
Jupiter Java版本 架构设计文档

<V1.0>

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

Revision History

Date	Issue	Description	Author
25/11/2004	<1.0>	初稿	陆建梁

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

目录

1.	引言	4
1.1	目的	4
1.2	范围	4
1.3	参考文献	4
2.	概述	4
3.	系统分层架构	4
3.1	架构风格的选择	4
3.2	本系统总体架构	5
4.	系统设计模式	5
4.1	类图	5
4.2	顺序图	7
5.	布署视图	8

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

1. 引言

1.1 目的

本文档提供了Jupiter Java版本软件架构的概览，采用若干架构视图描述系统的不同方面，以便表示构造系统所需的重要架构决策。

1.2 范围

本文档是为Jupiter的Java版本设计的，按照RUP的《软件架构文档模板》编写，用于指导Jupiter系统的Java版本。

1.3 参考文献

1. 《软件架构文档模板》，2002.5，Rational Software Corporation。
2. 《J2EE核心模式》，2002.1，Deepak Alur, John Crupi, Dan Malks, SUN MicroSystem Inc.。

2. 概述

软件架构的逻辑视图描述了该系统的主要结构和所采用的体系设计模式。设计软件架构可以最大程度的重用系统的设计和代码，还可以明确系统中每个模块和功能的功能，避免重复功能的多次开发。系统架构的逻辑视图也描述了最重要的组件，若干组件构成服务或子系统，子系统构成系统的层。

本软件架构主要受到下列系统约束的控制：

- 系统扩展性和灵活性需求，系统的设计需要具备足够的扩展性，以便于因发展或改变而对系统功能的调整和增加，便于系统升级和维护。系统的扩展性包括功能的扩展性和数据扩展性。
- 需要采用B/S结构，使用户能通过互联网访问系统数据，支持远程管理和移动办公。

本软件架构以逻辑视图表示，用Rational Rose工具基于统一建模语言(UML)开发的。

3. 系统分层架构

3.1 架构风格的选择

J2EE平台是为多层架构系统设计的，提供了较多的多层应用系统方案解决方案。例如基于WEB的系统中，WEB页面直接通过数据访问层去访问数据库；而基于EJB系统中，WEB页面可以通过EJB层和数据访问层去访问数据库。

选择由哪些层架构应用系统，取决于系统的可扩展性、可重用性及系统的复杂度。基于WEB系统开发较快，但由于所有的业务逻辑都固化在特定web页面中，所以系统的可扩展性和可重用性不太好，是适合小型系统的架构。基于EJB系统架构比较复杂，业务逻辑驻留在EJB层上，web页面只是展现数据，这样应用的具体表现方式和数据的处理过程是“松耦合”的关系，同时，EJB层采用数据访问对象与数据库交互，这样又把数据访问过程和对数据处理加工的过程分开了，即数据库系统和业务逻辑之间也是“松耦合”的关系，显然，系统的可扩展性和可重用性大大提

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

高，但构架的复杂度也增大，是比较适合中大型的复杂应用系统的架构。

Jupiter的Java版采用以Jsp、Servlet和Java类技术的Web方式，同时采用MVC的设计模式，将应用表现、业务逻辑、数据存取和控制分开来。集上述两种架构的优点为一体。

3.2 本系统总体架构

Jupiter的Java版本由应用层和业务服务层两层构成，如下图所示。

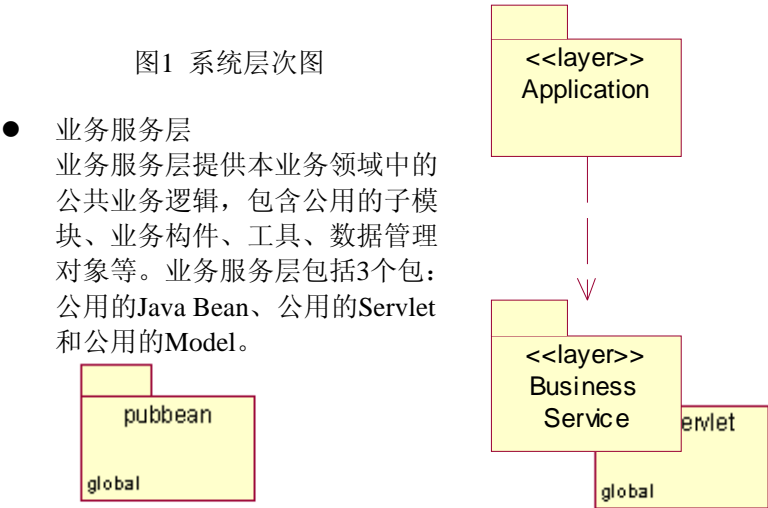


图2 业务服务层包图

- 应用层
应用层提供本系统的所有子系统和功能模块，包括和用户沟通、响应客户请求、以及和本系统需求特定的业务处理。Jupiter产生的Java代码应用层分为若干个包，具体数目由应用决定。

