开放式插件系统研究

北邮网络教育学院 苏占玖

主要内容

- 插件系统概念
- 插件系统主要类型的架构和特点
- 基于Java Portal标准的插件框架
- 基于OpenAPI的插件系统
- 基于OSGI的插件系统
- 选择插件架构

插件系统概念

- 也称为扩展，是一种遵循一定规范的应用程序接口编写出来的程序，用来构建高扩展性的软件系统
- 对于系统来说并不知道插件的具体功能，仅仅是为插件留下预定的接口，系统启动的时候根据插件的配置寻找插件，根据预定的接口把插件挂接到系统中
- 是软件代码的一种高级封装和复用形式
  - 控件组件是面向开发者的
  - 插件是面向最终用户
- 常见如**Firefox**插件、**IE**插件、**Eclipse**插件

开放式平台引入插件的必要性

- 软件系统需求多样，开放式平台要求高扩展性。
- 用户可选择使用或不使用哪些插件
- 第三方开发方便，在系统发布后进行功能扩充，不必重新编译，只需遵循接口规范开发新的插件。
- 利于模块化的开发方式。可以开发强大的插件管理系统，这样可以不需修改基本系统，仅仅使用插件就能构造出各种各样不同的系统，真正的可组装。

主要类型

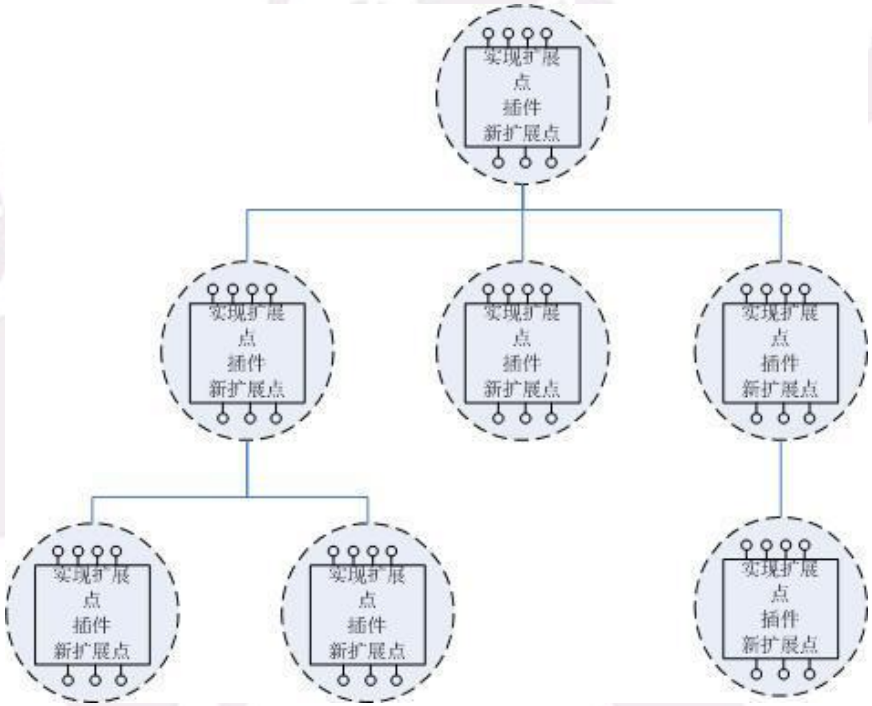
- 微内核 级联树形结构
  - **Eclipse、Atlassian's JIRA & Confluence**
- 巨内核（管理容器）并联结构
  - **Sakai等Java Portal的形式、校内网等SNS站点**

微内核

- 特点：扩展点
  - Eclipse的插件结构是由父插件管理子插件，插件之间由扩展点连接，最终形成树形的结构。Eclipse是众多扩展点和扩展共同组成的集合体。
- 界面呈现
  - 界面呈现方式由提供扩展点的父插件来决定。
- 插件交互
  - 插件之间的交互通过扩展点实现。父插件调用子插件实现的扩展点来触发子插件的动作。扩展点决定交互模式。
- 依赖关系
  - 在配置文件中指定插件运行需要依赖的插件，在装载过程中会按照依赖的关系顺序来装载。
- 延迟加载
  - 只有在调用执行动作的时候才会将真实的插件对象加载进来。由于在配置文件中已经描述了插件的一切信息，所以在不装载插件时，也可[illegible]出来。

