第33卷 第15期 计 算 机 工 程 2007年8月
Vol.33 No.15 Computer Engineering August 2007
·软件技术与数据库· 文章编号：1000—3428(2007)15—0103—03 文献标识码：A 中图分类号：TP311

一种基于新型事件驱动机制的Java构件交互方法

羌翼亭，陈昊鹏
(上海交通大学软件学院，上海 200030)

摘 要：提出了一种基于新型事件驱动机制的构件交互方法。这种方法使用统一的事件模型，以一致的方式处理各类事件，包括自动触发的自动事件，显式地调用事件服务API生成的手动事件以及反映事件路由器工作状态的路由事件。这种方法不要求修改构件源代码，且能够以灵活的事件路由器网络作为构件交互的核心，为Java构件交互问题给出了一个新的解决途径。

关键词：事件；构件交互；路由；自动触发；Java；容器；选择

New Event-driven Approach for Java Component Interaction

QIANG Yi-ting, CHEN Hao-peng
(School of Software, Shanghai Jiaotong University, Shanghai 200030)

【Abstract】 A new event-driven approach for interaction among Java components assembled is presented. This approach has an unified event model, which treats kinds of events with same way, including events triggered automatically, events created explicitly using API, and routing events from event router that reflects router state. It doesn't need source code modification or taking flexible router network as interaction core. It is thought as a new way to solve component interaction problem.

【Key words】 event; component interaction; route; automatically-triggered; Java; container; selection

1 概述

现有的构件交互技术基于消息传递的原理，消息发送者和接收者构件必须使用消息服务接口来发送和接收消息。相关的工业标准包括JMS和CORBA事件服务等，相应的产品也已趋向成熟，例如IBM的MQSeries和WBI[1]。另外，国内也有类似研究(如文献[2]论述的一种分布式的异步事件处理框架DAEHF)，它们都利用消息的异步能力和松散耦合能力，作为集成异构系统的一种解决方法。可以将面向消息的中间件看作是围绕着事件机制所做的进一步强化和扩展，通过提供安全、事务和持久服务，使得消息提供者和消息消费者之间的耦合更松散和可靠。但是现有这些构件交互技术和产品，包括理论研究[3~5]在内，它们的一个共同特点是基于预先定义的消息服务接口，由消息发送者/接收者使用该接口显式地发送/接收消息。其缺点在于，显式调用代码的存在增加了额外的工作量和出错几率，而且因为这通常是通过修改/生成源代码来实现，一来难以做到二进制兼容，以方便对既存二进制构件的复用，二来也因为相关源代码被强制加入了事件/消息支持代码，不易做到只为需要事件/消息支持的构件实例引入基于事件/消息的构件交互支持。所以它们都缺乏将事件/消息交互机制无缝地、透明地与实际应用结合的具体方法，进一步导致的问题是，使用的复杂化也不利于引入一个适合于各种规模的应用的简单的、统一的模型。

本文提出的事件/消息机制具有以下几个区别于现有事件/消息机制的特点：

(1)引入自动事件的概念，透明地、无缝地引入事件支持机制。事件触发代码在构件装载时/运行时被插入到构件的二进制执行代码中、在调用构件方法时触发自动事件，并且，通过使用 XML 等方式配置事件，使得事件定义独立于源代码和二进制代码，不再要求开发者显式地调用特定的消息/事件服务API，从而消除了对具体事件服务API的依赖。

(2)事件运行时支持系统使用一个简单的、统一的事件模型，以一致的方式处理各种类型的事件。无论是自动事件，还是显式地调用事件服务 API 生成的事件(以下称为手动事件)，或者是事件路由器生成的反映事件路由器工作状态的路由事件，运行时支持系统都不加区分，一致处理，再加上结合使用构件容器，运行时支持系统可以被无缝地被嵌入在应用系统中。

(3)基于(2)提出的模型，事件运行时支持系统能够提供可嵌套的、支持状态的、具有灵活的路由机制的事件路由服务。事件路由器负责连接触发者构件和接收者构件，在构件之间传递事件；同时路由器自身也是构件，发出的路由事件和其他事件一样，既可以被非路由器构件接收(可能进一步触发新的事件)，也可以被其他路由器接收和转发，从而形成一个事件路由器网络，构件连接在该事件路由网络上，通过发送和接收事件进行交互，但构件自身却无需意识到路由网络的存在与否，路由网络负责调度在事件触发构件和接受构件间传递的事件，成为构件交互的核心，自动事件的引入凸显了消息与事件的区别：“消息”是显式主动发送而非被动触发，使用“事件”能够突出“触发”概念。因此，本文统一用“事件”来表述。

