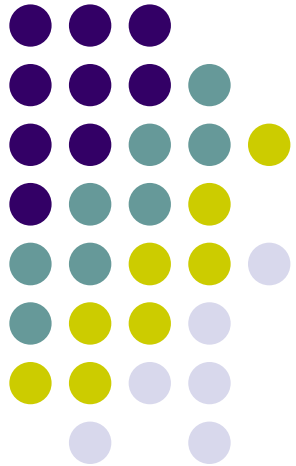
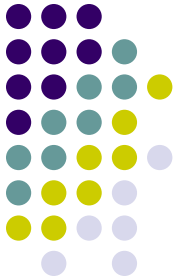
第二章 JavaEE 概述

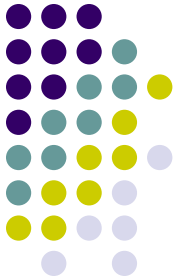




学习要点：

- 1．熟悉分布式的多层应用开发模型。
- 2．熟悉组件的含义与各种组件类型。
- 3．了解容器的含义与各种容器类型。
- 4．熟悉各种JavaEE APIs。
- 5. 了解打包与部署，熟悉JAR/WAR/ EAR文件格式及相互区别。
- 6. 了解JavaEE应用开发角色。

Java EE的产生和发展

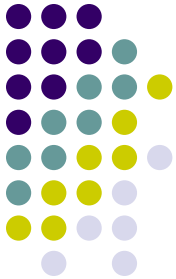


- 1991年4月：Sun Microsystems公司由James Gosling(Java之父)领导的一个叫“Green”的项目。
- 1995年5月：SUN在Sun World'95大会上正式发布Java，这一天被IT界视为Java语言的诞生日。
- 1996年1月：Sun公司发布了Java开发工具包(Java Development Kit)JDK 1.0。
- 1997年2月：Sun公司发布了JDK 1.1。
- 1998年：Sun公司发布了JKD 1.2(称为Java 2 platform)和JSP/Servlet、EJB规范。
- 1999年：Sun把Java 2 platform技术一分为三：J2SE、J2EE和J2ME。其中，J2EE就是本书将要介绍的Java EE 的前身，是第一个JDK企业版，它包含10个规范和相应的API，包括企业应用程序所需的常用的Web层、业务逻辑层、表示层以及消息传递服务。

Java EE的产生和发展

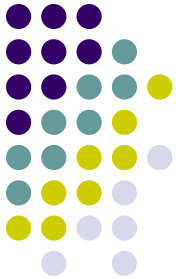
- **2001年 9月**：在JDK1.3发布16个月后，SUN发布了J2EE 1.3。该版本改进了约半数J2EE 1.2规范并引入了XML API、连接器架构和安全框架。
- **2003年11月**：在JDK1.4发布21个月后，SUN发布了J2EE1.4。该版本对J2EE1.3中13种技术中的9种进行了改进，并引入了新的Web服务和安全支持。
- **2004年10月**：Sun公司发布了期待已久的JDK1.5。为了突显这个版本的重大更新，SUN公司将平台称谓中的“2”字去掉，将原来简写名称中的“J”改为全称“Java”。相应的标准版称为Java SE 5.0，相应的企业版Java EE 5.0于2006年5月也正式发布。
- **2006年12月**：Java SE 6.0发布，但相应的企业版直到3年后的2009年12月才正式发布。
- **2013年6月**：最新版本的Java EE 7.0发布。

Java EE 5 概述

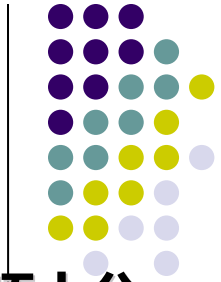


- 作为一个开放性的企业应用开发框架，由Sun公司所主导的JavaEE在之前一直被称为J2EE，但在J2EE1.4版本以后的1.5版本，Sun公司开始重新将这个1.5新版本命名为JavaEE 5.0（本书以下将均简称为JavaEE 5或JavaEE）。
- JavaEE 5标志着Java在企业应用开发技术方面向前发展的又一个里程碑。“Do more with less work”，这是JavaEE 5的正式宣言，也是其与J2EE 1.4最显著的区别。

1.2 Java EE新特性

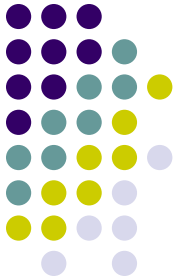


- Java EE 5旨在解决J2EE1.4的过度理论化计算模型和老态笨拙的开发效率和运行期性能。它重点关注目前Java应用开发的诸多热点：运行可靠性、开发效率、扩展灵活性及企业应用整合，顺应轻量级、简单化趋势，给开发者和企业带来真正的实惠。
- 使用新的Java EE平台将会使开发工作变得从未有过的简单和快速。Java EE的目标是为开发人员提供一个功能强大的API库的同时，通过改善系统架构和简化开发和部署的方式，从而缩短开发时间、降低开发难度、提升软件的性能。
- 软件的分布式架构、事务特性、便捷开发等相对于速度、安全和稳定性更加重要。



① 标注

- 即Annotation，是Java EE 5中引入的一项小特性，却是一项十分有效的新特性。标注本质上是一种元数据，通过在Java代码中加入元数据信息(Meta Data)，从而指导工具和库对代码的处理。
- 引入Annotation可以大大降低Java EE开发的成本，特别适应中小型系统的开发，简化这部分系统开发的步骤。
- 在POJO类中添加的Annotation，可实现多种功能的简化。如：
  - 定义和使用Web Service；
  - 开发EJB组件；
  - 映射Java类到XML；
  - 映射Java类到DB；
  - 指定外部依赖；
  - 指定部署信息，包括安全属性等。
- 有了Annotation，XML部署描述符不再是必须的。



② EJB 3

- EJB3是对重量级的EJB2的批判性思考和扬弃。
- EJB3不再需要EJB home接口，也不需要实现SessionBean接口，一个简单的POJO对象就可以代表实体对象，并支持继承和多态。
- 困扰开发者的EJB部署描述符不再是必须。
- EJB3的持久化变得更加简化、轻量级
- EJB3的查找也变得更加简便，JNDI API不再是必须的。
- EJB3还使用了Interceptor，在业务方法被调用前进行拦截，因而更加容易实现灵活的AOP编程。

③ JPA

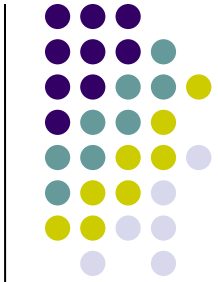
- 即Java 持久化API，它是一个轻量级的对象持久化模型，是Java EE 5的又一大亮点。
- 在JPA中，实体就是POJO对象。实体对象不再是组件，也不再需要位于EJB模块中。
- 可使用标注来指定对象关系映射信息，同时兼容XML描述符。
- JPA支持命名查询，即通过元数据表示的静态查询，使得重用查询变得非常简单。
- JPA支持简单的打包规则。例如，实体Bean可以是EJB JAR、应用程序客户端JAR、WEB-INF/lib、WEB-INF/classes的一部分。
- JPA支持分离的实体，可以对它们进行序列化，从而能够通过网络传输。
- JPA提供了功能强大的EntityManager API。可方便地执行实体的创建、读取、更新和删除(CRUD)操作，使编程变得更加简便。

④ 更简单、更广泛的Web Service支持

- Java EE 5提供了更简单、更广泛的Web Service支持，包括：
  - JAX-WS 2.0
  - JAXB 2.0
  - Web Services Metadata
  - SAAJ 1.3
- JAX-WS 2.0继承自JAX-RPC1.1，实现了更简单的编程模型并集成JAXB 2.0，支持可扩展的传输协议，支持异步客户端，并支持REST应用。
- JAXB 2.0全面支持XML Schema，能够绑定Java类到XML Schema，支持更小的内存占用和更快的编组(marshalling)、更灵活的编出(unmarshalling)，支持局部绑定XML文档到JAXB对象。

⑤ 依赖注入(DI)

- 通过依赖注入使资源访问变得更加容易。可注入的资源包括：
  1. SessionContext
  2. DataSource
  3. Other EJB
  4. Web services
  5. message queue

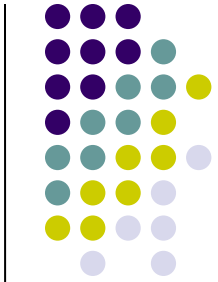


⑥ 泛型(Generics)

