ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.17, No.5, May 2006, pp.1195−1203
DOI: 10.1360/jos171195

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 一个 J2EE 应用服务器的 Web 容器集成框架*

林 泊, 周明辉+, 刘天成, 黄 罡, 梅 宏

(北京大学 信息科学技术学院 软件研究所,北京 100871)

## A Web Container Integration Framework in J2EE Application Servers

LIN Bo, ZHOU Ming-Hui+, LIU Tian-Cheng, HUANG Gang, MEI Hong

(Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

+ Corresponding author: Phn: +86-10-62757670, Fax: +86-10-62751792, E-mail: zhmh@sei.pku.edu.cn, http://www.sei.pku.edu.cn

**Lin B, Zhou MH, Liu TC, Huang G, Mei H. A Web container integration framework in J2EE application servers. *Journal of Software*, 2006,17(5):1195−1203.** http://www.jos.org.cn/1000-9825/17/1195.htm

**Abstract**: Regarding the shortcomings found with the traditional method of integrating a J2EE (Java 2 platform enterprise edition) Web Container, a two-layered Web container integration framework is proposed: the outer layer is independent of the Web container implementation and is the interface between the Web container and management tools, deployment tools and other modules; the inner layer is an extension to the Web container. The framework has been implemented on the J2EE application server PKUAS (Peking University Application Server). Test results have shown that this framework can satisfy the design goals of a pluggable, unified configurable Web container, and its performance is satisfactory.

**Key words**: J2EE application server; Web container; integration framework

摘 要: 针对 J2EE(Java 2 platform enterprise edition)应用服务器集成 Web 容器的传统实现方式存在的不足,提出一个两层结构的 Web 容器集成框架:外层独立于 Web 容器实现,满足管理工具、部署工具等其他模块与 Web 容器的交互;内层则是对特定 Web 容器的包装、扩展或改良.该框架已在 J2EE 应用服务器 PKUAS(Peking University Application Server)上得以实现.测试结果表明,该框架具有良好的可插拔性,实现了应用服务器配置和管理机制的统一,且性能良好.

关键词: J2EE 应用服务器;Web 容器;集成框架
中图法分类号: TP393 文献标识码: A

J2EE(Java 2 platform enterprise edition)[1]是 Sun 公司于 1999 年推出的 Java 企业计算平台规范与技术.它通过提供企业计算环境所必需的各种服务,使得部署在 J2EE 平台上的基于构件的分布式应用可以实现高可用性、安全性、可扩展性和可靠性,是目前应用最为广泛的面向 Web 的应用系统结构规范.

* 本文为 2005 年中国计算机大会推荐优秀论文.Supported by the National Natural Science Foundation of China under Grant Nos.60125206, 60233010, 90412011 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2005AA112030, 2005AA113031 (国家高技术研究发展计划(863)); the National Grand Fundamental Research 973 Program of China under Grant No.2002CB312003 (国家重点基础研究发展计划项目(973))

Received 2005-06-15; Accepted 2006-01-18

J2EE 采用多层分布式应用模型(如图 1 所示),包括表示层、业务层和数据层.其中业务层的 Web 容器、EJB 容器和 J2EE 服务构成 J2EE 应用服务器,为部署在其上的 J2EE 应用构件提供运行支持,日渐成为构造 Internet 应用的主流平台.一般地,Web 容器在接收到客户的 Web 请求时,会由相应的 Web 构件进行简单的业务处理,之后再将请求转发给 EJB 容器,由相应的 EJB 构件处理复杂业务逻辑并与数据层进行交互.

Web 容器实现了 Servlet/JSP[2]标准,负责为部署在其中的 Web 构件(Servlet 和 JSP(JavaServer Pages))提供运行支持.由于 Servlet/JSP 规范本身是一个独立的规范,并且早于 J2EE 规范产生,因此,Web 容器实现的发展历程也要久于 J2EE 应用服务器(独立的 Web 容器实现一般被称作 Web 服务器),其各方面技术已经相当成熟稳定.常见的开源产品如 Tomcat,Jetty 等均经受过大规模应用的考验.为了充分利用 Web 容器多年来积淀的技术、最大程度地节约开发成本,集成第三方开源 Web 服务器已成为现今开源 J2EE 应用服务器实现的惯例.

