

02-手写Mybatis-2

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

2 设计

2.3 需求4




- 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。**

2.3.1

SQL语句有了，参数对应关系也有了，我们想想怎么执行SQL吧。

问题：

- 1、**要执行SQL，我们得要有DataSource，谁来持有DataSource？**

Configuration
<ul style="list-style-type: none"> - mappedStatements:Map<String,MappedStatement> - xmlMapperBuilder:XMLMapperBuilder - mapperAnnotationBuilder:MapperAnnotationBuilder - dataSource:DataSource 
<ul style="list-style-type: none"> + addMappedStatement(MappedStatement ms) + getMappedStatement(String id) : MappedStatement + hasMappedStatement(String id) : boolean + addXmlMapper(InputStream inputStream,String resource) + addMapper(Class<?> type) + addMappers(String packageName) + addMappers(String packageName,Class<?> superType) + addMappers(String packageName,Class<?> annotation) + addMappers(String packageName,Class<?> superType, Class<?> superType) + setDataSource(DataSource dataSource)  + getDataSource():DataSource 

2、谁来执行SQL? Configuration ?

不合适，它是配置对象，持有所有配置信息！

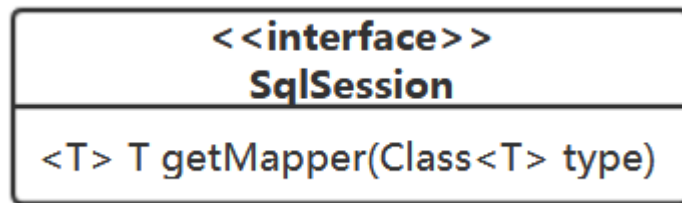
既然是来做事的，那就先定义一个接口吧：SqlSession



该为它定义什么方法呢？

需求4、框架完成接口对象的生成，JDBC执行过程。

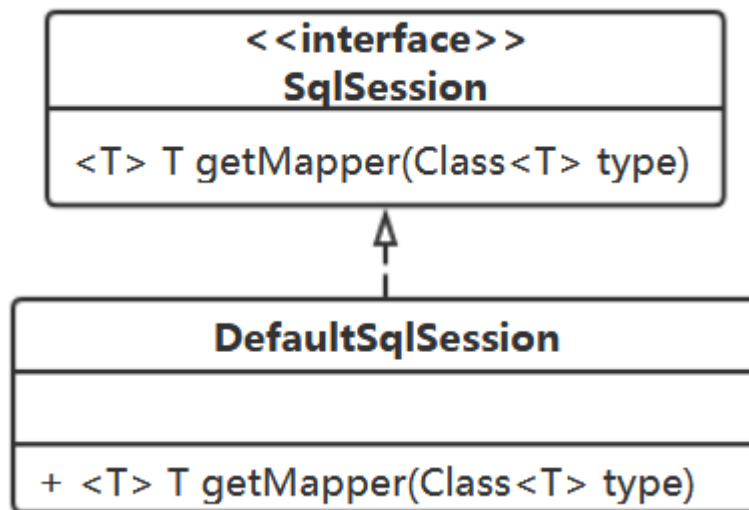
用户给定Mapper接口类，要为其生成对象，用户再使用这个对象完成对应的数据库操作。



使用示例：

```
 UserDao userDao = sqlSession.getMapper(UserDao.class);
 userDao.addUser(user);
```

