



社会保险核心平台三版培训

技术培训一系统架构概述



北京利博赛社保信息技术有限公司

培训简介

❖ 培训内容

- 核三系统架构相关技术知识

❖ 培训目标

- 了解核三的系统架构
- 能够对核三系统架构进行定制扩展

❖ 适用对象

- 本地化项目的系统架构师、技术人员

❖ 学员要求

- 熟悉J2EE的相关技术与架构
- 熟悉Struts、Hibernate等技术
- 理解AOP等概念

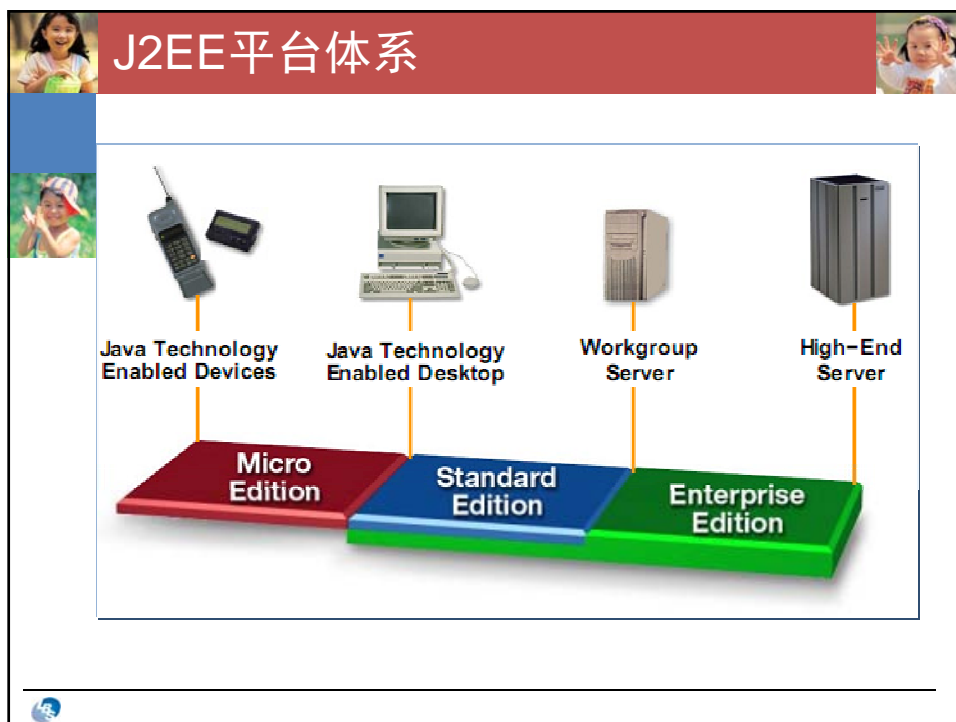



目录	
1	J2EE体系结构
2	核三系统架构概述
3	架构对技术问题的解决
4	架构与模式
5	环境介绍

J2EE体系结构


J2EE体系结构


- ❖ J2EE概述
- ❖ J2EE平台的优势
- ❖ J2EE标准
- ❖ J2EE应用程序方案






J2EE平台体系






- ❖ **Java平台体系**
 - J2SE、JavaSE
 - J2EE、JavaEE
 - J2ME、JavaME




- ❖ **J2SE**


▪ J2SE1. 4. 2	2003-06-26
▪ J2SE5. 0	2004-10
▪ JavaSE6. 0	2006年
- ❖ **J2EE**


▪ J2EE1. 3	2001年9月24日
▪ J2EE1. 4	2003年
▪ JavaEE5	2006年4月






什么是J2EE






- ❖ **企业版**
 - 标准、开放的基础平台
 - 用于开发、部署、管理企业应用
 - 多层结构
 - 可使用Web
 - 以服务器为中心



- ❖ **Open and standard based platform for developing, deploying and managing n-tier, Web-enabled, server-centric enterprise applications**




J2EE概念

- ❖ J2EE
 - 以J2SE为基础平台
 - 一套规范
 - 加上企业软件提供者的产品
- ❖ J2EE的交付内容
 - J2EE规范 (Platform Specification)
 - 参考实现 (Reference Implementation)
 - 兼容性测试套件 (Compatibility Test Suite)
 - J2EE蓝图 (J2EE BluePrints)


J2EE平台服务


- ❖ J2EE服务组件用于支持EJB、Servlet、JSP、WebService组件

The diagram illustrates the J2EE platform architecture. It shows a central 'Container' (yellow) which supports 'Enterprise Beans', 'Web Services', 'Servlets', and 'JSP Pages'. To the left of the container are 'JavaBeans™' and 'Applets'. To the right are 'Connectors'. Above the container are 'Tools' (AVK, IDE), 'BluePrints', and 'Tutorial'. Below the container are 'Management', 'Deployment', 'CORBA', 'Security', 'Database', 'Directory', and 'XML'. The entire platform is based on the 'Java™ 2 SDK, Standard Edition'.




J2EE体系结构






- ❖ J2EE概述




- ❖ J2EE平台的优势


- ❖ J2EE标准


- ❖ J2EE应用程序方案






企业计算需求





Challenges Portability Diverse Environments Time-to-market Core Competence Assembly	Key Technologies J2SE™ J2EE™ JMS Servlet JSP Connector XML Data Binding	Products App Servers Web Servers Components Databases Object to DB tools
		Legacy Systems Databases TP Monitors EIS Systems



J2EE平台解决方案

❖ 针对企业解决方案的平台

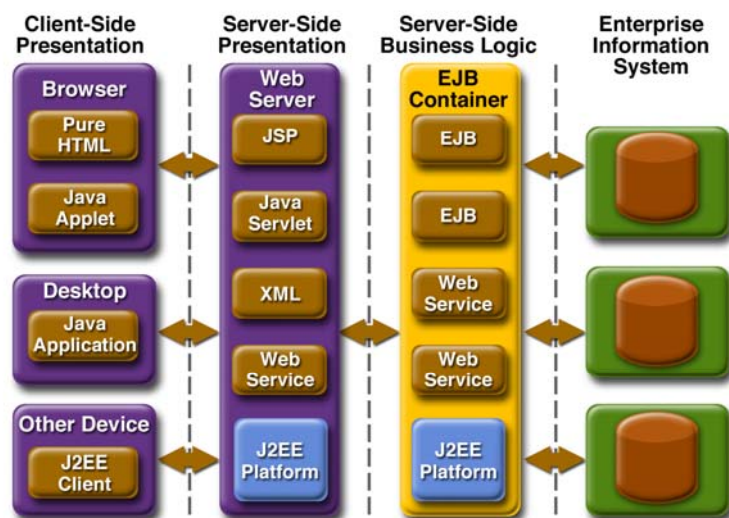
- J2SE: 可移植 (Write Once, Run Anywhere) 、JDBC、CORBA、案例模型
- J2EE支持的企业组件: EJB、Servlet、JSP、XML、WebService


❖ 更容易的中间件

- 将复杂的工作交给中间件
 - 快速开发与部署
 - 可移植性与伸缩性
 - 集成遗留资产的困难性
- 统一的J2EE标准、统一的基于组件的应用模型
- 简单性、可移植性、可伸缩性、已有资产集成


❖ 工业标准


J2EE蓝图：企业应用模型与最佳实践






J2EE的优势





- ❖ **容器与连接器：隐藏复杂性，增强可移植性**
 - 组件是应用开发的核心
 - 容器和连接器由中间件提供商实现
- ❖ **灵活的用户交互**
 - 桌面、笔记本、PDA、手机、其它设备
 - 独立客户端、HTML、Java applets
 - 服务器端：Servlet、JSP
- ❖ **EJB组件模型**
- ❖ **与WebService互操作性**
- ❖ **加速开发与部署**





J2EE体系结构





- ❖ **J2EE概述**
- ❖ **J2EE平台的优势**
- ❖ **J2EE标准**
- ❖ **J2EE应用程序方案**





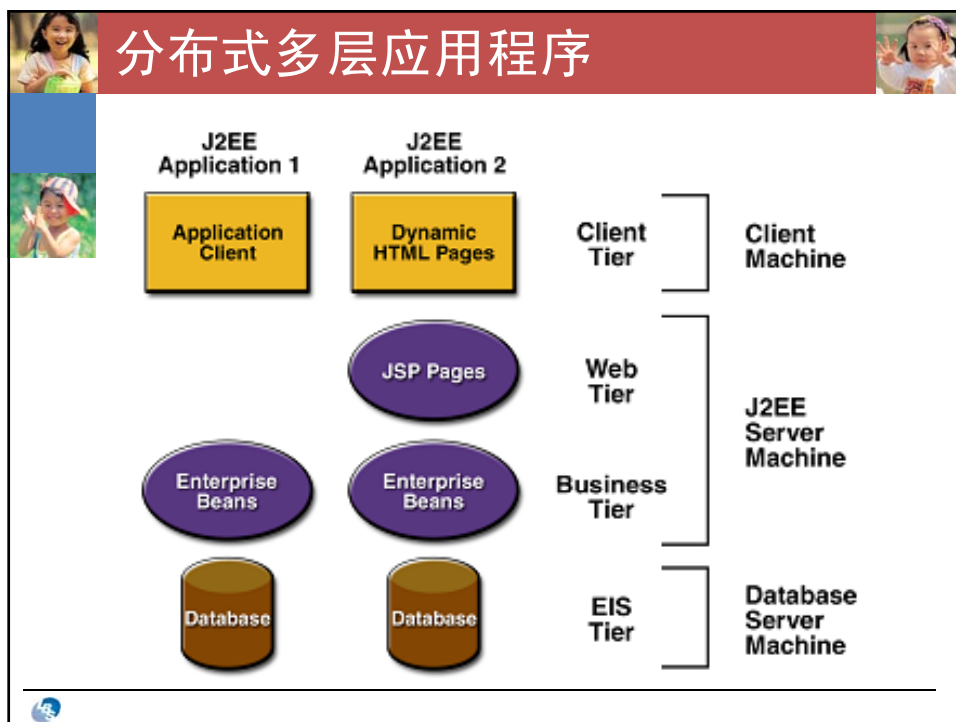
J2EE标准







- ❖ 分布式多层应用程序
- ❖ J2EE容器
- ❖ Web Service支持
- ❖ 应用程序打包
- ❖ 应用开发角色分工
- ❖ J2EE API
- ❖ 应用服务器