- 一个集合可以存放任何类型的对象，相应地，访问集合中的对象时我们也不得不对他们进行强制类型转换。
- Java EE 5通过引入泛型，允许约定集合元素的类型来获得强类型检查，避免等到运行时出现类型转换错误，也可免除显式的强制类型转换(cast)。
- 以下是使用泛型前后的对比示例：

未使用泛型	使用泛型

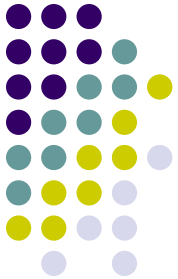
ArrayList list = new ArrayList();
list.add(0, new Integer(42));
int total = (Integer) list.get(0))
.intValue();
	ArrayList<Integer> list = new ArrayList<Integer>();
	list.add(0, new Integer(42));
	int total = list.get(0).intValue();



⑦ 自动装箱和自动拆箱

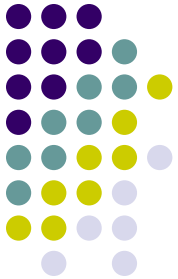
- Java EE 5之前，集合不能存放基本类型。简单数据类型(int，long，float等)和其对应的包装类型(Integer，Long，Float等)之间不能自动转换。
- 自动装箱(Auto-boxing)和自动拆箱(Auto-unboxing)机制解决了简单类型和包装类型之间的自动转换问题。自动装箱特性让Java自动包装一个简单数据类型(例如int)到对应的包装类型(Integer)中。自动拆箱是相反的过程，即将一个包装类型(例如Integer)自动转换为它所对应的简单数据类型(int)。
- 下面是一个自动装箱和拆箱的示例：

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(0, 42);                    // 42 由int自动装箱为Integer
int total = list.get(0);            // 第一个Integer元素被自动拆箱为int
```



⑧ 枚举类型

- Java EE 5中引入一个全新的、类型安全的枚举类型(Enumeration)，之前的Java版本由于不支持枚举类型而常常导致程序中需要写很多public static final常量声明语句。
- 本质上，一个枚举类型是一个命名常量的列表。枚举类型通过新的关键字enum来声明。

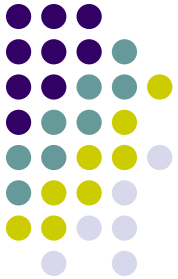


- 下面是定义一个枚举的示例代码：

```
public enum Color {
    Red,
    White,
    Blue
}
```

- 然后可以这样来使用：Color myColor = Color.Red ;
- 枚举类型还提供了两个有用的静态方法values()和valueOf()。下面的for循环将逐个输出枚举的命名常量。

```
for (Color c : Color.values() ) System.out.println(c);
```



⑨ 增强的for循环

- Java EE 5增加了“For-Each”形式的循环。For-Each循环的引入简化了集合的遍历，为大量集合、数组等的循环处理提供了便利。
- For-Each循环有众多好处，如防止数组越界，避免强制类型转换(case)，能在编译时就发现类型等。
- 下面是遍历集合list并对其所有元素进行处理的代码示例：

以往代码：

```
for(Iterator i=list.iterator(); i.hasNext(); ) {
    MyClass myObject = (MyClass)i.next();
    myObject.process();
}
```

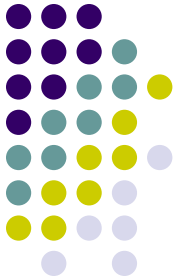
For-Each代码：

```
for ( MyClass myObject :list )
    myObject.process();
}
```

⑩ 可变参数(Varargs)

- Java EE 5允许声明能接受可变数目参数的方法。Varargs的引入使得创建带有可变数目参数的方法变得更加容易。下面是使用不定参数的例子：

```
// 方法定义
void argtest(Object ... args) {
    for (int i=0; i<args.length; i++) {
        // 循环体
    }
}
// 调用方式
argtest("test", "data");
```



11 静态导入(Static Import)

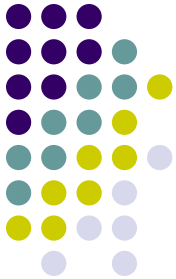
- 过去每次使用静态成员(属性或方法)时都要在它前面冠以相应的类名，如格式ClassName.staticMember。
- Java EE 5引入静态导入，即直接导入类的静态成员，从而允许直接通过静态成员的名字来访问它们，无需再给出他们的类名。例如：

