基于敏捷方法的轻量级 J2EE 架构的应用①

Application of Lightweight J2EE Structure Based on Agile Method

戚 琦　廖建新　王 纯　武家春
（北京邮电大学网络与交换技术国家重点实验室 北京 100876）
（东信北邮信息技术有限公司 北京 100083）

摘要：本文提出 Struts + Spring + Hibernate 三个开源框架的松耦合层次结构，加之 Tomcat 开源服务器组成一个轻量级的 J2EE 架构。文章结合敏捷软件开发方法说明了轻量级 J2EE 架构设计实现的过程，并针对目标系统彩信短信管理平台设计了 Struts + Spring + Hibernate 的系统概念模型。本文希望通过目标系统的开发探索敏捷开发的思想在轻量级 J2EE 系统架构中的应用。

关键词：敏捷建模 J2EE Struts Spring Hibernate

1 引言

随着企业级应用的发展，从 web 应用系统到 J2EE（Java 2 Platform Enterprise Edition）领域已存在众多框架，在系统架构分析时，如何应用这些已有框架为系统量身定做一个合适的架构，对 J2EE 设计开发者提出了挑战。敏捷建模的思想提倡使用一组简单的实践和增量式的建模完成开发，系统的需求分析、架构设计等应该尽可能保持简单，并且具有较高的可扩展性。EJB（Enterprise Java Bean）是为解决如分布式对象和远程事务等复杂问题而设计[1]，基于 EJB 的 J2EE 架构也并非适用于所有系统。对于大多运行在一个 JVM（Java Virtual Machine）上的系统，如果没有使用 RMI（Remote Method Invocation）等远程对象访问技术，采用重量级的 EJB 是没有必要的。在广大的 J2EE 开发者在追寻一种轻量级 J2EE 架构同时，都从设计的角度阐述了轻量级 J2EE 架构的特点和优势[2]，然而从软件工程的角度分析，Struts + Spring + Hibernate 的轻量级 J2EE 架构在设计建模方面是符合敏捷开发的思想的。本文在此主要基于敏捷建模的思想分析轻量级 J2EE 架构。

本文的目标系统是彩信短信营销管理平台，该系统为各省移动通信公司提供 web 方式的管理接入，以满足移动对维护自身数据的需求并可以向用户提供更为丰富和方便的服务。

2 基于敏捷思想的轻量级 J2EE 系统架构

2.1 敏捷建模的思想

系统架构设计在软件的生命周期中是十分重要的，是从需求分析到模块设计的一个承上启下的阶段，关系着整个系统的成败。我们在对敏捷建模（AM）方法进行了研究分析后，通过实践发现其中的一些核心原则和思想对于系统架构设计有着重要的指导意义。在此，我们讨论敏捷建模的以下三个核心原则对于构建轻量级 J2EE 架构的启发。

主张简单。敏捷思想告诉我们“不应该为支持未来可能的需求而过度地架构系统”[3]。例如，目标系统的企业级应用并不复杂，只是使用一些简单的 JAVA 对象提供透明的事务以支持移动数据业务的管理，不需要分布式计算也没有远程方法调用。因此，系统的设

① 基金项目：国家杰出青年科学基金（No. 60525110）；新世纪优秀人才支持计划（No. NCET-04-0111）；高等学校博士学科点专项科研基金资助课题（No. 20030013006）；电子信息产业发展基金项目（基于3G 的移动业务应用系统）；电子信息产业发展基金重点项目（下一代网络核心业务平台）；电子信息产业发展基金项目（基于内容的综合通信网络计费平台）；国家高技术产业化信息化装备专项项目（支持数据增值业务的移动智能网系统）

计人员就没有必要承担 EJB 如此众多的 java 文件和部署文件，运行该系统的服务器也不需要承担运行重量级 J2EE 容器的负担。

包容变化。需求随着时间在发展，开发者和用户对需求的理解也都随着时间在改变[3]。要做到包容变化，需要在软件开发工作中采用一种增量的方法，每次只改动系统的一小部分，而不是试图在一个大的发布版本中完成一切。因此，包容变化要求系统架构应该具有可伸缩性，能够较好地支持增量开发的方法。轻量级的 J2EE 系统架构中，表现层使用 Struts 框架，使页面设计与业务逻辑分离，同时，业务逻辑层的 Spring 框架降低了组件之间的耦合度，很好的支持了系统变化。

快速反馈。敏捷建模的思想要求开发人员尽可能早的通过代码验证模型的正确性[3]，这一思想对代码的可测试性提出了要求。Spring 框架作为轻量级 J2EE 系统的核心，以 JavaBean 为基础，通过反向控制实现了按接口编程，使得单元测试非常容易，修改代码后也不需要启动容器，在很大程度上提高了代码的可测试性。

