

通用Mapper

<http://git.oschina.net/free/Mapper>

详细文档请看项目首页

通用Mapper的优势

- 使用通用Mapper可以让你方便的直接使用单表的增删改查方法
- 你不需要使用MyBatis生成器来生成一大堆的XML内容和接口方法。
- 你只需要继承通用Mapper的接口即可获得大量直接可用的方法。
- 表有变化的时候，只需要修改实体类，不需要重新生成接口和XML。

如何使用通用Mapper

- 1.引入通用Mapper依赖或者Jar包
- 2.创建自己项目的基础Mapper(Dao)
- 3.配置拦截器
- 4.配置Spring扫描接口
- 5.修改(或生成)实体类
- 6.创建实体对应的具体的Dao
- 7.在Service中使用

1.引入通用Mapper依赖或者Jar包

如果你使用Maven，只需要添加如下依赖：

```
<dependency>
  <groupId>tk.mybatis</groupId>
  <artifactId>mapper</artifactId>
  <!-- 建议使用最新版本, 最新版本请从项目首页查找 -->
  <version>x.x.x</version>
</dependency>
```

如果你想引入Jar包，你可以从下面的地址下载：

<https://oss.sonatype.org/content/repositories/releases/tk/mybatis/mapper>

<http://repo1.maven.org/maven2/tk/mybatis/mapper>

由于通用Mapper依赖JPA，所以还需要下载persistence-api-1.0.jar：

<http://repo1.maven.org/maven2/javax/persistence/persistence-api/1.0/>

当前最新版本为3.1.2

2.创建自己项目的基础Mapper(Dao)

```
1 package com.netstar.util;
2
3 import tk.mybatis.mapper.common.Mapper;
4 import tk.mybatis.mapper.common.MySqlMapper;
5
6 /**
7  * 通用DAO基础接口，其他的DAO继承该接口即可
8  *
9  * @author liuzenghui
10  *
11  * @param <T>
12  */
13 public interface NetStarDao<T> extends Mapper<T>, MySqlMapper<T> {
14
15 }
16
```

这里创建了一个名为NetStarDao<T>的接口，继承了通用Mapper中的Mapper<T>和MySqlMapper<T>接口。

NetStarDao<T>继承后包含的方法

```
NetStarDao<T>
A delete(T) : int - tk.mybatis.mapper.common.base.delete.DeleteMapper
A deleteByExample(Object) : int - tk.mybatis.mapper.common.example.DeleteByExampleMapper
A deleteByPrimaryKey(Object) : int - tk.mybatis.mapper.common.base.delete.DeleteByPrimaryKeyMapper
A insert(T) : int - tk.mybatis.mapper.common.base.insert.InsertMapper
A insertList(List<T>) : int - tk.mybatis.mapper.common.special.InsertListMapper
A insertSelective(T) : int - tk.mybatis.mapper.common.base.insert.InsertSelectiveMapper
A insertUseGeneratedKeys(T) : int - tk.mybatis.mapper.common.special.InsertUseGeneratedKeysMapper
A select(T) : List<T> - tk.mybatis.mapper.common.base.select.SelectMapper
A selectByExample(Object) : List<T> - tk.mybatis.mapper.common.example.SelectByExampleMapper
A selectByExampleAndRowBounds(Object, RowBounds) : List<T> - tk.mybatis.mapper.common.rowbounds.SelectByExampleRowBoundsMapper
A selectByPrimaryKey(Object) : T - tk.mybatis.mapper.common.base.select.SelectByPrimaryKeyMapper
A selectByRowBounds(T, RowBounds) : List<T> - tk.mybatis.mapper.common.rowbounds.SelectRowBoundsMapper
A selectCount(T) : int - tk.mybatis.mapper.common.base.select.SelectCountMapper
A selectCountByExample(Object) : int - tk.mybatis.mapper.common.example.SelectCountByExampleMapper
A selectOne(T) : T - tk.mybatis.mapper.common.base.select.SelectOneMapper
A updateByExample(T, Object) : int - tk.mybatis.mapper.common.example.UpdateByExampleMapper
A updateByExampleSelective(T, Object) : int - tk.mybatis.mapper.common.example.UpdateByExampleSelectiveMapper
A updateByPrimaryKey(T) : int - tk.mybatis.mapper.common.base.update.UpdateByPrimaryKeyMapper
A updateByPrimaryKeySelective(T) : int - tk.mybatis.mapper.common.base.update.UpdateByPrimaryKeySelectiveMapper
```

- 在insert和update中带Selective后缀的方法，在insert和update时，不会插入或者更新值为null的列。
- 带Example的方法，支持MyBatis生成器生成的XXXExample和通用Mapper自带的Example(XXX.class)类，Example方法能实现大部分的复杂单表操作，详细的用法可以自己尝试（最后面有个例子）。
- 具体方法的含义可以查看项目文档中的《Mapper3通用接口大全》

3.配置拦截器

在MyBatis配置文件中添加MapperInterceptor拦截器配置：

```
<configuration>
  <settings>
    <setting name="lazyLoadingEnabled" value="false" />
    <setting name="cacheEnabled" value="false" />
    <setting name="useGeneratedKeys" value="true" />
    <setting name="defaultExecutorType" value="REUSE" />
  </settings>

  <plugins>
    <plugin interceptor="tk.mybatis.mapper.mapperhelper.MapperInterceptor">
      <property name="mappers" value="com.netstar.util.NetStarDao"/>
    </plugin>
  </plugins>
</configuration>
```

