电子商务技术基础

第三章 商务逻辑层及其技术

第三章 商务逻辑层及其技术

- **第一节 商务逻辑层的构成**
- 第二节 中间件与组件的开发
- 第三节 应用服务器
- 第四节 EJB组件的开发
- 第五节 面向服务的系统开发

第一节 商务逻辑层的构成

- 商务逻辑层描述商务处理过程和商务规则，该层所定义的应用功能是电子商务应用系统开发的重点；提供辅助功能的通用软件，通过与其它软硬件的集成构成支持商务逻辑的商务支持平台。
- 电子商务系统的大系统特征，电子商务系统体系结构的演变，使得商务支持平台承担了大量的商务逻辑处理任务，不但导致应用软件规模和复杂度的增大，而且对系统硬件、网络等集成技术和系统管理都提出了更高的要求。
- 因此从软件实现上，推动了基于组件的分布式系统开发方法和开发技术的应用和发展，从系统管理的需要出发，产生了应用服务器的思想。

第一节 商务逻辑层的构成

- **商务逻辑划分成两个层次**
  - **企业的核心商务逻辑，依靠电子商务应用程序实现。**
  - **支持核心商务逻辑运行的软硬件环境，通过不同的技术产品来集成。**
    - 商务服务平台：CRM, SCM等。
    - 商务支持平台：目录管理、搜索引擎等。
    - 基础支持平台：开发环境工具、系统管理工具、组件与服务集成环境（JDBC,ODBC,EJB,XML等）。
    - Web服务平台：数据库平台、操作系统/计算机硬件与网络基础设施。
- **构成支持平台的技术产品主要包括：Web服务器、商务支持软件、集成与开发工具、计算机主机、网络、其他系统软件（如操作系统、管理工具软件等）。**
  - **通常Web服务器、商务支持软件、部分集成开发工具被集中在一个称之为“应用服务器”的软件包中。**

第一节 商务逻辑层的构成

- 商务逻辑层在物理上可以简化为以下三个部分：
  - 核心商务应用：应用软件（实现商务逻辑）。
  - 应用服务器（为应用软件提供软件支持平台）。
  - 其它支持的软硬件：其他支持软件、计算机主机及网络（为应用软件提供硬件支持平台）。

表 3-1 商务逻辑层的构成

核心商务逻辑应用（企业宣传、网上销售、网络银行等）	
商务服务平台	集成为软件包(应用服务器)
商务支持平台	
基础支持平台	
web 服务器平台、数据库平台	其他支持软硬件
操作系统	
计算机硬件及网络基础设施	

- 从实现上：应用系统设计开发、应用平台的搭建和软硬件系统集成。

第三章 商务逻辑层及其技术

- 第一节 商务逻辑层的构成
- **第二节 中间件与组件的开发**
- 第三节 应用服务器
- 第四节 EJB组件的开发
- 第五节 面向服务的系统开发

第二节 中间件与组件的开发

1、分布处理概述

分布处理出现的背景

- 在企业级应用中，硬件系统集成商基于性能、价格、服务等方面的考虑，通常在同一系统中集成来自不同厂商的硬件设备、操作系统、数据库平台和网络协议等，由此带来的异构性给应用软件的互操作性、兼容性以及平滑升级能力带来了严重问题。
- 随着基于网络的业务不断增多，传统的客户/服务器模式的分布应用方式越来越显示出在运行效率、系统网络安全性和系统升级能力等方面的局限性。

第二节 中间件与组件的开发

1、分布处理概述

什么是分布处理

- **分布处理/分布式处理**：指网络中两个或两个以上的相同或不同的软件相互共享信息资源或共同处理任务，这些软件可以位于同一台计算机中，也可以部署在网络中的任一结点位置，基于分布计算模型的软件系统具有均衡系统负载、共享网络资源的技术优势。
- **分布式处理系统**是将不同地点或具有不同功能或拥有不同数据的多台计算机用通信网络连接起来，在控制系统的统一管理控制下，协调地完成信息处理任务的计算机系统。
  - 分布式处理系统包含硬件，控制系统，接口系统，数据，应用程序和人等六个要素。
  - 控制系统包含了分布式操作系统，分布式数据库以及通信协议等。

第二节 中间件与组件的开发

1、分布处理概述

分布处理的特点和优势

- 统一性：使用一个统一的操作系统；
- 共享性：所有的分布式系统中的资源（物理的或信息的）是共享的；
- 透明性：用户不知道分布式系统是运行在多台计算机上，在用户眼里分布式系统中的许多计算机就像是一台计算机，对用户来讲是透明的；
- 独立性：多个主机都处于平等地位，在逻辑上或物理上是独立存在的；
- 经济性：以较低的成本获得较高的运算性能；
- 可靠性：多个独立的CPU可以支持整个系统的正常工作，适用于核电站等高可靠的环境；
- 异构性：系统中包含的节点可以由不同的计算与通信硬件组成，组成系统的软件可以包括不同的编程语言和开发工具；
- 安全性：只有特许用户可访问敏感数据或执行关键操作；
- 开放性：大多数分布式系统是开放的，因为在系统运行期间，可以添加或改变节点、组件和应用程序。

第二节 中间件与组件的开发

2、中间件(middleware)

中间件出现的背景

- 在大型的电子商务系统中
  - 存在多种硬件系统平台；
  - 在这些硬件系统上存在各种电子商务应用软件和用户界面；
  - 这些硬件系统平台可能采用不同的网络协议和网络体系结构来进行连接。
- 为了把这些异构系统集成起来并开发新的应用，提出了中间件的概念。

第二节 中间件与组件的开发

2、中间件(middleware)

什么是中间件

- 中间件：一种独立的系统软件或服务程序，是位于平台(硬件和操作系统)和应用之间的通用服务，具有标准的程序接口和协议。
- 标准的程序接口和协议定义了一个相对稳定的高层应用环境，如果底层的计算机硬件和系统软件进行更新后，只要更新中间件，并保持中间件对外的接口定义不变，电子商务的应用软件不需要任何修改，就可以实现系统的升级。
- 分布式应用软件借助中间件在不同的技术之间共享资源。

第二节 中间件与组件的开发

3、组件(component)

组件技术出现的背景

- 随着分布处理的发展，应用软件的功能、性能、规模和复杂性成倍增长。
- 分布处理强调各种应用软件之间的互操作性，类似于电子产品的即插即用。
- 组件是实现中间件最有效的技术手段。组件规范描述了开发可重用组件及组件间相互通信的标准。按照组件规范，通过重用已有的组件，电子商务的开发者就可以像搭积木一样快速地构造自己的中间件，不仅节省时间和经费，提高工作效率，而且产生的中间件更加规范、更加可靠。

第二节 中间件与组件的开发

3、组件(component)

什么是组件

- 组件：可以重复使用的一段程序，其概念范围小到GUI的按钮，大到文字编辑器或电子表格。
- 组件技术：提高大型软件可重用性的一种技术，其最基本的出发点是通过软件模块化、软件模块标准化，使大型软件可以利用一个个能够重复使用的“软件零件”进行组装，加快系统的开发速度、降低复杂度和成本。
- 组件技术的核心：创建和利用组件。

第二节 中间件与组件的开发

3、组件(component)

组件技术的特点

- 组件技术是从面向对象技术发展而来的，但它是一种更高层次上的对象技术。
- 组件技术独立于语言和面向应用程序，只规定组件的外在表现形式，而不关心其内部实现方法。
- 组件既可用面向对象的编程语言实现，也可用非面向对象的过程语言实现。

第二节 中间件与组件的开发

3、组件(component)

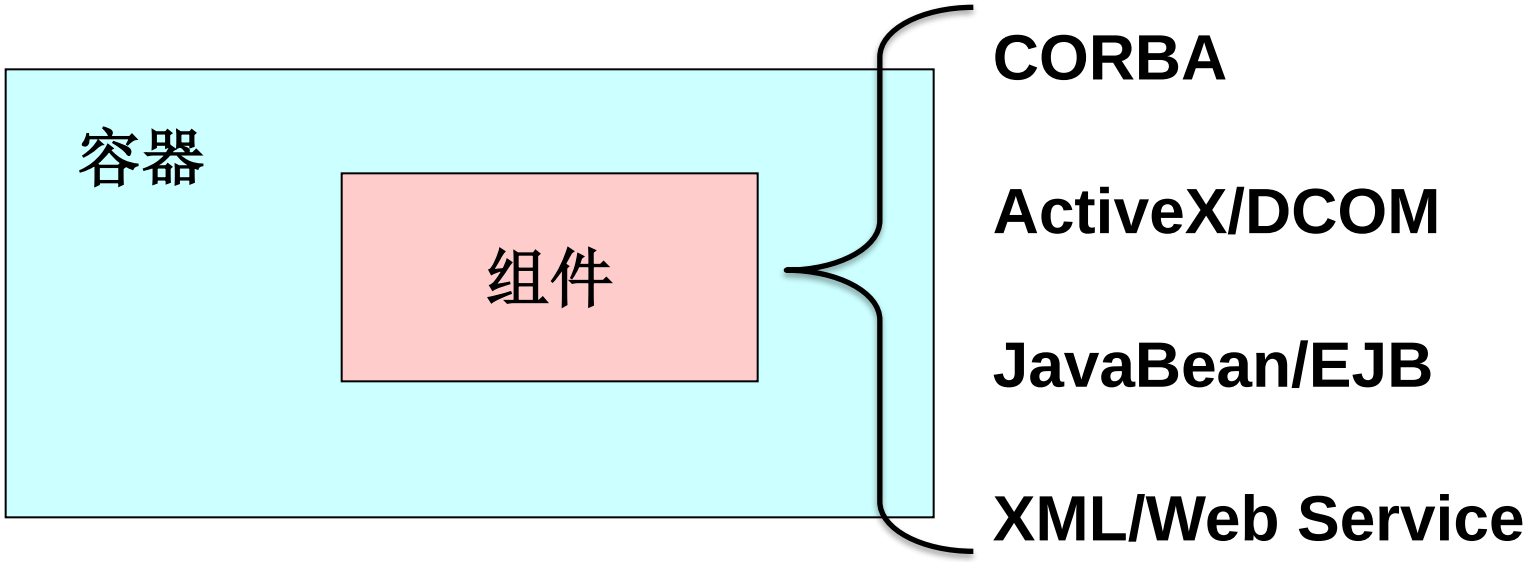
组件模型

- 组件模型由组件与容器两种主要成分构成。
- 组件通过其接口向外界提供功能入口，接口是组件内一组功能的集合，包含的是功能函数的入口，外界通过接口引用或接口指针来调用组件内的功能函数。
- 容器类似于装配车间，是一种存放相关组件的“器皿”，用于安排组件、实现组件间的交互，有多种形式，如表格、页面、框架、外壳等。
- 容器也可以作为另一容器的组件。

第二节 中间件与组件的开发

3、组件(component)

组件技术类型



第二节 中间件与组件的开发

3、组件(component)

CORBA

- 公共对象请求代理体系结构(Common Object Request Broker Architecture, CORBA)，是1991年提出的应用软件体系结构和对象技术规范，其核心是一套标准的语言、接口和协议，以支持异构分布应用程序间的互操作性及独立于平台和编程语言的对象重用。
- CORBA是一种定义分布式组件如何实现互操作的规范，遵照这些规范开发出的分布处理软件环境可以在几乎所有的主流硬件平台和操作系统上运行。

第二节 中间件与组件的开发

3、组件(component)

COM

- 组件对象模型(Component Object Model, COM)是微软公司推出的开放的组件标准，它规定了对象模型和编程要求，使COM对象可以与其他对象相互操作。这些对象可以用不同的语言实现，其结构也可以不同。
- COM规范包括COM核心、结构化存储、统一数据传输、智能命名和系统级的实现（COM库）。
  - COM核心规定了组件对象与客户通过二进制接口标准进行交互的原则。
  - 结构化存储定义了复合文档的存储格式以及创建文档的接口。
  - 统一数据传输约定了组件之间数据交换的标准接口。
  - 智能命名给予对象一个系统可识别的唯一标识。

