知识库升级

1 概述

Smartbi 使用代码进行知识库版本的维护，当扩展包中需要往知识库中添加库表、修改库表或自动插入初始化数据时，就必须添加相应的升级类进行知识库版本的维护。

2 知识库升级类的编写规范

1、在扩展包中必须通过applicationContext.xml文件声明至少一个组件（见 自定义Module），该组件的类名为类似于 **smartbi.extension.demo.KnowledgeBaseUpgrade**，并必须将此组件向framework模块注册，例如：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN" "http://www.springframework.org/dtd/spring-beans-2.0.
dtd">
<beans>
	<bean id="framework" class="smartbi.framework.Framework" factory-method="getInstance">
		<property name="modules">
			<map>
				<entry><key><value>KnowledgeBaseUpgradeModule</value></key><ref bean="
KnowledgeBaseUpgradeModule" /></entry>
			</map>
		</property>
	</bean>
	<bean id="KnowledgeBaseUpgradeModule" class="bof.ext.KnowledgeBaseUpgrade.
KnowledgeBaseUpgradeModule" factory-method="getInstance">
		<property name="daoModule" ref="dao"/>
	</bean>
</beans>
```

2、升级类固定在module所在包的upgrade包中，上述示例中升级包名为 smartbi.extension.demo.upgrade。

3、编写继承 smartbi.repository.UpgradeTask类的升级类，以UpgradeTask开头，后接New或者版本号（譬如UpgradeTask_0_0_1），第一个版本必须命名为UpgradeTask_New；

4、升级类有三个方法需要实现：

a. public abstract boolean doUpgrade(Connection conn, DBType type); 升级类的具体实现，Smartbi会调用该方法执行升级任务，返回true表示升级成功，返回false表示升级失败。 由于数据库类型存在差异，创建表、添加字段等SQL的语法不相同，因此建议使用 smartbi.repository.UpgradeHelper类中的方法生成相应的SQL语句，避免不同类型的知识库迁移时存在问题，同时如果使用了Statement、PreparedStatement记得关闭。
b. public abstract String getNewVersion(); 声明当知识库升级成功以后当前组件的知识库版本号，格式应该为"0.0.1"。Smartbi会根据返回的版本号查找下一个升级类继续下一版本的升级。 例如：当UpgradeTask_New.getNewVersion()返回"0.0.1"时，Smartbi会试图加载UpgradeTask_0_0_1升级类，若加载成功并执行升级成功，Smartbi会继续试图加载UpgradeTask_0_0_1.getNewVersion()返回版本号"0.0.2"的升级类UpgradeTask_0_0_2，直到查找不到更高版本的升级类为止
c. public String getDate(); **必须实现**，因为自从 "2019-07-01 00:00:00"后，产品升级类会先按此方法返回的时间排序，其次是模块版本号顺序，如未实现此方法，默认返回的是2019-07-01 00:00:00，这可能导致你新写的升级类比产品新写的升级类先执行，加入此升级类对产品模块升级类或其他扩展包升级类有依赖，就会出现启动时升级失败的状况。

注意：升级类执行完毕后会在知识库的 “t_systemconfig” 表中创建升级记录，key值为module所在包的包路径。

3 getDate说明

getDate返回值为String类型，需要返回正确时间格式的字符串，例如”2019-7-1 00:00:00”。

getDate是2019-7-1 00:00:00才新增的方法，在此之前都是按照模块顺序执行升级类，这可能会引起一个问题。

假设存在两个模块A和B，A模块比B模块优先级高：

B模块中有一个升级类：create table c ……

A模块中有一个升级类：alter table c …

由于A模块优先级更高，会先执行A模块的升级类，此时会导致升级失败，而模块的优先级不好改变，因此引入了新的决定因素——时间，只要B模块升级类返回的时间小于A模块升级类返回的时间，即可使得B模块创表升级类比A模块改表升级类优先执行。

4 升级类执行顺序

假设有两个模块A和B，A模块比B模块优先级高：

A模块中有三个升级类，分别是A1（date：2020-8-27 00:00:00，version：0.0.2），A2（date：2020-8-29 00:00:00，version：0.0.3），A3（date：2020-8-30 00:00:00，version：0.0.4）

B模块中有两个升级类，分别是B1（date：2020-8-28 00:00:00，version：0.0.2），B2（date：2020-8-30 00:00:00，version：0.0.3）

默认按照时间顺序升级，若时间相同则按模块顺序升级，因此上述例子执行顺序为，A1（时间优先）->B1（时间优先）-A2（时间优先）->A3（同时间，模块顺序优先）->B2

ps：A1（date：2020-8-27 00:00:00，version：0.0.2）为升级类名称（date：getDate返回值，version：getNewVersion返回值）

5 示例说明

（1）单个模块内按getNewVersion确定执行顺序示例

在升级类"UpgradeTask_New.java"中创建了一个知识库表，该表与"KnowledgeBaseObject.java"中的描述相对应。另外还有两个升级类"UpgradeTask_0_0_1.java" 和 "UpgradeTask_0_0_2.java"，其中前者仅仅是修改了模块的版本号，后者则往上述知识库表中增加一个字段。总共有3个升级类，该模块的当前版本为最后一个升级类(在此为"UpgradeTask_0_0_2.java")的"getNewVersion()"方法的返回值(源码中该值为"0.0.3")。测试链接为：http://localhost:18080/smartbi/vision/demo/knowledgeBaseUpgradeModuleDemo.html，测试界面如下：

示例代码下载：KnowledgeBaseUpgrade

（2）多模块根据getDate和getNewVersion确定执行顺序示例

在B模块的升级类"UpgradeTask_New.java"中创建了一个知识库表，该表与"KnowledgeBaseObject.java"中的描述相对应，并且该升级类返回日期为"2019-07-01 00:00:00"。

在A模块的升级类"UpgradeTask_New.java"中往上述知识库表中增加一个字段，并且该升级类返回日期为"2019-07-02 00:00:00"。

首先执行返回时间较小的B模块升级类，创建表” t_ext_knowledge_base_object”，接着执行A模块的升级类增加一个字段。

示例代码下载：KnowledgeBaseUpgrade2

6 视频教学

视频教学点击下载：知识库升级