hujiantao224

浏览: 7141 次

性别:

来自: 深圳

我现在离线

详细资料 留言簿

搜索本博客

搜索

最近访客 >>更多访客

sixme qym0925

lcyangily hrwhat

博客分类

- 全部博客 (54)
- spring (5)
- java综合 (21)
- hibernate (2)
- jquery (1)
- extjs (0)
- struts (3)
- javascript基础 (2)
- 杂谈 (2)

其他分类

- 我的收藏 (10)
- 我的论坛主题贴 (0)
- 我的所有论坛贴 (0)

java规则引擎的原理 | jQuery index

2010-02-24

DRL文件语法规则

文章分类:Java编程

一个典型的DRL文件:

Java代码

```
①package com.sample  //包名，不可以与关键字冲突
②import com.sample.DroolsTest.Message;//本文件需要导入的类
③global java.util.List myGlobalList;//全局变量
④//定义函数体
function String hello(String name) {
   return "Hello "+name+"!";
}

⑤rule "myRule"
no-loop true  //执行一次后，是否能被再次激活
salience 100  //优先级别
⑥when
m : Message( status == Message.HELLO, message : message )
⑦then
m.setMessage( "Goodbye cruel world" );
      m.setStatus( Message.GOODBYE );
      update( m );
   myGlobalList.add( "Hello World" );//使用global 变量
   System.out.println( hello( "Bob" ) );//调用定义函数
End
```

Java代码

```
①package com.sample  //包名，不可以与关键字冲突
②import com.sample.DroolsTest.Message;//本文件需要导入的类
③global java.util.List myGlobalList;//全局变量
④//定义函数体
function String hello(String name) {
   return "Hello "+name+"!";
}

⑤rule "myRule"
no-loop true  //执行一次后，是否能被再次激活
salience 100  //优先级别
⑥when
m : Message( status == Message.HELLO, message : message )
⑦then
m.setMessage( "Goodbye cruel world" );
      m.setStatus( Message.GOODBYE );
      update( m );
   myGlobalList.add( "Hello World" );//使用global 变量
   System.out.println( hello( "Bob" ) );//调用定义函数
End

①package com.sample  //包名，不可以与关键字冲突
②import com.sample.DroolsTest.Message;//本文件需要导入的类
③global java.util.List myGlobalList;//全局变量
④//定义函数体
function String hello(String name) {
   return "Hello "+name+"!";
```

我的精华良好贴 (0)

最近加入圈子

存档

- 2010-08 (1)
- 2010-06 (3)
- 2010-05 (3)
- 更多存档...

评论排行榜

- Hibernate延迟加载机制
- java ClassLoader
- datetimepicker struts2 标签
- Junit 4
- java格式化format

RSS
Add to Google
订阅到 鲜果
[什么是RSS?]

```
}

⑤rule "myRule"
no-loop true  //执行一次后，是否能被再次激活
salience 100  //优先级别
⑥when
m : Message( status == Message.HELLO, message : message )
⑦then
m.setMessage( "Goodbye cruel world" );
m.setStatus( Message.GOODBYE );
update( m );
   myGlobalList.add( "Hello World" );//使用global 变量
   System.out.println( hello( "Bob" ) );//调用定义函数
End
```

①package com.sample
包名，不可以与关键字冲突。一个包通过名称空间描绘，这样很好的保持了一组规则的独立性。
②import
标记就像java中的含义一样。对于任何要用在规则中的对象，你需要指定完整的路径和类型名。Drools从同名的java包中自动导入类。
③global
如果多个包定义了同样名称的全局变量，它们必须使用同样的类型，并且全部指向同一个全局值。全部变量通常用来返回数据，获得提供数据或服务给规则使用。为了使用全局变量，你必须：
在规则文件中声明全局变量并使用它，如：

```
global java.util.List myGlobalList;
rule "Using a global"
when
   eval( true )
then
   myGlobalList.add( "Hello World" );
end
```

在working memory上设置全局变量的值。最好是在将fact插入working memory之前设置完所有全局变量，如：

```
List list = new ArrayList();
WorkingMemory wm = rulebase.newStatefulSession();
wm.setGlobal( "myGlobalList", list );
```

④function
相对于正常的java类，函数是在你的规则代码中放置语言代码的方法。它们不可能做任何超过你可以在帮助类（在java中定义，被设置入规则的Working Memory中的类）中做到的事情。使用函数的优点是可以将逻辑保存在一个地方，并且你可以在需要的时候改变函数（这样做各有优缺点）。函数最大的用处是被规则的推论（then）部分中的行为所调用，特别是当一个行为操作需要反复被调用时（将公用代码抽取出来成为一个函数）。
⑤rule 名称可以在“”下取任何名字。
属性列表：
属性 类型 默认值 功能描述
no-loop Boolean false 设置no-loop为true可以阻止该规则被再次激活。
salience integer 0 优先级数字高的规则会比优先级低的规则先执行。
agenda-group String MAIN 只有在具有焦点的agenda group中的规则才能够激发。
auto-focus Boolean false 如果该规则符合激活条件，则该规则所在agenda-group自动获得焦点，允许规则激发。
activation-group String N/A 在同名activation-group中的规则将以互斥的方式激发
dialect String "java" or "mvel" 指定在LHS代码表达式或RHS代码块中使用的语言。
date-effective String, 包含日期/时间定义 N/A 规则只能在date-effective指定的日期和时间之后激活。
date-exptires String, 包含日期/时间定义 N/A 如果当前时间在date-expires指定的时间之后，规则不能激活。

duration long N/A 指出规则将在指定的一段时间后激发，如果那个时候规则的激活条件还是处于true的情况下。
⑥ LHS (when) 条件元素
为了能够引用匹配的对象，使用一个模式绑定变量如‘$c’。变量的前缀使用的$是可选的，但是在复杂的规则中它会很方便用来区别变量与字段的不同。

```
$c : Cheese( type == "stilton", price < 10, age == "mature" )
```