分布式多层应用程序







- ❖ **J2EE组件**
 - 客户端组件：客户端应用程序、applet
 - Web组件：Servlet、JSP
 - 业务组件：EJB
- ❖ **J2EE客户端**
 - Web客户端（瘦客户端）
 - 动态Web页面
 - 浏览器
 - Applet
 - 客户端应用程序
 - GUI（Swing、AWT）、Command line
 - 直接EJB访问、Http







分布式多层应用程序





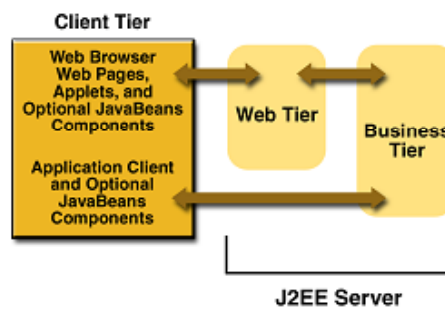
- ❖ **J2EE组件**
- ❖ **J2EE客户端**
- ❖ **Web组件**
- ❖ **业务组件**
- ❖ **EIS（Enterprise Information System）数据库层**



分布式多层应用程序

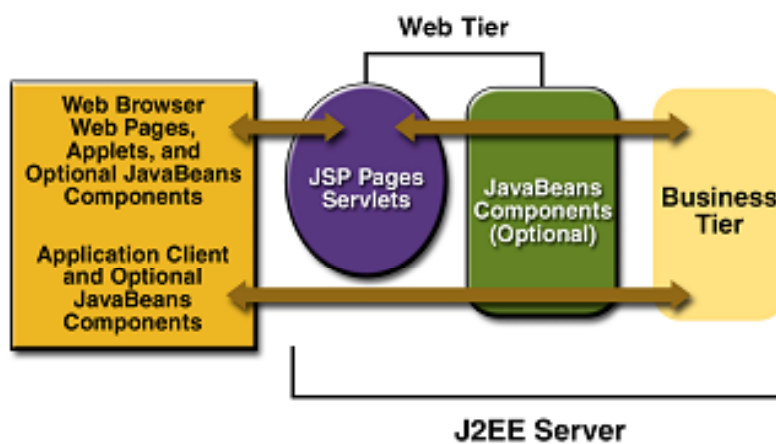
❖ J2EE客户端

- JavaBean组件体系结构
 - J2EE规范之外的组件技术
 - 客户端与服务器端都可以使用
- 与J2EE Server通讯



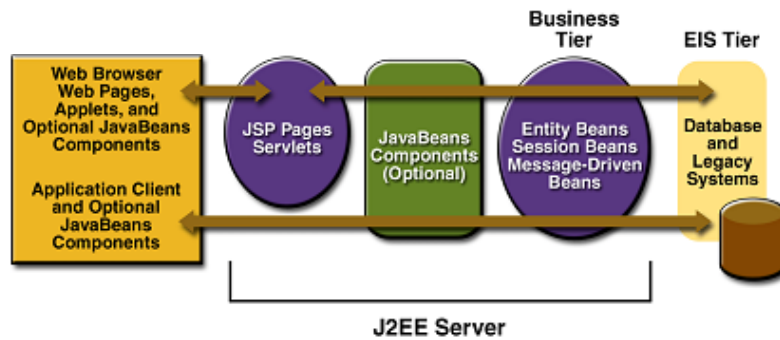
分布式多层应用程序

❖ Web组件



分布式多层应用程序

❖ 业务组件



❖ EIS (Enterprise Information System) 数据库层

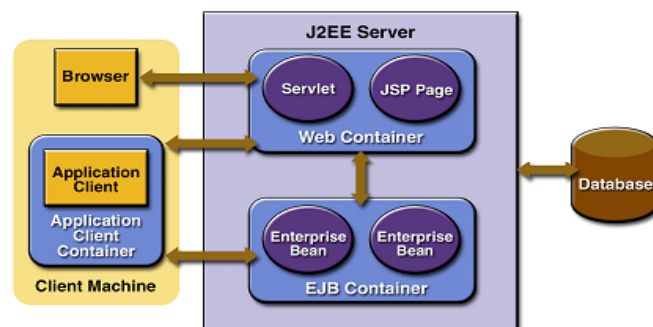
J2EE 容器


❖ 容器提供的服务

- 安全、JTA、JNDI、远程调用


❖ 容器的类型


- J2EE Server (WebContainer、EJBContainer)、Application client container、Applet container







Web Service支持







- ❖ XML
- ❖ SOAP Transport Protocol
- ❖ WSDL Standard Format
- ❖ UDDI and ebXML Standard Formats









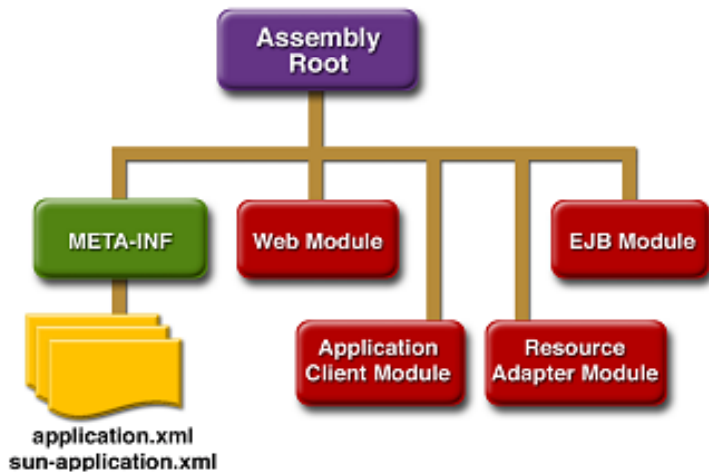
应用程序打包






- ❖ EAR文件打包结构







```
graph TD; AR[Assembly Root] --- META-INF; AR --- WM[Web Module]; AR --- ACM[Application Client Module]; AR --- RAM[Resource Adapter Module]; AR --- EJB[EJB Module]; META-INF --- app_files[application.xml<br/>sun-application.xml]
```


The diagram illustrates the structure of an EAR (Enterprise Archive) file. At the top is the 'Assembly Root' (purple box). It branches into five components: 'META-INF' (green box), 'Web Module' (red box), 'Application Client Module' (red box), 'Resource Adapter Module' (red box), and 'EJB Module' (red box). The 'META-INF' component is further detailed with a stack of yellow folders representing files: 'application.xml' and 'sun-application.xml'.








应用开发角色分工







- ❖ J2EE产品提供者
- ❖ 工具提供者
- ❖ 应用组件提供者
 - EJB组件开发者
 - Web组件开发者
 - 客户端应用开发者
- ❖ 应用装配者
 - 将Jar、War打包为EAR、定制部署描述符、验证EAR格式正确且遵从J2EE规范
- ❖ 应用部署与管理人员








J2EE1.4 API







- ❖ Enterprise JavaBeans Technology
- ❖ Java Servlet Technology
- ❖ JavaServer Pages Technology
- ❖ Java Message Service API
- ❖ Java Transaction API
- ❖ JavaMail API
- ❖ JavaBeans Activation Framework
- ❖ Java API for XML Processing
- ❖ Java API for XML-Based RPC
- ❖ SOAP with Attachments API for Java
- ❖ Java API for XML Registries
- ❖ J2EE Connector Architecture







J2EE1.4 API







- ❖ **JDBC API**
- ❖ **Java Naming and Directory Interface**
- ❖ **Java Authentication and Authorization Service**
- ❖ **Simplified Systems Integration**
 - 统一的跨层EJB应用模型
 - 简化“请求-响应”机制： JSP与Servlet
 - 可靠的安全模型： JAAS
 - 基于XML的数据交换集成： JAXP, SAAJ, and JAX-RPC
 - 简化互操作性： J2EE Connector
 - 简单的数据库连通性： JDBC
 - 企业应用集成： MDB、JMS、JTA、JNDI






