01-手写Mybatis-1

作者：王建安（Mike）Email：wang_jian_an@139.com QQ：3266399810

1 需求分析

1.1、Mybatis是什么？

一个半自动化的orm框架（Object Relation Mapping)。

1.2、Mybatis完成什么工作？

在面向对象编程中，我们操作的都是对象，Mybatis框架是一个数据访问层的框架，帮我们完成对象在数据库中的存、取工作。

为什么称为半自动化？

关系型数据库的操作是通过SQL语句来完成的，Mybatis在帮我们做对象的存取时，需要我们提供对应的SQL语句，它不自动帮我们生成SQL语句，而只帮我们完成：

- **1 对象属性到SQL语句参数的自动填充；**
- **2 SQL语句执行结果集到对象的自动提取；**

所以称为半自动的。而我们了解的另一个ORM框架Hibernate则是全自动的。

半自动化的不足：我们得辛苦一点编写SQL语句。

半自动化的优点：我们可以完全把控执行的SQL语句，可以随时灵活调整、优化。

1.3、为什么要用Mybatis?

1、mybatis学习、使用简单

2、半自动化的优点

1.4、为什么要学好Mybatis?

一线互联网公司出于性能、调优、使用简单、完全可控的需要，在数据库访问层都是采用Mybatis。

1.5 、为什么要用orm框架？

都是为了提高生产效率，少写代码，少写重复代码！

不用orm框架，能用什么来完成数据的存取？

jdbc

看看jdbc编程的代码示例

```
@Component
public class UserDao {

    @Autowired
    private DataSource dataSource;
    // 新增用户保存到数据库
    public void addUser(User user) throws SQLException {
        try (
                // 1、获取连接
                Connection conn = dataSource.getConnection();
                // 2、创建预编译语句对象
                PreparedStatement pst = conn.prepareStatement(
                        "insert into
t_user(id,name,sex,age,address,phone,wechat,email,account,password) "
    " values(?,?,?,?,?,?,?,?,?,?)");)
        {
            // 3、设置参数值
            int i = 1;
            pst.setString(i++, user.getId());
            pst.setString(i++, user.getName());
            pst.setString(i++, user.getSex());
            pst.setInt(i++, user.getAge());
            pst.setString(i++, user.getAddress());
            pst.setString(i++, user.getPhone());
            pst.setString(i++, user.getWechat());
            pst.setString(i++, user.getEmail());
```

```
            pst.setString(i++, user.getAccount());
            pst.setString(i++, user.getPassword());

            // 4、执行语句
            int changeRows = pst.executeUpdate();
        }
    }

    // 多条件查询用户
    public List<User> queryUsers(String likeName, int minAge, int
maxAge, String sex) throws SQLException {
        // 1、根据查询条件动态拼接SQL语句
        StringBuffer sql = new StringBuffer(
                "select
id,name,sex,age,address,phone,wechat,email,account,password from t_user
where 1 = 1 ");
        if (!StringUtils.isEmpty(likeName)) {
            sql.append(" and name like ? ");
        }
        if (minAge >= 0) {
            sql.append(" and age >= ? ");
        }
        if (maxAge >= 0) {
            sql.append(" and age <= ? ");
        }
        if (!StringUtils.isEmpty(sex)) {
            sql.append(" and sex = ? ");
        }

        try (Connection conn = dataSource.getConnection();
                PreparedStatement pst =
conn.prepareStatement(sql.toString());) {
            // 2 设置查询语句参数值
            int i = 1;
            if (!StringUtils.isEmpty(likeName)) {
                pst.setString(i++, "%" + likeName + "%");
            }
            if (minAge >= 0) {
                pst.setInt(i++, minAge);
            }
            if (maxAge >= 0) {
                pst.setInt(i++, maxAge);
```

```
            }
            if (!StringUtils.isEmpty(sex)) {
                pst.setString(i++, sex);
            }

            // 3 执行查询
            ResultSet rs = pst.executeQuery();

            // 4、提取结果集
            List<User> list = new ArrayList<>();
            User u;
            while (rs.next()) {
                u = new User();
                list.add(u);
                u.setId(rs.getString("id"));
                u.setName(rs.getString("name"));
                u.setSex(rs.getString("sex"));
                u.setAge(rs.getInt("age"));
                u.setPhone(rs.getString("phone"));
                u.setEmail(rs.getString("email"));
                u.setWechat(rs.getString("wechat"));
                u.setAccount(rs.getString("account"));
                u.setPassword(rs.getString("password"));
            }

            rs.close();

            return list;
        }
    }

}
```

