一、上节课总结：

1，查询单条记录使用selectlist（），查询一条记录使用selectone（）；前者返回的是多个查询记录的集合，后者返回的是一个记录；前者需要使用一个list进行装载多个map，后者只需要一个map集合就能将其装载起来。

2，当需要带一个条件进行查询的时候：

```
@Test
	/*
	 * 根据传递的参数查询数据信息
	 */
	public void getUsersByTiaojian() throws IOException {
		SqlSessionFactoryBuilder ssf = new SqlSessionFactoryBuilder();
		InputStream ins = Resources.getResourceAsStream("mybatis.xml");
		SqlSessionFactory sqlSessionFactory = ssf.build(ins);
		SqlSession session = sqlSessionFactory.openSession();
//查询的条件---->根据users表中id的值进行查询
		Long id = 1L; ( 其中selectone或者selectlist中的第二个参数需要引用一个对象，而不能使用一个基本数据类型的数据，否则会报错 )
		System.out.println("带条件的查询：");
		List<Map<String, Object>> list = session.selectList("cn.java.dao.impl.userDaoImpl.getUsersByTiaojian", id);
		for (Map<String, Object> map : list) {
			System.out.println(map);
		}
	}
```

在局部配置文件中：

```
<!--
	parameterType：指定参数类型
	 -->
	<select id="getUsersByTiaojian" resultType="map" parameterType="long">
	<!-- #{角标}其中角标从0开始，0表示第一个参数，没有参数的话，parameter参数不需要写-->
	select *from users where id =#{0};
	</select>
```

二、本节课课程笔记：

1，多个条件进行查询

单元测试方法：

```
@Test
	public void getUsersByTiaojian() throws IOException{
		SqlSessionFactoryBuilder sessionFactoryBuilder = new SqlSessionFactoryBuilder();
```

```
            InputStream ins = Resources.getResourceAsStream("mybatis.xml");
            SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
            SqlSession session = sessionFactory.openSession();
            //同时传递多个参数,使用map
            //map中的数据是无序的，在map中取数据需要使用key进行取出
            Map<String, Object> hashmap = new HashMap<String, Object>();
            hashmap.put("id",1l);
            hashmap.put("username", "王二麻子");
            hashmap.put("password","123");
            Map<String, Object> map =
session.selectOne("cn.dao.daoImpl.getUsersByTiaojian",hashmap);
            System.out.println(map);
      }
```

配置文件中的内容：

```
<!-- 多个条件查询 parameterType：指定参数类型 -->
      <select id="getUsersByTiaojian" resultType="map" parameterType="map">
            <!-- #{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>
```

多个参数的时候，可以存放在集合map，如果是单个参数的话，可以直接通过#{0}来进行获取，如果是多个参数的话，需要使用map、实体类等进行传参，获取的时候只能通过key值进行获取。

2，使用实体类进行传参的demo实例：

user实体类：

```
package cn.java.entity;

import java.io.Serializable;

public class user implements Serializable{
private Long id;
private String username;
private String password;
public user() {
      // TODO Auto-generated constructor stub
}
@Override
public String toString() {
      return "user [id=" + id + ", username=" + username + ", password=" +
password + "]";
}
public Long getId() {
```

```
        return id;
}
public void setId(Long id) {
        this.id = id;
}
public String getUsername() {
        return username;
}
public void setUsername(String username) {
        this.username = username;
}
public String getPassword() {
        return password;
}
public void setPassword(String password) {
        this.password = password;
}
}
```