J2EE1.2 Platform APIs

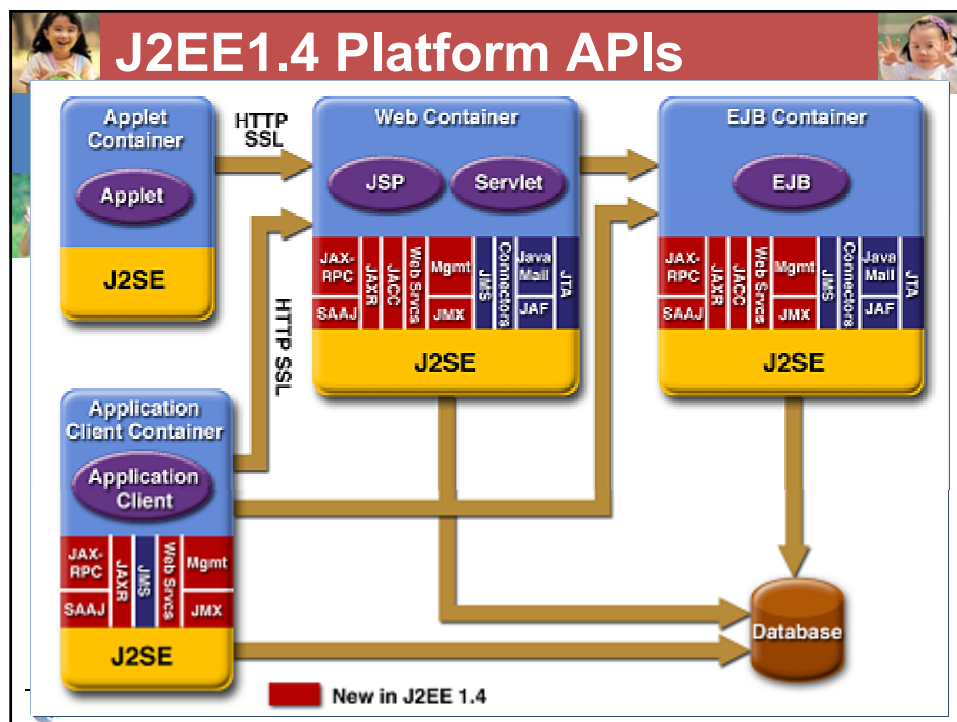


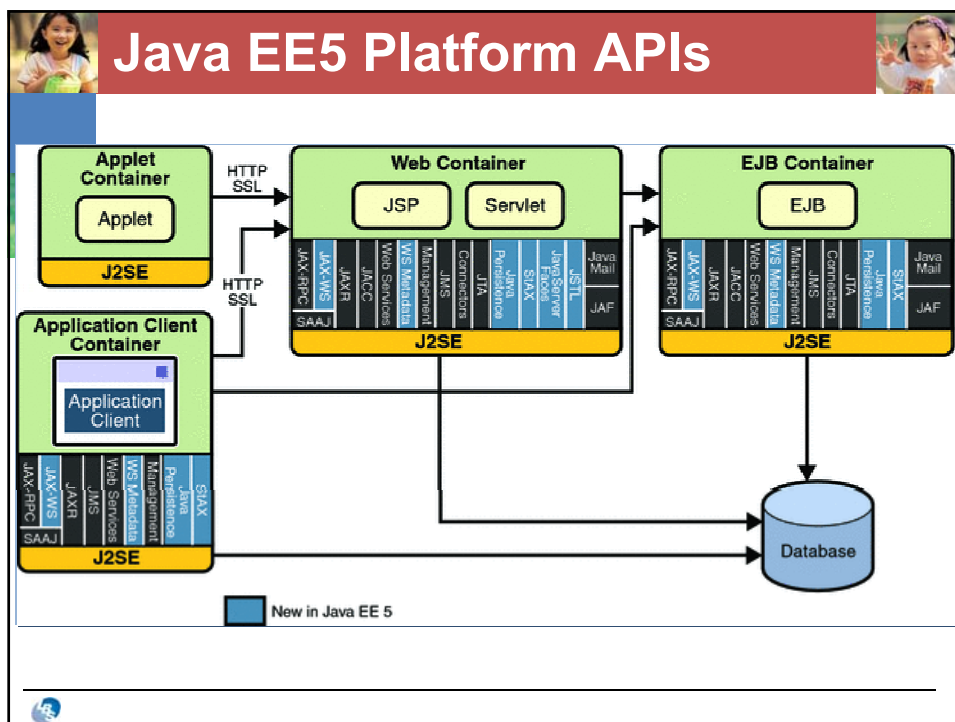


	Version
Java 2 SDK, Standard Edition	1.2
RMI/ IIOP	1.0
JDBC™	2.0
Java Messaging Service	1.0
JNDI	1.2
Servlet	2.2
JavaServer Pages™	1.1
JavaMail	1.1
JavaBeans™ Activation Framework	1.0
Enterprise JavaBeans	1.1
Java Transaction API	1.0




J2EE1.3 Platform APIs	
Enterprise JavaBeans Technology	2.0
JDBC API	2.0
Java Servlet Technology	2.3
JavaServer Pages Technology	1.2
Java Message Service	1.0
Java Naming and Directory Interface	1.2
Java Transaction API	1.0
JavaMail API	1.2
JavaBeans Activation Framework	1.0
Java API for XML Processing	1.1
J2EE Connector Architecture	1.0
Java Authentication and Authorization Service	1.0







应用服务器

- ❖ 仅包含**Web容器**的应用服务器
 - Tomcat
- ❖ 包含**Web容器**和**EJB容器**的应用服务器
 - Weblogic
 - WebSphere
- ❖ 应用服务器的不同版本对应支持**J2SE**和**J2EE**的不同版本
 - J2SE1.4+J2EE1.4
 - J2SE1.5+J2EE1.4