来为它定义一个实现类：DefaultSqlSession



它该持有什么属性吗？

用户给入一个接口类，DefaultSqlSession中就为它生成一个对象？

万一给入的不是一个Mapper接口呢？

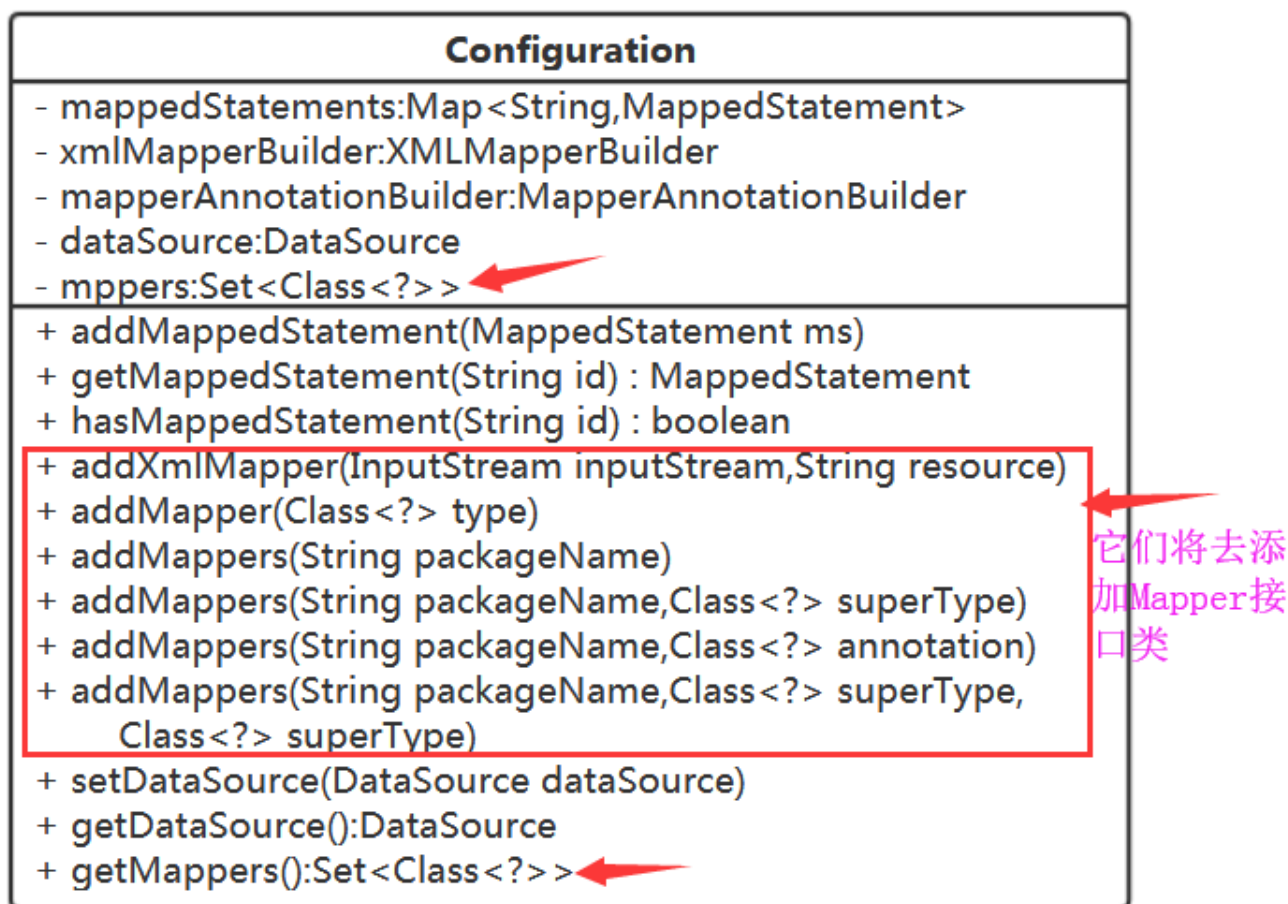
也为其生成一个对象就不合理了？

那怎么判断给入的接口类是否是一个Mapper接口呢？

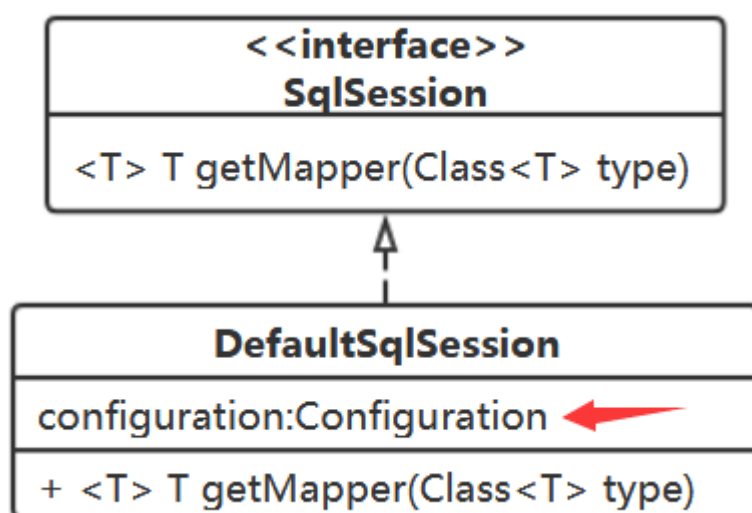
那就只有在配置阶段扫描、解析Mapper接口时做个存储了。

存哪，用什么存？

这也是配置信息，还是存在Configuration 中，就用个Set来存吧。



DefaultSqlSession中需要持有Configuration



2.3.2 对象生成

1、如何为用户给入的Mapper接口生成对象？

很简单，JDK动态代理

```
Proxy.newProxyInstance(mapperInterface.getClassLoader(), new Class[] {  
    mapperInterface }, invocationHandler);
```

写一版DefaultSqlSession的实现：

```
package com.study.mike.mybatis.session;  
  
import java.lang.reflect.InvocationHandler;  
import java.lang.reflect.Proxy;  
  
import com.study.mike.mybatis.config.Configuration;  
  
public class DefaultSqlSession implements SqlSession {  
  
    private Configuration configuration;  
  
    public DefaultSqlSession(Configuration configuration) {  
        super();  
        this.configuration = configuration;  
    }  
  
    @Override  
    public <T> T getMapper(Class<T> type) {  
        //检查给入的接口  
        if (!this.configuration.getMappers().contains(type)) {  
            throw new RuntimeException(type + " 不在Mapper接口列表中!");  
        }  
        //得到 InvocationHandler  
        InvocationHandler ih = null; // TODO 必须要有一个  
        // 创建代理对象  
        T t = (T)Proxy.newProxyInstance(type.getClassLoader(), new  
Class<?>[] {type}, ih);  
        return t;  
    }  
}
```

问题：每次调用getMapper(Class type)都需要生成一个新的实例吗？

代理对象中持有InvocationHandler，如果InvocationHandler能做到线程安全，就只需要一个实例。

还得看InvocationHandler，先放这吧，把InvocationHandler搞定先。

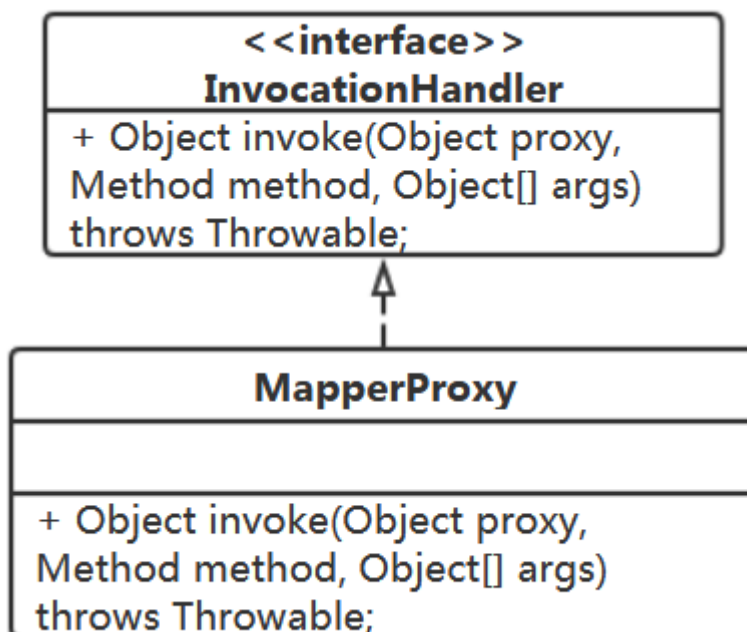
2.3.3 执行SQL的InvocationHandler

了解 InvocationHandler

InvocationHandler 是在代理对象中完成增强。我们这里通过它来执行SQL。

```
public interface InvocationHandler {  
    /**  
    @param proxy 生成的代理对象  
    @param method 被调用的方法  
    @param args  
    @return Object 方法执行的返回值  
    */  
    public Object invoke(Object proxy, Method method, Object[] args)  
        throws Throwable;  
}
```

来实现我们的InvocationHandler：MapperProxy



```

package com.study.mike.mybatis.session;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;

public class MapperProxy implements InvocationHandler {

    @Override
    public Object invoke(Object proxy, Method method, Object[] args)
    throws Throwable {
        // TODO 这里需要完成哪些事？
        return null;
    }
}