用JdbcTemplate的代码示例：

```
@Component
public class UserDaoUseJdbcTemplate {
    @Autowired
    private JdbcTemplate jdbcTemplate;
    // 新增用户
    public void addUser(User user) throws SQLException {
```

```
        String sql = "insert into
t_user(id,name,sex,age,address,phone,wechat,email,account,password) "
            + " values(?,?,?,?,?,?,?,?,?,?)";

      jdbcTemplate.update(sql, user.getId(), user.getName(),
user.getSex(),
              user.getAge(), user.getAddress(),user.getPhone(),
user.getWechat(),
              user.getEmail(), user.getAccount(), user.getPassword());
      }

    // 多条件查询用户
    public List<User> queryUsers(String likeName, int minAge, int
maxAge, String sex) throws SQLException {
        // 1、根据查询条件动态拼接SQL语句
        StringBuffer sql = new StringBuffer(
                "select
id,name,sex,age,address,phone,wechat,email,account,password from t_user
where 1 = 1 ");
        List<Object> argList = new ArrayList<>();
        if (!StringUtils.isEmpty(likeName)) {
            sql.append(" and name like ? ");
            argList.add("%" + likeName + "%");
        }
        if (minAge >= 0) {
            sql.append(" and age >= ? ");
            argList.add(minAge);
        }
        if (maxAge >= 0) {
            sql.append(" and age <= ? ");
            argList.add(maxAge);
        }
        if (!StringUtils.isEmpty(sex)) {
            sql.append(" and sex = ? ");
            argList.add(sex);
        }

        return jdbcTemplate.query(sql.toString(), argList.toArray(),
new RowMapper<User>() {
            public User mapRow(ResultSet rs, int rowNum) throws
SQLException {
                User u = new User();
```

```
                u.setId(rs.getString("id"));
                u.setName(rs.getString("name"));
                u.setSex(rs.getString("sex"));
                u.setAge(rs.getInt("age"));
                u.setPhone(rs.getString("phone"));
                u.setEmail(rs.getString("email"));
                u.setWechat(rs.getString("wechat"));
                u.setAccount(rs.getString("account"));
                u.setPassword(rs.getString("password"));

                return u;
            }
        });
    }
}
```

参数设置代码、结果集处理代码、JDBC过程代码都会大量重复，毫无技术含量！

那就写个框架做了它！显示我们的牛B!

1.6 框架确切需求

1、用户只需定义持久层接口（dao接口）、接口方法对应的SQL语句。

2、用户需指明接口方法的参数与语句参数的对应关系。

3、用户需指明查询结果集与对象属性的映射关系。

4、框架完成接口对象的生成，JDBC执行过程。

2 设计

2.1 需求1

1、用户只需定义持久层接口（dao接口）、接口方法对应的SQL语句。

设计问题：

1、我们该提供什么样的方式来让用户定义SQL语句？

2、SQL语句怎么与接口方法对应？

3、这些SQL语句、对应关系我们框架需要获取到，谁来获取？又该如何表示存储

2.1.1 SQL定义方式

XML：独立于代码，修改很方便（不需改代码）

注解：直接加在方法上，零xml配置。

问题：SQL语句可做增、删、改、查操作，我们是否要对SQL做个区分？

答：要，因为jdbc中对应有不同的方法 executeQuery executeUpdate

xml ：设计增删改查的元素：

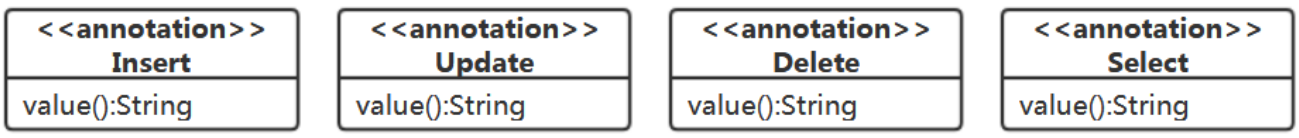
mybatis-SQL.dtd

```
<!ELEMENT insert(#PCDATA) >
<!ELEMENT update(#PCDATA) >
<!ELEMENT delete(#PCDATA) >
<!ELEMENT select (#PCDATA) >
```

元素体中定义SQL，示例

```
<insert>insert into t_user(id,name,sex,age) values(?,?,?,?)</insert>
```

