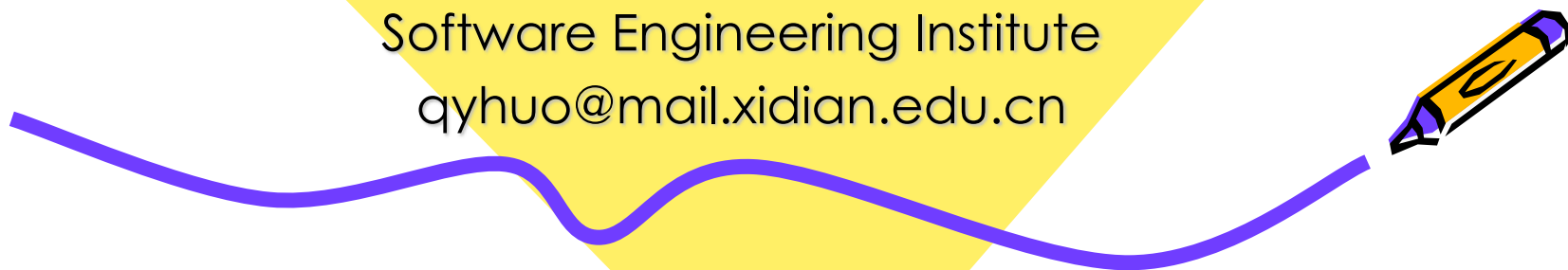




Web应用架构

Qiuyan Huo 霍秋艳
Software Engineering Institute
qyhuo@mail.xidian.edu.cn



Introduction

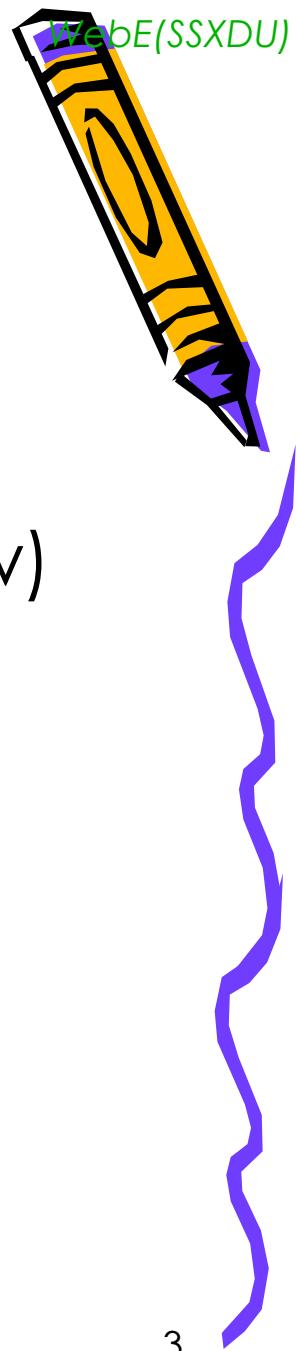
- We build a ...
 - Bird house
 - Garden house
 - One-family house
 - Hotel****
 - Church
- Basic tools: hammer, nails, saw, wood -> go!
- Draw a plan, Construct
- make a sketch(草图), discuss, change plan, draw outside view, compute static's construct
- Make sketches, models, plans, detail function units (kitchen, lifts, stairs), plan interior decoration of entrance hall, ...
- Design of acoustics(声学), check spiritual(精神上的) aspects of design, impact of church spire(尖顶) on city skyline



Objectives of Architectures



- Well-structured(结构良好的)
- Encapsulation / hiding(封装/隐藏)
- Performance (e.g., efficient traffic flow)
- Extensibility(可扩充性)
- Mobility(灵活性)



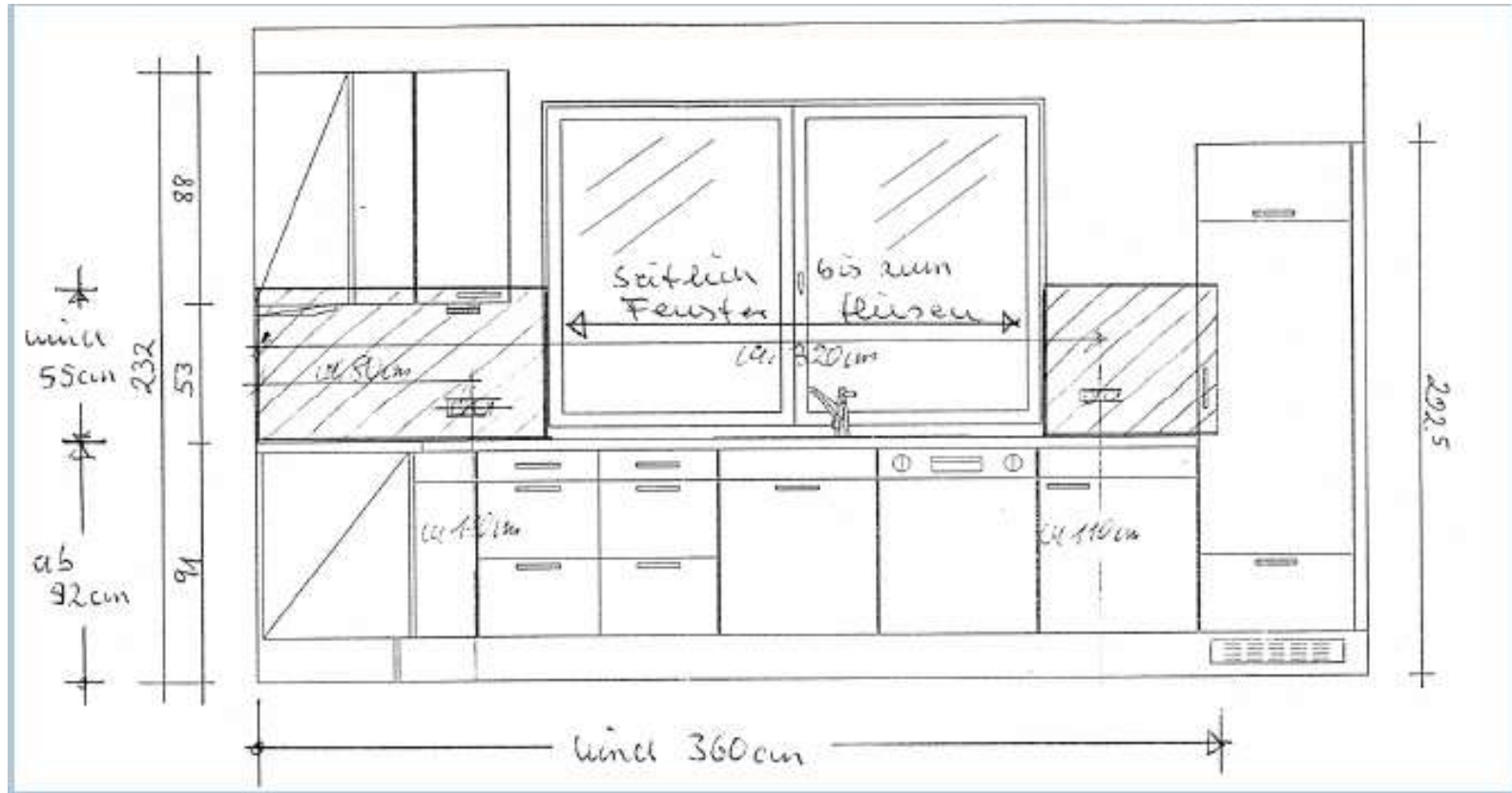
Architecture: floor plan



Architecture: sheer(垂直的) plan

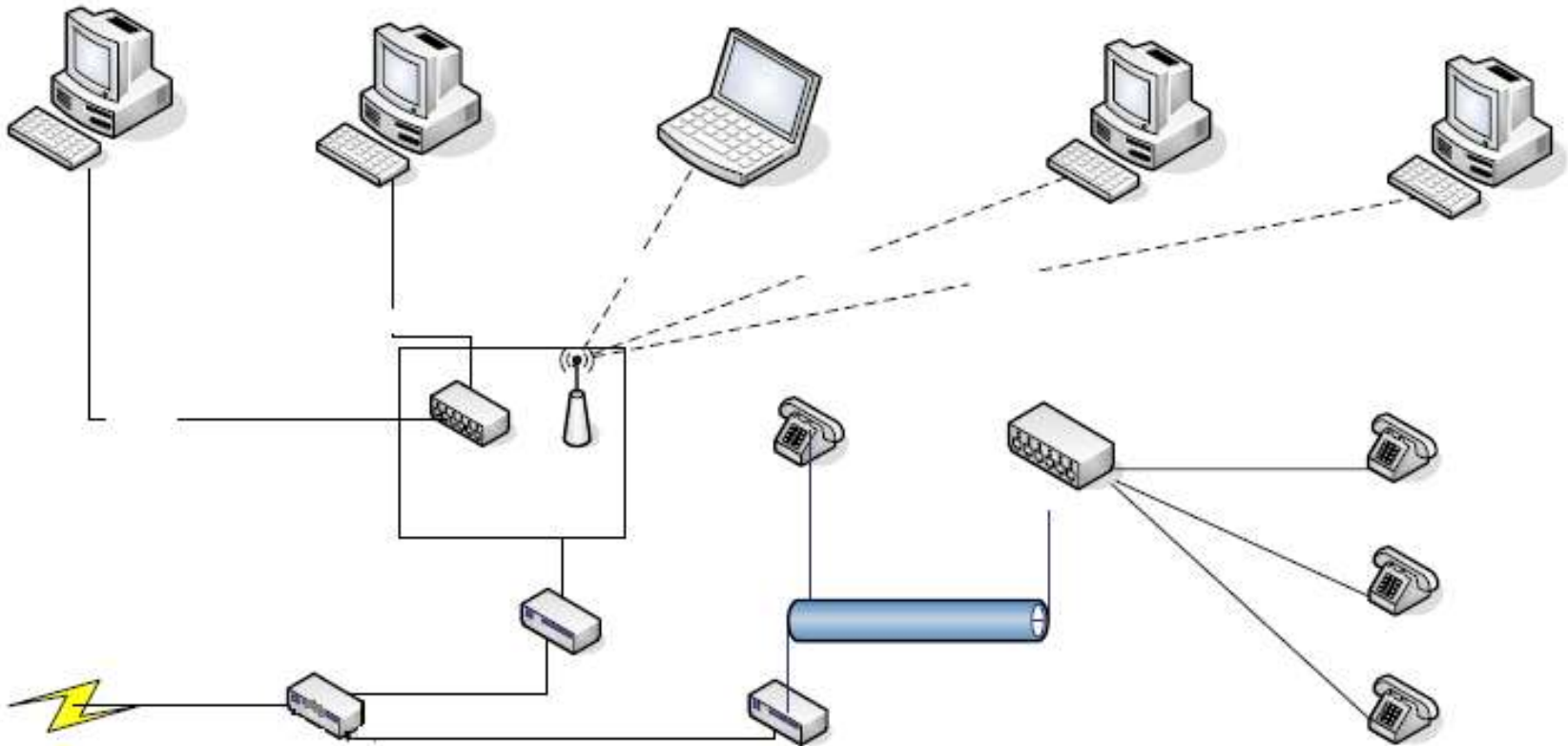


Architecture: tiler(铺瓦工) plan





Architecture: network plan

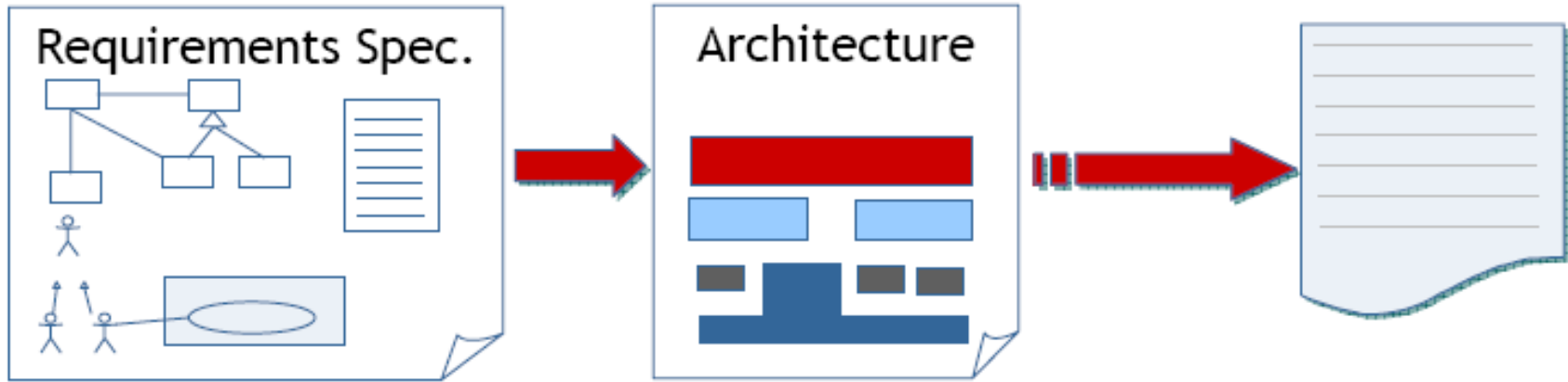


Architecture of buildings

- **objectives** for a realization
- One architect – several architectural draftsmen
– many construction worker
- Different abstraction levels of architectures
 - District in town
 - Complex of buildings
 - Single building
 - Floor
 - Room
- Different views on architectures
 - **Physical** views (floor plan, sheer plan, 3-D view)
 - **Logical** views (network plan, waste water plan, static)
 - Different levels of detail



Software Architecture



- Programming-in-the-large
 - Overall structure of the system (系统整体架构、体系结构)
 - Usually called software architecture (软件架构)

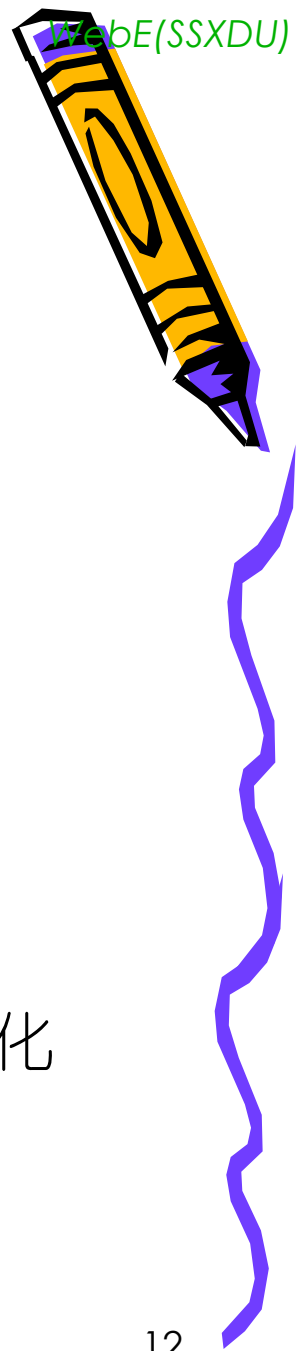
软件架构的特性

- 来源于需求规格说明
 - 功能和非功能需求
- 几个架构师，很多程序员
- 可运行软件的抽象
- 定义模块或组件以及它们之间的交互
- 抽象层次的问题
 - 对于盖房设计图而言，没有清晰地区分不同组件类型 (building - floor - room)
 - 软件架构中任何事物全部都是软件—不同的抽象



软件架构的特性

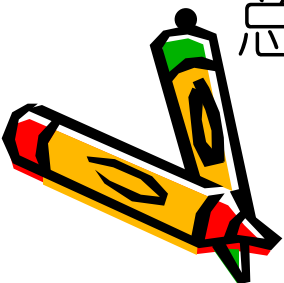
- 系统的特定视图
- 系统的多种架构视图可能并存
- 架构视图的适合度取决于特定视角(e.g., distribution, communication, security)
- 和建筑物架构比较
 - 软件架构更多描述逻辑组件
 - 只有部署图为物理视图
 - 软件架构更动态性——复杂的行为和不断地演化



Web应用架构

- Web应用架构及其特性
- 模型驱动架构
- 层次架构
 - 2/3/N层架构
- 集成架构
 - 门户/EAI/SOA
- 面向数据的架构
 - 数据为中心/Web文档管理/流媒体数据

总结与展望



WEB应用架构及其特性



设计模式(Patterns)

- 在特定设计环境中反复出现的设计问题并给出相应的解决方案
 - 描述特定问题中组件及其职责、相互联系和相互作用
- 使用已有的设计知识来支持开发高质量的软件系统

