B/S体系软件设计——J2EE

胡晓军

J2EE的定义

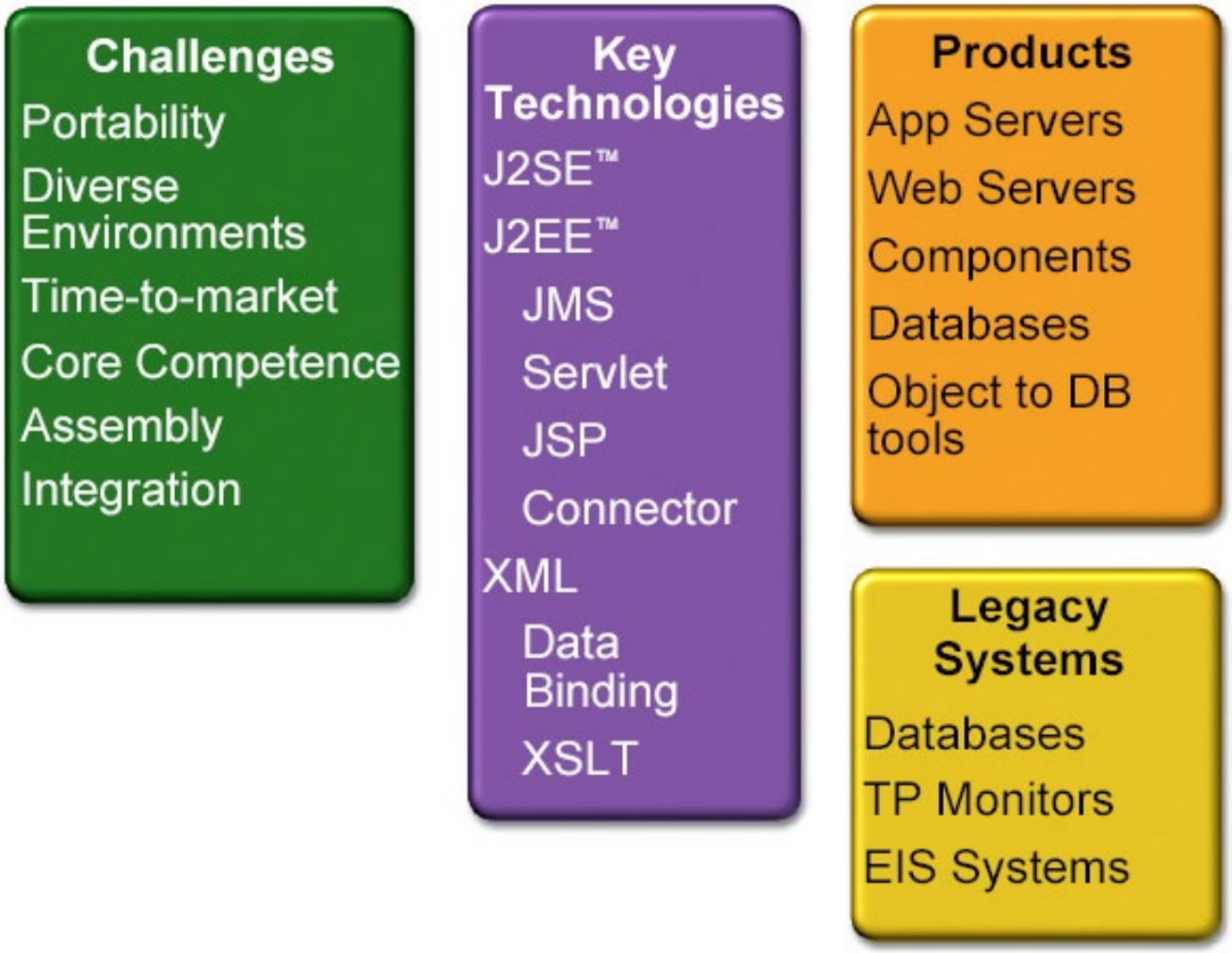
- Java 2 Platform, Enterprise Edition (J2EE)
  - Open and standard based platform for developing, deploying and managing n-tier, Web-enabled, server-centric, and component-based enterprise applications
  - J2EE 1.2 - 1.4
- Java Platform, Enterprise Edition (Java EE)
  - Java Platform, Enterprise Edition (Java EE) builds on the solid foundation of Java Platform, Standard Edition (Java SE) and is the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation web applications
  - Java EE 5 - 8
- Jakarta EE
  - Jakarta EE is a set of specifications that enables the world wide community of java developers to work on cloud native java enterprise applications. The specifications are developed by well known industry leaders that instills confidence in technology developers and consumers.
  - Eclipse Foundation
  - Jakarta EE 8-9

企业级应用的特性

- 企业级应用解决业务问题
- 包括
  - 安全存储
  - 可恢复性
  - 处理各种类型的业务数据
- 可能会有多种接口
  - 基于**Web**的客户接口
  - 给内部用户的**GUI**应用
  - 给外部应用提供数据
  - 与其他业务系统实现交互接口
- 需要支持大量用户（从几百到几千，甚至更多）
- 可能需要从多个存储中整合数据
- 需要实现一整套的业务规则
- 可能会与很多其他企业级应用通信
- 随着业务的增长或变更，企业级应用也要随之灵活变更
- 如果应用出现问题，企业会因此产生各种类型的损失

