有状态网络的J2EE技术 PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/289/2021_2022__E6_9C_89_E7_8A_B6_E6_80_81_E7_c104_289961.htm Web 应用程序协议被分成两大类别：无状态（stateless）和有状态（stateful），协议的状态指的是它“ 记忆 ”从一个传输到下一个传输的信息的能力。因为有状态连通性是大多数企业应用程序的基本需求之一，并且因为 Web 应用程序依赖于 HTTP（内在的无状态协议），所以聪明的开发人员已经找到了许多技巧来在 HTTP 上模拟有状态连接。有状态信息可以存储在 HTML 表单字段中、附加到超链接或者存储在客户机端的 cookie 中。客户机和服务器之间的有状态交互可以在 Web 层或业务层上进行管理。要在 Web 层上管理状态，我们使用与 HTTPSession API 结合的 servlet.要在业务层上管理状态，我们使用有状态会话 EJB 组件。在接下来的章节里，我们将探究这两种开发选项。 Web 层 Servlet 体系结构的 HTTPSession API 允许应用程序开发人员管理跨网络的客户机/服务器交互（或会话）的状态。HTTPSession 接口定义了 HTTPSession API 的核心功能。它为 J2EE 应用程序提供了一种方法，使它可以识别跨多个页面请求的单个客户机，以及将数据存储在与那个客户机相关联的服务器上。通过该接口，servlet 容器创建和管理客户机和服务器之间的会话。该会话由 HTTPSession 对象表示，它可以跨来自相同客户机的多个连接和页面请求持续存在一段特定的时间。Servlet 使用该接口来查看与处理有关会话的信息，如创建时间和上一次访问会话的时间。该接口还允许 servlet 将对象绑定到会话，从而以一

种跨多个连接（来自相同客户机）持续存在的方式将该信息与特定的客户机进行关联。 Servlet 体系结构 Servlet 体系结构并没有因为使用 HTTPSession API 而发生改变。就象在无状态网络中一样，servlet 代表客户机执行业务请求，并且充当控制器、视图，或者二者同时兼任。Servlet 还可以有效地处理用户交互，如内容格式化和显示、基本请求处理和安全性请求及更多。就象在无状态网络中一样，servlet 最好用于管理客户机交互，那么通常可使用助手类（如 JavaBeans）来应付繁重的处理或者与后端组件相互操作（interface）。 因此，HTTPSession 接口允许 servlet 容器创建和管理客户机会话，并且使 servlet 能访问与会话相关的信息、将对象绑定到会话以及访问先前绑定的对象。到现在为止，一直都还不错。但是 servlet 容器如何跟踪通过无状态协议（如 HTTP）通信的客户机呢？为了实现这一点，为每个 HTTPSession 对象都提供一个唯一的标识，以确保每个客户机会话和与会话相关的数据可以被唯一标识。考虑到 HTTP 内在的无状态本质，在每次请求时，该会话标识必须被客户机传递给服务器，以便于 servlet 容器将客户机与正确的会话相关联。会话标识可以用三种方式中一种进行传递：作为 HTML 表单中的参数（通常是隐藏字段）；作为附加在查询字符串后的参数；或者作为 cookie 的属性。不管会话标识如何传递，servlet 容器都将拦截它，检查它，并找到与之关联的 HTTPSession 对象。 Servlet 性能 由 Servlet 体系结构创建的轻量级线程模型决不会因为 servlet 或 JSP 文件创建、读取或修改 HTTPSession 对象而受到破坏。该对象只是将对象引用存储为简单键-值对的散列表或类似的集合。同样，HTTPSession 内存空间的实现本身也是轻

量级的，只需要存储（或许序列化）会话对象和相应的会话标识。简而言之，servlet 可以支持与 HTTP 客户机的有状态交互，而且对应用程序设计或容器资源产生最小的影响。 业务层 J2EE 为在业务层上处理状态提供了内置的支持。与无状态会话 bean 一样，有状态会话 bean 也被映射到业务过程。两者之间的关键区别是：无状态 bean 及其数据在单个客户机请求的生命周期内存活，而有状态 bean 却维护与客户机的对话并且它们的数据跨多个请求持续存在。与 servlet 不同，有状态会话 bean 不需要任何特殊的对象，也不需要使用额外的接口来创建有状态连接。EJB 容器提供了所有有状态会话 bean 管理。对于 bean 而言，所有必要的工作就是在其部署描述符中将其声明为 stateful. EJB 体系结构 从体系结构的观点看，有状态会话 bean 与其无状态的同类没有任何差别。两种类型的 bean 都可以很好地充当视图、控制器或模型；二者通常都可以实现虚包（Facade）模式或业务委派（Business Delegate）模式；二者都可以与多个客户机类型一起使用。有状态会话 bean 可以通过 servlet（或 JSP 文件）、帮助 servlet（或 JSP 文件）的 JavaBean 和另一个企业 bean 访问，或者直接通过 applet、Swing 应用程序或其它 Java 应用程序，或者甚至是使用 IIOP 协议的非 Java 客户机访问。 管理有状态 bean 正如以前阐述的，会话 bean 是最轻量级类型的企业 bean 类型。特别地，无状态会话 bean 可以方便地被容器合用，因为它们只需要维护每个请求的状态。 相反，有状态会话 bean 与容器资源并不那样友好。有状态会话 bean 的池不能象无状态 EJB 组件的池那样用来容纳任何客户机请求。有状态 bean 只能处理来自一个客户机的请求，直到该客户机释放其对那个特殊 bean