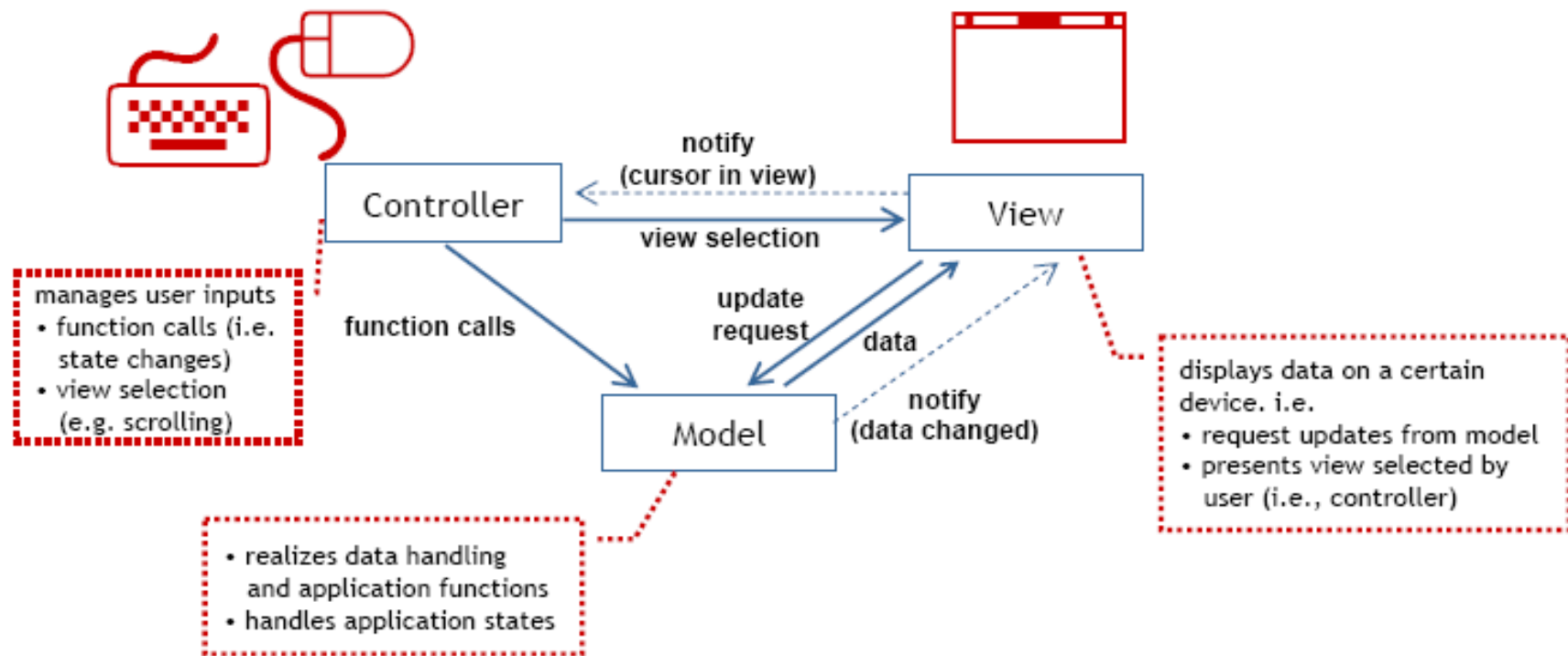


设计模式(Patterns)

- 三个不同的抽象层次
 - 架构模式 (Architectural Pattern)
 - 系统的基本结构方案
 - 包括子系统架构及其职责、关系和相互作用(E.g., MVC)
 - 设计模式 (Design Pattern)
 - 特定环境下, 解决通用设计问题的组件结构、关系和相互作用(E.g., Publisher-Subscriber)
 - 编程模式 (Idiom)
 - 在程序设计语言中某些功能的特定实现方式(e.g., Counted-Pointer in C++)



Model – View – Controller (MVC)



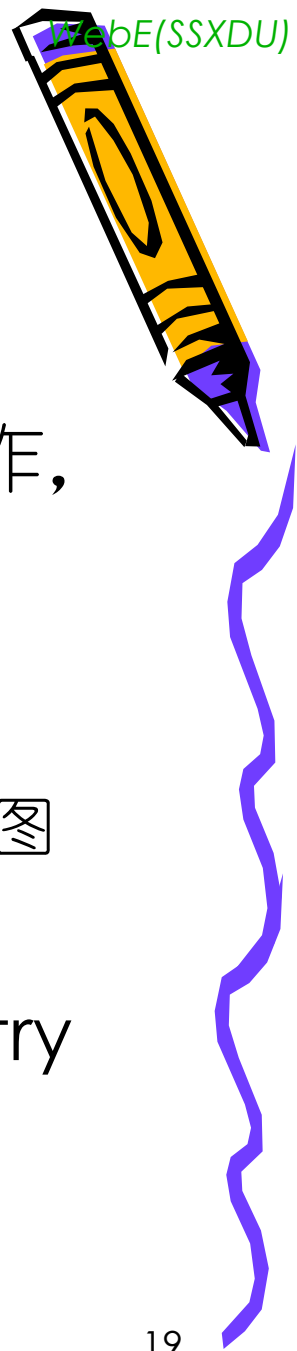
eBusiness应用模式

- 应用服务器 (Application Server) 模式
- 服务器端会话 (Server-side Session) 模式
- 垂直分片 (Vertical Slice) 模式
- 服务器集群故障转移 (Fail-over Through Server Clustering) 模式和任务分配 (Job Draining) 模式
- 心跳 (Heartbeat) 模式
- Web监视器 (Web Inspector) 模式
- 业务上下文感知目标检索 (Business Context-Aware Object Retrieval) 模式
- 预制业务对象 (Prefabricated Business Objects) 模式



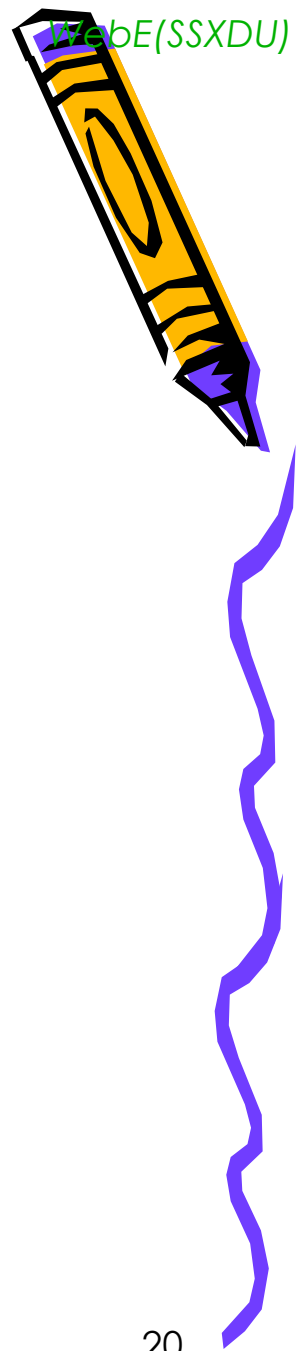
框架(Framework)

- 另一种重用架构知识的方式
- 框架是一组组件的综合，这些组件相互协作，为一类相关应用提供了可重用的框架结构
 - 通用功能已经被实现的可复用软件系统
 - 可以进行实例化
 - 作为特定领域应用的基本架构和基本功能的蓝图
- 框架中包含的知识可以被应用全部采纳
- E.g., Struts、Webwork、Spring、Tapestry 和JSF, Rails, Backbone,

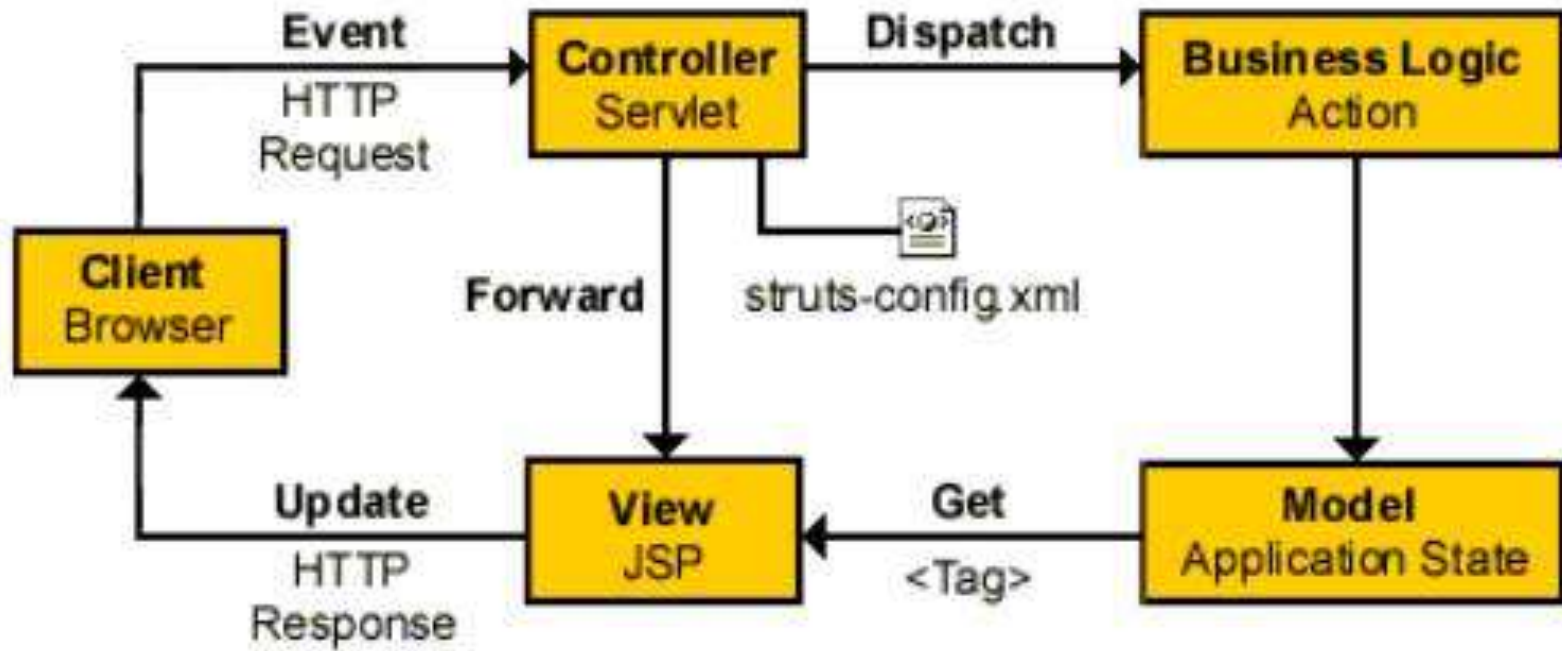


框架(Framework)

- Benefits:
 - 简单重用框架的架构和功能
- drawbacks:
 - 使用需要专业知识
 - 不同框架之间并没有集成标准
 - 开发出的应用对开发商过于依赖等