```

思考：在 MapperProxy.invoke方法中需要完成哪些事？

```

package com.study.mike.mybatis.session;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;

public class MapperProxy implements InvocationHandler {

    @Override
    public Object invoke(Object proxy, Method method, Object[] args)
    throws Throwable {
        // TODO 这里需要完成哪些事？
        // 1、获得方法对应的SQL语句

        // 2、解析SQL参数与方法参数的对应关系，得到真正的SQL与语句参数值

        // 3、获得数据库连接

        // 4、执行语句

        // 5、处理结果

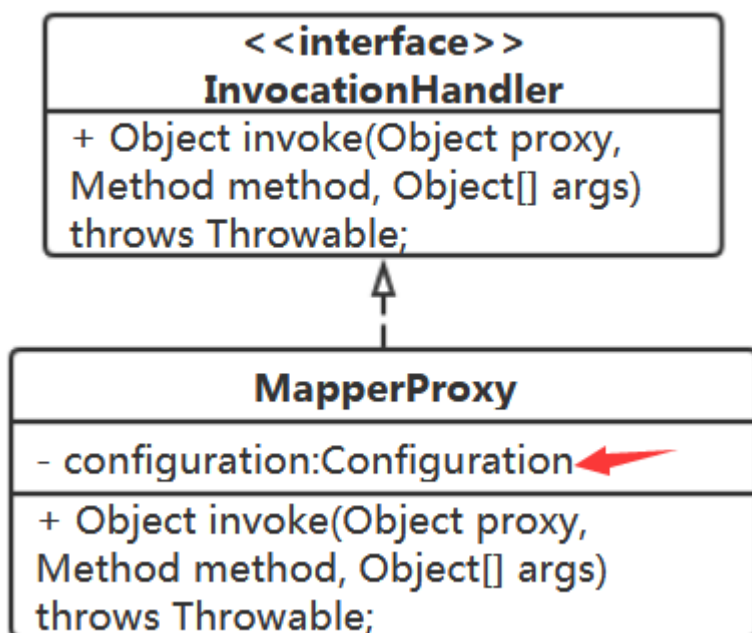
        return null;
    }
}

```

```
}
```

1、获得方法对应的SQL语句

要获得SQL语句，需要用到Configuration，MapperProxy中需持有Configuration。



Configuration
<ul style="list-style-type: none"> - mappedStatements:Map<String,MappedStatement> - xmlMapperBuilder:XMLMapperBuilder - mapperAnnotationBuilder:MapperAnnotationBuilder - dataSource:DataSource - mappers:Set<Class<?>>
<ul style="list-style-type: none"> + addMappedStatement(MappedStatement ms) + getMappedStatement(String id) : MappedStatement + hasMappedStatement(String id) : boolean + addXmlMapper(InputStream inputStream,String resource) + addMapper(Class<?> type) + addMappers(String packageName) + addMappers(String packageName,Class<?> superType) + addMappers(String packageName,Class<?> annotation) + addMappers(String packageName,Class<?> superType, Class<?> superType) + setDataSource(DataSource dataSource) + getDataSource():DataSource + getMappers():Set<Class<?>>

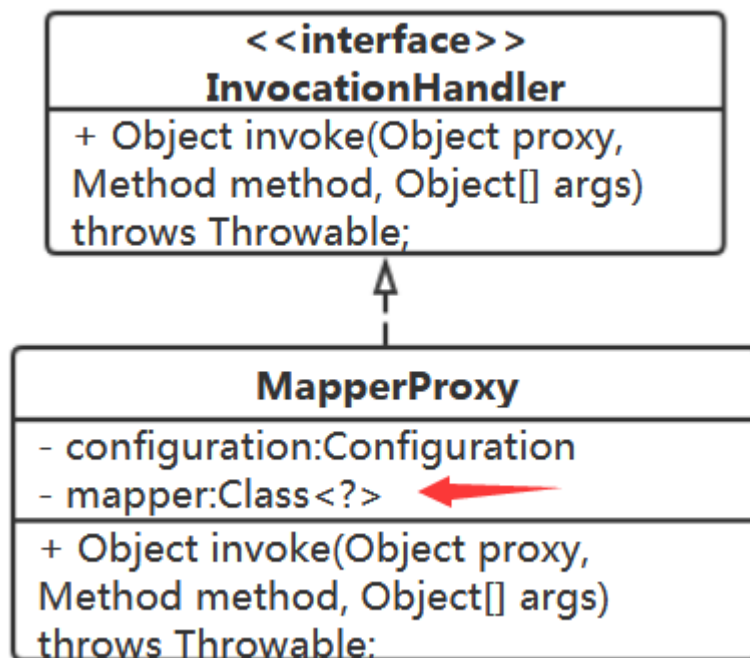
问题：id怎么得来？

id是类名.方法名。从invoke方法的参数中能得来吗？

```
public Object invoke(Object proxy, Method method, Object[] args)
```

method参数能得到方法名，但得到的类名不是Mapper接口类名。

那就直接让MapperProxy持有其增强的Mapper接口类吧！简单！



2、解析SQL参数与方法参数的对应关系，得到真正的SQL与语句参数值

逻辑：

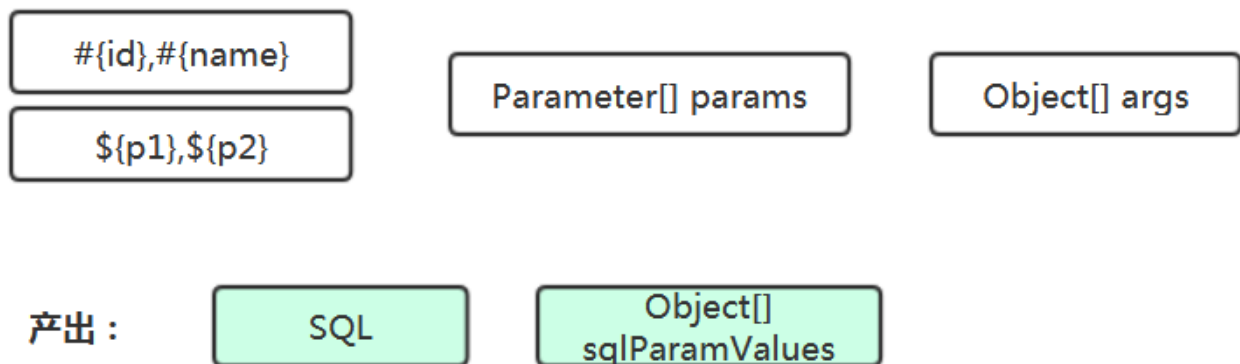
- 1、查找SQL语句中的 `#{属性}`，确定是第几个参数，再在方法参数中找到对应的值，存储下来，替换 `#{属性}` 为？。
- 2、查找SQL语句中的 `${属性}`，确定是哪个参数，再在方法参数中找到对应的值，替换 `${属性}`。
- 3、返回最终的SQL与参数数组。

解析过程涉及的数据：SQL语句、方法的参数定义、方法的参数值

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

```
Parameter[] params = method.getParameters();
```

```
public Object invoke(Object proxy, Method method, Object[] args)
```