```
// 静态导入
   import static java.lang.System.*;
   import static java.lang.Math.*;
   … …
// 调用静态成员
   out.println();
   random();
```

JavaEE 概述

- JavaEE总览
  - 分布式的多层应用开发模型
  - JavaEE编程思想
  - JavaEE组件
  - JavaEE容器
  - JavaEE APIs
- JavaEE应用打包与部署
  - 什么是打包与部署
  - JAR/WAR/EAR文件格式
- JavaEE应用开发角色
- 更多内容

分布式的多层应用开发模型



- 所谓**企业级应用**，并不是特指为企业开发的应用软件，而是泛指那些为大型企业或组织创建的应用程序。不同于简单的桌面应用程序，企业级应用程序一般具有以下**特点**：
  - **分布式**：通过局域网运行在一个组织内部，或通过Internet连接分布在世界各地的部门或用户。
  - **快速适应性**：企业组织需要不断地改变业务规则来适应社会信息的高速变化，相应的，企业应用程序必须具备及时适应需求的改变的能力，同时又尽可能地减少资金的投入。
  - **高安全性**：企业应用系统必须保证其运行的更高安全必性和可靠性。
  - **可扩展性**：企业应用系统除了要考虑应对企业不断增长的信息资源，还要充分考虑用户群体的膨胀给应用带来的性能上的扩展需求。
  - **集成化**：为了最大限度地利用信息资源，要求企业应用系统必须尽可能集成已有的遗留应用，这是一个很大的挑战。

分布式的多层应用开发模型

- JavaEE 5支持按照N层的分布式系统体系结构与组件模型来开发企业应用。企业应用系统的逻辑功能可被划分为不同的逻辑层，不同的逻辑层可被部署运行于不同的物理环境中。每个逻辑层可采用相应的各种组件技术开发实现各种不同的应用组件。
- 图2-1显示了一个典型的三层JavaEE桌面应用系统结构和一个典型的四层JavaEE Web应用系统结构。

分布式的多层应用开发模型

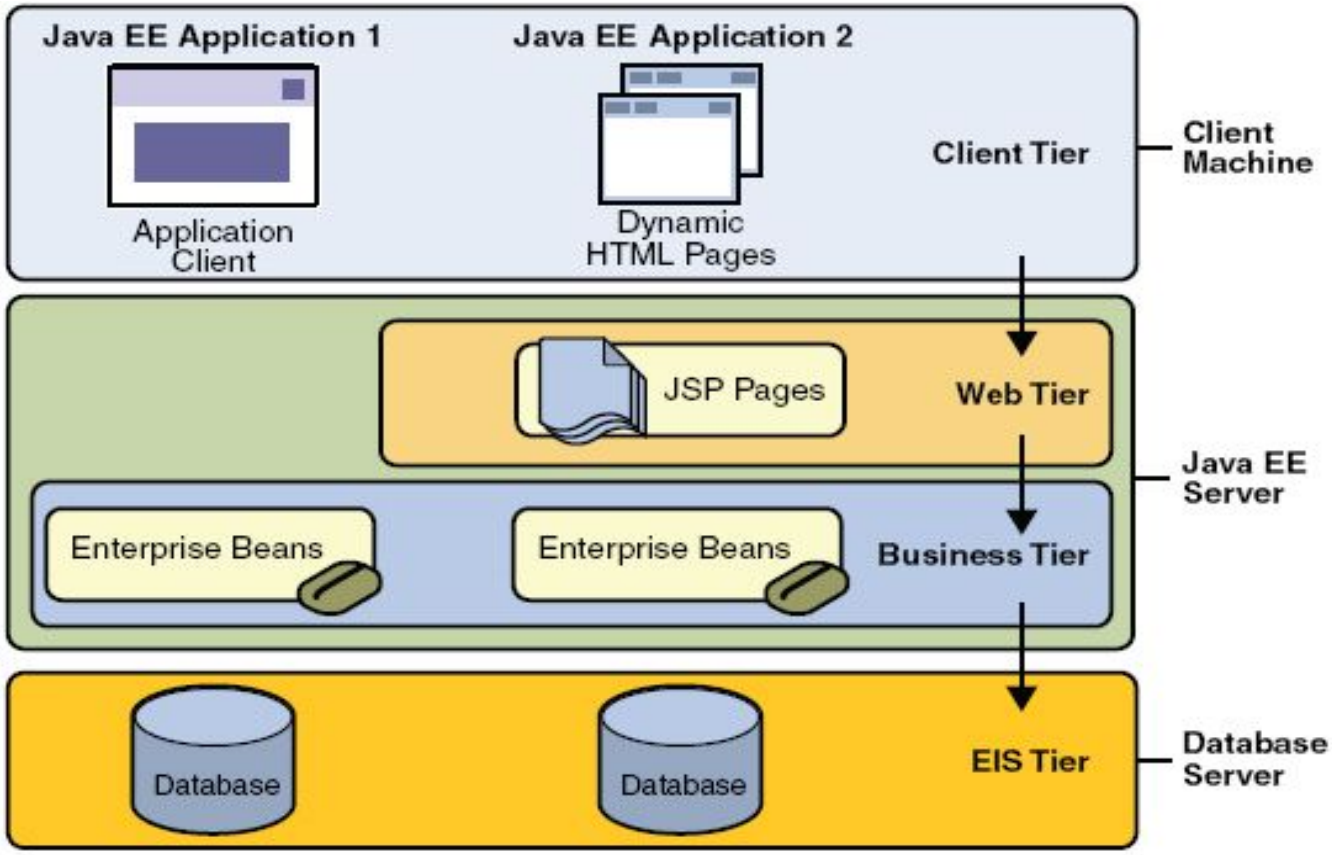
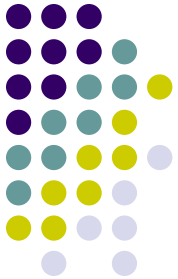


图2-1 典型的JavaEE应用系统层次结构

分布式的多层应用开发模型



- 客户端层（Client-Tier）：包括各种客户端应用界面组件如HTML页面组件、Applet组件与Windows桌面应用组件，它们均是部署运行在客户端的机器上。
- Web层（Web-Tier）：包括各种JSP编写的动态页面组件与Servlet组件，它们运行在JavaEE的应用服务器上。
- 业务层（Business-Tier）：主要由Java 企业Bean（Enterprise Bean）构成，它们是被部署运行于JavaEE的应用服务器上。
- 企业信息系统层（Enterprise Information System-Tier）：一般指各种存储应用数据的关系数据库系统或文件系统，它们一般是部署运行在专门的数据库服务器上。在某些分布式企业应用中，涉及从其它企业遗留应用系统中存取数据，因此企业信息系统层可能还会包含一些企业遗留应用系统。

Java EE编程思想

- Java EE为满足多层体系结构的企业级应用开发的需要，提出了“组件＋容器”的编程思想。
- Java EE应用的基本软件单元是Java EE组件。所有的Java EE组件都运行在特定的运行环境之中。组件的运行环境被称为容器。
- 容器为组件提供必需的底层基础功能，容器提供的底层基础功能被称为服务。组件都是在容器的Java虚拟机中进行初始化，并调用容器提供的标准服务来与外界交互。容器提供的标准服务包括：命名服务、数据库连接、持久化、Java消息服务、事务支持、安全服务等。

Java EE编程思想

- 组件与容器之间的交互是通过“部署描述文件”来定义的。每个发布到服务器上的组件除了要包含自身实现的代码文件外，还需要包含一个部署描述文件(.XML文件)。
- 部署描述文件中详细地描述了组件所要调用的容器服务的说明信息及参数等。部署描述文件就像组件与容器间达成的一个“契约”，容器根据部署描述文件的内容为组件提供服务，组件根据部署描述文件中的内容来调用容器提供的服务。
- 为了简化编程和应用部署，Java EE 5规范支持在组件的实现代码中使用标注来替代复杂的部署描述文件。

JavaEE组件

- 一个JavaEE组件是一个自包含的功能性软件单元，这个软件单元与它的相关类和文件组装进一个JavaEE应用程序，这个组件可以和其它的组件进行会话与通讯。
- JavaEE组件使用Java语言编写并像普通Java程序一样被编译。JavaEE组件区别于一个普通Java类的不同点是：JavaEE组件是遵照JavaEE组件技术规范，具有良好功能封装性和可装配性的软件单元。
- 下面分别简要介绍JavaEE中包含的各种组件类型：