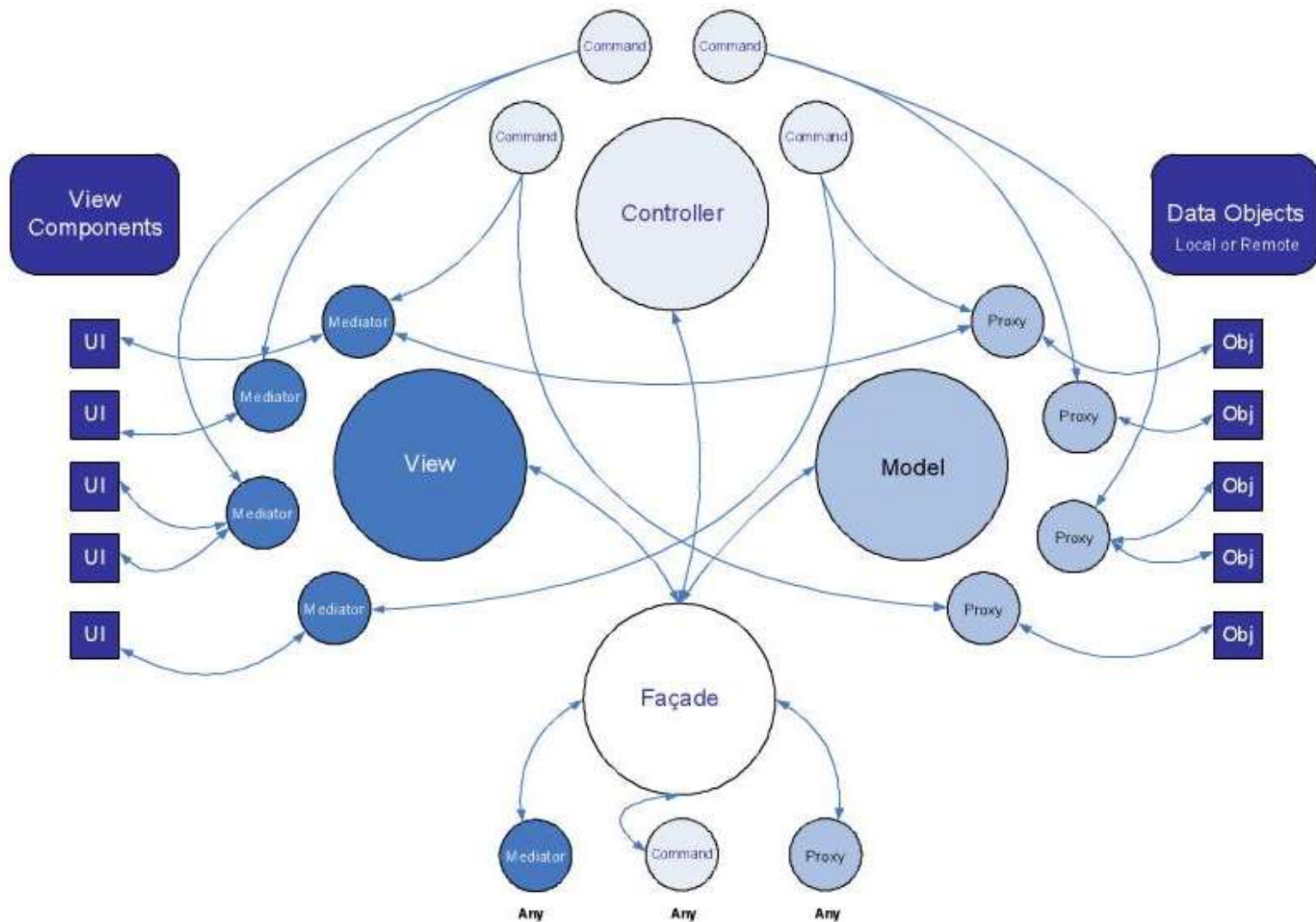


Struts

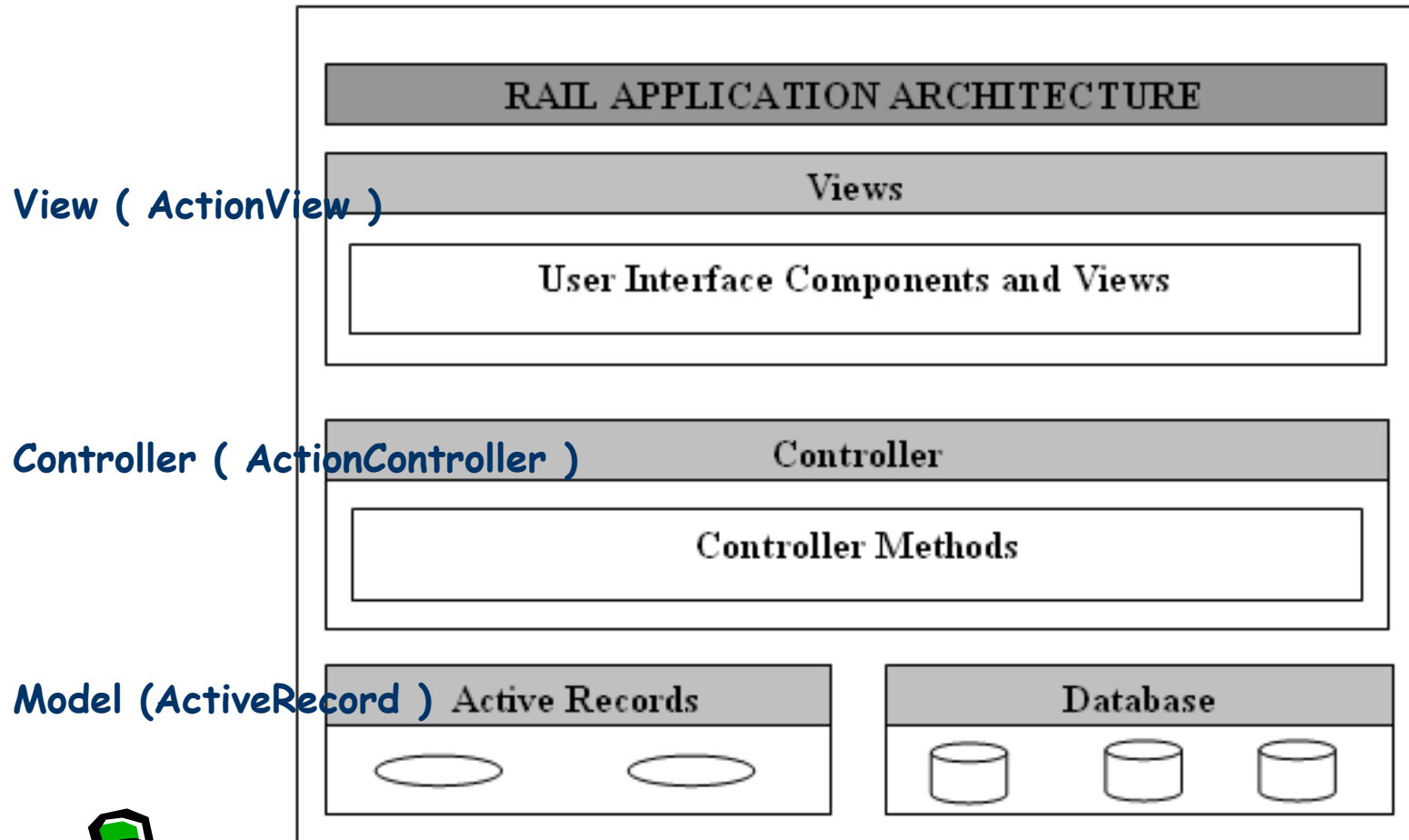


Struts在MVC方面是最好的

PureMVC



Ruby on Rails MVC framework



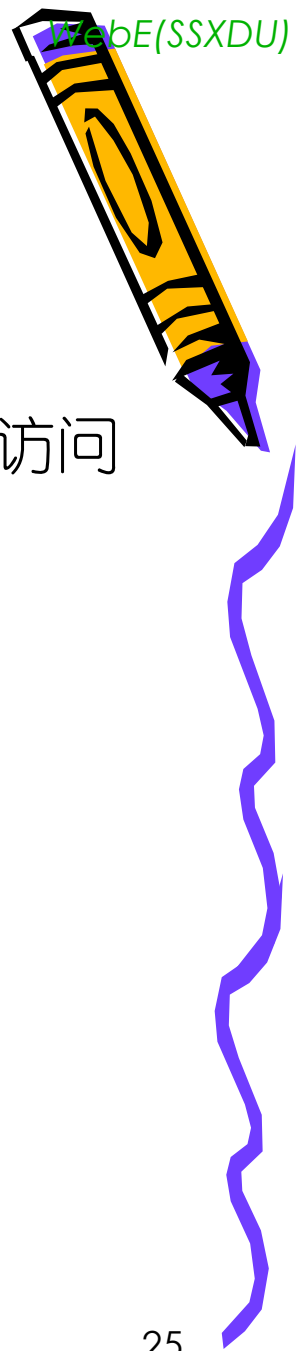
架构分类

- 层次特性
 - 分而治之：将应用根据逻辑划分成不同的部分，然后单独考虑并单独编码实现，每一部分实现不同的功能，并可能分布在不同的主机，不同的操作系统上，通常以层（layer）来描述划分的领域。
 - E.g., Java EE架构、企业应用集成（EAI）
- 数据特性
 - 根据数据的结构化或非结构化特性



架构分类

- 分布对象中间件 (DOM)
 - 基于远程过程调用 (RPC) 机制, 允许透明地访问远程对象。
- 虚共享内存 (VSM)
 - VSM模型支持分布进程访问同样的数据。
- 面向消息中间件 (MOM)
 - 异步消息交互
- P2P
 - 对等点可以直接通信而无需服务器
- 面向服务的中间件 (SOM)
 - 在DOM系统的基础上加上服务的概念



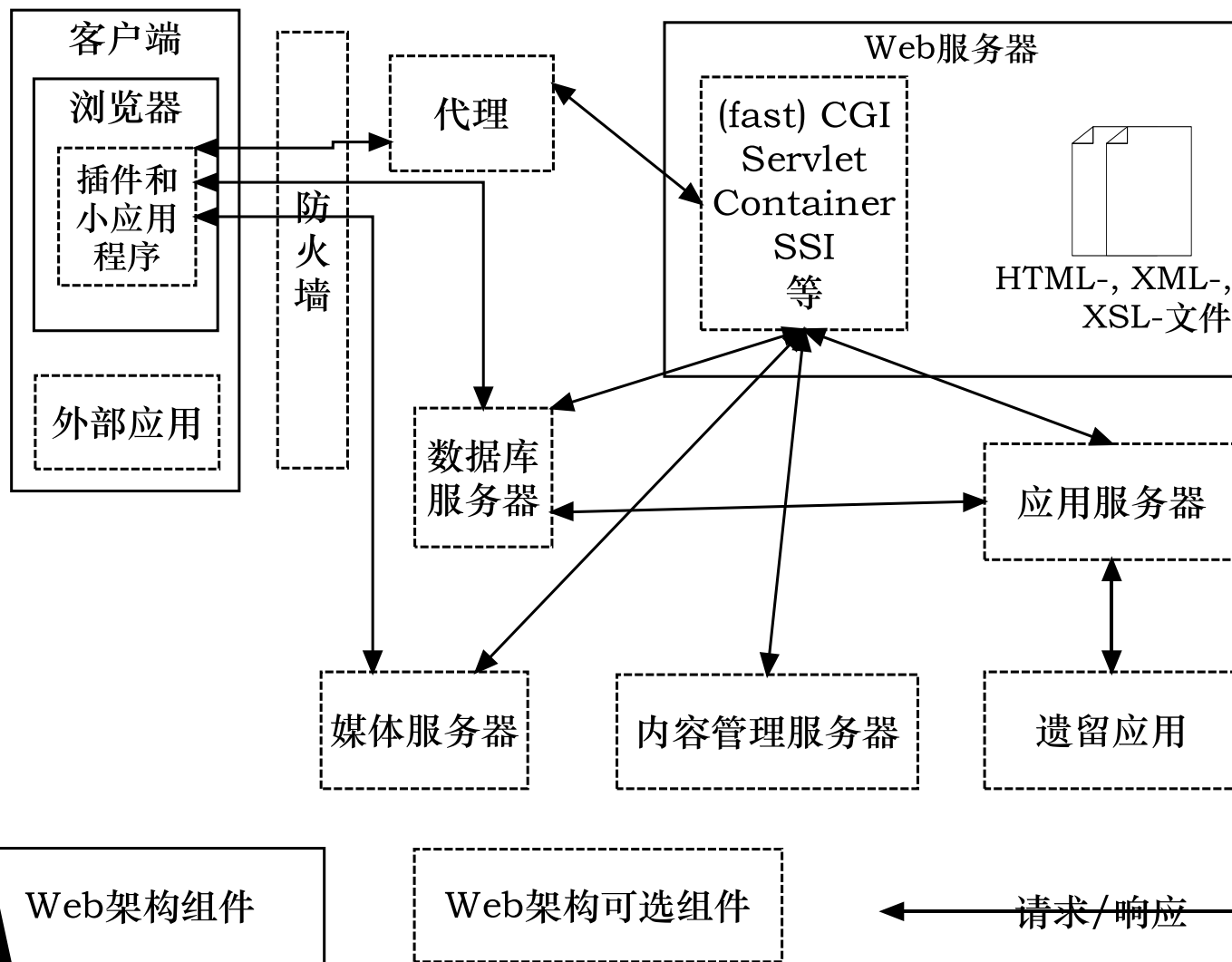
Web应用架构特性

- Web基础架构
 - Web平台架构 (Web Platform Architecture, WPA)
 - Web应用架构 (Web Application Architecture, WAA) 强调Web应用所处的问题域
- WPA
 - 应用服务器如Java EE、.NET平台提供会话处理、协议包装、数据访问等基础服务
 - 特定架构:安全 (如防火墙)、性能 (缓存代理) 或数据集成 (如EAI) 等方面
- 众多不同系统使得更难评估和维护各种质量需求
- 技术架构的异构性和不成熟性

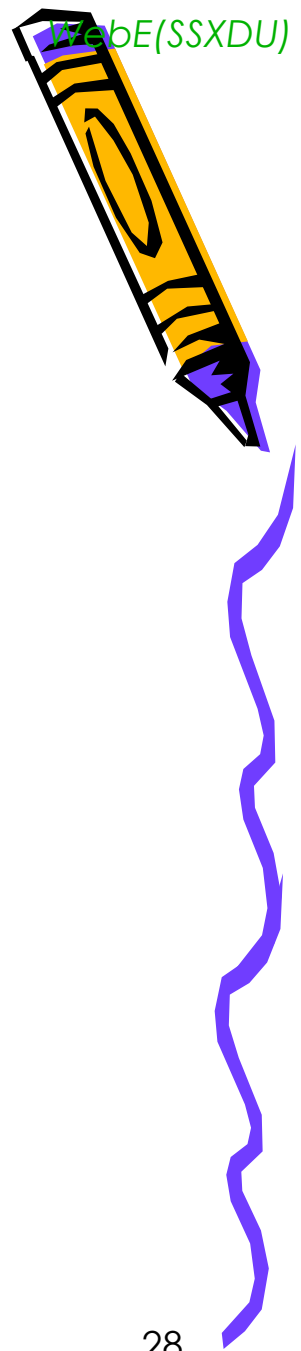
国际化

在构建Web应用架构的时候就要考虑所有这些特性.

Web应用架构--通用组件

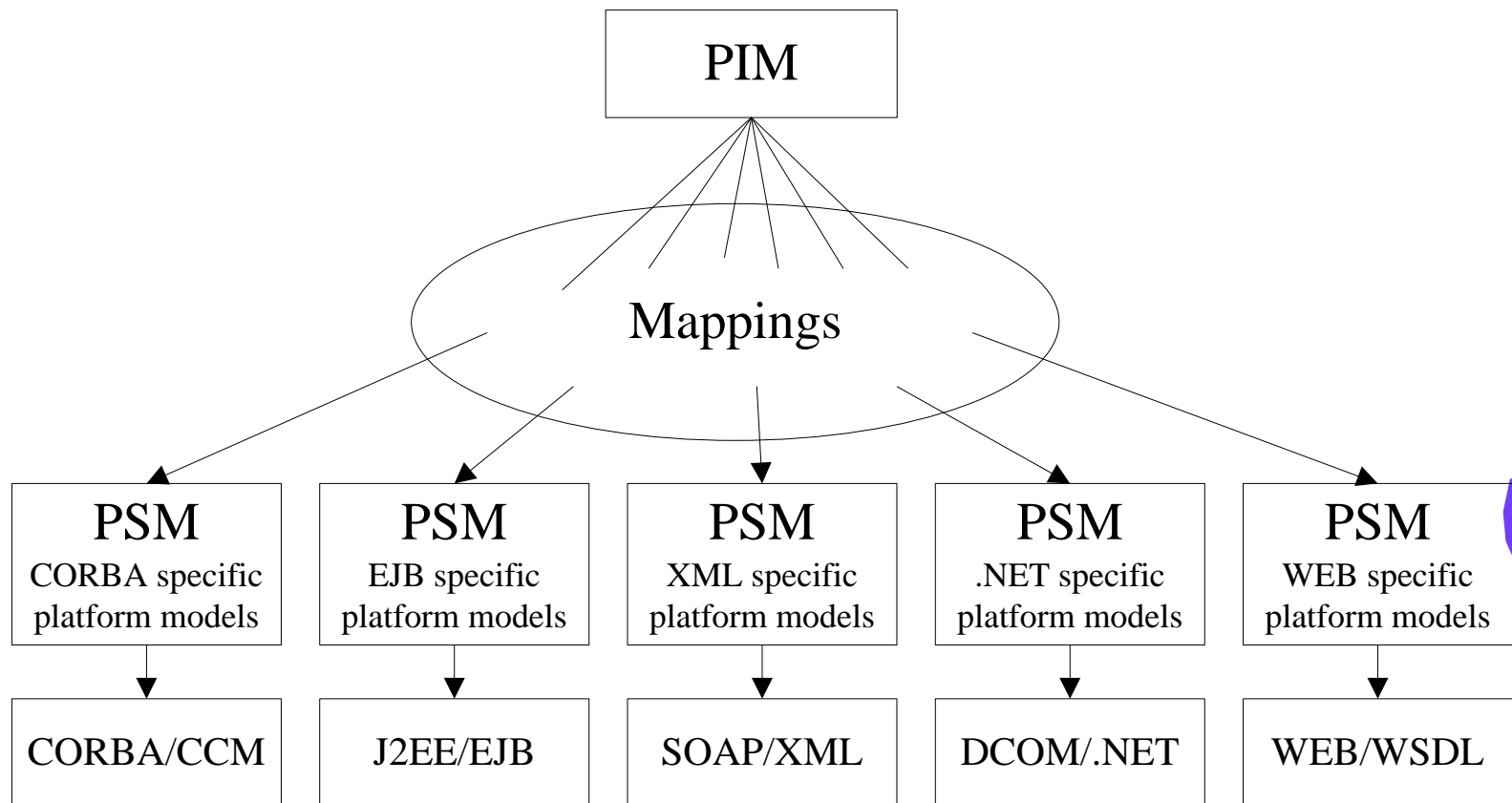


模型驱动架构



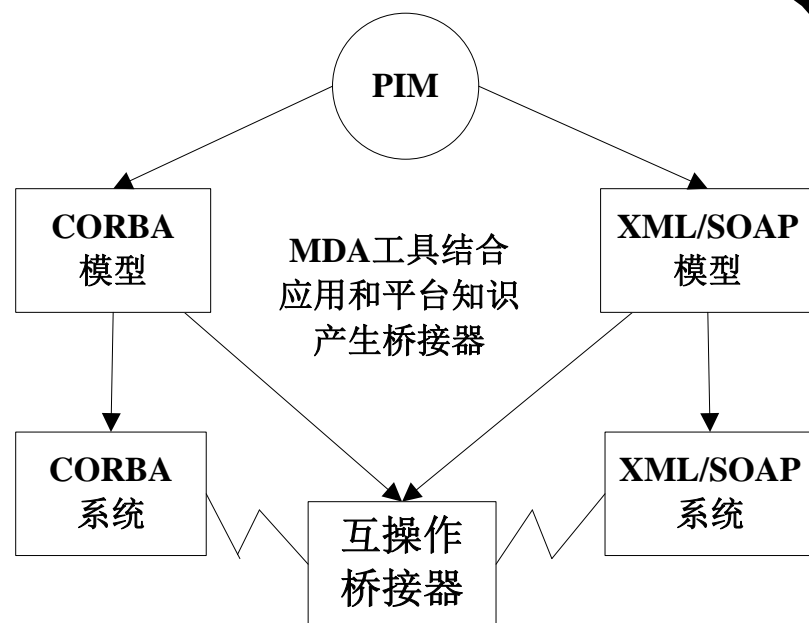
模型驱动架构(MDA)

- MDA是一种开放的，独立于软件供应商的架构，支持众多应用领域和技术平台。



MDA的优势

- 提高生产效率
 - 提高了可重用性
 - 增强可移植性
 - 支持互操作性
-
- 提高了系统的可验证性
 - 便于维护

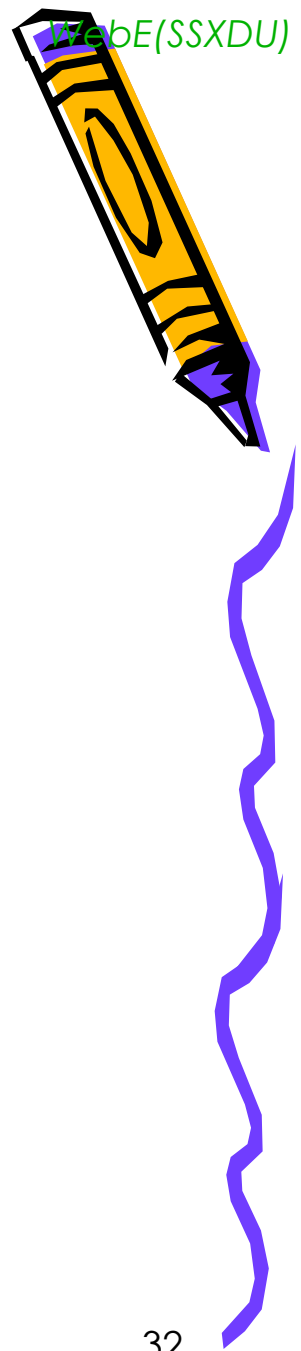


MDA工具

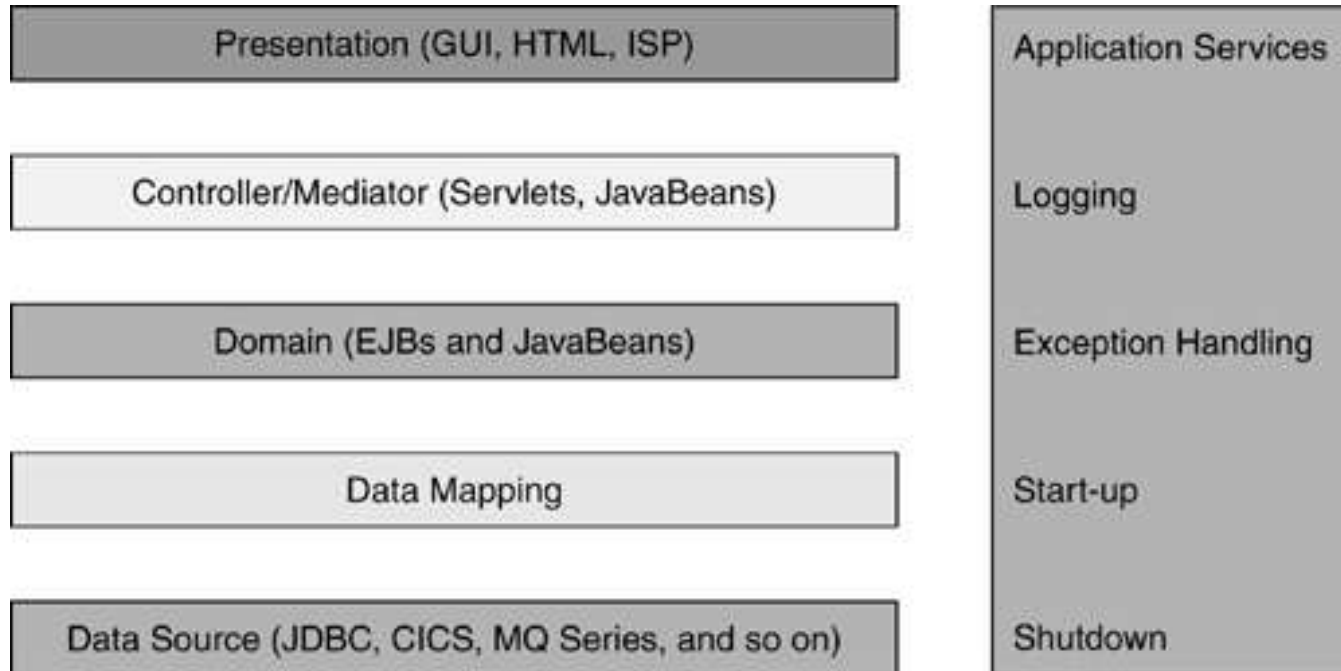
- 完全式，即完全支持整个开发过程，从UML建模，模型转换和代码生成。
 - Metanology Corporation的Meta Development Environment (MDE) ,
Interactive Object Software的Arcstyler,
Compuware Corp的OptimalJ, IBM的Rational XDE
- 不完全式，即支持将从其它模型工具得到的模型生成代码
 - Codagen Technology Corp的Architect,
Telelogic的Tau



