基于J2EE的企业信息系统解决方案

The solution based on J2EE Enterprise information system

林颖贤 （集美大学信息工程学院 361021）
林大滨 （厦门经济信息中心 360000）

摘要：J2EE平台是由一整套服务(Services)、应用程序接口(APIs)和协议构成，它对开发基于Web的多层应用提供了功能支持。本文首先分析支撑J2EE的核心技术：JDBC、JNDI、EJB、JSP、servlets的特点及实现方法，提出基于J2EE的企业信息系统解决方案。

关键词：J2EE技术 JBOSS应用服务器

1 引言

Java最初是在浏览器和客户端机器中粉墨登场的，当时，很多人质疑它是否适合做服务器端的开发，现在，随着对Java2平台企业版(J2EE)第三方支持的增多，Java被广泛接纳为开发企业级服务器端解决方案的首选平台之一。

二层化应用通常被称为client/server应用，是大家谈论最多的。服务器提供的唯一服务就是数据库服务。在这种解决方案中，客户端程序负责数据访问、实现业务逻辑、用合适的样式显示结果、弹出预设的用户界面、接受用户输入等。client/server结构通常在第一次部署的时候比较容易，但难于升级或改进，而且经常基于某种专有的协议-通常是某种数据库协议。它使得重用业务逻辑和界面逻辑非常困难，更重要的是，在Web时代，二层化应用通常不能体现出很好的伸缩性，因而很难适应Internet的要求。

Sun设计J2EE的部分起因就是想解决二层化结构的缺陷。于是，J2EE定义了一套标准来简化N层企业级应用的开发。它定义了一套标准化的组件，并为这些组件提供了完整的服务。J2EE还自动为应用程序处理了很多实现细节，如安全、多线程等。

用J2EE开发N层应用包括将二层化结构中的不同层面切分成许多层。一个N层化应用能够为以下的每种服务提供一个分开的层：

(1) 显示：在一个典型的Web应用中，客户端机器上运行的浏览器负责实现用户界面。

(2) 动态生成显示：尽管浏览器可以完成某些动态内容显示，但为了兼容不同的浏览器，这些动态生成工作应该放在Web服务器端进行，使用JSP、Servlets，或者XML(可扩展标记语言)和XSL(可扩展样式表语言)。

(3) 业务逻辑：业务逻辑适合用Session EJBs(后面将介绍)来实现。

(4) 数据访问：数据访问适合用Entity EJBs(后面将介绍)和JDBC来实现。

(5) 后台系统集成：同后台系统的集成可能需要用到许多不同的技术，至于何种最佳需要根据后台系统的特征而定。

事实上，多层方式可以使企业级应用具有很强的伸缩性，它允许每层专注于特定的角色。例如，让Web服务器负责提供页面，应用服务器处理应用逻辑，而数据库服务器提供数据库服务。由于J2EE建立在Java2平台标准版(J2SE)的基础上，所以具备了J2SE的所有优点和功能。包括"编写一次，到处可用的可移植性、通过JDBC访问数据库、同原有企业资源进行交互的CORBA技术，以及一个经过验证的安全模型。在这些基础上，J2EE又增加了对EJB（企业级Java组件）、Java servlets、Java服务器页面(JSPs)和XML技

术的支持，

2 J2EE技术

最常用的J2EE技术应该是JDBC、JNDI、EJB、JSP和servlets，图1表示了在一个分布式应用中，J2EE技术的各个方面通常在何处发挥作用。

2.1 Java Database Connectivity (JDBC)

JDBC API以一种统一的方式来对各种各样的数据库进行存取。和ODBC一样，JDBC为开发人员隐藏了不同数据库的不同特性。另外，由于JDBC建立在Java的基础上，因此还提供了数据库存取的平台独立性。JDBC定义了4种不同的驱动程序：

(1) JDBC-ODBC Bridge：在JDBC出现的初期，JDBC-ODBC桥显然是非常有实用意义的，通过JDBC-ODBC桥，开发人员可以使用JDBC来存取ODBC数据源。不足的是，他需要在客户端安装ODBC驱动程序，换句话说，必须安装Microsoft Windows的某个版本。使用这一类型你需要牺牲JDBC的平台独立性。另外，ODBC驱动程序还需要具有客户端的控制权限。

