文章编号：1005-8451（2014）02-0029-03

基于OSGi和用户权限的客户端集成研究

王胜东
（中铁信弘远（北京）软件科技有限责任公司，北京 100038）

摘 要：铁路调度信息系统包括多个业务场景，每个场景的应用架构涵盖浏览器/服务器（B/S）模式和客户端/服务器（C/S）模式，将不同场景应用在客户端进行集成是当前的主要问题。OSGi技术为客户端集成提供了通用的解决方案。开发OSGi插件进行既有应用的集成，在插件启动类中进行系统界面加载；根据用户对系统集成的需求，对用户权限进行统一的管理和分配；根据用户权限，OSGi框架进行应用的启动加载，从而实现既有应用系统的整合，并且按照权限内容实现功能定制。

关键词：铁路调度信息系统；OSGi；访问控制；客户端集成
中图分类号：U284.5：TP39 **文献标识码：**A

Research on client integration based on OSGi & user privileges

WANG Shengdong
(Sinorail Hong Yuan(Beijing) Information Software Development Co., Ltd., Beijing 100038, China)

Abstract: There were several scenarios in the Transportation Dispatching Information System, each scenario had own application frameworks which covered from browser/server to client/server. The different scene in the client application integration was the key issue. OSGi technology provided a general solution for the client integration. The OSGi bundles were developed to integrate applications, interfaces of existing system were loaded in the bundle activator. User privileges were managed and assigned according to requirements, the union client application platform was launched by OSGi framework and customized by user privileges.

Key words: Transportation Dispatching Information System; Open Service Gateway Initiative(OSGi); access control; client integration

铁路调度信息系统负责铁路运输业务的调度信息化管理，支持运输业务中的资源调度工作。运输调度业务系统管理范围复杂，需要多项资源进行综合协调管理；按照调度业务事前计划、事中监控、事后统计的方式，将调度业务场景分为计划场景、指挥场景、统计场景、非常规事件处理时的命令场景，在这样4个场景情况下，就可以进行整体应用的划分和规划。在当前的业务系统中，不同的应用场景采取了不同的应用架构模式，对于调度命令、生产指挥等需要进行及时响应场景，采用C/S模式；对于统计分析、计划报表等数据统计显示场景，采用B/S模式。

调度信息系统应用是基于Java语言开发的，OSGi为系统的集成提供了统一的解决方案。通过OSGi插件开发，进行已有系统的集成；使用OSGi生命周期API函数，可以在已开发插件中进行既有系统的加载，从而充分利用既有系统代码。对于不同的用户，其系统集成权限是不一致的；基于不同的用户需求，进行用户权限的统一管理，实现系统的统一集成。

收稿日期：2013-10-25
作者简介：王胜东，工程师。

1 OSGi的基本原理和集成优势

OSGi委员会成立于1999年，当前最新版本OSGi规范为2012年发布的第5版。OSGi定义了动态的Java模块系统，利用OSGi可以更好的控制代码结构，进行模块的生命周期管理，降低系统的耦合性。OSGi框架如图1所示，安全层基于Java 2安全机制构建，并且进行了某些空白的补充。模块层定义了Java模块化模型，弥补了Java部署模型的一些缺点，模块层严格定义了插件之间共享或者隐藏Java包的规则。生命周期层为插件提供了生命周期API函数，定义了插件的安装、更新、卸载等行为。服务层为Java插件开发者提

供了动态的编程模型，实现了服务的实现与接口的分离。

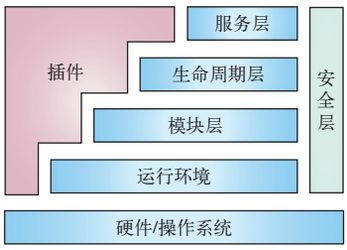

图 1 OSGi框架结构

Java 语言本身是有模块化概念的，但是 Java 语言的模块化基本上是基于面向对象语言为基础的，一般只能使用访问控制符 (如 public、private) 来进行访问权限的设定；一旦设置为 public，就会暴露过多的实现细节。当多个 JAR 文件一起工作时，常常会引起“类路径地狱”问题，主要是某一个 JAR 文件中的类与另外一个 JAR 文件中的类版本不一致导致的；在大规模系统开发中，诸如日志记录或 XML 解析等经常存在不同版本。而且标准的 Java 环境很难实现动态的插件升级管理。OSGi 架构对上述问题提供了很好的解决方案，使用模块化机制对应用进行了逻辑、物理的封装，使用生命周期函数进行插件的动态管理。

2 基于角色的用户权限管理

采用基于角色的权限管理，其权限分配模型如图 2 所示。

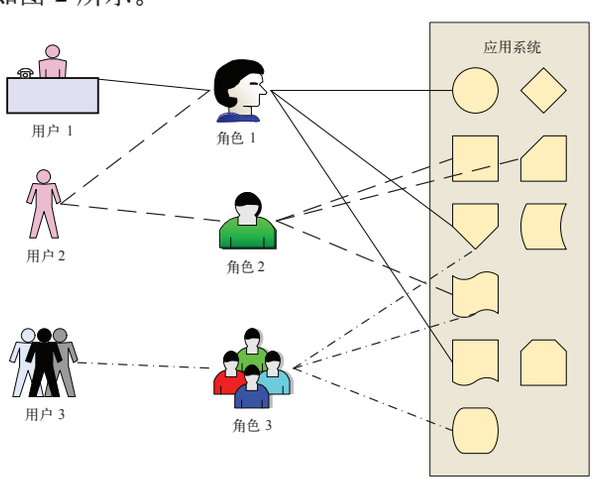

图 2 基于角色的权限分配

基于角色的用户授权可以解决用户权限的集中管理问题，如图 2 的角色 1、角色 2、角色 3，在实际的调度业务系统中，经常会根据调度员负责的业务工作，将具体工作设置为具体岗位，对每一个具体岗位的应用权限进行统一分配，把用户与具体角色进行关联实现用户授权。还有一种情况如图 2 中的用户 2，调度业务中常常存在轮换岗位的需求，需要在不同的岗位之间进行工作调换，这种情况需要对一个用户分配多个角色，在用户登录系统时进行角色选择，再进行用户权限分配。