层次架构

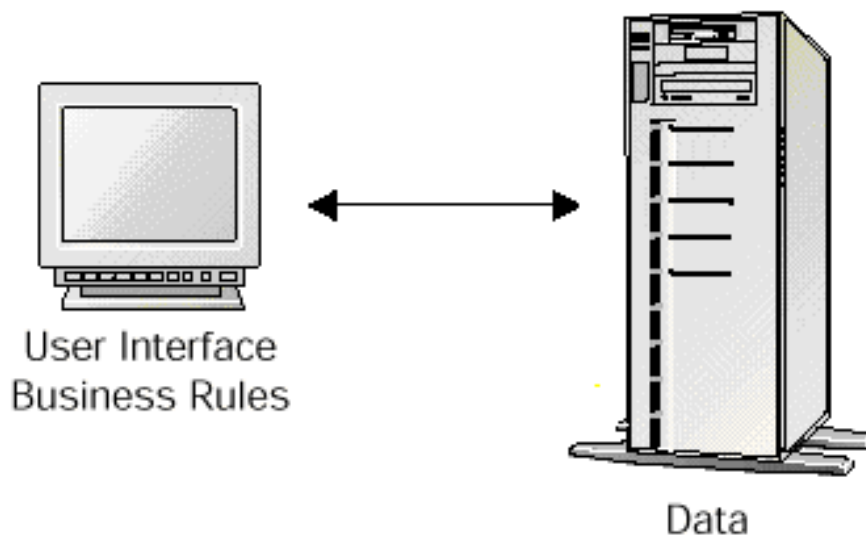


层次架构

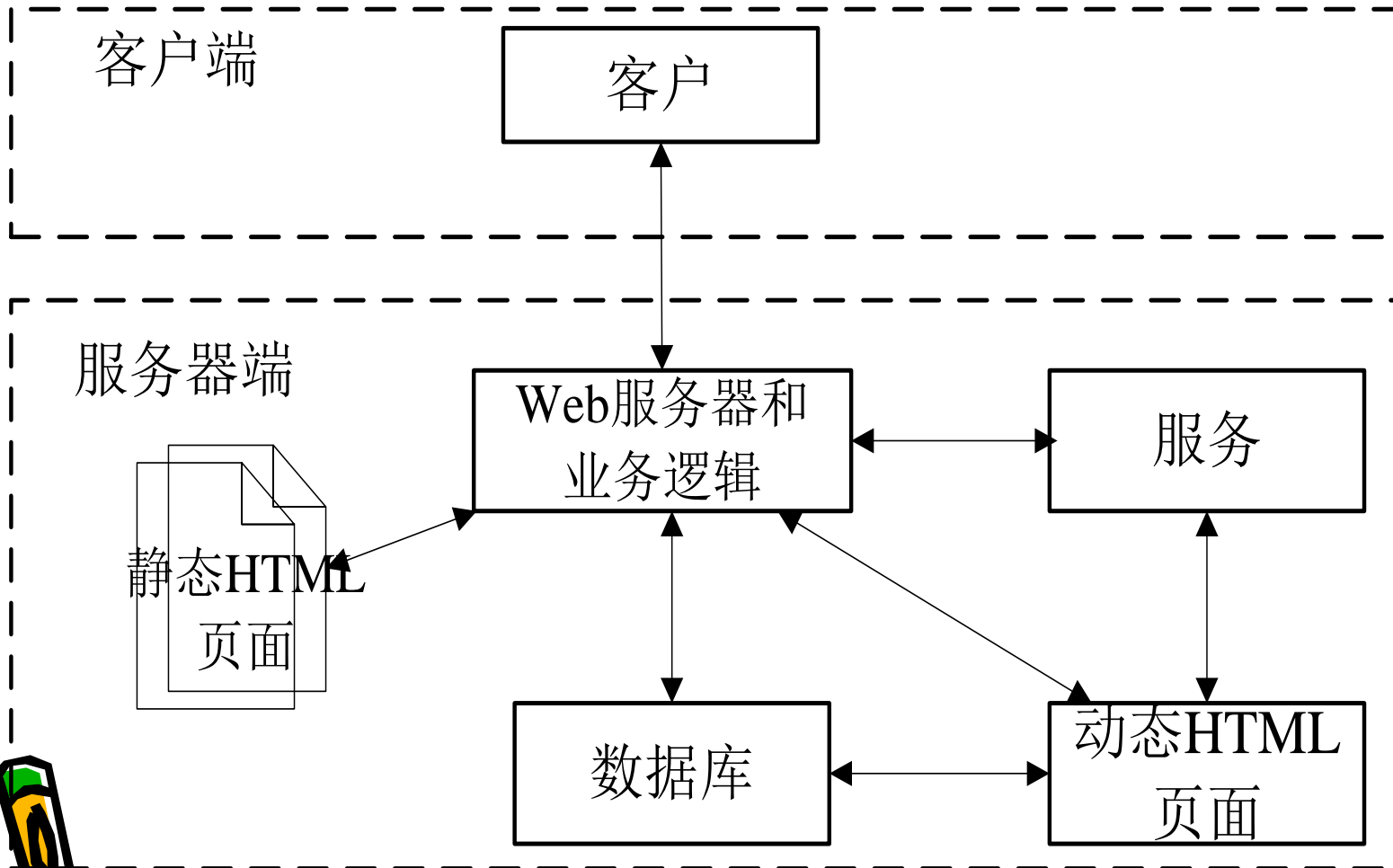


两层架构

- 两层（2-Tier）架构也称为客户/服务器架构，应用逻辑分布在服务器端，向客户端提供服务。

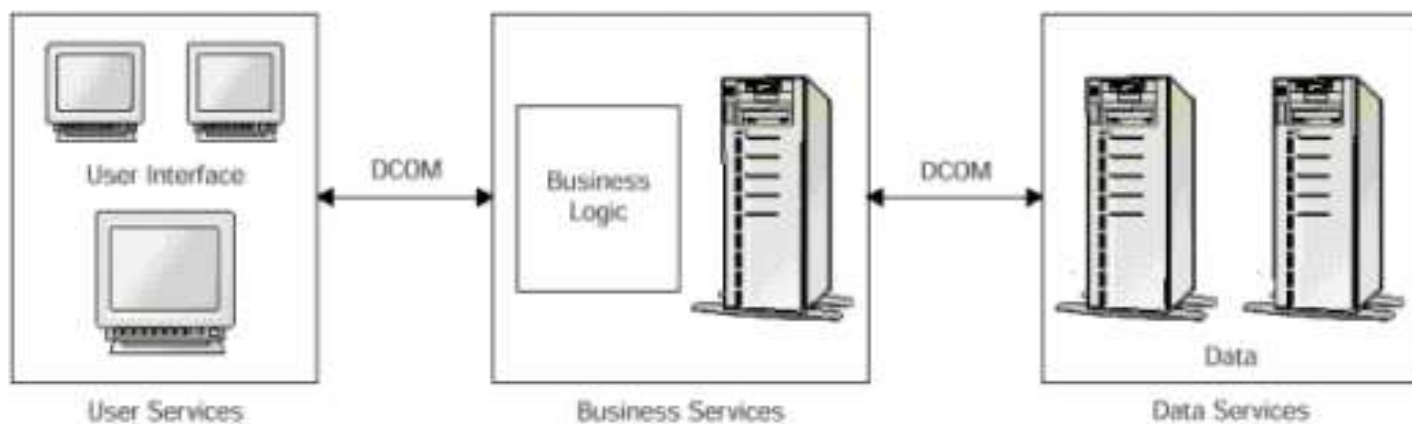
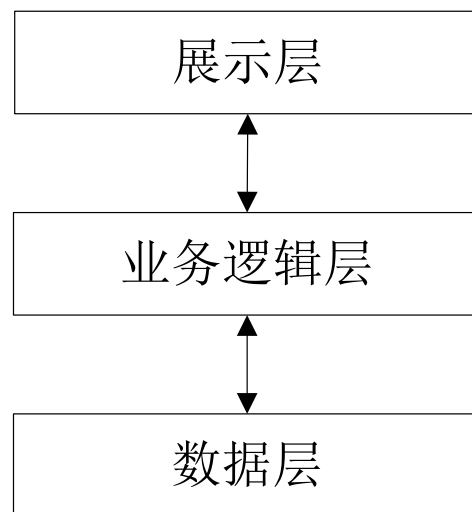


Web应用的两层架构

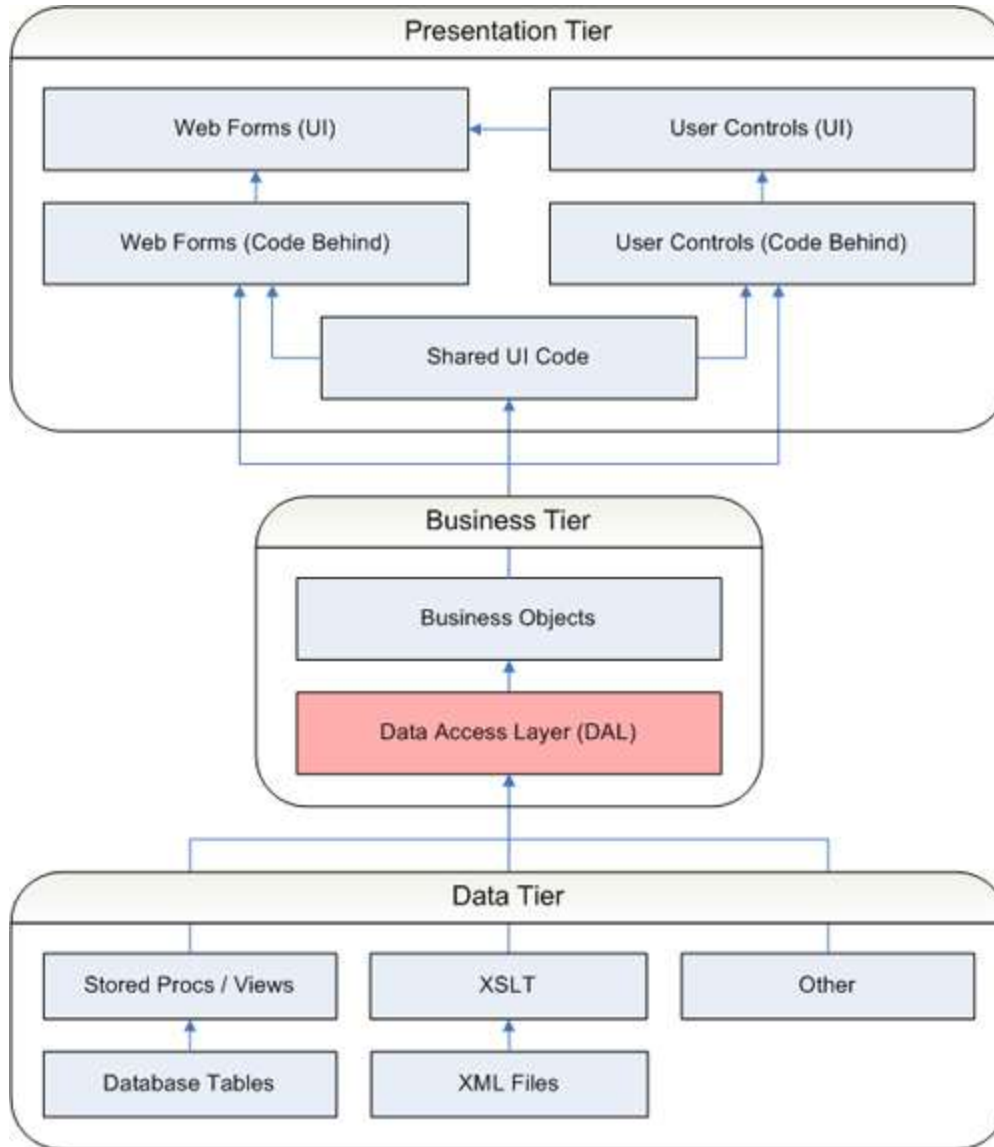


Web应用的三层架构

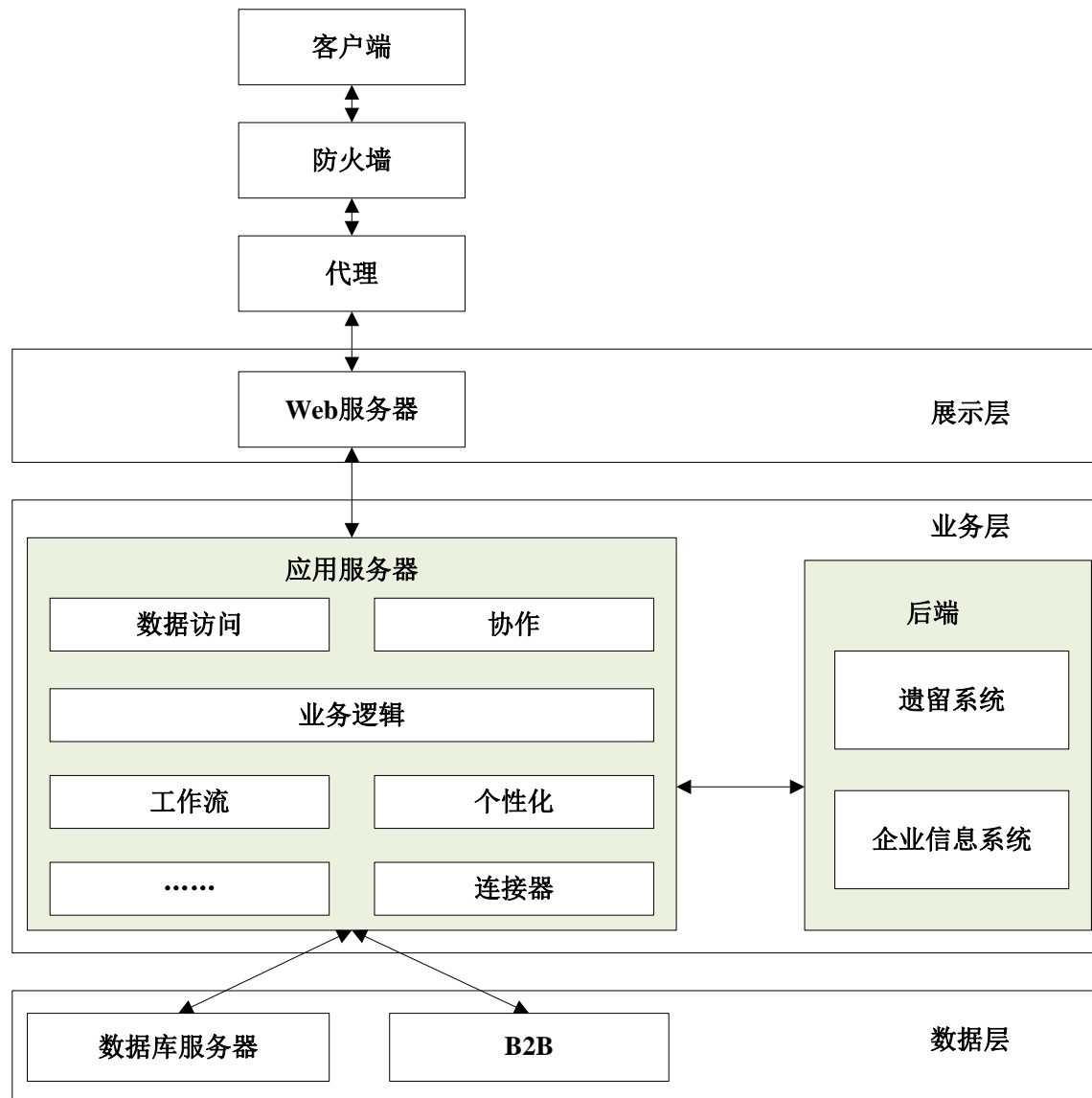
- 将Web应用中组件划分
 - 展示层: 封装和用户或者其它系统交互 (i.e., 浏览器)
 - 应用或逻辑层: 组成业务逻辑
 - 数据层: 封装持久存储



