Java 理论与实践: Web 层的状态复制

**简介：** 大多数具有一定重要性的 Web 应用程序都要求维护某种会话状态，如用户购物车的内容。如何在群集服务器应用程序中管理和复制状态对应用程序的可伸缩性有显著影响。许多 J2SE 和 J2EE 应用程序将状态存储在由 Servlet API 提供的 HttpSession 中。本月，专栏作家 Brian Goetz 分析了状态复制的一些选项以及如何最有效地使用 HttpSession 以提供好的伸缩性和性能。

不管正在构建的是 J2EE 还是 J2SE 服务器应用程序，都有可能以某种方式使用 Java Servlet —— 可能是直接地通过像 JSP 技术、Velocity 或者 WebMacro 这样的表示层，也可能通过一个基于 servlet 的 Web 服务实现，如 Axis 或者 Glue。Servlet API 提供的一个最重要的功能是会话管理 —— 通过 HttpSession 接口进行用户状态的认证、失效和维护。

**会话状态**

几乎每一个 Web 应用程序都有一些会话状态，这些状态有可能像记住您是否已登录这么简单，也可能是您的会话的更详细的历史，如购物车的内容、以前查询结果的缓存或者 20 页动态问卷表的完整响应历史。因为 HTTP 协议本身是无状态的，所以需要将会话状态存储在某处并与浏览会话以某种方式相关联，使得下次请求同一 Web 应用程序的页面时可以容易地获取。幸运的是，J2EE 提供了几种管理会话状态的方法 —— 状态可以存储在数据层，用 Servlet API 的 HttpSession接口存储在 Web 层，用有状态会话 bean 存储在 Enterprise JavaBeans（EJB）层，甚至用 cookie 或者隐藏表单字段将状态存储在客户层。不幸的是，会话状态管理不当会带来严重的性能问题。

如果应用程序能够在 HttpSession 中存储用户状态，这种方法通常比其他方法更好。在客户端用 HTTP cookie 或者隐藏表单字段存储会话状态有很大的安全风险 —— 它将应用程序的一部分内部内容暴露给了非受信任的客户层。（一个早期的电子商务网站将购物车内容（包括价格）存储在隐藏表单字段中，从而可以很容易被非法利用，让任何了解 HTML 和 HTTP 的用户可以以 0.01 美元购买任何商品。噢）此外，使用 cookie 或者隐藏表单字段很混乱，容易出错，并且脆弱（如果用户禁止在浏览器中使用 cookie，那么基于 cookie 的方法就完全不能工作）。

在 J2EE 应用程序中存储服务器端状态的其他方法是使用有状态会话 bean，或者在数据库中存储会话状态。虽然有状态会话 bean 在会话状态管理方面有更大的灵活性，但是在可能的情况下，将会话状态存储在 Web 层仍然有好处。如果业务对象是无状态的，那么通常可以仅仅添加更多 Web 服务器来扩展应用程序，而不用添加更多 Web 服务器和更多 EJB 容器， 这样的成本一般要低一些并且容易完成。使用 HttpSession 存储会话状态的另一个好处是 Servlet API 提供了一种会话失效时通知的容易方法。在数据库中存储会话状态的成本可能难以承受。

servlet 规范没有要求 servlet 容器进行某种类型的会话复制或者持久性，但是它建议将状态复制作为 servlet 首要 *存在理由（raison d'etre）* 的重要部分，并且它对作为进行会话复制的容器提出了一些要求。会话复制可以提供大量好处 —— 负载平衡、伸缩性、容错和高可用性。相应地，大多数 servlet 容器支持某种形式的 HttpSession 复制，但是复制的机制、配置和时间是由实现决定的。

[**回页首**](http://www.ibm.com/developerworks/cn/java/j-jtp07294/#ibm-pcon)

**HttpSession API**

简单地说， HttpSession 接口支持几种方法，servlet、JSP 页或者其他表示层组件可以用这些方法来跨多个 HTTP 请求维护会话信息。会话绑定到特定的用户，但是在 Web 应用程序的所有 servlet 中共享 —— 不特定于某一个 servlet。一种考虑会话的有用方法是，会话像一个在会话期间存储对象的 Map —— 可以用 setAttribute 按名字存储会话属性，并用getAttribute 提取它们。 HttpSession 接口还包含会话生存周期方法，如 invalidate() （它通知容器应丢弃会话）。清单 1 显示 HttpSession 接口最常用的元素：