JavaEE组件

- ***1．客户端组件***

  JavaEE客户端组件可分为Web客户端组件和应用程序客户端组件两种类型。

  - **Web客户端组件: 是指运行在浏览器中的客户端组件。在JavaEE中支持两种不同的Web客户端组件：一种是基于HTML、XML及Javascript编写的静态Web页面组件；另一种则是运行在浏览器中的小客户端应用程序即Applet组件。Web客户端有时候也被称作瘦客户端。**

JavaEE组件

- **应用程序客户端组件: 是指运行在客户端桌面环境中的客户端组件。**
  - **窗体应用客户端组件：可为用户提供了一个比使用标记语言的Web客户端更加丰富强大的用户交互界面，其一般是采用Java Swing框架或AWT（Abstract Window Toolkit）开发实现的窗体图形用户界面。**
  - **命令行（console）行式的应用客户端：是一个类似DOS操作界面的一个存文本字符界面的客户端。与前面提到的瘦客户端相对应，往往我们将应用程序客户端组件称作为富客户端。**

JavaEE组件

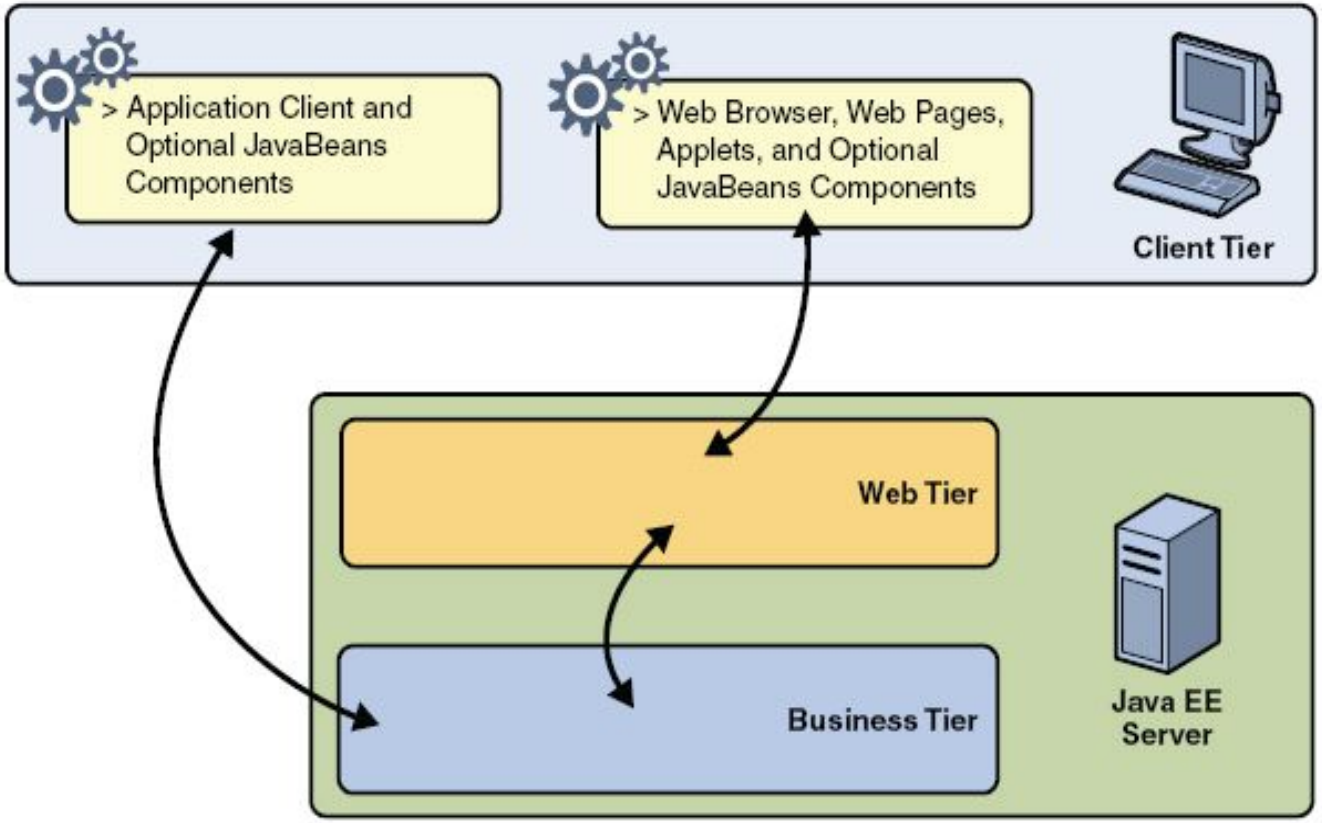
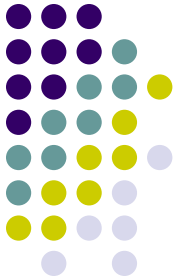


图2-2 JavaEE中的客户端组件

JavaEE组件



- *2. Web 组件*

  一个JavaEE Web组件可以是一个Servlets组件或是一个JSP页面。

  - Servlets是按照Servlets技术规范编写的Java类，这个类可以动态处理一个从Web客户端发出的HTTP请求（Request）和给出回复（Response）。
  - JSP页面是一种Java语言与静态Web页面混编的文本文件，JSP页面运行时被编译为Servlets类的形式执行，但它提供了一种更易于进行HTML动态页面的编排形式。

JavaEE组件

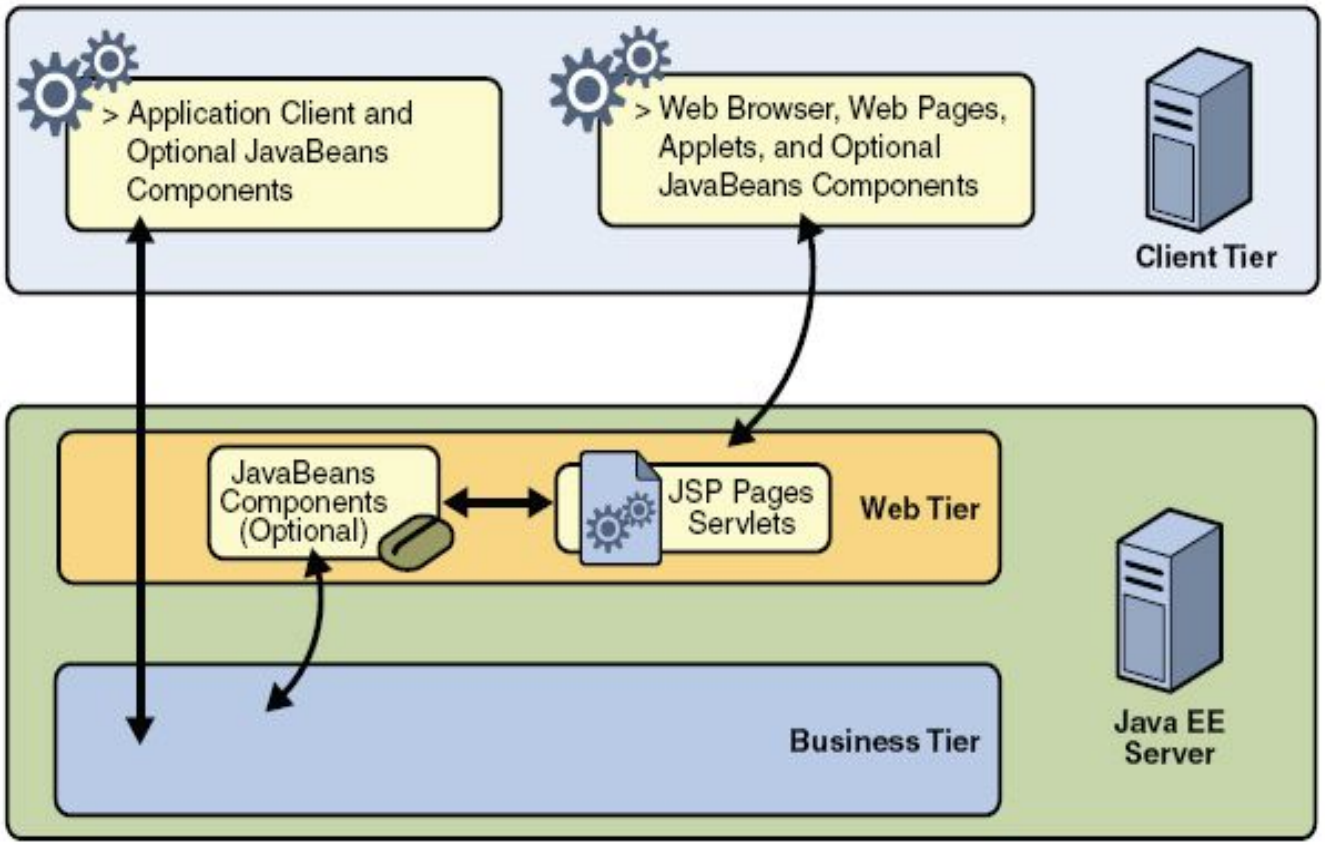


图2-3 JavaEE中的Web组件

JavaEE组件

- ***3. 业务组件***
  - 一个企业分布式应用的业务逻辑被封装在业务层组件中。JavaEE 中的业务组件主要包括会话Bean（Session Beans）组件、消息驱动Bean（Message-Driven Beans）组件和Java持久化实体（Java Persistence Entities）组件。
  - 在JavaEE 5中，原有的EJB组件被Java持久化实体所替代，而会话Bean组件和消息驱动Bean组件被统称作企业Bean组件。

JavaEE组件

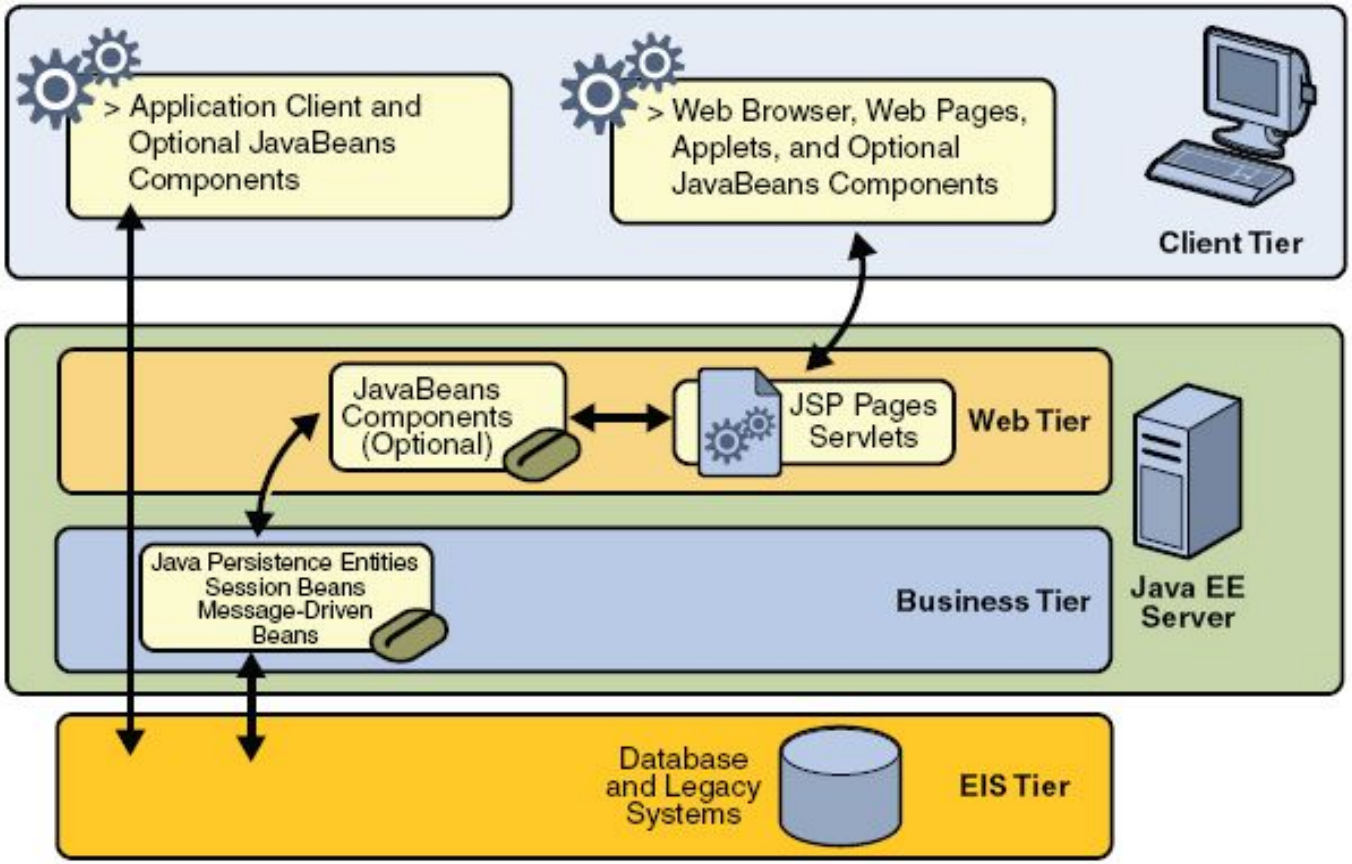
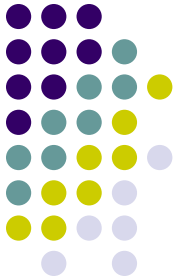


图2-4 JavaEE中的业务组件

JavaEE容器

- 基于组件和平台独立的JavaEE使分层的企业分布式应用程序容易开发，因为各种应用的功能逻辑被封装在可重用的JavaEE组件中。另外JavaEE服务器以容器的形式为所有JavaEE组件提供底层公共服务，因此您不必再为这些底层公共服务伤脑筋，而可以专注于解决应用系统的商业问题。
- 容器（Container）是组件和支持组件功能的底层特定平台（如数据库、分布式的网络环境及Java虚拟机等）之间的接口。在运行Web组件、企业Bean组件或者JavaEE客户端组件之前，您必须将它们装配到一个JavaEE应用程序中，并部署它们到容器中。

JavaEE容器



- 装配的过程包括为JavaEE应用的每个组件和JavaEE应用本身设置容器的配置信息。这些配置信息定制JavaEE服务器支持的底层服务，包括安全，事务管理，Java命名和目录接口（JNDI）查找和远程连接等。
- 容器服务
  - JavaEE安全模型让您配置Web组件或者企业Bean组件，以使系统资源只被授权用户访问。
  - JavaEE事务模型让您指定属于同一个事务的多个方法以使这些方法作为一个原子操作被执行。

JavaEE容器

- JNDI查找服务为企业应用中的多种命名和目录服务提供统一接口使组件可以统一访问这些命名和目录服务。
- JavaEE远程连接模型管理客户端组件和企业Bean组件之间的底层通信。企业Bean被创建后，客户端组件调用企业Bean组件的方法就像调用一个本地方法一样。
- JavaEE体系结构提供可配置服务意味着同一个JavaEE应用中的组件可以根据不同的部署环境而有不同的行为。
- 容器也管理着很多不可配置的服务，如企业Bean和Servlets组件的生命周期、数据库连接池、数据持久化机制和JavaEE平台API的访问权等等。

JavaEE容器

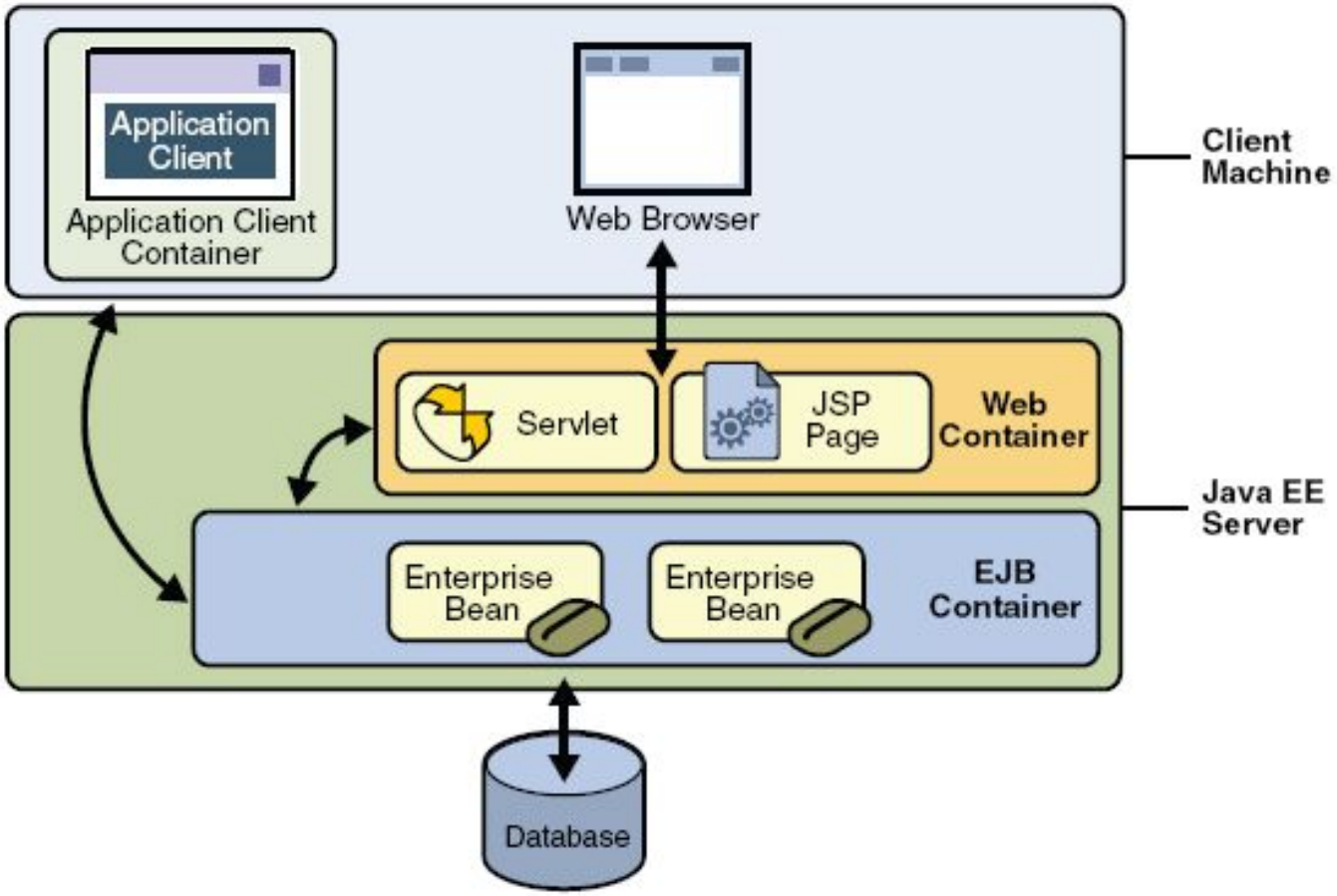


图 2-5 JavaEE的容器类型

JavaEE容器