2 统一的事件定义和模型

本文使用以下符号约定来给出统一的事件定义：

$$E(Id, Ps(Cat, Src, Obj, Ts, Ag, Rz, Ex, Ctx, Fg, CasE, Eps(Pr+))) \quad (1)$$

其中：E是事件(Event)；Id表示唯一性的ID；Ps是属性集合。已预定义的一些属性包括类别Cat、触发点Src、触发者构件

作者简介：羌翼亭(1975－)，男，硕士研究生，主研方向：构件技术；陈昊鹏，博士、讲师

收稿日期：2006-08-07 **E-mail：**suntoech@sh163.net

实例 Obj、触发时间 Ts、参数 Ag、结果 Rz、异常 Ex、上下文 Ctx 和级联事件 CasE 等，也可以在扩展属性集 Eps 中自定义额外的属性 Pr。

事件服务处理的是用式(1)统一描述的各种事件，执行事件路由时，并不区分事件类型，也不关注具体事件是自动事件还是手动事件或路由事件，而是以一致的方式处理各种事件的串/并联、逻辑(与/或/非)组合、时序组合、级联/中间事件定义、同步/异步、队列、存储/转发和 Multicasting。

基本的事件模型仍沿用 ECA 定义，其中，C(Condition)可以被扩展至 Routing，以更准确地表示从单一的条件选择机制到可嵌套的支持状态的路由机制；而在构件环境中，A(Action)必然是一个或多个接收者构件的某个方法。

3 自动事件

3.1 自动事件的触发模型

自动事件是包含事件触发点(Src)的方法被调用时自动生成的事件。自动事件同手动事件一样符合式(1)对事件的定义，并按 Src 区分出 4 个类别(Cat)：

(1)“进入”事件(Enter)：方法被调用后，在进入方法体前触发；

(2)“离开”事件(Leave)：方法体执行结束，且无异常发生时触发；

(3)“异常”事件(Exception)：方法执行过程中抛出异常时触发；

(4)“完成”事件(Final)：离开方法体之后，结束方法调用之前触发，且无论之前有无异常抛出都将触发；Final 事件在 Leave 事件或 Exception 事件后被触发。

3.2 自动事件触发模型的实现及对构件运行性能的影响

依据 3.1 节对自动事件及其触发模型的定义，可用 Java 伪代码来描述自动事件触发的基本实现：

```
try{
        触发 Enter 事件；
        if(响应 Enter 事件时指定了 Rz)   return Rz；
        执行方法体
        触发 Leave 事件；
        return Rz；
}catch(Exception Ex){
        触发 Exception 事件；
        if(响应 Exception 事件时 Ex 被抑制)  return Rz；
        throw Ex；
}finally{
        触发 Final 事件；
        if(响应 Final 事件时 Ex 被抑制)        return Rz；
        if(响应 Final 事件时 Ex 被重置)        throw Ex；
        }
```

具体实现运行时插入自动事件支持代码时，可以利用已有的(动态)AOP 机制，在 around Advice 中实现触发事件，也可以直接实现字节码的运行时插入和修改。本文的试验为考察自动事件对构件运行性能的影响、尽量排除其他因素干扰而选择了后一种方式。试验结果表明，一方面，以 int 型的 Java Bean Setter 方法为测试对象时，对于一个直接来自于上述模型的实现，即使所有事件响应方法都为空，方法执行时间也至少增加 40 倍；在采取了一些优化措施以后，则可以降到 10 倍以下，而如果做到不触发不必要的事件，更能够将性能损失控制在 10%左右；另一方面，对于计算 Fibonacci(50)这样较耗时的方法，无论触不触发事件，性能损失都已相对可以忽略了。

4 事件的路由

简单的、统一的事件模型使得引入一个适合于各种规模应用的事件运行时支持系统也变得容易，这样的事件运行时支持系统的核心是事件路由，不同复杂程度的事件路由网络支持不同规模的应用。

4.1 事件路由器功能及其结构

事件路由器是事件运行时支持系统执行事件路由功能的基本功能单元。事件路由器连接触发者构件和接收者构件，以指定规则将触发者构件产生的事件转发给接收者构件。

事件路由器由 1 个或多个事件接收端(前端)，1 个事件选择子(连接点)和 1 个或多个事件发送端(后端)共 3 部分组成。事件从前端进入，经连接点到达后端，继而被发送到接收者构件。这样就能支持 1 对 1、1 对多和多对多等组合关系。