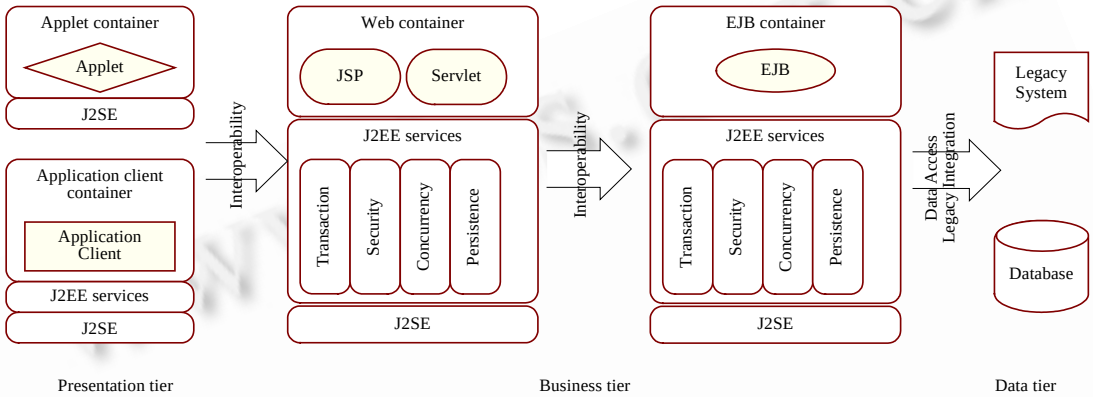


Fig.1 J2EE architecture

图 1 J2EE 体系结构

然而,在代码级集成一个独立的第三方产品势必面临许多问题,包括:用户界面不一致、管理接口不一致、配置方式不一致以及内部结构不一致等.传统紧耦合的集成方式无法解决这些问题,主要体现为:

第一,对集成的 Web 容器只在高层进行操作,而对其内部模块的可控制性较差.在这种方案下,应用服务器只负责设定相关的参数和环境变量,而后调用 Web 容器自身的启动代码进行启动.对于 Web 容器的内部模块应用服务器很难进行控制.

第二,配置文件、管理方式与应用服务器本身的机制不统一.由于集成没有深入到 Web 容器的内部模块,从而导致了对 Web 容器的配置以及管理都需通过 Web 容器提供的机制.也就是说,用户无法通过应用服务器的配置文件和管理工具[3]对整个应用服务器进行统一的配置或管理.

第三,对集成的 Web 容器有过强的依赖性.Servlet/JSP 标准仍然在升级演变,第三方 Web 容器产品也有自己的版本更新周期,而且,一般来说要短于应用服务器的更新周期.对 Web 容器的强依赖性,意味着升级更换 Web 容器的代价大为提高,往往需要改动应用服务器中许多其他模块的代码,这就使得应用服务器中的 Web 容器的实现很难跟上集成的第三方产品的更新速度,导致应用服务器的竞争力下降.

为了解决以上问题,本文提出一个 J2EE 应用服务器中 Web 容器的集成框架,并给出该框架在北京大学自主开发的应用服务器——PKUAS(Peking university application server)上的实现过程.

本文第 1 节分析 Web 服务器的基本结构和不同实现的共同点.第 2 节详细阐述两层结构的 Web 容器集成框架.第 3 节在 J2EE 应用服务器 PKUAS 上实现该框架,并对实现结果进行性能评测与分析.第 4 节对目前的相关工作进行考察,并与本文提出的框架进行比较.最后一节总结全文.