以上因素的存在导致企业级应用一般都比较复杂，需要有一个良好的架构在支撑

基于J2EE的企业级应用环境



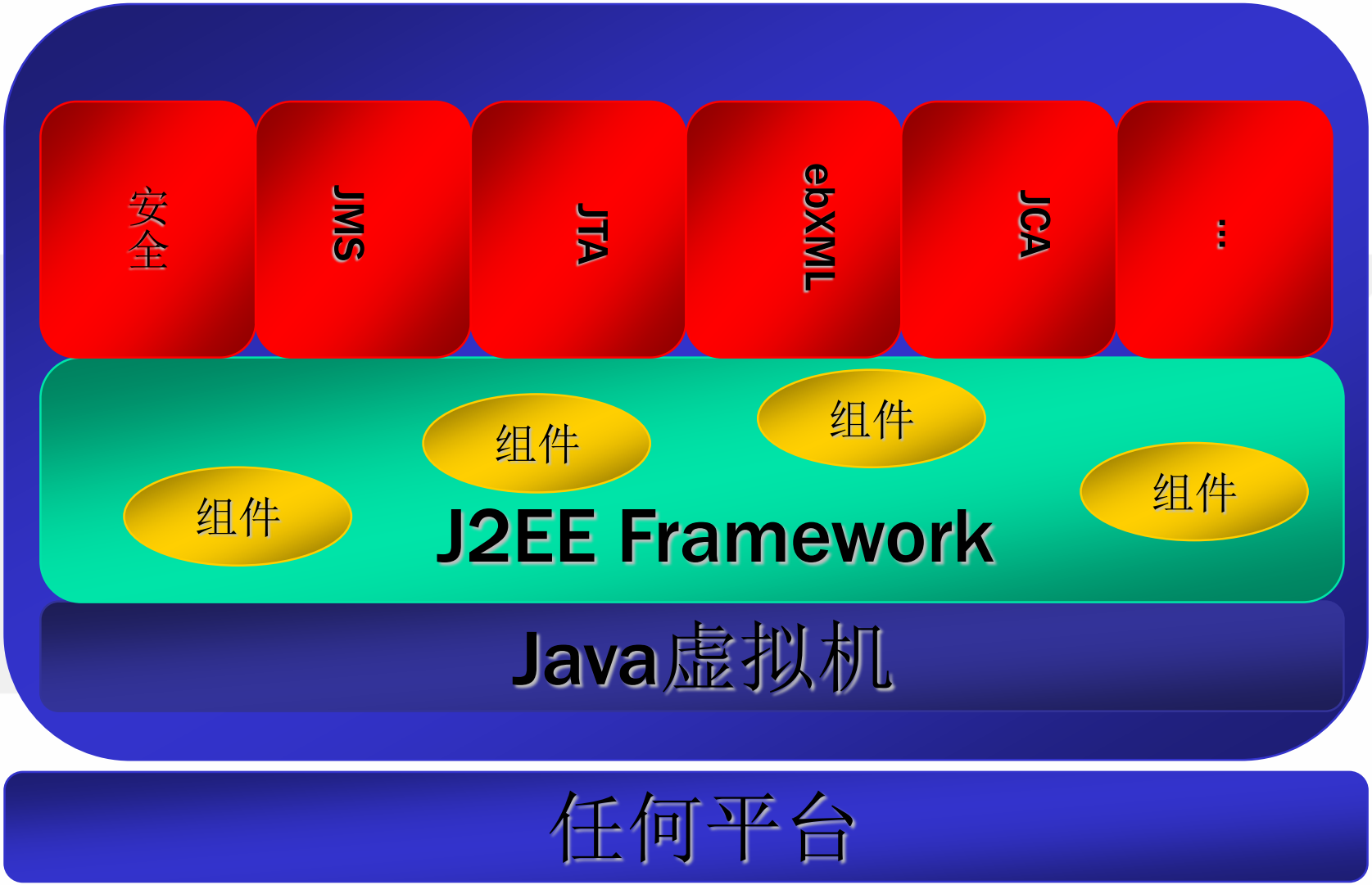
J2EE的优点

- **J2EE可以提供**
  - 分布式、可移植构件的框架
  - 简化服务器端中间层构件的设计
  - 为构件和应用服务器提供标准**API**
- **选择J2EE的好处**
  - 更短的开发时间
    - 可重用组件
    - JSP
    - EJB
  - 自由的选择
    - 基于开放的标准
  - 简化的连接
    - XML，JDBC，RMI-IIOP，Web Service