4. 系统设计模式

Jupiter的Java版本主要采用MVC(Model-View-Controller)模式设计。本节给出了本系统基于MVC的设计模式。

4.1 类图

系统构架中的对象从逻辑上分为三类：view对象，负责处理系统数据的展现和与用户的交互；model对象，负责处理业务逻辑和数据加工；controller对象，负责协调view对象和model对象，即接受用户的请求，控制model对象对相应数据进行处理，最后把结果数据展现给用户。MVC架构将数据、表现和业务逻辑分离开来，具有良好的伸缩性、可维护性和可扩展性。

架构的view部分定义了与用户交互的接口，它由屏幕视图（JSP）和视图控制器(Servlet)组成。屏幕视图用于产生动态网页以响应客户；视图控制器提供管理和控制web请求的集中式入口，视图控制器将接收所有的请求，交给合适的controller对象处理，将处理结果交给下一个视图，以及实现必要的安全需求。

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

架构的model部分负责实现业务逻辑，以及和数据库或者其他数据的信息交互，包括业务逻辑处理对象和数据访问对象。

架构的controller部分负责协调model和view的通信和状态的同步，包括事件分发、数据校验和封装。数据交换通过对对象完成，简单高效。

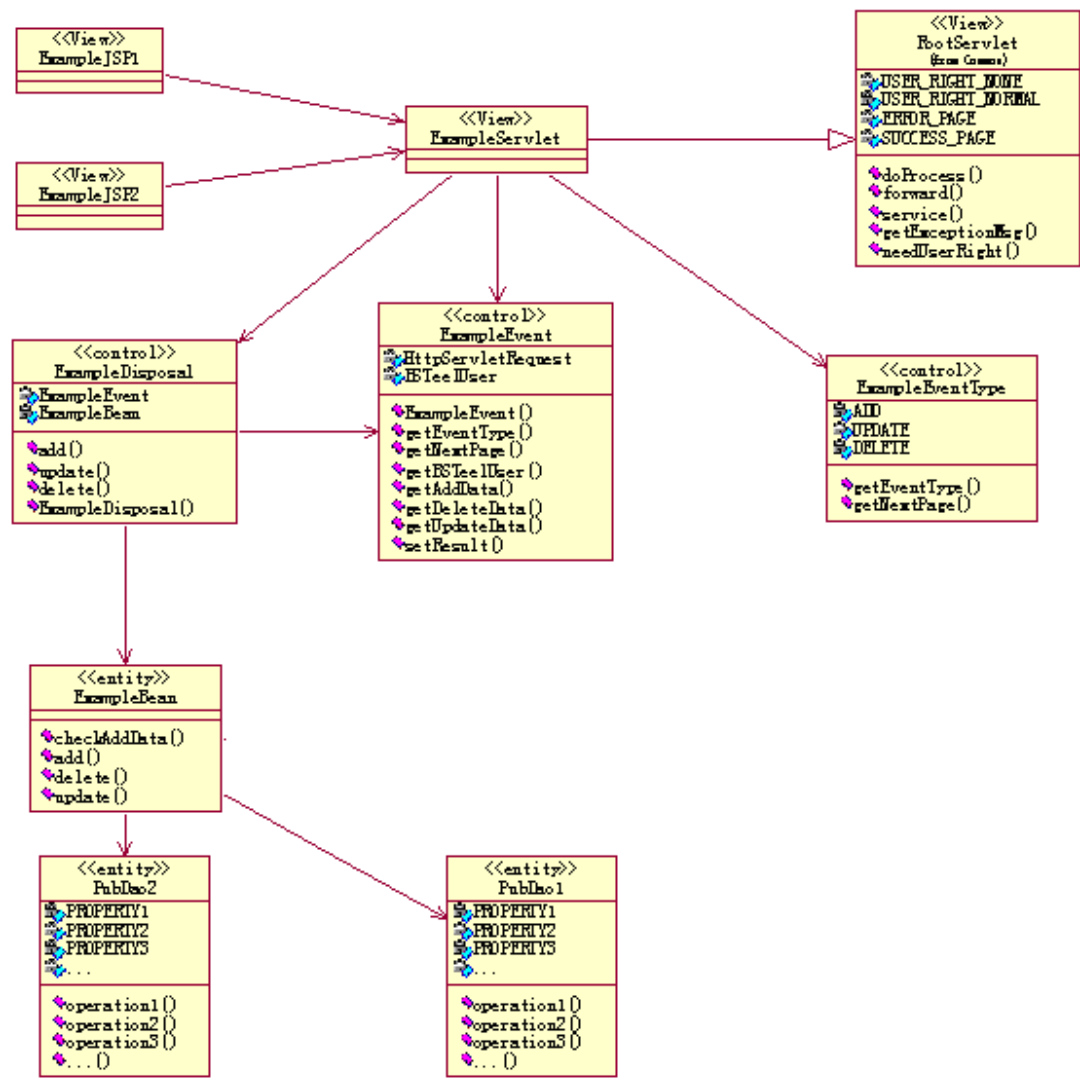


图4 系统设计模式的类图

表1：类说明表

Class	Steretype	Description
ExampleJSP1	view	客户端JSP显示页面
ExampleJSP2	view	客户端JSP显示页面
ExampleServlet	view	前端控制器，接收所有的请求，交给合适的controller对象处理，将处理结果交给下一个视图，以及实现必要的安全需求。
ExampleDisposal	controller	事件分发类，对应每一用户事件有一方法负责组合调用业务

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

Class	Steretype	Description
ExampleEvent	controller	逻辑处理类的一个或多个方法来完成该事件的操作。
ExampleEventType	controller	数据封装类，负责校验JSP页面提交的数据的有效性，并封装成需要的数据格式供业务逻辑处理类使用。对应每一用户事件有一相应方法封装整理其数据。
ExampleBean	model	事件类型类，定义所有用户事件，并定义事件处理完毕后的显示页面，如果不定义显示页面则系统自动显示默认成功页面。
PubDAO1	model	业务逻辑处理类，完成实际的业务逻辑处理和数据库操作，其方法对应的功能粒度比分发类小，比实体类大，一个或多个方法组合完成某事件的操作。具有一定的重用性。
PubDAO2	model	公用数据实体类1，负责公用数据的存取。
PubDAO2	model	公用数据实体类2，负责公用数据的存取。