2.2 轻量级 J2EE 架构

J2EE 架构至少包括三个层次：表现层、业务逻辑层和持久化层。目标系统没有使用远程服务，也没有分布式计算，但是要求系统有事务管理能力，并具有可扩展性。由此可见，如果系统选择使用 EJB 为核心的 J2EE 架构，则会导致系统过度设计，增加了应用的复杂度，使系统性能下降，单元测试困难。然而，Spring 框架利用其核心思想依赖注入和面向切面编程整合 Struts 和 Hibernate，实现了 J2EE 的三层结构，使表现层、业务逻辑层和持久化层彻底分离，保持了层次和组件之间的松耦合[4]。图 1 描述了轻量级 J2EE 的系统架构设计。

表现层框架 Struts。Struts 框架作为企业级 WEB 应用的框架，实现了 MVC 模式，将 Servlet、JSP、以及 Struts 标签和信息资源统一到一个框架中，再由基于 XML 的配置文件 Struts – config. xml 将模型、视图和控制器联系起来。Struts 框架使页面设计和控制器逻辑进行分离，方便了 J2EE 系统上层页面的开发，并具有组件的模块化、灵活性和可重用性等优点，很大程度上简化了 Web 应用程序的开发。Struts 框架使数据与逻辑分离，降低了模块之间的耦合，提高了组件间的内聚，程序的控制流程通过 XML 文件配置，当程序发生变化时，只需修改 XML 配置文件即可方便的对程序流程进行控制，提高了软件的可扩展性。可见，Struts 框架在诸多方面符合敏捷建模的思想，是轻量级 J2EE 架构表现层的较好选择。

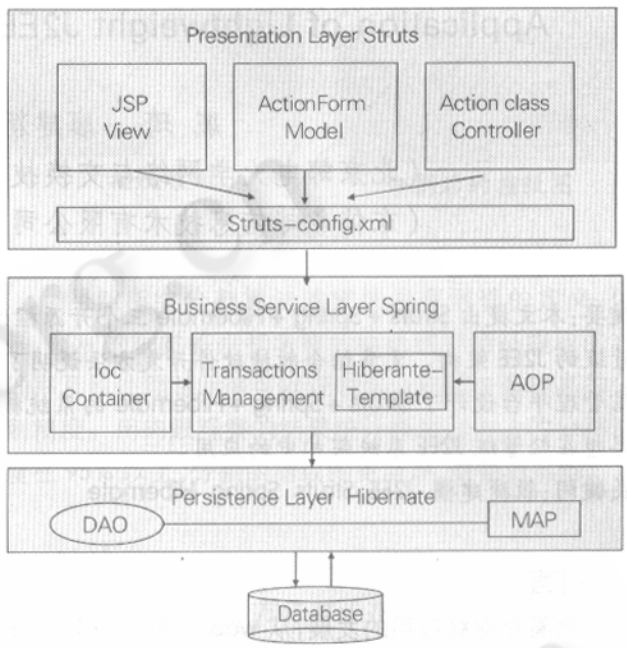

图 1　轻量级 J2EE 系统架构图

业务逻辑层架构 Spring。Spring 以简单、可测试和松耦合为理念，简化了企业级系统的开发[2]。它相对 EJB 技术而言，使系统在满足需求的前提下保持了组件尽量简单。Spring 通过接口松耦合的 JavaBean 模型提供了基于 Ioc（反向控制）容器的 BeanFactory 和基于 AOP（面向切面编程）框架的 JavaBean 组件。开发者使用 Spring 只需在 JavaBean 的配置文件 XXXBean. xml 或者 Config. xml 中修改有关属性，就可以完成业务方法层组件的配置与加载。

（1）反向控制。Ioc 是 Spring 框架的核心思想之一[1]。Spring 通过配置文件 XXXBean. xml 将对其他对象的引用通过组件提供的 Set 方法设定，在对象创建时由 Ioc 容器实现将依赖注入到对象中，从而降低了组件之间的耦合度。同 EJB 一样，Spring 的 Ioc 容器支持多个开发人员按组件进行开发，它通过配置文件管理所有系统应用的组件，并且在这些组件之间建立关联。然而，较 EJB 相比，Ioc 容器省去了在 EJB 容器中部署的复杂步骤，只需在 XXXBean. xml 配置文件中修改相应的属性，十分简洁方便，并且支持重用，容易测试。