## 1 Web 服务器分析

### 1.1 Web服务器

通常,一个完整的 Web 服务器包含一个或多个“虚拟主机”.所谓虚拟主机,就是在一个物理的服务器上配置多个域名,每个域名对应相同或不同的应用.这样,客户端看起来好像是有多个主机;Web 容器在收到请求时,根据客户端给出的域名来确定处理该请求的虚拟主机.在每个虚拟主机中又可以部署一个或多个 Web 应用.每个部署在服务器上的 Web 应用要提供对本应用的生命周期管理等功能.尽管不同的 Web 服务器的实现各不相同,但都按照服务器-虚拟主机-应用的层次结构进行设计和管理.

### 1.2 Tomcat结构分析

Apache Tomcat[4]是 Servlet/JSP 标准的参考实现,作为一个成功的、有着广泛应用的 Web 容器,Tomcat 有着整齐而清晰的内部结构,如图 2 所示.这是一个较为典型和完整的 Web 容器结构,其他实现或多或少具有类似的结构.其中关键的模块有:

Server.一个 Server 模块代表了整个 Tomcat 容器.它的属性代表了 Web 容器作为一个整体的特征.一个 Server 模块主要包括一个或多个 Service 模块以及顶层的命名资源.在运行期间,Server 模块应监听 port 属性所指定的端口,并对于该端口接受的每一个连接读入其所传入的第 1 行信息并与 shutdown 属性所指定的字串进行比较:如果两者相吻合,则执行关闭服务器的步骤;否则,不执行任何操作并断开连接.这种机制为容器的远程管理提供了可能.

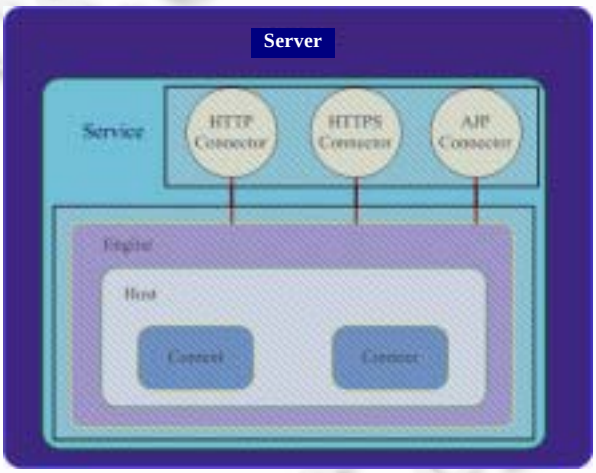


Fig.2 Inner structure of Tomcat

图 2 Tomcat 内部结构

Service.一个 Service 模块代表了共享同一个 Container 实例(通常是 Engine 实例)的一个或多个 Connector 实例所组成的逻辑整体.该 Container 负责处理这个 Service 中所有 Connector 接受的请求.Service 模块最主要的职责在于:允许加密的 Connector 以及普通的 Connector 连接到同一组 Web 应用.同一个 Connector 不能同时存在于两个不同的 Service 中.在同一个 Java 虚拟机中可以同时存在多个 Service 实例,但这些实例除了共享最基本的 Java 运行时类库以外,相互之间是完全独立的.

Connector.Connector 模块负责接受来自网络客户端的请求,并将请求的处理结果反馈给客户端.每个 Connector 实例实际上实现的是一种网络传输协议,它将通过这种协议传入的客户端请求进行分析,构造相应的 Request 和 Response 实例,找出适合相应该请求的 Container 实例,调用该 Container 的 invoke 方法,并将 Request 和 Response 实例作为参数传入,最终将处理结果或者错误信息反馈给客户端.

Engine.Engine 是 Container 的子接口,它代表了整个 Servlet 引擎.在运行中,Engine 永远是 Container 层次结构的最高级,所以,其 setParent()方法若如被调用将总是抛出 IllegalArgumentException 异常.一个 Engine 的子 Container 通常是一个或多个的 Host 实例.

Host.Host 代表的实际上就是常说的“虚拟主机”的概念.在一个 Tomcat Web 服务器中可以同时运行多个 Host,每个 Host 都与一个特定的主机名以及任意个“别名”相绑定.每个客户端的请求都会根据主机名映射到相应的 Host 进行处理.Host 的下级 Container 通常是一些 Context 实例.