第二节 中间件与组件的开发

3、组件(component)

DCOM

- 分布式组件对象模型(Distributed Componet Object Model, DCOM) 是一种分布组件对象模型，是COM在分布计算方面的自然延续，为分布在网络不同节点的两个COM组件提供了互操作的基础结构。
- DCOM增强COM的分布处理性能，支持多种通信协议，加强组件通信的安全保障。
- DCOM自动建立连接、传输信息并返回来自远程组件的答复，负责各种组件之间的信息传递，如果没有DCOM，则达不到分布计算环境的要求。

第二节 中间件与组件的开发

3、组件(component)

JavaBean

- JavaBean是一个基于Sun公司的JavaBeans规范的、可在编程工具中被可视化处理的可重用的软件组件，即在Java 编程环境中支持可重用的组件，是一般性的设计方法，适用于客户端或服务器上运行的Java程序。
- 由于许多JavaBean是图形用户界面(GUI)组件，所以JavaBean组件多被视为一种客户端技术，但是并不要求JavaBean都是可视的，并且也可以用于服务器环境中。

第二节 中间件与组件的开发

3、组件(component)

JavaBean

- JavaBean实质是一种符合某些命名和设计规范的Java类，其结构必须满足一定的命名约定，属性必须具有set和get的方法。
- 一个完整的JavaBean在类的命名上需要遵守4项规定：
  - 若类的成员变量的名字是xxx，则为了更改或获取成员变量的值，在类中使用方法getXxx()，获取属性xxx；使用方法setXxx()，修改属性xxx；
  - 对于boolean类型的成员变量，允许使用is代替get和set；
  - 类中方法的访问属性必须是public的；
  - 类中如果有构造方法，那么这个构造方法也是public的，并且是无参数的。

第二节 中间件与组件的开发

3、组件(component)

JavaBean

- 示例

直接访问私有变量sample是被禁止的(因为它被声明为private)，而getSample()提供了对外的安全访问入口。

——面向对象编程中封装的思想

SampleBean.java

```
package bean;
public class SampleBean {
private String sample="Hello，World";
public void setSample(String s)
{   if(s!=null)
      sample=s;
}
public String getSample()
{ return sample;}
}
```

第二节 中间件与组件的开发

3、组件(component)

JavaBean

- JSP网页文件调用JavaBean中间件
  - 在JSP中可以通过JSP指令来访问JavaBean的各种属性。JSP指令是一种特殊标记，用<JSP: >来表示，用于控制JSP引擎的动作。
- 在JSP中共有以下6种JSP指令，其中后三种专用于JavaBean。
  - <jsp:include>
  - <jsp:forward>
  - <jsp:plugin>
  - <jsp:setProperty>：用于为Bean对象中的属性赋值
  - <jsp:getProperty>：用于获取Bean对象中属性的值
  - <jsp:useBean>：在网页中创建一个Bean对象

第二节 中间件与组件的开发

Test1.jsp

```
<%@ page pageEncoding="GB2312" %>
<%@ page contentType="text/html;charset=GB2312" %>
<html>
<jsp:useBean id="myBean" scope="request" class="bean.SampleBean"/>
属性sample的初始值：<jsp:getProperty name="myBean" property="sample"/>
</p>
<jsp:setProperty name="myBean" property="sample" value="Hello，JSP"/>
属性sample的新值：<jsp:getProperty name="myBean" property="sample"/>
</html>
```

SampleBean.java

```
package bean;
public class SampleBean {
private String sample="Hello，World";
public void setSample(String s)
{   if(s!=null)
     sample=s;
}
public String getSample()
{ return sample;}
}
```

第三章 商务逻辑层及其技术

- 第一节 商务逻辑层的构成
- 第二节 中间件与组件的开发
- **第三节 应用服务器**
- 第四节 EJB组件的开发
- 第五节 面向服务的系统开发

第三节 应用服务器

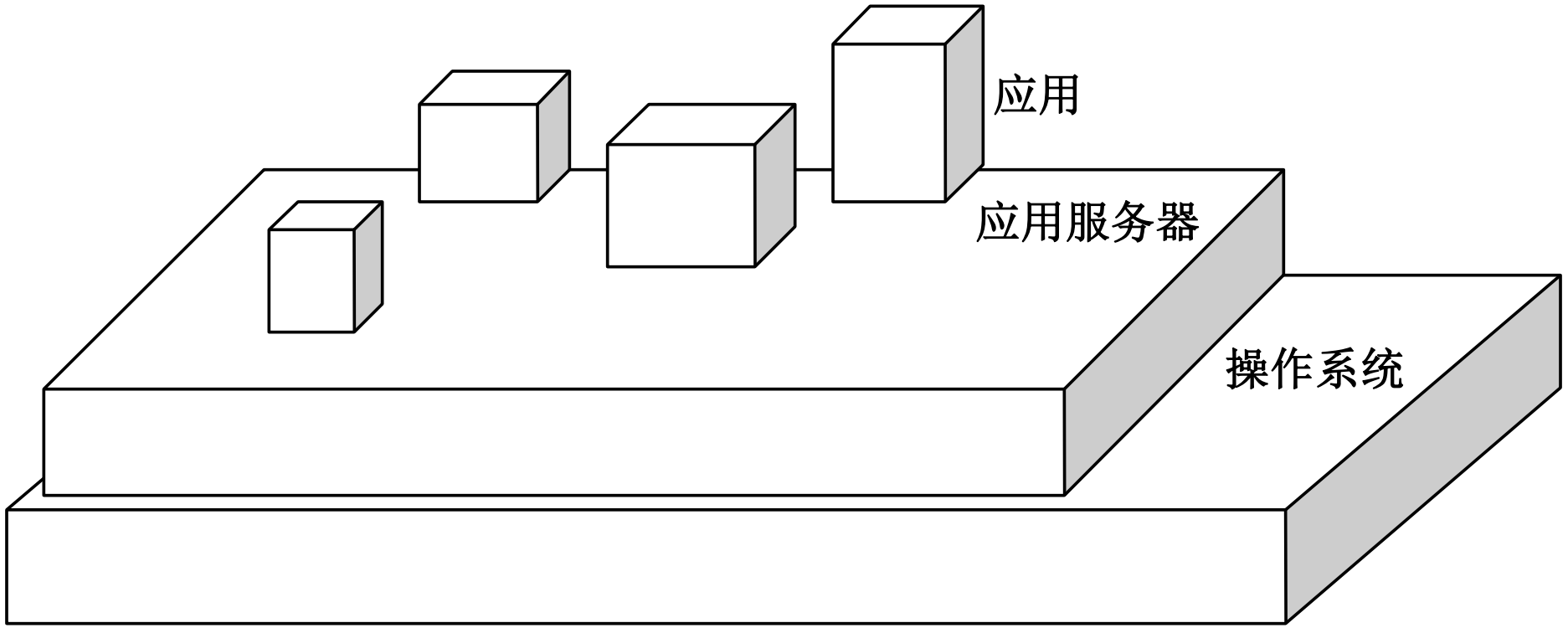
1、应用服务器概述

- 应用服务器是集中了商务逻辑层的WEB服务器、部分商务服务平台软件、商务支持平台软件和基础支持平台中的部分集成开发工具的软件包。
- 应用服务器是数据库与软件或应用程序之间的中间地段（中间层），用于处理用户与企业的业务应用程序和数据库之间的所有应用程序操作。
- 应用服务器是业务应用程序开发和部署的中央位置——没有它，运行业务的软件将处于孤立状态，从而使维护成本更高、升级更复杂，并显著降低性能。
- 应用服务器能提供负载均衡、线程池和服务恢复、Web服务等特性，为分布式的电子商务应用打下了良好基础 。