J2EE体系结构






- ❖ J2EE概述




- ❖ J2EE平台的优势


- ❖ J2EE标准


- ❖ J2EE应用程序方案






J2EE应用程序方案





- ❖ J2EE应用程序解决方案




- ❖ 多层应用程序模型

- ❖ B/S/S多层应用程序方案

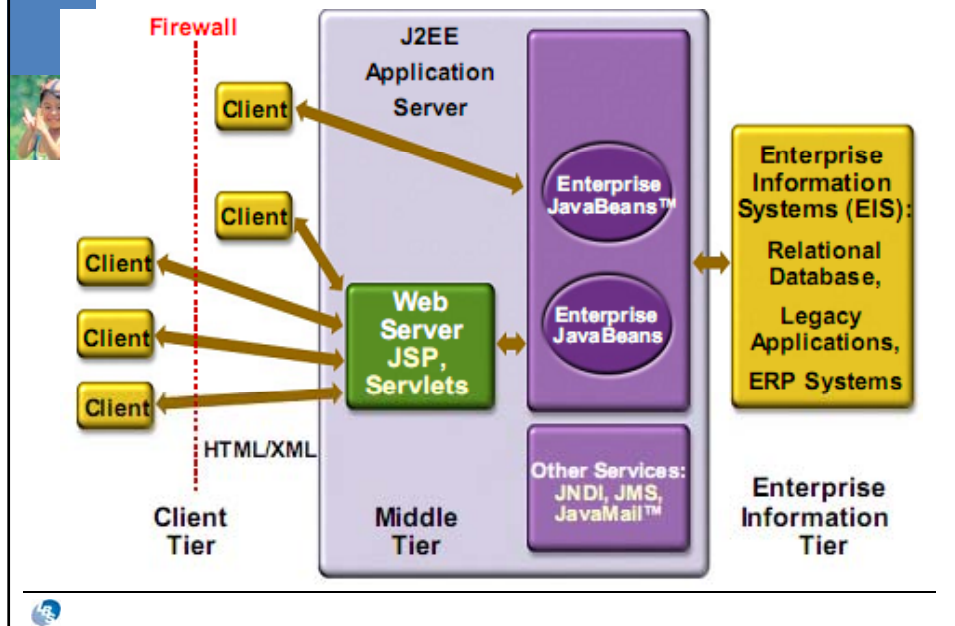
- ❖ 独立客户方案

- ❖ 以Web为中心的应用程序方案

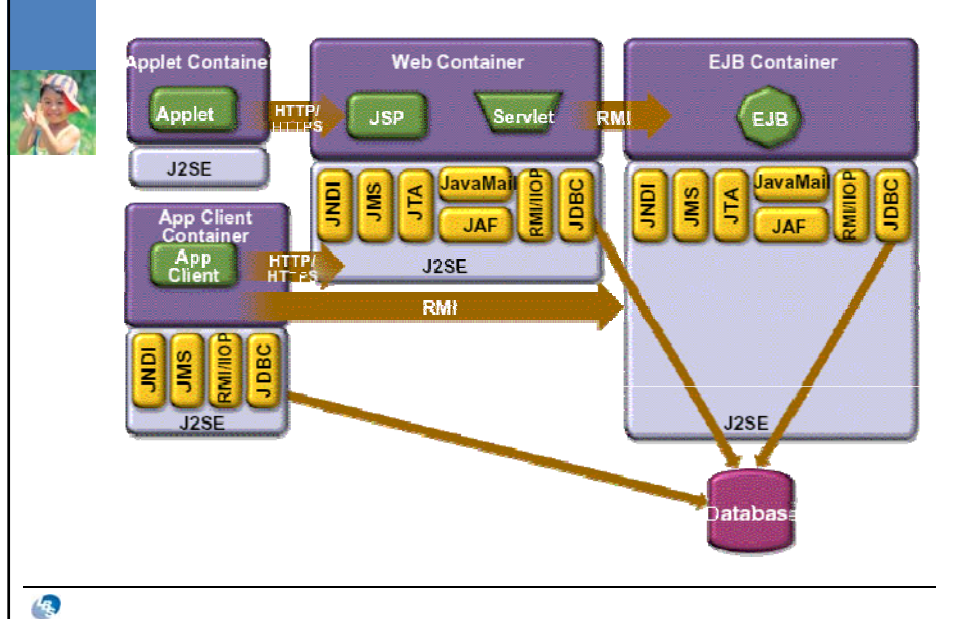
- ❖ B2B方案



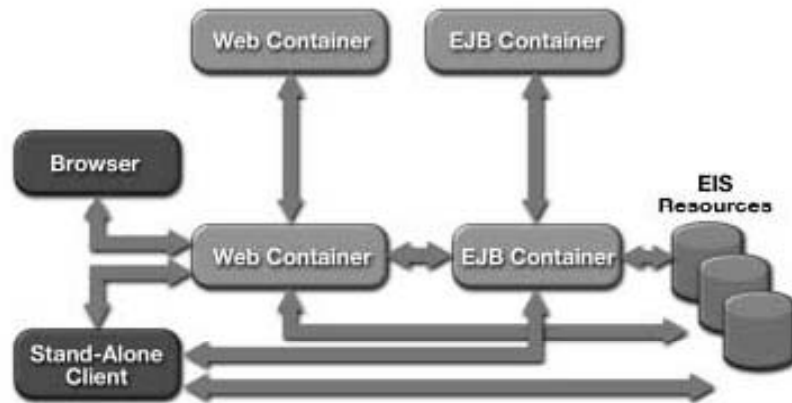
J2EE应用程序解决方案



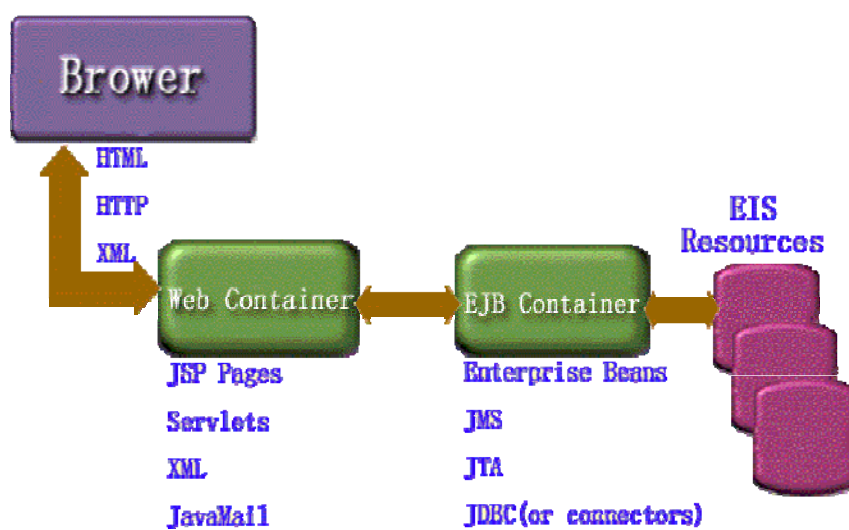
J2EE应用程序解决方案

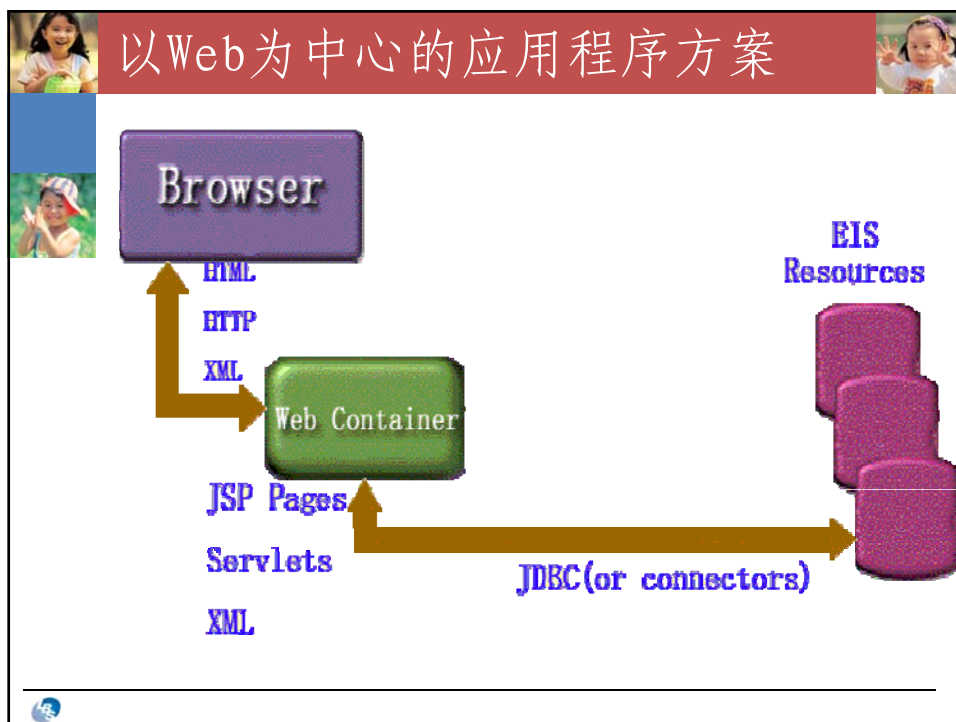
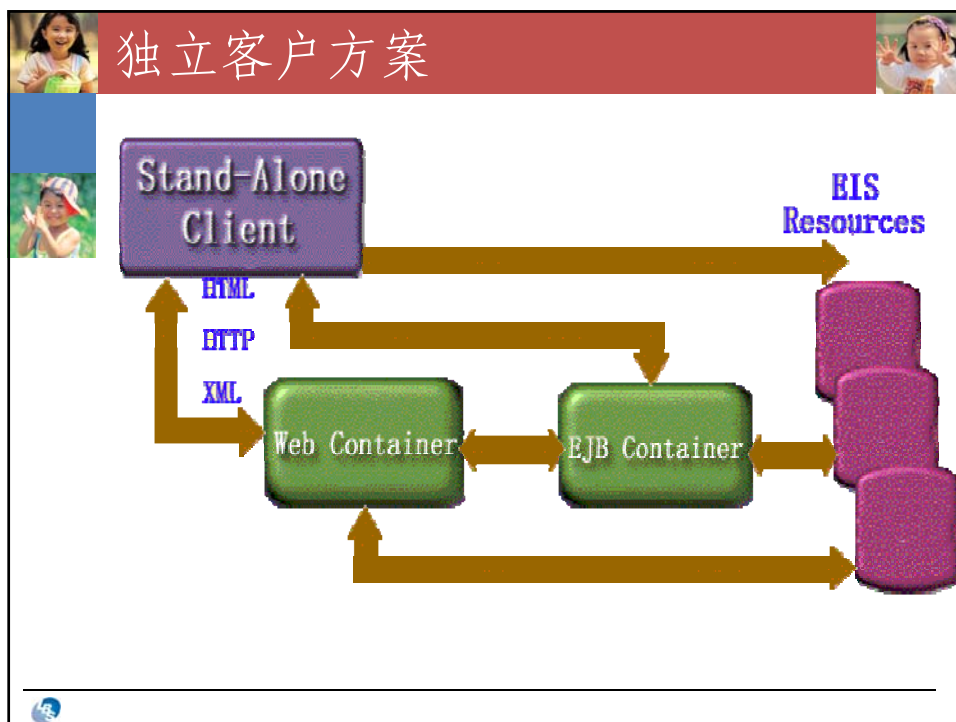


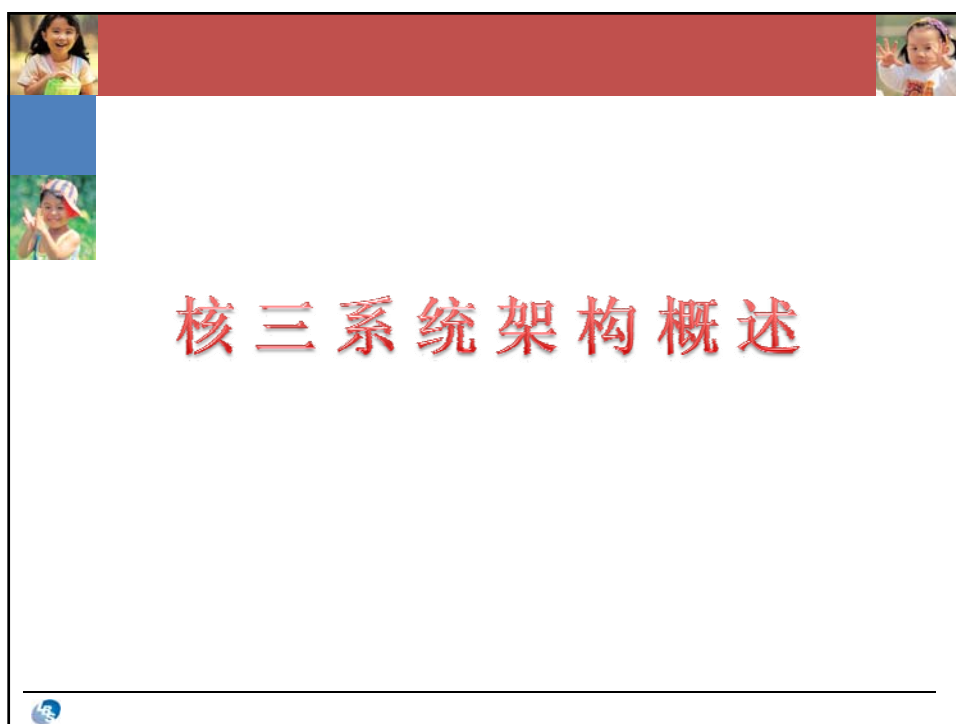
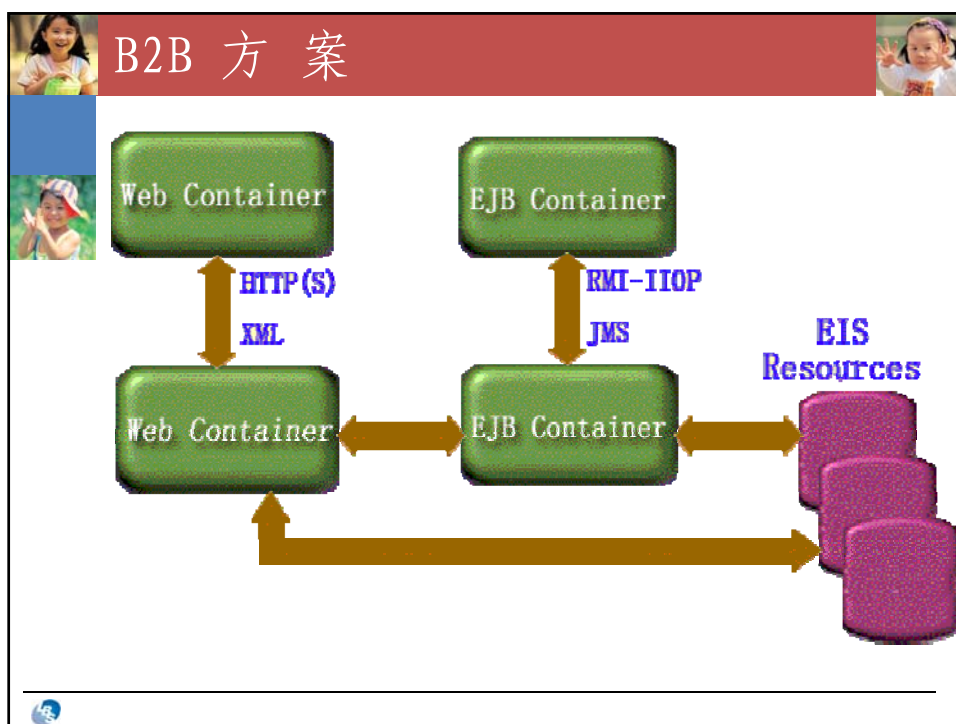
多层应用程序模型