注解：设计增删改查的注解：@Insert @Update @Delete @Select ，注解项定义SQL




```
@Documented
@Retention(RUNTIME)
@Target({ METHOD })
public @interface Insert {
    String value();
}
```

```
@Insert("insert into t_user(id,name,sex,age) values(?,?,?,?)")
public void addUser(User user);
```

2.1.2 SQL语句与接口方法对应

xml方式时，如何来映射SQL语句对应的接口方法？

为元素定义一个id，id的值为对应的类名.方法名，如何？

```
<insert id="com.study.mike.sample.UserDao.addUser">
    insert into t_user(id,name,sex,age) values(?,?,?,?)
</insert>
```

一个Dao接口中可能会定义很多个数据访问方法，id这么写很长，能不能便捷一点？

这是在做SQL与接口方法的映射，我们来加一个mapper元素，它可包含多个insert、update、delete、select元素，相当于分组，一个接口中定义的分到一组。在mapper中定义一个属性namespace，指定里面元素的名称空间，namespace的值对应接口类名，里面元素的id对应方法名。

mybatis-mapper.dtd

```
<!ELEMENT mapper (insert* | update* | delete* | select*)+ >
<!ATTLIST mapper namespace CDATA #IMPLIED >
<!ELEMENT insert(#PCDATA) >
<!ELEMENT update(#PCDATA) >
<!ELEMENT delete(#PCDATA) >
<!ELEMENT select (#PCDATA) >
```

这个xml文件命名为 userDaoMapper.xml，内容如下：

```
<mapper namespace="com.study.mike.sample.userDao">
    <insert id="addUser">
        insert into t_user(id,name,sex,age) values(?,?,?,?)
    </insert>
</mapper>
```

2.1.3 映射关系的获取与表示、存储

xml方式：

解析xml来获取

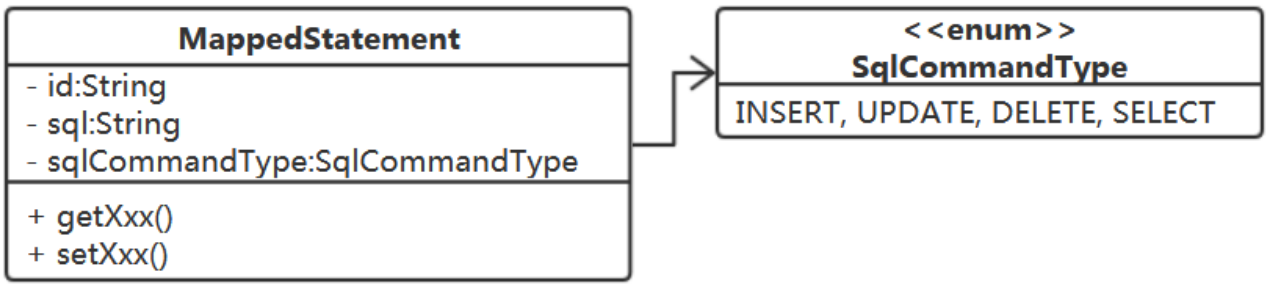
注解方式：

读取注解信息

问题：

1、怎么表示？

得设计一个类来表示从xml、注解获得的SQL映射信息。



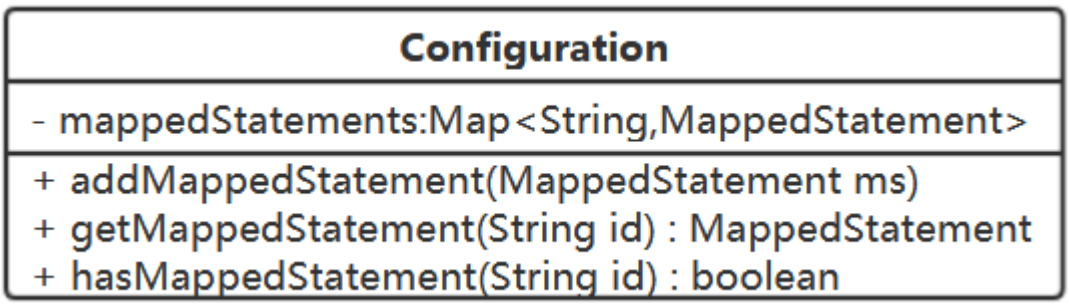
注意：id为唯一id.

