第32卷 第21期 计 算 机 工 程 2006年11月
Vol.32 No.21 Computer Engineering November 2006
·网络与通信· 文章编号：1000—3428(2006)21—0116—03 文献标识码：A 中图分类号：TP393.092

Web Services的状态管理机制研究

吴建斌[1]，吴家铸[2]
(1. 浙江师范大学信息科学与工程学院，金华 321004；2. 国防科技大学计算机学院，长沙 410073)

摘 要：一般情况下Web Services被认为是无状态的，但是在单个服务请求者或者在特定的业务上下文中需要维护状态或资源来实现相互交互，所以有状态Web Services的研究具有重要意义。通过对Web Services状态含义的探讨，分析了现有状态管理方法的利弊，从而提出新的管理机制，并提供了一个实现原型。

关键词：Web Services；状态；有状态资源；会话

Research of Stateful Web Services

WU Jianbin[1], WU Jiazhu[2]
(1. College of Information Science and Engineering, Zhejiang Normal University, Jinhua 321004;
2. Institute of Computer, National University of Defense Technology, Changsha 410073)

【Abstract】Web Services are once described as stateless, but frequently it is necessary to maintain state or resources in the context of single request or business. This makes the research of stateful Web Services important. The paper discusses the meaning of the state of Web Services, and analyses the methods of state management of Web Services. Finally, a new method of state management is put up, and the implementation phototype is provided.

【Key words】Web Services; State; Stateful resource; Session

虽然Web Services在工业界取得了一些成功，但是一直以来，Web Services被认为是无状态的。由于它不能记忆每次调用的结果，这就限制了Web Services操作不能使用前一次调用的结果。另一方面，虽然在绝大多数情况下都是通过诸如HTTP这样的不可靠、无状态的协议来访问Web Services的，但是实际上，Web Services通常可以代表单个服务请求者或者在特定的业务上下文中跨多个交互来维护状态或资源，所以有状态Web Services的研究具有重要意义。

1 Web Services的状态性

什么是Web Services的状态？文献[4]对Web Services中的“状态”所包含的内容作了阐述：

(1)在多次调用之间持久保存的服务内部数据或属性。比如运算类服务，服务需要保存上一次调用时计算的结果以进行下一次运算。

(2)为每个客户所保存的服务内部数据或属性。如银行类服务需要为每个客户保存账户信息。

(3)多个服务之间的依赖性信息。每个服务保存与它相关联的其他服务的地址、状态等信息。这主要用在组合服务应用中。

(4)系统级的服务状态，如服务生命周期状态、系统工作负荷等。这些信息对于Web Services的监视和管理是必要的。它们与Web Services内部逻辑无关，主要是为了管理的目的。

这个阐述包含了Web Services的所有状态信息。根据Web Services状态的这个阐述，可以从其生命存在的长短，将它们归结为静态状态和动态状态两种类型。静态状态往往在Web Services部署时确定，一般在运行期间不会改变，如对其他组件的引用信息等；而动态状态则会随着每次调用而发生改变。通常所说的有状态Web Services中的“状态”指的就是动态状态。而动态状态一般包括会话状态、Web Services对有状态资源操作的结果等。所以，文献[3]又将有状态Web Services分为两类：一类是会话Web Services，Web Services实例保存前一次操作的执行结果以备下次调用时使用；另一类是作用于有状态资源的Web Services，基于发送和接收的消息可以实现对有状态资源的操作。

2 Web Services的状态管理

目前学术界和工业界对如何实现Web Services的状态表示和管理存在一些争议。Parastatidis[1]认为Web Services本身无状态，要实现Web Services的有状态交互应该通过上下文方法来实现。而文献[2]则认为状态在分布式计算中具有重要的作用，应该从Web Services体系结构这个角度去解决问题。为此，他们提出了WS-Resource框架规范集，试图通过统一的规范实现互操作。实际上，由于两类有状态Web Services的不同状态特性，它们的状态管理机制也应该不同。本节对这两种有状态Web Services分别提出新的状态管理机制。

2.1 作用有状态资源的Web Services

Web Services接口经常提供请求者访问和操作动态状态的能力，如订飞机票等。这类状态一般是持续保存的，而且往往是Web Services之间相互交互的结果。这就是作用于有状态资源的Web Services。对于这类Web Services，OGSI工作组[3]提出“无状态实现，有状态接口”的管理机制，将状态的保存和管理功能委托给其他组件，如数据库或文件系统等。这种管理机制提高了系统的可靠性和可扩展性。但是要实现这种管理机制，主要需要解决的问题是Web Services和状态

