第五篇 Active Record模式

> **Active Record 适合非常简单的领域需求，尤其在领域模型和数据库模型十分相似的情况下。如果遇到更加复杂的领域模型结构（例如用到继承、策略的领域模型），往往需要使用分离数据源的领域模型，结合 Data Mapper （数据映射器）使用。**

具体到使用层面，我们之前讲过使用Data Mapper 做数据的持久层操作。

```
User user = new User();
user.setName("Tom");
user.setAge(18);

userMapper.insert(user);    //Mybatis Mapper模式
```

现在我们使用ActiveRecord模式，用法如下，注意二者的区别

```
User user = new User();
user.setName("Tom");
user.setAge(18);

user.insert();  //ActiveRecord模式
```

一、使ActiveRecord模式生效

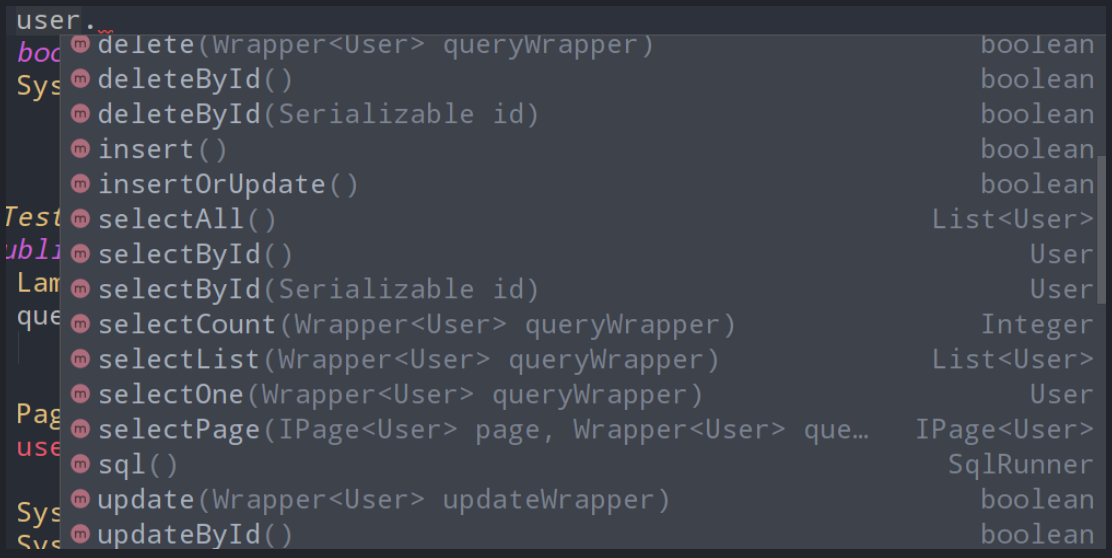
首先：需要让数据库表对应的数据持久层实体类。继承自Model，并实现序列化接口。

```java
@Data
@EqualsAndHashCode(callSuper = true)
public class User extends Model<User> implements
Serializable {
    private static final long serialVersionUID =
6401942840459021558L;
    private Long id;
    private String name;
    private Integer age;
    private String email;
}
```

然后定义一个Mapper继承自BaseMapper，T代表数据持久层实体类。

```java
public interface UserMapper extends BaseMapper<User> {
}
```

这样Mybatis Plus的ActiveRecord模式就生效了，默认的帮我们实现了如下的一些数据持久层方法。



二、增删改查的实现

增加：向持久层实体类User对应的数据库表user，插入一条数据。

```
@Test
public void testInsert() {
  User user = new User();
  user.setName("springboot");
  user.setAge(18);
  user.setEmail("springboot@163.com");
  boolean success = user.insert();
  assertEquals(true, success);
}
```

查询：从数据库表user查询所有数据

```
@Test
public void testSelect() {
  User user = new User();
  List<User> users = user.selectAll();
  users.forEach(System.out::println);
  assertEquals(6,users.size());
}
```

添加或更新：

- 未设置ID，被视为insert操作，向数据库插入数据
- 如果设置ID，则先查询是否有此id的记录，如果有此id记录，则视为update,如果没有则视为insert

```
@Test
public void testUpdate() {
  User user = User.builder()
                .id(1283915378849751041L)
                .name("mbp")
                .age(20)
                .build();
  boolean success = user.insertOrUpdate();
  assertEquals(true, success);
}
```

根据id在数据库表user中删除一条记录

```java
@Test
public void testDelete() {
  User user = new User();
  user.setId(1283915378849751041L);
  boolean success = user.deleteById();
  assertEquals(true, success);
}
```