**清单 1. HttpSession API**

|  |
| --- |
| public interface HttpSession {  Object getAttribute(String s);  Enumeration getAttributeNames();  void setAttribute(String s, Object o);  void removeAttribute(String s);  boolean isNew();  void invalidate();  void setMaxInactiveInterval(int i);  int getMaxInactiveInterval();  ...  } |

理论上，可以跨群集一致性地完全复制会话状态，这样群集中的所有节点都可以服务任何请求，一个简单的负载平衡器可以以轮询方式传送请求，避开有故障的主机。不过，这种紧密的复制有很高的性能成本，并且难于实现，当群集接近某一规模时，还会有伸缩性的问题。

一种更常用的方式是将负载平衡与会话 相似性（affinity） 结合起来 —— 负载平衡器可以将会话与连接相关联，并将会话中以后的请求发送给同一服务器。有很多硬件和软件负载平衡器支持这个功能，并且这意味着只有主连接主机和会话需要故障转移到另一台服务器时才访问复制的会话信息。

[**回页首**](http://www.ibm.com/developerworks/cn/java/j-jtp07294/#ibm-pcon)

**复制方式**

复制提供了一些可能的好处，包括可用性、容错和伸缩性。此外，有大量会话复制的方法可用：方法的选择取决于应用程序群集的规模、复制的目标和 servlet 容器支持的复制设施。复制有性能成本，包括 CPU 周期（存储在会话中的序列化对象）、网络带宽（广播更新），以及基于磁盘的方案中写入到磁盘或者数据库的成本。

几乎所有 servlet 容器都通过存储在 HttpSession 中的序列化对象进行 HttpSession 复制，所以如果是创建一个分布式应用程序，应当确保只将可序列化对象放到会话中。（一些容器对像 EJB 引用、事务上下文、还有其他非可序列化的 J2EE 对象类型有特殊的处理。）

**基于 JDBC 的复制**

一种会话复制的方法是序列化会话内容并将它写入数据库。这种方法相当直观，其优点是不仅会话可以故障转移到其他主机，而且即使整个群集失效，会话数据也可以保存下来。基于数据库的复制的缺点是性能成本 —— 数据库事务是昂贵的。虽然它可以在 Web 层很好地伸缩，但是它可能在数据层产生伸缩问题 —— 如果群集增长大到一定程度，扩展数据层以容纳会话数据会很困难或者成本无法接受。

**基于文件的复制**

基于文件的复制类似于使用数据库存储序列化的会话，只不过是使用共享文件服务器而不是数据库来存储会话数据。这种方式的成本一般比使用数据库的成本（硬件成本、软件许可证和计算开销）低，其代价则是可靠性（数据库可提供比文件系统更强的持久化保证）。

**基于内存的复制**

另一种复制方式是与群集中的一个或者多个其他服务器共享序列化的会话数据副本。复制所有会话到所有主机中提供了最大的可用性，并且负载平衡最容易，但是因为复制消息所消耗的每个节点的内存和网络带宽，最终会限制群集的规模。一些应用服务器支持与“伙伴（buddy）”节点的基于内存的复制，其中每一个会话存在于 主服务器上和一台（或更多）备份服务器上。这种方案比将所有会话复制到所有服务器的伸缩性更好，但是当需要将会话故障转移到另一台服务器上时会使负载平衡任务复杂化，因为它必须找出另外哪一台（几台）服务器有这个会话。

**时间考虑**

除了决定如何存储复制会话数据，还有什么时候复制数据的问题。最可靠但也最昂贵的方法是每次数据改变时复制它（如每次 servlet 调用结束）。不那么昂贵、但是在故障时会有丢失一些数据的风险的方法是在每超过 N 秒时复制数据。

与时间问题有关的问题是，是复制整个会话还是只试尝复制会话中改变了的属性（它包含的数据会少得多）。这些都需要在可靠性和性能之间进行取舍。Servlet 开发人员应当认识到在故障转移时，会话状态可能变得“过时”（是几次请求前的复制），并应当准备处理不是最新的会话内容。（例如，如果一个interview 的第 3 步产生一个会话属性，而用户在第 4 步时，请求被故障转移到一个具有两次请求之前的会话状态复制的系统上，那么第 4 步的 servlet 代码应预备在会话中找不到这个属性，并采取相应的行动 —— 如重定向，而不是认定它会在那里、并在找不到它时抛出一个NullPointerException 。）