Context.一个 Context 就是一个独立的 Web 应用.它包括了 Web 应用的所有 Servlet 类、JSP 文件、静态页面和图片、Jar 包、环境变量、各种配置参数以及其他各种资源.Context 的下级 Container 通常是 Wrapper 的

实例.

### 1.3 Jetty结构分析

Jetty 是由 Mort Bay Consulting 领导的开源 Java Web 容器项目[5].Jetty 项目起源于 Mort Bay 创始人 Greg Wilkin 的 Java 编程竞赛参赛作品,发展至今已有近 10 年的历史.如今,Jetty 以其小巧、高效以及高度的可嵌入性和稳定性得到了广泛的应用和认同.

与 Tomcat 类似,Jetty 也按照功能模块的结构组织,但不具有 Tomcat 中的父子结构层次,模块与模块之间是一种关联和使用的关系.比较显著的不同是,Jetty 中没有一个显示地代表“虚拟主机”的模块,而是把虚拟主机的功能分别在应用级和服务器级进行实现.在 Jetty 中,类 HttpServer 定义一个 Web 服务器;类 HttpListener 定义一个监听端口,类似于 Tomcat 中的 Connector;而类 HttpContext 对应于 Web 应用.

## 2 两层结构的 Web 容器集成框架

针对文章开始部分指出的问题,以第 1 节 Web 服务器的分析为基础,本文提出一个两层结构的 Web 容器集成框架,把各种不同 Web 容器实现的配置抽象并规范化成同一种表示和配置方式,由框架来做适配,实现 Web 容器的无缝集成,从而实现应用服务器配置和管理机制的统一;同时,给予用户选择具体 Web 容器的自由.这个框架能够将 Web 容器的具体实现对 Web 容器以外的模块完全屏蔽,同时,在这个框架下,各种不同结构的 Web 容器实现都能够被集成到应用服务器中来.

### 2.1 体系结构

为了有效隔离 Web 容器的具体实现,集成框架包括两个层次:面向应用服务器其他模块的外层和面向特定 Web 容器实现的内层,如图 2 所示(当作为应用服务器中的一个模块时,Web 容器需要提供被应用服务器其他模块访问的接口).框架的外层定义应用服务器其他模块与 Web 容器的交互,例如管理工具需要对 Web 容器实施启动、停止和配置等一系列管理操作,部署工具需要部署 Web 构件到 Web 容器上等.所有交互全部通过外层进行,而不对 Web 容器的内部实现有任何依赖性;框架的内层主要是针对特定 Web 容器实现的适配,通过一系列的映射或模拟,将特定 Web 容器所提供的功能“反映”为外层所暴露出来的结构.在这样一个框架层次下,实现者可以同时集成多种 Web 容器,而无须对应用服务器的其他模块作任何修改.