(2) JDBC-native driver bridge：JDBC本地驱动程序桥提供了一种JDBC接口，它建立在本地数据库驱动程序的顶层，而不需要使用ODBC。JDBC驱动程序将对数据库的API从标准的JDBC调用转换为本地调用。使用此类型需要牺牲JDBC的平台独立性，还要求在客户端安装一些本地代码。

(3) JDBC-network bridge：JDBC网络桥驱动程序不再需要客户端数据库驱动程序。它使用网络上的中间服务器来存取数据库，这种应用使得以下技术的实现有了可能，这些技术包括负载均衡、连接缓冲池和数据缓存等。由于第3种类型往往只需要相对更少的下载时间，具有平台独立性，而且不需要在客户端安装并取得控制权，所以很适合于Internet上的应用。

(4) Pure Java driver：第4种类型通过使用一个纯Java数据库驱动程序来执行数据库的直接访问。此类型实际上在客户端实现了2层结构。要在N-层结构中应用，一个更好的做法是编写一个EJB，让它包含存取代码并提供一个对客户端具有数据库独立性的服务。

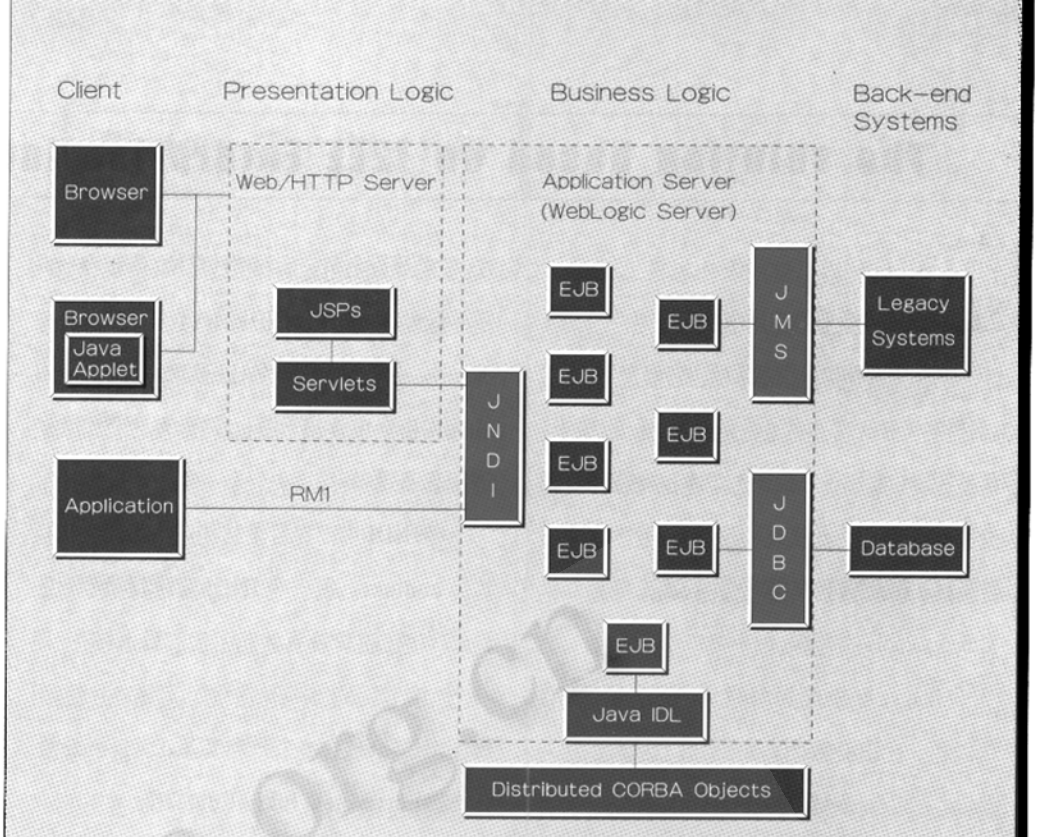

图1 一个N层应用结构的例子

JBOSS服务器为一些通常的数据库提供了JDBC驱动程序，包括Oracle, Sybase, Microsoft SQL Server以及Informix。以下让我们看一个Oracle实例。