第三节 应用服务器

1、应用服务器概述

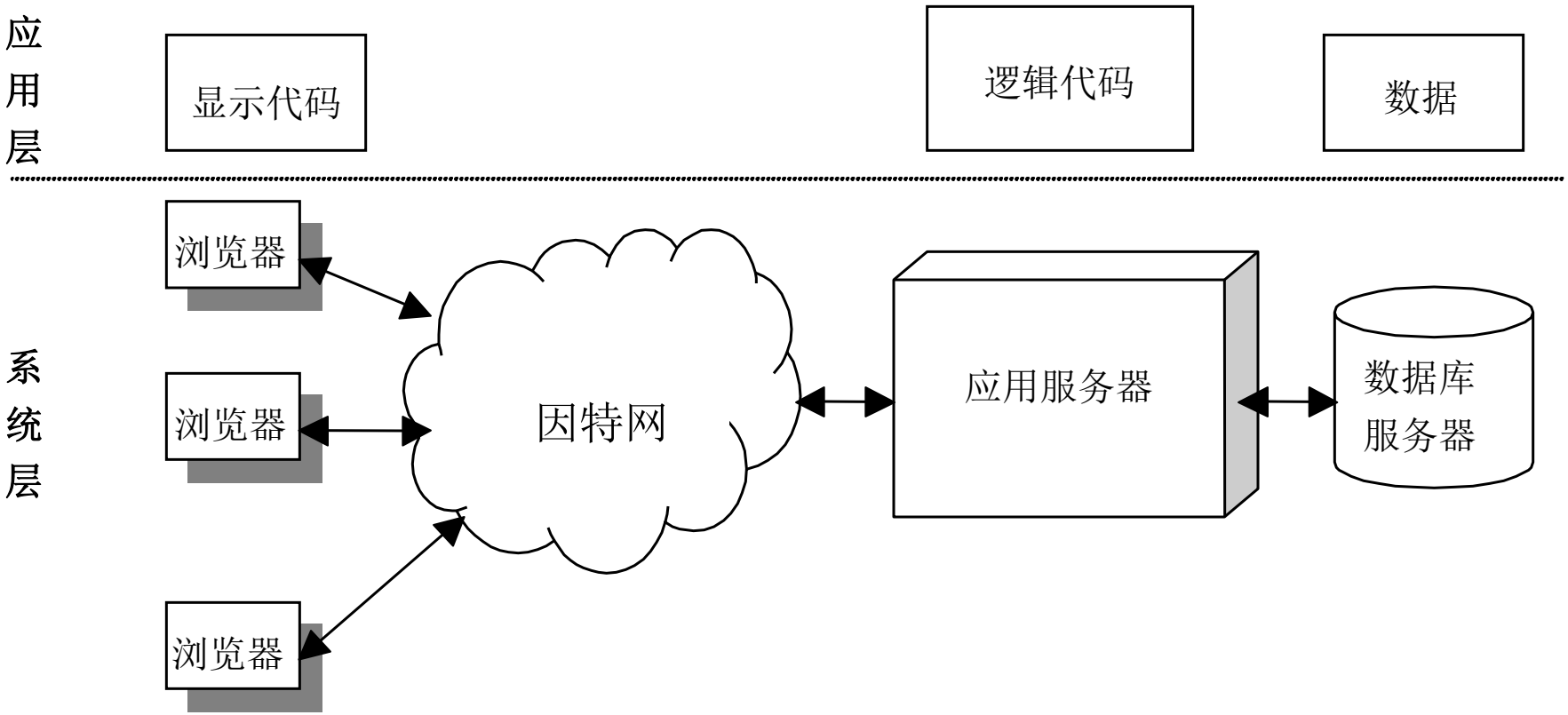
应用服务器的纵向位置



第三节 应用服务器

1、应用服务器概述

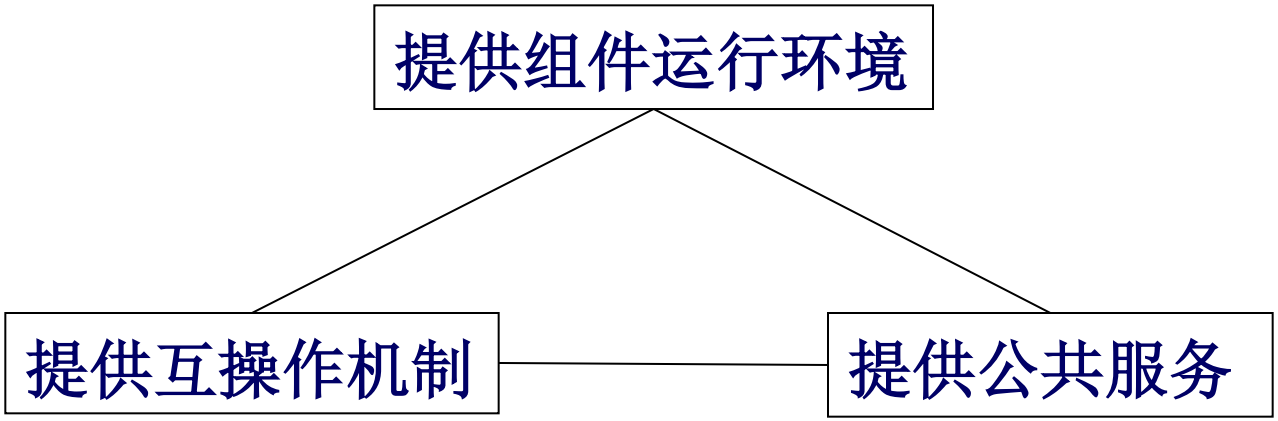
应用服务器的横向位置



第三节 应用服务器

1、应用服务器概述

应用服务器的功能



第三节 应用服务器

1、应用服务器概述

应用服务器的功能

- **提供高性能的应用程序运行环境**
  - **应用服务器一般通过容器为组件提供基本的运行环境。**
  - **具体功能包括：**
    - **内容缓存**：将用户经常访问的HTML页面或WEB处理结果存储在服务器的内存中，可以大大缩短系统响应时间；
    - **数据库连接缓存**：在WEB服务器和数据库服务器之间建立经常性的连接，简化拥护访问数据库的步骤并提高系统的效率；
    - **支持进程的多线程执行**：将一个进程分解成多个可以独立的线程并加以执行，提高应用程序的运行效率，缩短运行时间；
    - **大量用户访问情况下的负载均衡**：根据用户的访问量及服务器的处理能力动态地调整每个服务器的负载，提高系统的可靠性和性能；支持分布式联机处理。

第三节 应用服务器

1、应用服务器概述

应用服务器的功能

- 提供互操作机制
  - 这是针对分布性、异构性所提供的功能。
  - 所有的应用服务器皆提供了很强的高层通信服务。
  - 以屏蔽节点的物理特性，以及各节点在处理器、操作系统等方面的异构性。
  - 具体功能包括：
    - 逻辑层与表达层之间的通信；
    - 逻辑层与数据层之间的通信；
    - 逻辑层内部组件之间的通信等。

第三节 应用服务器

1、应用服务器概述

应用服务器的功能

- 为应用提供扩充性
  - 利用服务器集群技术将系统压力分摊在集群服务器的各个设备上。
  - 增加服务器的CPU以提高系统的处理能力从而应对不断增加的访问压力。
  - 利用动态负载均衡使服务器的性能和访问压力之间得以匹配。
- 提供会话管理
  - 记录和管理客户的每次人机会话过程，提供与商务活动和客户管理相关的信息。

第三节 应用服务器

1、应用服务器概述

应用服务器的功能

- **提供目录管理及内容管理**
  - 目录管理是一种用于集成和管理目录内容，从而对电子商务的采购到结算流程中的关键功能实现自动化的基于 Web 的高效率解决方案， 目录管理可以支持目录内容的访问、转换和集成。 通过目录管理，电子商务企业便能够以一种简单、灵活的方式与客户、供应商、合作伙伴以及员工共享基于目录的内容，以便进一步体现协同商务的优势。
  - 内容是任何类型的数字信息的结合体，可以是文本、图形图像、Web页面、业务文档、数据库表单、视频、声音文件等。
  - 内容管理可以定义为：协助组织和个人，借助信息技术，实现内容的创建、储存、分享、应用、更新，并在企业个人、组织、业务、战略等诸方面产生价值的过程。而内容管理系统就是能够支撑内容管理的一种工具或一套工具的组合。

第三节 应用服务器

1、应用服务器概述

应用服务器的功能

- 提供商务引擎
  - 为电子商务系统提供业务支持的软件产品，常见的如CRM、SCM软件等。
- 提供系统管理
  - 性能配置管理：围绕如何为商务应用配置合理的系统资源，如服务进程数的调整、结果缓存大小的调整等；
  - 存取控制管理：对系统资源的访问权限进行控制，保护特定内容的安全；
  - 系统日志管理：对系统访问、应用运行、存取失败等情况进行记录，从而为系统的故障诊断、分析和性能优化提供依据。

第三节 应用服务器

2、应用服务器的技术演变

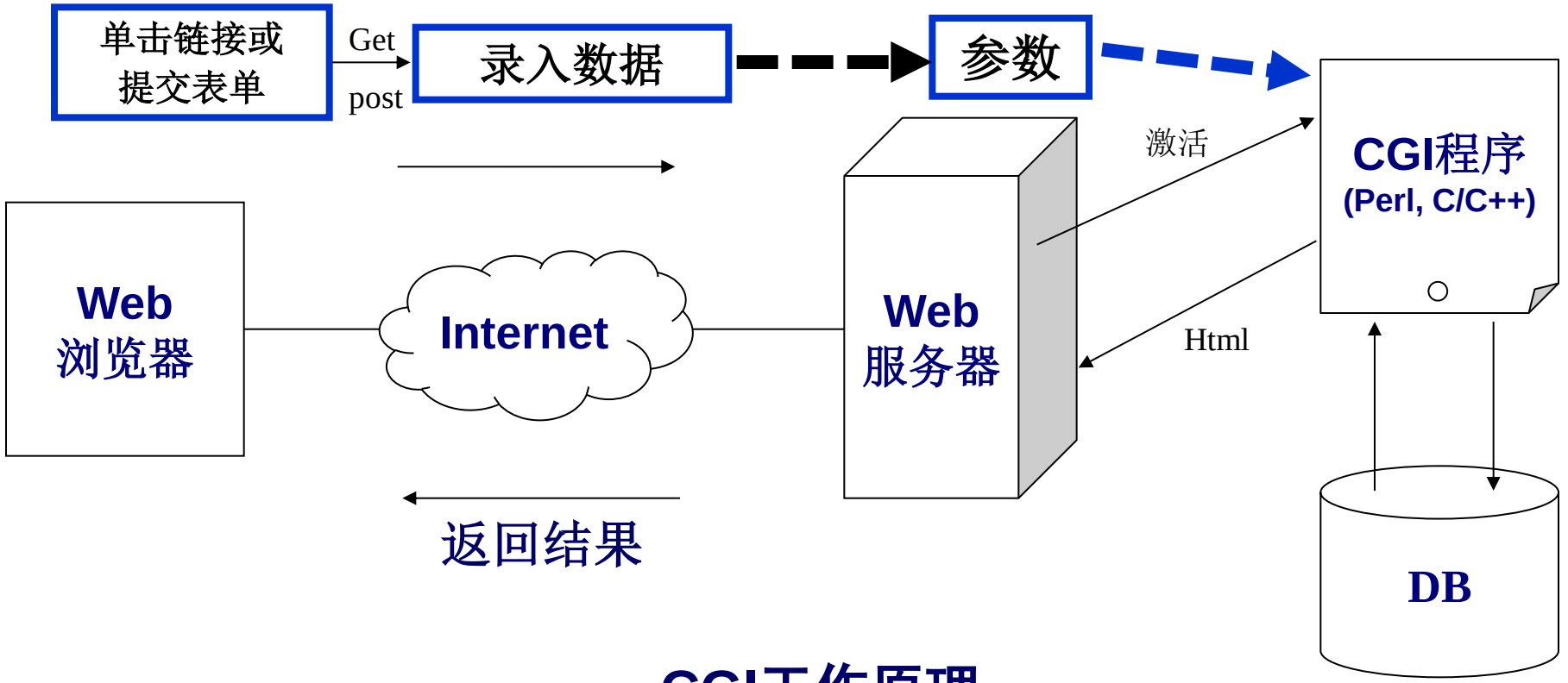
第一代 基于CGI的应用服务器

- 在Web服务器的基础上，添加了运行公共网关接口（Common Gateway Interface, CGI）程序的功能。
- 其基本特征是将HTML代码嵌入到相关的程序代码中。
- 通过CGI，Web服务器能将用户从浏览器中送来的数据，交给CGI程序进行处理，并能将处理的结果再传给浏览器。

第三节 应用服务器

2、应用服务器的技术演变

第一代 基于CGI的应用服务器



CGI工作原理

第三节 应用服务器

2、应用服务器的技术演变

第一代 基于CGI的应用服务器

- 浏览器通过HTML表单或超链接请求一个CGI应用程序的URL，利用HTTP中的GET或POST将客户数据发送到WEB服务器；
- WEB服务器收到请求后激活所指定的CGI应用程序，与此同时利用环境变量或标准输入流的方式将客户数据传给CGI程序；
- CGI应用程序执行所需要的操作（通常是基于浏览者输入的内容）；
- CGI应用程序把结果格式化为网络服务器和浏览器能够理解的文档（通常是HTML网页）；
- 网络服务器（WEB服务器）把结果返回给浏览器。

第三节 应用服务器

2、应用服务器的技术演变

第一代 基于CGI的应用服务器

- CGI的示例1——用C语言编写的一个简单CGI程序

```
#include "stdio.h"
#include "stdlib.h"
void main()
{ char *str1,*str2;
  printf("Content-type:text/html\n\n");
  printf("<html> \n");
  printf("<head><title> 用C编制CGI程序 </title></head> \n");
  printf(" <body> \n");
  printf(" <p>用C编写的CGI程序 </p> \n");
  str1 = getenv( "QUERY_STRING" );
  str2 = getenv( "ACCEPT" );
  printf("<p>query_string:");printf("%s",str1);printf("</p>\n");
  printf("<p>accept:");printf("%s",str2);printf("</p>\n");
  printf(" </body> </html> \n");
}
```

文件的标志，告诉浏览器以html的语法来解析此文件，而不是去下载它。

Getenv是一个C库中的函数，其作用是获取环境变量，getenv(“QUERY_STRING”)为获取GET来的信息，即获取?后的所有内容。

第三节 应用服务器

2、应用服务器的技术演变

第一代 基于CGI的应用服务器

- CGI的示例2——留言本工作实例
  - 用户在客户机上输入留言信息，接着按一下“留言”按钮（到目前为止工作都在客户端）。
  - 浏览器把留言信息传送到服务器的CGI目录下特定的CGI程序中，于是CGI程序在服务器上按照预定的方法进行处理（在本例中就是把用户提交的信息存入指定的文件中）。
  - CGI程序给客户端发送一个信息，表示请求的任务已经结束（此时用户在浏览器里将看到“留言结束”的字样）。
  - 整个留言的过程结束。

第三节 应用服务器

2、应用服务器的技术演变

第一代 基于CGI的应用服务器

- CGI优点
  - 初学者能很容易地构筑功能单纯的系统。
  - Internet上的中小规模Web应用基本上都可以采用CGI的形态。

第三节 应用服务器

2、应用服务器的技术演变

第一代 基于CGI的应用服务器

- CGI缺点
  - CGI程序是将HTML标识嵌入在传统的程序设计语言中，而不像JSP、ASP那样将控制代码嵌入在HTML标识中。所以在CGI程序中，如果要改变HTML的内容，就需要直接修改CGI程序，维护工作变得非常复杂；
  - CGI存在严重的扩展性问题。每个CGI程序在服务器上执行都会产生一个进程，进程需要占用系统资源，当多个用户并发地访问CGI程序时，产生的多个独立的进程将会耗费服务器上的大量资源，严重时甚至用尽服务器资源，导致系统瘫痪。