&& 和|| 约束连接符

```
Cheese( type == "stilton" && price < 10, age == "mature" )
Cheese( type == "stilton" || price < 10, age == "mature" )
```

第一个有两个约束而第二个组有一个约束，可以通过圆括号来改变求值的顺序。
单值约束

Matches 操作

```
Cheese( type matches "(Buffalo)?\S*Mozerella" )
Cheese( type not matches "(Buffulo)?\S*Mozerella" )
```

Contains 操作

```
CheeseCounter( cheeses contains "stilton" )
CheeseCounter( cheeses not contains "cheddar" )
```

memberof操作

```
CheeseCounter( cheese memberof $matureCheeses )
CheeseCounter( cheese not memberof $matureCheeses )
```

字符串约束

字符串约束是最简单的约束格式，将字段与指定的字符串求值：数值，日期，string或者boolean。

```
Cheese( quantity == 5 )
Cheese( bestBefore < "27-Oct-2007" )
Cheese( type == "stilton" )
Cheese( smelly == true )
```

绑定变量约束

变量可以绑定到Fact和它们的字段，然后在后面的字段约束中使用。绑定变量被称为声明。有效的操作符由被约束的字段类型决定；在那里会进行强制转换。绑定变量约束使用'=='操作符，因为能够使用hash索引，因此提供非常快的执行速度。

```
Person( likes : favouriteCheese )
Cheese( type == likes )
```

返回值约束

返回值约束可以使用任何有效的Java元数据类型或对象。要避免使用任何Drools关键字作为声明标识。在返回值约束中使用的函数必须返回静态常量（time constant）结果。之前声明的绑定可以用在表达式中。

```
Person( girlAge : age, sex == "F" )
Person( age == ( girlAge + 2) ), sex == 'M' )
```

复合值约束

复合值约束用在可能有多个允许值的时候，当前只支持'in' 和'not in'两个操作。这些操作使用圆括号包含用逗号分开的值的列表，它可以是变量，字符串，返回值或限定标识符。'in' 和'not in'运算式实际上被语法分析器重写成多个!= and ==组成的多重约束。

```
Person( $cheese : favouriteCheese )
Cheese( type in ( "stilton", "cheddar", $cheese )
```

多重约束

多重约束允许你对一个字段通过使用'&&' 或者'||'约束连接符进行多个约束条件的判断。允许使用圆括号分组，它会让这种约束看起来更自然。

```
Person( age ( (> 30 && < 40) || (> 20 && < 25) ) )
Person( age > 30 && < 40 || location == "london" )
```

内联的Eval约束

eval约束可以使用任何有效的语言表达式，只要它最终能被求值为boolean元数据类型。表达式必须是静态常量（time constant）。任何在当前模式之前定义的变量都可以使用，自动代入（autovivification）机制用来自动建立字段绑定变量。当构建器发现标识不是当前定义的变量名是，它将尝试将它作为对象的字段来访问，这种情况下，构建器自动在inline-eval中建立该字段的同名变量。

```
Person( girlAge : age, sex = "F" )
Person( eval( girlAge == boyAge + 2 ), sex = 'M' )
```

⑦RHS (then) 执行操作

这部分应当包含一系列需要执行的操作。规则的RHS部分应该保持简短的，这保持它是声明性和可读性的。如果你发现你需要在RHS中使用命令式或and/or条件代码，那你可能需要将规则拆分为多个规则。RHS的主要目的是插入，删除修改working memory数据。

"update(object, handle);" 将告诉引擎对象已经改变（已经被绑定到LHS中的那一个），并且规则需要重新检查。

"insert(new Something());" 将在working memory中放置一个你新建的对象。

"insertLogical(new Something());" 与insert类似，但是当没有更多的fact支持当前激发规则的真值状态时，对象自动删除。

"retract(handle);" removes an object from working memory.

⑧ Query

查询中仅仅包含规则LHS部分的结构（不用指定when或then）。它提供了查询working memory 中符合约束条件的对象的一个简单办法。

```
query "people over the age of 30"
   person : Person( age > 30 )
end
```

通过在返回的查询结果(QueryResults)上进行标准的for循环遍历，每一行将返回一个QueryResult，该对象可以用来存取组元中的每一个Column。这些Column可以通过声明的名称或索引位置存取。

```
QueryResults results = workingMemory.getQueryResults( "people over the age of 30" );
for ( Iterator it = results.iterator; it.hasNext(); ) {
   QueryResult result = ( QueryResult ) it.next();
   Person person = ( Person ) result.get( "person" );
   }
```

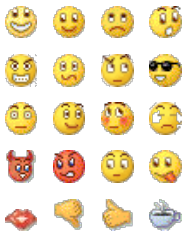
java规则引擎的原理 | jQuery index

23:00 | 浏览 (367) | 评论 (0) | 分类: java综合 | 相关推荐

评论

发表评论

表情图标



B I U Quote Code List Img URL Flash Table

字体颜色: 标准 字体大小: 标准 对齐: 标准

提示：选择您需要装饰的文字, 按上列按钮即可添加上相应的标签

您还没有登录，请登录后发表评论(快捷键 Alt+S / Ctrl+Enter) 提交