微内核扩展点形成的系统结构

- 此类系统插件用扩展点的机制连接起来，形成如图所示的系统结构。插件必须实现扩展点，以此插入到系统中，新增扩展点并不是必须的，但只有新增了扩展点的插件才可以被别人扩展。



巨内核

- 界面呈现
  - 由系统运行框架决定，框架来决定支持几种显示模式。通过配置文件可以在现有的模式之上随意组合形成复杂的界面。在这个过程中插件并不关心自己被放在什么地方，或者以什么形式呈现。
- 插件关系
  - 由系统框架（关系管理器）统一管理，插件本身不需要维护交互信息，只有在需要的时候才会从关系管理器取得。
- 依赖关系一般没有管理。

对比

- 微内核
  - 只有很小的运行框架核心，职责和功能简单，构成系统的几乎都是插件。由父插件负责管理子插件。可扩展性更强。内核可以比作一粒种子，整个系统为级联的树形结构。
- 巨内核
  - 系统运行框架比较复杂，它需要管理所有的插件，管理界面布局策略、导航、插件代理等职责。好处在于插件非常简单，只需要将业务部分用统一的接口公布出来就可以，开发时不需要考虑和其他插件的关系。插件为并联结构横向扩展。

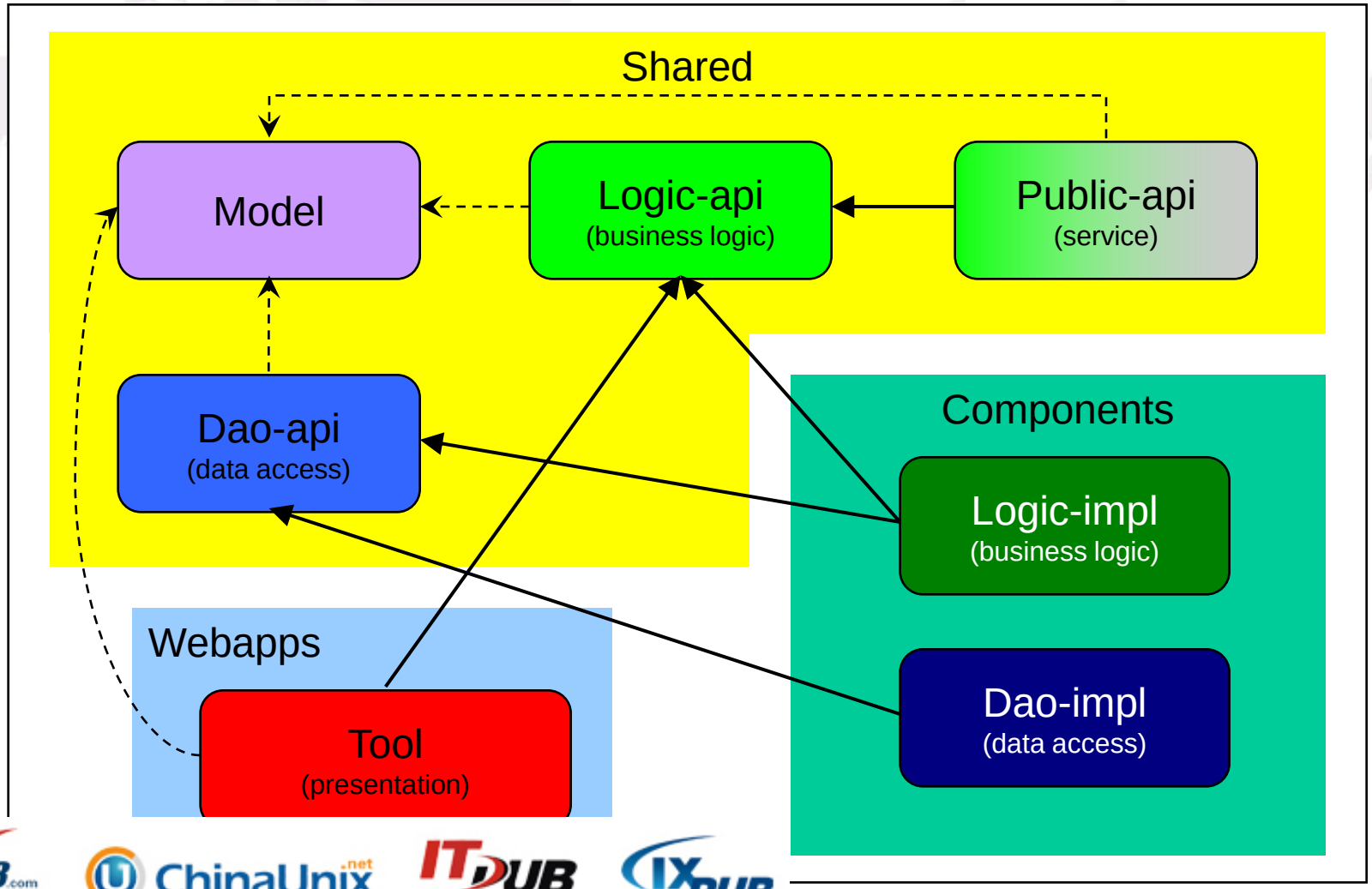
基于Java Portal标准的插件框架

- Portal
  - 提供包括内容聚合、单点登陆、个性化定制和安全管理等服务
- JSR 168和JSR 286
  - 本地Portlet规范，是部署在容器内用来生成动态内容的 Web 组件
- WSRP
  - 是Web Services for Remote Portlets规范，它定义了门户网站的远程 Web 服务。通过 Web Service 将远程内容抓取到本地，最后通过本地内容聚合引擎展示出来。
- 适用于企业信息门户系统，多系统整合

Sakai的插件框架

- 基于Portal标准框架
- 基于Tomcat5.5和Spring进行改造，使提供Portlet的所有webapp都可以发布公共service API。
- 应用分为Shared/Components/Webapps三个部分

Sakai应用结构图



Sakai插件关系

- 一个Webapp即为一个插件，一个插件可提供多个portlet
- 通过改造使插件之间可以Public API的方式通信，使各插件的业务组件可以重用。
- 运行时插件间的依赖关系没有配置管理
- 一般的Portal系统各Webapp间的组件调用如果不通过特殊处理就不容易做到。基本上就是内容聚合，复杂的业务视图比较难以呈现。

Sakai插件布局呈现

Sakai插件注册管理

- 通过XML配置（webapp/tools/sakai.melete.xml）

```
<registration>
    <tool
        id="sakai.melete"
        title="Modules"