[**回页首**](http://www.ibm.com/developerworks/cn/java/j-jtp07294/#ibm-pcon)

**容器支持**

Servlet 容器的 HttpSession 复制选项以及如何配置这些选项是各不相同的。IBM WebSphere ®提供的复制选项是最多的，它提供了在内存中复制或者基于数据库的复制、在 servlet 末尾或者基于时间的复制时间、传播全部会话快照（JBoss 3.2 或以后版本）或者只传播改变了的属性等选择。基于内存的复制基于 JMS 发布-订阅，它可以复制到所有克隆、一个“伙伴”复制品或者一个专门的复制服务器。

WebLogic 还提供了一组选择，包括内存中（使用一个伙伴复制品）、基于文件的或者基于数据库的。JBoss 与 Tomcat 或者 Jetty servlet 容器一同使用时，进行基于内存的复制，可以选择 servlet 末尾或者基于时间的复制时间，而快照选项（在 JBoss 3.2 或以后版本）是只复制改变了的属性。Tomcat 5.0 为所有群集节点提供了基于内存的复制。此外，通过像 WADI 这样的项目，可以用 servlet 过滤机制将会话复制添加到像 Tomcat 或者 Jetty 这样的 servlet 容器中。

[**回页首**](http://www.ibm.com/developerworks/cn/java/j-jtp07294/#ibm-pcon)

**改进分布式 Web 应用程序的性能**

不管决定使用什么机制进行会话复制，可以用几种方式改进 Web 应用程序的性能和伸缩性。首先记住，为了获得会话复制的好处，需要在部署描述符中将 Web 应用程序标记为 distributable，并保证在会话中的所有内容都是可序列化的。

**保持会话最小**

因为复制会话有随着会话中的对象图（object graph） 的变大而增加成本，所以应当尽可能地在会话中少放置数据。这样做会减少复制的序列化的开销、网络带宽要求和磁盘要求。特别地，将共享对象存储在会话中一般不是好主意，因为它们需要复制到它们所属的 *每一个*会话中。

**不要绕过 setAttribute**

在改变会话的属性时，要知道即使 servlet 容器只是试图做最小的更新（只传播改变了的属性），如果没有调用setAttribute ，容器也可能没有注意到已经改变的属性。（想像在会话中有一个 Vector ，表示购物车中的商品 —— 如果调用 getAttribute() 获取 Vector 、然后向它添加一些内容，并且不再次调用 setAttribute ，容器可能不会意识到 Vector已经改变了。）

**使用细化的会话属性**

对于支持最小更新的容器，可以通过将多个细化的对象而不是一个大块头放到会话中而降低会话复制的成本。这样，对快速改变的数据的改变也不会迫使容器去序列化并传播慢速改变的数据。

**完成后使之失效**

如果知道用户完成了会话的使用（如，用户选择注销登录），确保调用 HttpSession.invalidate() 。否则，会话将持久化直到它失效，这会消耗内存，并且可能是长时间的（取决于会话超时时间）。许多 servlet 容器对可以跨所有会话使用的内存的数量有一个限制，达到这个限制时，会序列化最先使用的会话并将它写到磁盘上。如果知道用户使用完了会话，可以使容器不再处理它并使它作废。

**保持会话干净**

如果在会话中有大的项，并且只在会话的一部分中使用，那么当不再需要时应删除它们。删除它们会减少会话复制的成本。（这种做法类似于使用显式 nulling 以帮助垃圾收集器，老读者知道我一般不建议这样做，但是在这种情况下，因为有复制，在会话中保持垃圾的成本要高得多，因此值得以这种方式帮助容器。）

[**回页首**](http://www.ibm.com/developerworks/cn/java/j-jtp07294/#ibm-pcon)

**结束语**

通过 HttpSession 复制，Servlet 容器可以在构建复制的、高可用性的 Web 应用程序方面给您减轻很多负担。不过，对于复制有一些配置选项，每个容器都不一样，复制策略的选择对于应用程序的容错、性能和伸缩性有影响。复制策略的选择不应当是事后的 —— 您应当在构建 Web 应用程序时就考虑它。并且，一定不要忘记进行负载测试以确定应用程序的伸缩性 —— 在客户替您做之前。