说明：这里是要确定SQL中的？N是哪个参数值。

这里要分三种情况：方法参数是0参数，单个参数、多个参数。

0参数：不需要考虑参数了。

单个参数：SQL中的参数取值参数的属性或就是参数本身。

多个参数：则需要确定SQL中的参数值取第几个参数的值。

多个参数的情况，可以有两种做法：

方式一：查找#{属性}，根据Parameter[]中的名称（注解名、序号）匹配确定是第几个参数，再到 Object[] args中取到对应的值。

方式二：先将Parameter[] 和 Object[] args转为Map，参数名（注解名、序号）为key，Object参数值为值；然后再查找SQL语句中的 #{ } \$ { }，根据里面的名称到map中取对应的值。

哪种方式更好呢？

我们来看下两者查找过程的输出：

方式一：

？	对应值
？N	参数索引号

方式二：

？	对应值
？N	属性名或注解名或参数索引号

方式一相较于方式二，看起来复杂的地方是要遍历Parameter[] 来确定索引号。

但请同学们思考一下：这种查找对应关系的事，需要每次调用方法时都做吗？方法的参数会有很多个吗？

这个对应关系可以在扫描解析Mapper接口时做一次即可。在调用Mapper代理对象的方法时，就可以直接根据索引号去Object[] args中取参数值了。

而 则每次调用Mapper代理对象的方法时，都需要创建转换Map。

而且方式一，单个参数与多个参数我们可以同样处理。

要在扫描解析Mapper接口时做参数解析我们就需要定义对应的存储结构，及修改MappedStatement了

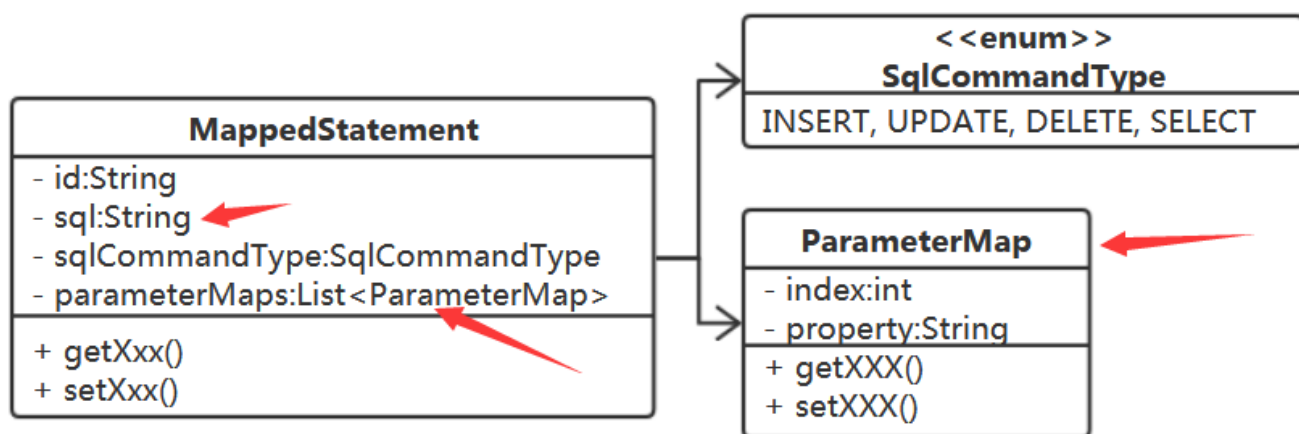
？N--- 参数索引号 的对应关系如何表示？

？N 就是一个数值，而且是一个顺序数（只是jdbc中的？是从1开始）。我们完成可以用List来存储。

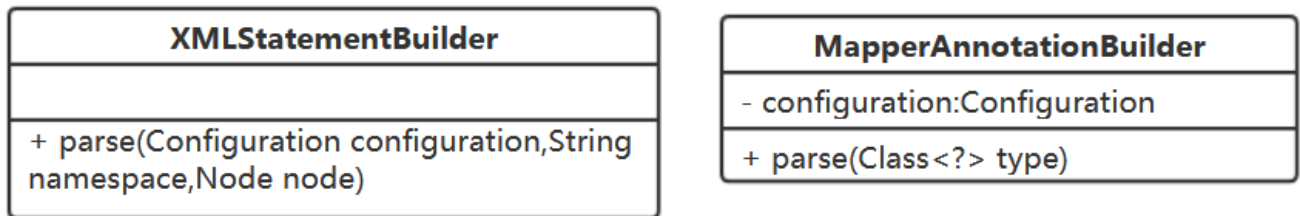
参数索引号，仅仅是个索引号吗？

```
@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);
```

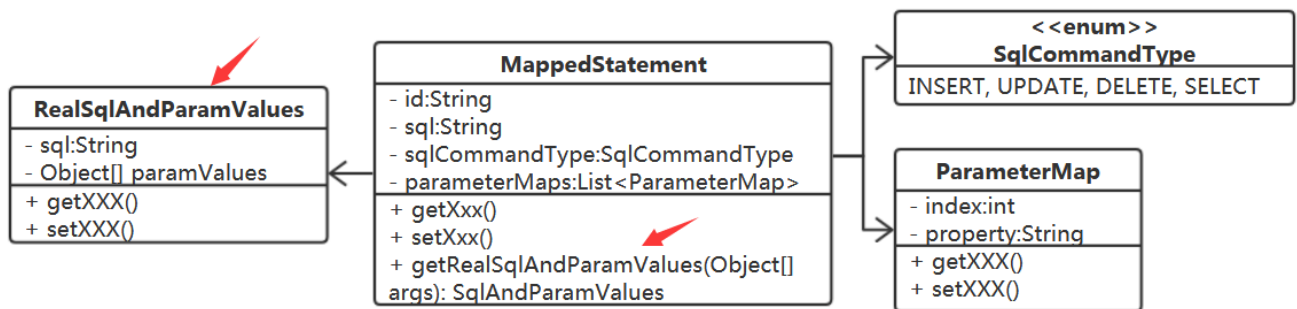
它应该是索引号、和里面的属性两部分。



解析阶段有它们俩完成这件事：



我们在MappedStatement中再增加一个方法来完成根据参数映射关系得到真正参数值的方法：



把MapperProxy的invoke方法填填看：

```

@Override
public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
    // TODO 这里需要完成哪些事？
    // 1、获得方法对应的SQL语句
    String id = this.mapper.getName() + "." + method.getName();
    MappedStatement ms = this.configuration.getMappedStatement(id);
    // 2、解析SQL参数与方法参数的对应关系，得到真正的SQL与语句参数值
    RealSqlAndParamValues rsp = ms.getRealSqlAndParamValues(args);
    // 3、获得数据库连接
    Connection conn =
this.configuration.getDataSource().getConnection();
    // 4、执行语句。
    PreparedStatement pst = conn.prepareStatement(rsp.getSql());
    // 疑问：语句一定是PreparedStatement？
    // 设置参数
    if (rsp.getParamValues() != null) {
        int i = 1;
        for (Object p : rsp.getParamValues()) {
            pst.setxxx(i++, p); //这里写不下去了.....如何决定该调用pst的哪
个set方法？
        }
    }
}
  