xml方式：id=namespace.id属性值

注解方式：id=完整类名.方法名

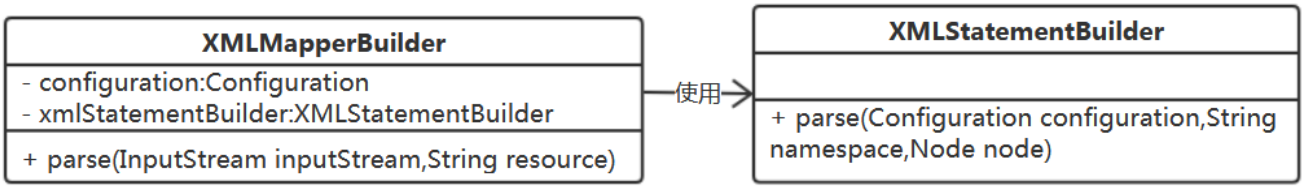
2、怎么存储得到的MappedStatement?

这些其实就是一个配置信息，我们定义一个Configuration类：



注意：key 为MappedStatement的id

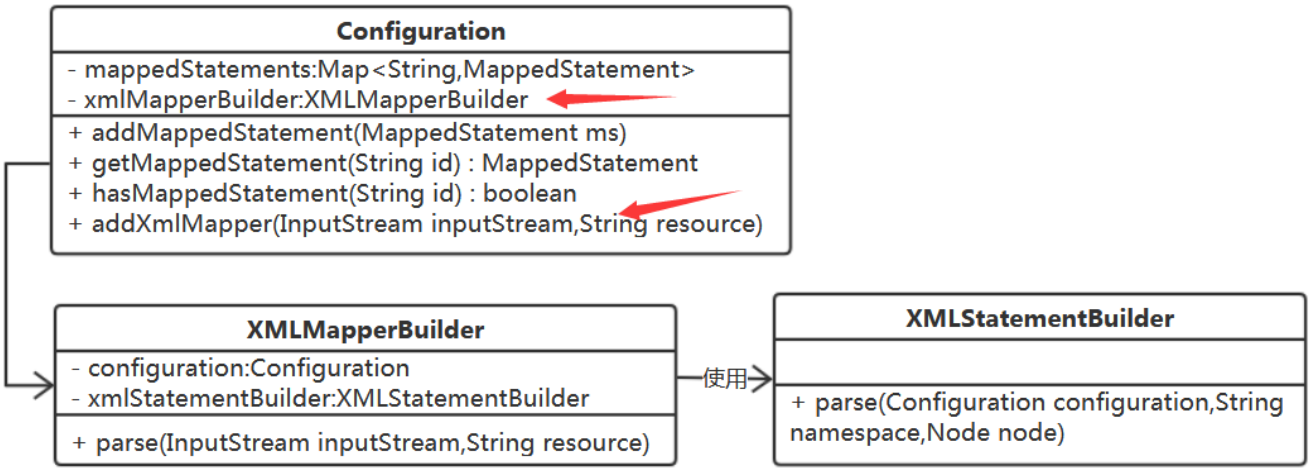
3、得有类来负责解析xml



XmlMapperBuilder负责解析xml文档（parse方法的resource参数用来指定inputStream的来源），它调用XMLStatementBuilder来解析里面的 insert、 update 、delete 、select 元素

4、谁来调用XMLMapperBuilder呢？

Configuration 吧，在它里面持有XmlMapperBuilder，增加一个添加mapper xml 的方法吧!



5、用户怎么来指定它们的xml mapper文件呢？

再加一个xml 配置文件来给用户来指定如何？

mybatis-config.xml

```
<configuration>
        <mappers>
                <mapper />
        <mapper />
        <mappers>
</configuration>
```

6、mapper中可以让用户如何来指定文件位置？

文件可以是在类目录下，也可是在文件系统目录下。如何区分？

规定：

类目录下的方式通过 resource属性指定；

文件系统文件通过 url属性指定，值采用URL 本地文件格式指定：file:///

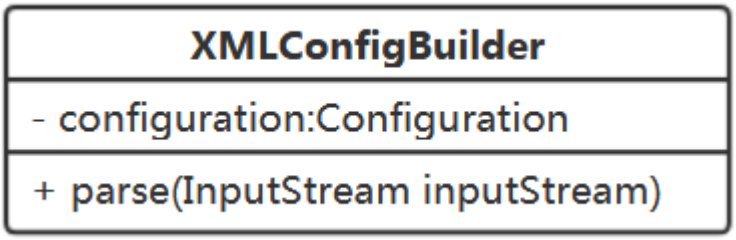
```
<configuration>
        <mappers>
                <mapper resource="com/mike/UserMapper.xml"/>
        <mapper url="file:///var/mappers/CourseMapper.xml"/>
        <mappers>
</configuration>
```