        description="Modules - ... ...">
        <configuration name="optional_properties"
    value="true" />
        <category name="course" />
        <category name="project" />
    </tool>
</registration>
```

- 每个Project site或Course site都可以单独选用不同的插件。
- 可配置插件显示名，定制插件呈现位置（右侧内容区或弹出）
- 可定制Portal页上小窗格布局和大小

基于OpenAPI的插件系统

- OpenAPI是指WEB应用通过HTTP协议对第三方开发组织或个人开放的应用编程接口。
- 专有API制定出来主要是为了制定者本身提供应用开发接口的目的，大部分都是专有API。例如facebook、taobao。
- 标准API，由大企业制定，或被标准化组织采纳，或已成事实标准。如google opensocial、RSS、Atom。

- 大部分采用标准REST和类REST形式
- 也可以以web service形式提供
- 数据格式RSS/Atom/JSON/XML/HTML/RSS或Atom扩展/自定义格式等基于文本的

- 可同时开发Client端和Server端插件
- 天生的分布式插件系统
- 松散组织管理的插件，只需注册一下，更为开放的甚至不需要注册。
- 插件可独立为外部应用，也可简单嵌入系统页面
- 可mashup多个网站的OpenAPI
- 一般需要配合安全认证系统授权
- 插件管理和实现比较简单适用于各种大型互联网应用（搜索引擎、SNS、资讯、电子商务……）

校内API

- 可选“外部应用”或“安装到校内”
- 运行状态可选iframe形式或xnml形式
- 各种语言版本的Client API
- 支持OpenSocial、Gadget小工具

插件界面呈现

- **xnml方式的应用，所有页面请求通过校内的API服务器在后台转发给具体应用。包括所有参数和当前用户的id、sessionid、api key。**
- **iframe的应用就简单多了，直接生成一个iframe，指向插件应用的起始地址，同时传参：当前用户的id、sessionid、api key。**

- xnml模式返回的内容是xnml格式，由校内的API服务器接收并对内容进行解析，然后显示在内容区的层里，展现给用户。为了防止一些非法操作，xnml规定了哪些html语法是合法的，xnml还定义了一些他自己的xml语法，可以在页面上直接显示出用户的姓名头像链接好友等等信息，而无需在程序里通过API调用去获取。
- iframe模式返回的页面内容为html直接显示在iframe里，跟普通的网页一样。