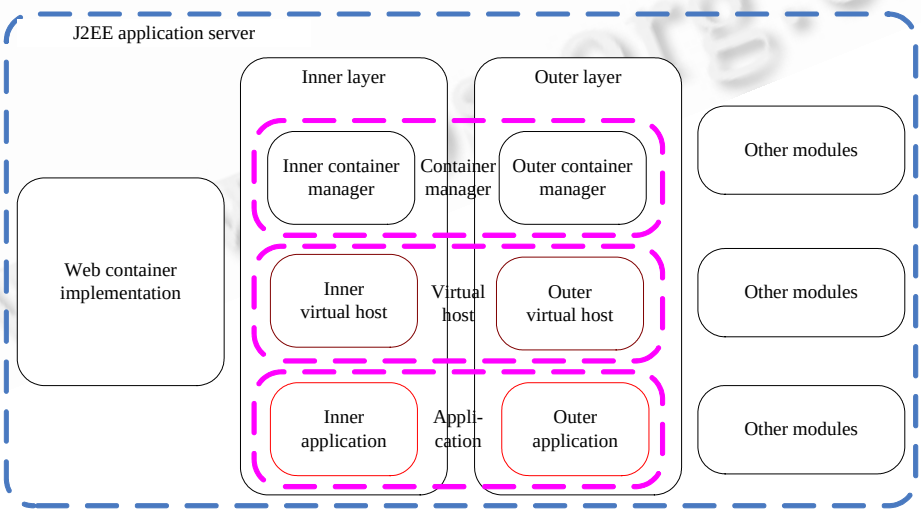


Fig.3 Web container integration framework architecture

图 3 Web 容器集成框架体系结构

根据前面对 Web 容器的关键功能的深入分析,我们为集成框架提炼出 3 类实体:容器管理器、虚拟主机和

应用.框架的内外层皆围绕这 3 类实体建立,以实现封装和隔离,如图 3 所示.这 3 类实体之间存在一对多的父子关系,构成一个 3 层的树状结构.对外界来说,每一个实体都是一个可操作的对象,它们反映了 Web 容器对外提供的主要功能.下面是这 3 类实体的具体描述:

容器管理器.该实体代表了一个 Web 容器,它主要为具体 Web 容器实现以及它所下属的子实体提供生命周期管理等.该实体定义 Web 容器的标识、监听的端口号、使用的协议类型(HTTP,HTTPS,AJP 等).容器管理器没有父实体,可以包含一个或多个虚拟主机作为子实体.

虚拟主机.该实体主要用于部署应用.其存在的目的是划分同一 Web 容器中不同应用的集合,使得应用集合之间相互独立.该实体为本虚拟主机和下属的子实体提供生命周期管理等.该实体的属性包括:虚拟主机的标识、虚拟主机的别名.该实体与一个父实体即容器管理器实体关联,同时可以包含任意多个应用作为子实体.

应用.该实体代表了 Web 容器中最小的可部署单元,为应用定义应用的上下文路径(同时也是应用的标识)、应用在本地的路径以及其他一些相关参数.该实体没有子实体.

### 2.2 接口描述

本节主要针对第 2.1 节提出的两层结构的 Web 容器集成框架定义内外层接口,为集成框架建立详细的技术视图.

#### 2.2.1 外层接口

Web 容器集成框架的外层(由容器管理器、虚拟主机和应用 3 类实体的外层组成)描述了 Web 容器需要对外提供的主要功能.针对 3 类实体外层,本文定义出 3 个接口,描述了 Web 容器对外提供的所有方法.应用服务器的其他模块通过这组方法访问 Web 容器提供的服务,或对 Web 容器进行管理.接口定义如下:

(1) WebController.该接口对应容器管理器实体,代表了整个 Web 容器,任何需要使用或操作 Web 容器的模块都需要通过该接口进行.该接口提供的方法包括启动与停止 Web 容器,或者添加、删除虚拟主机以及获取已定义的虚拟主机实例等;

(2) VirtualHost.该接口对应虚拟主机实体,代表一个虚拟主机.虚拟主机同时是部署 Web 应用的容器,一个应用在同一时间只能被部署在一个虚拟主机上.该接口提供的方法包括启动与停止该虚拟主机、部署与卸载 Web 应用等;

(3) VirtualApp.该接口对应于应用实体,代表了一个部署在服务器上的 Web 应用.通过该接口可以得到具体应用的信息,包括应用根目录的本地路径和映射的上下文路径等.另外,该接口还提供部署、启动、停止和卸载应用的方法.

这 3 个接口反映了 Web 容器的逻辑层次结构,能够满足应用服务器其他模块对 Web 容器的访问,并且完全独立于所集成的 Web 容器,很好地将 Web 容器的内部实现细节隐藏了起来.

#### 2.2.2 内层接口

Web 容器集成框架内层的主要功能是为外层接口到特定 Web 容器提供的方法进行适配,即对特定 Web 容器进行包装、扩展或改良,用于匹配外层接口.为了完成该功能,内层需要实现的内容包括:

(1) 接入应用服务器管理框架所需的接口.Web 容器集成到应用服务器中需要遵从一定的管理规则,即实现接入应用服务器内核的管理接口.例如,一般应用服务器都会采用 Java Management Beans (Java management extensions 规范[6]的组成部分)来实施统一管理.因此,Web 容器为纳入应用服务器整体管理框架也必须实现 MBeans 接口;

(2) 实现框架外层中定义的接口,将框架外层所定义的功能映射到 Web 容器所提供的功能.若集成的 Web 没有相应的功能,则通过类似功能进行模拟或者直接实现该功能.一般来说,特定的 Web 容器实现都会有相关模块进行匹配.