第三节 应用服务器

2、应用服务器的技术演变

第二代 基于ASP的应用服务器

- 基于ASP的应用服务器，克服了CGI的缺点，还提供了集成开发的工具和相关的实用组件，通过使用ActiveX控件来实现相关的核心商务逻辑功能，使得开发和发布动态网页变得更为容易。
- 在IIS系统中，ASP利用“插件”和API简化了Web应用程序的开发。

第三节 应用服务器

2、应用服务器的技术演变

第二代 基于ASP的应用服务器

- ASP优点
  - ASP代码可以直接放在HTML中，使用VBScript、JavaScript等简单易懂的脚本语言，程序编制具有灵活性。
  - 使用普通的文本编辑器即可进行编辑设计，无需编译便可在服务器端直接执行，可使用控件和API来访问数据库。
  - ASP的运行速度比CGI快。
  - 能够通过DLL组件扩展其功能，ASP组件支持MS的COM，所以能用多种语言(VB、VC、C++、Java等)编写组件以扩展应用程序。
  - 用户端浏览器支持HTML即可浏览ASP网页，因为ASP所使用的脚本语言均在WEB服务器端执行，不需要在用户端浏览器执行。
  - ASP对使用者的技术基础要求不高，容易上手。

第三节 应用服务器

2、应用服务器的技术演变

第二代 基于ASP的应用服务器

- **ASP缺点**
  - 主要在Windows及IIS服务器下运行，跨平台运行性能较差。
  - 代码逻辑混乱，难于管理。由于ASP是脚本语言混合HTML编程，所以很难看清代码的逻辑关系（有人称其为“面条”式程序）；随着程序的复杂性增加，使得代码的管理十分困难。
  - 代码的可重用性差。由于ASP是面向结构的编程方式，并且混合了大量的HTML标记，所以导致页面原型修改后整个程序都需要修改，大大降低了代码的重用性。
  - ASP是弱数据类型的语言，虽然使用方便，但所造成的出错机率较高。
  - ASP的功能较弱，一些底层操作只能通过组件来完成。
  - ASP缺乏完善的纠错和调试功能。
  - 存在安全问题：基于ASP的IIS存在漏洞、源代码易泄露、易遭到黑客攻击等。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- 基于Java技术的优点：跨平台性、安全性较好。
- 基于Java，易于实现跨平台的应用。
- 支持中间件功能，易于组建分布式的网络应用系统。
- 易于实现网络负载均衡。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- 基于Java的应用服务器的变迁
- 从Servlet到JSP
  - Servlet是把HTML内容嵌入在Java程序中，工作原理和CGI相似，是Java技术对CGI编程的回答，但与平台无关，且内部是以线程的形式提供服务，具备平台无关性和运行效率高的优点。
  - Servlet程序在服务器端运行，以接收来自Web浏览器的HTTP请求，动态地生成响应（可能需要查询数据库来完成这种请求），然后发送包含HTML或XML文档的响应到浏览器。
  - Applet是运行在Web浏览器端的Java程序，Servlet是运行在Web服务器端的Java程序。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

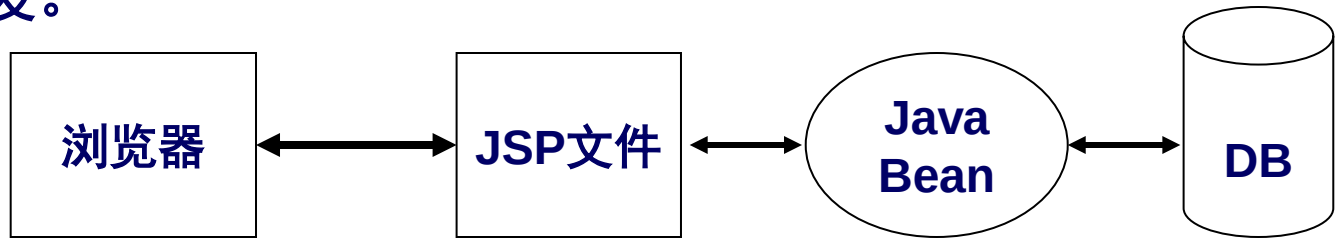
- 基于Java的应用服务器的变迁
- 从Servlet到JSP
  - 与传统的CGI和许多其他类似CGI的技术相比，Servlet具有更高的效率，更容易使用，功能更强大，具有更好的可移植性。但是Servlet的缺点是将HTML标识嵌入在Java程序中，编辑与发布HTML不直观、不方便。
  - JSP是把Java程序嵌入在HTML文件中，在后台编译完成，因此在改变HTML内容时不需要重新编译程序。在单纯的JSP编程模式下，通过应用JSP中的脚本标志，可直接在JSP页面中实现各种功能。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean
  - 随着商务应用复杂性的增加，嵌入HTML的Java代码也越来越庞大和复杂，对这些代码的管理和调试变得很困难。
  - 为了降低代码管理和程序调试的复杂度，将HTML代码和Java代码分离，即将页面显示和商业逻辑处理分开，采用JSP+Bean的形式。
  - JSP负责表达层的编辑，而Bean负责逻辑层的构建。
  - 该模式是JSP程序开发经典设计模式之一，适合小型或中型网站的开发。



第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean
  - JavaBean是Java的可重用组件技术，实质是一种符合某些命名和设计规范的Java类。在程序的开发中，将要进行的业务逻辑封装到这个类中，在JSP页面中通过动作标签来调用这个类，从而执行这个业务逻辑。此时的JSP除了负责部分流程的控制外，大部分用来显示页面，而JavaBean则负责业务逻辑的处理。
  - JavaBean通常封装成具有特定功能或处理某个业务，用以完成一些业务逻辑上的操作，如数据库的连接与访问、用户登录与注销等。

第三节 应用服务器

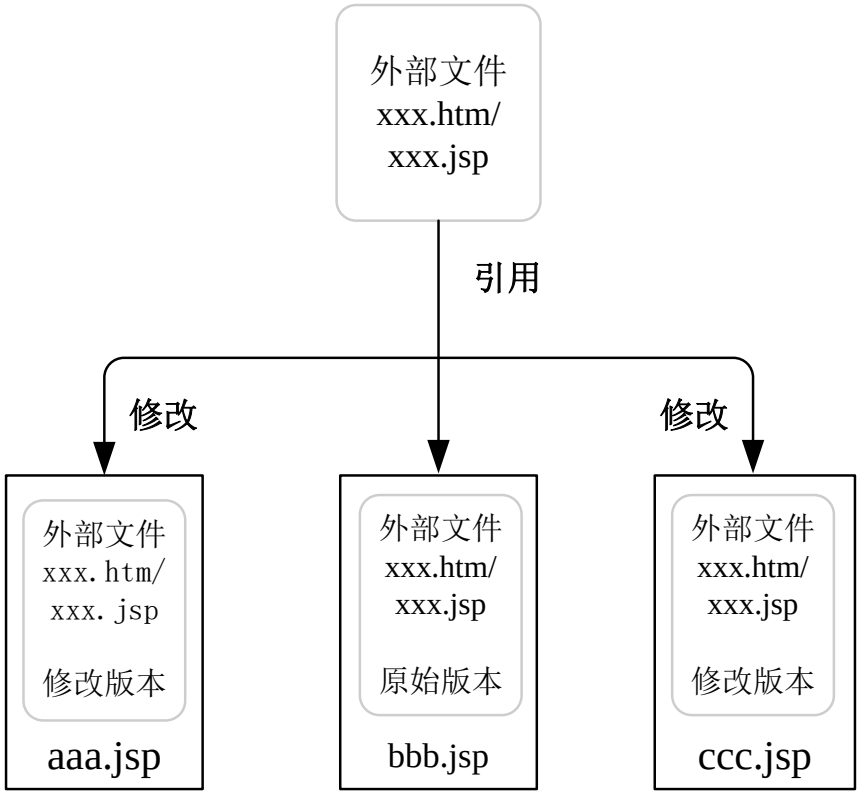
2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean

在JSP的Web应用中集成JavaBean组件：

- 由JavaBean中处理业务逻辑，然后在JSP中调用；
- 而JSP页面着重网页界面的设计；
- 可以实现业务逻辑和页面显示的分离。



第三节 应用服务器

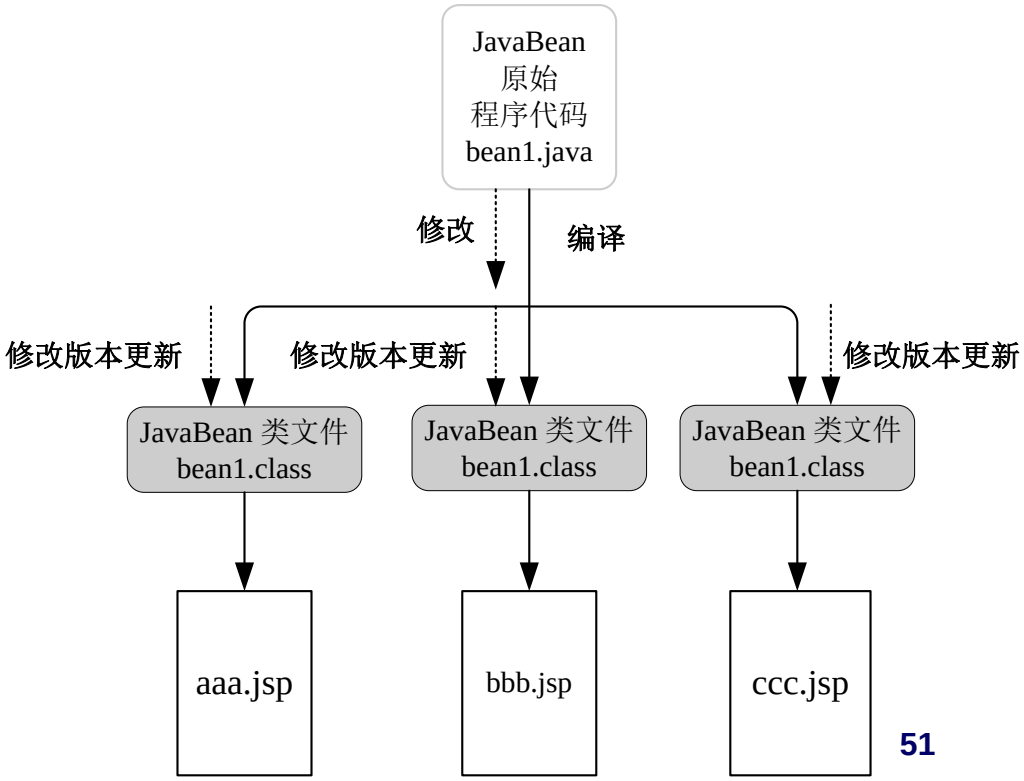
2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean

