```
1. Mybatis 是一个相对于全自动化 Hibernate 的半自动化 ORM 框架;
2. 下载地址: https://github.com/mybatis/mybatis-3/releases
MyBatis 3 官方简介 http://www.mybatis.org/mybatis-3/zh/index.html
3. 在 Web 项目中使用 Mybatis, 至少引入两个 jar 包到 lib 目录 下:
   mybatis-3.4.1. jar
   mysql-connector-java-5.1.39. jar
4. 和 Hibernate 类似,Mybatis 也有注解和 XML 映射的配置方式。
    也要有核心配置 XML 文件, 见附录。
5. 定义返回结果映射:
   <!-- 返回结果对象属性与表字段映射 -->
    <resultMap type="UserInfo" id="resultListUserInfo">
    <id column="id" property="id" />
    <result column="username" property="username" />
    <result column="password" property="password" />
    </resultMap>
    <!-- 查询所有记录 -->
6. CRUD: ----XML 方式:
    <!-- 根据 id 查询得到一个 User Info 对象 -->
    <select id="getUserInfoById" parameterType="Integer"</pre>
resultType="com.ddb.mybatis.domain.UserInfo">
    select username, password from user info where id=#{id:INTEGER}
    </select>
    <!-- 插入数据库表数据 -->
    <insert id="insertUserInfo"</pre>
parameterType="com. ddb. mybatis. domain. UserInfo">
    insert into user_info(username, password) value (#{username}, #{password})
    </insert>
    <!-- 更新数据库表数据 -->
    <update id="updateUserInfo"</pre>
parameterType="com. ddb. mybatis. domain. UserInfo">
   update user_info set
username=#{username:VARCHAR}, password=#{password:VARCHAR} where
id=#{id:INTEGER}
   </update>
    <!-- 根据 id 删除数据库表数据 -->
    <delete id="deleteUserInfo" parameterType="Integer">
   delete from user info where id=#{id:INTEGER}
    </delete>
7. CRUD: ---注解方式
Insert 语句映射
public interface StudentMapper
@Insert("INSERT INTO STUDENTS(STUD_ID, NAME, EMAIL, ADDR_ID, PHONE)
VALUES(#{studId}, #{name}, #{email}, #{address. addrId}, #{phone})")
```

```
int insertStudent(Student student);
Insert 自动生成主键
@Insert ("INSERT INTO STUDENTS (NAME, EMAIL, ADDR ID, PHONE)
VALUES (#{name}, #{email}, #{address. addrId}, #{phone})")
@Options(useGeneratedKeys = true, keyProperty = "studId")
int insertStudent(Student student);
Update 语句映射
@Update("UPDATE STUDENTS SET NAME=#{name}, EMAIL=#{email}, PHONE=#{phone} WHERE
STUD ID=#{studId}")
int updateStudent(Student student);
Delete 语句映射
@Delete("DELETE FROM STUDENTS WHERE STUD ID=#{studId}")
int deleteStudent(int studId);
Select 语句映射
@Select("SELECT STUD_ID AS STUDID, NAME, EMAIL, PHONE FROM STUDENTS WHERE
STUD ID=#{studId}")
Student findStudentById(Integer studId);
8. 动态 SQL: ----XML 方式
If 条件
<select id="searchCourses" parameterType="hashmap"</pre>
resultMap="CourseResult"></select>
SELECT * FROM COURSES WHERE TUTOR ID= #{tutorId}
<if test="courseName != null">
AND NAME LIKE #{courseName}
</if>
<if test="startDate != null">
AND START DATE >= #{startDate}
</if>
<if test="endDate != null">
AND END DATE <= #{endDate}
</if>
</select>
choose, when 和 otherwise 条件
<select id="searchCourses" parameterType="hashmap" resultMap="CourseResult">
SELECT * FROM COURSES
<choose>
<when test="searchBy == 'Tutor'">
WHERE TUTOR ID= #{tutorId}
<when test="searchBy == 'CourseName'">
WHERE name like #{courseName}
</when>
<otherwise>
```

```
WHERE TUTOR start_date >= now()
</orthorwise>
</choose>
</select>
Where 条件
<select id="searchCourses" parameterType="hashmap"</pre>
resultMap="CourseResult">
SELECT * FROM COURSES
<where>
<if test=" tutorId != null ">
TUTOR ID= #{tutorId}
</if>
<if test="courseName != null">
AND name like #{courseName}
</if>
<if test="startDate != null">
AND start date >= #{startDate}
</if>
<if test="endDate != null">
AND end date <= #{endDate}
</if>
</where>
</select>
trim 条件
<select id="searchCourses" parameterType="hashmap" resultMap="CourseResult">
SELECT * FROM COURSES
<trim prefix="WHERE" prefixOverrides="AND | OR">
<if test=" tutorId != null ">
TUTOR_ID= #{tutorId}
</if>
<if test="courseName != null">
AND name like #{courseName}
</if>
</trim>
</select>
foreach 循环
<select id="searchCoursesByTutors" parameterType="map"</pre>
resultMap="CourseResult">
SELECT * FROM COURSES
<if test="tutorIds != null">
<where>
tutor id IN
<foreach item="tutorId" collection="tutorIds"</pre>
open="(" separator=", " close=")">
```

```
#{tutorId}
</foreach>
</where>
</if>
</select>
set 条件
<update id="updateStudent" parameterType="Student">
update students
<set>
<if test="name != null">name=#{name},</if>
<if test="email != null">email=#{email},</if>
<if test="phone != null">phone=#{phone}, </if>
</set>
where stud id=#{id}
</update>
9. 动态 SQL: ----注解方式--参见 HTML 培训页内容
10. 在 mybatis-config.xml 中的注册映射文件或类
XML:
<mappers> <mapper
resource="letian/mybatis/mapper/UserMapper.xml"/> </mappers>
注解为:
<mappers> <mapper class="letian.mybatis.dao.UserMapper" />
</mappers>
11.和 Spring 整合:
   XML: 在会话工厂中加载所有的映射文件,见下面红色部分
   <bean id="sqlSessionFactory"</pre>
class="org.mybatis.spring.SqlSessionFactoryBean">
   cproperty name="configLocation"
value="classpath:mybatis/mybatis.cfg.xml"></property>
   property name="mapperLocations"
   value="classpath:mybatis/mapper/**/*.xml" />
   cproperty name="dataSource" ref="dataSource" />
   property name="typeAliasesPackage" value="
   com. xxx. base. data. db. domain. config. device;
   com. xxx. base. data. db. domain. upgrade; "/>
   </bean>
<!-- 配置会扫描包下的所有接口,然后创建各自接口的动态代理类。全用注解,上面的
XML 属性不配置也可以-->
   <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
   property name="basePackage"
value="com. xxx. base. data. db. mapper. *, com. xxx. core. data. db. mapper. *" />
   </bean>
```

