第五章 企业项目开发--mybatis注解与xml并用

本章的代码建立在第四章《Java框架整合--切分配置文件》的项目代码之上

在实际开发中,我们在使用mybatis的时候,会注解与xml形式一起使用。

1、二者的使用场景

xml使用场景(3个):

- 条件不定的查询 (eg.下边代码中的getAdminByConditions方法)
- 增加对象返回自增主键 (eg.下边代码的insertAdminWithBackId方法)

注意:前两条是硬性的,是注解所解决不了的,而第三条只是建议。

除了以上这三条之外,其他的都使用去注解就好。

2、代码实现

基本代码不变,这只列出修改过得代码:

2.1 ssmm0-userManagement:

A dminController

```
1 package com.xxx.web.admin;
2
 3 import java.util.List;
 4
5 import org.springframework.beans.factory.annotation.Autowired;
 6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.RequestMapping;
8 import org.springframework.web.bind.annotation.RequestParam;
9 import org.springframework.web.bind.annotation.ResponseBody;
10 import org.springframework.web.servlet.ModelAndView;
11
12 import com.xxx.model.userManagement.Admin;
13 import com.xxx.service.userManagement.AdminService;
14
15 /**
16 * adminController
17 */
18 @Controller
19 @RequestMapping("/admin")
20 public class AdminController {
21
22
       @Autowired
23
      private AdminService adminService;
24
25
       /**
26
       * 管理员注册
27
28
       @ResponseBody
29
       @RequestMapping("/register")
30
       public boolean register(@RequestParam("username") String username,
31
                               @RequestParam("password") String password){
32
           Admin admin = new Admin();
33
           admin.setUsername(username);
           admin.setPassword(password);
34
35
36
           boolean isRegisterSuccess = adminService.register(admin);
37
38
           return isRegisterSuccess;
39
40
```

```
/**
41
       * 管理员登录
42
       */
43
44
      @RequestMapping("/login")
      public ModelAndView login(@RequestParam("username") String username,
45
                               @RequestParam("password") String password) {
46
47
          Admin admin = adminService.login(username, password);
48
49
          ModelAndView modelAndView = new ModelAndView();
50
          if (admin == null) {
              modelAndView.addObject("message", "用户不存在或者密码错误!请重新输入");
51
              modelAndView.setViewName("error");
52
53
          }else{
              modelAndView.addObject("admin", admin);
54
              modelAndView.setViewName("userinfo");
55
          }
56
57
58
          return modelAndView;
59
60
                  61
      /**
62
63
       * 根据username或password查找List<Admin>
       */
64
65
      @ResponseBody
      @RequestMapping("/findAdmin")
66
      public List<Admin> findAdmin(@RequestParam(value="username", required=false) String username,
67
                                     @RequestParam(value="password", required=false) String password,
68
69
                                     @RequestParam("start") int start,
                                     @RequestParam("limit") int limit) {
70
          List<Admin> adminList = adminService.findAdmin(username, password, start, limit);
71
72
          return adminList;
73
      }
74
      /**
75
       * 插入一个用户并返回主键
76
77
       * 注意:get请求也会自动装配(即将前台传入的username和password传入admin)
78
       */
79
      @ResponseBody
80
      @RequestMapping("/insert")
81
      public Admin insertAdminWithBackId(Admin admin) {
82
          return adminService.insertAdminWithBackId(admin);
83
      }
84 }
```

说明:在这里增加了两个方法,具体看代码与注释

注:

• springMVC通过get方式传递的属性值username、password也能自动装配到对象admin中

2.2、ssmm0-data:

AdminService

```
package com.xxx.service.userManagement;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.xxx.dao.userManagement.AdminDao;
```

```
9 import com.xxx.model.userManagement.Admin;
10
11 /**
12 * 管理员service
13 */
14 @Service
15 public class AdminService {
16
       @Autowired
17
      private AdminDao adminDao;
18
19
      public boolean register(Admin admin) {
20
          return adminDao.register(admin);
21
22
23
       public Admin login(String username, String password) {
24
           return adminDao.login(username, password);
25
26
       /*******以下方法是为了测试mybatis中使用xml*******/
27
       public List<Admin> findAdmin(String username, String password, int start, int limit) {
28
          return adminDao.findAdmin(username, password, start, limit);
29
30
       }
31
       public Admin insertAdminWithBackId(Admin admin) {
32
          int record = adminDao.insertAdminWithBackId(admin);
33
34
          if (record==1) {
               return admin; //这时的admin已经被赋予主键了
35
36
          }
          return null;
37
38
       }
39 }
```