在JSP的Web应用中集成JavaBean组件：

- 由JavaBean中处理业务逻辑，然后在JSP中调用；
- 而JSP页面着重网页界面的设计；
- 可以实现业务逻辑和页面显示的分离。

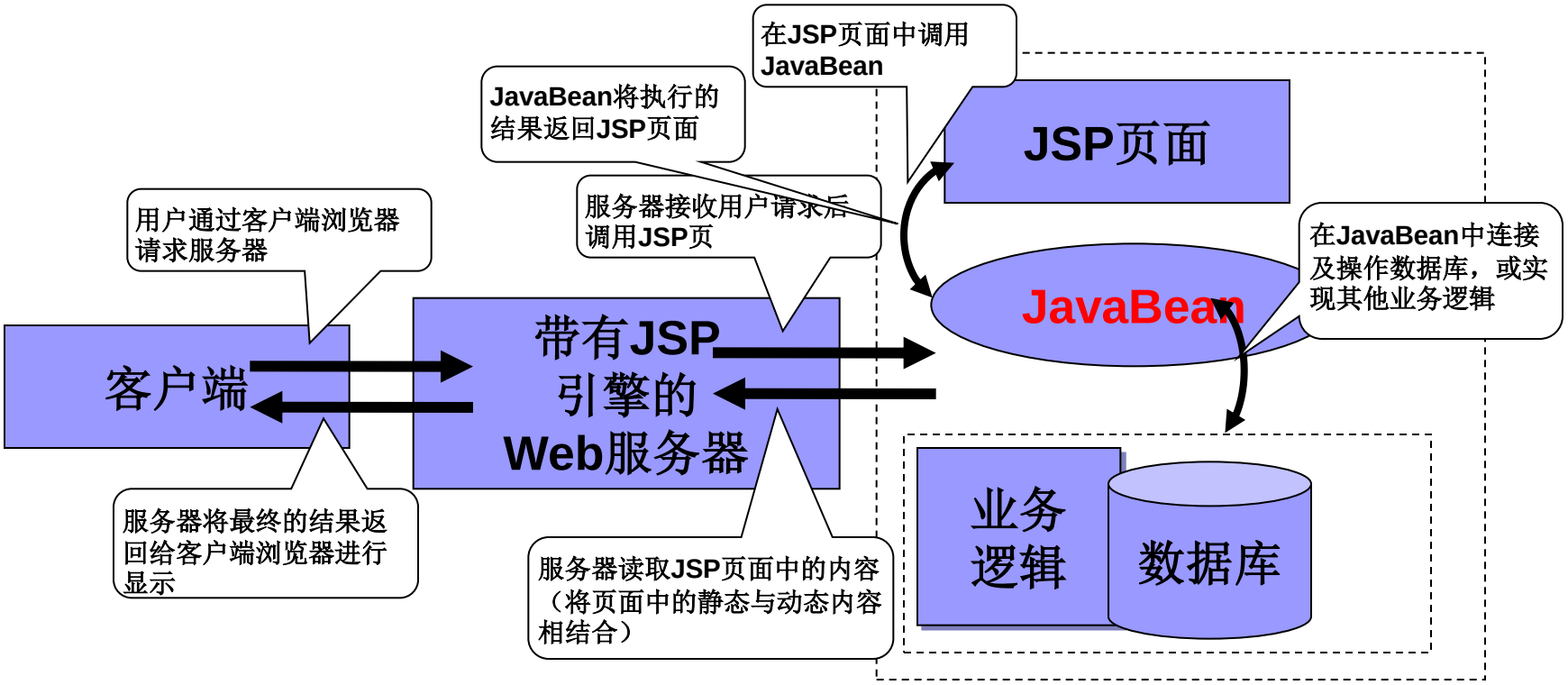


第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean



JSP+Bean模式对客户端请求的处理流程

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean
  - 优点
    - 纯净的JSP页面：因为业务逻辑和数据库操作已经从JSP页面中剥离出来，JSP页面中只需嵌入少量的Java代码甚至不使用Java代码。
    - 可重用的组件：设计良好的JavaBean可以重用，甚至可以作为产品销售，在团队协作的项目中，可重用的JavaBean将会大大减少开发人员的工作量，加快开发进度。
    - 方便进行调试：复杂的操作都封装在一个或者数个JavaBean中，错误比较容易定位。
    - 易维护易扩展：系统的升级或者更改往往集中在一组JavaBean中，而不用编辑所有的JSP页面。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之一(Model 1)：JSP+Bean
  - 缺点
    - 用该模式开发大型项目时，程序流向由一些互相能够感知的页面决定，当页面很多时要清楚地把握其流向是比较复杂的，当用户修改一页时可能会影响相关的很多页面，使得程序的修改与维护变得异常困难。
    - 程序逻辑开发与页面设计混合在一起，既不便于分工合作也不利于代码的重用，程序的健壮性和可伸缩性都不好。
  - 应用范围
    - 该模式的表现逻辑和控制逻辑全部耦合在页面中，这种处理方式对一些规模很小、只有几个简单页面的项目比较适用。

第三节 应用服务器

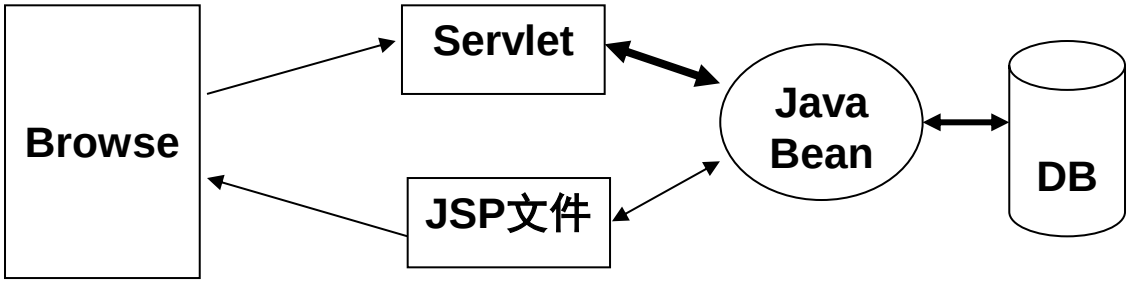
2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之二(Model 2)：

  JSP + Servlet + JavaBean (MVC模式)

  - JSP主要实现页面(表达层)的构建和显示。
  - Servlet主要是实现与用户的交互及控制功能，接受用户的请求，控制JSP来产生响应，即执行业务逻辑并负责程序的流程控制。
  - JavaBean主要实现业务逻辑处理。

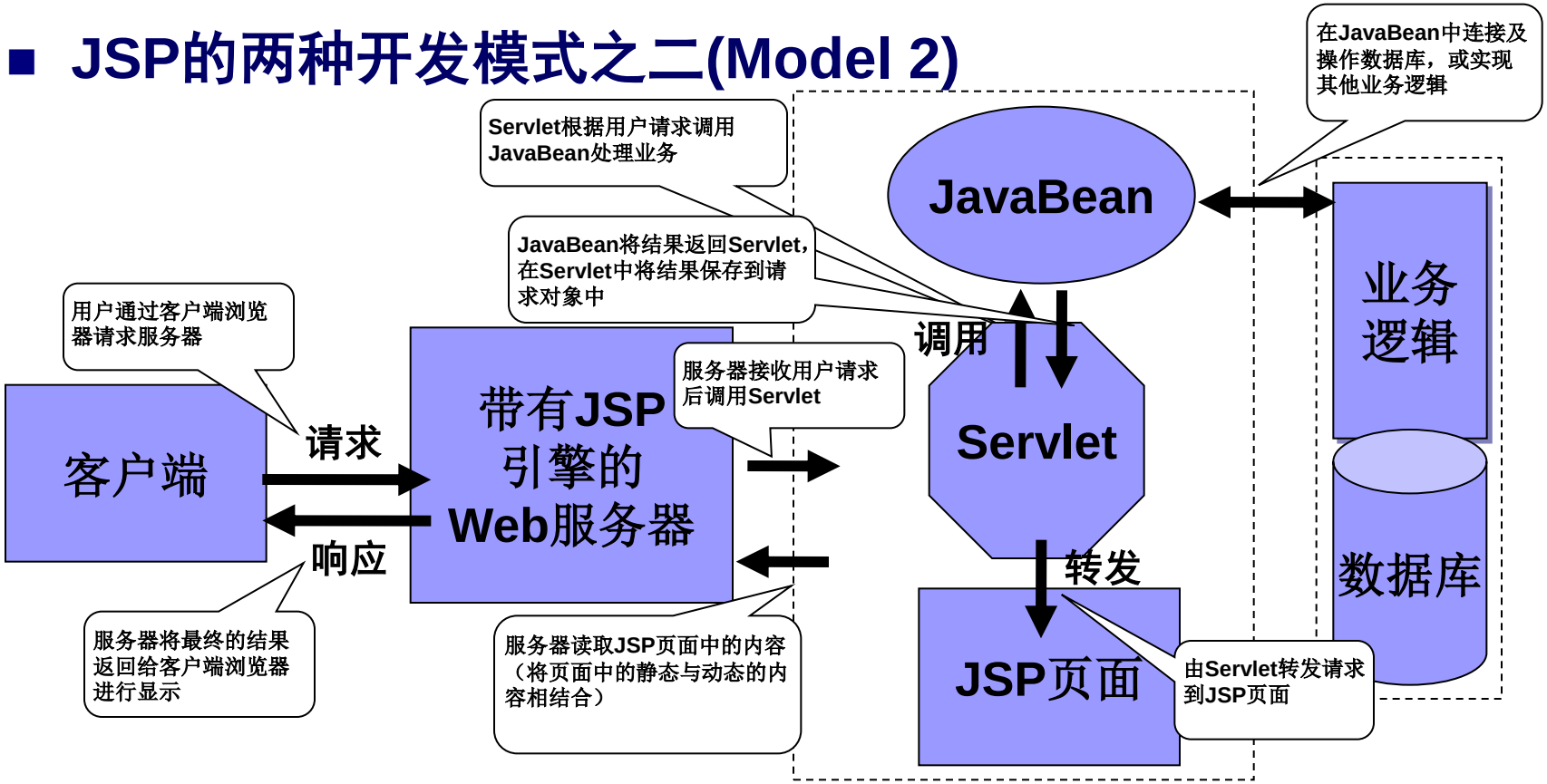


第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式之二(Model 2)



JSP+Servlet+JavaBean模式对客户端请求的处理流程

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

- JSP的两种开发模式
  - JSP Model 1 (JSP+Bean)
  - JSP Model 2 (JSP+Servlet+JavaBean)
- 对于Web的开发者和设计者来说，直接使用JSP是很直观的，但随着代码的增多会使JSP页面设计复杂且调试困难。使用Servlet控制器，大多数的商业逻辑在从JavaBeans传给JSP之前就已经调试通过了。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

JSP的两种开发模式的比较

- 最显著的区别是模型1以“页面为中心"，模型2以"程序为中心"。
  - 如果开发一个典型的Web应用，只是页面之间的链接，则适合选择模型1。
  - 如果每个链接或按钮操作之后需要大量的处理才能确定下一步要显示的内容，则适合采用模型2。
- 考虑应用是面向“请求”的还是面向“响应”的。
  - Servlets是面向“请求”的。
  - JSP更加是面向“响应”的，因为JSP页面将HTML的响应发送给浏览器。如果HTML代码要远远多于Java（或者只有少量的逻辑来决定要显示给用户的内容），则模型1更适合。

第三节 应用服务器

2、应用服务器的技术演变

第三代 基于Java的应用服务器

JSP的两种开发模式的比较

- 考虑请求与响应之间的映射关系。如果对于每一个的请求，只有一个响应，即请求和响应是一对一的，则没有必要使用Servlet。
- 如果每个请求会导致比较复杂的逻辑运算，并且可能返回的视图也不相同，则应使用 Servlet来做出决定和重定向视图，特别是如果需要支持不同的显示格式，例如在同一个通道中使用HTML和XML。Servlet能包含逻辑，来决定客户端是什么，基于这一点来返回不同的文档格式。

第三节 应用服务器

2、应用服务器的技术演变

第四代 基于Java组件和应用服务的应用服务器

- 使用以EJB为中心的服务器端的组件技术，组件与服务是构建分布式应用的关键技术 ；
- 应用服务器具备了三方面的技术：
  - 开发环境集成：能创建新组件、并能将已有组件加以集成；
  - 应用程序的集成：能集成传统的应用程序和新型应用程序；
  - 应用程序的配置功能：由于Web应用程序是分布式地，其组件运行在不同的服务器上，并且有大量的用户对其进行访问，因此需要配置平台的支持，以便在用户剧增时能有效地扩展，并保持系统的稳定。