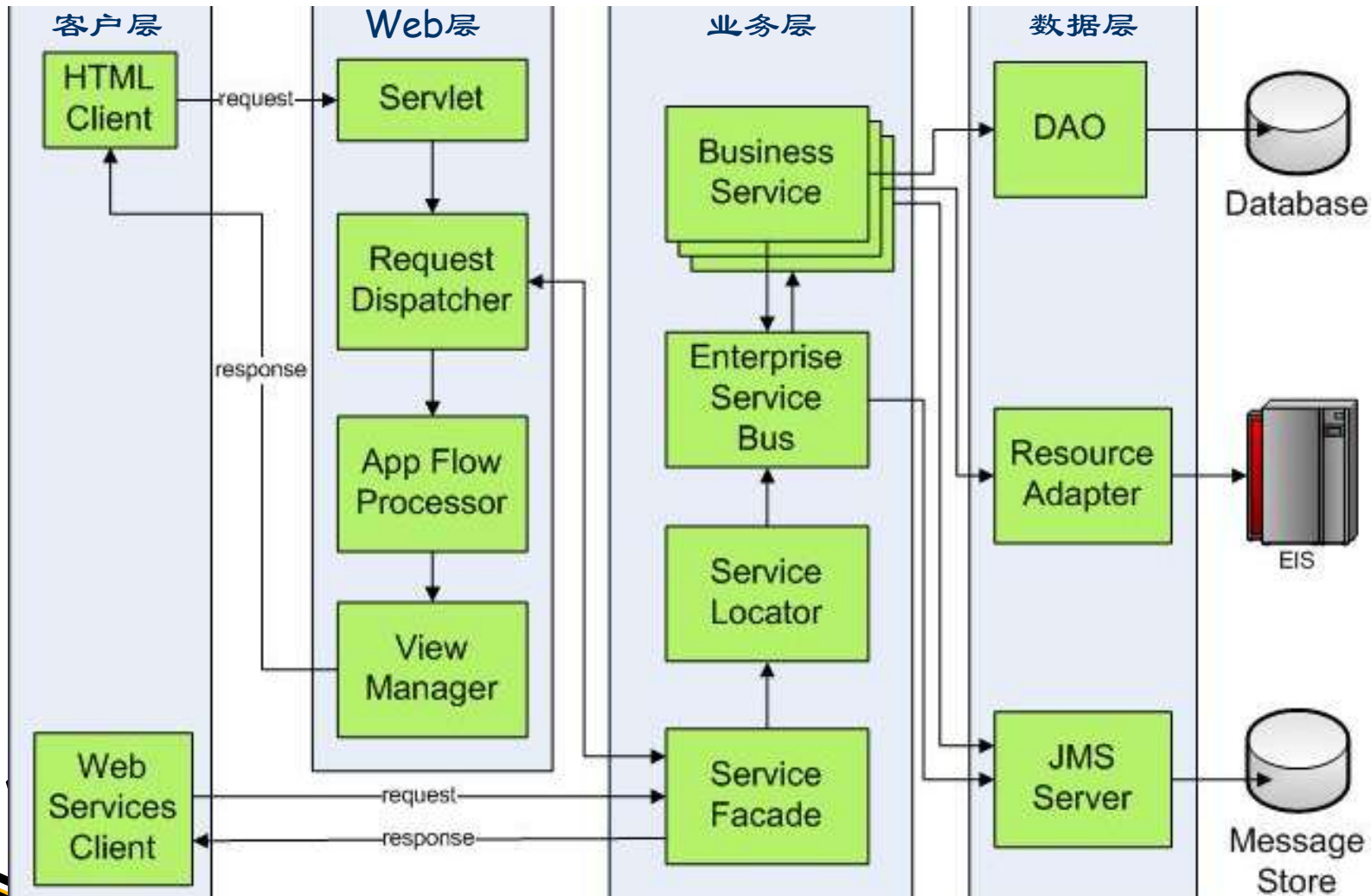
ASP.NET应用架构



Web应用的N层架构

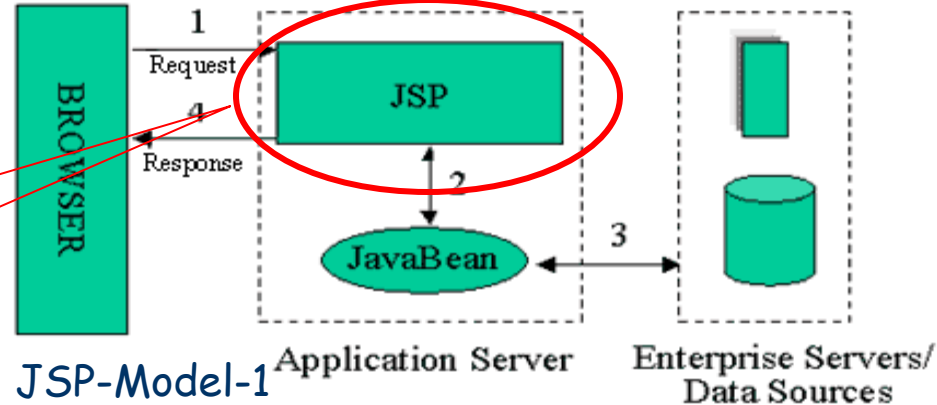


Java EE应用架构

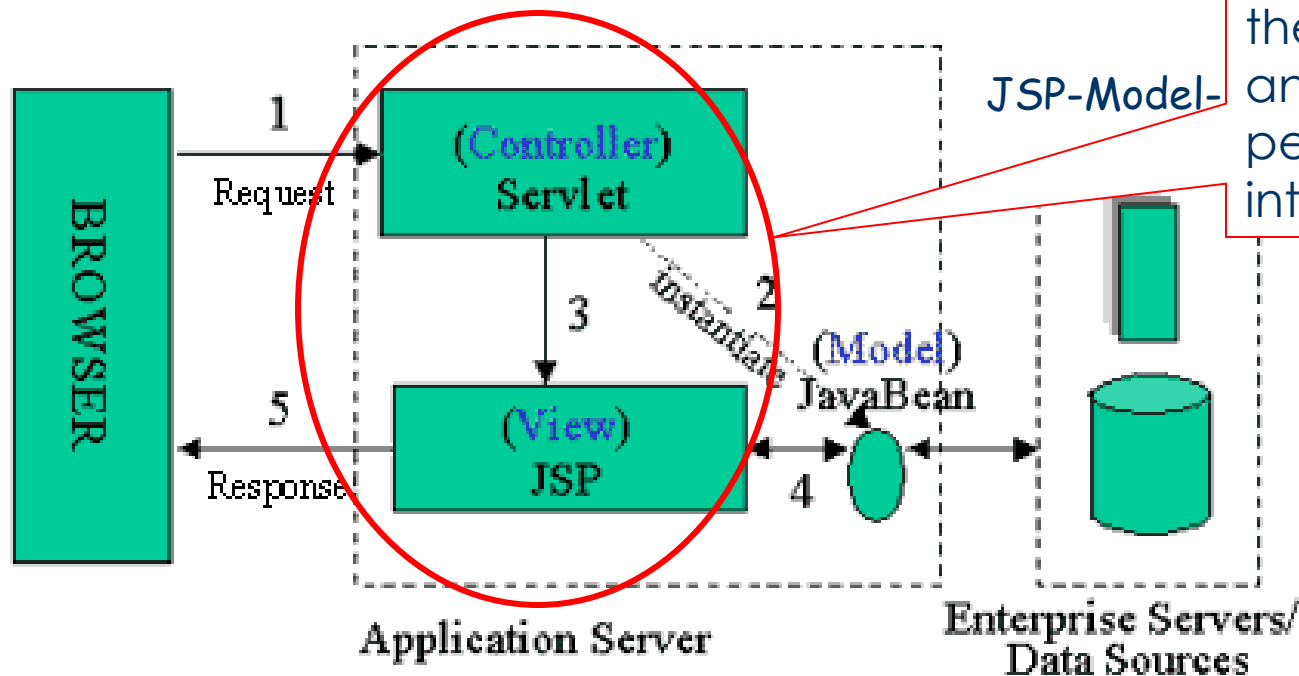


JSP-Model-2

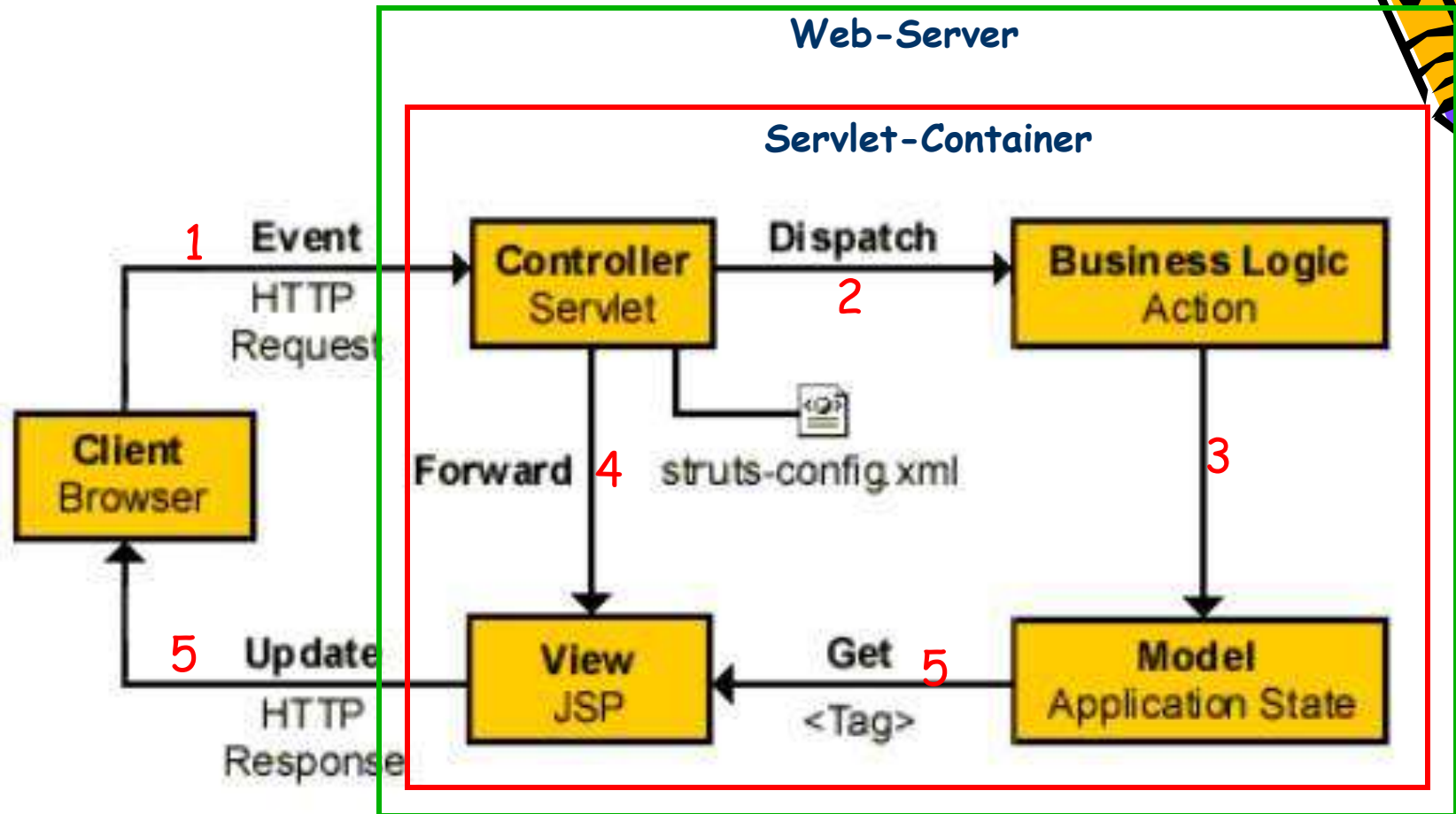
the JSP page alone is responsible for processing the incoming request and replying back to the client.



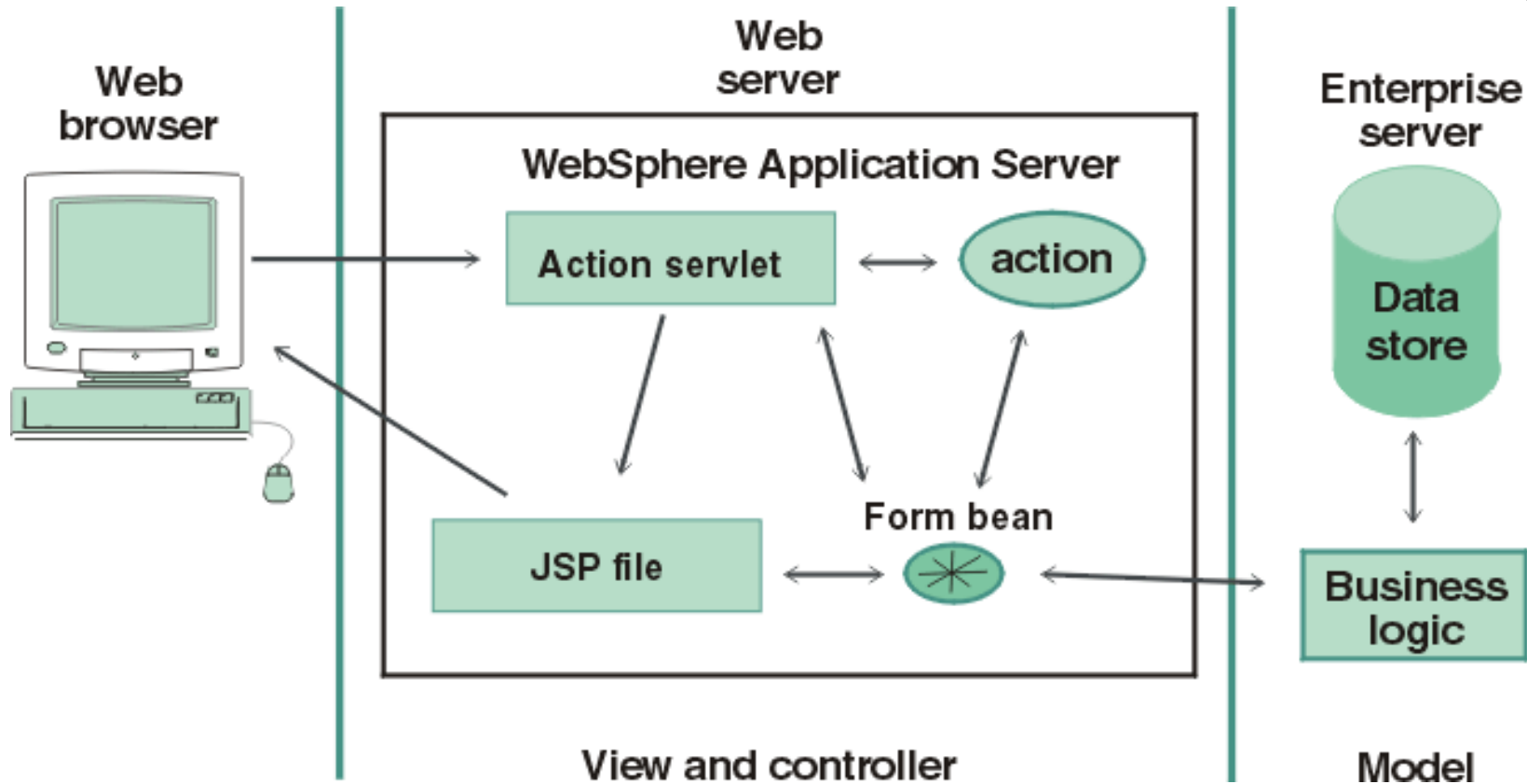
using JSP to generate the presentation layer and servlets to perform process-intensive tasks.