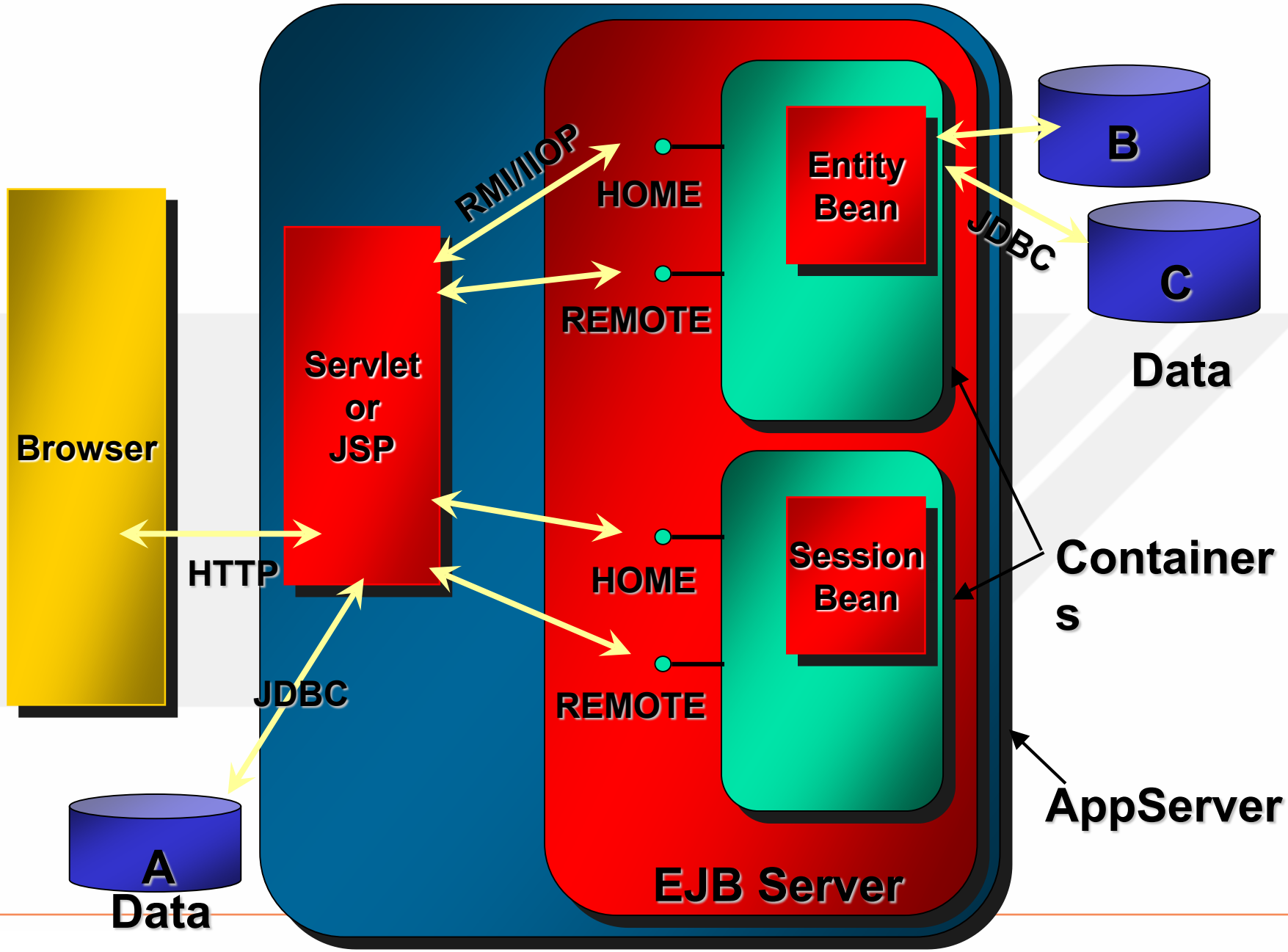
什么是J2EE

- 运用Java技术开发企业应用的标准
- 包括了
  - 多层应用开发模型
  - 开发平台 - APIs 和服务
  - 测试软件包
  - 参考实现
- 将所有Sun的企业技术集合在一个体系结构下的平台
  - 特定版本下的EJB, Servlet, JSP
  - Java Web Server
  - JNDI, JDBC, JTA, JMS, JavaMail,…

J2EE技术架构



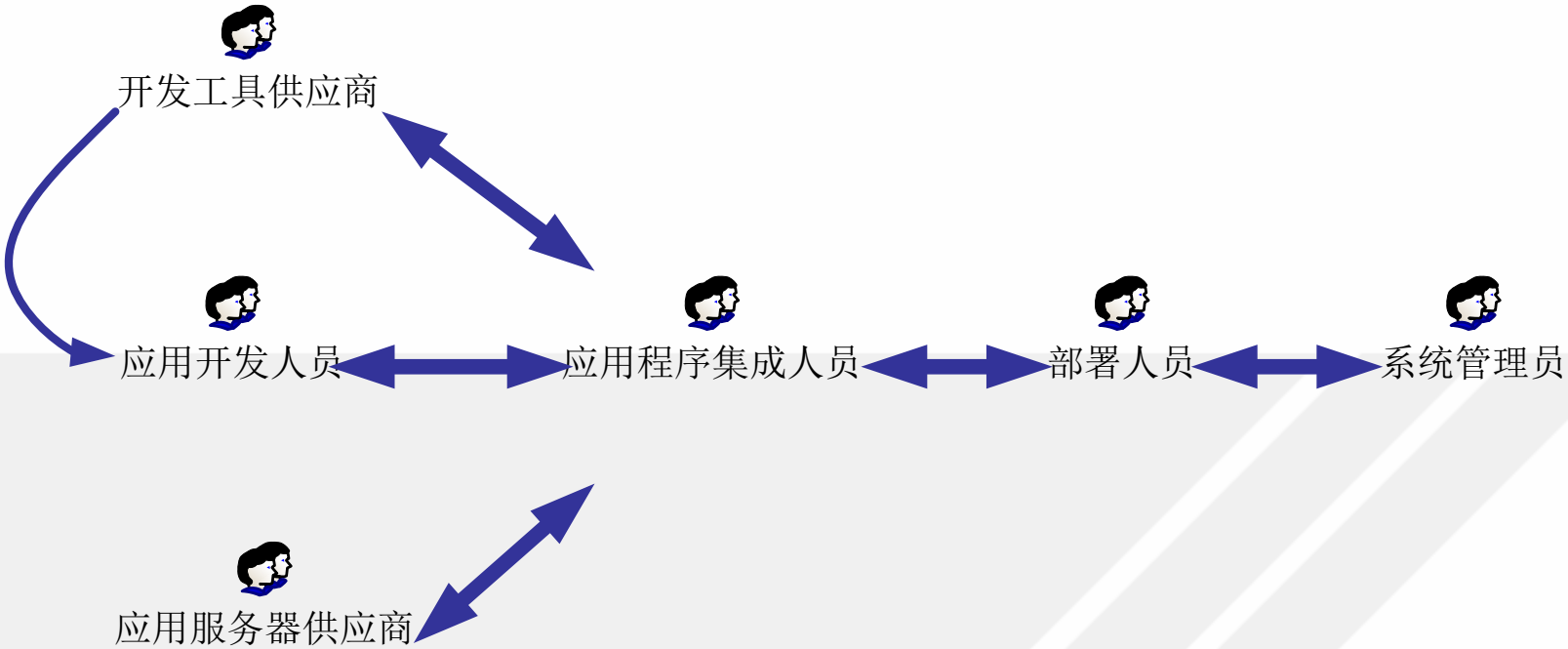
J2EE应用体系架构


RMI/IIOP
HOME
REMOTE
Entity Bean
B
C
JDBC
Data
Servlet or JSP
Browser
HTTP
HOME
REMOTE
Session Bean
Containers
JDBC
A
Data
EJB Server
AppServer