定义 mybatis-config.dtd

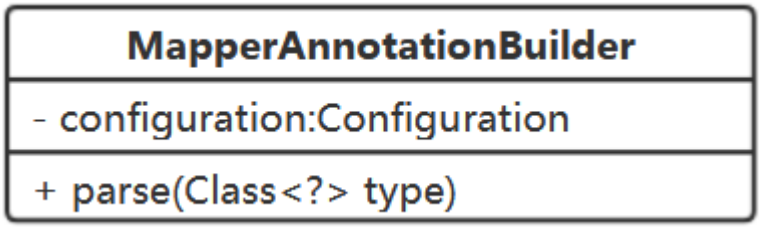
```
<!ELEMENT configuration (mappers?)+ >

<!ELEMENT mappers (mapper*)>

<!ELEMENT mapper EMPTY>
<!ATTLIST mapper
resource CDATA #IMPLIED
url CDATA #IMPLIED
>
```

7、增加了一个config xml文件，就的有类来解析它。

增加解析 mybatis-config.xml配置文件的类

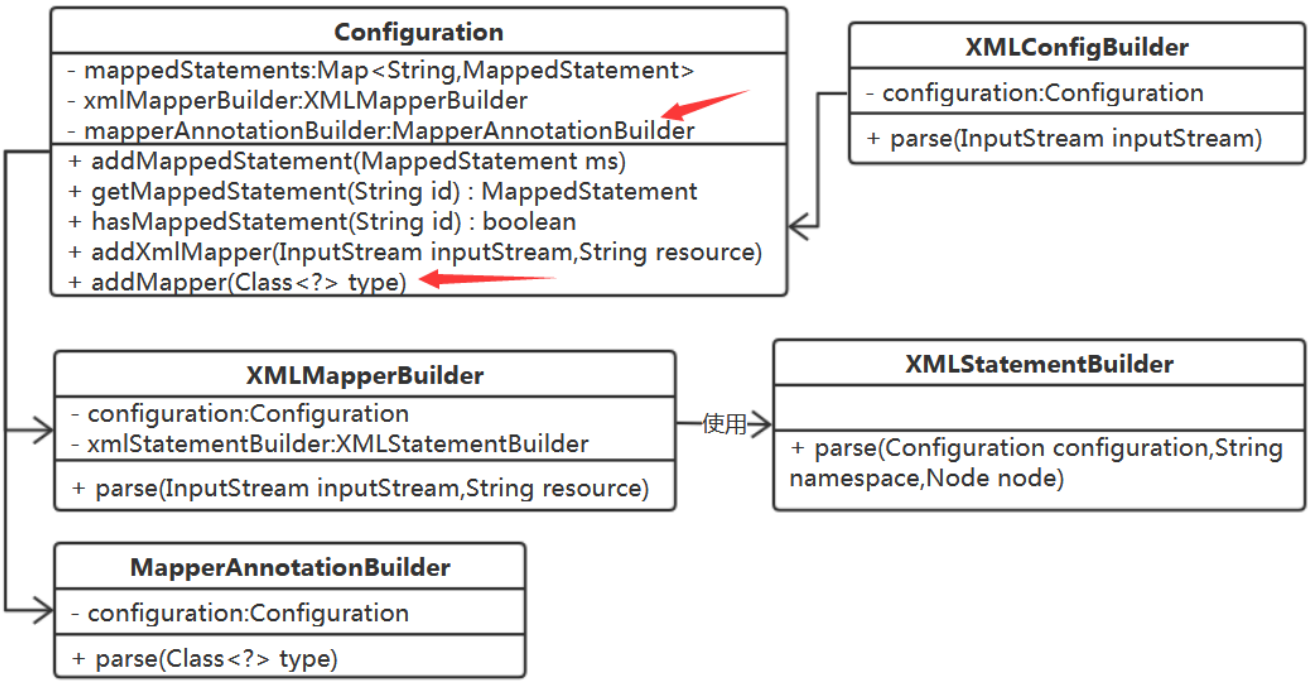


8、注解的方式需要获取SQL映射信息，也得有个类来做这件事



9、谁来使用MapperAnnotationBuilder？

Configuration吧，在它里面持有MapperAnnotationBuilder，增加添加Mapper接口类的方法。



10、用户如何来指定他们的Mapper接口类？

1、在mybatis-config.xml的mappers中通过mapper指定？

```
<configuration>
        <mappers>
                <mapper resource="com/mike/UserMapper.xml"/>
        <mapper url="file:///var/mappers/CourseMapper.xml"/>
        <mappers>
</configuration>
```

