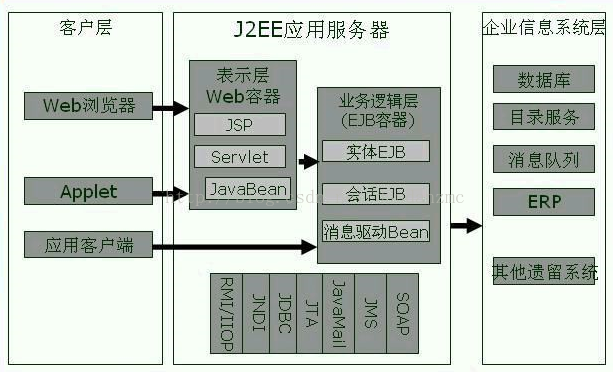
**J2EE架构和MVC设计模式**

**J2EE架构：**

J2EE是Java2平台企业版（Java 2 Platform,Enterprise Edition），它的核心是一组技术规范与指南，提供基于组件的方式来设计、开发、组装和部署企业应用。J2EE使用多层分布式的应用模型。

J2EE分层：



   客户层，运行在客户计算机上的组件，用户与系统的接口逻辑，通过http协议的来访问应用服务器。  
        表示层，运行在J2EE服务器上的组件，通过与业务逻辑层互动，将用户需要的数据以适当的方式输出。  
        业务逻辑层，同样是运行在J2EE服务器上的组件。  
        企业信息系统层（EIS），是指运行在EIS服务器上的软件系统。

  以上层次一般也指三层应用，也就是客户层+J2EE应用服务层+企业信息系统层，分布在三个不同位置：客户计算机、J2EE服务器及后台的数据库或过去遗留下来的系统。

**MVC设计：**

MVC英文即Model-View-Controller，即把一个应用的输入、处理、输出流程按照Model、View、Controller的方式进行分离，这样一个应用被分成三个层——模型层、视图层、控制层。

视图(View)代表用户交互界面，对于Web应用来说，可以概括为HTML界面，但有可能为XHTML、XML和Applet。随着应用的复杂性和规模性，界面的处理也变得具有挑战性。一个应用可能有很多不同的视图，MVC设计模式对于视图的处理仅限于视图上数据的采集和处理，以及用户的请求，而不包括在视图上的业务流程的处理。业务流程的处理交予模型(Model)处理。比如一个订单的视图只接受来自模型的数据并显示给用户，以及将用户界面的输入数据和请求传递给控制和模型。

　　模型(Model)：就是业务流程/状态的处理以及业务规则的制定。业务流程的处理过程对其它层来说是黑箱操作，模型接受视图请求的数据，并返回最终的处理结果。业务模型的设计可以说是MVC最主要的核心。目前流行的EJB模型就是一个典型的应用例子，它从应用技术实现的角度对模型做了进一步的划分，以便充分利用现有的组件，但它不能作为应用设计模型的框架。它仅仅告诉你按这种模型设计就可以利用某些技术组件，从而减少了技术上的困难。对一个开发者来说，就可以专注于业务模型的设计。MVC设计模式告诉我们，把应用的模型按一定的规则抽取出来，抽取的层次很重要，这也是判断开发人员是否优秀的设计依据。抽象与具体不能隔得太远，也不能太近。MVC并没有提供模型的设计方法，而只告诉你应该组织管理这些模型，以便于模型的重构和提高重用性。我们可以用对象编程来做比喻，MVC定义了一个顶级类，告诉它的子类你只能做这些，但没法限制你能做这些。这点对编程的开发人员非常重要。

　　业务模型还有一个很重要的模型那就是数据模型。数据模型主要指实体对象的数据 保存（持续化）。比如将一张订单保存到数据库，从数据库获取订单。我们可以将这个模型单独列出，所有有关数据库的操作只限制在该模型中。

　　控制(Controller)可以理解为从用户接收请求, 将模型与视图匹配在一起，共同完成用户的请求。划分控制层的作用也很明显，它清楚地告诉你，它就是一个分发器，选择什么样的模型，选择什么样的视图，可以完成什么样的用户请求。控制层并不做任何的数据处理。例如，用户点击一个连接，控制层接受请求后, 并不处理业务信息，它只把用户的信息传递给模型，告诉模型做什么，选择符合要求的视图返回给用户。因此，一个模型可能对应多个视图，一个视图可能对应多个模型。

**MVC与J2EE架构的对应关系：**

MVC与J2EE架构的对应关系是:View处于Web Tier或者说是Client Tier，通常是JSP/Servlet，即页面显示部分。Controller也处于Web Tier，通常用Servlet来实现，即页面显示的逻辑部分实现。Model处于Middle Tier，通常用服务端的JavaBean或者EJB实现，即业务逻辑部分的实现。