事件接收端或者是接收指定事件并直接传递给事件选择器，或者自身就是一个事件路由器(称为子事件路由器)。每个事件接收端“关注”构件的 1 个/组触发点(Src)，多个事件接收端并联工作，将来自多个触发点的事件传递到事件选择子。根据未被传递到发送端时是否保存接收到的事件，可以将其分为有状态和无状态。

事件选择子 fan-in 从前端进来的事件并 fan-out 至后端的事件发送端。按转接过程中是否计算转接条件，可以分为直通选择子和条件选择子：直通选择子直接传递事件到后端发送出去，而条件选择子只在条件成立时才导致传递事件(传递的事件不再是由前端进来的原事件，而是代表了该次选择动作的 select 路由事件，原事件作为级联事件保存在 CasE 属性中)。条件选择运算分为布尔运算和算术运算两类，例如 AND(与)、OR(或)和 NOT(非)等布尔运算，以及=、>和<等算术运算，运算的对象是事件的各个属性，对于布尔运算，缺省作为运算对象的属性是触发标志(Fg)，对于算术运算，缺省是触发时间(Ts)。

事件发送端负责将来自事件选择子的事件发送给接收者构件。不同的事件发送端具有不同的控制转发过程的方式，例如同步还是异步，立即转发还是延时/定时转发，如何执行 QoS，如何存储事件，如何执行事件统计等。转发目标的范围也可以不限于同一 JVM：事件因此可以被发送给远程接收者，例如远程构件或独立的事件/消息服务器。

4.2 复合事件路由器和路由器网络

事件路由器自身也是构件，因此路由器发出的路由事件和其他事件一样，既可以被非事件路由器构件接收处理(可能进一步触发新的事件)，也可以被其他事件路由器接收和转发，从而形成一个事件路由器网络，构件连接在该事件路由网络上，通过发送和接收事件进行交互，事件路由网络因此成了构件交互的核心。

作为前端的事件接收端自身如果也是一个事件路由器(子路由器)，则其所在事件路由器(父路由器)成为一个复合路由器，子路由器的发送端发送出来的事件进入父路由器的前端成为输入。复合事件路由器可以继续被嵌套，组合成一个具有多级子路由器的更复杂的复合路由器--路由器树。复合路由器适用于只具有一个从第 1 级事件接收端到根事件选择子的事件转送的主路径、同时需要针对多级事件中间状态(由路由事件携带)进行不同处理的情况；使用基本路由器(非复合路由器)组成的路由器网络也可以达到此目的，但是因为基本路由器是网状结构，所以不具有一个清晰的事件发送的主路径。

4.3 路由事件

作为构件的事件路由器工作时会产生多种反映其内部工作状态的事件，称为路由事件。Select 路由事件是其中最重要的一种，其他路由事件包括 override、reset、dicard、repeat 和 dropdown，使用式(1)的 Cat 属性来区分路由事件的类别。

在事件接收端一侧，有状态接收端保存的前次事件被进入同一接收端的新事件覆盖时将产生 override 路由事件，而当因同级的其他接收端新进入的事件满足了事件选择子的路由条件，导致同级所有接收端被重置时将产生 reset 路由事件。

对于事件选择子，除了 select 路由事件以外，也会产生其他路由事件。根据条件未满足时是否保存当前事件的状态，条件选择子可细分为有状态、同步无状态和异步无状态 3 种。对于有状态条件选择子，选择条件未满足时当前事件在一定时间内被保存在指定存储区域中供下次条件运算作为运算对象的事件之一；对于同步无状态条件选择子，选择条件未满足时将使执行线程挂起，等待能够导致下次条件运算的新事件的进入；对于异步无状态的事件选择子，选择条件未满足时不会导致执行线程挂起，事件运行时支持系统会代之以使用一个监视线程，等待选择条件满足。3 种条件选择子都可以做超时设置，如果因超时等原因导致事件被丢弃，就触发 discard 路由事件。3 类事件选择子的状态设置为路由提供了事件同步机制，而与之相联系的超时机制及 discard 路由事件反映了事件同步机制的工作状态。

其他的路由事件还包括，如果检测到同一事件多次进入同一事件路由器的重复次数达到或超过了指定上限次数，将触发 repeat 路由事件(通常这用于检测事件的无限循环)；如果某些条件运算后发现该节点已经进入不可能再满足选择条件的状态，则将触发 dropdown 路由事件。