test代码：

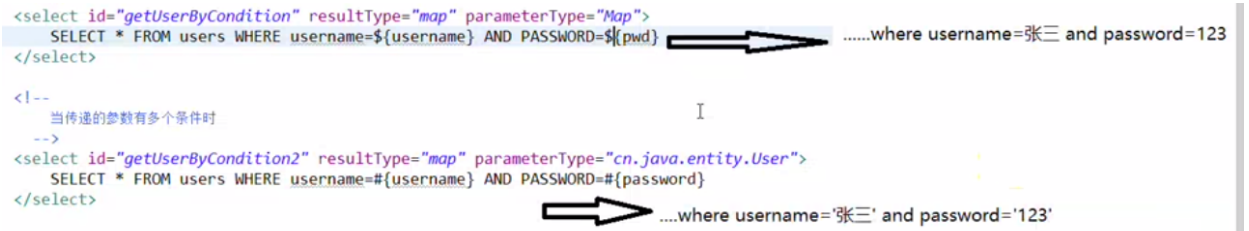
```
@Test
        public void getShiTi() throws IOException{
                SqlSessionFactoryBuilder sessionFactoryBuilder = new
SqlSessionFactoryBuilder();
                InputStream ins = Resources.getResourceAsStream("mybatis.xml");
                SqlSessionFactory sessionFactory = sessionFactoryBuilder.build(ins);
                SqlSession session = sessionFactory.openSession();
                //同时传递多个参数,使用实体类对象
                user user = new user();
                user.setId(1L);
                user.setUsername("王二麻子");
                user.setPassword("123");
                Map<String, Object> map =
session.selectOne("cn.dao.daoImpl.getShiTi",user);
                System.out.println(map.get("id"));
                System.out.println(map.get("username"));
                System.out.println(map.get("password"));
        }
```

局部配置文件的配置信息：

```
<!-- 多个条件查询 parameterType：指定参数类型 -->
        <select id="getShiTi" resultType="map" parameterType="cn.entity.user">
                <!-- #{对象中存储的属性名字}，-->
                select *from users where id=#{id} and username=#{username} and
password=#{password};
        </select>
```

3，mybatis中$和#获取数据的区别

（1）注意在$（）中的值不是el表达式，el表达式只有在jsp中才有。



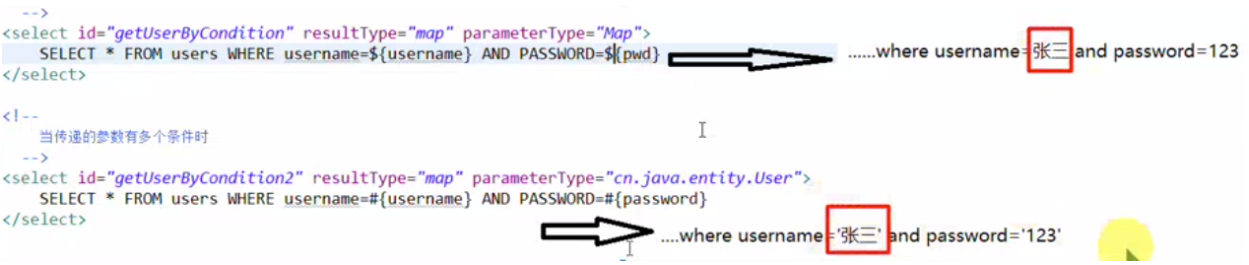
①"$"无论传递的是什么数据，结果就是什么的数据值，并不会将其进行数据的转换；但是"#"会把任何参数都按照一个字符串进行处理，默认情况下会给每个参数加上一个单引号。

```
<!-- 多个条件查询 parameterType：指定参数类型 -->
	<select id="getShiTi" resultType="map" parameterType="cn.entity.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>

	<select id="tests11" resultType="map" parameterType="map">
	//在数据库中是没有双引号的，如果一个参数它在数据库中的数据类型是字符串，则需要加上单引号，如果是数值类型的参数，不需要进行任何处理
		select *from users where id='${id}' and username='${username}' and
password='${password}';
	</select>
```

区别：

①见下图：



②“#”获取参数时，可以防止SQL注入；”$“不能防止SQL的注入。---->安全性

结论：因此能够使用“#”获取参数的时候，尽量使用“#”进行参数的获取。

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。