J2EE主要技术概览

- 展示层 - ServLet/JSP
- 中间层 - EJB
- 中间层可用的企业服务
  - 事务服务 JTA
  - 目录服务 JNDI
  - 消息服务 JMS
  - 异步组件 Message-Driven EJB
- 数据层 - JDBC
- 远程调用 - RMI/IIOP
- 使用现有资源 - JCA

J2EE角色组成



开发工具商：提供组件开发工具、应用程序集成工具、应用程序部署工具、界面编辑工具。
应用服务器提供商：负责设计和开发J2EE应用服务器(依据J2EE规范)
应用开发人员:负责开发组件、写部署描述文件、设计界面(显示数据)
应用集成人员:负责将各种界面、组件、描述文件、驱动程序、其他属性文件集成
部署人员：负责将J2EE集成文件部署的应用服务器
系统管理员：负责调整服务器的运行参数、监视服务器的运行情况

J2EE 应用开发

- 编写和编译组件代码
  - **Servlet, JSP, EJB**
- 编写组件的部署描述符
- 把组件装配成可部署的软件包
- 将软件包部署到**J2EE**应用服务器上

开发工具

- Eclipse
  - Open Source
  - 插件丰富
  - myEclipse
- IntelliJ Idea

应用服务器产品

Oracle WebLogic

IBM WebSphere

Sun Java System Application Server

JBoss（EJB、JMS等）

Tomcat（只实现了Web容器）

Resin（一个高性能的J2EE应用服务器）

应用服务器功能

- 负载均衡
- 故障容错
- **Web** 服务
- 网络透明
- 遗留集成
- 事务管理
- 安全性
- 消息
- 多线程
- 持久性
- 数据库连接
- 资源合并
- 开发、测试和封装功能

Servlet开发

- 如果某个类要成为**Servlet**，则它应该从**HttpServlet** 继承，根据数据是通过**GET**还是**POST**发送，重载**doGet**、**doPost**方法之一或全部。 **doGet**和**doPost**方法都有两个参数，分别为**HttpServletRequest** 类型和**HttpServletResponse** 类型。
- **HttpServletRequest**提供访问有关请求的信息的方法，例如表单数据、**HTTP**请求头等等。
- **HttpServletResponse**除了提供用于指定**HTTP**应答状态（**200**，**404**等）、应答头（**Content-Type**，**Set-Cookie**等）的方法之外，最重要的是它提供了一个用于向客户端发送数据的**PrintWriter** 。
- 对于简单的**Servlet**来说，它的大部分工作是通过**println**语句生成向客户端发送的页面。
- 必须导入**java.io**包（要用到**PrintWriter**等类）、**javax.servlet**包（要用到**HttpServlet**等类）以及**javax.servlet.http**包（要用到**HttpServletRequest**类和**HttpServletResponse**类）。
- **doGet**和**doPost**这两个方法是由**service**方法调用的，有时你可能需要直接重载**service**方法，比如**Servlet**要处理**GET**和**POST**两种请求时

JSP

- JavaServer Pages（JSP）是一种实现普通静态HTML和动态HTML混合编码的技术。
- JSP并没有增加任何本质上不能用Servlet实现的功能。但是，在JSP中编写静态HTML更加方便，不必再用 println 语句来输出每一行HTML代码。更重要的是，借助内容和外观的分离，页面制作中不同性质的任务可以方便地分开：比如，由页面设计专家进行HTML设计，同时留出供Servlet程序员插入动态内容的空间。
- JSP一般在Web服务器端被编译成Servlet后执行
- 通过定制的Tag实现扩展

JSP Constructs 1

- JSP文件有固定的文件名后缀.jsp
- 注释 <%-- 注释 --%>
- Declaration <%! int x = 0; %>
- Expression <%= expression %>
  - 向Response流中输出数据
  - 类似于在浏览器中执行print
  - 在表达式语句后面不能有分号
- Scriptlets – 包含java代码
  - <% 代码片段 %>

```
<% if (value.getName().length != 0) { %>
    <H2>The value is: <%= value.getName() %></H2>
<% } else { %>
    <H2>Value is empty</H2>
<% } %>
```

JSP Constructs 2

- 所有JSP页面中都隐含存在几个固有的对象
  - “request” – Browser’s Request Object
    - Use to get HTTP headers, length etc..
  - “response” - HttpResponse Object
  - “session” – internal HttpSession Object
  - “pageContext”
  - “application”
  - “out”, same as <%= %>
  - “config” – servlet configuration
  - “page”
  - “exception”

JSP Constructs 3

- **JSP Directives**
  - 是针对**JSP**的指示不会产生任何输出

“page” directive

<%@ page import=“java.io.*” %>

通常用来引入**java**类的路径

“include” directive

<%@ include file=“header.htm” %>

用来包含一些静态文件

“taglib” – 列出标签库描述文件的位置

在使用特定标签库的时候需要

在Web应用中使用Java Beans

- 一般用来实现数据传递和业务组件
- 与Java Beans在Swing和AWT中的用法相似
- 必须没有构造函数或者构造函数没有参数
- 每个Bean属性都必须有setter和getter方法
- JSP标签也使用Java Bean

