https://blog.csdn.net/u010410135/article/details/70231103

**利用AOP为Spring Data Jpa的接口Repository添加全局自定义过滤**

2017年04月18日 20:18:35 [行走的安子](https://me.csdn.net/u010410135) 阅读数 6145

 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/u010410135/article/details/70231103

安子我最近在做一个权限控制模块，大抵是有权限的人能从数据库中查到的信息比权限低的人查的多一些。工程用的是JPA hibernate，想做一个过滤器来设置条件过滤。

一开始安子我想过直接用aspectj来做过滤，但是不知道为啥：

@Around("this(net.risesoft.repository.DyGroupRepository)")  
不太好使，也不知道为啥不好使比较尴尬，这个问题留给路过的大牛来指导吧。

废话到此，**言归正传**~

首先研习了一下源码，发现jpaRepositoryFactory父类RepositoryFactorySupport里面代理建立repository的时候有这么一段：

1. @SuppressWarnings({ "unchecked" })
2. public <T> T getRepository(Class<T> repositoryInterface, Object customImplementation) {
3. RepositoryMetadata metadata = getRepositoryMetadata(repositoryInterface);
4. Class<?> customImplementationClass = null == customImplementation ? null : customImplementation.getClass();
5. RepositoryInformation information = getRepositoryInformation(metadata, customImplementationClass);
6. validate(information, customImplementation);
7. Object target = getTargetRepository(information);
8. *// Create proxy*
9. ProxyFactory result = new ProxyFactory();
10. result.setTarget(target);
11. result.setInterfaces(new Class[] { repositoryInterface, Repository.class });
12. result.addAdvice(ExposeInvocationInterceptor.INSTANCE);
13. if (TRANSACTION\_PROXY\_TYPE != null) {
14. result.addInterface(TRANSACTION\_PROXY\_TYPE);
15. }
16. for (RepositoryProxyPostProcessor processor : postProcessors) {
17. processor.postProcess(result, information);
18. }
19. if (IS\_JAVA\_8) {
20. result.addAdvice(new DefaultMethodInvokingMethodInterceptor());
21. }
22. result.addAdvice(new QueryExecutorMethodInterceptor(information, customImplementation, target));
23. return (T) result.getProxy(classLoader);
24. }

着重注意这里：

1. for (RepositoryProxyPostProcessor processor : postProcessors) {
2. processor.postProcess(result, information);
3. }

**发现这里有一个List<RepositoryProxyPostProcessor> postProcessors**

**这个是用来为用ProxyFactory生成的Repository绑定一系列advice（MethodInterceptor拦截器）的一个封装list。**

接下来安子就有思路了，只需要把我们**自己的拦截器**添加到这个list里面就好了。

无巧不成书，很快发现RepositoryFactorySupport里有这个方法：

1. public void addRepositoryProxyPostProcessor(RepositoryProxyPostProcessor processor) {
2. Assert.notNull(processor, "RepositoryProxyPostProcessor must not be null!");
3. this.postProcessors.add(processor);
4. }

接下来就可以开始行动了！

写自己的**自定义拦截器逻辑**：

1. public abstract class SubSecrecyFilter {
2. public static Object dofilter(MethodInvocation Invocation) throws Throwable{
3. System.out.println("将要执行方法："+Invocation.getMethod().getName());
4. Object obj=Invocation.proceed();
5. System.out.println(Invocation.getMethod().getName()+"已经被执行");
6. return obj;
7. }
8. }

写RepositoryProxyPostProcessor的**实现**：

1. public class SecurecyPostProcessor implements RepositoryProxyPostProcessor{
2. @Override
3. public void postProcess(ProxyFactory factory,RepositoryInformation repositoryInformation) {
4. factory.addAdvice(SecurecyAdvice.instance);
5. }
6. static enum SecurecyAdvice implements MethodInterceptor{
7. instance;
8. @Override
9. public Object invoke(MethodInvocation invocation) throws Throwable {
10. return SubSecrecyFilter.dofilter(invocation);
11. }
12. }
13. }

改一下默认的JpaRepositoryFactoryBean来把我们自己定义的**SecurecyPostProcessor**  
插入List<RepositoryProxyPostProcessor> postProcessors ：

1. public class SubRepositoryFactoryBean<R extends JpaRepository<T, I>, T, I extends Serializable>
2. extends JpaRepositoryFactoryBean<R, T, I> {
3. protected RepositoryFactorySupport createRepositoryFactory(EntityManager entityManager) {
4. JpaRepositoryFactory jpaFac = new JpaRepositoryFactory(entityManager);
5. jpaFac.addRepositoryProxyPostProcessor(new SecurecyPostProcessor());
6. return jpaFac;
7. }
8. }