如何来区分它是个Mapper接口呢？

给mapper加一个属性class来专门指定Mapper类名

```
<configuration>
        <mappers>
                <mapper resource="com/mike/UserMapper.xml"/>
        <mapper url="file:///var/mappers/CourseMapper.xml"/>
        <mapper class="com.study.mike.dao.UserDao" />
        <mappers>
</configuration>
```

mybatis-config.dtd

```
<!ELEMENT configuration (mappers?)+ >

<!ELEMENT mappers (mapper*)>

<!ELEMENT mapper EMPTY>
<!ATTLIST mapper
resource CDATA #IMPLIED
url CDATA #IMPLIED
class CDATA #IMPLIED
>
```

问题：

1、这样一个一个类来指定，好繁琐？能不能指定一个包名，包含包下所有接口、子孙包下的接口类？

2、包含包下所有的接口，好像不是很灵活，能不能让用户指定包下所有某类型的接口？

如是什么类型的类，或带有某注解的接口。

好的，这很容易，在mappers元素中增加一个package元素，pacakge元素定义三个属性

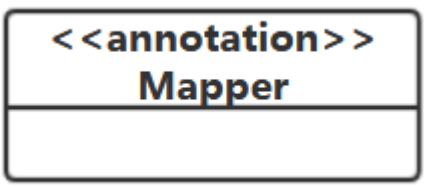
mybatis-config.dtd

```
<!ELEMENT configuration (mappers?)+ >

<!ELEMENT mappers (mapper*,package*)>

<!ELEMENT mapper EMPTY>
```

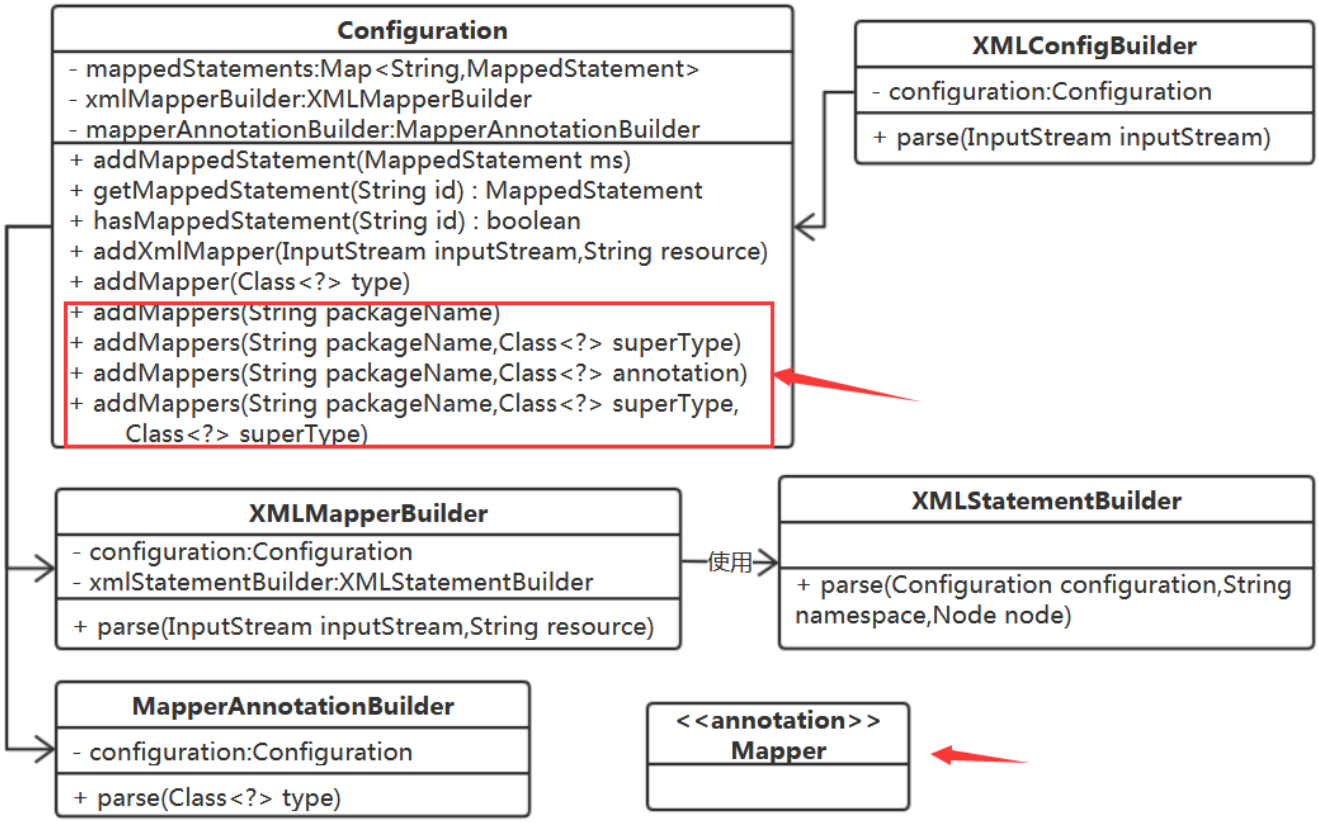
```
<!ATTLIST mapper
resource CDATA #IMPLIED
url CDATA #IMPLIED
class CDATA #IMPLIED
>

<!ELEMENT package EMPTY>
<!ATTLIST package
name CDATA #IMPLIED
type CDATA #IMPLIED
annotation CDATA #IMPLIED
>
```

```
<configuration>
        <mappers>
                <mapper resource="com/mike/UserMapper.xml"/>
        <mapper url="file:///var/mappers/CourseMapper.xml"/>
        <mapper class="com.study.mike.dao.UserDao" />
        <package name="com.study.mike.mapper" />
        <package name="com.study.mike.mapper"
type="com.study.mike.MapperInterface"/>
        <package name="com.study.mike.mapper"
                 annotation="com.study.mike.mybtis.annotation.Mapper"/>
        <package name="com.study.mike.mapper"
type="com.study.mike.MapperInterface"
                 annotation="com.study.mike.mybtis.annotation.Mapper"/>
        <mappers>
</configuration>
```