第三节 应用服务器

2、应用服务器的技术演变

应用服务器发展方向

- 应用服务器技术正朝着面向服务的方向发展，朝着集成化、可扩展的方向发展。
- 功能兼容性：应用服务器会集成越来越多的功能，有的功能是应用服务器厂家自己开发的，有的是第三方开发的，它们组成了一个统一的整体。
- 多种技术兼容性/跨平台：应用服务器向着兼容多种技术标准（如CORBA, DCOM, EJB, RMI, XML, Web Service等）的方向发展，可在多个平台上运行，能连接多种不同的数据库（如Oracle, Sybase, DB2, SQL server, informix, MySql等）。

第三章 商务逻辑层及其技术

- 第一节 商务逻辑层的构成
- 第二节 中间件与组件的开发
- 第三节 应用服务器
- **第四节 EJB组件的开发**
- 第五节 面向服务的系统开发

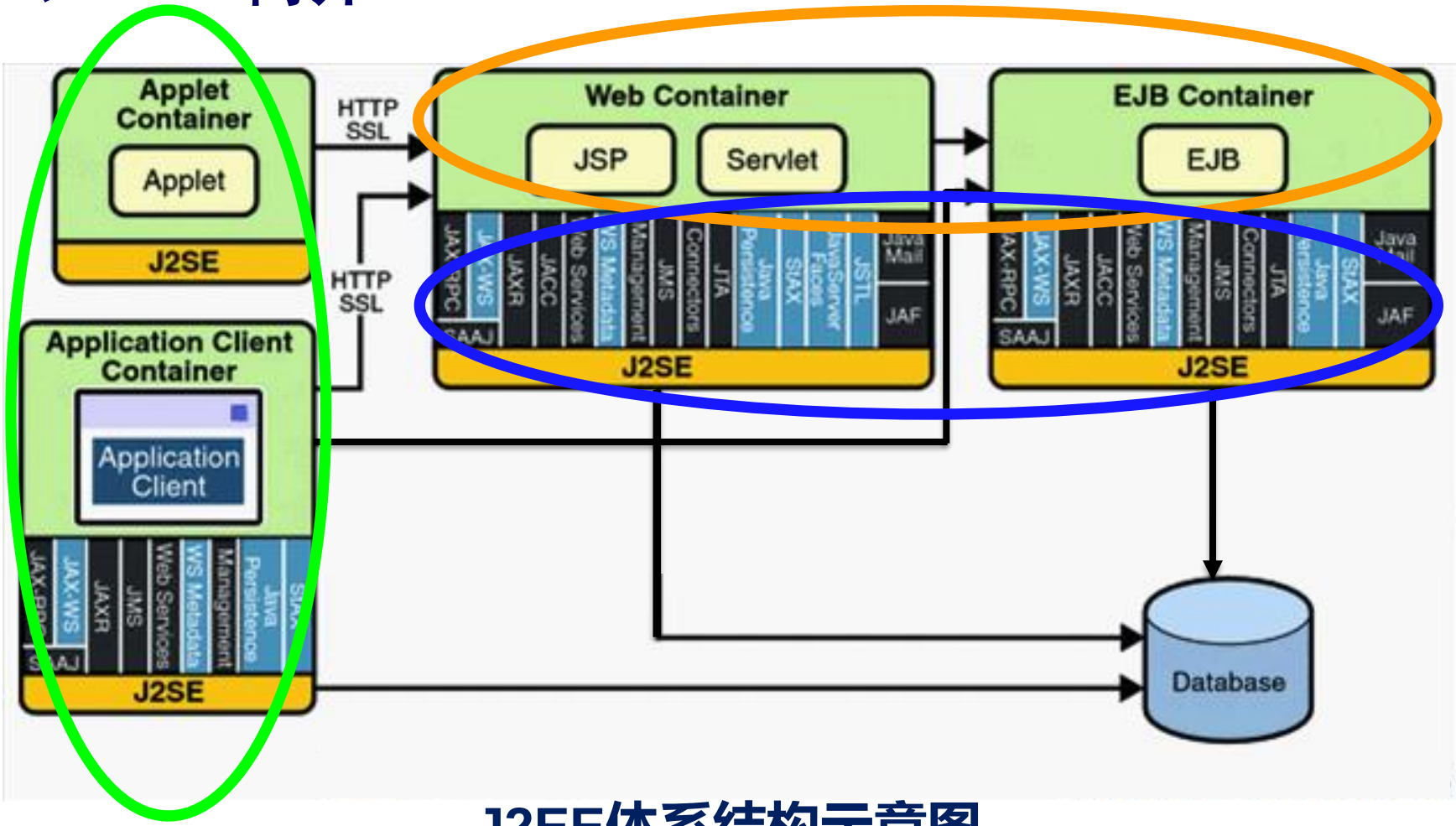
第四节 EJB组件的开发

1、J2EE简介

- 什么是J2EE？
  - J2EE(Java2 Platform, Enterprise Edition)是一个分布式的服务器应用程序设计环境，即一个特殊的Java环境，提供了各应用的运行基础框架环境和一套用来创建应用的Java扩展API。
- J2EE的组成
  - J2EE是很多技术的集合体，其组成包括(X)HTML，JDBC，Servlet/JSP，JMS，JNDI，EJB，XML，Web Service等。
  - J2EE可以分成3个主要应用，即Servlet/JSP，EJB，XML/Web Service 和一些支撑技术，例如JDBC和JNDI。

第四节 EJB组件的开发

1、J2EE简介



J2EE体系结构示意图

第四节 EJB组件的开发

1、J2EE简介

- 绿圈部分
  - Applet容器 (Applet container)
  - 应用程序客户端容器(Application client container)
- 这两部分表示系统的客户端（基于浏览器的Applets和独立的Java应用程序），它们使用HTTP协议与J2EE 平台进行通信。
- J2EE平台的职责是将特定的应用程序URL转换为应用程序组件“地址”并正确地发送命令。

第四节 EJB组件的开发

1、J2EE简介

- 橙圈部分
  - 是开发人员实际编写的业务组件。
  - JSP(Java Server Pages)和Servlet可以直接使用 URL定位，通常用于处理各种类型的输入，查询后端系统的数据然后将结果以HTML 或 XML 格式呈现给发出请求的客户端。JSP和Servlet在Web容器范围内运行，它们再加上图片资源、其他 Java 类(JavaBean)以及可能的配置设置一起构成Web应用程序。
  - EJB(Enterprise JavaBeans)是另一种类型的J2EE业务组件，它与Web应用程序的不同之处在于，它不在Web容器中运行，而是需要驻留在EJB容器中，EJB主要是面向管理数据库事务的，是对关系数据库进行读取和写入操作的。

第四节 EJB组件的开发

1、J2EE简介

- 蓝圈部分
  - 显示了所有主要的J2EE 应用服务器产品都支持的通用API服务层，为Web容器和EJB容器提供相关服务，这两套组件可以使用相同的服务进行工作。
  - 该层中的每一个方框都表示一个标准的Java API，常用的API有：
    - JNDI（Java Naming and Directory Interface，Java命名和目录接口）：用于定位网络上的服务器、目录和其他组件并与它们进行交互。
    - JDBC（Java Database Connectivity，Java数据库连接）：更为常见的 ODBC 的 Java 版本，用于向关系数据库提交查询和读取结果。

第四节 EJB组件的开发

2、J2EE特点

- 简化结构：J2EE平台支持简化的、基于组件开发模型，由于J2EE基于Java编程语言和J2SE平台，提供了“编写一次，随处运行”的可移植性，遵循J2EE标准的所有服务器都支持该模型。
- 支持异构环境：基于J2EE的应用程序不依赖任何特定操作系统、中间件或硬件，因此，设计合理的基于J2EE的程序只需开发一次就可以部署到各种平台，这在典型的异构企业计算环境中是十分关键的。
- 兼容性高：J2EE标准允许客户订购与J2EE兼容的第三方的现成组件，把其部署到异构环境中，节省了由自己制订整个方案所需的费用；因为主要的IT供应用商都采纳EJB体系结构，不同供应商的产品只要符合EJB体系结构，就都是可互操作的。
- 重用性好：由于在EJB模型中，各个软件组件都是严格分离的，因此，可以从现有的软件组件装配出服务器端应用程序，这与从现有的JavaBean可以装配出客户端应用程序一样，使软件能够重用。

第四节 EJB组件的开发

2、J2EE特点

- 可伸缩性强： J2EE平台可以满足在企业系统上进行商业运作的大批新客户，基于J2EE平台的应用程序可被部署到各种操作系统上，可提供更为广泛的负载平衡策略，允许多台服务器集成部署，实现可高度伸缩的系统，满足未来商业应用的需要。
- 提高开发效率：由于组件技术的使用，可以按照开发人员的技能对应用程序开发进行分工，并行开发，提供整体开发效率。例如，图形设计师创建JSP模板，商业逻辑由该领域的专家完成，JSP页面和EJB由Java工程师完成，应用程序的装配和部署由团队中其他的成员完成，其中许多工作可以同时进行，有助于加速应用程序的开发。
- 易于维护：基于组件的设计简化了应用程序的维护。由于组件可以被独立地更新和替代，通过更新应用程序中特定的组件，新的功能可以被很容易地增加。

第四节 EJB组件的开发

2、J2EE特点

- 保护现有投资：J2EE架构可以充分利用用户有的投资，通过允许将现有的信息系统和资产“包裹”在J2EE应用程序中，不要求客户更换现有技术。J2EE拥有广泛的业界支持和一些重要的“企业计算”领域供应商的参与，每一个供应商都对现有的客户提供了不用废弃自己已有投资的升级途径。
- 完整的Web服务支持：J2EE 提供了一个框架，以便在 Java 平台上开发和部署 Web 服务。

第四节 EJB组件的开发

3、EJB组件简介

- EJB（Enterprise JavaBean）是J2EE中的核心技术。
  - 不是一个产品，而是Java服务器端服务框架的规范。
  - 用于开发和部署多层结构的、分布式的、面向对象的Java应用系统的跨平台的组件体系结构。
  - 定义了一个用于开发基于组件的企业多重应用程序的标准。

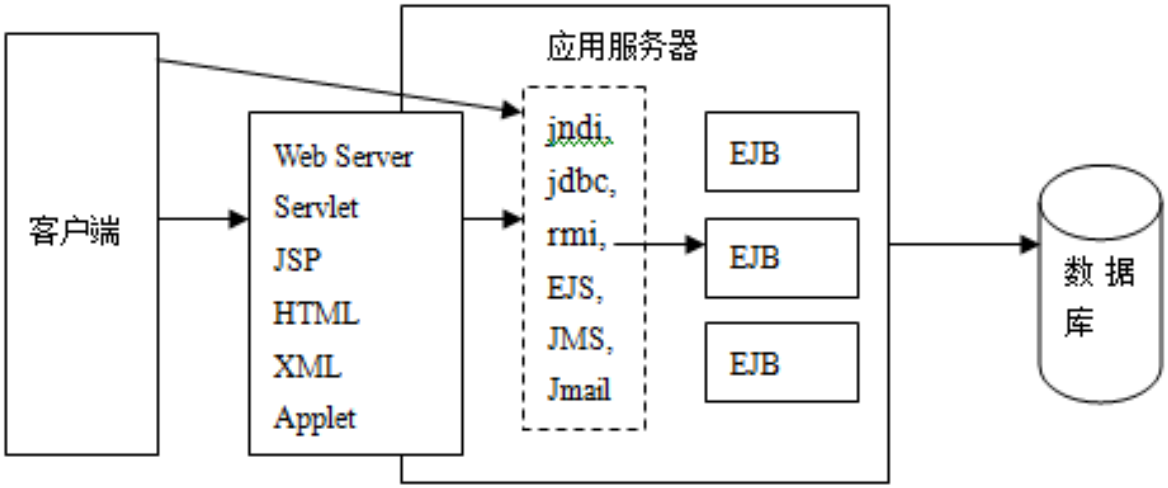


图 3-5　EJB 组件架构模型

第四节 EJB组件的开发

3、EJB组件简介

- EJB定义：用于开发和部署多层结构的、分布式的、面向对象的Java应用系统的跨平台的组件体系结构。
- 采用EJB可以使开发商业应用系统变得容易，应用系统可以在一个支持EJB的环境中开发，开发完成之后部署在其他支持EJB规范的服务器平台上的环境中，随着需求的改变，应用系统可以不加修改的迁移到其他功能更强、更复杂的服务器上，并且具有可扩展性、交互性、多用户安全特性。
- EJB是部署在服务器上的可执行组件或商业对象。
- EJB是设计成运行在服务器上，并由客户端调用的非可视远程对象。