B/S/S多层应用程序方案












架构概述






核心平台三版的系统架构是基于JavaEE平台的多层应用体系结构，同时兼容B/S/S和C/S/S两种应用结构。



- ❖ 系统架构的基础
- ❖ 系统架构要解决的问题
- ❖ 架构总图及说明
- ❖ 架构的层次调用关系
- ❖ 系统技术标准
- ❖ 系统整体数据流图





架构概述





- ❖ 系统架构的基础



- ❖ 系统架构解决的问题
- ❖ 架构总图及说明
- ❖ 架构的层次调用关系
- ❖ 系统整体数据流图
- ❖ 系统技术标准
- ❖ 系统框架的组成



1、系统架构的基础

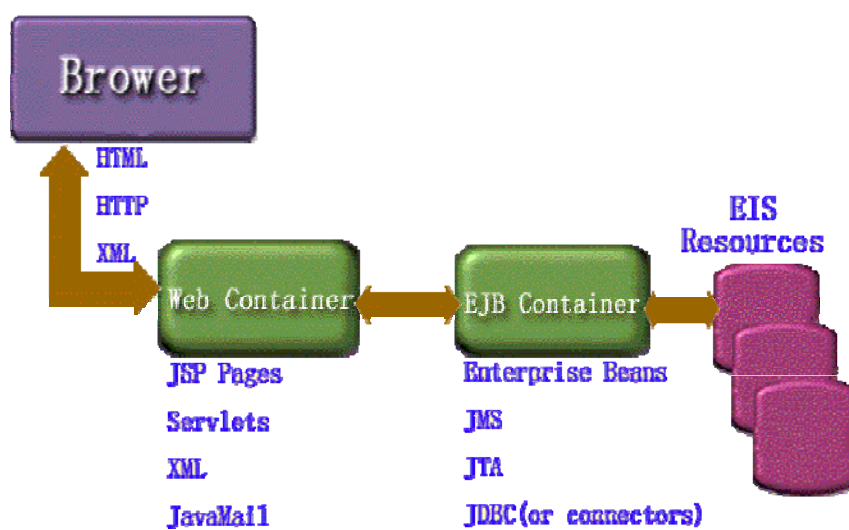
❖ 核心平台三版系统架构基于JavaEE标准应用方案的多层应用程序方案，同时兼容独立客户端方案，并在此基础上针对社保行业的特点进行了进一步的扩展与定制。

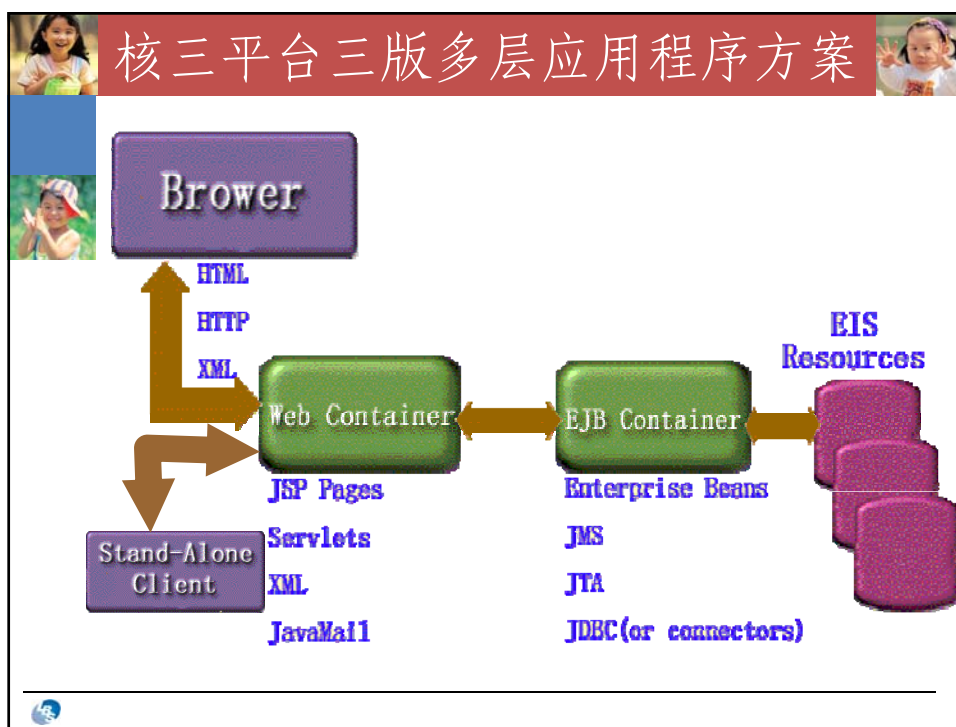
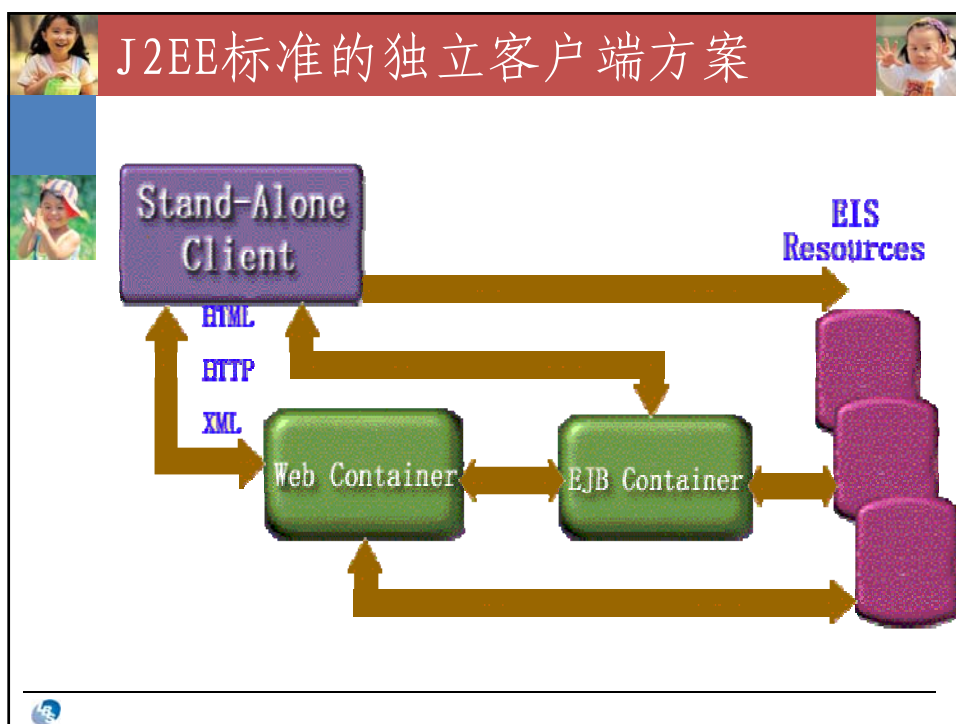
■ J2EE标准应用方案的多层应用程序方案