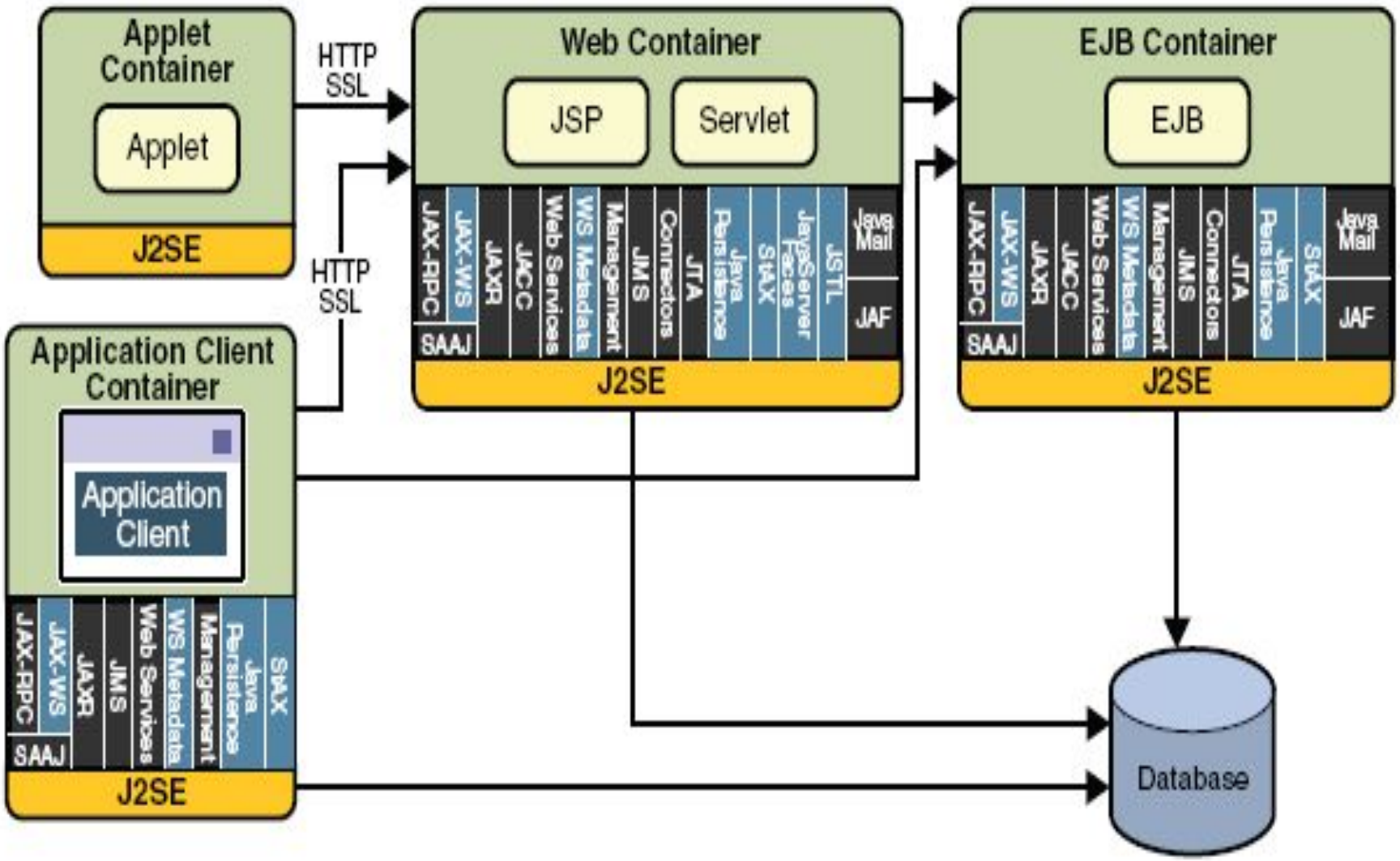
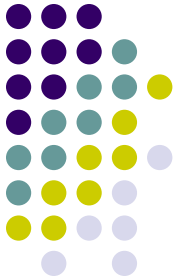


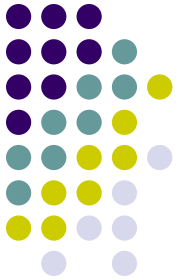
图 2-6 JavaEE容器服务与接口

JavaEE APIs



- **Java Server Pages 技术**
  - JSP技术为创建显示动态生成内容的Web页面提供了一个简捷而快速的方法。JSP技术让您可以直接将servlet小片断放到基于文本的文档中。一个JSP页面是一个基于文本的文档。该文档包含两种类型：静态模版数据，它可以被任何基于文本的格式表达，比如HTML、WML和XML。JSP元素决定了JSP页面怎样构建动态内容。

JavaEE APIs



- JSP技术在多个方面加速了动态Web页面的开发：
- 将Web页面的静态编排和动态编程进行分离
- 强调可重用的组件
- 采用标识简化页面开发

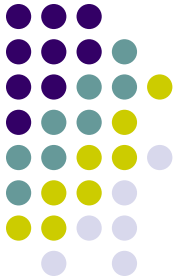
JavaEE APIs

- **Java Servlet 技术**
  - Servlet技术是JavaEE中提供的实现动态网页的又一种技术。Servlet与JSP技术相比，Servlet技术是JSP技术的基础，因一个JSP页面在服务器端被JSP引擎解释执行成一个Servlet类。只不过由于Servlet是一个实现了特殊接口的 Java类，它不像JSP技术那样具有静态Web页面编排与动态页面内容编程分开的优点。

JavaEE APIs

- Servlet常被用于服务器端实现针对HTTP流的处理。一个Servlet程序负责处理它所对应的一个或一组URL地址的访问请求，接收访问请求信息和产生响应内容。Servlet与普通java类相比，不同点只是一个Servlet类的输入信息的来源和输出结果的目标与普通的Java类不一样，所以，普通Java类所能完成的大多数任务，Servlet都可以完成。

JavaEE APIs

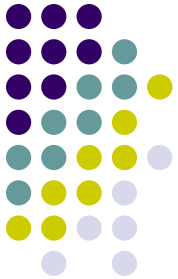


- Servlet具有如下的一些基本功能：
  - 创建并返回一个包含基于客户请求性质的动态内容的完整的 HTML页面。
  - 创建可嵌入到现有 HTML 页面中的一部分 HTML 页面（HTML 片段）。
  - 与其它服务器资源（包括数据库和基于 Java 的应用程序）进行通信。
  - 用多个客户机处理连接，接收多个客户机的输入，并将结果广播到多个客户机上。例如，Servlet 可以是多参与者的游戏服务器。

JavaEE APIs

- 当允许在单连接方式下传送数据的情况下，在浏览器上打开服务器至applet的新连接，并将该连接保持在打开状态。当允许客户机和服务器简单、高效地执行会话的情况下，applet也可以启动客户浏览器和服务器之间的连接。可以通过定制协议或标准（如 IIOP）进行通信。
- 对特殊的处理采用 MIME 类型过滤数据，例如图像转换和服务器端包括（SSI）。
- 将定制的处理提供给所有服务器的标准例行程序。例如，Servlet 可以修改如何认证用户。

JavaEE APIs



- **EJB技术**
  - 一个企业JavaBeans(EJB)组件或企业bean是一个商业逻辑模块代码的主体。在JavaEE服务器上，企业bean作为一个创建块可以单独被使用或者和其它的企业bean一起使用来执行业务逻辑。
  - JavaEE中有两种企业beans：会话beans（Session beans）和消息驱动beans（Message-Driven beans）。

JavaEE APIs

- 会话beans表示的是与客户端的会话，当客户端运行结束，会话beans及其中的数据生命周期结束。消息驱动beans综合了会话beans和消息监听器的特征，允许企业组件异步地接受JMS消息。
- 在JavaEE5中，实体beans已经被JavaEE持久性API实体替代。一个实体就代表数据表中的一条持久数据。如果客户端访问结束，或者服务器停止，数据持久管理负责将数据存储。

JavaEE APIs

- **Java 消息服务**
  - Java消息服务（JavaMessage Service，JMS）是一个消息标准，它允许JavaEE应用程序组件产生、发送、接收和读取消息。它能够进行分布式的宽松连接的、可靠的和异步的交流。

JavaEE APIs

- **Java事务API**
  - JTA为划分的事务提供一个标准接口。JavaEE体系结构提供默认的自动提交功能来处理事务提交和回滚。自动提交意味着每次读写操作后，其它应用程序看到的都是更新后数据。然而，如果应用程序执行两个互相依赖的分开的数据库访问操作，您会希望使用JTA API来划分到哪里才是一个完整的事务，包含两种操作，开始操作、回滚和提交操作。

JavaEE APIs

- **JavaMail API**