第四节 EJB组件的开发

4、EJB与JavaBean的联系和区别

- EJB与JavaBean的联系
  - EJB与JavaBean都是用一组特性创建，以执行其特定任务的对象或组件。
  - EJB与JavaBean都有从当前所驻留服务器上的容器获得其它特性的能力，因此 Bean的行为根据特定任务和所在环境的不同而有所不同。

第四节 EJB组件的开发

4、EJB与JavaBean的联系和区别

- EJB与JavaBean的区别
  - 用JavaBean创建服务器端应用程序，需要设计整个服务框架。而对于EJB已经提供了服务框架，大大简化了系统开发过程。
  - EJB组件是分布式的，这是与JavaBean最本质的区别。
  - EJB组件模型和JavaBean组件模型是不同的。
    - JavaBean – 可视化组件，JavaBean规范详细解释了组件事件、登记、传递、识别和属性使用、定制和持久化应用编程接口和语义。
    - EJB – 非可视化组件，无用户界面，详细定义可移植的部署Java组件的服务框架模型，其中没有事件处理和属性操作，仅是在运行的通过部署描述符进行描述。

第四节 EJB组件的开发

4、EJB与JavaBean的联系和区别

- EJB与JavaBean的区别
  - JavaBean是构建本地应用的组件，而EJB是构建网络应用的分布式组件。
  - JavaBean一般用于客户端，而EJB一般用于服务器端。
  - JavaBean通常有GUI的支持，而EJB则没有。
  - JavaBean的关注的核心特征是方法、属性、事件、自省和定制，而EJB的关注的核心特征是方法、命名、处理、安全和生命周期。
  - JavaBean实现了设计与运行环境的分离，EJB实现了商业逻辑与容器的分离。
  - JavaBean对应COM的概念，EJB对应的是DCOM的概念。

第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)

- 在EJB规范中定义了六种不同的角色，并提倡构建EJB时分解成不同的部分进行。
- 每一个角色可以由不同的团体(开发商)来担任，而且每个角色所作的工作都必须遵循EJB规范，以保证彼此之间的兼容性。
- 因为每个部分都是各自分别进行的，构建一个企业级部署所需的时间就会显著的减少，而且由于每个团体的分工明确，长期积累的经验可以使得系统的每个部分都达到较高的质量。

第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)

- EJB组件开发者(Enterprise Bean Provider)
  - 技术开发专家，负责开发执行商业逻辑规则的EJB组件，开发出的EJB组件打包成ejb-jar文件。
  - 定义EJB的remote接口和home接口，编写执行商业逻辑功能的EJBclass。
  - 提供部署EJB的部署文件，包括EJB的名字和用到的资源配置，如JDBC等。
  - EJB组件开发者不需要精通系统级的编程，因此，不需要知道系统级的处理细节，如事务、同步、安全、分布式计算等。

第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)

- 应用组合者(Application Assembler)
  - 利用各种EJB组合一个完整的应用系统。
  - 有时需要提供一些相关的程序，如在一个电子商务系统里，应用组合者需要提供JSP程序。
  - 应用组合者必须掌握所用的EJB的home和remote接口，但不需要知道这些接口的实现。
- 部署者(Deployer)
  - 部署者是某个EJB运行环境的专家，负责将包含EJB组件的ejb-jar文件部署到用户的系统环境中。系统环境包含某种EJB服务器和EJB容器。
  - 利用EJB容器提供的工具生成一些类的接口，使EJB容器能够利用这些类和接口在运行状态管理EJB。
  - 安装EJB组件和其他上一步生成的类到EJB容器中。

第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)

- EJB容器提供者(EJB Container Provider)
  - 系统级的编程专家，还要具备一些应用领域的经验。
  - 提供了部署工具为部署好的EJB组件提供运行环境。
  - 提供交易管理、安全管理等服务。
  - 其工作主要集中在开发一个可伸缩的，具有交易管理功能的集成在EJB 服务器中的容器。
  - EJB 容器提供者为EJB组件开发者提供了一组标准的、易用的API访问EJB 容器，使EJB组件开发者不需要了解EJB服务器中的各种技术细节。

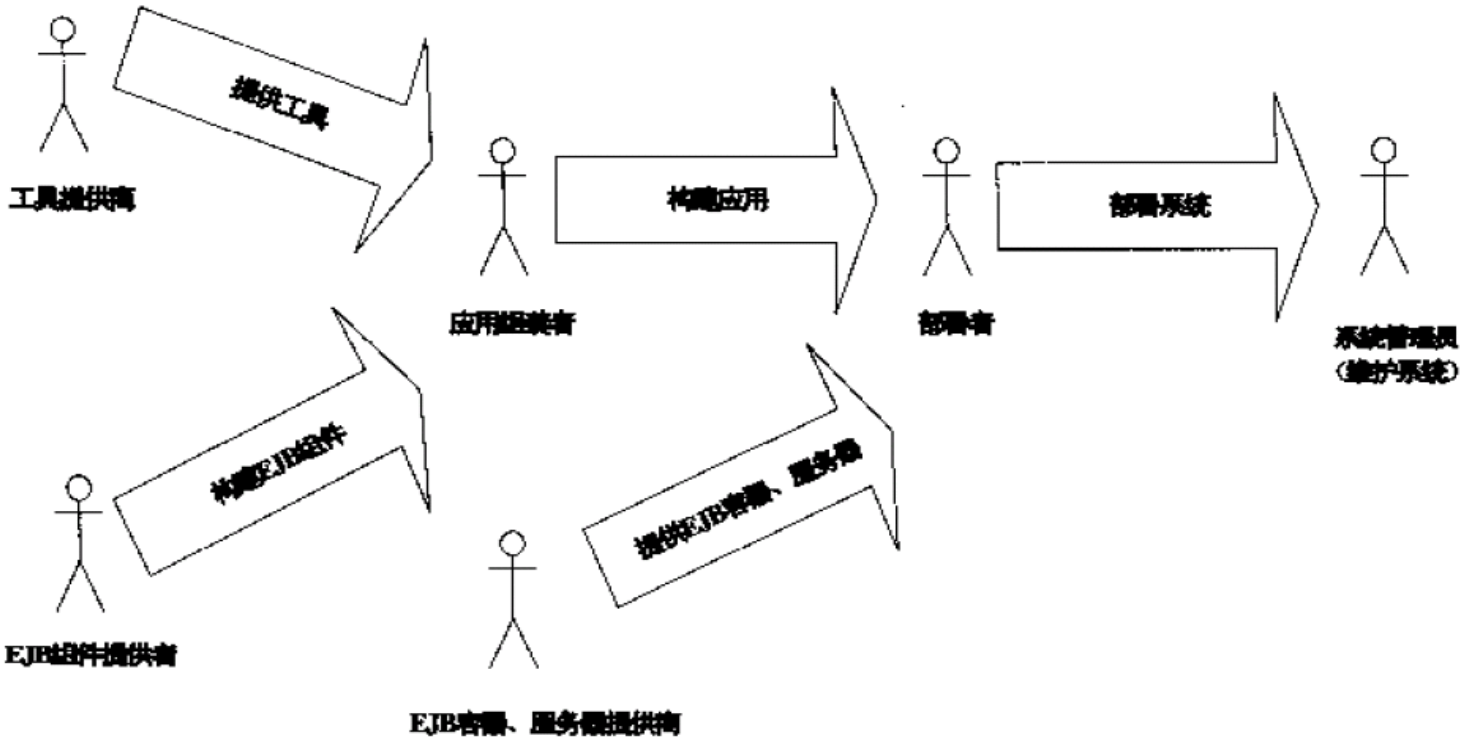
第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)

- EJB服务器提供者(EJB Server Provider)
  - EJB服务器提供者是系统领域的专家，精通分布式交易管理，分布式对象管理及其它系统级的服务。
  - 主要指操作系统开发商、中间件开发商、数据库开发商等。
  - 一般与EJB容器提供者为同一家开发商。
- 系统管理员(System Administrator)
  - 为EJB服务器和容器提供一个企业级的计算和网络环境。
  - 利用EJB 服务器和容器提供的监测管理工具监测EJB组件的运行情况。

第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)



EJB开发体系中的6个角色之间的关系

第四节 EJB组件的开发

5、EJB开发体系中的角色 (6大角色)

基于组件的应用系统开发的组织

	视图	控制器	模型	应用
内容	页面内容和布局	应用流程	商业逻辑	运行环境
角色	页面制作者	应用组装者	组件提供者	WEB管理员
部件	HTML,JSP, XML, Applets	Servlets JavaBeans, XML	JavaBeans, EJB	配置数据 站点使用分析

第四节 EJB组件的开发

6、EJB的类型

- 会话Bean：用于实现业务逻辑，每当客户端请求时，容器就会选择一个会话Bean为客户端服务。会话Bean可以直接访问数据库，但更多是通过实体Bean实现数据访问。
- 会话Bean一般完成一个清晰的解耦的任务，例如注册用户、订单登记、数据库操作。
- 会话Bean基于是否维护过渡状态分为有状态或者无状态。

第四节 EJB组件的开发

6、EJB的类型

- 有状态会话Bean
  - 状态缓存，维护一个跨越多个方法调用的会话状态，在引用期间维护Bean中的所有实例数据的状态值，这些数据在引用期间可以被其他方法所引用，其他用户不会共享同一个Session Bean的实例。
  - 持续性操作，当一个客户端请求一个有状态会话Bean实例时，将会得到一个会话实例，该Bean的状态只为给客户端维持。
  - 有状态会话Bean是暂时的，该状态不会在会话结束，系统崩溃或者网络失败时保留，也可把一个Bean状态保存到文件系统或数据库中。
  - 由于调用时需要维护状态，消耗较多的内存，只有需要维护客户状态时才使用，适合少量用户使用。
  - 典型例子是购物车，当一个客户第一次打开购物车时，系统为他分配一个购物车的会话Bean，在以后，每当客户选购了商品将改变购物车的商品记录，而这些记录数据将保存到用户会话数据中。

第四节 EJB组件的开发

6、EJB的类型

- 无状态会话Bean
  - 状态不保留，没有中间状态，不保持追踪一个方法调用另一个方法传递的信息，因此一个无状态方法的每一次调用都独立于它的前一个调用。
  - 因为不维护状态，所以无状态会话Bean由容器管理。当客户端请求一个无状态的Bean实例时，可以接收来自由容器管理的无状态会话Bean实例集中的一个实例。
  - 在EJB中是最简单的一种Bean。这里的无状态不是指没有状态，而是这些状态被保持在客户端，每次调用时容器只对客户提供业务逻辑，不保存客户端的任何数据状态， 。
  - 能够被共享，适合大量用户同时使用，所以容器可以维护更少数量的实例来为大量的客户端服务。
  - 应存储在服务器端的数据被存储在客户端中，每次调用数据都要以参数的方式传递给Bean，如果是比较复杂的数据集合，则网络需要传递大量数据，造成更多的负载。在客户端维护状态还要注意安全性问题，如果数据状态敏感，则不要使用无状态会话Bean。
  - 典型例子是税费计算。