最后一步也最关键更改**xml配置**：

<jpa:repositories factory-class="net.risesoft.repository.factory.SubRepositoryFa

至此大功告成！  
安子实测：

<span style="font-size:10px;">将要执行方法：getBindUserBydpmiid

Hibernate: select dybinduser0\_.bindid as bindid1\_3\_, dybinduser0\_.dpmiid as dpmiid2\_3\_, dybinduser0\_.personid as personid3\_3\_ from dy\_binduser dybinduser0\_ where dybinduser0\_.dpmiid=?

getBindUserBydpmiid已经被执行

Hibernate: select acrolenode0\_.ID as ID1\_3\_, acrolenode0\_.DN as DN2\_3\_, acrolenode0\_.CREATETIME as CREATETI3\_3\_, acrolenode0\_.CUSTOMID as CUSTOMID4\_3\_, acrolenode0\_.DESCRIPTION as DESCRIPT5\_3\_, acrolenode0\_.NAME as NAME6\_3\_, acrolenode0\_.PARENT\_ID as PARENT\_I7\_3\_, acrolenode0\_.PROPERTIES as PROPERTI8\_3\_, acrolenode0\_.SHORTDN as SHORTDN9\_3\_, acrolenode0\_.systemCnName as systemC10\_3\_, acrolenode0\_.systemName as systemN11\_3\_, acrolenode0\_.TABINDEX as TABINDE12\_3\_, acrolenode0\_.TYPE as TYPE13\_3\_ from RC8\_AC\_ROLENODE acrolenode0\_ where acrolenode0\_.PARENT\_ID=? order by acrolenode0\_.TABINDEX asc

Hibernate: select acrolenode0\_.ID as ID1\_3\_, acrolenode0\_.DN as DN2\_3\_, acrolenode0\_.CREATETIME as CREATETI3\_3\_, acrolenode0\_.CUSTOMID as CUSTOMID4\_3\_, acrolenode0\_.DESCRIPTION as DESCRIPT5\_3\_, acrolenode0\_.NAME as NAME6\_3\_, acrolenode0\_.PARENT\_ID as PARENT\_I7\_3\_, acrolenode0\_.PROPERTIES as PROPERTI8\_3\_, acrolenode0\_.SHORTDN as SHORTDN9\_3\_, acrolenode0\_.systemCnName as systemC10\_3\_, acrolenode0\_.systemName as systemN11\_3\_, acrolenode0\_.TABINDEX as TABINDE12\_3\_, acrolenode0\_.TYPE as TYPE13\_3\_ from RC8\_AC\_ROLENODE acrolenode0\_ where acrolenode0\_.PARENT\_ID=? order by acrolenode0\_.TABINDEX asc

将要执行方法：findgroupbydpmiidandflag

Hibernate: select dygroup0\_.groupid as groupid1\_5\_, dygroup0\_.dpmiid as dpmiid2\_5\_, dygroup0\_.dysecrecyid as dysecrec5\_5\_, dygroup0\_.flag as flag3\_5\_, dygroup0\_.senddate as senddate4\_5\_ from dy\_group dygroup0\_ where dygroup0\_.flag=? and dygroup0\_.dpmiid=? order by dygroup0\_.groupid asc

findgroupbydpmiidandflag已经被执行<strong>

</strong>将要执行方法：getArticleBygroup

Hibernate: select dyarticle0\_.articleid as articlei1\_1\_, dyarticle0\_.creatdate as creatdat2\_1\_, dyarticle0\_.groupid as groupid3\_1\_, dyarticle0\_.hotnum as hotnum4\_1\_, dyarticle0\_.img as img5\_1\_, dyarticle0\_.num as num6\_1\_, dyarticle0\_.text as text7\_1\_, dyarticle0\_.title as title8\_1\_, dyarticle0\_.zy as zy9\_1\_ from dy\_article dyarticle0\_ where dyarticle0\_.groupid=?

getArticleBygroup已经被执行</span>

到这里就结束了，如果有不同意见最好留言，因为安子之前在网上也没找到其他方法，倒是有一篇文章：

http://blog.csdn.net/zhongzh86/article/details/54313706

给了我很大的灵感。

接下来附镇文图T-54：

AOP Jpa Hibernate Repository 自定义 过滤 权限控制