  - JavaEE应用程序能够使用 JavaMail API来发送e-mail通知。JavaMail API拥有两部分：一个被应用程序组件用来发新的应用程序级的接口和一个服务器提供商接口。JavaEE平台包括了允许应用程序组件发送网络邮件的带有一个服务器提供商的JavaMail。

JavaEE APIs

- **JavaBeans激活框架**
  - JavaBeans 激活框架（JavaBeans Activation Framework，JAF）被包括进来，因为JavaMail要使用它。它提供标准的服务来决定一个任意数据段的类型，包装访问，揭示在它上面允许的操作和产生一个适当的JavaBeans组件来执行这些操作。

JavaEE APIs

- 针对XML处理的Java API
  - JAXP支持XML文档的处理，使用DOM，SAX和XSLT。JAXP是应用程序可以解析和转换XML文档，它独立于特殊的XML处理实现。
  - JAXP也提供命名空间的支持。为了灵活的设计，JAXP让您运行于XML兼容的XSL处理分析器，从您的应用程序范围到W3C计划支持的范围。

JavaEE APIs

- **针对Java而附加API的SOAP**
  - 针对Java而附加API的SOAP（The SOAP with Attachments API for Java，SAAJ）是一个底层API，JAX-RPC依赖这个底层API。它授予符合SOAP1.1规范的消息的产生和消灭和附加注解的SOAP。大多数开发者并不使用SAAJ API，而是使用上层的JAX-RPC API。

JavaEE APIs

- **XML注册的Java API**
  - JAXR允许您通过Web访问业务和一般用途的注册信息。JAXR支持ebXML Registry/Repository 标准和UDDI规范。
  - 商业提交资源共享和在其它已经提交的资源上查找。

JavaEE APIs

- 基于XML的RPC的Java API
  - JAX-RPC使用SOAP标准和HTTP协议，客户端程序能够跨越Internet进行基于XML的remote procedure calls (RPCs)。JAX-RPC也支持WSDL，所以您能够导入和导出WSDL文档。
  - 使用JAX-RPC和 WSDL，您就能简单地在运行在基于JAVA和非基于JAVA的平台上（如.NET）的客户端和服务器端进行内部操作。

JavaEE APIs

- JAX-RPC依赖HTTP传输协议。允许您产生一个服务性的应用程序，结合安全套接字层（SSL）的Java技术的HTTP协议和传输层安全（TLS）协议来建立基本的或共有的认证。SSL和TLS通过对消息进行加密确保消息的完整性。
- 认证是证实一个团体是否有授权访问某些信息的标准的途径，也就是抵制欺骗性的系统使用和传输。设置一个JAX-RPC Web服务来保护数据传输很重要。

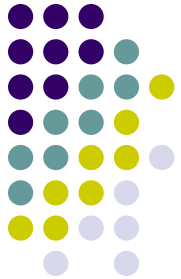
JavaEE APIs

- **JavaEE 连接器体系结构**
  - 提供一个基于Web Service的、面向性能的、安全的、稳定的、基于消息的和支持事务的集成机制，这种集成实现既可以是同步的也可以是异步的。通过JavaEE连接器体系结构集成到JavaEE平台的企业应用系统被展现为Web Service的形式。进行企业应用集成和端到端的业务集成时，JAX-WS和JavaEE连接器体系结构是互补应用的技术。

JavaEE APIs

- **Java认证和授权服务（JAAS）**
  - **为JavaEE应用程序提供一种途径来认证和授权一个企业应用用户或用户组。**
  - **JAAS是一个基于Java编程语言的标准可插拔的授权模块（the Standard Pluggable Authentication Module--PAM）框架，这个框架扩展了Java 2 平台的安全体系结构以支持基于用户的安全授权机制。**

JavaEE APIs



- **JDBC API**
  - 允许从Java的方法里调用SQL命令。当您超越默认的容器管理持久性使用session bean来访问数据库，您在企业bean里只能使用JDBC API。您也能在JSP页面或servlet里使用JDBC API来直接访问数据库而不通过企业bean。
  - JDBC API 包含两部分：被应用程序组件使用的应用程序级别的访问数据库接口和用来附加JDBC驱动到JavaEE平台服务提供商接口。

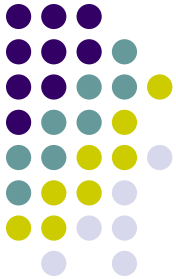
JavaEE APIs

- Java命名和目录接口(JNDI)
  - 提供命名和目录功能。使用JNDI，一个JavaEE应用程序能够储存和检索任何命名类型的Java对象。
  - JavaEE命名服务提供应用程序客户端，企业beans和Web组件访问JNDI命名环境的服务。

JavaEE APIs