作者简介：吴建斌(1976－)，男，硕士、讲师，主研方向：Web Services，网络；吴家铸，副教授

收稿日期：2005-11-28 **E-mail：**wjb@zjnu.cn

之间的关联问题。为此，OGSI工作组提出隐含的资源模式。在这种模式下，它将状态资源标识和Web Services端点地址一起包含在端点引用信息中，然后客户端动态获取端点引用信息，并根据引用信息访问Web Services及其状态。

2.1.1 状态管理模型

隐含的资源模式通过标准化的操作有利于加强 Web Services 的互操作性，但是这种模式是以增加客户端与服务端的通信时间作为代价的。因此，本文提出一种新的实现模型。在这个模型中，假设 S 是一个有状态资源集合，S={s1, s2, …, sn}；设 M 是作用于有状态资源 Web Services 的状态管理模型；设 *c*.first 表示一个客户 *c* 是否首次访问这个 Web Service；设 SOAP(*c*). stateId 为客户 *c* 所产生的 SOAP 消息中包含的状态标志值。则可有如下定义：

定义 1 （有状态资源S）定义为四元组<I, R, V, f>，其中I＝{id1, id2, …, idn}是标识集合，R＝{r1, r2, …, rn}表示状态资源元素集合，$V=V_1 \cup V_2 \ldots V_n$，$V_i$是状态资源元素的值域集合，函数f：I×R→V，定义了标识对应的状态资源值。

定义 2 （作用于有状态资源 Web Services 的状态管理模型 M）定义为<C, W, S, G>，其中 C 表示访问 Web Services 的客户集合，W 表示 Web Services，S 是有状态资源集合，G＝{G1, G2}, G1 表示客户和状态之间的关联规则，G2 表示 Web Services 与状态之间的关联规则。

定义 3 （客户与状态之间的关联规则 G1）可表示如下：

若客户 *c* ∈C,且 *c*.first=true,则 SOAP（*c*）.stateId=0；

若客户 *c* ∈C,且 *c*.first=false,则 SOAP（*c*）.stateId 可从本地获取。

定义 4 （Web Services 与状态之间的关联规则 G2）可以表示如下：

（1）若 SOAP(*c*).stateId=0，则 Web Services 创建状态，执行结束后返回 stateId 给客户；

（2）若 SOAP(*c*).stateId≠0，则 Web Services 可根据 stateId 操作状态。

其具体实现框架如图 1 所示。其中，服务请求者（Service Requestor）是 Web Services 发出请求的客户端；Web Service 端是执行 Web Services 的逻辑组件，主要负责调用 Web Services 和状态；状态管理器（State Manager）是 Web Services 状态的逻辑管理部件，主要作用是创建或载入状态元素对象，并用状态标识与 Web Services 相关联。在这个实现模式中，客户端第 1 次访问时不需要等获得状态标识后再向 Web Services 发出请求。它可以直接向 Web Services 发出请求，服务端在相应的储存系统（如数据库，文件系统等）中创建状态元素后返回状态标识给服务端。服务端等执行结束时再将执行结果和标识一起返回给客户端，这样可以大大缩短客户端与 Web Services 之间的通信时间。

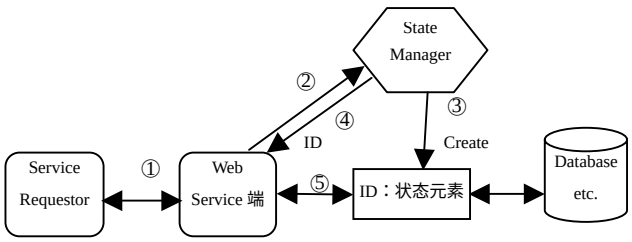

图 1 作用于有状态资源的 Web Services 状态访问模型

其具体执行步骤如下：

(1)客户端第 1 次访问时，直接向 Web Services 发出请求，然后由 Web Serivces 向状态管理器发出请求。

(2)状态管理器在状态资源库（如数据库等）中创建相应的状态元素并分配一个不重复的状态标识，同时将这个状态标识返回给 Web Services。

(3)Web Services 根据这个状态标识操作相应的状态元素，执行结束后将执行结果和状态标识一起返回给客户端。

(4)客户端得到这个状态标识后将它保存在本地系统中。

(5)客户端在后续访问时，可以将状态标识包含在请求消息中向 Web Services 发出请求，Web Services 根据请求消息中的状态标识操作状态。

2.1.2 Web Service 与状态元素之间的关联实现

要实现这个管理模式，需要将 Web Service 与状态元素之间进行关联。分析 Web Services 的状态特点，Web Service 与状态元素之间的关系应该为多对一的关系，即一个状态元素对象可为多个 Web Services 操作，如多个 Web Services 之间的依赖关系等。所以，Web Services 对象与状态元素对象之间的关联关系可通过在 Web Services 对象上加入状态标识属性来表示；而为保持状态元素的通用性，可以不用考虑它与 Web Services 之间的关联关系。当然，有些 Web Service 可以同时操作多个状态值，这可以通过将这些状态值包装在一个状态元素中，用多个 SubStateElement 来表示。其具体关联关系如图 2 所示。