■ J2EE标准应用方案的独立客户端方案

■ 核心平台三版多层应用程序方案

J2EE标准的多层应用程序方案







架构概述






- ❖ 系统架构的基础





- ❖ **系统架构解决的问题**
- ❖ 架构总图及说明
- ❖ 架构的层次调用关系
- ❖ 系统整体数据流图
- ❖ 系统技术标准
- ❖ 系统框架的组成






核心平台三版主要解决的技术问题




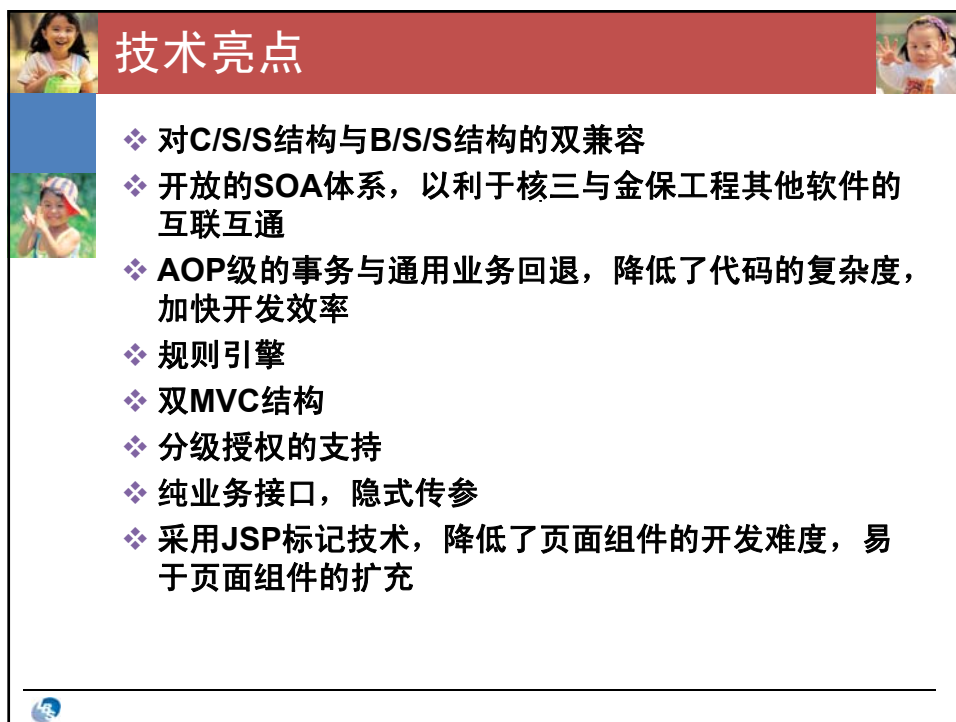
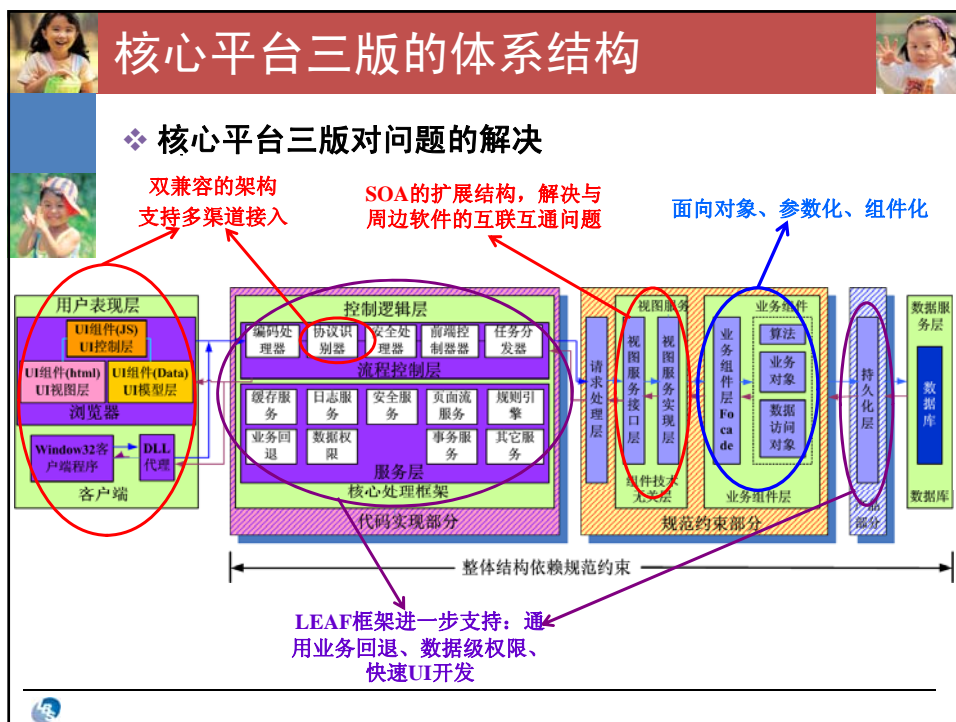



- ❖ **兼容多渠道访问的问题（兼容B/S/S和C/S/S）**




- ❖ **由封闭式单软件架构向开放式集成架构转变（SOA）**
- ❖ **通用业务回退**
- ❖ **数据级权限问题**









技术亮点





- ❖ X-ART UI组件对大规模页面开发的快速支持
- ❖ 通用业务回退的技术底层支持
- ❖ 数据级权限
- ❖ 组件技术无关性体系的建立，更好的适应了本地化厂商对技术多样性支持的要求。
- ❖ 持久层对O/R Mapping中间件的引入简化了开发，使持久层的开发更易于管理
- ❖ 支持定时任务
- ❖ 使用多线程满足特殊业务需求





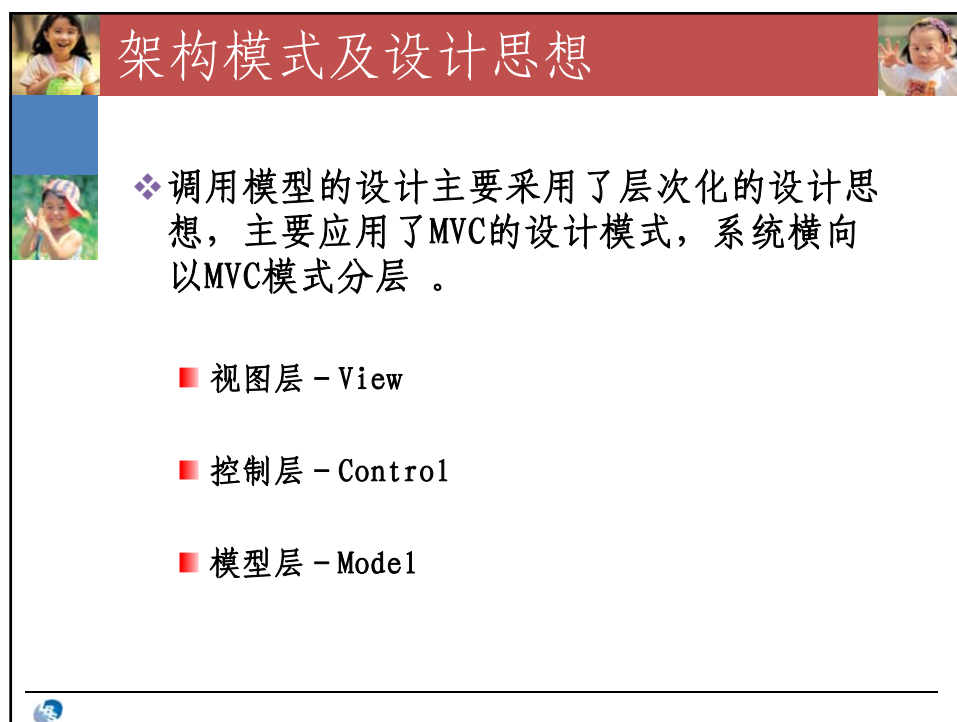
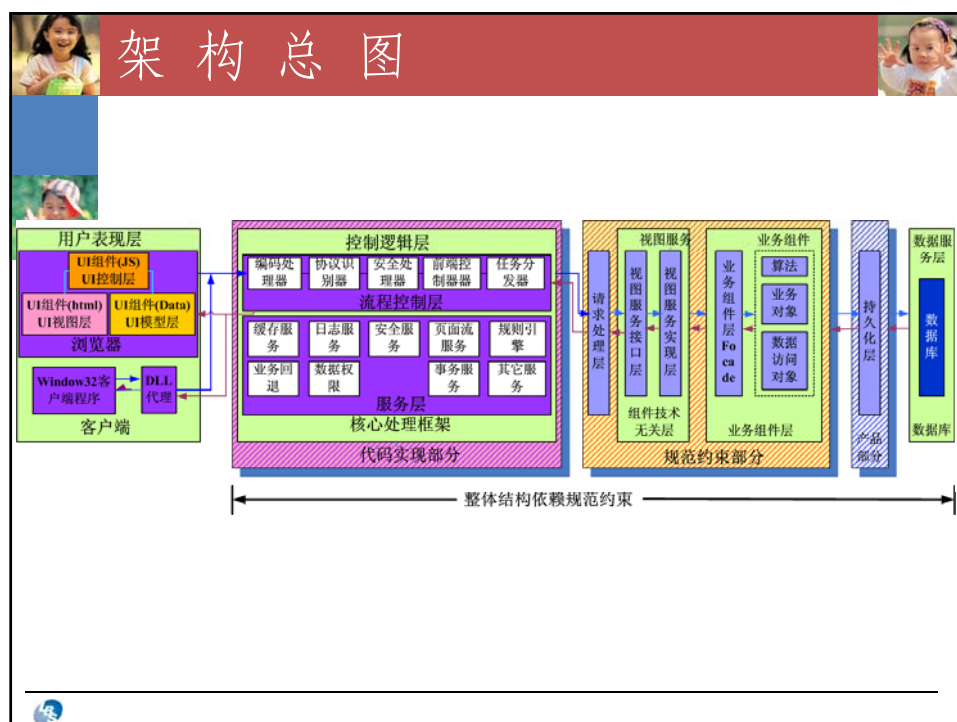
架构概述







- ❖ 系统架构的基础
- ❖ 系统架构解决的问题
- ❖ 架构总图及说明
- ❖ 架构的层次调用关系
- ❖ 系统整体数据流图
- ❖ 系统技术标准
- ❖ 系统框架的组成










架构 - 视图层







- ❖ 在MVC设计模式的实现中，视图层一般为浏览器上显示的页面，核三框架支持两种形式的视图层——浏览器、传统应用客户端。
- ❖ 浏览器做为视图层的技术
 - 使用HTML、JSP、JSF、Applet等界面技术
 - 实现客户端零维护
- ❖ 传统客户端通过DLL函数库与服务器通讯。
 - 使用Delphi、PowerBuilder、.net等技术开发客户端应用
 - 与第三方应用的无缝连接性
 - 加强了客户端的交互能力
 - 加强了客户端对打印、报表的支持
 - 保留了用户对界面的操作习惯
 - 客户端自动在线更新






架构 - 视图层





- ❖ 对浏览器做为视图层的支持
 - 提供了一套灵活、强大的标签库，大简化了Web界面的开发
 - 基于Struts1.x的MVC视图层支持
- ❖ 对传统客户端视图层的支持
 - 为应用提供了与Web客户端一致业务处理模式
 - 客户端使用dll函数库简化与服务器端的交互，dll函数库由本地化厂商自行开发