## 3 框架实现和测试

### 3.1 框架在PKUAS上的实现

本节将以在 PKUAS 中集成 Apache Tomcat 和 Jetty 为例,说明该框架的实现过程.

#### 3.1.1 PKUAS 简介

PKUAS 是北京大学信息科学技术学院软件研究所自主开发的构件运行支撑平台[7],符合 J2EE1.3 与 EJB2.0 规范[1].PKUAS 借鉴源自操作系统的微内核思想,设计了基于微内核、高度构件化的平台体系结构.PKUAS 抽取构件运行平台的基本功能形成一个内核,将平台的其他功能封装在各个相对独立的模块内(称为系统构件),允许用户定制与扩展这些系统构件,在系统启动阶段由内核装配成构件运行支撑平台.PKUAS 的微内核实现基于 JMX 技术.

#### 3.1.2 框架外层的实现

由于 PKUAS 采用基于 JMX 技术的微内核,框架中的 3 类实体分别实现为 3 种 Mbean:实体外层由 MBean 的接口给出;而实体内层则是 Mbean 的实现.具体地定义 3 类实体外层如下:

- 容器实体外层 WebContainerMBean:定义方法 setHttpPort,findHost,removeHost,addHost;
- 虚拟主机实体外层 VirtualHostMBean:定义方法 findApp,deploy,undeploy;
- 应用实体外层 ApplicationMBean:定义方法 getContextPath,getLocalPath.

对应于上面给出的 3 类实体外层的定义,对 Web 容器的参数设置全部在 PKUAS 的模块配置文件 services.conf 中进行,主要的选项包括:

```
〈SERVICE CLASS=“pku.as.web.tomcat4.WebContainer”〉
〈ATTRIBUTE NAME=“HttpPort” VALUE=“8080” /〉
...
〈Deployer NAME=“Host”〉
    〈ATTRIBUTE Name=“appBase” Value=“application” /〉
    〈Application localDir=“...” contextPath=“...” /〉
        ...
〈/Deployer〉
〈/SERVICE〉
```

此定义并不因为集成了不同的 Web 容器实现而发生变化,只由其中 SERVICE 标签的 CLASS 属性用于选择特定的 Web 容器实现.另外,通过 Web 容器框架外层接口提供的方法,PKUAS 管理工具[8]可提供统一的管理界面对 Web 容器进行管理,如图 4 所示.

Fig.4 Management tool

图 4 管理工具

### 3.1.3 框架内层的实现

根据第 1.1.2 节中的分析,在 Tomcat 的模块中,Server 和 Service 只是提供一些高层的抽象机制,对于本文的 Web 容器框架来说,这些机制将在框架内层中实现.这样,实现者可以重新设计 Web 容器的配置方式,将其整合到应用服务器的配置文件中去,同时也能够把 Tomcat 的内部实现细节完全隐藏起来,仅暴露出用户真正关心的配置选项.图 2 中阴影部分表示了 PKUAS 采用的部分.对应 Tomcat 的 3 类实体内层具体实现如下:

- 容器实体内层 WebContainer:包装了 Tomcat 中的 Engine 模块;
- 虚拟主机实体内层 VirtualHost:包装了 Tomcat 中的 Host 模块;
- 应用实体内层 Application:包装了 Tomcat 中的 Context 模块.

表 1 给出了上一节定义的实体外层方法到实体内层方法及其对应调用的 Tomcat 方法的对应关系.此外,PKUAS 还实现了集成 Jetty[9]的内层实体.其具体集成过程与集成 Tomcat 类似.表 2 给出了外层实体、内层实体以及对应封装的 Jetty 类的对应关系.

