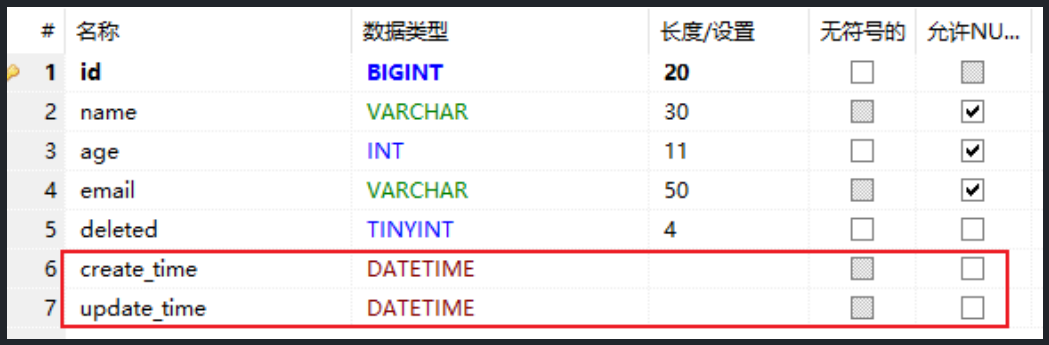
第九篇 常用字段默认值默认填充

一、填充字段处理

需求案例：在插入数据的时候自动填充createTime和updateTime为当前插入数据的时间，在数据更新的时候修改updateTime为修改数据的时间。**不需要人为手动赋值**。

- 在user数据表先添加2个日期类型的字段create_time和update_time：



#	名称	数据类型	长度/设置	无符号的	允许NU...
1	id	BIGINT	20	☐	☐
2	name	VARCHAR	30	☐	☑
3	age	INT	11	☐	☑
4	email	VARCHAR	50	☐	☑
5	deleted	TINYINT	4	☐	☐
6	create_time	DATETIME		☐	☐
7	update_time	DATETIME		☐	☐

- 使用**@TableField**注解标记实体类中对应的这两个字段需要填充：

```
@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class User {
    private Long id;
    private String name;
    private Integer age;
    private String email;

    @TableField(fill = FieldFill.INSERT)
    private Date createTime;
    @TableField(fill = FieldFill.INSERT_UPDATE)
    private Date updateTime;
}
```

FieldFill是一个枚举，用于指定在何种情况下会自动填充，有如下几种可选值：

- DEFAULT：默认不处理
- INSERT：插入时自动填充字段
- UPDATE：更新时自动填充字段
- INSERT_UPDATE：插入和更新时自动填充字段

二、自定义填充默认数值

编写公共字段填充处理器类，该类继承了MetaObjectHandler类，重写insertFill和updateFill方法，我们在这两个方法中获取需要填充的字段以及默认填充的值。

> - **填充处理器MyMetaObjectHandler在Spring Boot中需要声明@Component或@Bean注入**
> - **strictInsertFill和strictUpdateFill方法第二个参数写的是实体类里的属性名，不是对应数据库字段名。**

```
@Component
public class MyMetaObjectHandler implements
MetaObjectHandler {

    @Override
    public void insertFill(MetaObject metaObject) {
        this.strictInsertFill(metaObject, "createTime",
Date.class, new Date());
        this.strictInsertFill(metaObject, "updateTime",
Date.class, new Date());
    }

    @Override
    public void updateFill(MetaObject metaObject) {
        this.strictUpdateFill(metaObject, "updateTime",
Date.class, new Date());
    }
}
```

如果是3.3.0后面的版本，比如3.3.1.8，也可以改用下面更简单的写法（3.3.0不要用该方法，有bug）

```
@Component
public class MyMetaObjectHandler implements
MetaObjectHandler {

    @Override
    public void insertFill(MetaObject metaObject) {
        this.fillStrategy(metaObject, "createTime", new
Date());
        this.fillStrategy(metaObject, "updateTime", new
Date());
    }

    @Override
    public void updateFill(MetaObject metaObject) {
        this.fillStrategy(metaObject, "updateTime", new
Date());
    }
}
```

在一些比较旧的版本，为填充字段设置值的API如下，3.3.0之后已经不建议使用

```
this.setFieldValByName("createTime",new
Date(),metaObject);
this.setFieldValByName("updateTime",new
Date(),metaObject);
```

三、开始测试

- 插入一条数据，注意我们没有为createTime和updateTime赋值

```
@Test
public void testInsert() {
    User user = User.builder().name("软件学
子").age(19).email("soft@niit.edu.cn").build();
    int row = userMapper.insert(user);
    assertEquals(1, row);
}
```

运行的结果是：createTime和updateTime被自动赋值

id	name	age	email	create_time	update_time
1	test	20	test@163.com	2021-04-05 19:30:39	2021-04-05 19:30:41
2	软件学子	19	soft@niit.edu.cn	2021-04-05 20:27:53	2021-04-05 20:27:53

根据id更新一条数据，注意我们没有为updateTime赋值

```
@Test
public void testUpdate() {
  User user = new User();
  user.setId(2L);
  user.setName("新名字");
  user.setAge(18);
  user.setEmail("newName@niit.edu.cn");
  int row = userMapper.updateById(user);
  assertEquals(1, row);
}
```

运行的结果是：updateTime在执行数据记录修改操作时被自动赋值

id	name	age	email	create_time	update_time
1	test	20	test@163.com	2021-04-05 19:30:39	2021-04-05 19:30:41
2	新名字	18	newName@niit.edu.cn	2021-04-05 20:27:53	2021-04-05 20:34:35