```

```
}  
// 5、处理结果  
  
return null;  
}
```

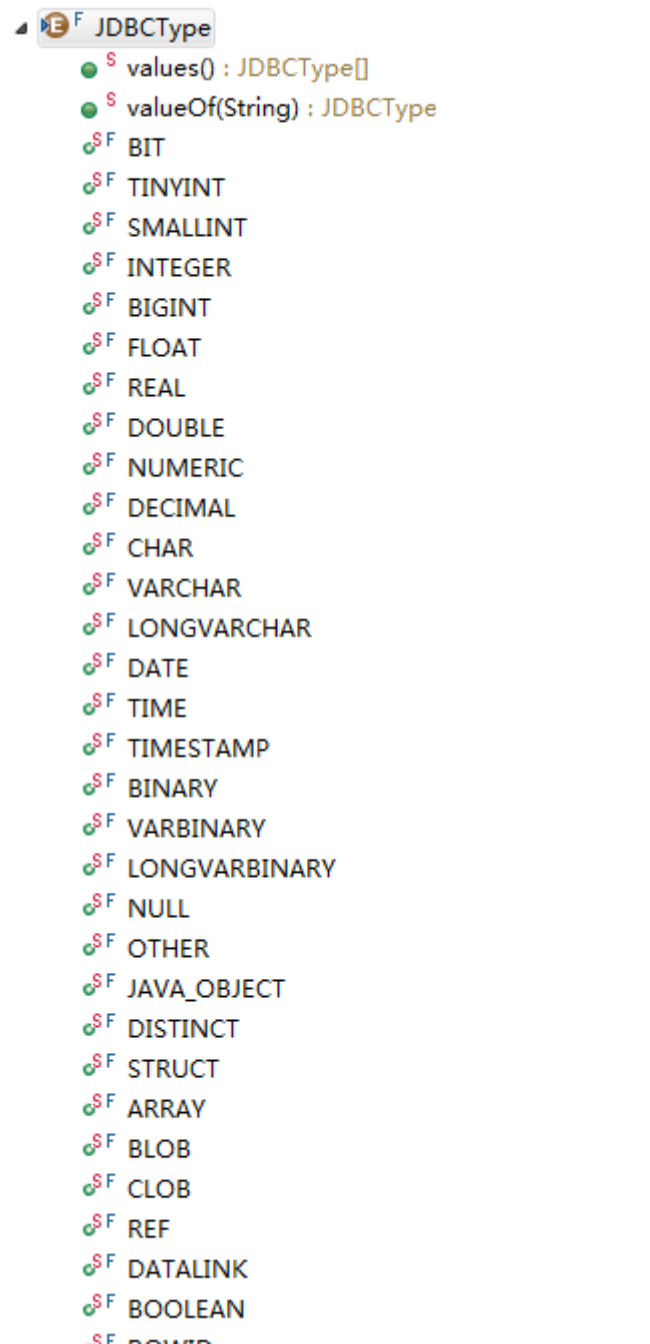
2.4 JavaType、JdbcType转换

2.4.1 认识它们

JavaType：java中的数据类型。

JdbcType：Jdbc规范中根据数据库sql数据类型定义的一套数据类型规范，各数据库厂商遵照这套规范来提供jdbc驱动中数据类型支持。

```
package java.sql;  
  
/**  
 * <P>Defines the constants that are used to identify  
 * SQL types, called JDBC types.  
 * <p>  
 * @see SQLType  
 * @since 1.8  
 */  
public enum JDBCType implements SQLType {  
  
    /**  
     * Identifies the generic SQL type {@code BIT}.  
     */  
    BIT(Types.BIT),  
    /**
```



请看看PreparedStatement的set方法

疑问：为什么我们在这里需要考虑它呢？

pst.setxxx(i++, p)，我们不能根据p的类型选择set方法吗？

你看pst的set方法中与对应的：

- ❶ PreparedStatement
 - A setNull(int, int) : void
 - A setBoolean(int, boolean) : void
 - A setByte(int, byte) : void
 - A setShort(int, short) : void
 - A setInt(int, int) : void
 - A setLong(int, long) : void
 - A setFloat(int, float) : void
 - A setDouble(int, double) : void
 - A setBigDecimal(int, BigDecimal) : void
 - A setString(int, String) : void
 - A setBytes(int, byte[]) : void
 - A setDate(int, Date) : void
 - A setTime(int, Time) : void
 - A setTimestamp(int, Timestamp) : void

我们判断p的类型，然后选择不可吗？像下面这样

```
int i = 1;
for (Object p : rsp.getParamValues()) {
    if (p instanceof Byte) {
        pst.setByte(i++, (Byte) p);
    } else if (p instanceof Integer) {
        pst.setInt(i++, (int) p);
    }
    else if (p instanceof String) {
        pst.setString(i++, (String) p);
    }
    ...
    else if(...)
}
```

我们来看一下这种情况：

字段	SQL数据类型(mysql)	JDBCType	JavaType
start_time	Date	DATE(Types.DATE)	java.util.Date
start_time	Time	TIME(Types.TIME)	java.util.Date
start_time	Timestamp	TIMESTAMP(Types.TIMESTAMP)	java.util.Date

还请看PreparedStatement的set方法：

- A setAsciiStream(int, InputStream, int) : void
 - A setUnicodeStream(int, InputStream, int) : void
 - A setBinaryStream(int, InputStream, int) : void
- InputStream怎么选择？

- ^A setNCharacterStream(int, Reader, long) : void
- ^A setNClob(int, NClob) : void
- ^A setClob(int, Reader, long) : void
- ^A setBlob(int, InputStream, long) : void
- ^A setNClob(int, Reader, long) : void

Reader时怎么选？

❶ PreparedStatement

- ^A setObject(int, Object, int) : void
- ^A setObject(int, Object) : void
- ^A setObject(int, Object, int, int) : void
- ^D setObject(int, Object, SQLType, int) : void
- ^D setObject(int, Object, SQLType) : void