AdminDao

```
1 package com.xxx.dao.userManagement;
3 import java.util.List;
 4
5 import org.springframework.beans.factory.annotation.Autowired;
 6 import org.springframework.stereotype.Repository;
7
8 import com.xxx.mapper.userManagement.AdminMapper;
9 import com.xxx.model.userManagement.Admin;
10
11 /**
12 * 管理员DAO
13 */
14 @Repository
15 public class AdminDao {
       @Autowired
16
       private AdminMapper adminMapper;
17
18
19
       public boolean register(Admin admin) {
           return adminMapper.insertAdmin(admin) == 1?true:false;
20
21
       }
22
23
       public Admin login(String username ,String password) {
24
          return adminMapper.selectAdmin(username, password);
25
26
27
       public List<Admin> findAdmin(String username, String password, int start, int limit){
```

```
return adminMapper.getAdminByConditions(username, password, start, limit);

public int insertAdminWithBackId(Admin admin) {

return adminMapper.insertAdminWithBackId(admin);

}

}
```

AdminMapper

```
1 package com.xxx.mapper.userManagement;
2
3 import java.util.List;
4
5 import org.apache.ibatis.annotations.Insert;
6 import org.apache.ibatis.annotations.Param;
7 import org.apache.ibatis.annotations.Result;
8 import org.apache.ibatis.annotations.Results;
9 import org.apache.ibatis.annotations.Select;
10
11 import com.xxx.model.userManagement.Admin;
12
13 /**
  * 管理员Mapper
14
15 */
16 public interface AdminMapper {
17
      /************注解*********/
18
19
       @Insert("INSERT INTO userinfo(username, password) VALUES(#{username},#{password})")
       public int insertAdmin(Admin admin);
20
21
22
       @Select("SELECT * FROM userinfo WHERE username = #{username} AND password = #{password}")
23
      @Results(value = {
              @Result(id = true, column = "id", property = "id"),
24
              @Result(column = "username", property = "username"),
25
              @Result(column = "password", property = "password") })
26
      public Admin selectAdmin(@Param("username") String username,
27
                               @Param("password") String password);
28
29
       /**************/
30
31
      /**
       * 条件不定式查询
32
       * 我们这里使用@Param指定参数,这样的话,在AdminMapper.xml中就不用再使用parameterType属性了;否则得写
33
parameterType属性
34
       */
35
      public List<Admin> getAdminByConditions(@Param("username")String username,
36
                                              @Param("password")String password,
                                              @Param("start") int start,
37
                                              @Param("limit")int limit);
38
39
       /**
40
       * 返回主键
41
       */
42
43
       public int insertAdminWithBackId(Admin admin);
44 }
```

注意:在用xml传参的时候,

- 如果你直接传参,eg.insertAdminWithBackId(Admin admin),则在xml中的insertAdminWithBackId方法处要添加parameterType;
- 如果你用了注解传参的话, eg.getAdminByConditions(@Param("username")String username),则在xml中的getAdminByConditions方法处

不用添加parameterType,当然,注解传参的时候,不能传引用类型,一般只传基本类型,eg.insertAdminWithBackId(@Param("admin")Admin admin)就是不行的

接口定义好之后,需要添加两个配置文件+修改两个配置文件。目录结构如下:

AdminMapper.xml (该xml的名字最好与对应接口的接口名完全相同)

