必须能提供适当的能够包含Pattern和命令基本元素的列表结构。使用第二种形式可以非常简便的创建多行Switch命令，因为包含整个列表的大括号使每行末尾的反斜扛变得不再必要。由于第二种形式中Pattern参数都包含在大括号中，命令和变量的替换都不能在它们上面执行，这导致某些情况下第二种形式的行为会不同于第一种形式。

用“Switch”替代“IF”
正如你在上面的语法中看到的那样，Switch语句执行了一个由String参数表示的值的比较。这也就是说，在执行单String值的“与”和“或”逻辑的时候，Switch是If命令的一个完美的替代品。你在许多的iRules中都可以找到类似的例子

```
if { [HTTP::uri] equals "/foo" } {
  # do something...
} elseif { ([HTTP::uri] equals "/bar") || ([HTTP::uri] equals "/foobar") } {
  # do something else...
} else {
  # don't do anything...
}
```

你在这个例子中可以看到，如果一个HTTP::uri String的值通过了条件，就开始执行这段代码的Body。这是Switch语句的一个完美候选。下面这个例子就等同于上面的If/else语句。

```
switch [HTTP::uri] {
```

```
  "/foo" {
    # do something...
  }
  "/bar" -
  "/foobar" {
    # do something else...
  }
  default {
    # don't do anything...
  }
}
```

还有很重要的一点是，破折号('-')在"/bar"String之后进行比较。这就告诉TCL编译器对后面的那行执行一个逻辑或操作。所以基本上说，比较"/bar"或"/foobar"的是一样的。你或许会发现这两个选项的比较对可读性的提升帮助不大，但是当测试超过两到三个之后，这一点会变得愈发的明显。它还能带来一些潜在的好处：一般来说，Switch命令要比If语句速度更快，这是因为If命令的执行的时候会导致额外的表达式进行取值操作。由于Switch语句只在一个比较值上有效，取值的过程可以进行内部优化。如果只有一两个比较的时候，这点或许体现得不够明显，但是比较的数量越多，区别也就越大。

使用Switch的不同方式

“精确”匹配比较
Switch命令和很多其它你会用到的比较模式相似。运行Switch的“-exact”参数可以完成String与String的精确比对，就类似于两个String之间的"equals"操作符。这里没有通配符或者正则表达式，但它在不使用这些功能的时候可以被高度的优化。上面的例子就是一个选择使用"-exact"的例子（如果没有指定参数，-exact参数是默认的）。

你真的需要正则表达式吗？

我们可以在iRules中经常看到下面的语句：

```
if { [HTTP::uri] matches_regex "/foo*" } {
  # do something
}
```

上面的iRules有一个明显的问题;使用正则表达式会导致资源的过度消耗。正则表达式对CPU的消耗非常大，所以只有当别无选择的时候才应该考虑使用它。至少，当需要判断的内容在条件的执行开始、中间或末尾的时候，你应该把上面的matches_regex替换成"starts_with"或是"ends_with"或是"contains"。在"string"这个TCL命令中包含了子命令"string match"。这允许通配符和字符范围执行一个“可怜的”正则表达式。从Unix的角度来看，这和Unix命令行下的文件名匹配是非常相似的。"string match"的用法如下：

string match ?-nocase? *pattern string*
*****
匹配String里任意的字符序列，也包括空字符。
?
匹配String里任意的单个字符。
[*chars***]**
匹配给定的一组字符中的任意一个。例如该字符中出现x-y的形式的序列，那么就可以匹配在x和y之间的任意一个字符，包括xy本身。当使用**-nocase**的时候，这个范围的两端之间的内容首先被转换成小写字母。然而当大小写铭感匹配{A-z}匹配'_',('_'落在'Z'和'a'之间)的时候，使用**-nocase**的话就相当于{[A-Za-z]}（也就是在前面提到的意思）。
*****x*
匹配单个字符X。这样可以避免Pattern内部***?[]**字符的特殊编译。

如何把它关联到Switch命令呢？幸运的是，"-glob"参数允许基于TCL String匹配比较格式的比较。绝大多数情况下，可以用精心放置的通配符和字符范围来完全取代正则表达式。

```
switch -glob [HTTP::uri] {
  "/foo*" {
    # this will match on any string that starts with "/foo"
  }
  "*bar" {
    # this will match on any string that ends with "bar"
  }
  "/foobar[a-zA-Z]" {
    # This will match with any string that starts with "/foobar" followed by any case character from
  }
 "/foo?bar" {
    # this will match any string that starts with "/foo", followed by any character, and followed by
  }
}
```

就像你看到的那样，在Switch语句中"globbing"选项非常强大而且能涵盖大多数的比较情况。

当匹配比较不能满足需求的时候
当你需要进行更加深入的校验的时候，就可以使用"-regexp"参数。但如果描述正则表达式的细节的话，就超出了这篇文章的范围。所以如果你有兴趣，你可以参阅"re_syntax"的相关文档
(http://www.tcl.tk/man/tcl8.4/TclCmd/re_syntax.htm)。祝你好运
由于我们不推荐使用正则表达式，我将忽略这一部分。它的格式和"-glob"选项几乎是相同的，只是用你想使用的正则表达式替代了比较String。

Switch命令并不像你想象中的那么简单

多个比较源

如果你想用if/else命令来比较多个值，可以采用如下方法：

```
if { ([HTTP::host] equals "www.foo.com") and ([HTTP::uri] starts_with "/foo") } {
  # do something
}
```

```

```

使用Switch语句，你应该这样做：

```
switch [HTTP::host] {
  "www.foo.com" {
    switch -glob [HTTP::uri] {
      "/foo*" {
        # do something
      }
    }
  }
}
```

"逻辑与"比较

通过一个破折号(-)将条件结合在一起的功能使得执行"if (a or b) do c"这样的测试变得非常容易，但是对"if (a and b) do c)"则无能为力。要建立一个“逻辑与”比较，你必须在Switch语句的Body中嵌入一个Switch或是IF。上面的例子很好的说明了这一点。

简单“布尔运算”类型测试

如果命令不仅常被用来测试String比较，还可以测试一些例如“String长度”的命令，或是例如“大于”的数字比较。在Switch语句中是不能使用这样的测试的。但实际上，你可以，但是你必须先完成这个测试，并返回相应的值，然后再把这个值放到一个String格式中，最后将这个String放到Switch语句中。所以说在这样的情况下，你最好还是使用IF语句。

结 论

如果你发现你的iRules中充满了一条又一条的if/elseif/elsef/elseif/命令，那么考虑改用更加高效的"switch"命令吧，它不仅会使iRules运行得更快(因为要进行String比较)，但是更多的时候，它更易于阅读和维护。

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.
Corporate Headquarters
info@f5.com

F5 Networks
Asia-Pacific
apacinfo@f5.com

F5 Networks Ltd.
Europe/Middle-East/Africa
emeainfo@f5.com

F5 Networks
Japan K.K.
f5j-info@f5.com