3 用户权限分类整理

在进行系统集成过程中，将用户权限分为两类，用户界面权限和应用功能权限。用户界面权限是指在系统加载的过程中，需要加载的界面内容；应用功能权限是指在系统使用过程中，具体的业务操作范围。

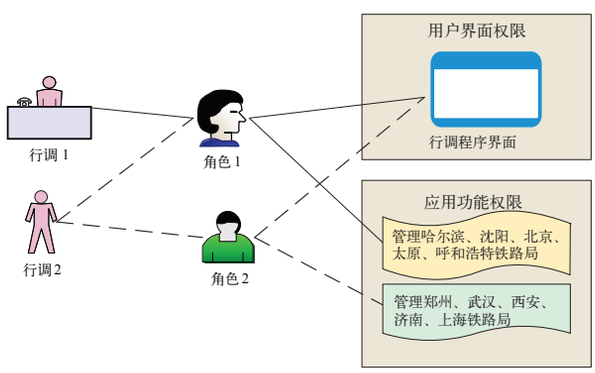

图 3 用户权限分类

如图 3 所示，类似岗位的用户如行调 1 和行调 2，因为具体业务操作完全相同，需要加载相同的界面；但是其管理范围不一样，行调 1 管理哈尔滨、沈阳、北京、太原、呼和浩特 5 个铁路局，而行调 2 管理郑州、武汉、西安、济南、上海 5 个铁路局。这是两类不同类型的权限内容，用户界面权限在系统开发设计阶段进行设定，在后期系统运行阶段调整范围不大，只有在系统更新或升级时才进行改变。应用功能权限需要根据业务的变动进行相应的改变，完成对于业务功能调整的支持。进行权限的分类管理，可以方便系统的集成和功能内容的调整。

4 集成插件实现

根据用户的界面权限，进行相应的 OSGi 插

件开发。将插件开发与应用开发分离，降低两者之间的关联，采用了 Java 反射技术进行关联分拆。

具体的实现代码如图 4 所示，在反射技术的支持下，插件开发时不知道需要加载页面的具体类名，可以在系统运行时进行动态的配置和调整，降低系统之间的耦合性，保持插件开发与应用开发的独立性。

```
/**
 * 通过反射获取Swing Composite 类
 *
 * @param classPath
 * @return
 */
public static Object getSwingComposite(String classPath) {
    Class<?> clazz = null;
    Object composite = null;
    try {
        clazz = Class.forName(classPath);
        Object obj = clazz.newInstance();
        return obj;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return composite;
}
```

图 4　使用反射获得Swing Composite类

在客户端框架启动过程中，首先会根据用户名、密码进行权限加载，用户界面权限在插件中进行初始界面的加载，应用功能权限包含了各个子系统初始化条件和信息，可以在初始化过程中根据这些权限进行系统初始化情况的设定。

对于 B/S 应用模式，可以采用 SWT 中自带的 org.eclipse.swt.browser.Browser 进行 Web 页面加载，从而方便多种模式的应用集成。B/S 模式的认证方式支持两种情况，由于在客户端框架中进行 Browser 类的 URL 设定，所以在设定过程中可以使用 HTTP GET 方式进行用户名、密码的传入，在相应的 B/S 应用中，就可以按照传输的认证信息进行用户权限的设定；如果采用数字证书认证方式，只要在用户客户端已经安装了有效的数字证书，既可以通过系统认证。

在进行插件开发过程中，需要对插件的权限进行统一的管理，搭建了插件管理系统，其界面如图 5 所示。在插件管理系统中，将插件的启动类统一存储在数据库中，通过管理界面进行管理，实现动态用户界面加载。

5　不同集成需求的实现

根据系统集成的设计思想，进行了客户端集成框架的开发；OSGi 框架采用了 Equinox 作为具体实现，客户端应用集成了 C/S 模式的 Java

图 5　插件管理界面

Swing 和 SWT 客户端界面，同时集成了 B/S 模式的 Web 页面。集成场景分为命令、计划、指挥和统计 4 个场景，每一个场景下的插件可以进行动态的配置，其集成界面如图 6 所示。

图 6　应用系统集成界面

利用 OSGi 进行集成时，其能够加载的插件必须事先写在配置文件中，针对不同的用户，最终加载的插件个数不同，提前需要安装的插件都相同。为了方便系统升级和新功能添加，必须预留一部分插件，插件的使用个数和预留个数必须进行一个合理的折中选择。

6　结束语

本文对 OSGi 基本原理进行了介绍，讨论了 OSGi 进行系统集成的优势。基于角色进行了用户权限的配置管理，根据具体的业务场景和权限管理需求，进行了用户权限的分类整理。根据铁路调度信息系统所涵盖的命令、计划、指挥、统计场景进行了系统分类，利用 Java 反射模式进行了插件开发与应用开发的分离，降低了系统的耦合性，同时兼容了浏览器 / 服务器模式和客户端 / 服务器模式，实现了客户端应用系统的集成。

责任编辑　陈　蓉