- Java命名和目录接口(JNDI)
  - 一个JavaEE组件使用JNDI接口定位它的环境的命名上下文。一个组件产生一个javax.naming. InitialContext对象并且在名称 java：comp/env下查找环境命名上下文。一个组件的命名环境被直接存储在环境的命名上下文中或者存储在直接或间接的子上下文中。
  - 一个JavaEE组件能过访问系统提供的命名和用户提供的对象。
  - 因为JNDI是独立于任何指定的实现，应用程序能使用JNDI访问多层命名和目录服务，包括已经存在的的命名和目录服务，如：LDAP、NDS、DNS和NIS。这就允许JavaEE应用程序和旧的应用程序和系统共存

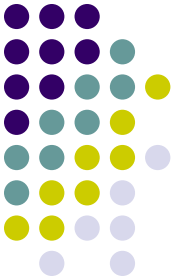
Java EE应用的打包与部署



基本概念

- Java EE应用打包与部署是指将一个Java EE应用中的相关组件封装到一个单元(通常称为包)中，并将它部署到符合Java EE规范的应用服务器上。
- 包(package)可以是单独部署的模块，如Web模块(.war)或EJB模块(.jar)，也可以是完整的Java EE应用(.ear)。一个Java EE应用被打包成为某种标准的格式以后就可以部署到任何符合Java EE规范的Java EE应用服务器上，并开始运行。
- 经过打包的Java EE模块是Java EE组件(如企业beans、JSP页面、servlet等)的集合，并带有这种容器的部署描述文件(.xml)。运行时，Java EE应用服务器读取配置描述文件并将配置相应的组件。

Java EE应用的打包与部署

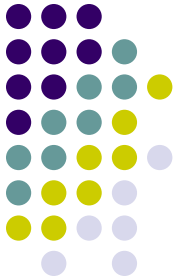


- 在Java EE中，主要存在着以下4种基本类型的模块。
  - EJB模块：包含企业Bean的类文件和一个EJB配置描述文件。EJB模块包装成扩展名为.jar的JAR文件。
  - Web模块：包括JSP文件、Servlet类文件、HTML文件、图形文件和一个Web配置描述文件。Web模块被打包成扩展名为.war的Web应用档案。
  - 资源适配器模块：包含所有的Java接口、类、本地库、其他文档和资源适配器配置描述文件。资源适配器模块被包装成扩展名为.jar的JAR文件。
  - 应用程序客户端模块：包含类文件和一个应用程序客户端配置描述符。应用程序客户端模块被包装成扩展名为.jar的JAR文件。

Java EE应用的打包与部署

- 一个需要部署的Java EE应用有时可能包含以上4种中的多种模块类型，这时就需要将其打包成一个EAR文件(企业应用档案)。一个EAR文件是一个以.ear为扩展名的标准Java档案(JAR)。

Java EE应用的打包与部署

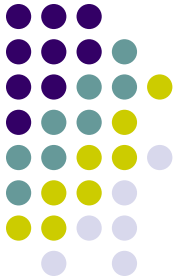


- JAR/WAR/EAR文件的格式
  - Java归档文件(Java Archive File，JAR)是Java规范的一种与平台无关的文件格式，它允许将许多文件组合成一个压缩文件。
  - JAR文件格式以流行的ZIP文件格式为基础。与ZIP文件不同的是，JAR文件不仅用于压缩和发布，而且还用于部署和封装库、组件和插件程序，可提供给编译器和JVM直接使用。在JAR中包含特殊的文件，如清单(manifests)和部署描述符文件，用于指示工具如何处理特定的JAR。
  - JAR文件可以有多种用途。如可以用于发布和使用类库，可以作为应用程序和扩展的构建单元，可以作为组件、Applet或插件程序的部署单位，还可用于打包与组件相关联的辅助资源。

Java EE应用的打包与部署

- JAR文件的特点如下：
  - **安全性**：可以对JAR文件添加数字签名，以便给软件授权和防止软件被篡改。
  - **减少下载时间**：如将一个Applet和相关资源打包到一个JAR文件中，浏览器就可以在一个HTTP事务中下载这个JAR文件，而不必对每一个文件打开一个新连接。
  - **压缩**：JAR格式允许压缩文件以提高存储效率。
  - **平台扩展**：可以使用JAR文件对Java核心平台进行功能扩展(如Java 3D和JavaMail就是由Sun开发的扩展例子)。
  - **包密封**：存储在JAR文件中的包可以选择进行密封，以增强版本一致性和安全性。密封一个包意味着包中的所有类都必须在同一JAR文件中找到。
  - **包版本控制**：一个JAR文件可以包含相关的厂商和版本信息。
  - **可移植性**：处理JAR文件的机制是Java平台核心API的标准部分。

Java EE应用的打包与部署



- 与JAR文件略有不同，WAR文件除了可以组合JSP文件和Servlet类文件等Web组件程序文件外，还可以包含静态HTML文件、图形文件等多种其他类型的文件。
- EAR文件主要用于将多个Java EE应用模块所构成的一个完整应用程序打包，它可以包含多个JAR文件和WAR文件。从打包粒度上讲，按由小至大的顺序排列的话，则分别是JAR、WAR和EAR。