架构—控制层



- ❖ 核心平台三版的系统框架本身就是其控制层，控制层也是框架的主要组成部分。控制层在设计中分成两个部分：

- 流程控制

- 控制业务的流转

- 系统服务

- 提供系统日志、安全、事务等服务



架构—模型层



- ❖ 模型层完成了主要的业务逻辑处理，系统框架为应用提供了易于扩展的模型层接口与处理机制，为应用的业务层开发提供了基础支持，使业务层容易开发、能够最大限度的重用。
- ❖ 系统框架提供了系统管理、日志等基本支撑功能
- ❖ 在组件的数据交互接口不变的前提下，如果业务逻辑发生变化则只需要修改组件内部逻辑，实现了两个优点：
 - 层次间的松耦合
 - 业务逻辑的封装



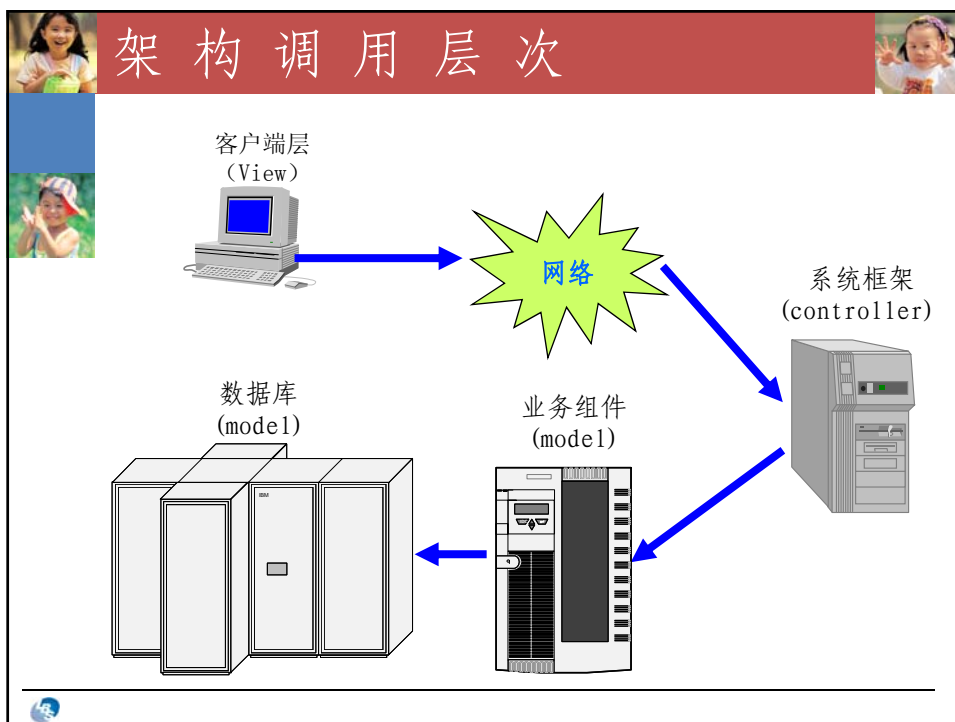


架构概述



- ❖ 系统架构的基础
- ❖ 系统架构解决的问题
- ❖ 架构总图及说明
- ❖ **架构的层次调用关系**
- ❖ 系统整体数据流图
- ❖ 系统技术标准
- ❖ 系统框架的组成

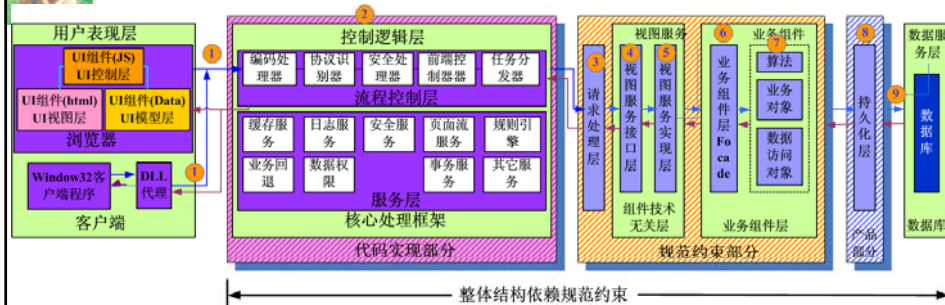




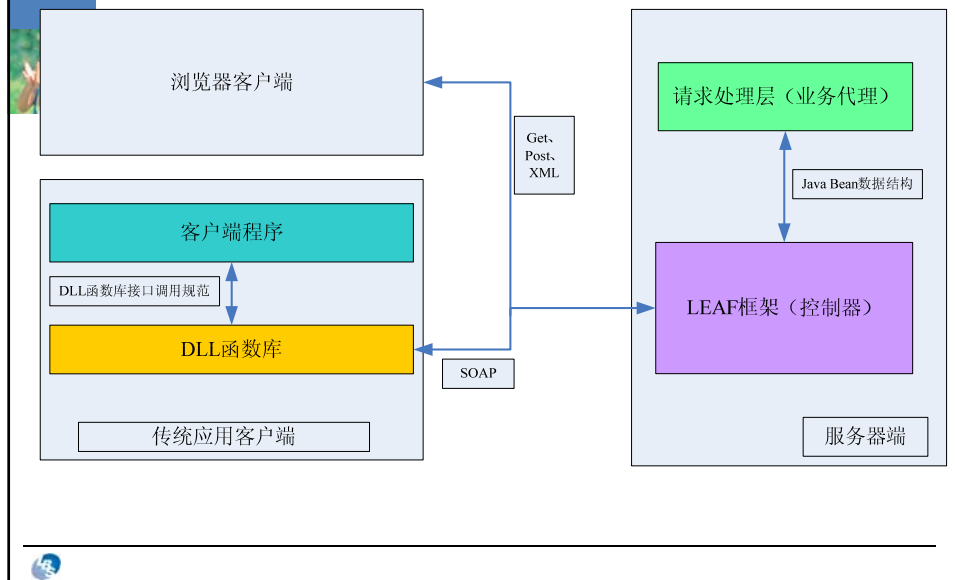
各个层次间的接口

- ❖ 客户端层与系统框架通讯的接口
 - Http get、Http post、XmlHttp
 - Http SOAP XML
- ❖ 系统框架与业务请求处理层的接口
 - Struts ActionForm
 - Http请求参数
- ❖ 业务请求处理层与业务组件的接口
 - 可序列化的业务对象
- ❖ 业务组件与数据库的接口
 - Hibernate3.x
 - JDBC

各个层次间调用顺序

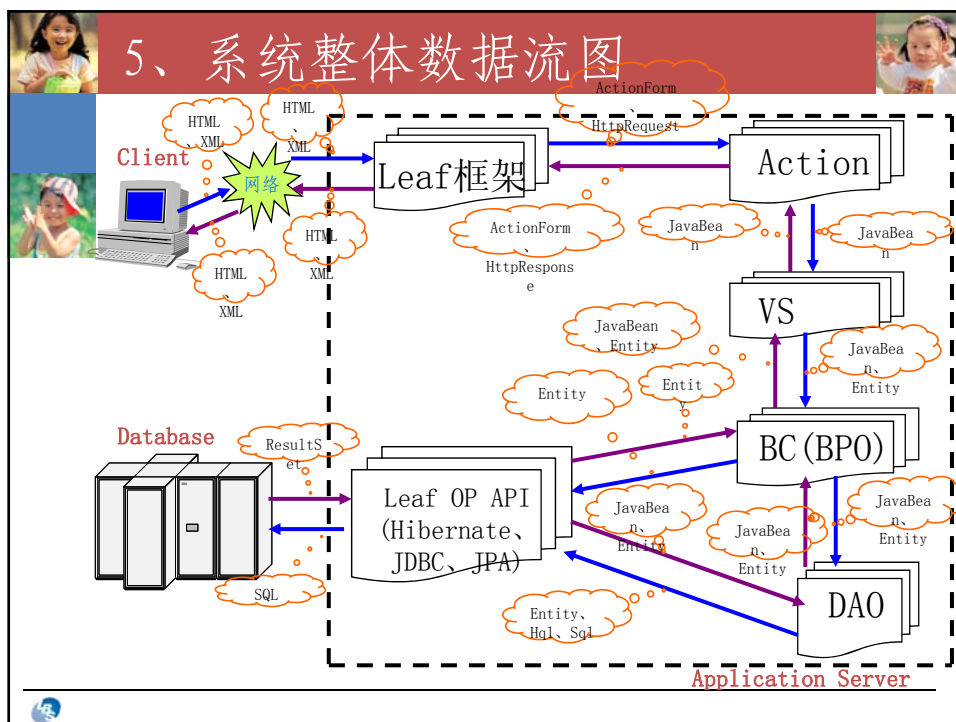



层次调用图-1




架构概述


- ❖ 系统架构的基础
- ❖ 系统架构解决的问题
- ❖ 架构总图及说明
- ❖ 架构的层次调用关系
- ❖ **系统整体数据流图**
- ❖ 系统技术标准
- ❖ 系统框架的组成






