

插件简述

1、扩展插件概述

1.1、扩展插件是什么

Smartbi 插件又称扩展包，是用于扩充或改造 Smartbi 产品功能的一个插件包，开发完后一般打包成*.ext文件。

它实际和 java web 项目类似（war包），里面可以有图片、js、html、css、java、servlet、filter等，只是结构会遵循 Smartbi 的规范，并且多了一些特殊的配置文件，其中有个不同就是 java web 项目WEB-INF目录下的内容对应 Smartbi 扩展包下的META-INF目录；而 web.xml 文件则对应扩展包中的 extension.xml，所以在扩展包中，servlet和filter是需要配置在extension.xml 文件中（但配置结构不变）。

文档目录：

- 1、扩展插件概述
 - 1.1、扩展插件是什么
 - 1.2、扩展插件的特点
 - 1.3、扩展插件使用场景
 - 1.4、扩展插件原理
 - 1.5、如何开发及调试
- 2、术语及定义
- 3、常见问题
 - 3.1、为什么要使用扩展插件
 - 3.2、如何在Smartbi中加载扩展插件
 - 3.3、Smartbi是否可以同时加载多个扩展插件
 - 3.4、扩展插件的加载策略是什么
 - 3.4.1、扩展策略
 - 3.4.2、扩展插件的声明方式
 - 3.4.3、扩展插件的加载顺序
 - 3.5、在WebSphere部署扩展插件的问题

1.2、扩展插件的特点

- 扩展插件通过 Smartbi 加载执行。
- 扩展插件中的文件会覆盖 Smartbi war包中相同路径下的同名文件。
- 扩展插件可以调用 Smartbi 的内部对象。
- 扩展插件可以通过扩展点对 Smartbi 进行定制开发，能够把新功能无缝集成到系统中。

1.3、扩展插件使用场景

通常会用扩展包实现客户现场要求的特定需求，譬如：

- 1、新增独立的功能，譬如新增个jsp、servlet之类，除了可能调用smartbi接口外，其他都独立存在的功能，这类一般不受smartbi影响，只要按开发java web项目思路开发即可。
- 2、在产品内置功能模块界面增加、删除或修改某个功能点，譬如在灵活分析上增加导出到ftp功能之类，这类会依据smartbi有没有提供修改入口，具体问题具体分析，请见扩展包原理。
- 3、用户单点登陆（比如客户现场已经有统一登陆平台，这时候就需要这个功能）之类产品提供了[服务端扩展点](#)功能。
- 4、具体示例请见 [插件开发示例库](#)。

1.4、扩展插件原理

要理解扩展包为什么能做那些之前，建议先快速看下[扩展包的目录结构](#)。如果能想象扩展包是一个类war包的东西，只是 Smartbi 通过插件机制将其作为smartbi.war包的补充， 插件补充方式是：

- 1、可以覆盖产品同名文件，达到改造产品功能目的：如图片、js、html、css、jsp等，扩展包中的文件会覆盖与smartbi.war中同路径下的同名文件，譬如扩展包名.ext\vision\a.js是会替换smartbi.war\vision\a.js文件的，具体示例请见[替换Smartbi文件](#)，至于多个扩展包之间加载顺序请见 [如何设置扩展包加载优先级](#)。
- 2、新增jsp、servlet、filter等实现个性需求。
- 3、利用同一个 js 文件中多个相同的同名方法，最后一个会生效的原理，smartbi 提供了使用 **同名.js.patch** 方式修改产品内置同名 js 脚本功能。在加载 js 脚本时，会遍历所有扩展包，然后将扩展包中同路径及同名的 .js.patch 文件内容附加到产品内置 js 文件的后面，最后返回的是合并后的 js文件，具体请见 [如何修改Smartbi JS文件](#)，这是最常用的扩展方式，通常如果需要服务端实现部分逻辑的话，就要结合自定义module实现特定功能。
- 4、使用系统提供的各种前端插入点修改产品特定功能点，也就是在扩展包中的 ConfigurationPatch.js 文件中配置插入点，达到修改功能的目的。ConfigurationPatch.js 加载原理是：
 1. 客户端请求 smartbi；
 2. 系统初始化时会加载 Configuration.js 配置文件；
 3. 加载 Configuration.js 时，系统会遍历所有扩展包中的 ConfigurationPatch.js，将其合并到 Configuration.js 中并返回到前端，这里的合并是按 json 合并方式合并，也就是两个文件中的 json 对象按 smartbi 指定的合并规则，最后合并为一个 json 对象返回，前端在对应的插入点会读取这些配置项实现相应的功能。
- 5、自定义验证类等服务端扩展点，这都是接口性质。