（2）面向切面。敏捷建模的思想认为开发者不应该将同样或类似的代码写在多个地方。Spring 引入 AOP 概念，将通用的操作模块化到特定对象中[1]，如记录操作员日志，称之为切面，在需要这一功能的地方，通过 XXXBean. xml 配置文件声明式地应用这些功能，从而在一个方法执行之前、之后或其中运行通知。使用 AOP 后，公共服务可以作为切面被织入到目标对象中，却不会增加目标对象的复杂性，AOP 的概念可以形象的如图 2 所示。

图 2　面向切面逻辑说明图

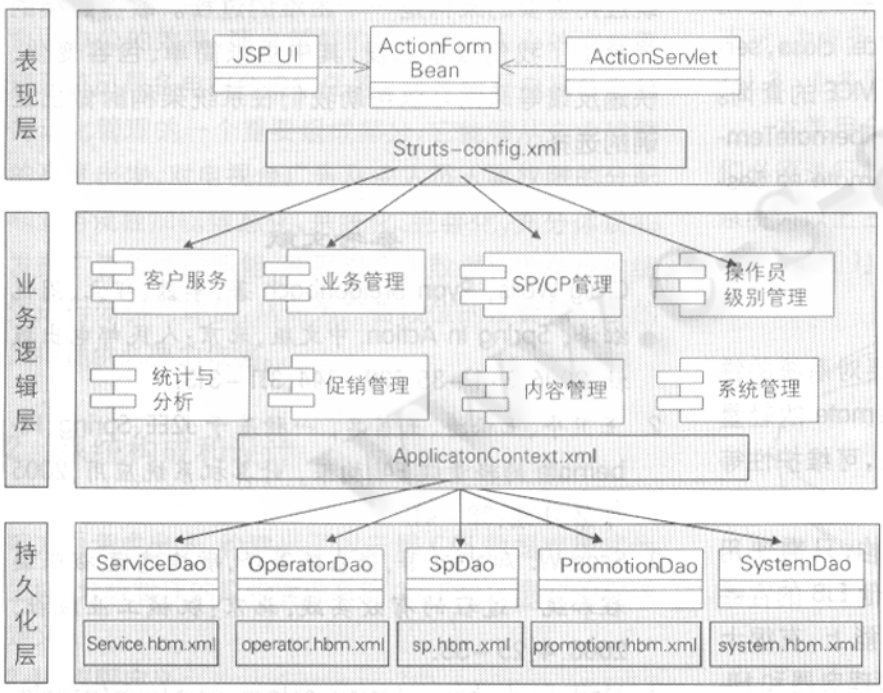

图 3　系统概念模型图

（3）事务管理。事务是企业级应用开发的重要组成部分，它保证一系列操作完全成功。如果其中某步出现问题，前面进行的 n 步操作要进行回滚。Spring 框架作为轻量级 J2EE 系统的核心，提供了对程序控制事务管理和声明式事务管理的功能。Spring 框架将事务管理抽象化，如果系统只使用了一个数据库作为持久化资源，Spring 可以利用 Hibernate 等持久化层，对事务管理进行支持。另一方面，对于声明式的事务管理，Spring 提供了一个用标准 java 语言编写的 AOP 框架作为事务管理的基础。

持久化层架构 Hibernate。Hibernate 是对 JDBC API 轻量级的封装，是连接 Java 应用程序和关系数据库的中间件[5]。它的工作原理是通过文件把值对象和数据库表之间建立起一个映射关系，开发者可以以操作对象的方式来操作数据库。轻量级 J2EE 系统架构中，Hibernate 代替 EJB 的 Entity Bean 在完成 Java 对象的持久化的同时对开发者屏蔽了所有数据库访问的细节。EJB 的 Entity Bean 对硬件内存的要求较高，对于开发者而言也过于复杂，导致了系统开发进度延长，软件成本增加；传统的最直接的数据库访问方式 JDBC API，运行效率较高，但会使整个系统存在由于没有完全释放连接的风险，同时也导致了代码的可维护性和可移植性较差。Spring 对于 Hibernate 持久化框架提供了强大的支持，利用 HibernateTemplate 类使 Hibernate 的优势得到了充分发挥。

3　应用敏捷方法实现系统

3.1　系统的概念模型

根据 Struts + Spring + Hibernate 轻量级 J2EE 的架构，我们设计了彩信短信营销管理平台的系统概念模型。系统的表现层由 Struts 框架的 JSP 文件和对应的 ActionForm 以及 Action 组成，业务逻辑层根据 Spring 框架设计了八个功能组件，操作员日志记录功能则由面向切面方法织入各个功能组件。功能组件中利用 Spring 提供的 Hibernate Template，封装了对持久化层的访问。其具体结构如图 3 所示。