JSP Actions

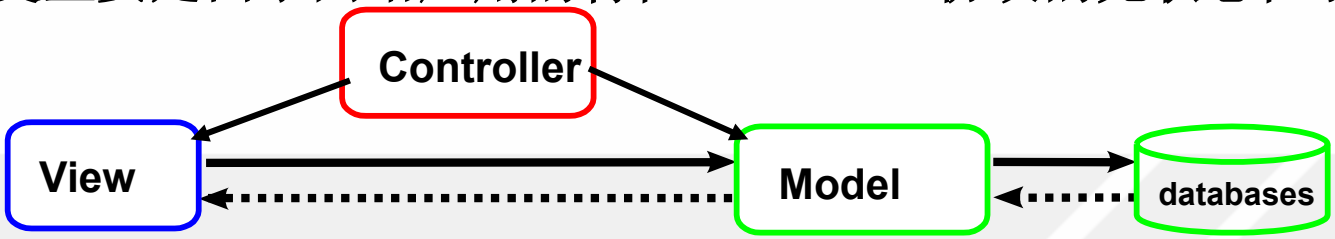
- JSP actions是一些会影响输出流的特殊标签，通常用在Java Bean上
  - 常用的JSP action
    - <jsp:useBean>, <jsp:getProperty>, <jsp:setProperty>
    - 以下代码用来在输出流中显示student bean的lastName属性
    - <jsp:useBean id="student" scope="request" class="StudentValue" />
    - <jsp:getProperty name="student" property="lastName" />

J2EE设计模式(JSP为中心)

- 优点
  - 非常容易结合业务逻辑(JSP:UseBean)、服务器处理过程和HTML(<html>)，在JSP页面中同时实现显示、业务逻辑和流程控制，从而快速完成应用开发
- 缺点
  - 应用的实现一般是基于过程的，一组JSP页面完成一个业务流程，如果要进行改动，必须在多个地方进行修改。这样非常不利于应用扩展和更新。
  - 由于应用不是建立在模块上，业务逻辑和表示逻辑混合在JSP页面中，没有进行抽象和分离。不利于应用系统业务的重用和改动。

J2EE设计模式(Model 2)

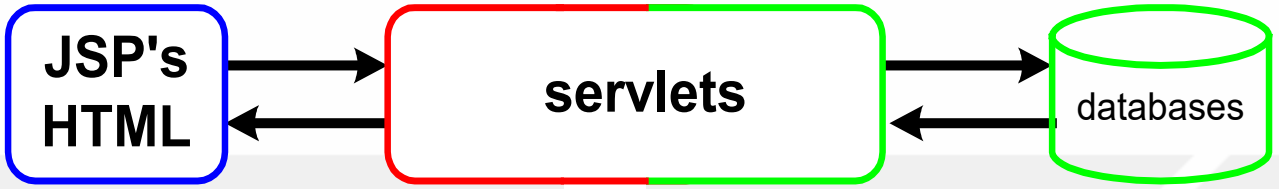
- 目前最通用的Web应用的架构是Model 2
- Model 2是经典的MVC（模型－视图－控制器）模型的Web应用变体，这个改变主要是由于网络应用的特性——HTTP协议的无状态性引起的



  - Model – 业务逻辑和数据：基于输入和定义的业务流程执行计算 或其它操作 (javabean or ejb)
  - View – 表示：显示输入输出数据 (可以是 HTML，jsp，也可以是一个 windows 应用)
  - Controller – 协调view 和 model，在它们之间交换数据(action or servlet)
- Model 2的优点
  - 可维护性
    - 业务逻辑和表现逻辑分离
    - 该架构应用较多，熟悉的人较多（如Struts）
  - 可重用性
  - 安全性
    - 所有的请求会被转发到一个固定点，Controller可以统一处理安全

