一，mybatis中的模糊查询以及utils代码的抽取：

"$"无法取出一个参数。

1，进行模糊查询的时候，like关键字的后面参数无论什么数据类型都保存在一对单引号内部。

模糊查询的Junit测试代码：

```
@Test
	public void getUserByMH() throws IOException{
		SqlSessionFactoryBuilder sessionFactoryBuilder = new
SqlSessionFactoryBuilder();
		InputStream ins = Resources.getResourceAsStream("mybatis.xml");
		SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
		SqlSession session = sessionFactory.openSession();
		Map<String, Object> map =
session.selectOne("cn.dao.daoImpl.getUserByMH");
		System.out.println(map);
	}
```

模糊查询的局部配置语句：

```
<select id="getUserByMH" resultType="map" parameterType="map">
		select *from users where username like '${username}%';
	</select>
```

2，模糊查询去参数需要使用“$”符号。若使用#需要使用如下的语句：mysql有一个内置函数concat用于拼接字符串。

```
<select id="getUserByMH1" resultType="map" parameterType="map">
		select *from users where username like concat(#{username},'%');
	</select>
```

总结：

①敲入一个关键字，按住Ctrl+0查看对应的方法或者相关的东西。

②concat在mysql中的作用是将多个字符串拼接成为一个字符串。

③在进行获取数据的时候，能尽量使用#就尽量使用#，因为$不安全，而#比较安全。

④只要需要传递参数，就需要在局部配置文件中的parameterType=“值”将值改为对应的数据类型。

3，抽取mybatis操作数据库的公共Utils类：（要求mybatis.xml和局部配置文件都是可以正常运行的，而不会抱任何配置问题有的错误）

```
package cn.dao.mybatisUtils;
import java.io.IOException;
import java.io.InputStream;
```

```
import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactory;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;
public class mybatisUtil {
public static SqlSession sessions() throws IOException{
	SqlSessionFactoryBuilder sessionFactoryBuilder = new
SqlSessionFactoryBuilder();
	InputStream ins = Resources.getResourceAsStream("mybatis.xml");
	SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
	SqlSession session = sessionFactory.openSession();
	return session;
}
}
```

Junit中的测试代码：

```
@Test
	public void getUserByMH1(){
		SqlSession session =null;
		try {
			session = mybatisUtil.sessions();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		// 传递一个参数
		Map<String, Object> mapsMap = new HashMap<String, Object>();
		mapsMap.put("username", "张");
		Map<String, Object> map =
session.selectOne("cn.dao.daoImpl.getUserByMH", mapsMap);
		System.out.println(map);
	}
```

或者将公共的代码抽取出来，放在一个init方法里面，并用before注解给该init方法进行注解，这样在后面的代码中如果使用到session对象就可以直接使用了：

```
private SqlSession session = null;
	@Before
	public void init() throws IOException {
		SqlSessionFactoryBuilder sessionFactoryBuilder = new
SqlSessionFactoryBuilder();
		InputStream ins = Resources.getResourceAsStream("mybatis.xml");
		SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
		session = sessionFactory.openSession();
	}
 @Test
	public void getUserByMH1() {
```

```
        // 传递一个参数
        Map<String, Object> mapsMap = new HashMap<String, Object>();
        mapsMap.put("username", "张");
        Map<String, Object> map =
session.selectOne("cn.dao.daoImpl.getUserByMH", mapsMap);
        System.out.println(map);
    }
```