**Table 1** Outer entity methods and the corresponding methods provided by Tomcat

**表 1** 实体外层方法到实体内层方法及其对应的 Tomcat 方法

| WebContainer | | | |
|---|---|---|---|
| SetHttpPort | findHost | removeHost | addHost |
| Connector.setContainer | Engine.findChild | Engine.removeChild | Engine.addChild |
| VirtualHost | | | |
| findApp | deploy | undeploy | |
| Host.findChild | Host.install | Host.removeChild | |
| VirtualApp | | | |
| getContextPath | | getLocalPath | |
| Context.getPath | | Context.getDocBase | |

**Table 2** Outer entities and their corresponding Jetty classes

**表 2** 实体外层与对应的 Jetty 类

| WebContainer | VirtualHost | VirtualApp |
|---|---|---|
| HttpServer | Does not correspond to any specific class, instead its functions are mapped to HttpServlet and HttpContext, respectively. | HttpContext |

PKUAS 中使用框架对 Tomcat 和 Jetty 这两种开源 Web 容器实现的无缝集成,有效地说明了该框架的可行性和灵活性.

## 3.2 性能评测

对 Web 容器的性能测试与对其他应用程序的性能测试存在几点不同:第一,Web 容器是一个平台,它的作用是支撑部署在其上的 Web 应用.通常在用户使用 Web 服务时,其性能主要由两个因素决定:Web 容器本身的性能和 Web 应用的性能.对 Web 容器本身的性能评估应当尽量排除应用程序的因素,但测试一个没有部署任何应用的 Web 容器是毫无意义的.解决方案就是选择尽量简单的应用作为测试对象;第二,Web 容器是 J2EE 应用服务器整体中的一个部分,很多时候它需要和应用服务器中的其他构件进行协同工作,但在测试 Web 容器的时候应该尽量排除其他构件的影响,这就意味着应当选择“纯粹”的 Web 应用,即不需要使用其他构件功能的应用.

最终,本文选择使用北京航空航天大学研发的 SOAP(simple object access protocol)消息中间件——Orientware XLinker[9]中的 TestEcho 作为测试集.TestEcho 是一个非常简单的 Web Service,包含两种方法——add 和 echo;它只需 Web 容器就可运行,没有用到 J2EE 应用服务器中的其他任何构件的功能,很好地满足了前文对测试集的要求.

集成了 Tomcat 的 PKUAS 和独立的 Tomcat 的测试结果分别见表 3 和表 4.

**Table 3** Test results on PKUAS

表 **3** PKUAS 测试结果

| Test name | Run count | Failure count | Min. response time (ms) | Avg. response time (ms) | Max. response time (ms) |
|---|---|---|---|---|---|
| add(int,int) | 4 368 | 48 | 15 | 540 | 9 969 |
| echo(string) | 4 321 | 52 | 0 | 521 | 8 281 |

**Table 4** Test results on Tomcat

表 **4** Tomcat 测试结果

| Test name | Run count | Failure count | Min. response time (ms) | Avg. response time (ms) | Max. response time (ms) |
|---|---|---|---|---|---|
| add(int,int) | 4 635 | 47 | 0 | 532 | 21 312 |
| echo(string) | 4 583 | 52 | 0 | 523 | 21 203 |

以上测试结果表明,集成框架没有对集成的 Web 容器造成明显的性能损失,这说明本文设计的集成框架是成功的.虽然从理论上说,集成框架的抽象层需要额外的处理开销,但实际上,这层抽象层只是介于 PKUAS 应用服务器各模块与集成的 Web 容器之间,而不是位于客户端与 Web 容器之间.客户端在访问 Web 应用时并不需要通过抽象层,而是直接到达 Web 容器进行处理,从而不会造成明显的性能损失.抽象层的额外开销只发生在启动、部署或通过管理工具进行管理的时候,而在这些时候,这种额外的开销是可以忽略不计的.这也就证明了本文提出的集成框架的合理性.