J2EE设计模式（ MVC 实现方法1 ）

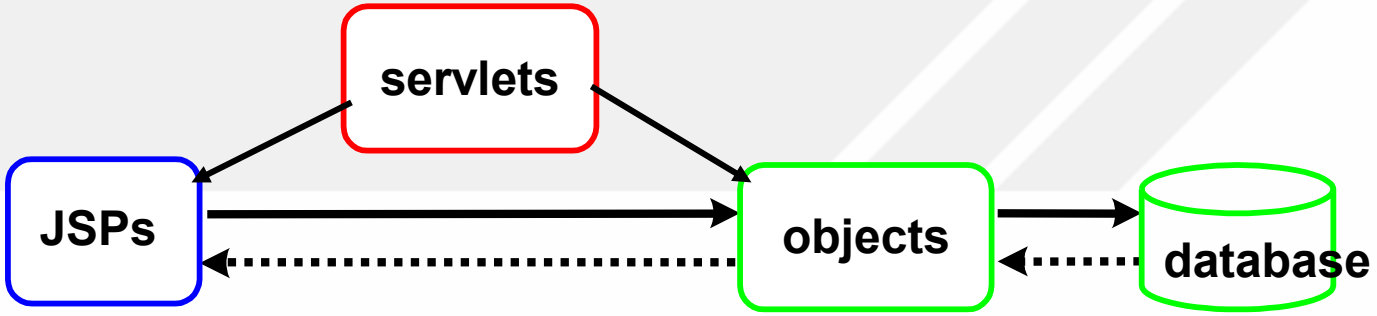
- MVC 的隐含式 J2EE 实现



- View 由 JSP 和 HTML 页面组成
- Servlet 可以同时是 controller 和 model
  - 直接存取数据

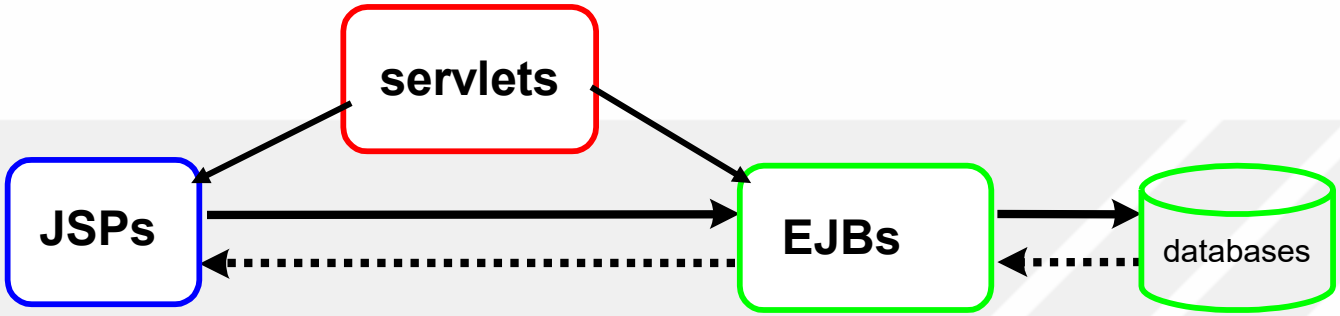
J2EE设计模式（MVC 实现方法2）

- 将业务逻辑处理放在 **Servlet** 之外
- **HttpServlet** 的子类应该仅做 **servlet** 份内的工作
  - 管理 **request**、**response** 及 **HttpSession** 对象
- 将业务逻辑写在传统的**Java**类
  - 仅传送普通的 **Java** 类，不是 **servlet** 相关的类（例如 **request, response** 或 **session**）
  - 比 **servlet** 容易开发、测试和重用