JSP-Model Implementation in Struts

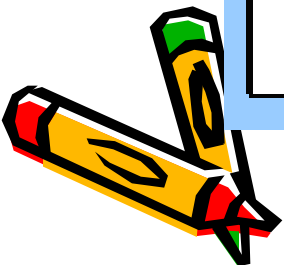
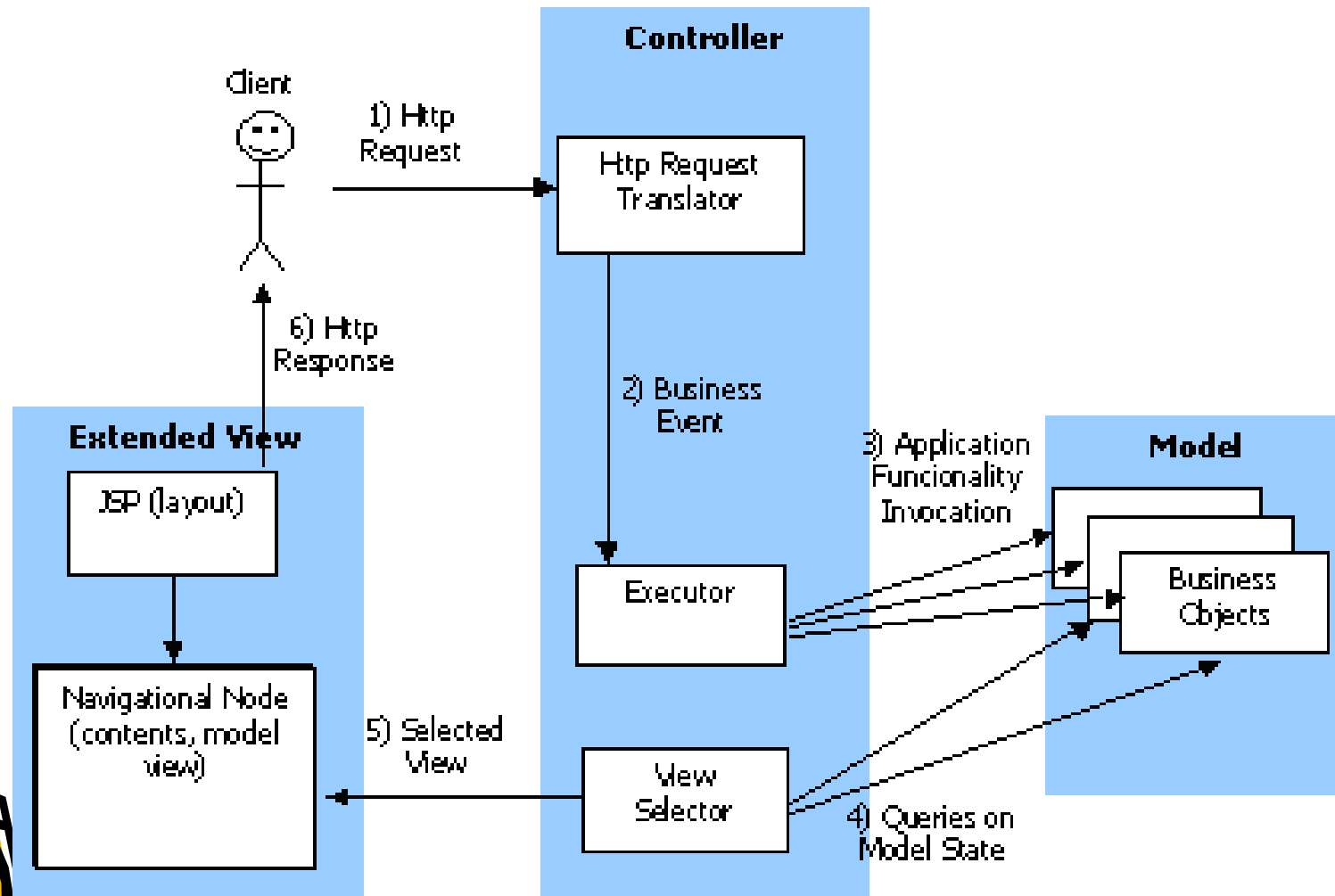


WebSphere: A example of JSP-Model Implementation in Struts



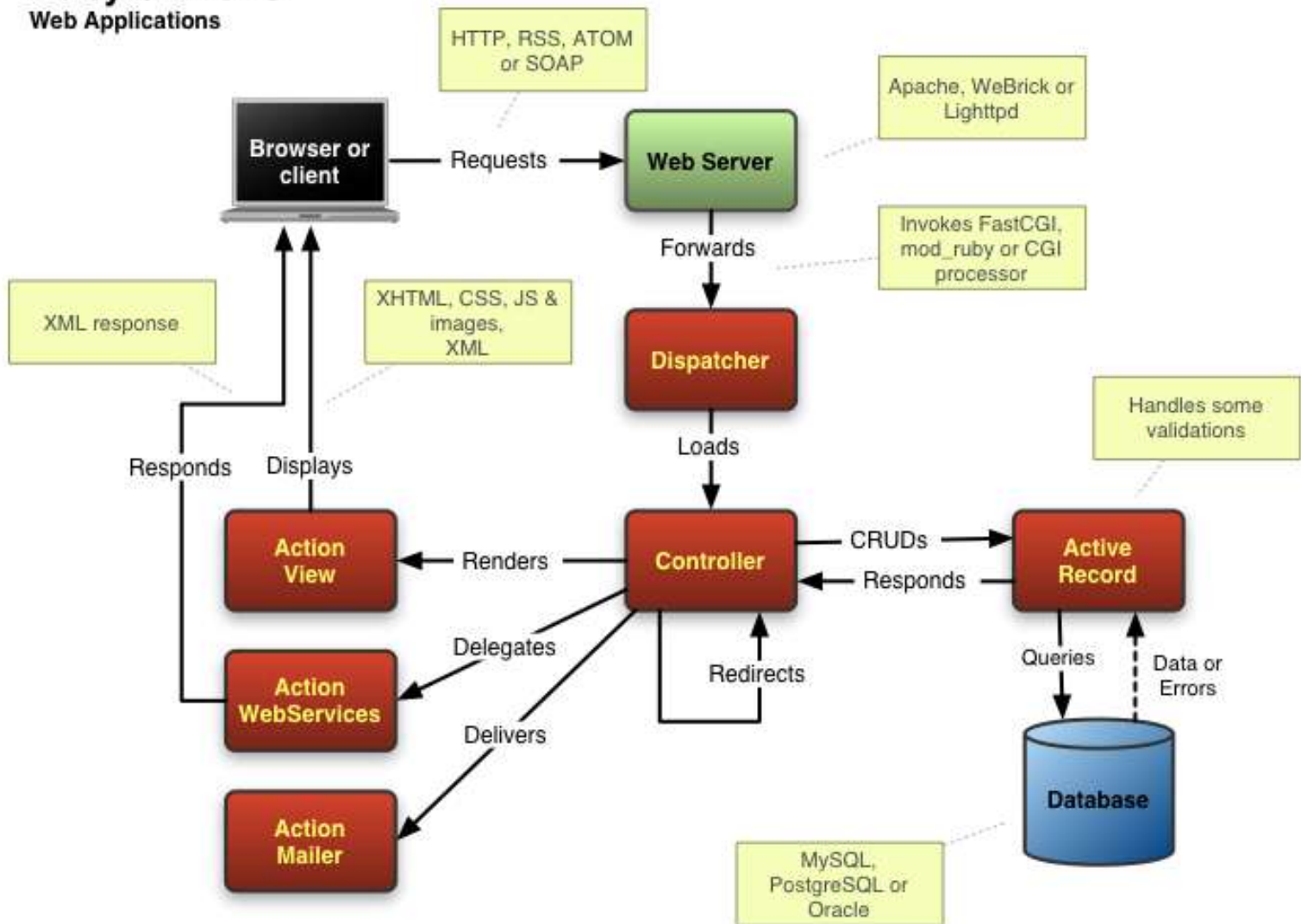


OOHDM-Java2



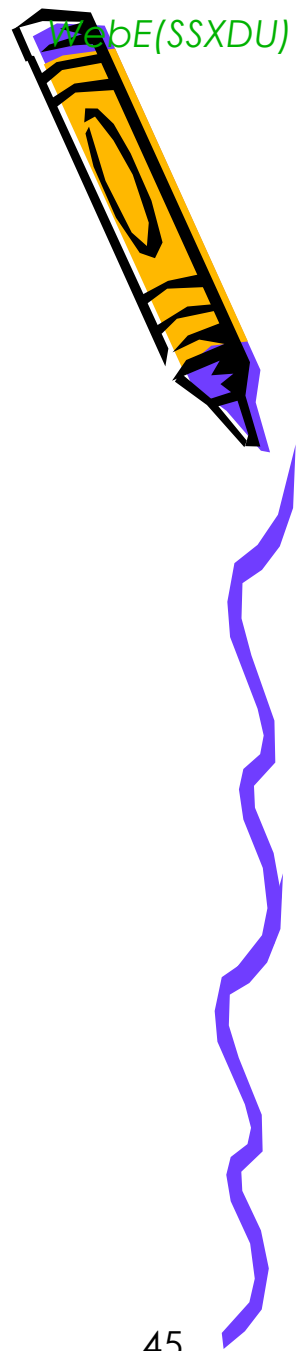
Ruby on Rails

Web Applications



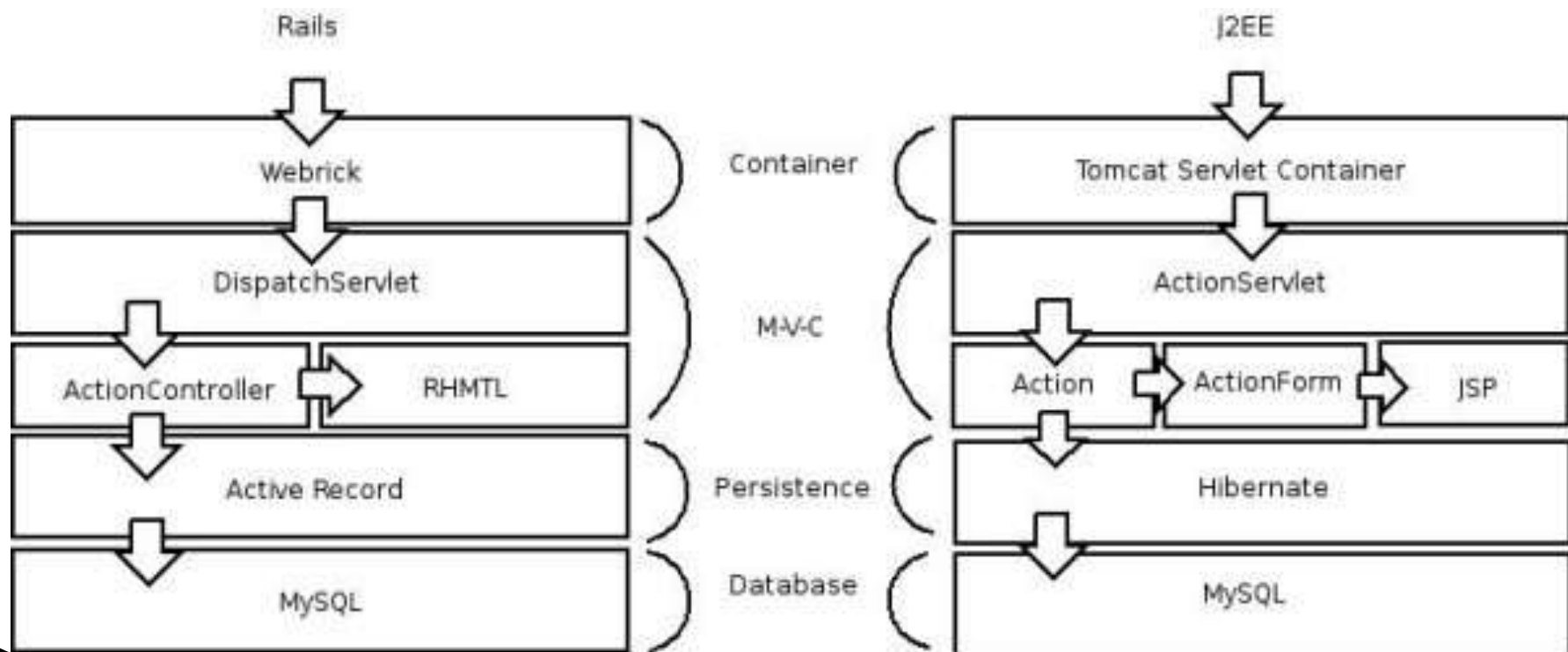
Web应用的N层架构

- 优点：重用
 - 松耦合(Loose-coupling) – 局部变化对全局影响小
 - 代码更易维护性
 - 模块可扩展性更好
- 缺点
 - 不必要的复杂性
 - 更多故障点



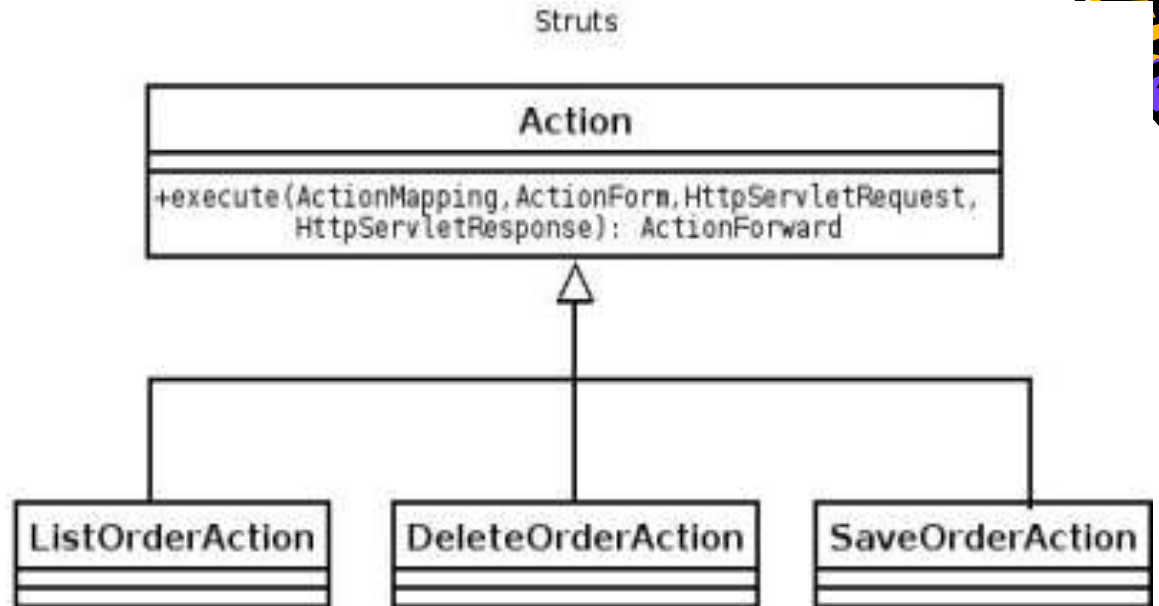
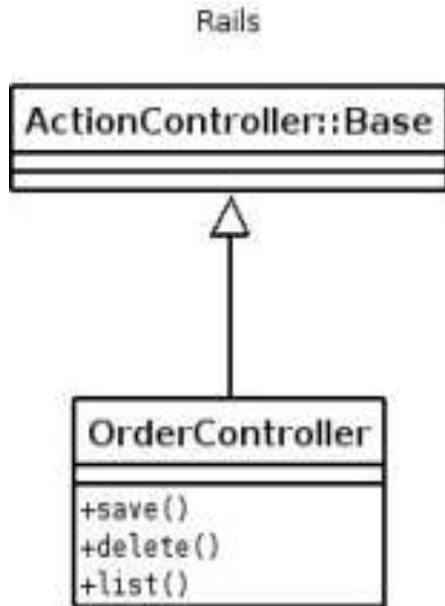
RoR(Ruby on Rails)架构

- Rails 和J2EE Web (Tomcat servlet容器, Struts Web应用框架, 以及Hibernate持久框架)