JavaEE应用打包与部署

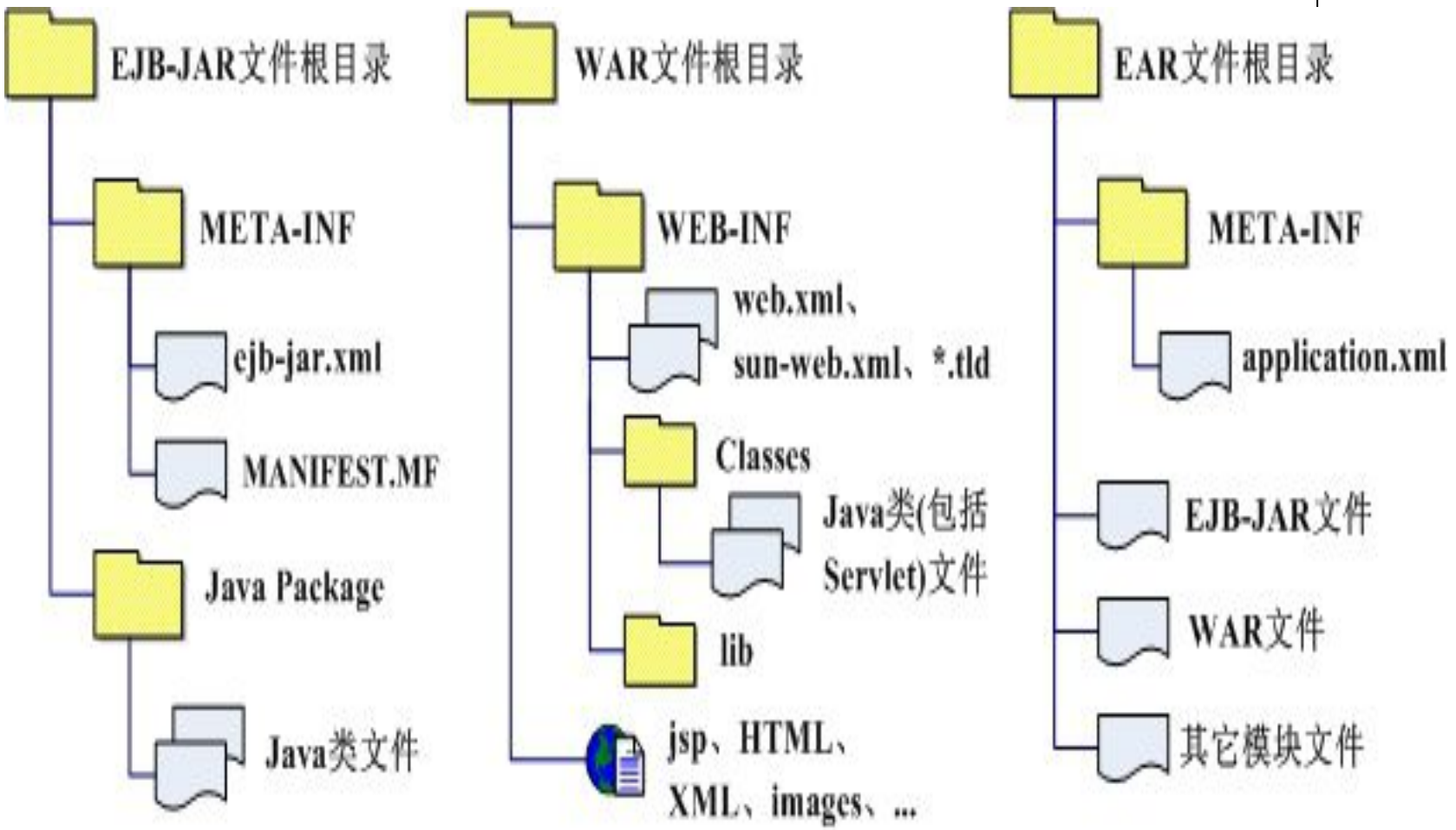
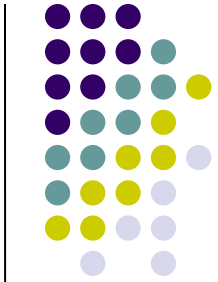


图2-7 JAR、WAR和EAR文件的一般目录结构

JavaEE应用开发角色



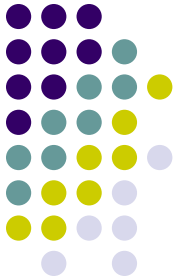
- JavaEE中不同类型的可重用的应用模块使将应用程序的开发和部署可分配给不同的角色即不同的开发与部署职责的人员或团队来完成，因此不同的个人或者团队可以更好地分工合作。
- JavaEE产品提供商设计并实现JavaEE规范定义的JavaEE平台、API和其它特性。典型的JavaEE产品提供商如操作系统、数据库系统、应用服务器、Web服务器厂商，它们根据Java2平台企业版规范实现JavaEE平台。

JavaEE应用开发角色

- 工具提供商是那些提供开发、装配和打包工具的组织或个人。组件开发者、装配者和部署者使用这些工具来工作。

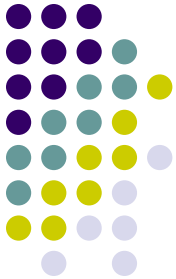
- 应用程序组件开发者是开发JavaEE应用程序可使用的企业Bean、Web组件、Applet和应用程序客户端组件的组织或个人。

JavaEE应用开发角色



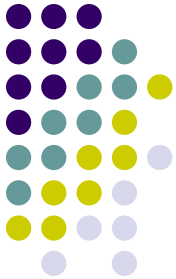
- 企业Bean开发者提供企业Bean的EJB JAR文件，他的工作步骤如下：
  - 编写并编译源文件
  - 配置部署描述符文件
  - 将编译后的类文件和部署描述符文件打
  - 包为一个EJB JAR文件

JavaEE应用开发角色



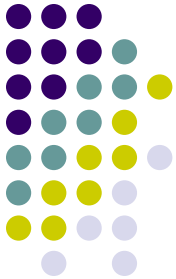
- Web组件开发者的工作任务是提供WAR文件：
  - 编写并编译servlet源文件
  - 编写JSP和HTML文件
  - 配置部署描述符文件
  - 将.class、.jsp、.html和部署描述符文件打包为一个WAR文件

JavaEE应用开发角色



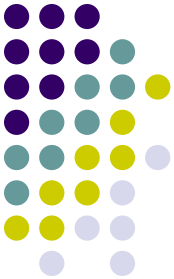
- JavaEE应用程序客户端开发者也提供一个JAR文件：
  - 编写并编译源文件
  - 配置部署描述符文件
  - 将.class类文件和部署描述符文件打包进一个JAR文件

JavaEE应用开发角色



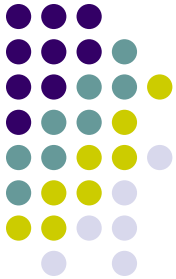
- 应用程序组装者将从组件开发者获得的组件文件装配成一个JavaEE应用程序EAR文件。组装者可以编辑部署描述符文件。组装者的任务：
  - 组装EJB JAR和WAR文件到一个JavaEE应用程序EAR文件
  - 配置JavaEE应用程序的部署描述符文件
  - 确认EAR文件的内容符合JavaEE规范

JavaEE应用开发角色



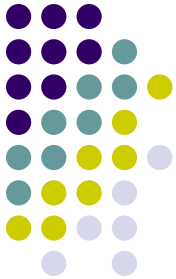
- 应用程序部署者和系统管理员配置和部署JavaEE应用程序，在程序运行时管理计算机和网络结构，并且监控运行时环境。包括设置事务控制、安全属性和指定数据库连接。任务如下：
  - 将JavaEE应用程序EAR文件添加到JavaEE服务器
  - 修改JavaEE应用程序的部署描述符为特定运行环境配置应用程序
  - 部署JavaEE应用程序到JavaEE服务器

更多内容



- 有关JavaEE 5.0框架更多内容可参考链接
  - JavaEE 5.0技术规范：
    http://java.sun.com/JavaEE/technologies/index.jsp
  - JavaEE 蓝皮书（Blueprints）
    http://java.sun.com/reference/blueprints/index.html
  - Java Servlet 技术规范：
    http://java.sun.com/products/servlet/download.html#specs
  - JavaServer Pages 2.0技术规范：
    http://java.sun.com/products/jsp/download.html#specs

更多内容



- JavaServer Faces 1.1技术规范：

http://java.sun.com/javaee/javaserverfaces

- Java API for XML Web Services 2.0技术规范：

https://jax-ws.dev.java.net/spec-download.html

- Enterprise JavaBeans 3.0技术规范：

http://java.sun.com/products/ejb/docs.html