一，通过逆向工程生成实体类驼峰命名的方法

数据库的命名千万不要使用驼峰命名，而是多个单词之间使用下划线隔开。

注意：在数据库中建表或者其他的命名规范的时候，多个单词之间通过"_"分隔，并且每个单词都采用小写的字母。---------->解决逆向工程中生成实体类驼峰命名的问题。

二，通过mybatis添加数据记录时，如何返回主键（***）

（一，实体对象接收的方式）

1，局部配置文件中的配置信息：

```
<!--
	useGeneratedKeys：表示使用主键作为我们数据库操作的返回值 ；
	parameterType：表示传递的参数的实体类所在的类的路径
	keyProperty：表示将主键封装到某一个实体类中对应的属性中
	 -->
	<insert id="insertData" useGeneratedKeys="true" keyProperty="id"
		parameterType="cn.java.entity.user">
		insert into users(username,password) values('anqili','2222');
	</insert>
```

2，java中Junit测试代码：

```
@Test
	public void insertData() {
		// id永远都表示的是影响的行数
		/*
		 * Integer id = session.insert("cn.java.dao.one2manyDao.insertData");
		 * session.commit(); System.out.println(id);
		 */
		// 返回主键的方法
		// user并不是作为参数传递过去，而是作为接收参数的载体
		user user = new user();
		//如果user如下写，那么user既表示返回之间的载体，又作为SQL操作时传递参数
		user.setUsername("111");
		user.setPassword("121");
		Integer id = session.insert("cn.java.dao.one2manyDao.insertData", user);
		session.commit();
		System.out.println(user.getId());
		System.out.println(id);
	}
```

（二）通过map的方式

1，局部配置文件中的配置信息：

```
<insert id="testMap" useGeneratedKeys="true" keyProperty="id"
parameterType="map">
        insert into users(username,password)
        values('anqili','2222');
    </insert>
```

2，java中Junit的测试demo：

```
@Test
    public void testMap() {
        Map<String, Object> map = new HashMap<String, Object>();
        Integer id = session.insert("cn.java.dao.one2manyDao.testMap", map);
        session.commit();
        System.out.println(id);
        System.out.println("主键为" + map.get("id"));
    }
```

三，p6spy工具的使用

①导入p6spy的jar包；

②导入p6spy的配置文件；

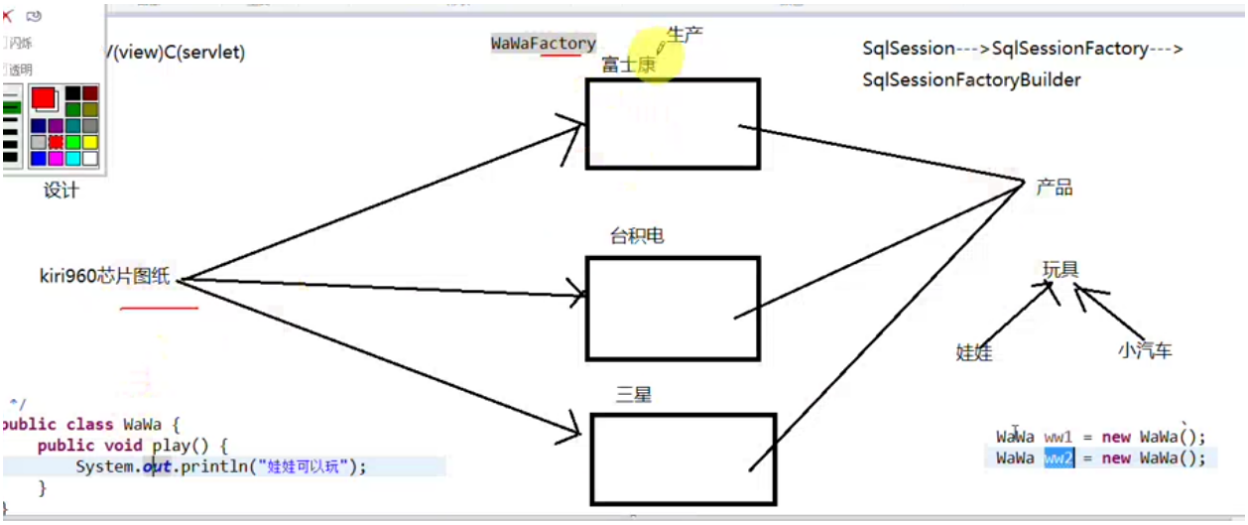
③改写mybatis.xml配置文件。

四，工厂模式与单例模式

1，工厂模式的优点：①分工明确；②易于拓展

工厂模式：①降低模块之间的耦合。②利于拓展。

多个对象都是同一个对象的话，表示这个是一个单例模式。



饿汉式：不管有没有对象，都要new一个对象；

懒汉式：先判断这个对象存不存在，如果不存在，在new对象，如果存在就不需要进行new对象。单例模式中懒汉式可能不安全，他有可能会有多个对象。

,2，单例模式：（必须会手写）饿汉式和懒汉式

```
// 饿汉式
public static Car getCarInstance() {
    return car;
}

// 懒汉式(非线程安全的)
public static Car getCarInstance2() {
    if (car2 == null) {
        car2 = new Car();
    }
    return car2;
}
}
```

3，单例模式和静态工厂模式之间的联系。