第四节 EJB组件的开发

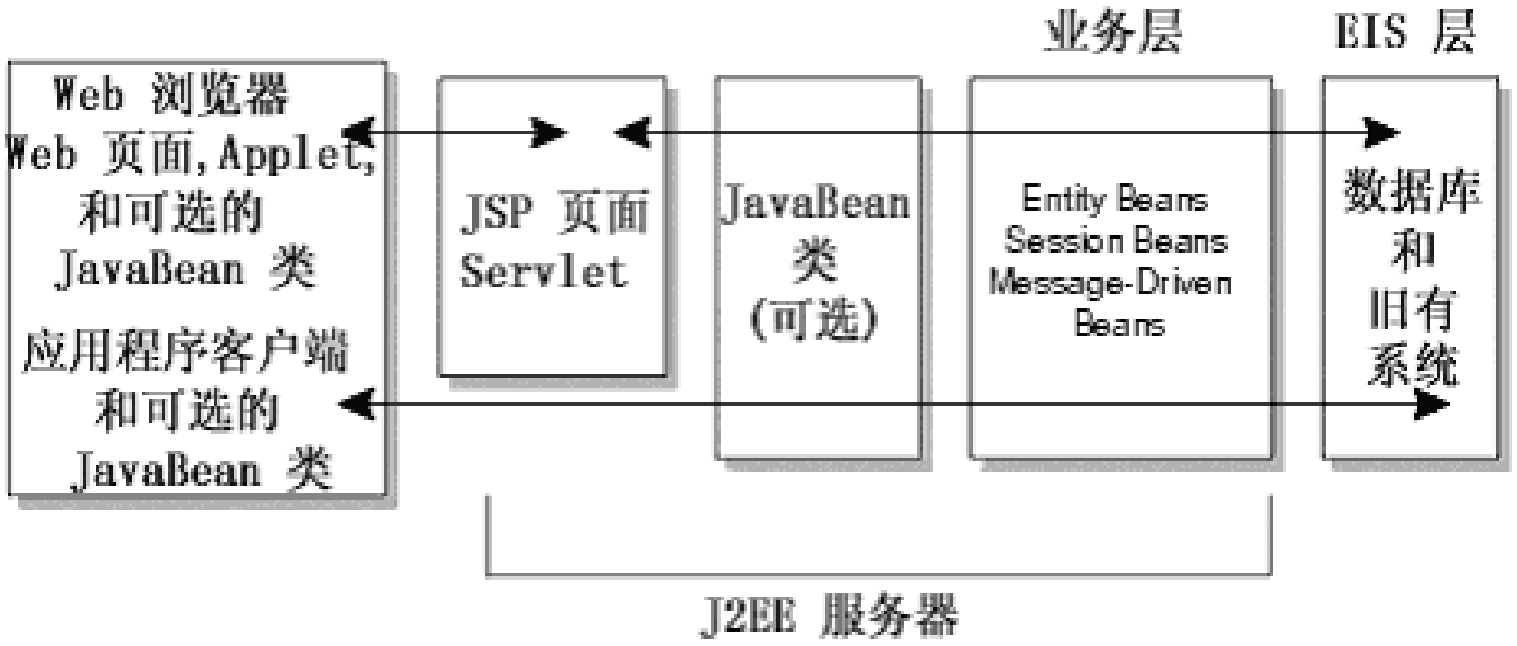
6、EJB的类型

- 实体Bean
  - 管理数据库或另外一个应用系统中持久化数据的一个对象。例如客户表中的一个记录，或者员工表中的一个员工记录。
  - 允许共享访问，可被多个客户端共享。例如一个员工实体能够被多个计算员工每年工资总额或者更新员工地址的客户端使用。
  - 实体Bean对象特定变量能够保持持久化。
- 消息驱动Bean
  - Java消息服务(Java Message Service，JMS)与EJB的集成，处理异步消息。

第四节 EJB组件的开发

6、EJB的类型

业务层组件（EJB）



第三章 商务逻辑层及其技术

- 第一节 商务逻辑层的构成
- 第二节 中间件与组件的开发
- 第三节 应用服务器
- 第四节 EJB组件的开发
- **第五节 面向服务的系统开发**

第五节 面向服务的系统开发

1、SOA系统开发

- 面向服务的系统架构及其特征
- 信息系统（包括电子商务系统）的软件开发，从最初的面向过程、面向对象，到后来的面向组件、面向集成，直到最近的面向服务架构(Service-Oriented Architecture, SOA) 。
- SOA是一个组件模型，将应用程序的不同功能单元即服务(service)，通过服务间所定义的接口和契约联系起来。接口采用中立的方式定义，独立于具体实现服务的硬件平台、操作系统和编程语言，使得构建在这样的系统中的服务可以使用统一和标准的方式进行通信。
  - 它是一种软件系统架构
  - 服务是整个SOA实现的核心
  - SOA具有跨越不同编程语言不同平台的交互能力

第五节 面向服务的系统开发

1、SOA系统开发

- SOA基本特征
  - 服务的封装：将服务封装成用于业务流程的可重用组件的应用程序函数。
  - 服务的重用：可重用性的设计。
  - 服务的互操作：同步或异步机制。
  - 服务是自治的：不需要宿主进行组件存放和管理。
  - 服务之间的松耦合度：服务请求者不知道提供者实现的技术细节，通过消息调用不是API或文件格式。
  - 服务是位置透明的：业务与服务分离，用户不知道服务的具体位置和实现细节。

第五节 面向服务的系统开发

2、Web Service

- Web Service是指由企业发布的完成其特别商务需求的在线应用服务，是封装成单个实体并发布到网络上供其他程序使用的功能集合，其他公司或应用软件能够通过Internet来访问并使用Web Service。
- Web Service是下一代的WWW，它是用于创建开放分布式系统的构件，它允许在Web站点上放置可编程的元素，能进行基于Web的分布式计算和处理；Web Service可以使公司和个人迅速且廉价地向全世界提供其数据服务。

第五节 面向服务的系统开发

2、Web Service

- **Web Service的概念**
  - Web Service是独立的、模块化的应用程序，能够在网络（WWW）上被描述、发布、查找和调用。
  - Web Service是最近制定的一组标准，目的是利用成熟的Web技术，通过SOAP协议、WSDL服务描述语言和UDDI统一描述发现集成协议来实现跨语言、跨平台、跨网络之间的分布处理与组件应用。

第五节 面向服务的系统开发

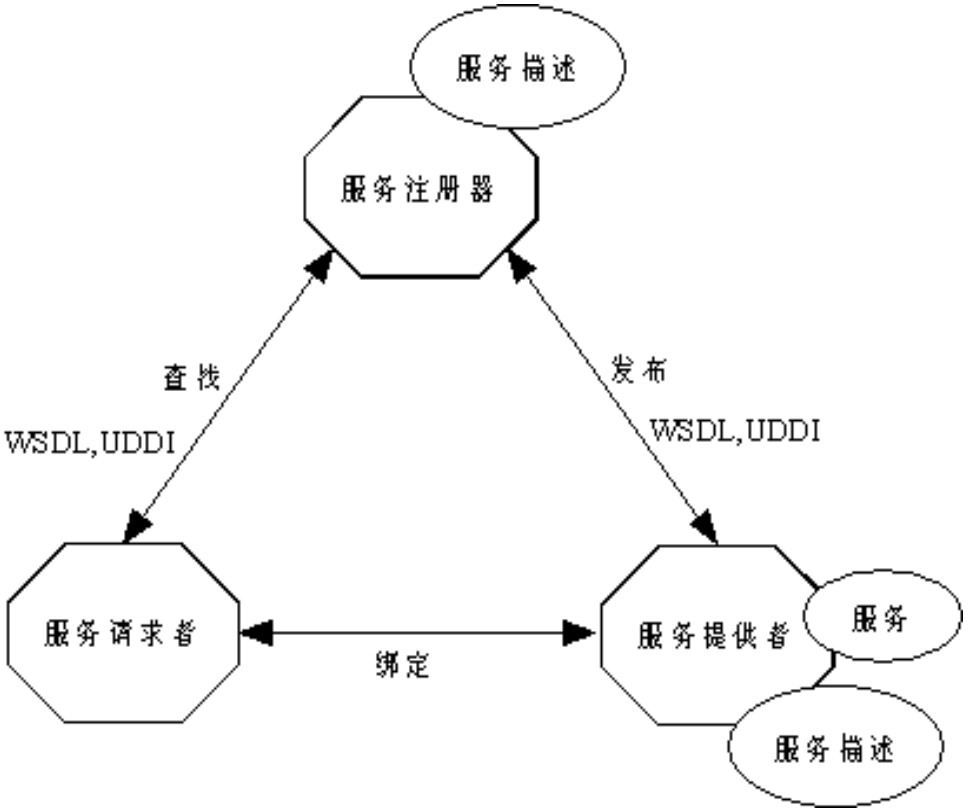
2、Web Service

- Web Service是目前用来实现SOA的常用技术标准，其所涉及技术主要有：
  - XML (可扩展标记语言，Extensible Markup Language)。
  - SOAP (简单对象访问协议，Simple Object Access Protocol)：是一个基于XML的，用于在分布式环境下交换信息的轻量级协议。SOAP在请求者和提供者对象之间定义了一个通信协议。
  - WSDL (Web服务描述语言，Web Services Description Language)：定义了一个 XML词汇表，该词汇表依照请求消息和响应消息，在服务请求者和服务提供者之间定义一种契约。WSDL 是一种XML 文档，在应用开发环境中，使用WSDL将集成服务的流程自动处理到请求者应用程序。
  - UDDI (统一描述、发现和集成，Universal Description, Discovery and Integration)：UDDI为发布服务的可用性和发现所需服务定义了一个标准接口（基于 SOAP 消息）。UDDI 实现将发布和发现服务的SOAP请求解释为用于基本数据存储的数据管理功能调用。

第五节 面向服务的系统开发

2、Web Service

- Web Service的体系结构：包含三个角色（服务提供者、服务请求者、服务代理者）以及三个操作（发布、查找、绑定）。



第五节 面向服务的系统开发

2、Web Service

- 服务提供者通过在服务代理者那里注册来配置和发布服务，服务请求者通过查找服务代理者那里的被发布服务的登记记录来找到服务，服务请求者绑定服务提供者并使用可用的服务。
- 在Web Service中，发布、查找和绑定三个操作使用三种不同的技术。
  - 发布服务使用UDDI(统一描述、发现和集成)；
  - 查找服务使用UDDI 和WSDL(Web Service描述语言)的组合；
  - 绑定服务使用WSDL 和SOAP(简单对象访问控制)。
- 在三个操作中，绑定操作是最重要的，包含了服务的实际使用，也容易发生互操作性问题。由于服务提供者和服务请求者对SOAP规范的支持才解决了这些问题，并实现了无缝互操作性。

第五节 面向服务的系统开发

2、Web Service

- Web Service的特点
  - 封装性：从用户的角度来看，Web Service是部署在WEB上的对象/组件。
  - 互操作性/高度集成性：任何Web Service都可以与其他Web Service进行交互。由于有了SOAP（Simple Object Access Protocol）这个所有主要供应商都支持的新标准协议，因而避免了在CORBA、DCOM 和其他协议之间转换的麻烦。还因为可以使用任何语言来编写 Web Service，因此开发者无需更改其开发环境，就可生产和使用 Web Service。

第五节 面向服务的系统开发

2、Web Service

- Web Service的特点
  - 普遍性：Web Service使用 HTTP 和 XML 进行通信。因此，任何支持这些技术的设备都可以拥有和访问 Web Service。
  - 易于使用：Web Service有来自IBM 和微软这样的供应商的免费工具箱让开发者快速地创建和部署Web Service。此外，某些工具箱还可以让已有的COM组件和JavaBean方便地成为 Web Service。
  - 行业支持：几乎所有主要的软件供应商都支持SOAP 和周边Web Service技术。

本章小结

- 商务逻辑层的两大层次构成
- 应用服务器的概念及其技术演变
- JSP的两种开发模式
- 分布式处理系统、中间件的概念
- 组件的概念、JavaBean组件
- EJB的定义、EJB的6大类角色、EJB的类型
- SOA的基本特征
- Web Service的特点及相关技术

Thank You !