插件注册管理

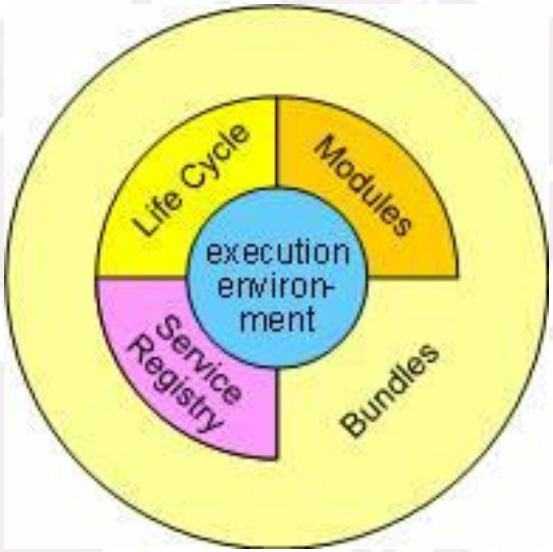
- 由网站的开发用户在系统内注册和配置
- 不负责管理插件之间的依赖和调用关系。插件本身是独立部署的系统，开发自由度很大。

基于OSGI的插件系统

- 成为JSR291 的规范标准
- 分布式OSGI (CXF)
- 微内核插件体系结构
- 被越来越多的企业应用系统采用，Eclipse/Websphere/Weblogic/Spring/Alipay
- 实现：Equinox、Felix

核心框架

- L0: 执行环境
- L1: 模块管理
- L2: 生命周期管理
- L3: 服务注册



OSGI特点

- 实现热插拔
  - 通过安装新的 Bundle、更新或停止现有的 Bundle 来实现系统功能的插拔
- 可动态改变行为
  - 通过动态切换Bundle实现
- 稳定、高效
  - 微内核机制有效提高了系统的稳定性
  - OSGI 的动态性、延迟加载原则保证了系统运行的高效，只有在请求发生时 OSGI 才去完全加载、启动相应的 Bundle、Service。

- 利于模块化设计开发
  - 基于 OSGI 的系统采用规范的模块化开发、部署方式构建系统，可积累
- 面向服务的组件模型设计思想
- 可扩展的设计
  - 扩展点
  - 声明服务

OSGI Bundle

- 在 OSGI中所有模块的部署都必须以 Bundle的方式来进行部署
- 关于Bundle的所有信息都在Meta-inf目录下的MANIFEST.MF中进行描述
  - Bundle的名称、描述、开发商、classpath、需要导入的包以及输出的包等等
- Bundle 通过实现 BundleActivator 接口去控制其生命周期
  - 在 Activator 中编写 Bundle启动、停止时所需要进行的工作，同时也可以在 Activator中发布或者监听框架的事件状态信息，以根据框架的运行状态做出相应的调整
- Bundle是个独立的概念，在OSGI框架中对于每个Bundle采用的是独立的classloader机制
  - 通过导入导出包和发布Service组件和其他Bundle协作
- 可通过系统中预设的扩展点来扩充系统的功能

的扩展点来扩展该 Bundle的功

Spring-OSGI

- Spring Dynamic Modules实现Spring和OSGI两大流行框架的整合
- 更好的模块间的应用逻辑隔离，这些模块具有运行时强制的模块边界
- 同时部署同一个模块（或库）的不同版本
- 动态发现和使用系统内其他模块提供的服务
- 在运行着的系统中动态地安装、更新和卸载模块
- 使用 Spring 框架提供在模块内部和模块之间进行实例化、配置、整合组件的能力
- 对于企业级开发人员来说是一个简单和熟悉的POJO编程模型，而且又可以利用 OSGi平台的动态特性

OSGI应用于企业应用

- OSGI原来用于嵌入式应用、桌面应用，现在越来越多的WEB应用架构开始OSGI化。
- 更能满足开放式系统高可扩展性和灵活性要求，满足企业系统需求多样性的要求
- 使界面呈现方式更加灵活多变易扩展满足复杂的业务视图需求

选择插件架构

- **Portal**适用于企业门户的信息聚合
- **OpenAPI**更适用于运营中的用户规模庞大互联网应用
- 以复杂业务处理为核心的企业应用更适合使用**OSGI**，来满足复杂多变的业务需求。

组合应用

- **(OSGI+OpenAPI)*SOA** 构建更具扩展性的强大的分布式系统
- 构建开放式远程教育系统

谢谢各位！

北邮网络教育学院
苏占玖
suzhanj@gmail.com