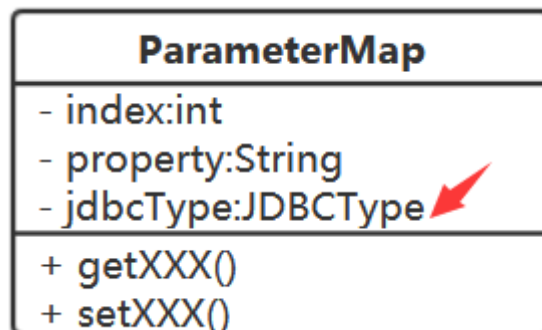
java对象只能存为Object吗？
我想将对象存为json字符串、xml字符串可以吗？

上面前两种情况怎么处理？

这个需要用户说明其要使用的JDBCType，不然鬼知道他想要什么。

让用户怎么指定呢？

```
# {user.id, jdbcType=TIME}
```



javaType有需要指定不呢？

好像不需要，那就暂放下。

第3种情况怎么处理？

这是一个未知的问题，鬼知道将来使用我的框架的人会需要怎么处理他们的对象呢！

如何以不变应万变呢？

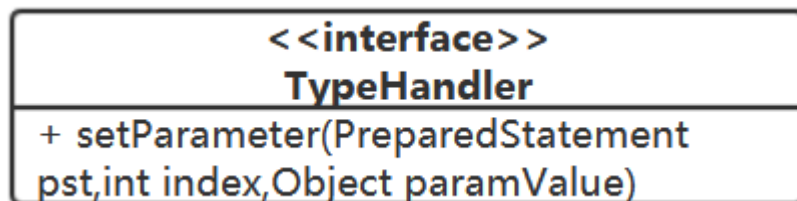
面向接口编程

定义一个什么样的接口呢？

该接口的用途是什么？

完成Object p 的pst.setXXX ()。

2.4.2 TypeHandler



下面这个if-else-if的代码是否可以通过TypeHandler，换成策略模式？

```
int i = 1;
for (Object p : rsp.getParamValues()) {
    if (p instanceof Byte) {
        pst.setByte(i++, (Byte) p);
    } else if (p instanceof Integer) {
        pst.setInt(i++, (int) p);
    }
    else if (p instanceof String) {
        pst.setString(i++, (String) p);
    }
    ...
    else if(...)
}
```

定义一些常用数据类型的TypeHandler.

先不急着去定义，我们来考虑一下下面的问题。

这个怎么使用它呢？

在MapperProxy.invoke()中？

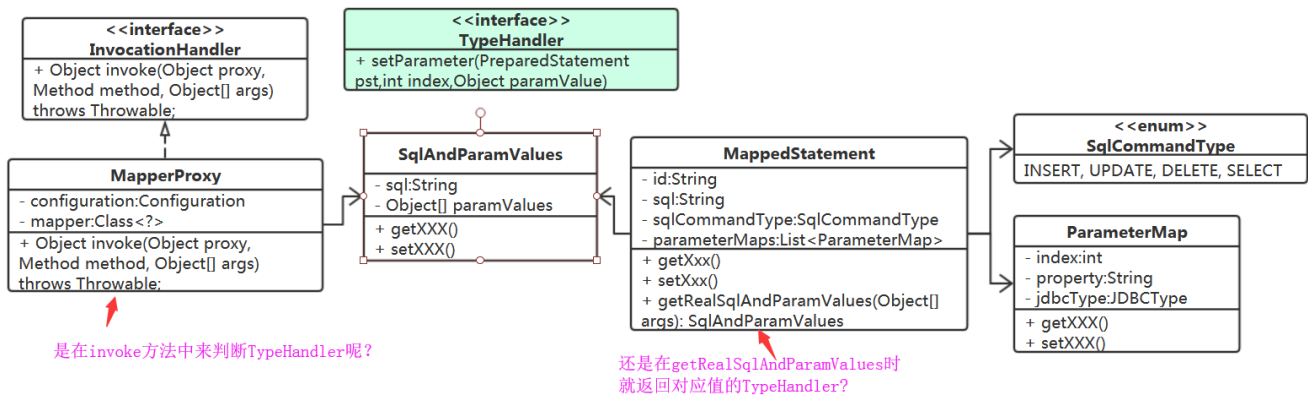
```
int i = 1;
for (Object p : rsp.getParamValues()) {
    TypeHandler th = getTypeHandler(p.getClass); //还需要别的参数吗？
    th.setParameter(pst,i++,p);
}
```

还需要JDBCType。

```

int i = 1;
for (Object p : rsp.getParamValues()) {
    TypeHandler th = getTypeHandler(p.getClass, jdbcType);
    //jdbcType 从哪来?
    th.setParameter(pst, i++, p);
}

```

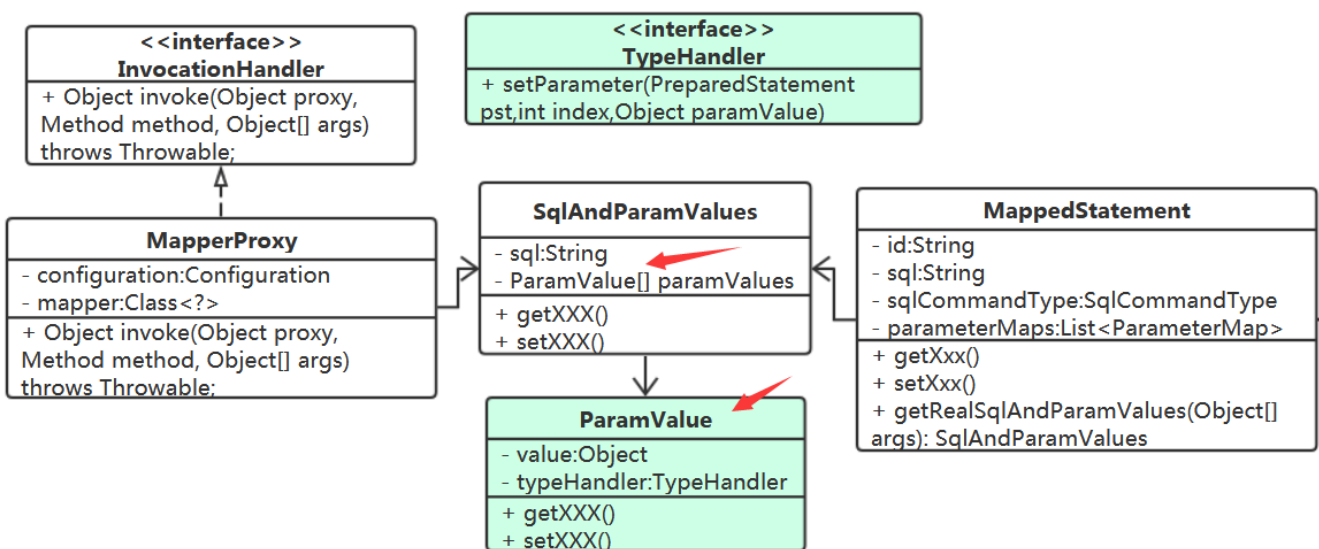


问题：

1、是在invoke方法中来判断TypeHandler呢？还是在MappedStatement的getRealSqlAndParamValues时就返回值对应的TypeHandler？