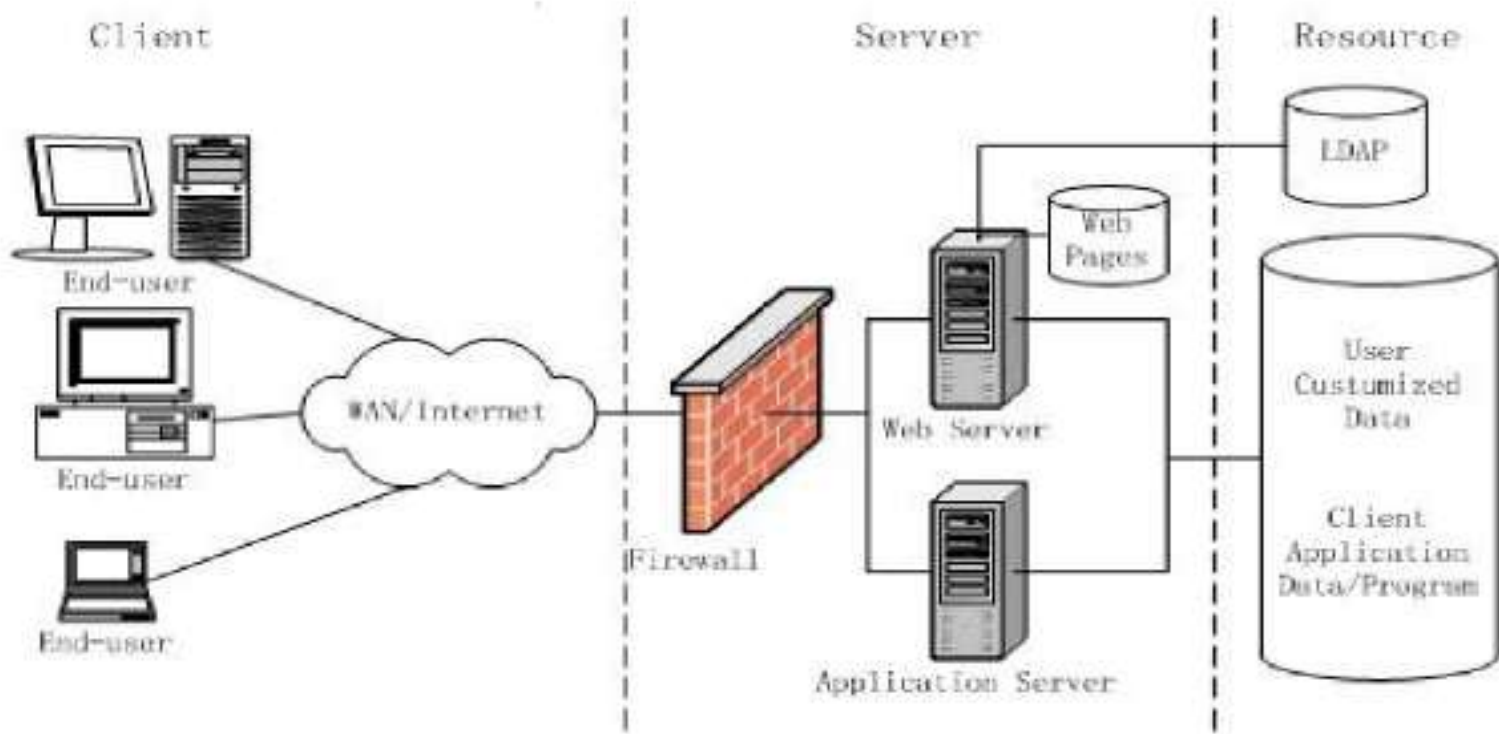


Rails和J2EE 层次对照

Rails和Struts的动作层次结构



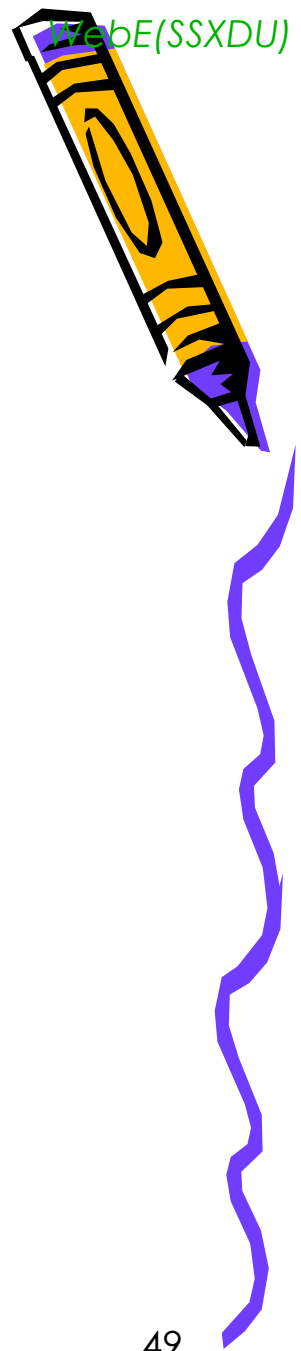
Embed Services – Example 1



Services and interconnections with respect to workflow, security and business logic.

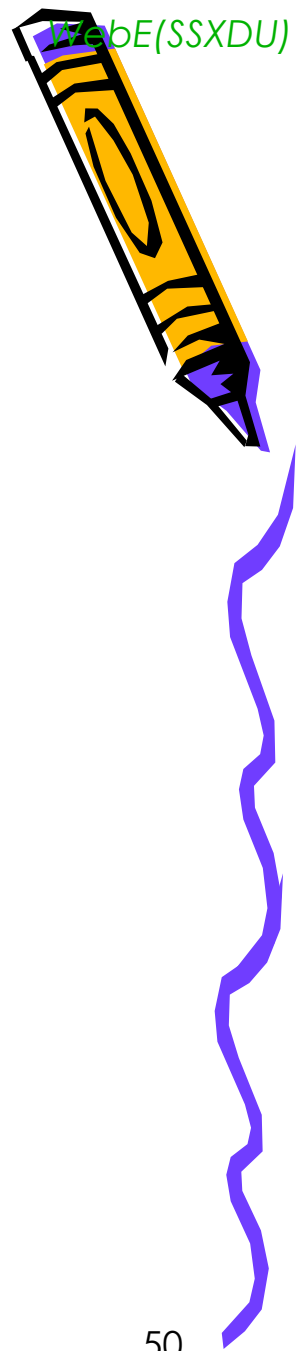
SCS System: A new architecture for Web-based Application Wu and Zhao (2003)

集成架构



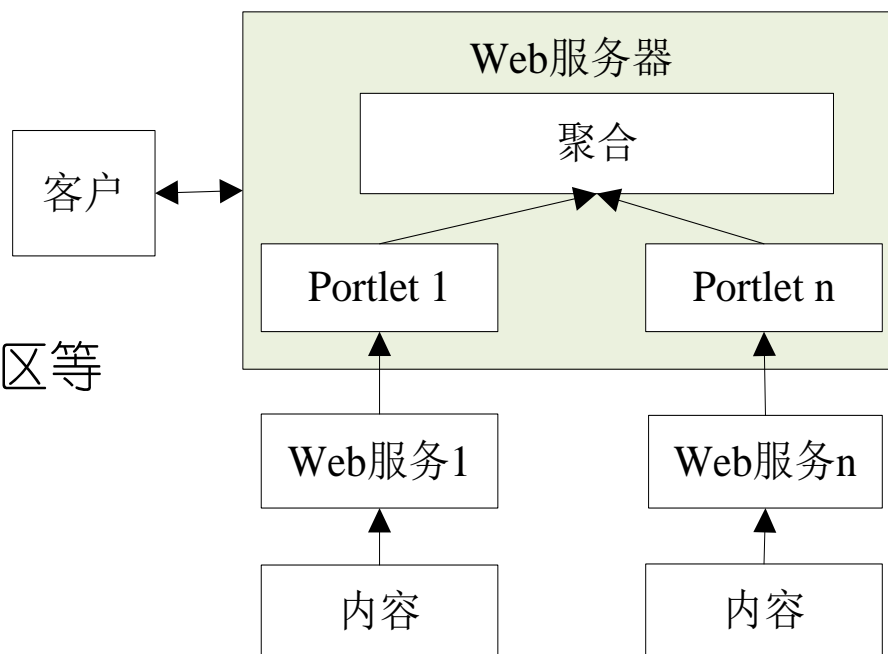
集成架构

- Web应用和外部系统或内部系统的集成
 - 展示层面
 - 应用逻辑层面
 - 内容层面



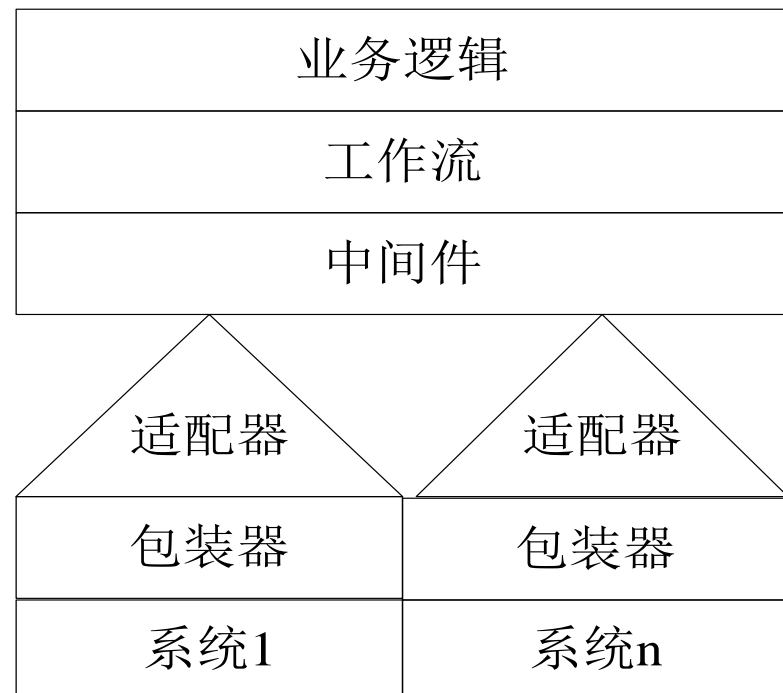
门户

- 门户 (Portal) 是指基于Web技术, 并针对具体用户或社区的应用平台。
 - 展示层面的集成, 集成多样化内容服务的Web应用
 - 水平门户
 - 新浪、腾讯等
 - 垂直门户
 - W3C、语义Web社区等



EAI (Enterprise Application Integration, 企业应用集成)

- 强调内容层面和应用逻辑层面集成的架构
- 集成不同数据源、基于各种不同平台、用不同方案建立的异构应用
- 集成遗留系统
- 采用中间件

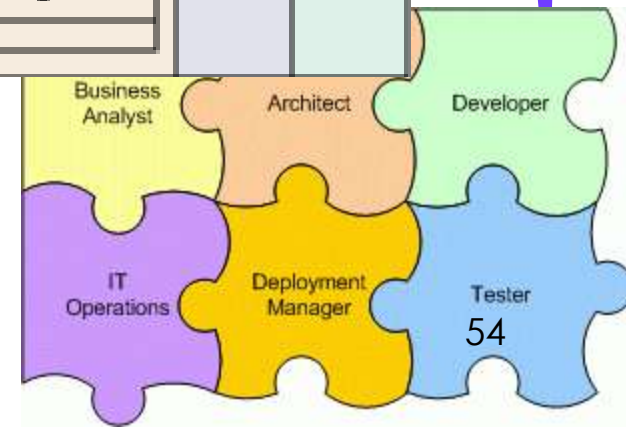
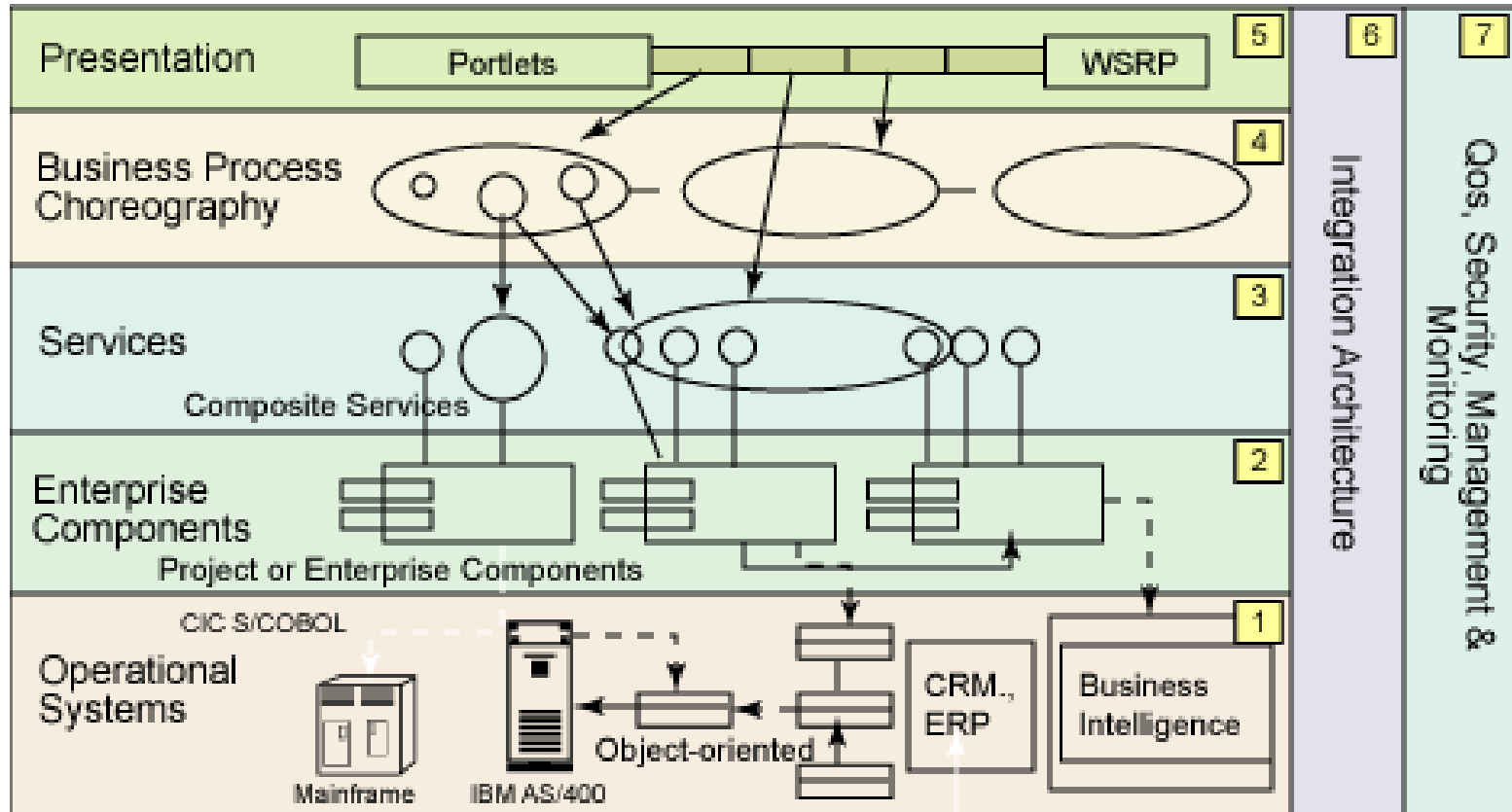


SOA (Service-Oriented Architecture, 面向服务的架构)

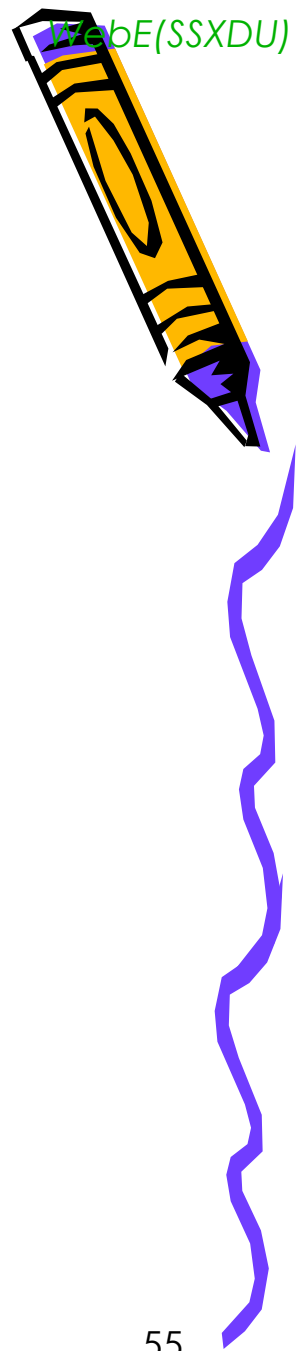
- 提供一种集成框架
- 关键是“服务”
 - 将应用程序的业务功能单元称为服务，通过这些服务之间定义好的接口和约定进行集成，形成一种架构模型，从而构成整个应用。
 - Web服务是实现SOA的方式之一。