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
 3
 4 <!-- 指定工作空间,要与接口名相同,源代码没有去看,猜测应该是通过"这里的namespace.下边方法的id"来定位方法的 -->
5 <mapper namespace="com.xxx.mapper.userManagement.AdminMapper">
      <!-- 指定字段映射 -->
 6
      <resultMap type="Admin" id="adminResultMap">
 7
          <id property="id" column="id" jdbcType="INTEGER" />
 8
 9
          <result property="username" column="username" jdbcType="VARCHAR" />
10
          <result property="password" column="password" jdbcType="VARCHAR" />
      </resultMap>
11
12
      <select id="getAdminByConditions" resultMap="adminResultMap"><!-- 返回结果为上边指定的adminResultMap -->
13
14
          <![CDATA[ SELECT * FROM userinfo WHERE 1=1 ]]>
          <if test="username != null"><![CDATA[ AND username = #{username} ]]></if>
15
          <if test="password != null"><![CDATA[ AND password = #{password} ]]></if>
16
17
          <![CDATA[ ORDER BY id ASC LIMIT #{start}, #{limit} ]]>
      </select>
18
19
      <!-- 若不需要自动返回主键,将useGeneratedKeys="true" keyProperty="id"去掉即可 -->
20
21
      <insert id="insertAdminWithBackId" parameterType="Admin" useGeneratedKeys="true" keyProperty="id">
22
         <! [CDATA[
         INSERT INTO userinfo
23
24
25
             username,
26
             password
27
         )
         VALUES
28
29
30
             #{username, jdbcType=VARCHAR},
31
             #{password, jdbcType=VARCHAR}
32
33
         ]]>
34
     </insert>
35
36 </mapper>
```

注意:

- 该xml的名字最好与对应接口的接口名完全相同 (eg.AdminMapper.xml对于应接口AdminMapper)
- parameterType有无参照上边对AdminMapper处所讲的注意点
- 返回自增主键有两种方法,我这里列出了最常用的也是最简单的一种

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE configuration
      PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
 3
 4
      "http://mybatis.org/dtd/mybatis-3-config.dtd">
5
 6 <configuration>
7
      properties>
 8
          cproperty name="dialect" value="mysql" />
9
      </properties>
10
11
      <typeAliases>
          <!-- 这样会将com.xxx.model包及其子包下的所有类起别名为相应的简单类名 -->
12
          <package name="com.xxx.model"/>
13
14
          <!-- 如果这样去起别名的话,每一个模型类都要写一个,就比较麻烦 -->
          <!-- <typeAlias alias="Admin" type="com.xxx.model.userManagement.Admin"/> -->
15
16
      </typeAliases>
17 </configuration>
```

注意:这个文件一般用于指定属性和别名。

- 通常,属性只指定数据库方言即可;
- 有两种别名方式指定,请参照上述代码给出的注释进行选择,一般而言,都会选择package方式的

spring-data.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org
 3
/schema/context"
      xsi:schemaLocation="http://www.springframework.org/schema/beans"
 4
 5
                             http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
 6
                             http://www.springframework.org/schema/context
 7
                             http://www.springframework.org/schema/context/spring-context-3.2.xsd">
 8
      <!-- 注解扫描 -->
 9
10
       <context:component-scan base-package="com.xxx" />
11
       <!-- 引入数据源,这里变量的读取都是从ssmm0的pom.xml中读取的 -->
12
13
       <bean id="xxxDataSource" class="org.apache.tomcat.jdbc.pool.DataSource" destroy-method="close">
          cproperty name="driverClassName" value="${jdbc.driverClassName}" />
14
15
           cproperty name="url" value="${jdbc.url}" />
          cproperty name="username" value="${jdbc.username}" />
16
          cproperty name="password" value="${jdbc.password}" />
17
18
       </bean>
19
       <!-- 引\lambdamybatis -->
20
21
       <bean id="xxxSqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
          cproperty name="dataSource" ref="xxxDataSource" />
22
23
           <!-- 以下两个属性是专门为xml方式配置的,若只使用注解方式,这两个属性行可以不配置 -->
          cproperty name="configLocation" value="classpath:mybatis.xml"/>
24
25
          cproperty name="mapperLocations">
              t>
26
                  <value>classpath*:mapper/admin/*Mapper.xml</value>
27
28
              </list>
29
          </property>
30
       </bean>
31
       <bean id="xxxMapperScannerConfigurer" class="org.mybatis.spring.mapper.MapperScannerConfigurer">
```