J2EE设计模式（ MVC 实现方法3 ）

- 典型的 J2EE 方案

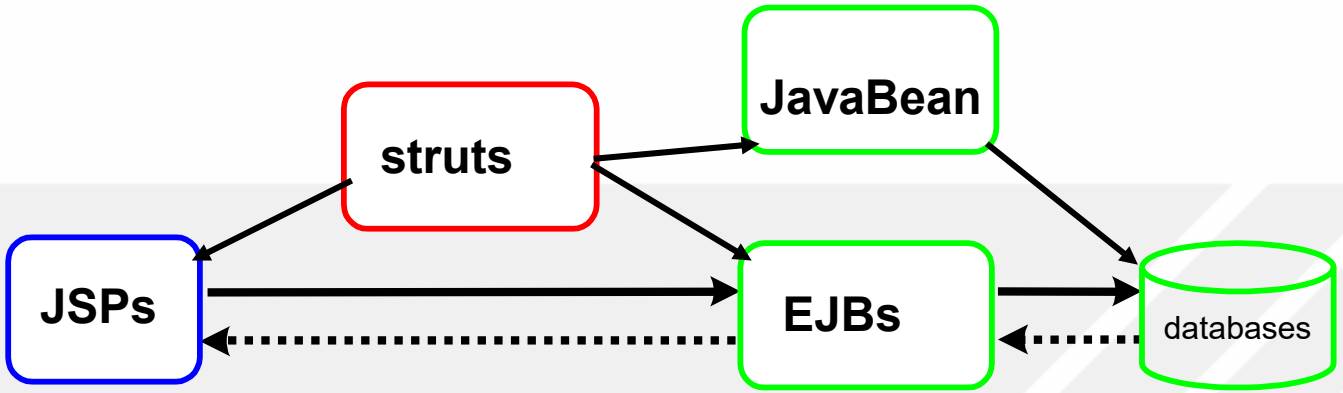


- View 由JSP 和静态 HTML 组成
- Controller 是 servlet
- Model 是 EJB

- 很象使用传统的 Java 对象 – 只是运行在远程

J2EE设计模式（ MVC 实现方法4 ）

- 典型的Struts 方案

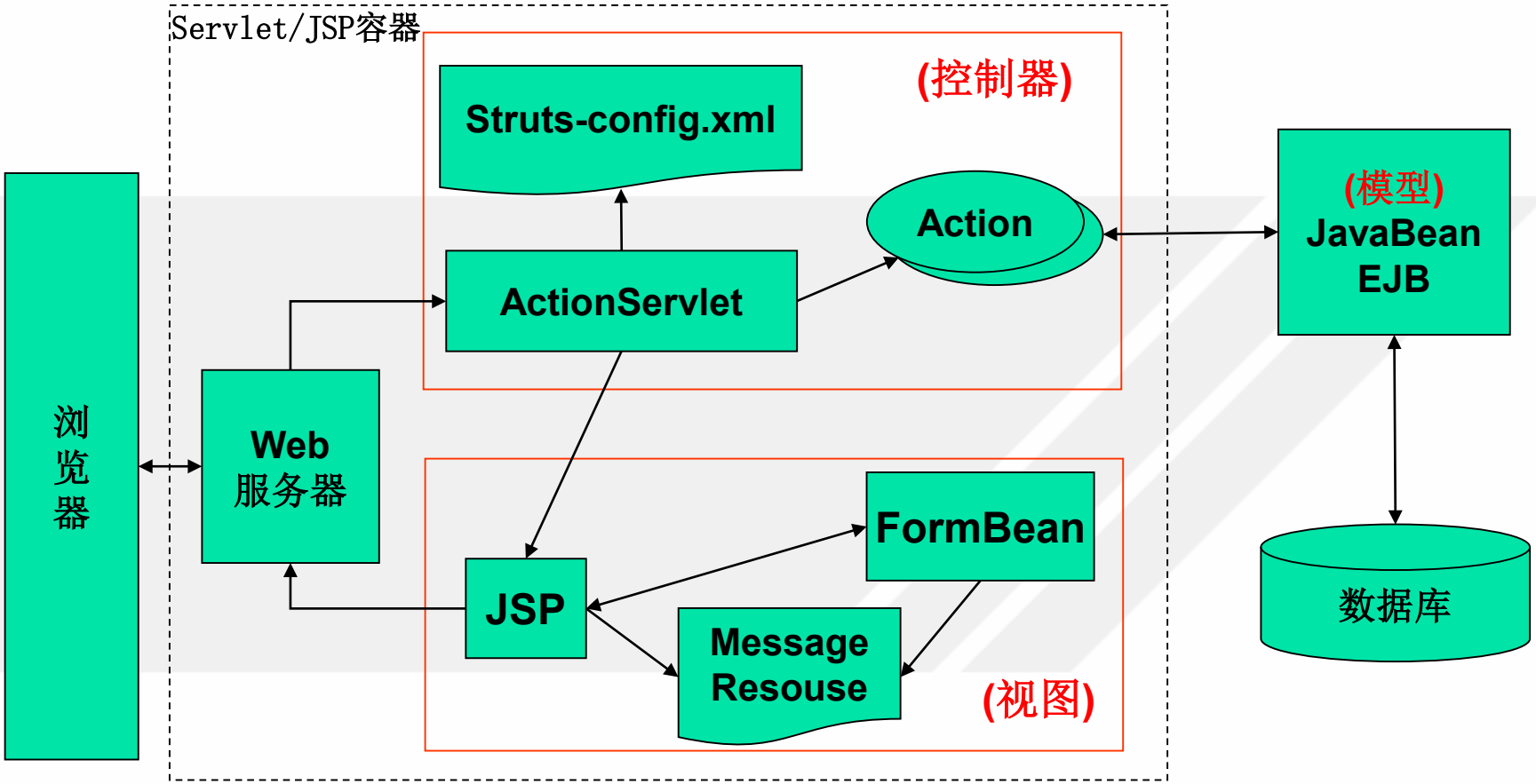


- View 由JSP 和静态 HTML 组成
- Controller 是 Struts
- Model 是 EJB或javabean

开发框架

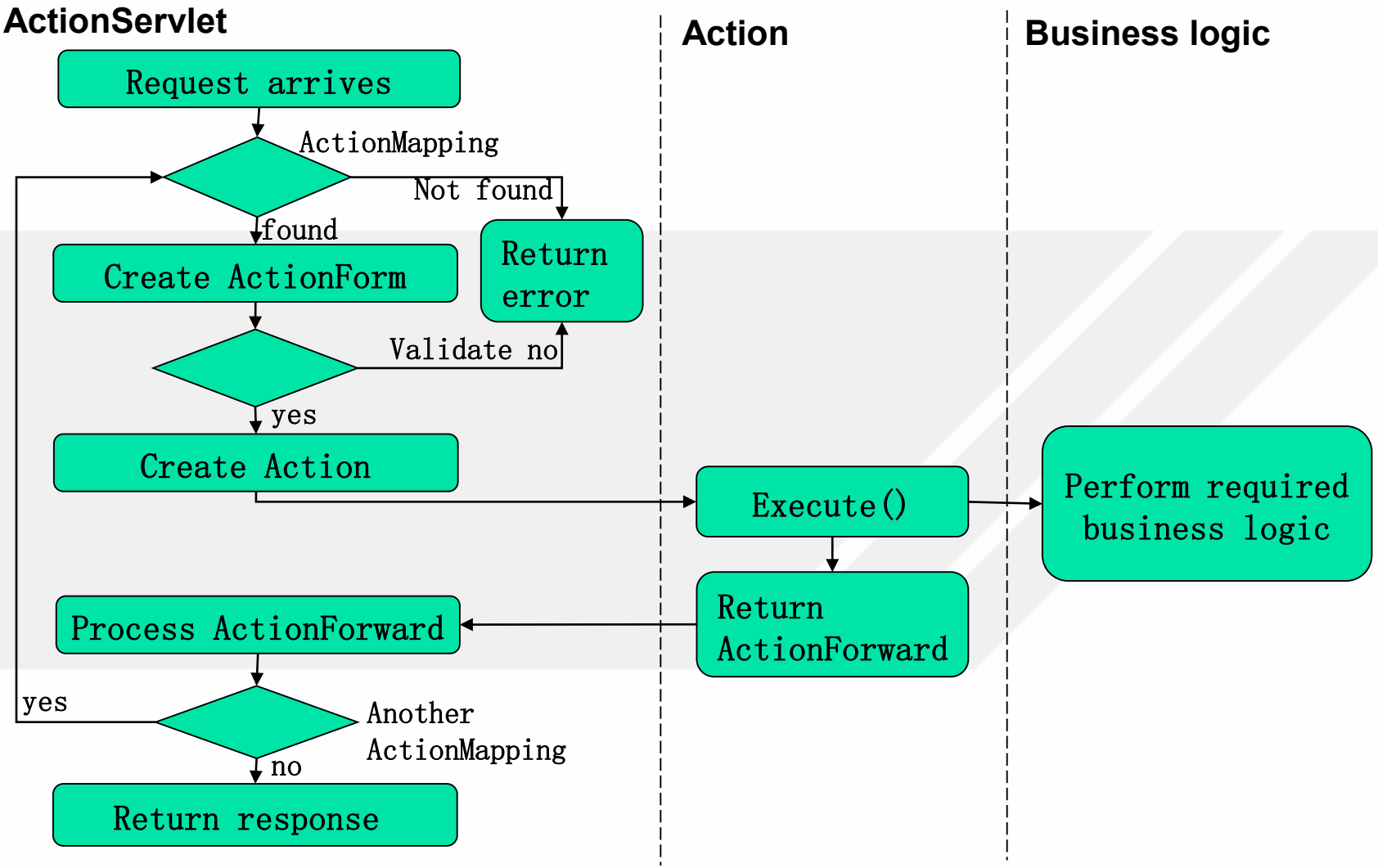
- JSF+EJB+JPA
- SSH(struts+Spring+Hibernate)
- SSM(SpringMVC+Spring+MyBatis)
- Spring Boot
- Spring Cloud