2.1.1 JDBC实例

在这个例子中我们假定你已经在Oracle中建立了一个PhoneBook数据库，并且包含一个表，名为CONTACT_TABLE，它带有2个字段：NAME和PHONE。开始的时候先装载Oracle JDBC driver，并请求driver manager得到一个对PhoneBook Oracle数据库的连接。通过这一连接，我们可以构造一个Statement对象并用它来执行一个简单的SQL查询。最后，用循环来遍历结果集的所有数据，并用标准输出将NAME和PHONE字段的内容进行输出。

```
import java.sql.*;
public class JDBCExample
{
public static void main ( String args [] )
{
try
{
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection conn = DriverManager.getConnection
("jdbc:127.0.0.1:", username,p assword);
Statement stmt = conn.createStatement();
String sql = "SELECT name, phone FROM
CONTACT_TABLE ORDER BY name";
ResultSet resultSet = stmt.executeQuery (sql);
String name;
String phone; while (resultSet.next() )
{
name = resultSet.getString (1).trim();
phone = resultSet.getString (2).trim();
System.out.println (name + ", " + phone);
}
}
catch (Exception e)
{
// Handle exception here
e.printStackTrace();
}
}
```

}

2.1.2 JDBC在企业级应用中的应用

以上实例其实是很基本的，它假定了一个2层结构。在一个多层的企业级应用中，更大的可能是在客户端和一个EJB进行通信，该EJB将建立数据库连接。为了实现和改进可伸缩性和系统性能，JBOSS服务器提供了对连接缓冲池Minerva Pools的支持。

Minerva Pools减少了建立和释放数据库连接的消耗。在系统启动以后即可建立这样的缓冲池，此后如再有对数据库的请求，JBOSS服务器可以很简单地从缓冲池中取出数据，数据缓冲池Minerva Pools可以在JBOSS服务器的jboss.jcml和jboss.conf 文件中进行定义。例如要定义与oracle数据库连接，配置连接池如下：

<mbean code= "org.jboss.jdbc.XADataSourceLoader" name= "DefaultDomain:service= XADataSource,name=XmmissPool">
<attribute name= "PoolName">XmmissPool</attribute><attribute name= "DataSourceClass">org.opentools.minerva.jdbc.xa.wrapper.XADataSourceImpl</attribute>
<attribute name="URL">jdbc:oracle:thin:@127.0.0.1:1521:ora8i</attribute>
<attribute name= "JDBCUSER">xmmiss</attribute>
<attribute name= "Password">xmmiss</attribute>
</mbean>

在企业级应用的另一个常见的数据库特性是事务处理，事务是一组申明statement，它们必须做为同一个statement来处理以保证数据完整性，缺省情况下JDBC使用 auto-commit 事务模式，这可以通过使用Connection类的setAutoCommit() 方法来实现。

2.2 Java Naming and Directory Interface (JNDI)

JNDI提供了访问名字与目录服务的统一方式，支持平面名字空间和树式结构，可以存储许多不同类型的对象，JNDI的妙处在于简单性与统一性。知道基本JNDI API调用之后，就可以从各种目录中读出数据，只要有这个目录的JNDI服务提供者。

在JNDI中，在目录结构中的每一个结点称为context。每一个JNDI名字都是相对于context的。这里没有绝对名字的概念存在，对一个应用来说，它可以通过使用InitialContext 类来得到其第一个context: Context ctx = new InitialContext();

应用可以通过这个初始化的context经有这个目录树来定位它所需要的资源或对象，例如，假设你在Weblogic服务器中展开了一个EJB并将home接口绑定到名字 myApp.myEJB，那么该EJB的某个客户在取得一个初始化context以后，可以通过以下语句定位home接口：MyEJBHome home = ctx.lookup("myApp.myEJB");

在这个例子中，一旦你有了对被请求对象的参考，EJB的home接口就可以在它上面调用方法，如果要更进一步地在context中查找对象，JNDI也提供了一些方法来进行以下操作：