```
<!--
32
              这里就是包名为什么就做com.xxx.mapper.user而非com.xxx.user.mapper,
33
              这样的话,比如说有两个项目com.xxx.mapper.user和com.xxx.mapper.hotel,value只需写作com.xxx.mapper即
34
可
              否则, value就要写作com.xxx.user.mapper,com.xxx.hotel.mapper
35
36
          cproperty name="basePackage" value="com.xxx.mapper" />
37
          cproperty name="sqlSessionFactoryBeanName" value="xxxSqlSessionFactory" />
38
39
      </bean>
40
41 </beans>
```

说明:只增加了两个属性配置,看代码与注释。

注:关于class path与class path*的具体区别自己去查,简单来说就是两句话:**classpat h只加载第一个找到文件;classpt h*加载找到的多个文件**。

pom.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
 2 corject xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4 0 0.xsd">
 3
 4
       <modelVersion>4.0.0/modelVersion>
 5
 6
 7
       <groupId>com.xxx
 8
       <artifactId>ssmm0</artifactId>
       <version>1.0-SNAPSHOT
 9
 10
       <name>ssmm0</name>
11
       <packaging>pom</packaging><!-- 父模块 -->
12
 13
       <!-- 管理子模块 -->
 14
1.5
       <modules>
          <module>userManagement</module><!-- 具体业务1-人员管理系统 -->
 16
          <module>data</module><!-- 封装数据操作 -->
17
 18
       </modules>
 19
 20
       properties>
 21
          ct.reporting.outputEncoding>UTF-8/project.reporting.outputEncoding>
 22
       </properties>
 23
 24
 25
       <!-- dependencyManagement不会引入实际的依赖,只是作为一个依赖池,供其和其子类使用 -->
       <dependencyManagement>
 26
          <dependencies>
 27
28
              <!-- json -->
 29
              <dependency>
 30
                 <groupId>com.alibaba
 31
                  <artifactId>fastjson</artifactId>
 32
                 <version>1.1.39
 33
              </dependency>
              <!-- servlet -->
 34
 35
              <dependency>
                 <groupId>javax.servlet
 36
 37
                 <artifactId>javax.servlet-api</artifactId>
                 <version>3.0.1
 38
                 <scope>provided</scope>
 39
              </dependency>
 40
              <!-- spring -->
 41
 42
              <dependency>
                 <groupId>org.springframework
 43
```

```
44
                  <artifactId>spring-core</artifactId>
 45
                  <version>3.2.6.RELEASE
              </dependency>
 46
 47
              <dependency>
                  <groupId>org.springframework
 48
                  <artifactId>spring-beans</artifactId>
 49
                  <version>3.2.6.RELEASE
 50
              </dependency>
 51
              <dependency>
 52
                  <groupId>org.springframework
 53
 54
                  <artifactId>spring-context</artifactId>
                  <version>3.2.6.RELEASE
 55
              </dependency>
 56
              <dependency>
 57
 58
                  <groupId>org.springframework
 59
                  <artifactId>spring-web</artifactId>
                  <version>3.2.6.RELEASE
 60
              </dependency>
 61
              <dependency>
 62
 63
                  <groupId>org.springframework
                  <artifactId>spring-webmvc</artifactId>
 64
                  <version>3.2.6.RELEASE
 65
 66
              </dependency>
              <!-- 这个是使用velocity的必备包 -->
 67
              <dependency>
 68
                  <groupId>org.springframework
 69
                  <artifactId>spring-context-support</artifactId>
 70
 71
                  <version>3.2.6.RELEASE
              </dependency>
 72
              <!-- mysql -->
 73
 74
              <dependency>
                  <groupId>mysql
 75
 76
                  <artifactId>mysql-connector-java</artifactId>
                  <version>5.1.27
77
 78
                  <scope>runtime</scope>
              </dependency>
 79
              <!-- 数据源 -->
 80
              <dependency>
 81
 82
                  <groupId>org.apache.tomcat
 83
                  <artifactId>tomcat-jdbc</artifactId>
                  <version>7.0.47
 84
 85
              </dependency>
              <!-- mybatis -->
 86
              <dependency>
 87
                  <groupId>org.mybatis
 88
                  <artifactId>mybatis</artifactId>
 89