为了用户使用方便，我们给定义一个@Mapper注解，默认规则：指定包下加了@Mapper注解的接口，如何？



加了package元素，又得在Configuration中增加对应的方法了：



补充一条，**约定俗成的规则：指定包下扫到的@Mapper接口，例如UserDao，还可以在包下定义 UserDao.xml SQL定义文件，会被加载解析。**

2.2 需求2

需求2、用户需指明接口方法的参数与语句参数的对应关系。

2.2.1 语句参数指定

看下面的Mapper示例

```
@Mapper
public interface UserDao {

    @Insert("insert into t_user(id,name,sex,age) values(?,?,?,?)")
    void addUser(User user);
}
```

User对象的属性如何与 values(?)对应？

靠解析 t_user(id,name,sex,age) 可行吗？

难度太大！

万一User的name叫xname呢！

既然靠我们来解析不行，那就请用户指明吧。用户如何来指明呢？

我们来给定个**规则：`?` 用 `#{属性名}` 代替，我们来解析SQL语句中的`#{属性名}`来决定参数对应。**

```
@Insert("insert into t_user(id,name,sex,age) values(#{id},#{name},#
{sex},#{age})")
void addUser(User user);
```

万一是这种情况呢？

```
@Insert("insert into t_user(id,name,sex,age) values(#{id},#{name},#
{sex},#{age})")
void addUser(String id,String name,String sex,int age);
```

这样可以吗？能拿到方法参数的名称吗？

1. JDK版本必须是1.8及以上
2. 编译时候必须有编译选项：javac -parameters打开，默认是关闭的

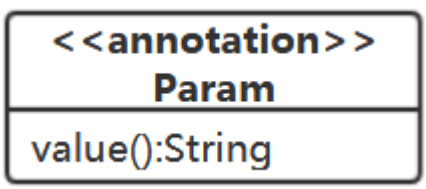
不一定能拿到（Parameter.getName()）。咋办？

方法一：序号参数，拿不到名称，但它是有顺序的，用户可以这样来写他们的SQL:

```
@Insert("insert into t_user(id,name,sex,age) values(#{0},#{1},#{2},#
{3})")
void addUser(String id,String name,String sex,int age);
```