3.2　系统实现

Spring – struts 实现。系统的表现层上，使用 Struts

的 MVC 结构，通过 Struts 配置文件，将 JSP 页面文件与 Action 控制类对应起来，并将 Spring 业务逻辑层的业务组件通过 Spring 的 Acton 代理类注入 Struts 的 Action 控制类中。例如，帮助用户订购业务的控制类 HelpUserSubsribeAction. java，作为 Struts 框架的基本 Servlet 即 Action 的子类，它实现了 execute 方法，并通过 HelpUserService 类型的私有成员变量 helpUserService 和 setHelpUserService 方法实现了 Spring 业务逻辑组件 HelpUserService 的注入。

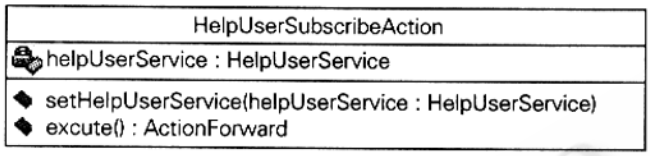

图 4 HelpUserSubscribeAction 类图

Spring – Hibernate 实现。系统的业务逻辑组件中，使用 Spring 提供的 HibernateTemplate 模板，执行 Hibernate 的数据库访问操作，当处理失败时，可由 Hibernate 提供的事务管理机制进行回滚。例如，实现业务数据库访问的 ServiceDaoHibernate 类，通过 HibernateTemplate 模板，在 getService 方法中只需使用 (Service)? hibernateTemplate. get (Service. class, serviceid) 语句，就可以实现对数据库表 SERVICE 的查询。此外，在 spring 配置文件中需要配置 HibernateTemplate 类，从而在代码中不必实例化 Hibernate 的 SessionFactory 类，简化了 Hibernate 的使用。

4 轻量级 J2EE 架构的性能分析

敏捷开发的思想，要求系统架构者要对系统的性能提出评价。基于 Struts + Spring + Hibernate 的轻量级 J2EE 架构，在系统运行性能、可扩展性，可维护性等多方面的性能上具有优势。

系统运行的性能。轻量级 J2EE 架构，只需使用 Tomcat 服务器，配置运行小巧灵活，较其他 EJB 的容器 WebLogic、WebSphere 等，在系统运行性能上，有很大优势。同时，由于使用了 Spring 的事务管理容器和 Hibernate 的架构，保证了系统多用户访问时，数据库连接池稳定性和数据库并发操作的可靠性，以及事务操作失败的回滚能力。

可维护性。J2EE 三层架构在模型层、视图层和控制层之间划分责任，减少了代码的重复度，并使应用程序维护起来更简单。应用 Struts + Spring + Hibernate 构建的系统，降低了各层之间的接口的耦合度，如果系统的表现层需要改用其他框架，其业务逻辑层或持久层不会受到直接的影响；业务逻辑层的某个组件的执行逻辑的修改或添加新的组件代码也不会影响到 Struts 的表现层。可以说，轻量级 J2EE 系统为开发者奠定了包容变化的基础。

可重用性。Spring 是一个开放的框架，开发者可以选择仅仅使用它任何一个独立的部分，或向其架构中添加所需的其他框架，所以在 Spring 框架中，EJB 的使用也是开发者的一个实现选择。Hibernate 框架将数据源的物理实现细节与其用户完全分离，它保证了在底层数据源变化的时候，向用户提供的接口不发生改变。因此，Hibernate 的应用使系统可以适应多种数据存储介质。

5 结束语

J2EE 应用开发最困难的就是架构选型的问题，如何在纷繁复杂的架构模式中选择一种适合自身系统应用需要的架构是一个困难的过程。敏捷建模的思想给了我们很多启发，其中主张简单、包容变化、快速反馈等原则可以帮助我们在系统架构时做出正确的选择。

参考文献

1 Craig Walls, Ryan Breidenbach 著，李磊、程立、周悦红译，Spring in Action. 中文版，北京：人民邮电出版社，2006. 3：4 – 35，139 – 144，311 – 313.

2 王卫平、王松涛、王茗名，一种基于 J2EE、Spring、Hibernate 的轻量级 EAI 构架，计算机系统应用，2005 年第 11 期：38 – 41.

3 Scott W. Ambler 著，张嘉路等译，敏捷建模 极限编程和统一过程的有效实践，北京：机械工业版社，2003. 4：25 – 33.

4 Mark Eagle http://www. onjava. com/pub/a/onjava/2004/04/07/wiringwebapps. html.

5 孙卫琴编著，精通 Hibernate：Java 对象持久化技术详解，北京：电子工业出版社 2005. 5：1 – 45.