注意这里的com.netstar.util.NetStarDao就是我们直接创建的基础接口

4. 配置Spring扫描接口

```
<!-- 自动扫描Mapper接口生成代理类 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="com.netstar.**.dao"/>
    <!-- 只有继承了NetStarDao的接口才自动生成代理类，其他的还需要手工编写实现类 -->
    <property name="markerInterface" value="com.netstar.util.NetStarDao"/>
    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
</bean>
```

增加该配置后，Spring会自动扫描com.netstar.**.dao包下继承了NetStarDao接口的所有接口类，并且自动生成接口的代理类。

5.修改(或生成)实体类

```
@Table(name = "basfile")
public class BasFile implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(generator = "JDBC")
    private Long id;

    private Long orgid;

    private String filepath;

    private String filetype;

    private Integer filelength;

    private Long userid;

    private Date uploadtime;

    private String originalfilename;

    private String description;
```

对实体增加适当的注解配置（通用Mapper包含了一个可以生成带注解实体类的MyBatis生成器插件，详细信息看项目文档）

6.创建实体对应的Dao

创建BasFile对应的BasFileDao接口，创建完该接口后，该接口就拥有了NetStarDao包含的全部方法。

```
1 package com.netstar.cus.bas.dao;
2
3 import com.netstar.cus.bas.model.BasFile;
4 import com.netstar.util.NetStarDao;
5
6 public interface BasFileDao extends NetStarDao<BasFile> {
7
8 }
```

如果你需要添加自己的接口方法，在这个接口中添加即可。

```
public interface BasFunsDao extends NetStarDao<BasFuns> {

    List<BasFuns> selectByUserId(long userId);

}
```

另外还要创建一个对应的xml，如下：

```
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 <mapper namespace="com.netstar.cus.bas.dao.BasFunsDao">
5     <!-- 根据userId获取权限信息 -->
6     <select id="selectByUserId" parameterType="long" resultType="com.netstar.cus.bas.model.BasFuns">
7         select e.id,a.*
8         from BasFuns a
9         join BasRoleFuns b on
10            a.id=b.funsId
11         join BasRole c on c.id=b.roleId
12         join BasUserRole d on
13            d.roleId=c.id
14         join BasUser e on e.id=d.userId
15         where
16            e.id= #{id}
17     </select>
18 </mapper>
```

需要保证namespace的值是接口的完整名称，另外这个xml也要被扫描到。

```
<!-- 系统数据库 -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="configLocation" value="classpath:mybatis-config.xml" />
    <property name="mapperLocations" value="classpath*:com/netstar/**/dao/*.xml" />
    <property name="dataSource" ref="dataSource" />
</bean>
```

实体类注解简单介绍

- **@Id** 主键字段必须加这个注解，联合主键的时候，可以给多个字段加**@Id**注解。和**PrimaryKey**有关的方法都需要该注解标记的字段，所以实体中至少要有有一个标记**@Id**的字段。
- **@Table, @Column**注解，默认情况下驼峰命名和数据库的下划线命名互相转换，**sysUser**对应表中的**sys_user**，当实体类名和字段名不是按照这种默认规则转换时，使用这俩注解指定，例如**@Table(name="sysuser")**，列类似。
- **@Transient**注解，当实体中的字段不是表中的字段时，需要加这个注解，这样在和表转换时，会忽略该字段。
- **@GeneratedValue(generator="JDBC")**注解，这里只介绍这一种，其他的参考Mapper文档，这个配置可以自动回写自增字段的值，相当于XML配置中的**userGeneratedKeys="true" keyProperty="id"**
- **@OrderBy**注解可以指定表默认的排序规则

7.在Service中使用

```
@Service
public class KeyWordServiceImpl implements KeyWordService {

    @Autowired
    private KeyWordDao keyWordDao;

    /**
     * 获取关键字
     *
     * @param id
     * @return
     */
    @Override
    public KeyWord getKeyWordById(Long id) throws Exception {
        return keyWordDao.selectByPrimaryKey(id);
    }

    /**
     * 分页查询关键字信息
     *
     * @param keyWord
     * @param pageNum
     * @param pageSize
     * @return
     */
    @Override
    public List<KeyWord> getKeyWords(KeyWord keyWord, int pageNum, int pageSize) throws Exception {
        //根据提供的条件判断keyWord对象，然后生成查询条件
        Example example = new Example(KeyWord.class);
        example.createCriteria().andEqualTo("pdfid", keyWord.getPdfid());

        //返回查询结果
        return keyWordDao.selectByExample(example);
    }
}
```

直接在Service中调用相应的方法即可。

Example方法用法

由于Example方法比较特殊，这里举两个例子：

```
OrgPackagePublishExample example = new OrgPackagePublishExample();
Criteria criteria = example.createCriteria();
//机构ID
if(publish.getOrgid() != null){
    criteria.andOrgidEqualTo(publish.getOrgid());
}
//开始时间
if(publish.getStarttime() != null){
    criteria.andStarttimeGreaterThanOrEqualTo(publish.getStarttime());
}
//结束时间
if(publish.getEndtime() != null){
    criteria.andEndtimeLessThanOrEqualTo(publish.getEndtime());
}
//状态
if(publish.getPackagestate() != null){
    criteria.andPackagestateEqualTo(publish.getPackagestate());
}
return publishDao.selectByExample(example);
```

MyBatis生成器生成的XXXExample

```
Example example = new Example(KeyWord.class);
example.createCriteria()
    .andEqualTo("pdfid", keyWord.getPdfid())
    .andLike("keyword", "%" + keyWord.getKeyword() + "%");
//返回查询结果
return keyWordDao.selectByExample(example);
```

通用Example(XX.class)