1. **在Spring4中使用通用Mapper**

Spring4增加了对泛型注入的支持，这个特性对通用Mapper来说，非常的有用，可以说有了这个特性，可以直接在Service中写Mapper<UserInfo> mapper，可以通过BaseService<T>来实现通用的Service。

这篇文档主要讲解通用Mapper在Spring4中的\*\*最佳用法\*\*。

1. **一、在Spring4中配置通用Mapper**

和Spring3中的配置一样，配置方法请看[这里](http://git.oschina.net/free/Mapper)有关Spring配置的信息。

如果有人不明白完整的配置什么样，可以看下面的例子：

* [Mybatis-Spring4项目](https://github.com/abel533/Mybatis-Spring/tree/spring4)
* [applicationContext.xml](https://github.com/abel533/Mybatis-Spring/blob/spring4/src/main/resources/applicationContext.xml)

1. **二、继承**Mapper<T>**实现自己的实体接口类**

这里以[Country2Mapper](https://github.com/abel533/Mybatis-Spring/blob/spring4/src/main/java/com/isea533/mybatis/mapper/Country2Mapper.java)为例：

public interface Country2Mapper extends Mapper<Country2> {

//省略其他自己增加的方法

}

如果你点进去上面的Country2Mapper查看，会发现里面还有一些Example的方法，这些是代码生成器生成的，生成的方法不包含通用的CRUD，只有Example的方法，还有一个对应的[Country2Mapper.xml](https://github.com/abel533/Mybatis-Spring/blob/spring4/src/main/java/com/isea533/mybatis/mapper/Country2Mapper.xml)。

这个例子主要说明，除了通用Mapper的方法外，你可以添加自己的方法，和原来的没有区别。

这里的实体[Country2](https://github.com/abel533/Mybatis-Spring/blob/spring4/src/main/java/com/isea533/mybatis/model/Country2.java)代码如下：

@Table(name="country")

public class Country2 {

@Id

private Integer id;

private String countryname;

private String countrycode;

//省略getter和setter方法

}

这里配置对应的表名为country。只有一个主键id。

1. **三、在Service中使用**

在Service中的使用方式有很多种。

1. **第一种、直接注入上面定义的**Country2Mapper

@Service

public class DemoService {

@Autowired

private Country2Mapper mapper;

public List<Country2> selectPage(int pageNum,int pageSize){

PageHelper.startPage(pageNum, pageSize);

//Spring4支持泛型注入

return mapper.select(null);

}

}

这种方式太常见，太普通，这里不多解释。

1. **第二种、泛型注入**

这种方式用的就很少了，但是Spring4支持泛型注入，因此在第一种的基础上，我们可以写出如下的代码：

@Service

public class DemoService {

@Autowired

private Mapper<Country2> mapper;

public List<Country2> selectPage(int pageNum,int pageSize){

//这里用到了分页插件PageHelper

PageHelper.startPage(pageNum, pageSize);

//Spring4支持泛型注入

return mapper.select(null);

}

}

对于不了解泛型注入的，可能会不习惯Mapper<Country2> mapper这种写法，实际上这么写的优势并不明显。还不如第一种明确。

但是通过第二种，我们可以引出第三种，也可能会是很常用的通用Service。

1. **第三种、通用Service**

一般操作数据库都在Service中进行，不可避免的就要写出大量重复的CRUD方法，如果能有一个通用的Service，肯定也会减少很多工作量。

这里通过简单扩展来讲，更复杂的封装，各位可以根据自己的情况动手实践。

如下简单例子：

@Service

public abstract class BaseService<T> {

@Autowired

protected Mapper<T> mapper;

public int save(T entity){

return mapper.insert(entity);

}

public int delete(T entity){

return mapper.deleteByPrimaryKey(entity);

}

/\*\*

\* 单表分页查询

\*

\* @param pageNum

\* @param pageSize

\* @return

\*/

public List<T> selectPage(int pageNum,int pageSize){

PageHelper.startPage(pageNum, pageSize);

//Spring4支持泛型注入

return mapper.select(null);

}

}

创建如上所示的抽象类BaseService<T>，这里封装三个方法仅作为简单的例子。需要更复杂逻辑的可以自行摸索。

然后修改刚才的DemoService例子：

@Service

public class DemoService extends BaseService<Country2>{

}

由于BaseService<T>封装了单表的分页插件，因此目前的DemoService中没有任何代码。

假如我们要增加一个包含校验的保存方法。添加如下代码：

@Service

public class DemoService extends BaseService<Country2>{

public int save(Country2 country2) {

if (country2 == null) {

throw new NullPointerException("保存的对象不能为空!");

}

if (country2.getCountrycode() == null || country2.getCountrycode().equals("")) {

throw new RuntimeException("国家代码不能为空!");

}

if (country2.getCountryname() == null || country2.getCountryname().equals("")) {

throw new RuntimeException("国家名称不能为空!");

}

return super.save(country2);

}

}

上面只是个例子，是否抛出异常各位不用计较。

从这个例子应该也能看到，当使用Spring4和通用Mapper的时候，是多么的方便。

1. **关于继承**Mapper<T>

我一开始为什么要设计为必须继承Mapper<T>实现自己的Mapper呢？

主要考虑到两个方面。

1. 通过<T>可以方便的获取泛型的类型，在通用的方法中就不需要传递实体类型。
2. 通过继承的Mapper,例如Country2Mapper，有独立的Mapper就意味着有独立的命名空间，可以缓存结果，并且不需要拦截器就能实现。

现在有了Spring4后，又有了一个很重要的原因。

* **支持泛型注入**，可以实现自己的通用Service，在通用Mapper基础上再次简化操作，加快开发效率。

1. **最后**

**如果之前说通用Mapper不如Mybatis-Generator自动生成好**，我也只能说看个人喜好，不需要通用Mapper的可以不用，通用Mapper只是为了满足一部分的人需要。

现在来看，\*\*如果还有人说通用Mapper不如Mybatis-Generator自动生成好\*\*，我会建议他看看[这篇文档](http://git.oschina.net/free/Mapper/blob/master/UseMapperInSpring4.md)

实际上，不需要说那个更好，适合自己的才好。

另外看完这篇文档后，不需要再说\*\*通用Mapper不如Mybatis-Generator自动生成好\*\*，因为我和一些朋友正在翻译\*\*Mybatis-Generator\*\*，最后还会提供\*\*Mybatis-Generator和通用Mapper的集成插件\*\*，可以用\*\*Mybatis-Generator\*\*直接生成实体类、继承通用Mapper的实体Mapper以及XML文件。

**Mybatis-Generator中文文档地址**：<http://generator.sturgeon.mopaas.com/>

**Mybatis-Generator官方英文地址**：<http://mybatis.github.io/generator/index.html>

这个文档还没有翻译完，而且译者水平有限，如果发现翻译错误或者不合适的地方，可以在下面的地址提ISSUE

[提交ISSUE](http://git.oschina.net/free/mybatis-generator-doc-zh/issues)

上面这个地址只是生成后的项目文档地址，并不是我们直接用来翻译的项目。