(1) 将一个对象插入或绑定到context。这在你展开一个EJB的时候是很有效的。

(2) 从context中移去对象。

(3) 列出context中的所有对象。

(4) 创建或删除子一级的context。

2.3 Enterprise Java Beans (EJB)

J2EE技术之所以赢得某体广泛重视的原因之一就是EJB，它们提供了一个框架来开发和实施分布式商务逻辑，由此很显著地简化了具有可伸缩性和高度复杂的企业级应用的开发，EJB规范定义了EJB组件在何时如何与它们的容器进行交互作用，容器负责提供公用的服务，例如目录服务、事务管理、安全性、资源缓冲池以及容错性。

EJB规范定义了3中基本的bean类型：

(1) Stateless session beans: 提供某种单一的服务，不维持任何状态，在服务器故障发生时无法继续存在，生命期相对较短，例如，一个stateless session bean可能被用于执行温度转换计算。

(2) Stateful session bean: T提供了与客户端的会话交互，可以存储状态从而代表一个客户，典型例子是购物车。Stateful session bean在服务器故障时无法继续生存，生命期相对较短，每一个实例只用于一个单个的线程。

(3) Entity beans: 提供了一致性数据的表示通常存放在数据库中 在服务器故障发生后能继续存在。多用户情况下可以使用EJB来表示相同的数据。对于Entity beans 又分为Container-Managed Persistence 简称CMB 和Bean-Managed Persistence 简称BMB ,entity EJB的一个典型例子是客户的帐号信息。

尽管有以上的区别，所有的EJB还是有许多的共同之处，它们都处理home interface。它定义了一个客户端是如何创建与消亡EJB的，可以在bean中对定义了客户端方法的远程接口进行调用；bean类则执行了主要的商务逻辑。

如果一个EJB已经被开发了或者从第三方进行了购买，它就必须在应用服务器中进行部署发布。在JBoss上部署EJB几乎就象安装和运行应用服务器本身一样简单。我们首先必须在构造ejb-jar.xml文件,我们还必须增加一个名为jboss.xml的部署描述器，把它加入到META-INF目录，它的作用是把EJB名称绑定到一个JNDI树上的名称。部署JBoss EJB简单到只需把jar文件复制到部署目录，该目录就在JBoss安装的根目录的deploy目录之下。

一旦EJB被发布，客户端就可以使用它的JNDI名字来定位EJB，首先，它必须得到一个到home接口的reference。然后，客户端可以使用该接口，调用一个create()方法来得到服务器上运行的某个bean实例的句柄；最后，客户端可以使

用该句柄在 bean 中调用方法。

2.4 JavaServer Pages (JSP)

JSP和ASP是相对应的,但更具有平台对立性。他们被设计用以帮助 Web 内容开发人员创建动态网页，并且只需要相对较少的代码。即使 Web 设计师不懂得如何编程也可以使用 JSP，因为 JSP 应用是很方便的。JSP页面由HTML代码和嵌入其中的 Java 代码所组成。服务器在页面被客户端所请求以后对这些 Java 代码进行处理，然后将生成的 HTML 页面返回给客户端的浏览器。

下面我们来看一个JSP的简单实例。它只显示了服务器的当前日期和时间。

```
<html>
<head>
<title>Sample JSP Page</title>
</head>
<body>
<h1>Date JSP sample</h1>
<h2>
<% response.setHeader( efresh 5); %> The current date is <%= new Date() %>.
</h2>
</body>
</html>
```

支持 JSP 服务器有许多，这里我们介绍 Tomcat，Tomcat 很可能成为下一代 Java Web Server的主流。因为Tomcat受到Sun公司的全力支持，并由非常强大的开发组织apache来进行发展，这一工程被称为 Jakarta 计划。从战略上看，Sun 现在正借助apache的影响来开发server端的java技术，这就是tomcat，因此可以相信tomcat已经或者即将是一个较理想的 jsp & servlet 开发和支撑平台。