或者将公共的代码存储在一个static静态代码块中。

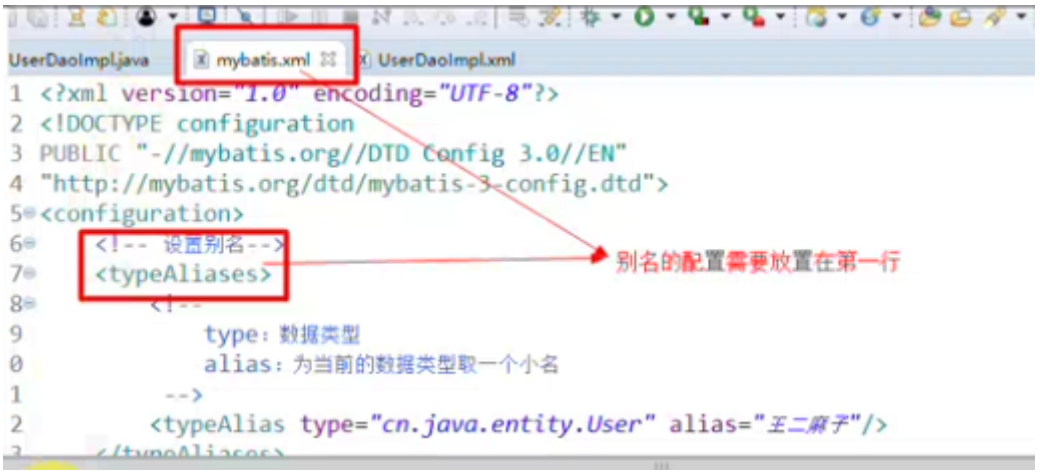
测试代码如下：

```
// 定义一个静态的sqlsession对象
    static SqlSession session = null;
    // 对其进行初始化
    static {
        SqlSessionFactoryBuilder sessionFactoryBuilder = new
SqlSessionFactoryBuilder();
        InputStream ins = null;
        try {
            ins = Resources.getResourceAsStream("mybatis.xml");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
        session = sessionFactory.openSession();
    }
    @Test
    public void getUserByMH1() {
        // 传递一个参数
        Map<String, Object> mapsMap = new HashMap<String, Object>();
        mapsMap.put("username", "张");
        Map<String, Object> map =
session.selectOne("cn.dao.daoImpl.getUserByMH", mapsMap);
        System.out.println(map);
    }
```

二，别名（三种方式，复杂度都差不多，建议直接使用第一种方式进行别名的选择）

（1），别名的配置信息需要写在主配置文件中去。

（2），设置别名要放在首行，放在其他地方会报错。



1，逐个定义

```
<typeAliases>
   <typeAlias type="com.gzw.mybatis.Role" alias="role"/>
</typeAliases>
```

2，在实体类上写注解（注意：不能直接单独使用注解，需要在主配置文件中写

<typeAliases>

    <typeAlias type="com.gzw.mybatis.Role" />

</typeAliases>，

然后才能在实体类的上边使用如下的注解方式进行别名的选取）

```
@Alias("role3")
public class Role {
   ......
}
```

3，扫描包，经过如下的配置后，可以在局部配置文件中直接写类的名字进行访问了

```
 <typeAliases>
            <package name="cn.entity"/>
      </typeAliases>
<!-- 多个条件查询 parameterType：指定参数类型 -->
      <select id="getShiTi" resultType="map" parameterType="user">
            <!-- #{存储的名字}，map中的数据是无序的，不能使用下标获取 -->
            select *from users where id=#{id} and username=#{username} and
password=#{password};
      </select>
```

测试：逐个定义的demo：

主配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-config.dtd">
```

```
<configuration>
	<typeAliases>
		<typeAlias alias="User" type="cn.entity.user" />
	</typeAliases>
	<environments default="mysql">
		<environment id="mysql">
			<transactionManager type="JDBC"></transactionManager>
			<dataSource type="POOLED">
				<property name="driver" value="com.mysql.cj.jdbc.Driver" />
				<property name="url"
					value="jdbc:mysql://localhost:3306/db_mybatis?
useSSL=false&serverTimezone=UTC&characterEncoding=utf8" />
				<property name="username" value="root" />
				<property name="password" value="AQL271422" />
			</dataSource>
		</environment>
	</environments>
	<mappers>
		<mapper resource="cn/dao/daoImpl.xml" />
	</mappers>
</configuration>
```

局部配置文件中的内容

```
<!-- 多个条件查询 parameterType：指定参数类型 -->
	<select id="getShiTi" resultType="map" parameterType="User">
		<!-- #{存储的名字}，map中的数据是无序的，不能使用下标获取 -->
		select *from users where id=#{id} and username=#{username} and
password=#{password};
	</select>
```