选择后者更合适！

那SqlAndParamValues中的参数值就不能是Object[]。它是值和TypeHandler两部分构成。



MapperProxy中的代码就变成下面这样了：

```

int i = 1;
for (ParamValue p : rsp.getParamValues()) {
    TypeHandler th = p.getTypeHandler()
    th.setParameter(pst,i++,p.getValue());
}

```

2、MappedStatement又从哪里去获取TypeHandler呢？

我们会定义一些常用的，用户可能会提供一些。用户怎么提供？存储到哪里？

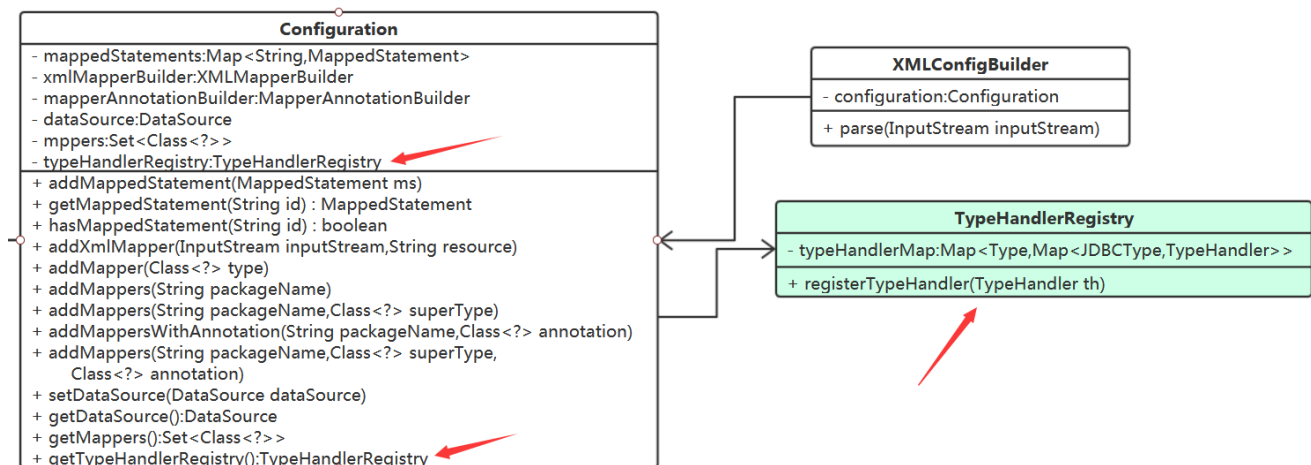
Configuration 吧，它最合适了。以什么结构来存储呢？

这里涉及到查找，需要根据 参数的javaType、jdbcType来查找。

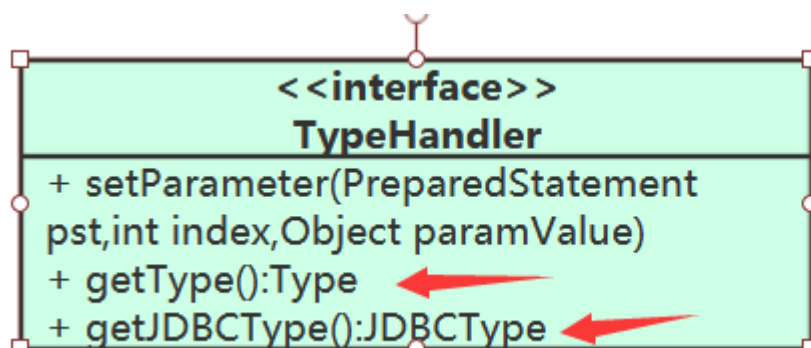
那就定义一个map吧，如下这样如何？

```
Map<Type,Map<JDBCType,TypeHandler>> typeHandlerMap;
```

我们定义一个TypeHandlerRegistry类来持有所有的TypeHandler,Configuration 中则持有TypeHandlerRegistry。



同时我们完善一下TypeHandler



3、用户如何来指定它们的TypeHandler?

在mybatis-config.xml中增加一个元素来让用户指定吧。

mybatis-config.dtd

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

<!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
>

<!ELEMENT typeHandlers (typeHandler*,package*)>

<!ELEMENT typeHandler EMPTY>
<!ATTLIST typeHandler
class CDATA #REQUIRED
>
```

既可以用typeHandler指定单个，也可用package指定扫描的包，扫描包下实现了TypeHandler接口的类。

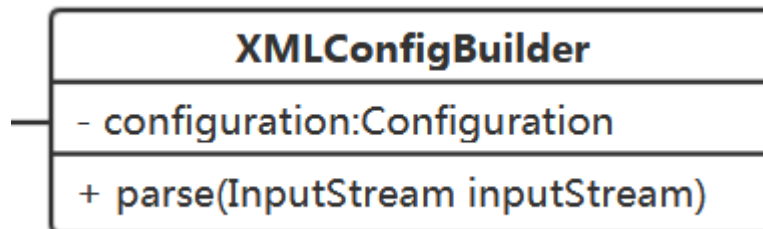
mybatis-config.xml