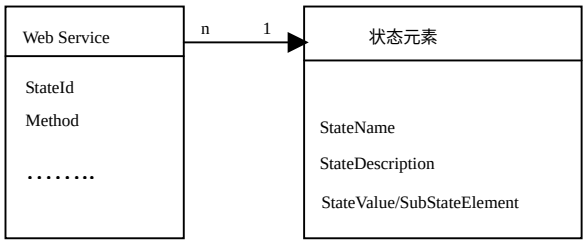

图 2 Web Service 与状态元素之间的实体—关系图

2.2 会话 Web Services 的状态管理机制

分析会话 Web Services 的状态，其具有以下几个特点：

(1)会话 Web Services 的状态是暂时的。其生命周期起始于会话开始，当会话结束时状态即宣告结束。

(2)状态特定于某一次会话，即特定于某个 Web Service 实例或复合 Web Services 实例。而有状态资源的状态则不局限于某个 Web Service。

(3)会话 Web Services 的状态是服务端自动创建的，对请求者是透明的。

(4)通常情况下，Web Services 本身不需要提供状态的访问接口。

(5)但必要时在会话期间请求者可以访问、修改或删除会话 Web Services 的状态。

由于会话 Web Services 的状态特点，因此不能直接采用作用在有状态资源的 Web Services 状态管理模型。目前，OGSI 工作组在会话 Web Services 的状态管理方面还没有提出相应的方案。而通过对现有的同类产品（如 Axis 等）分析，它们一般采用 HTTP 会话和 cookies 机制。但是这将局限于 HTTP 协议。

为此，我们提出一种类似于作用在有状态资源的 Web Services 状态模型，即采用状态管理器机制，如图 3 所示。在这个模型中，各模型组件基本与图 1 相同。但是由于会话

Web Services 状态的暂时性，因此会话状态对象用 Session 对象来表示。也就是说，状态管理器直接创建 Session 对象来保存 Web Services 状态，而不是将会话状态保存在文件或数据库中；同时，通过使 Session 对象的实现独立于 Web Services 实例来达到 Web Services 实现的无状态。这样，当某一个 Web Services 实例失效时，可以获取另一个空闲实例来执行。

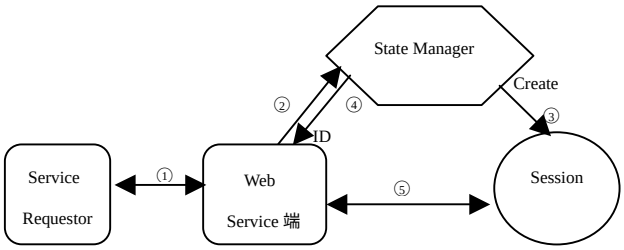

图 3　会话 Web Services 的状态访问模型

3 Web Services 状态实现及其他问题

在上述所提出的 Web Services 状态管理模型中，状态管理器是关键组件。它的接口原型可以用图 4 表示。

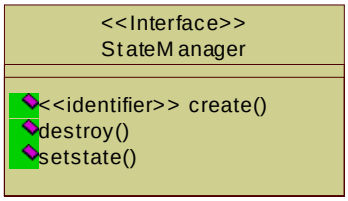

图 4 状态管理器接口原型

当然，Web Services 状态的实现涉及客户端和服务端两方面的问题。要完全实现 Web Services 状态，除了需要上面的一些管理机制的支持外，还需要解决其他的一些问题，如：客户应用程序如何与状态标识关联、请求消息如何与状态标识关联等。这些问题需要依赖于客户端的支持。

对于客户端应用程序来讲，其关联状态标识可采用两种方法：

(1)显式关联。将关联信息包含在客户端应用程序的设计中，如在购物车例子中，设计人员可将购物车标识符作为每次交互的输入。

(2)隐式关联。这种方法不需要直接在客户端应用程序中放入关联信息。当客户端发送消息时，系统自动加入关联数据。为了提供这种能力，采用的方法一般包括使用底层通信协议（例如，带有 HTTP 的 cookies）的特征或者在 Web Services 客户端运行时系统中要求显式的支持。采用 cookies 方法较简单，但是必须受特定传输协议的限制。