                  <version>3.1.1
 90
              </dependency>
 91
              <dependency>
 92
                  <groupId>org.mybatis
 93
94
                  <artifactId>mybatis-spring</artifactId>
 95
                  <version>1.1.1
              </dependency>
 96
              <!-- velocity -->
 97
              <dependency>
 98
99
                  <groupId>org.apache.velocity
                  <artifactId>velocity</artifactId>
100
101
                  <version>1.5</version>
102
              </dependency>
              <dependency>
103
104
                  <groupId>velocity-tools
                  <artifactId>velocity-tools-generic</artifactId>
105
                  <version>1.2</version>
106
107
              </dependency>
              <!-- 用于加解密 -->
108
```

```
109
               <dependency>
110
                  <groupId>commons-codec
111
                  <artifactId>commons-codec</artifactId>
112
                  <version>1.7</version>
               </dependency>
113
114
              <dependency>
                  <groupId>org.bouncycastle
115
116
                  <artifactId>bcprov-jdk15on</artifactId>
                  <version>1.47</version>
117
              </dependency>
118
              <!-- 集合工具类 -->
119
              <dependency>
120
121
                  <groupId>org.apache.commons
                  <artifactId>commons-collections4</artifactId>
122
                  <version>4.0</version>
123
124
              </dependency>
              <!-- http -->
125
126
              <dependency>
                  <groupId>org.apache.httpcomponents
127
                  <artifactId>httpclient</artifactId>
128
129
                  <version>4.2.6
              </dependency>
130
           </dependencies>
131
132
       </dependencyManagement>
133
       <!-- 引入实际依赖 -->
134
135
       <dependencies>
136
           <!-- json -->
137
           <dependency>
              <groupId>com.alibaba
138
139
              <artifactId>fastjson</artifactId>
           </dependency>
140
141
           <!-- spring -->
142
           <dependency>
143
               <groupId>org.springframework</groupId>
               <artifactId>spring-core</artifactId>
144
145
           </dependency>
           <dependency>
146
               <groupId>org.springframework
147
148
               <artifactId>spring-beans</artifactId>
           </dependency>
149
150
           <dependency>
              <groupId>org.springframework
151
152
               <artifactId>spring-context</artifactId>
153
           </dependency>
           <!-- 集合工具类 -->
154
155
           <dependency>
               <groupId>org.apache.commons
156
               <artifactId>commons-collections4</artifactId>
157
158
           </dependency>
       </dependencies>
159
160
161
       <build>
162
           <resources>
              <!-- 这里配置了这一块儿true,才可以让指定文件(这里是src/main/resources/spring-data.xml)读到
163
pom.xml中的配置信息
                   ,值得注意的是,如果src/main/resources下还有其他文件,而你不想让其读pom.xml, 你还必须得把
src/main/resources下的其余文件再配置一遍,配置为false(不可读pom.xml),
                  如下边的注释那样,否则,会报这些文件(在这里,就是*.properties)找不到的错误
165
166
167
              <re>ource>
                  <directory>src/main/resources</directory>
168
                  <filtering>true</filtering>
169
170
                  <includes>
                      <include>*.xml</include>
171
```

```
172
                  </includes>
              </resource>
173
              <!--
174
175
              <resource>
                  <directory>src/main/resources</directory>
176
                  <filtering>false</filtering>
177
178
                  <includes>
                     <include>*.properties</include>
179
180
                  </includes>
              </resource>
181
182
              -->
              <resource>
183
184
                  <directory>src/main/resources</directory>
                  <filtering>false</filtering>
185
                  <includes>
186
187
                  <!-- 这里如果不加这一条,那么在spring-data.xml中配置的xml将找不到classpath:mapper/admin