5 总结与展望

本文介绍了一种基于新型事件驱动机制的构件交互方法的相关概念和模型，基于这种事件驱动机制的构件交互方法因为使用统一的事件模型，具有概念一致、无需修改构件源代码和能够使用灵活的事件路由网络来表达复杂的构件交互关系等特点，为构件交互问题给出了一个新的解决途径。

有待开展的可能的工作包括：给出形式化语义以研究验证该事件模型的完备性；探讨该模型在实际系统中的应用；研究基于该事件模型、使用事件模拟基于构件的软件系统的行为；完成基本可用的使用该模型作为交互机制的 Java 构件容器内核和相关支持工具等。

参考文献

1 Brodie A. 公共事件基础架构: 从技术预览到产品发布[Z]. IBM Developer Works, 2005-06-01.

2 王保进, 王志刚, 李明树. 一种用于软件通信体系结构的构件模型[J]. 计算机工程与应用, 2004, 40(31): 25-28.

3 Luckham D C, Vera J. An Event-based Architecture Definition Language[J]. IEEE Transactions on Software Engineering, 1995, 21(9): 717-734.

4 Barrett, Clarke, Tarr, et al. Framework for Event-based Software Integration[J]. ACM Transactions on Software Engineering and Methodology, 1996, 5(4): 378-421.

5 Snoeck M. Event-based Software Architectures[C]//Proc. of OOIS'03 Conference in the Springer. 2003.

6 Haupt M. Towards Event-based Aspect-oriented Runtime Environments[R]. Software Technology Group, Darmstadt University of Technology, Tech. Report: TUD-ST-2002-01, 2002.

7 梅 宏, 陈 锋, 冯耀东. ABC: 基于软件体系结构、面向构件的软件开发方法[J]. 软件学报, 2003, 14(4).

（上接第 102 页）

识库的事实与规则，也可以以运行时得到的对象属性值或值与值的关系作为事实。推理的触发是由事件引起的。

智能 CGF 对象 IntelligentPO 的描述可以用一个九元组表示如下：

IntelligentPO::=<SM , A , X , Y , Op , Ω , Θ , Π , RE>

式中，RE 是智能引擎，它一方面管理知识，在运行时可以动态增加知识库的事实与规则，也可以以运行时得到的对象属性值或值与值的关系作为事实，并根据事件来触发推理过程。

知识表达采用一阶谓词，又称为公式，原子公式是公式。

如果Ξ、ζ是公式，则如下各式都是公式：

$\neg\Xi$

$\Xi \wedge \zeta$

$\Xi \vee \zeta$

$\Xi \rightarrow \zeta$

$\Xi \leftrightarrow \zeta$

如果Ξ是公式，x 是个体变元，则如下各式都是公式：

$\ni x\Xi$

$\forall x\Xi$

知识在公式建模后，由智能引擎自动转化为 HORN 子句，以便于计算机处理。推理采用逆向推理。

7 小结

总之，运用 CGF 对象的建模方法能够使预制更加独立，它的静态结构和行为模式不因环境的改变而改变，以支持最大限度的可重用，使由此产生的软构件具有较好的通用性、自治性和移动性，并可以通过搭积木的方式柔性快捷地进行仿真应用构建。

参考文献

1 Zeigler B P. DEVS Representation of Dynamical Systems: Event-based Intelligent Control[J]. Proceedings of the IEEE, 1989, 77(1): 72.

2 Zeigler B P. Multifacetted Modeling and Discrete Event Simulation [M]. London: Academic Press, 1984.

3 Zeigler B P. Object-oriented Simulation with Hierachical, Modular Models[M]. London: Academic Press, 1990.

4 Statecharts H D. A Visual Formalism for Complex Systems in Science of Computer[J]. Science of Computer Programming, 1987, 8(3): 231.

5 Coleman D, Hayes F, Bear S. Introducing Objectcharts, or How to Use Statecharts in Object Oriented Design[J]. IEEE Transactions on Software Engineering, 1992, 18(1): 9.

6 Berleant D. Combining Qualitative and Quantitative Simulation: In Brief[C]//Proc. of the 2nd Conference on AI, Simulation and Planning in High Autonomy Systems, Cocoa Beach, FL. 1991.

7 Fishwick P A. Hierarchical Reasoning: Simulating Complex Processes over Multiple Levels of Abstraction[D]. University of Pennsylvania, 1986.

8 Fishwick P A. Computer Simulation Modeling: Methodology, Algorithms and Programs[Z]. 1992.