[**[原创]JBPM源码解读之：Join**](http://www.blogjava.net/ycyk168/archive/2008/11/14/240616.html)

       前面我有一篇[《JBPM源码解读之：Fork》](http://www.blogjava.net/ycyk168/archive/2008/11/05/238844.html)，大致分析了JBPM对于Fork的实现方式，其实Fork和Join是不可分割的一对，Fork实现分拆，Join实现汇集。先让我们看一下《JBPM 3.2.3 User Guide》中关于Join的描述：  
 The default join assumes that all tokens that arrive in the join are children of the same parent. This situation is created when using the fork as mentioned above and when all tokens created by a fork arrive in the same join. A join will end every token that enters the join. Then the join will examine the parent-child relation of the token that enters the join. When all sibling tokens have arrived in the join, the parent token will be propagated over the (unique!) leaving transition. When there are still sibling tokens active, the join will behave as a wait state.  
     下面就让我们从JBPM源码的角度分析一下Join是如何关闭每一个到达它的Token，是如何检查各个子Token与父Token的父子关系，又是如何重新激活父Token实现流程的继续流转，更重要的也是我写这篇文章的原因：我们如何能从Join默认的实现方式中得到启发，让我们能够更好的理解JBPM整个运转流程，更好的驾驭整个项目。  
  
 我们来看Join类的成员变量:

 1http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    /\*\* \*//\*\*  
 2http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* specifies wether what type of hibernate lock should be acquired. null  
 3http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* value defaults to LockMode.Force  
 4http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif     \*/  
 5http://www.blogjava.net/Images/OutliningIndicators/None.gif    String parentLockMode;  
 6http://www.blogjava.net/Images/OutliningIndicators/None.gif  
 7http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    /\*\* \*//\*\*  
 8http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* specifies if this joinhandler is a discriminator. a descriminator  
 9http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* reactivates the parent when the first concurrent token enters the join.  
10http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 注：  
11http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif     \*/  
12http://www.blogjava.net/Images/OutliningIndicators/None.gif    boolean isDiscriminator = false;  
13http://www.blogjava.net/Images/OutliningIndicators/None.gif  
14http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    /\*\* \*//\*\*  
15http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* a fixed set of concurrent tokens.  
16http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif     \*/  
17http://www.blogjava.net/Images/OutliningIndicators/None.gif    Collection tokenNames = null;  
18http://www.blogjava.net/Images/OutliningIndicators/None.gif  
19http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    /\*\* \*//\*\*  
20http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* a script that calculates concurrent tokens at runtime.  
21http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif     \*/  
22http://www.blogjava.net/Images/OutliningIndicators/None.gif    Script script = null;  
23http://www.blogjava.net/Images/OutliningIndicators/None.gif  
24http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    /\*\* \*//\*\*  
25http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* reactivate the parent if the n-th token arrives in the join.  
26http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif     \*/  
27http://www.blogjava.net/Images/OutliningIndicators/None.gif    int nOutOfM = -1;

      parentLockMode用于控制Hibernate的锁机制，这点我们暂且不去深究，有意思的是下面几个变量：isDiscriminator、tokenNames、script、nOutOfM共同决定了Join节点内被的具体行为，本文稍后会具体解说。

1http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gifpublic void read(Element element, JpdlXmlReader jpdlReader) http://www.blogjava.net/Images/dot.gif{  
2http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        String lock = element.attributeValue("lock");  
3http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        if ((lock != null) && (lock.equalsIgnoreCase("pessimistic"))) http://www.blogjava.net/Images/dot.gif{  
4http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            parentLockMode = LockMode.UPGRADE.toString();  
5http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
6http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif    }

       Join的read方法很简单，只是起到了读取JPDL中对lock的配置，并没有关心其他成员变量的初始化，这也就直接说明了isDiscriminator、tokenNames、script、nOutOfM这几个变量均属于运行期，也就是我们没有办法在配置文件里头像Fork一样配置Script，就算配了Join也不认。  
 execute(ExecutionContext executionContext)方法是每个继承自Node的类的核心方法，Join类也是在这个方法中实现了Join的控制机制。在解读这个方法之前必须应该明白的一件事是：Join的execute方法就像Fork的Node-Leave 事件一样是会执行多次的，同样这取决于与Join搭配的Fork具体产生了几个子Token。