1.5、如何开发及调试

请见[插件开发快速入门](#)，具体涉及项目需求如何开发调试建议一定看下 [Smartbi前端框架介绍及调试定位入门](#)。

[插件开发教程](#) 以常见的可能碰到的知识点去组织，日常工作中可能遇到的各种问题，看了一遍也加深对扩展包的理解。

2、术语及定义

术语	定义
扩展点	为了能够在不改动Smartbi代码的情况下，让新开发的插件无缝集成到系统中，Smartbi系统框架提供了大量的功能插入接口，这些接口即称之为“扩展点”。
一级模块	“浏览”、“定制”、“管理”、“个人设置”、“帮助”等功能模块入口。
二级模块	一级模块的子模块入口。
任务面板	Smartbi 模块主界面中大图标展现的快捷功能操作入口区。
资源树	以树的方式展现Smartbi资源。
知识库	用于存储Smartbi系统资源信息的数据库我们称之为知识库。
知识库对象	使用Hibernate运行机制，知识库表映射到一个POJO实体类，该实体类指知识库对象。
知识库升级	Smartbi使用升级类代码对知识库版本进行维护，当扩展插件中需要修改知识库对象时就必须添加相应的升级类进行知识库版本的维护。
Java查询	指基于Java数据源中Java查询对象作为查询源的一种业务查询。

3、常见问题

3.1、为什么要使用扩展插件

- **从项目角度上看：** 把项目定制化的需求做成扩展插件，这样能够最大限度地控制项目化的需求对产品的影响，控制不同项目之间相互影响，项目需求响应的及时性也能得到保证。
- **从产品开发角度上看：** 把产品新功能模块的开发做成扩展插件，这样可以有效避免新功能开发对产品的影响，从而最大限度的保证产品的质量。

3.2、如何在Smartbi中加载扩展插件

- **项目定制化需求开发的扩展插件：** 启动Smartbi服务器，访问http://server_ip:port/smartbi/config。在config界面上，指定扩展插件的存放路径，比如“C:/SmartbiExtension/”目录，然后保存配置。
- **研发内部开发的新功能扩展插件：** 修改 smartbi.war/WEB-INF/extensions/extensions.list 文件，每一行为一个扩展插件路径的声明。除了指定 .ext 扩展包文件，也可以声明加载指定目录下的插件，比如“..\ExtensionManager.SandBox\src\web\“，这时要指定到src/web目录。

3.3、Smartbi是否可以同时加载多个扩展插件

Smartbi 里面允许同时加载多个扩展插件。把多个插件放到同一个目录中，然后通过config配置界面把扩展插件路径指向该目录，或者在extensions.list文件中添加多个扩展插件路径。具体配置方法，请参见“3.2、如何在Smartbi中加载扩展插件”。

3.4、扩展插件的加载策略是什么

3.4.1、扩展策略

Smartbi 服务器启动时，会自动加载所有扩展插件，并调用相关的激活方法。若扩展插件中存在与smartbi.war中同名的文件时，会优先读取扩展插件中的文件，当多个扩展插件中存在同名文件时会以第一个扩展插件优先，当然我们也可以设置各个扩展包的加载优先级，但ConfigurationPatch.js 会自动合并，不存在覆盖问题。

3.4.2、扩展插件的声明方式

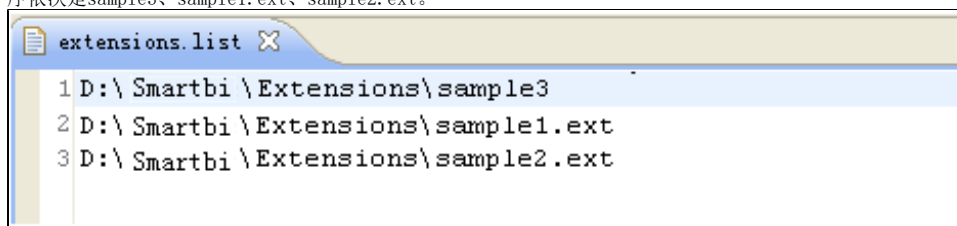
产品允许采用两种方式来设置扩展包的加载路径。

- 通过 smartbi.war/WEB-INF/extensions/extensions.list 文件进行声明。
- 通过 Smartbi 配置界面 config.jsp 进行声明。

3.4.3、扩展插件的加载顺序

- 首先加载“smartbi.war/WEB-INF/extensions/extensions.list”文件中声明的扩展包，再加载Smartbi配置界面中指定的目录。

- 在extensions.list文件中，每一行为一个扩展插件路径的声明，系统按顺序加载各个扩展插件。例如下面声明了三个扩展插件，系统加载顺序依次是sample3、sample1.ext、sample2.ext。



- 通过Smartbi配置界面声明扩展插件的存放路径时，该路径中可以保存多个以.ext结尾的扩展插件或者包含多个扩展插件的目录。其对扩展插件的加载顺序是按扩展插件的名字顺序来进行加载，假如该目录中存放了三个插件，则系统依次加载sample1.ext、sample2.ext、sample3。

3.5、在WebSphere部署扩展插件的问题

当使用 WebSphere 作为服务器时，可能会出现无法加载扩展插件内的图片、Servlet的情况，在出现此情况下，请按以下步骤进行修改：

1. 展开服务器 → 应用程序服务器。
2. 点击 **server1**。
3. 点击 **Web容器设置** → **Web容器**。
4. 点击 **定制属性**。
5. 点击 **新建**。
6. 输入名称：com.ibm.ws.webcontainer.invokefilterscompatibility，其值为 true。
7. 保存并重新启动应用服务器。