Struts是一个成熟的基于MVC的Web应用框架。



Struts 1.x应用架构

Struts 1.x工作流程

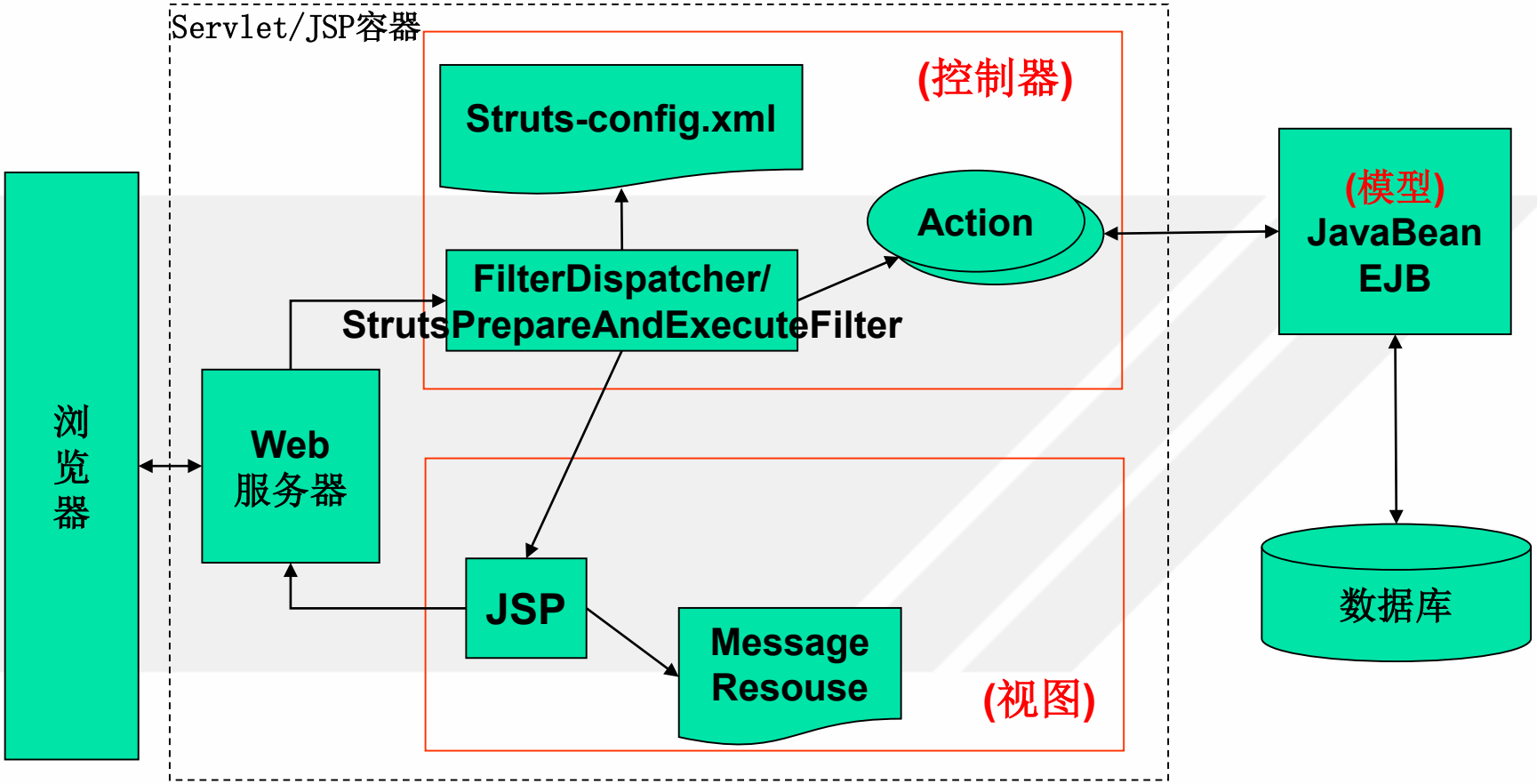


Struts 2.x

- 架构与struts 1.x不同，使用Filter实现
- Action可以用POJO实现，与servlet解耦

Struts 2.x

Struts是一个现成的、优秀的、基于MVC的Web应用框架。



Struts 2.x应用架构

J2EE应用部署

- 所有J2EE模块会被打包成一个应用
  - 应用打包构成一个EAR(Enterprise Archive)文件
  - EAR文件在META-INF目录下有一个部署描述文件application.xml
- 不同的应用服务器有不同的部署方式
  - 将文件Copy到应用服务器的特定目录
  - 商用的应用服务器一般提供管理界面部署应用，可以设置较多运行参数

部署描述符

- 提供容器如何管理和控制J2EE组件行为的操作指令
  - 事务性
  - 安全性
  - 持久性
- 通过配置实现组件行为的定制 (不需要通过代码进行定制)
  - **XML**文件
- 使代码具有可移植性
- 三种部署描述符
  - **EAR: application.xml**
  - **WAR: web.xml**
  - **EJB: ejb-jar.xml**

Web部署描述符

- /WEB-INF/web.xml
  - J2EE标准的一部分
- 定义一个web应用中的servlet
  - 给servlet和URL之间作映射
  - 一个servlet可以映射到多个URL上
- 定义web应用中可以使用的资源
- 定义安全
- 定义其他内容
  - 设定Welcome file列表
  - 设定会话超时
  - 错误页面映射