至于状态标识如何包含在请求消息中，考虑到 SOAP 消息格式及其传输机制特性，通常将状态标识包含在 3 个地方：(1)将状态标识包含在特定底层传输消息头或 URL 中，但这需要依赖于底层传输；(2)利用 SOAP 消息头的可扩展性，将状态标识包含在 SOAP 消息头中；(3)将状态标识放在 SOAP 消息体中，如作为调用方法的属性等。对于作用于有状态资源的 Web Services，由于其本身的操作对象就是状态，因此可将状态标识包含在 SOAP 消息体中；而对于会话 Web Services 而言，状态是其操作的结果，通常其并不提供状态的访问接口，所以可将状态标识包含在 SOAP 消息头中。

4 小结

综上所述，本文指出了现有 Web Services 状态管理机制的弊端，分别对两种有状态 Web Services 提出新的状态管理机制。这种机制在提高执行性能方面具有明显改善。

参考文献

1 Parastatidis S, Webber J, Watson P, et al. A Grid Application Framework Based on Web Services Specifications and Practices[R]. North East Regional e-Science Centre, University of Newcastle, 2004.

2 Foster I, Kesselman C, Nick J, et al. The Physiology of the Grid:An Open Grid Services Architecture for Distributed Systems Integration [EB/OL]. http://www.globus.org/research/papers/ogsa.pdf, 2002.

3 Foster I, Frey J, Graham S, et al. Modleing Stateful Resources with Web Services[EB/OL]. http://www-106.ibm.com/developerworks/library/ws- resource/ ws-modelingresources.pdf, 2004-03.

4 Ai Ting, Wang Caixia, Xie Yong. Analysis of State Management in Web Services and Various Extensions[EB/OL]. http://www.comp.nus.edu.sg/~wangxb/5505_final_report_XieYong.pdf, 2004.

5 Globus Alliance, IBM, HP. The WS-Resource Framework[EB/OL]. http://www.globus.org/wsrf/, 2004.

6 Graham M J, Caltech. Building Stateful Web Services[EB/OL]. http://grist.caltech.edu/sc4devo/presentations/files/SC4DEVO1.ppt, 2004-07.

（上接第 115 页）

```
64 bytes from 2001:251:1a05::111: icmp_seq=23 ttl=63 time=0.436 ms
64 bytes from 2001:251:1a05::111: icmp_seq=24 ttl=63 time=0.437 ms
64 bytes from 2001:251:1a05::111: icmp_seq=25 ttl=63 time=0.421 ms
64 bytes from 2001:251:1a05::111: icmp_seq=29 ttl=63 time=1.0 ms
64 bytes from 2001:251:1a05::111: icmp_seq=30 ttl=62 time=0.504 ms
64 bytes from 2001:251:1a05::111: icmp_seq=31 ttl=62 time=0.489 ms
```

图 9　ping6 在接入控制系统交接时结果

7 结束语

本文提出了基于 MIPv6 的 AAA 接入控制方案并且在 Linux 平台下扩展 MIPv6 实现了系统原型，给出实验结果。实验结果表明，此方案有效可行，它实现了用户在不同网络间漫游接入控制的透明交接，既方便了用户对网络资源的使用，又使得运营商对 MIPv6 无线网络资源实行有效的控制和管理。下一步的工作计划以电信级别的应用标准实现新的接入客户端和服务器，实现电信业务级别的应用。

参考文献

1 Johnson D B. Perkins C. Mobility Support in IPv6[S]. IETF RFC 3775, 2004-06.

2 Calhoun P, Loughney J. Diameter Base Protocol[S]. IETF RFC3588, 2003-09.

3 Faccin S M, Le F, Patil B, et al. DIAMETER Mobile IPv6 Application [Z]. Draft -le-aaa-DIAMETER-mobileipv6-03, 2003-04.

4 Thomson S, Narten T. IPv6 Stateless Address Autoconfiguration[S]. IETF RFC 2462, 1998-12.

5 Horman N. Understanding and Programming with Netlink Sockets (Version0.3)[Z]. http://people.redhat.com/nhorman/papers/netlink.pdf, 2003-12-06.

6 Russell R. Linux 2.4 Packet Filtering HOWTO[Z]. http://www.netfilter.org/documentation/HOWTO/packet-filtering-HOWTO.html, 2002-01.