```

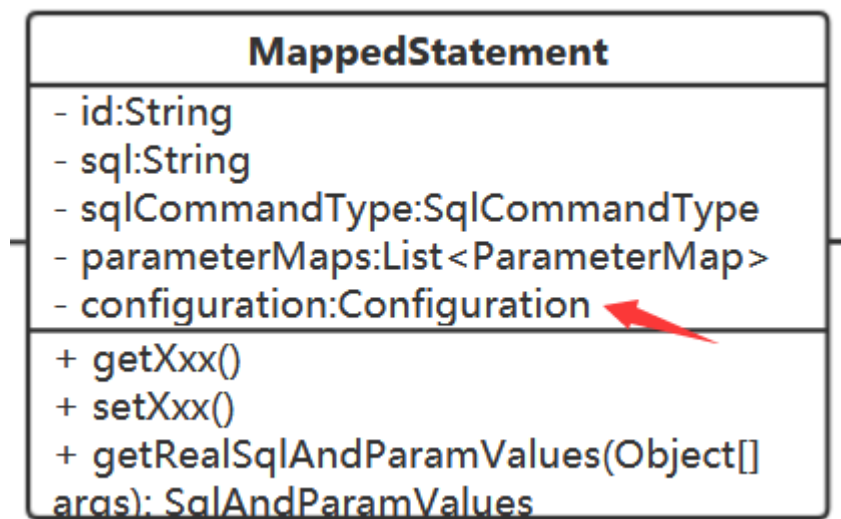
<configuration>
  <mappers>
    <mapper resource="com/mike/UserMapper.xml"/>
    <mapper url="file:///var/mappers/CourseMapper.xml"/>
    <mapper class="com.study.mike.dao.UserDao" />
    <package name="com.study.mike.mapper" />
  </mappers>
  <typeHandlers>
    <typeHandler class="com.study.mike.type.XoTypeHandler" />
    <package name="com.study.mike.type" />
  </typeHandlers>
</configuration>

```

解析注册的工作就交给XMLConfigBuilder

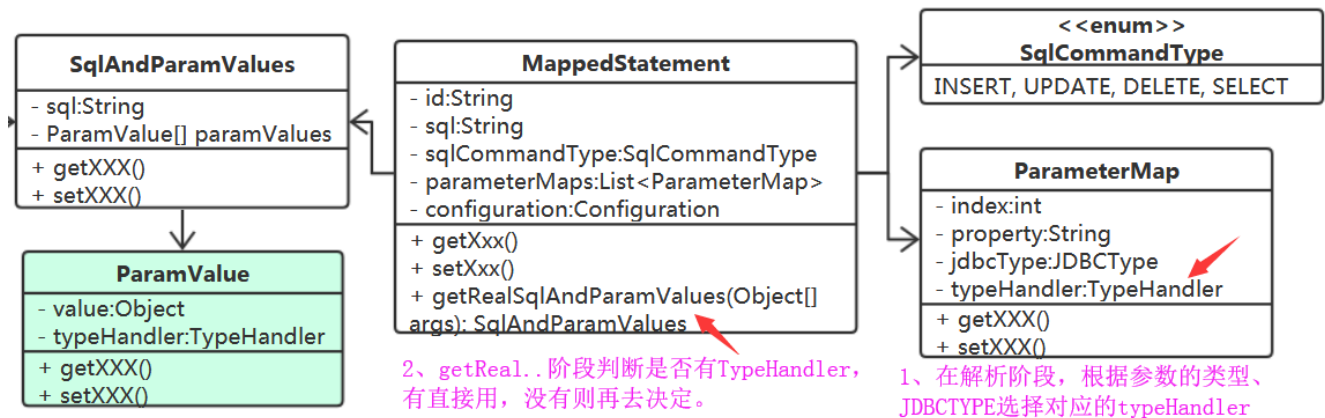


4、MappedStatement中来决定TypeHandler,它就需要Configuration



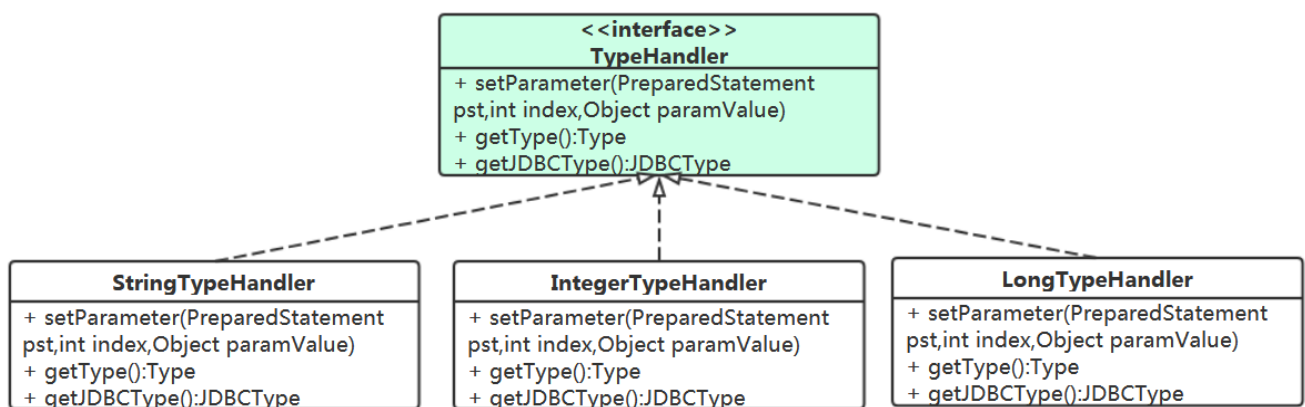
5、可不可以在解析语句参数关系时，就决定好TypeHandler?

可以。我们在ParameterMap中增加typeHandler属性。



用户在SQL语句参数中必须要指定JDBCTYPE吗？

常用的数据类型可以不指定，我们可以提供默认的类型Handler。



```
public class StringTypeHandler implements TypeHandler {

    @Override
    public Type getType() {
        return String.class;
    }

    @Override
    public JDBCType getJDBCType() {
        return JDBCType.VARCHAR;
    }

    @Override
    public void setParameter(PreparedStatement pst, int index, Object paramValue) throws SQLException {
```

```
        pst.setString(index, (String) paramValue);
    }

}
```

用户在SQL中参数定义没有指定DBCType，则我们可以直接使用我们默认的TypeHandler

如 #{user.name}

我们判断它的参数类型为String，就可以指定它的TypeHandler为 StringTypeHandler。可能它的数据库类型不为VACHAR，而是一个CHAR定长字符，没关系！因为pst.setString对 VARCHAR、CHAR是通用的。

2.5 执行结果处理

2.5.1 执行结果处理要干的是什么事

pst.executeUpdate()的返回结果是int，影响的行数。

pst.executeQuery()的返回结果是ResultSet。

在得到语句执行的结果后，要转为方法的返回结果进行返回。这就是执行结果处理要干的事
根据方法的返回值类型来进行相应的处理。

这里我们根据语句执行结果的不同，分开处理：

1、pst.executeUpdate()的返回结果是int。

方法的返回值可以是什么？

void、int、long，其他的不可以！

2、pst.executeQuery()的返回结果是ResultSet

方法的返回值可以是什么？

可以是void、单个值、集合。

单个值可以是什么类型的值？

任意值

集合可以是什么类型？

List、Set

集合的元素可以是什么类型的？

任意类型的

结果集中的列如何与结果、结果的属性对应？

根据结果集列名与属性名对应

如果属性名与列名不一样呢？

则需用户显式说明映射规则。

需要考虑JDBCType --- JavaType的处理吗？

2.6 不同的语句类型

Statement

PreparedStatement

CallableStatement