4.2 顺序图

以新增操作事件流为例，下图给出了对象间的动态合作关系，即顺序图。

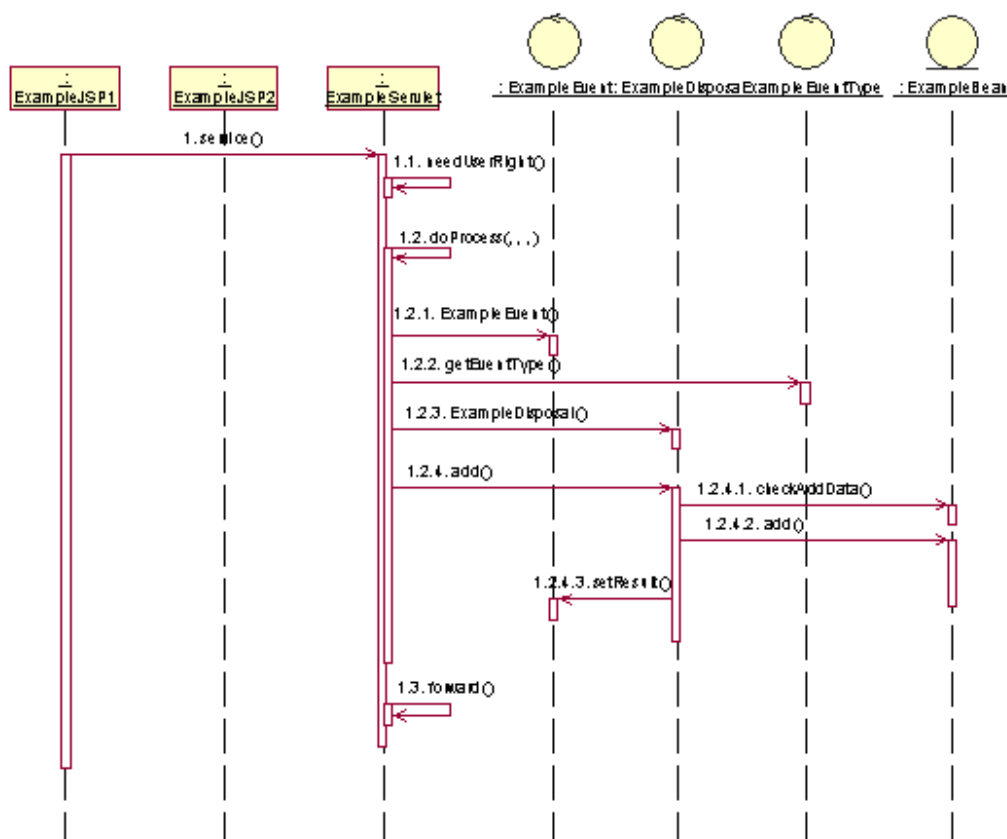


图5 系统设计模式的顺序图

	Issue: <1.0>
架构设计文档	Issue Date: <1/8/2003>

5. 部署视图

由于本方案采用Web模式设计，客户端只需要安装基本的操作系统和互联网浏览器就可以使用本系统，因此，系统的网络拓扑非常简单，如下图所示。图中显示，本系统的网络结构主要由数据库服务器、应用服务器、WEB服务器和客户端组成。物理上，这些服务器可在一台机器上运行，而可分别占用一台机器。应用采用ASP方式运行。

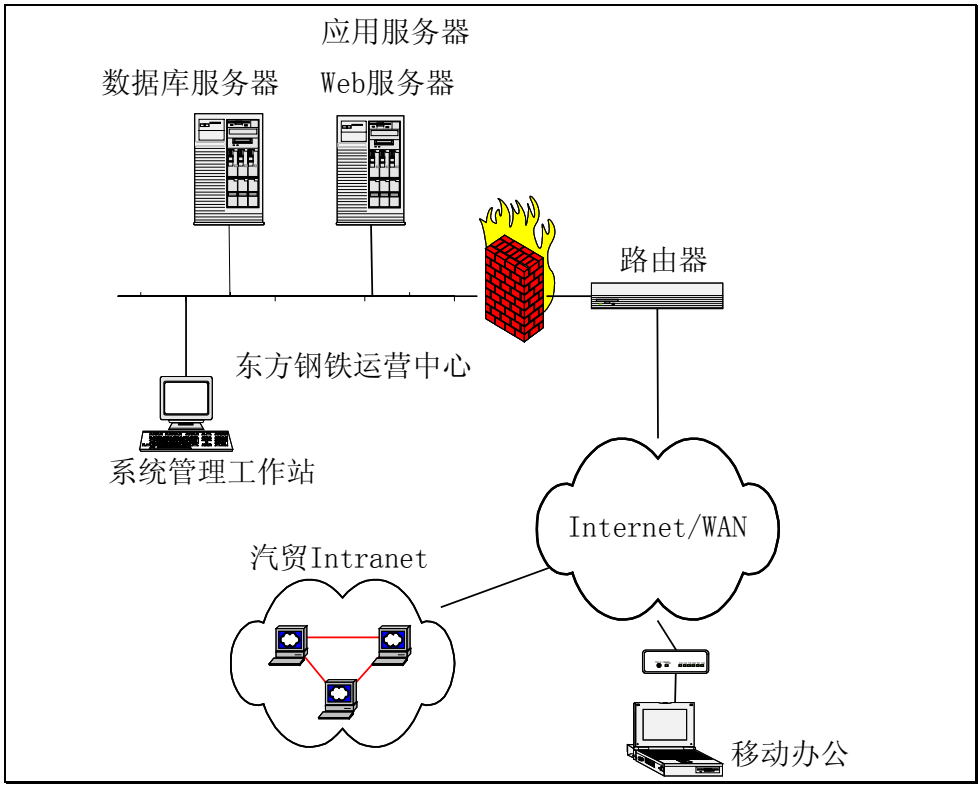


图5. 系统部署视图