/AdminMapper.xml -->
                     <include>mapper/**/*.xml</include>
188
                  </includes>
189
190
              </resource>
          </resources>
191
       </build>
192
193
       <!--
194
          profiles可以定义多个profile,然后每个profile对应不同的激活条件和配置信息,从而达到不同环境使用不同配置信息
195
的效果
          注意两点:
196
197
           1) <activeByDefault>true</activeByDefault>这种情况表示服务器启动的时候就采用这一套env(在这里,就是
prod )
          2) 当我们启动服务器后,想采用开发模式,需切换maven的env为dev,如果env的配置本身就是dev,需要将env换成rc或
198
prod,点击apply,然后再将env切换成dev,点击apply才行
       -->
199
       ofiles>
200
          <!-- 开发env -->
201
          ofile>
202
203
              <id>dev</id>
204
              <activation>
205
                  <activeByDefault>false</activeByDefault>
206
                  property>
207
                     <name>env</name>
                     <value>dev</value>
208
209
                  </property>
              </activation>
210
              properties>
211
                  <env>dev</env>
212
213
                  <jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>
214
                  <!--
215
                      对于jdbc.url中内容的配置,如果需要配置 @amp;时,有两种方法:
216
                     1) 如下边这样,使用<![CDATA[XXX]]>包起来
217
                      2)使用jdbc.properties文件来读取此pom.xml,然后spring.xml再读取jdbc.properties文件显然,
218
前者更方便,而且还省了一个jdbc.properties的文件,但是,有的时候,还是会用后者的;
                      在使用后者的时候,注意三点:
219
                     1)需要修改上边的build中的内容
220
                      2)需要在spring.xml中配置<context:property-placeholder
221
location="classpath:jdbc.properties"/>
                      3)将jdbc.properties放在ssmm0-data项目中,之后需要将ssmm0-data项目的env配置为dev
222
223
224
                  <jdbc.url><![CDATA[jdbc:mysql://127.0.0.1:3306/blog?zeroDateTimeBehavior=convertToNull&</pre>
amp;useUnicode=true&characterEncoding=utf-8]]></jdbc.url>
                  <jdbc.username>root</jdbc.username>
225
                  <jdbc.password>123456</jdbc.password>
226
227
              </properties>
228
          </profile>
           <!-- 预上线env -->
229
```

```
230
            file>
                <id>rc</id>
231
232
                <activation>
233
                    <activeByDefault>false</activeByDefault>
234
                    property>
                        <name>env</name>
235
                        <value>rc</value>
236
237
                    </property>
                </activation>
238
239
                properties>
240
                    <env>rc</env>
241
                    <jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>
242
243
                    <!-- 假设的一个地址 -->
                    <jdbc.url><![CDATA[jdbc:mysql://10.10.10.100:3306/blog?zeroDateTimeBehavior=convertToNull&</pre>
244
amp;useUnicode=true&characterEncoding=utf-8]]></jdbc.url>
                    <jdbc.username>root2</jdbc.username>
245
                    <jdbc.password>1234562</jdbc.password>
246
247
                </properties>
            </profile>
248
            <!-- 线上env -->
249
            file>
250
                <id>prod</id>
251
252
                <activation>
253
                    <activeByDefault>true</activeByDefault>
254
                    property>
255
                        <name>env</name>
256
                        <value>prod</value>
257
                    </property>
                </activation>
258
259
                properties>
260
                    <env>prod</env>
261
                    <jdbc.driverClassName>com.mysql.jdbc.Driver</jdbc.driverClassName>
262
                    <!-- 假设的一个地址 -->
263
                    <jdbc.url><![CDATA[jdbc:mysql://99.99.99.999:3307/blog?zeroDateTimeBehavior=convertToNull&</pre>
264
amp;useUnicode=true&characterEncoding=utf-8]]></jdbc.url>
                    <jdbc.username>sadhijhqwui</jdbc.username>
265
266
                    <jdbc.password>zxczkchwihcznk=</jdbc.password>
267
                </properties>
            </profile>
268
        </profiles>
269
270 </project>
```

说明:只在resource部分增加了一行关于"接口.xml"的过滤配置(作用看注释)

测试:测试的具体操作见前一章。