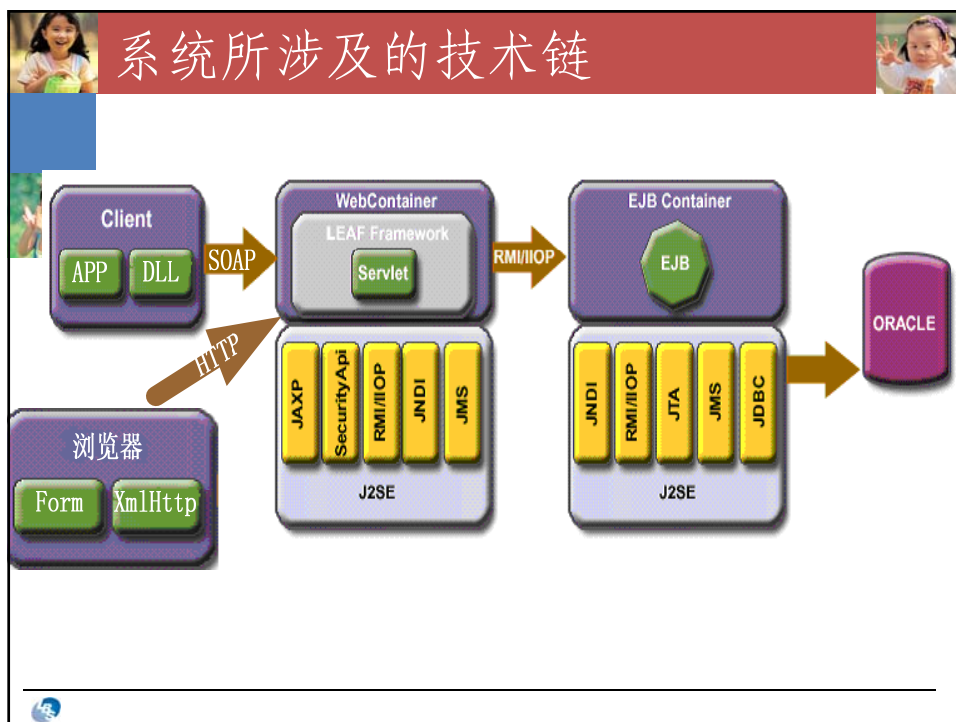
系统技术标准





- ❖ J2EE1.4
 - Servlet2.4、JSP2.0、EJB2.1
- ❖ JavaEE5
 - Servlet2.5、EJB3.0、JSF1.2、JSTL1.2
- ❖ 其它规范
 - SOAP规范







架构概述





- ❖ 系统架构的基础
- ❖ 系统架构解决的问题
- ❖ 架构总图及说明
- ❖ 架构的层次调用关系
- ❖ 系统整体数据流图
- ❖ 系统技术标准
- ❖ **系统框架的组成**






核心平台三版的组件结构












系统框架的组成





- ❖ 应用控制器
- ❖ 应用服务
 - Web UI标签库
 - 实用工具api
 - 持久化服务
 - 日志服务
 - 安全服务
 - 业务回退、数据权限服务
 -
- ❖ 系统管理
 - 授权管理
 - 日志管理
 - 通用回退、系统参数、用户消息、.....






架构对技术问题的解决













架构对技术问题的解决







- ❖ 对C/S/S结构与B/S/S结构的双兼容
 - 通过协议识别过滤器，实现统一接收来自浏览器和独立客户端的请求，转换为统一的请求格式。
 - 业务实现上除了**Action**返回的数据不同之外，其它的实现完全相同，可以最大程度地复用业务实现代码。
- ❖ 开放的SOA体系，以利于核三与金保工程其他软件的互联互通
 - 视图服务层可以被再包装为**WebService**、**EJB**等服务组件，与其它软件互联互通。
 - 协议识别过滤器支持直接接收来自第三方软件的**SOAP**请求，并对请求做出响应。
 - **SOAP Client API**提供了简捷方便的向三方软件发送**HTTP SOAP**请求的功能。







架构对技术问题的解决







- ❖ AOP级的事务与通用业务回退，降低了代码的复杂度，加快开发效率
 - 简单地将**VS**的方法名以**TA**和**ROLTA**结尾，就能实现事务与通用回退的支持。
- ❖ 规则引擎
 - 在数据库中配置的算法规则实现了动态的算法分配。
- ❖ 双MVC结构
 - 客户端的**MVC**结构使得大规模地创建复杂的**UI**界面和交互得到了简化。
- ❖ 分级授权的支持
 - 系统管理中操作权限的分配实现了分级授权，对复杂的分级授权模型提供了支持。







架构对技术问题的解决







- ❖ 纯业务接口，隐式传参
 - 业务层接口参数为纯业务参数，当需要访问诸如当前用户这样的状态信息时，可以通过隐式传参获取。
 - 保证了业务接口的纯粹性，避免了技术参数对业务接口的污染。
- ❖ 采用JSP标记技术，降低了页面组件的开发难度，易于页面组件的扩充
 - J2EE1.4中的JSP标记技术，可以方便地创建JSP片断进行复用，能够实现单纯用JSP标签无法实现的功能。
- ❖ X-ART UI组件对大规模页面开发的快速支持
 - 丰富的UI组件库对大规模复杂UI开发有重要作用。







架构对技术问题的解决







- ❖ 通用业务回退的技术底层支持
 - 基于数据库实现的通用业务回退功能，实现了用最少的开发工作量实现足够的功能。
- ❖ 数据级权限
 - 实现了基于统筹区的数据级权限支持api。
- ❖ 组件技术无关性体系的建立，更好的适应了本地化厂商对技术多样性支持的要求。
- ❖ 持久层对O/R Mapping中间件的引入简化了开发，使持久层的开发更易于管理
- ❖ 定时任务
 - 引入了基于QuartZ的定时任务功能。






架构对技术问题的解决





- ❖ 使用多线程编写特殊业务
 - 对于耗时的复杂业务，以多线程方式实现，加快执行速度。
- ❖ 操作权限的过滤与校验
 - 基于HTTP的url进行授权与校验。
 - 使用Servlet过滤器对操作权限进行检查。
- ❖ 应用的初始化与清理
 - 使用ServletListener在应用启动时初始化系统参数与缓存，在应用关闭时清理资源。
- ❖ 防止忘记关闭数据库连接的设计
 - 在过滤器中始终关闭数据库连接，防止意外的忘记关闭。





架构与模式










架构与设计模式







- ❖ 架构模式：MVC
 - Model
 - View
 - Controller
- ❖ 设计模式
 - 截取过滤器 (Filter)
 - 前端控制器 (Front Controller)
 - 业务代表 (BusinessDelegate)
 - 会话外观 (SessionFacade)
 - 值对象模式 (ValueObject)






架构模式：MVC





- ❖ 名称
 - 模型 - 视图 - 控制器 (Model-View-Controller, MVC)
- ❖ 概述
 - 通过将数据描述、数据表现和应用操作几个部分分离，增加系统的可复用程度。支持多个同步的数据视图
- ❖ 意图
 - 通过分解软件系统中的不同层 (layer)，简化系统的维护，提高可扩展性、灵活性和封装程度






设计模式 - 截取过滤器







- ❖ 动机
 - 常见的处理，比如：检测数据、记录每个请求、每个请求的完成情况等
 - 需要集中化常见逻辑
 - 服务应能正确地添加或者删除，而不影响已有的组件，这样我们可以以多种方式组合使用组件
- ❖ 策略
 - 自定义过滤器策略 (Custom Filter Strategy)
 - 标准过滤器策略 (Standard Filter Strategy)
 - 基过滤器策略 (Base Filter Strategy)
 - 模板过滤器策略 (Template Filter Strategy)
- ❖ 核三系统框架中的应用
 - SafetyFilter、EncodingFilter



设计模式 - 前端控制器





- ❖ 动机
 - 每个请求都要完成常见的系统服务
 - 以一个集中点处理请求
- ❖ 策略-Servlet前端策略
 - 使用控制器作为处理请求的最初联系点，该控制器管理着请求的处理，包括调用EJB缓存服务读取配置文件、处理错误、统一请求和输出的调用控制以及管理创建输出逻辑的选择
 - 提供一个控制和管理web请求处理的集中式入口点，提高代码跨请求的重用性
- ❖ 核三系统框架中的应用
 - Struts ActionServlet



设计模式 - 业务代表



❖ 动机

- 表示层客户需要访问业务服务
- 业务服务API会变化
- 希望降低表示层客户端与业务服务的耦合
- 希望降低客户端和业务服务之间的网络流量

❖ 策略-代表代理策略

- 业务代表向其所封装的会话Bean提供客户端方法的代理功能
- 业务代表来降低表示层客户端和业务服务层的耦合，业务代表隐藏了业务服务的实现细节，提供更简单的、统一的接口更好的向客户端提供服务

❖ 核三系统框架中的应用

- Action



设计模式 - 会话外观



❖ 动机

- 通过隐藏业务组件之间所有的复杂交互活动，向客户端提供一个更简单的接口
- 减少通过网络并跨越服务层被直接暴露给客户端的业务对象的数目
- 向客户端隐藏业务组件之间的低层交互和相互依赖关系
- 提供统一的粗粒度服务层，以分离业务对象实现和业务对象抽象
- 避免把底层业务对象暴露给客户端，实现两个层之间的松耦合

❖ 策略-无状态会话外观策略


- 把会话bean用作外观以封装参与工作流的业务对象之间交互的复杂，向客户端提供统一的粗粒度服务访问层
- 管理着业务数据和参与本工作流的业务服务对象之间的交互活动，他封装了与需求有关的业务逻辑

❖ 核三系统框架中的应用


- ViewService










环境介绍







- ❖ 系统架构支持的J2EE标准
 - 以J2EE1.3为主
 - Servlet、UI标签库、JSP
 - 部分特性需要J2EE1.4
 - JSP标记、JSP2.0
 - 可根据应用服务器的支持能力选择不同的EJB版本
 - 框架不限定EJB版本，应用可根据需要选择







环境介绍







- ❖ 支持的应用服务器
 - 支持J2EE1.4的应用服务器
 - 可获得J2EE1.4新特性带来的好处（JSP标记、JSP2.0等）
 - 支持JavaEE5的应用服务器
 - 可获取J2EE1.4新特性带来的好处（JSP标记、JSP2.0等）
 - 同时可使用EJB3.0，大大简化EJB开发





环境介绍





❖ 几种主流应用服务器对J2EE标准的支持

J2EE版本	Tomcat (不支持EJB技术)	Weblogic	Websphere
JDK1.2—1.4 J2EE1.3	4.1 Servlet2.3、JSP1.2	7.0	
JDK1.4 J2EE1.3	4.1	8.1	5.1
JDK1.4 J2EE1.4	5.0 (Servlet2.4、JSP2.0)		6.0
JDK1.5 J2EE1.4	5.5	9	6.1
Java5 JEE5	6.0	10	7.0



Thank You !