## 4 相关工作

目前,常见的 J2EE 应用服务器大体上可以分为开源和非开源两大类.非开源的商业应用服务器主要有 BEA WebLogic,IBM WebSphere 等;而开源的应用服务器则以 JBoss 为代表.

非开源的应用服务器通常使用专用的 Web 容器.由于没有代码,我们无法深入了解它们的 Web 容器实现方式.但可以肯定的是,用户是无法更换其中的 Web 容器的.如果相关的标准升级,用户只能等待厂商提供相应的升级包.

开源的应用服务器当以 JBoss 为代表,它是目前应用最广的开源 J2EE 应用服务器.JBoss 同样也是采用了集成第三方 Web 容器的方案.它所集成的 Web 容器包括 Tomcat 和 Jetty,用户可以根据实际需要进行选择.另外,JBoss 中对集成的 Web 容器的配置文件也已经统一到了 JBoss 自身的配置机制中.但是,JBoss 缺乏统一的管理工具,对集成的 Web 容器的管理仍需通过相应的 Web 容器提供的管理工具来进行.

## 5 结束语

本文分析并指出了传统 J2EE Web 容器实现方法的不足,针对这些不足提出了一种两层结构的 J2EE Web 容器集成框架.然后,面向两种特定的 Web 容器 Apache Tomcat 和 Jetty,给出了该框架在 J2EE 应用服务器 PKUAS 上的实现,并对其进行了性能评测.在此框架下,不同结构的 Web 容器能够被灵活地集成到应用服务器中来,极大地减轻了开发者维护和升级的负担.测试结果表明,该框架不会对 Web 容器造成明显的性能损失.下一步的研究工作主要是在本文提出的框架下集成更多的 Web 容器等,并在这一过程中进一步完善和扩展该框架,以适应更为广泛的需求.

**References**:

[1] SUN Microsystems. Java 2 platform enterprise edition specification. Version 1.4. 2003

[2] SUN Microsystems. Java servlet specification. Version 2.4. 2003. http://jcp.org/en/jsr/detail?id=154

[3] SUN Microsystems. Java 2 platform enterprise edition management specification (JSR-77). 2002. http://jcp.org/jsr/detail/77.jsp

[4] Apache Foundations. The apache tomcat project. 2006. http://jakarta.apache.org/tomcat/

[5] Consulting M. Jetty Java Servlet Server. 2006. http://jetty.mortbay.org/jetty/index.html

[6] SUN Microsystems. Java management extensions (JMX). JSR-000003. 2002. http://jcp.org/en/jsr/detail?id=3

[7] Huang G, Wang QX, Cao DG, Mei H. PKUAS: A domain-oriented component operating platform. ACTA ELECTRONICA SINICA, 2002,30(12A):39−43 (in Chinese with English abstract).

[8] Wang D. Design and implementation of the management framework in PKUAS [MS. Thesis]. Beijing: Peking University, 2004 (in Chinese with English abstract).

[9] Ge S, Hu CM, Du ZX, Wang Y, Lin XL, Huai JP. A Web service-based application supporting environment. In: Proc. of the National Software and Application. 2002. 97−102 (in Chinese with English abstract).

## 附中文参考文献:

[7] 黄罡,王千祥,曹东刚,梅宏. PKUAS:一种面向领域的构件运行支撑平台.电子学报,2002,30(12A):39−43.

[8] 王栋.构件运行支撑平台 PKUAS 中管理框架的设计和实现[硕士学位论文].北京:北京大学,2004.

[9] 葛声,胡春明,杜宗霞,王勇,林学练,怀进鹏.面向 Web Service 中间件的应用支撑环境.见:全国软件与应用学术会议论文集.2002.97−102.

林泊(1983－),男,硕士生,主要研究领域为中间件技术.

黄罡(1975－),男,副教授,主要研究领域为中间件技术,软件工程.

周明辉(1974－),女,博士,讲师,主要研究领域为分布计算,软件工程,可信计算.

梅宏(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,软件复用和软件构件技术,分布对象技术.

刘天成(1978－),男,博士生,主要研究领域为中间件技术,软件工程.