 1http://www.blogjava.net/Images/OutliningIndicators/None.gif      //取得Token  
 2http://www.blogjava.net/Images/OutliningIndicators/None.gif        Token token = executionContext.getToken();  
 3http://www.blogjava.net/Images/OutliningIndicators/None.gif        //得到Token的isAbleToReactivateParent属性  
 4http://www.blogjava.net/Images/OutliningIndicators/None.gif        boolean isAbleToReactivateParent = token.isAbleToReactivateParent();  
 5http://www.blogjava.net/Images/OutliningIndicators/None.gif  
 6http://www.blogjava.net/Images/OutliningIndicators/None.gif        //如果当前Token没有结束，则结束该Token，到这里Fork产生的子Token的生命周期也就结束了  
 7http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif        if (!token.hasEnded()) http://www.blogjava.net/Images/dot.gif{  
 8http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            token.end(false);  
 9http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif        }  
10http://www.blogjava.net/Images/OutliningIndicators/None.gif        //检查该Token是否具有激活父Token的能力，如果有方法继续，如果没有方法结束。  
11http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif        if (isAbleToReactivateParent) http://www.blogjava.net/Images/dot.gif{  
12http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            http://www.blogjava.net/Images/dot.gifhttp://www.blogjava.net/Images/dot.gif  
13http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif        }

  Join的这种处理方式，让我们可以方便的实现一种生活中经常用到的抄送机制，例如：在某个流程中有一个审批的环节，这个审批的默认执行人为李副处长，既定情况下，如果这位李副处长审批完毕，流程就应该继续，但是现在王正处长要求所有审批过的文件都要自己亲自过目，但是只是过目，王处长的这个“过目”的行为要求并不会影响流程的运行，意思就是说，王处长完全有可能在流程都已经结束过了才去过目。稍后我会另有文章具体介绍我的使用Fork+Join实现抄送的思路。  
  继续，呵呵，接下我们分析的方法都是在if (isAbleToReactivateParent)块内的，刚才我们说如果isAbleToReactivateParent == false那么整个方法就结束了。