可以吗？

方法二：注解指明名称，定义一个注解让用户使用，该注解只可用在参数上

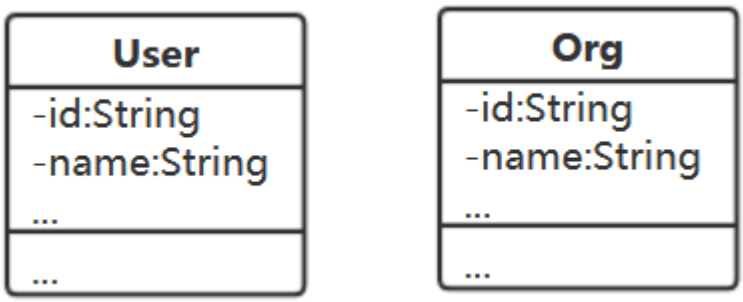


```
@Insert("insert into t_user(id,name,sex,age) values(#{id},#{name},#
{sex},#{age})")
void addUser(@Param("id")String id,@Param("name")String
name,@Param("sex")String sex,@Param("age")int age);
```

万一是这种情况呢？

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{id},#
{name},#{sex},#{age},#{id})")
void addUser(User user,Org org);
```

User和Org中都有id属性，name属性



好办，**如果方法参数是对象，则以`参数名.属性.属性`的方式指定SQL参数**：

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{user.id},#
{user.name},#{user.sex},#{user.age},#{org.id})")
void addUser(User user,Org org);
```

一样也可以使用@Param

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{user.id},#
{user.name},#{user.sex},#{user.age},#{org.id})")
void addUser(@Param("user")User user,@Param("org")Org org);
```

也可以使用参数序号：

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{0.id},#
{0.name},#{0.sex},#{0.age},#{1.id})")
void addUser(User user,Org org);
```

如果方法参数是这种情况呢?

```
@Insert("insert into t_user(id,name,sex,age) values(#{id},#{name},#
{sex},#{age})")
void addUser(Map map);
```

对应Map中的key

如果方法参数是这种情况呢?

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{user.id},#
{user.name},#{user.sex},#{user.age},#{org.id})")
void addUser(Map map,Org org);
```

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{user.id},#
{user.name},#{user.sex},#{user.age},#{org.id})")
void addUser(@Param("user")Map map,@Param("org")Org org);
```

再来看下下面的场景：

```
@Select("select id,name,sex from t_user where sex = #{sex} order by #
{orderColumn}")
List<User> query(String sex, String orderColumn);
```

`order by #{orderColumn}` `order by ?` 可以吗？

不可以，也就是说 方法参数不全是用来做SQL语句的预编译参数值的，有些是来构成SQL语句的一部分的。

那怎么让用户指定呢？

一样，定义个**规则：`${属性名}` 表示这里是字符串替换**

```
@Select("select id,name,sex from t_user where sex = #{sex} order by
${orderColumn}")
List<User> query(String sex, String orderColumn);
```

脑袋都快想炸了，先考虑这么多吧！

来考虑考虑怎么解析SQL中的参数映射吧！

2.2.2 SQL中参数映射解析

问题：

1、SQL中参数映射解析要完成的是什么工作？

- 解析出真正的SQL语句
- 获得方法参数与语句参数的对应关系 ： 问号N---哪个参数值

```
// 新增用户保存到数据库
    public void addUser(User user) throws SQLException {
        try (
                // 1、获取连接
                Connection conn = dataSource.getConnection();
                // 2、创建预编译语句对象
                PreparedStatement pst = conn.prepareStatement(
                        "insert into
t_user(id,name,sex,age,address,phone,wechat,email,account,password) "
    " values(?,?,?,?,?,?,?,?,?,?)");)
        {
            // 3、设置参数值
            int i = 1;
            pst.setString(i++, user.getId());
            pst.setString(i++, user.getName());
            pst.setString(i++, user.getSex());
            pst.setInt(i++, user.getAge());
            pst.setString(i++, user.getAddress());
            pst.setString(i++, user.getPhone());
            pst.setString(i++, user.getWechat());
            pst.setString(i++, user.getEmail());
            pst.setString(i++, user.getAccount());
            pst.setString(i++, user.getPassword());

        // 4、执行语句
        int changeRows = pst.executeUpdate();
        }
}
```

怎么解析？

```
@Insert("insert into t_user(id,name,sex,age,org_id) values(#{user.id},#
{user.name},#{user.sex},#{user.age},#{org.id})")
void addUser(@Param("user")Map map,Org org);
```

方式有：

正则表达式

antlr

怎么表示？

问号的index、值来源

2、这个解析的工作在何时做好？谁来做好？

还不清楚！