The layers of a SOA

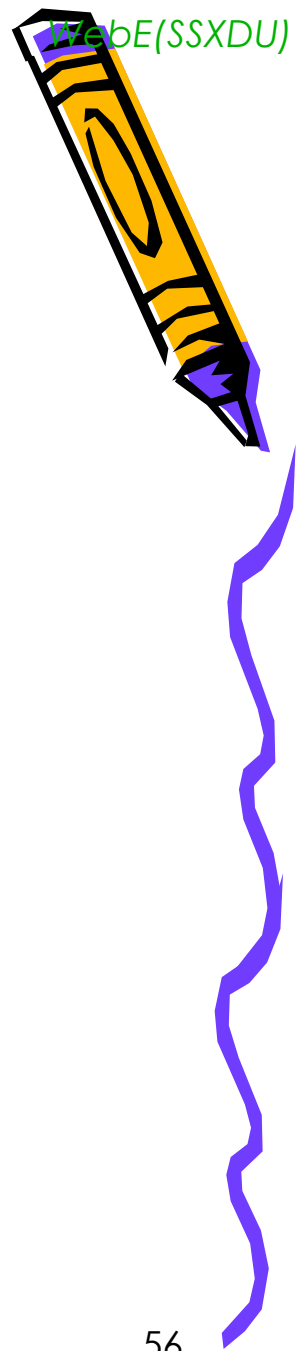


面向数据的架构



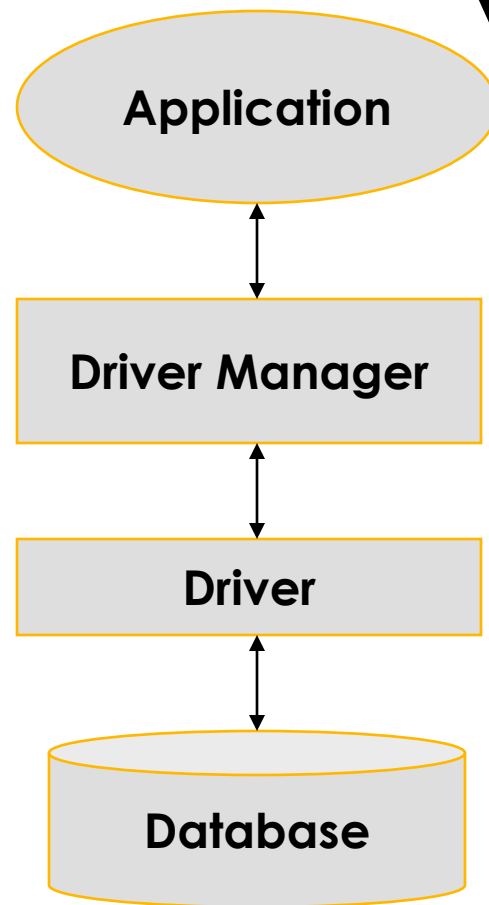
面向数据的架构

- 数据架构分类
 - 数据库中的结构化数据
 - 文档类存储于内容管理系统
 - 多媒体类数据存储与媒体服务器
- Web应用结合使用

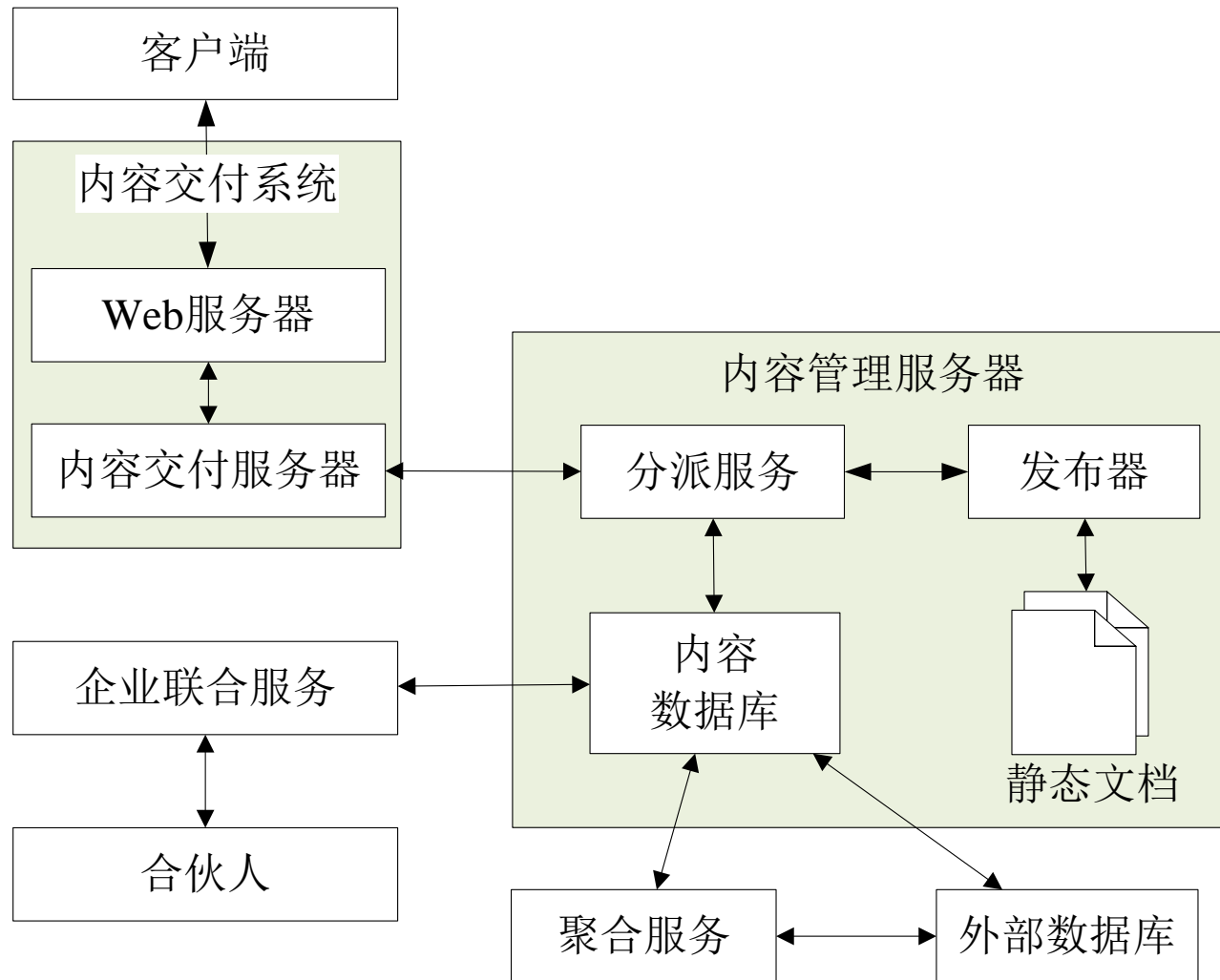


以数据库为中心的架构

- 以数据为中心的架构
 - 结构化数据(JDBC/ODBC)
 - 通过Web扩展或应用服务器进行访问
 - 很成熟
 - 易于实现



Web文档管理架构

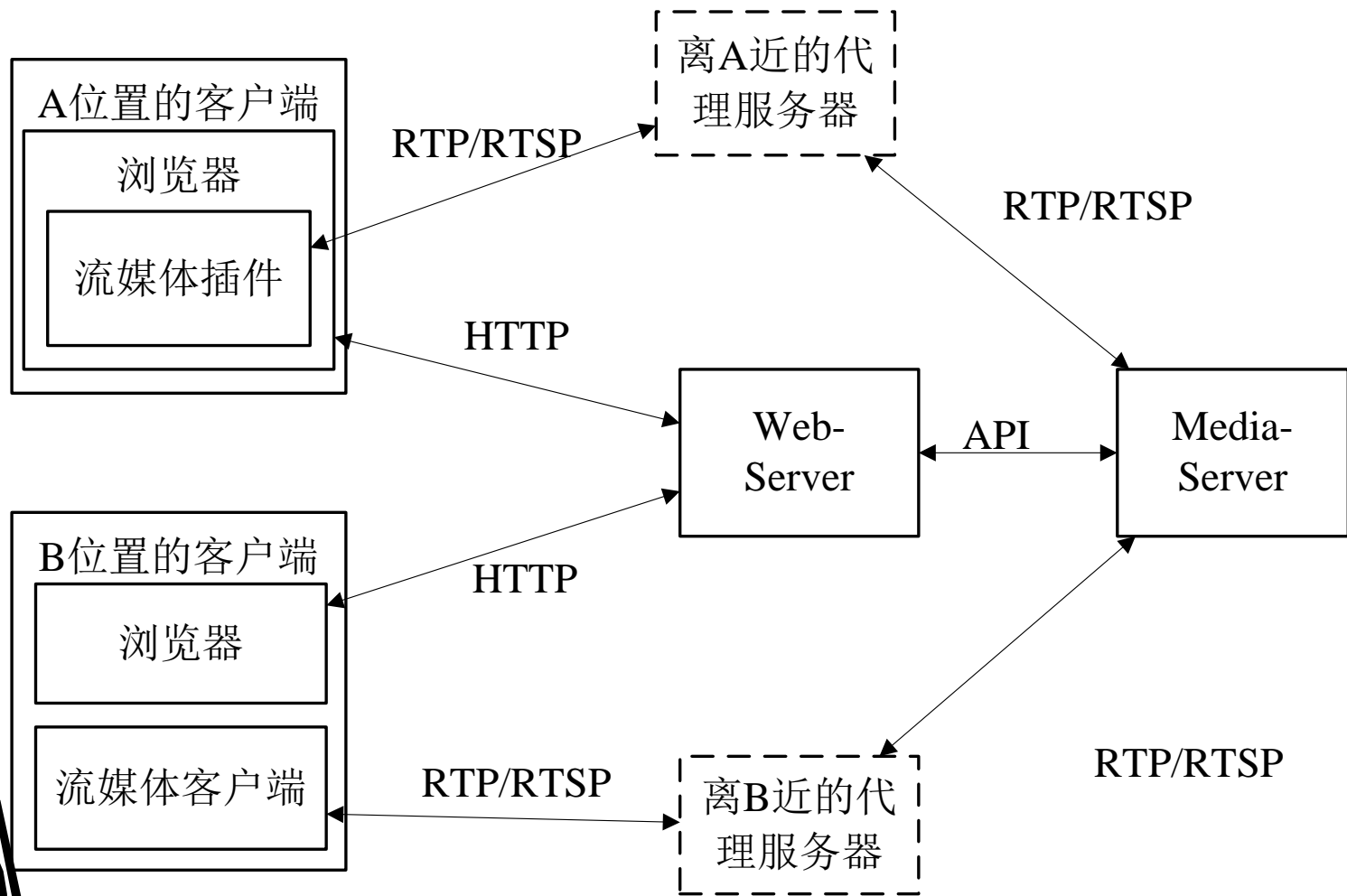


流媒体数据的架构

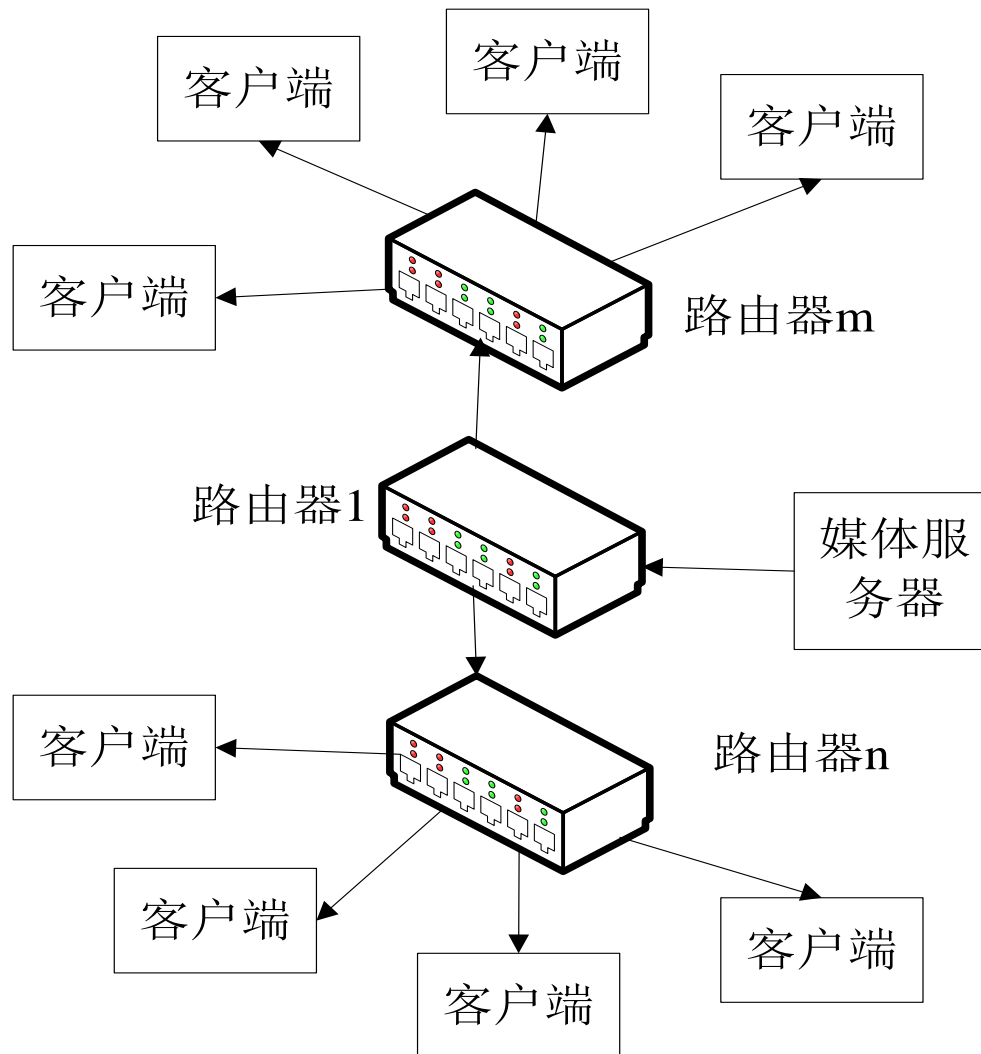
- 多媒体数据包括视频、音频等，通过标准的因特网协议（如HTTP、FTP等）进行传输--下载速度慢
- 流技术最小化多媒体内容播放的等待时间
 - RTP (Real Time Protocol, 实时协议)
 - RTSP (Real Time Streaming Protocol, 实时流协议)
 -
- 两种应用领域
 - 按需播放已存在的媒体，如：点播
 - 直播，如：Web casting



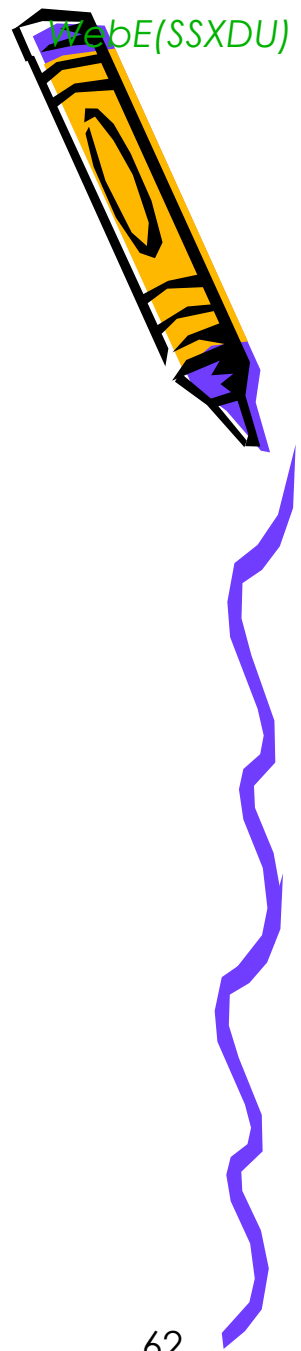
点对点连接的流媒体架构



使用广播的流媒体架构



总结与展望



总结

- 好的Web应用架构可以提高Web应用的开发效率、提高Web应用的可重用性，易于维护和扩展。
 - 模型驱动架构
 - 架构模式
 - 层次架构
 - 面向数据的架构
 - 集成架构



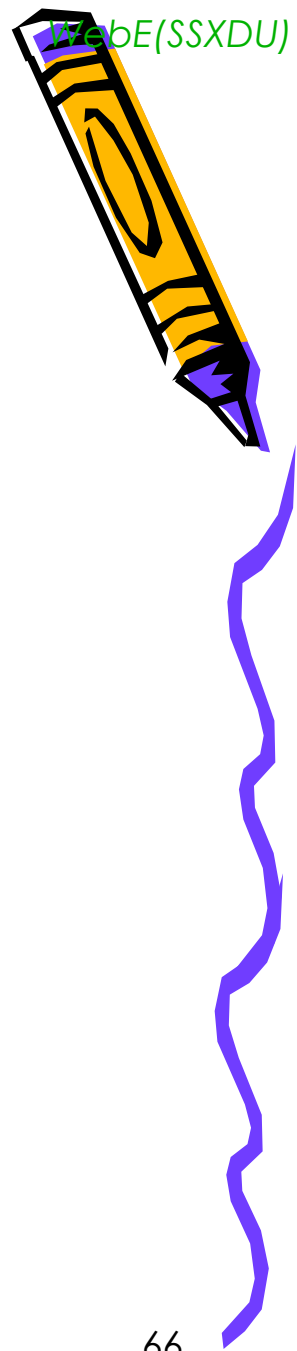
展望

- 基础设施
 - P2P→普适Web应用, NoSQL
 - 网格技术→增大Web应用的计算能力
- 普适计算和面向门户
 - 上下文
 - 集成淡化客户端和服务端之间的界线
- 数字电视和在线游戏



Project Task: Task5

- Web应用架构设计
 - 结合本章内容，完成Web应用架构
- 展示的主要内容之一
 - 10 minutes
 - ~2 slides
- 架构报告



参考资料

- Chapter 5
- Suggested
 - Jablonski, S., Petrov, I., Meiler, C., Mayer, U., Guide to Web Application and Platform Architectures, Springer-Verlag, 2004.
 - Mary Shaw, David Garlan: Software Architecture: Perspectives on an Emerging Discipline. Prentice Hall, April 1996.
 - Douglas C. Schmidt, Michael Stal, Hans Rohnert, Frank Buschmann: Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects, John Wiley & Sons, 2000.
 - Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: Design Patterns. Addison-Wesley, 1994.
 - Martin Fowler: Analysis Patterns : Reusable Object Models. Addison-Wesley. 1996.
 - Different definitions of software architecture:
<http://www.sei.cmu.edu/architecture/definitions.html>