## 12. resultType 和 resultMap 对比小结

- resultType:使用 resultType 实现较为简单,如果表字段名和对象属性名不一致查询时需要为表字段指定别名。如果没有查询结果的特殊要求建议使用 resultType。
   resultType 无法实现延迟加载。
- resultMap: 需要单独定义 resultMap,实现有点麻烦,如果对查询结果有特殊的要求,使用 resultMap 可以完成将关联查询映射 pojo 的属性中。resultMap 可以实现延迟加载。
- 比较复杂的映射建议使用 resultMap, 多用于对象的属性也是对象(或者连表查询),
  - 13. 注意事项:
    - A. 接口中的方法名一定要与 XML 中的实现方法的 ID 相一致,否则提示找不到;
    - B. XML 的命名空间要使用接口的全限定名;
    - C. 同一个 DAO 方法的实现,只能是 XML 或注解之一,共存则报错;
  - 14. Mybatis 和 Hibernate 的比较:

http://blog.csdn.net/jiuqiyuliang/article/details/45378065

## Mytbatis 核心配置文件参考:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"</pre>
"http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
properties resource="db. properties" />
<!-- 别名 -->
<typeAliases>
<typeAlias type="com.ddb.mybatis.domain.UserInfo" alias="UserInfo" />
</typeAliases>
<environments default="development">
<environment id="development">
<transactionManager type="JDBC" />
<!-- 配置数据库连接信息 -->
<!--
<dataSource type="POOLED">
cproperty name="driver" value="com.mysql.jdbc.Driver" />
cproperty name="url" value="jdbc:mysql://localhost:3306/train" />
property name="username" value="root" />
property name="password" value="root" />
</dataSource>
-->
<dataSource type="POOLED">
```

```
cproperty name="driver" value="${driver}" />
             property name="url" value="${url}" />
             cproperty name="username" value="${username}" />
             cproperty name="password" value="${password}" />
             </dataSource>
             </environment>
             </environments>
             <mappers>
             <!-- 注册模块的 xml 文件-->
             <mapper resource="com/ddb/mybatis/mapper/UserInfoMapper.xml"/>
             </mappers>
             </configuration>
Mybatis 当中传参的方式:
                                 一、使用 map 集合进行传参
        LIST<COMP.ILITATION TO THE TENTON OF THE PROPERTY OF THE PROPE
      List<com.iflytek.entity.User> getOneUserIf(Map<String, Object> parma);
第一步需要在接口当中的函数参数列表当中创建一个 map 的集合参数
     - - - NA - - > - 4 IM MA - - 4 D 4 A H 4 H 4 IV - - 10 MINING N F C
     7°<mapper namespace="com.iflytek.interfacepackage.User">
    8° <select id="getOneUserIf" resultType="User">
    9
                      select * from t_user
 10∘
                 <where>
11∘
                                     <if test="name !=null">
12
                                       name=#{name}
13
                                     </if>
                  </where>
 14
 15
                     </select>
 16 </mapper>
```

第二步 在后面的 XML 文件或者其他文件当中调用里面参数的 Key 值进行判断,获取 value 值使用#{属性名}进行获取

二、使用对象传参的方式

格式与 Map 集合方式类似,只不过在使用对象传参的形式需要对象类的属性创建 getter setter 方法