 1http://www.blogjava.net/Images/OutliningIndicators/None.gif            // the token arrived in the join and can only reactivate  
 2http://www.blogjava.net/Images/OutliningIndicators/None.gif            // the parent once  
 3http://www.blogjava.net/Images/OutliningIndicators/None.gif            token.setAbleToReactivateParent(false);  
 4http://www.blogjava.net/Images/OutliningIndicators/None.gif            总感觉这句是多余的，因为好像一个子Token只有一次机会进入Join节点，然后他的生命周期也就结束了，再者在execute方法的后面有  
 5http://www.blogjava.net/Images/OutliningIndicators/None.gif                    Iterator iter = parentToken.getChildren().values().iterator();  
 6http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                    while (iter.hasNext()) http://www.blogjava.net/Images/dot.gif{  
 7http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        ((Token) iter.next()).setAbleToReactivateParent(false);  
 8http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif                    }  
 9http://www.blogjava.net/Images/OutliningIndicators/None.gif            略去关于处理Hibernate锁机制的代码，因为它并不直接影响这个流程的运作.  
10http://www.blogjava.net/Images/OutliningIndicators/None.gif              
11http://www.blogjava.net/Images/OutliningIndicators/None.gif                boolean reactivateParent = true;  
12http://www.blogjava.net/Images/OutliningIndicators/None.gif  
13http://www.blogjava.net/Images/OutliningIndicators/None.gif                // if this is a discriminator  
14http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                if (isDiscriminator) http://www.blogjava.net/Images/dot.gif{  
15http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // reactivate the parent when the first token arrives in the  
16http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // join. this must be the first token arriving because  
17http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // otherwise  
18http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // the isAbleToReactivateParent() of this token should have  
19http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // been false  
20http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // above.  
21http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    reactivateParent = true;  
22http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
23http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // if a fixed set of tokenNames is specified at design  
24http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // timehttp://www.blogjava.net/Images/dot.gif  
25http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                } else if (tokenNames != null) http://www.blogjava.net/Images/dot.gif{  
26http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // check reactivation on the basis of those tokenNames  
27http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    reactivateParent = mustParentBeReactivated(parentToken, tokenNames.iterator());  
28http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
29http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // if a script is specified  
30http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                } else if (script != null) http://www.blogjava.net/Images/dot.gif{  
31http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
32http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // check if the script returns a collection or a boolean  
33http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    Object result = null;  
34http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    try http://www.blogjava.net/Images/dot.gif{  
35http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        result = script.eval(token);  
36http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    } catch (Exception e) http://www.blogjava.net/Images/dot.gif{  
37http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        this.raiseException(e, executionContext);  
38http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                    }  
39http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // if the result is a collection  
40http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    if (result instanceof Collection) http://www.blogjava.net/Images/dot.gif{  
41http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        // it must be a collection of tokenNames  
42http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        Collection runtimeTokenNames = (Collection) result;  
43http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        reactivateParent = mustParentBeReactivated(parentToken, runtimeTokenNames.iterator());  
44http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
45http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        // if it's a booleanhttp://www.blogjava.net/Images/dot.gif  
46http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    } else if (result instanceof Boolean) http://www.blogjava.net/Images/dot.gif{  
47http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        // the boolean specifies if the parent needs to be  
48http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        // reactivated  
49http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        reactivateParent = ((Boolean) result).booleanValue();  
50http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                    }  
51http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
52http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // if a nOutOfM is specified  
53http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                } else if (nOutOfM != -1) http://www.blogjava.net/Images/dot.gif{  
54http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
55http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    int n = 0;  
56http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // wheck how many tokens already arrived in the join  
57http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    Iterator iter = parentToken.getChildren().values().iterator();  
58http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    while (iter.hasNext()) http://www.blogjava.net/Images/dot.gif{  
59http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        Token concurrentToken = (Token) iter.next();  
60http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                        if (this.equals(concurrentToken.getNode())) http://www.blogjava.net/Images/dot.gif{  
61http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                            n++;  
62http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                        }  
63http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                    }  
64http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    if (n < nOutOfM) http://www.blogjava.net/Images/dot.gif{  
65http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        reactivateParent = false;  
66http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                    }  
67http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
68http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // if no configuration is specified..  
69http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                } else http://www.blogjava.net/Images/dot.gif{  
70http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // the default behaviour is to check all concurrent tokens  
71http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // and reactivate  
72http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // the parent if the last token arrives in the join  
73http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    reactivateParent = mustParentBeReactivated(parentToken, parentToken.getChildren().keySet().iterator());  
74http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif                }  
75http://www.blogjava.net/Images/OutliningIndicators/None.gif  
76http://www.blogjava.net/Images/OutliningIndicators/None.gif                // if the parent token needs to be reactivated from this join  
77http://www.blogjava.net/Images/OutliningIndicators/None.gif                // node  
78http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif                if (reactivateParent) http://www.blogjava.net/Images/dot.gif{  
79http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
80http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // write to all child tokens that the parent is already  
81http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // reactivated  
82http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    Iterator iter = parentToken.getChildren().values().iterator();  
83http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif                    while (iter.hasNext()) http://www.blogjava.net/Images/dot.gif{  
84http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                        ((Token) iter.next()).setAbleToReactivateParent(false);  
85http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif                    }  
86http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
87http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // write to all child tokens that the parent is already  
88http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    // reactivated  
89http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    ExecutionContext parentContext = new ExecutionContext(parentToken);  
90http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                    leave(parentContext);  
91http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif                }  
92http://www.blogjava.net/Images/OutliningIndicators/None.gif