2.5 Java servlets

servlet 提供的功能大多与 JSP 类似，不过实现的方式不同。JSP 通常是大多数 HTML 代码中嵌入少量的 Java 代码，而 servlets 全部由 Java 写成并且生成 HTML。servlet 是一种小型的 Java 程序，它扩展了 Web 服务器的功能。作为一种服务器端的应用，当被请求时开始执行，这和 CGI Perl 脚本很相似。Servlets 和 CGI 脚本的一个很大的区别是：每一个CGI在开始的时候都要求开始一个新的进程　而servlets是在servlet引擎中以分离的线程来运行的。因此 servlets 在可伸缩性上提供了很好的改进。

在开发 servlets 的时候，您常常需要扩展 javax.servlet.http.HttpServlet 类，并且 override 一些它的方法，其中包括：

(1) service(): 作为 dispatcher 来实现命令 - 定义方法。

(2) doGet(): 处理客户端的 HTTP GET 请求。

(3) doPost(): 进行 HTTP POST 操作 。

其他的方法还包括处理不同类型的HTTP请求 —可以参考 HttpServlet API 文档。

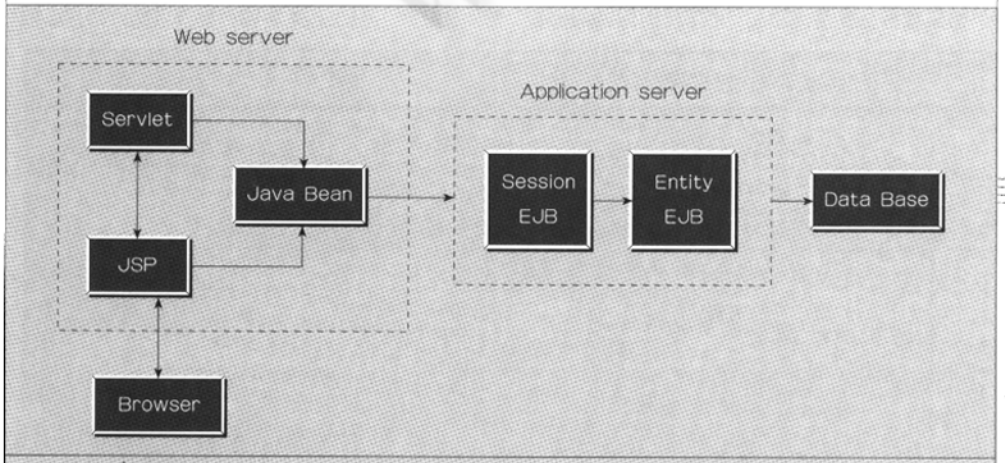

图 2

以上描述的是标准 J2EE Servlet API 的各种方法。

这里要指出的是JSP从应用原理上说实际上就是 servlet,应用服务器首先将 JSP 程序翻译成 servlet 程序，再编译为 java 类提供应用服务。

3 基于 J2EE 的企业信息系统配制方案选择

由上分析可给出如下配制：

操作系统：RedHat Linux 7.0

数据库：oracle 8i 或 MySQL

应用平台: Jboss2.2.2 + Tomcat3.2.2 + Apache

由于商业的 J2EE 应用服务器产品价格不匪(如 IBM 的 WebSphere,BEA 的 WebLogic 等) 如果用上述基于JBOSS的J2EE平台构建企业信息系统是一个既经济又理想的选择，由于J2EE标准规范，要将系统迁于到 WebSphere 或 WebLogic 所有的 java 程序都基本不要变动，只是重新部署整个应用系统即可。

4 总结

电子商务和信息技术的快速发展以及对它们的需求给应用程序开发人员带来了新的压力。必须用更少的金钱、更少的资源来更快地设计、开发企业应用程序。为了降低成本，并加快企业应用程序的设计和开发，J2EE 平台提供了一个基于组件的方法，来设计、开发、装配及部署企业应用程序。J2EE 平台提供了多层的分布式的应用模型、组件再用、一致化的安全模型以及灵活的事务控制。使得基于J2EE的解决方案不会被束缚在任何一个厂商的产品和 API 上。

[参 考 文 献]

1 布雷恩、赖特著，赵明昌译，JSP 数据库编程指南 [M]，北京希望电子出版社。

2 廖若雪，JSP 高级编程 [M]，机械工业出版社，2001。

3 http://java.sun.com/J2EE。

4 http://www.jboss.org。