实例的控制。有状态会话 bean 消耗了容器的大量时间和内存。为了保存客户机调用之间的 bean 状态，容器必须将 bean 实例保存在活动内存中，或者临时将状态写到持久性存储（如文件系统或数据库）中。将状态分配到持久性存储中就是所谓的钝化（passivation）。当以前钝化的企业 bean 被再次请求时，容器通过从池中检索 bean，并且利用钝化前 bean 的持久性状态对它进行初始化，来激活它。下图阐明了有状态会话 bean 的钝化和激活： 决定将 bean 保存在内存中还是对它进行钝化，然后再激活它，这取决于各个供应商。尽管在释放容器资源方面钝化机制非常有帮助，但是在防止服务器崩溃以避免丢失有状态会话 bean 的活动状态方面却无能为力。尽管一些供应商提供了会话恢复功能以解决这个问题，但它不是标准的，因此依赖该功能会降低应用程序的可移植性。但是，别担心！EJB 规范确实定义了一个接口（javax.ejb.SessionSynchronization），它可以向企业 bean 警报事务的状态，包括由于服务器崩溃而失败的事务（假定不是拔了服务器的插头）。实现 SessionSynchronization 接口的企业 bean 必须定义三个已声明的方法特征符：afterBegin（）、beforeCompletion（） 和 afterCompletion（boolean）。这些方法使 bean 可以从容器接收三个额外的回调，以允许正确处理 bean 中的事务状态。 EJB 组件性能 从性能的角度看，servlet 和无状态会话 bean 是相当具有竞争力的技术。它们都可以使用实例池为来自客户机的请求提供服务。但是，当您添加了应用程序状态管理时，巨大的性能差异就将显现。与可用作 Servlet 体系结构一部分的轻量级 HTTPSession 机制不同，有状态会话 bean 需要一个更加重量级的针对状态管理

（如上面概述的钝化／激活方案）的解决方案。这个针对有状态会话 bean 的常用解决方案需要花费服务器时间和资源来钝化 bean 状态、回收 bean 实例和激活 bean 状态。上述的每个过程都需要几次容器调用，以及需要直接对 bean 执行回调方法，以确保正确处理 bean 的状态。总而言之，有状态会话 EJB 组件为管理应用程序状态提供了一种重量级机制。 选择合适的技术 与无状态的 J2EE 体系结构不同，J2EE 应用程序不提供典型的配置来充当指南或蓝图。管理状态时，合适的体系结构取决于下列因素： 客户机是基于 Web（HTTP）的吗？ 对话状态需要包含到 GUI 中吗？ 服务器将处于哪种负载条件下呢？ 有状态组件需要能够在服务器崩溃后仍然有效吗？ 该组件需要哪种事务上下文呢？ 有状态数据有多重要？ 尽管上述一些问题似乎明显地倾向于其中一种技术，但是许多有状态方案实际上既需要使用 servlet 又需要使用 EJB 组件。至关重要的决定就是确定是在 Web 层还是在业务层上管理状态，或者同时在两个层上管理状态。在下一节中，我们将研究一些可能的企业应用程序方案，及其最适宜的解决方案。 应用程序客户机 标准的应用程序客户机是与另一个系统或组件相互操作的客户机。我们将研究三种典型的应用程序客户机方案，并且讨论每个客户机最适合的有状态解决方案： 如果客户机是基于 Java 的，并且与服务器处于相同的防火墙之后，那么您首先应该决定有状态交互模型是否是必需的。管理有状态会话 bean 是资源密集型的，因此您应该考虑更轻量级的备用方案。最佳解决方案就是使用 RMI 直接与应用程序服务器中的无状态会话 bean 对话。如果有状态解决方案是必需的，则请考虑使用带有简单事务层的无状态会话 bean，或者

在业务层上创建瘦 servlet 层。任何一种解决方案花费部分成本即可提供有状态体验。最后，如果您的客户机必须与跨多个请求的业务过程的状态进行紧耦合，并且不能接受添加 Web 层，那么有状态会话 bean 是显而易见的选择。