    读了这段逻辑很少，但是注释很多的代码相信已经明白了Join内部的核心了，有一种豁然开朗的感觉，JBPM的设计思想真的很迷人：几句简单的If、Else  
 就实现了让人看来很神秘的功能。这个时候我们可以来分析一下他的几个成员变量的作用了。  
   如果isDiscriminator为True，这个时候Join节点其实是起到一种选择器的作用：当第一个子Token到达Join之后，Join就会马上取消其他子Token执行自身execute方法的能力，而且流程会马上继续而不会再理会其他的子Token有没有到达Join或结束，因为根据我们上面的分析，isDiscriminator被设置为false的子token是不具备激活父Token的能力的。原来实现网上经常争论的用Join实现多选一是那么简单（呵呵）！  
   下一个有意思的是nOutOfM，代码写的很明白，当子Token到达Join的时候n就加1，如果n<nOutOfM Join就一个处于等待状态，直到n > nOutOfM 流程马上继续，这就Join实现多选多的机制，真的很简单，当然如果我们把nOutOfM设置为1，那么他所起到的作用就跟isDiscriminator一样了。  
   在说明其他两个变量之前，让我们先来看一下public boolean mustParentBeReactivated(Token parentToken, Iterator childTokenNameIterator)方法

 1http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    /\*\* \*//\*\*  
 2http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif     \* 检查有没有子Token的isAbleToReactivateParent为真，如果有一个子Token的isAbleToReactivateParent为真，就返回false  
 3http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif     \*/  
 4http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedBlock.gif    public boolean mustParentBeReactivated(Token parentToken, Iterator childTokenNameIterator) http://www.blogjava.net/Images/dot.gif{  
 5http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        boolean reactivateParent = true;  
 6http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif        while ((childTokenNameIterator.hasNext()) && (reactivateParent)) http://www.blogjava.net/Images/dot.gif{  
 7http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            String concurrentTokenName = (String) childTokenNameIterator.next();  
 8http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
 9http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif            Token concurrentToken = parentToken.getChild(concurrentTokenName);  
10http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif  
11http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockStart.gifhttp://www.blogjava.net/Images/OutliningIndicators/ContractedSubBlock.gif            if (concurrentToken.isAbleToReactivateParent()) http://www.blogjava.net/Images/dot.gif{  
12http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                log.debug("join will not yet reactivate parent: found concurrent token '" + concurrentToken + "'");  
13http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif                reactivateParent = false;  
14http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
15http://www.blogjava.net/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
16http://www.blogjava.net/Images/OutliningIndicators/InBlock.gif        return reactivateParent;  
17http://www.blogjava.net/Images/OutliningIndicators/ExpandedBlockEnd.gif    }

    isDiscriminator和nOutOfM实现了多选一和多选多，但是这两个变量起到的控制作用是死的，也就是我们并不能指定从Fork那里产生的子Token中哪几个到Join之后流程可继续。例如Fork创建了A、B、C、D、E五个Token，如果用前面那两个控制机制，这五个哪几个到Join之后流程会继续完全是不可控的，如果我们要实现必须是A、C、E到达Join之后流程才可继续，这样的需求用isDiscriminator和nOutOfM是无法实现的。别着急，tokenNames是专为解决这个问题而设置的，我们选定几个子Token塞给tokenNames，那么Join就会自动为我们做这些事情了。  
 相比tokenNames Script为我们提供了更灵活的控制机制。如果Script返回Collection类型，那么Script起到了tokenNames的作用，如果返回Bolean类型，那么Script起到isDiscriminator的作用，源码上注释的很清楚，我就不在这罗嗦了。  
     到这Join的代码我们也就读完了，如果上述四个成员变量都为默认值，那么Join也就按默认的行为执行，Join是JBPM源码中少数注释很全的类，这也说明这是JBPM开发组的得意之作。通过四个属性我们可以非常灵活的使用Join实现很多实用的效果。

          文章原创，转载请注明出处！