Mybatis初步复习

1.SpringBoot整合Mybatis

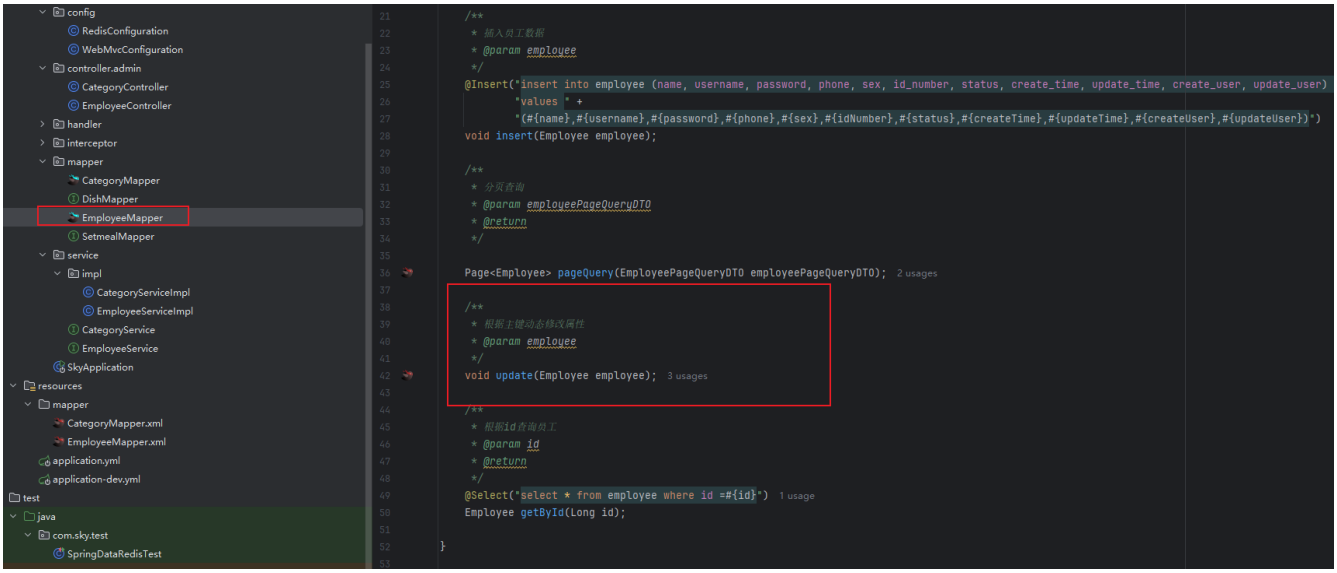
1.1第一步添加依赖

```
<!--        mybatis启动器-->

        <dependency>
            <groupId>org.mybatis.spring.boot</groupId>
            <artifactId>mybatis-spring-boot-starter</artifactId>
            <version>3.0.5</version>
        </dependency>
```

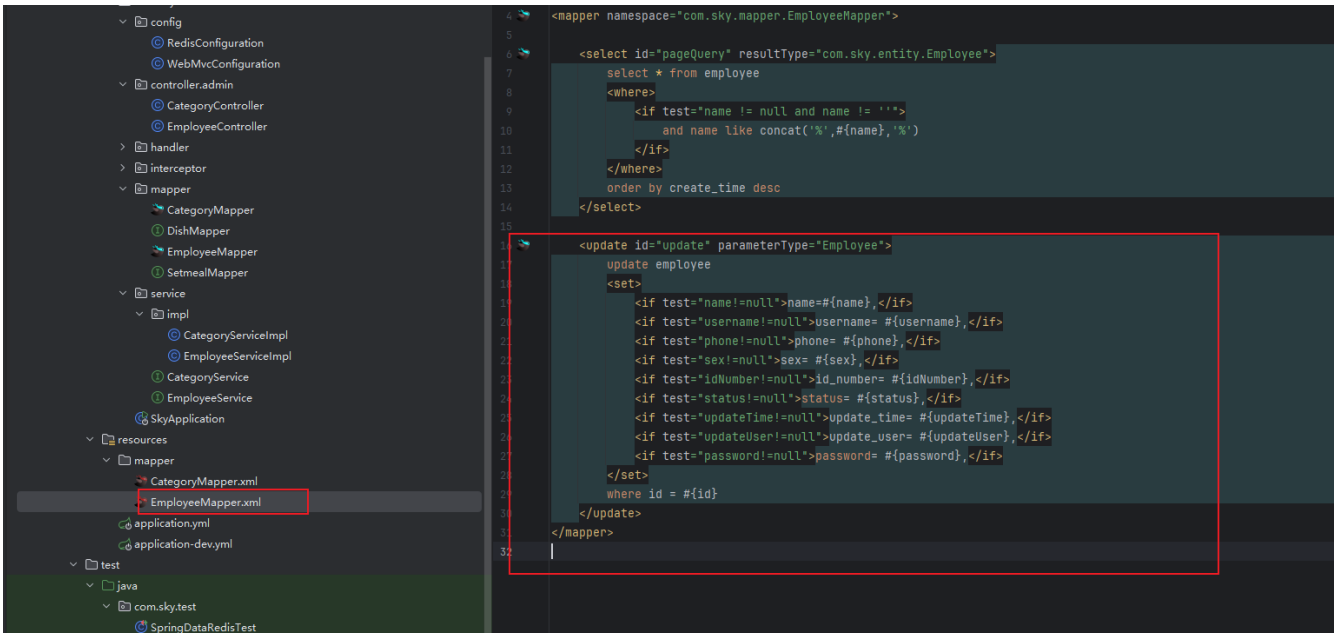
1.2进行配置

```
#配置Mybatis
mybatis:
  configuration:
    #在映射为java对象，将表中的下划线命名自动转换成驼峰式命名
    map-underscore-to-camel-case: true
    #日志前缀    可选
    log-prefix: mybatis.
    #日志实现类   可选
    log-impl: org.apache.ibatis.logging.commons.JakartaCommonsLoggingImpl
  #动态sql文件存储位置
  mapper-locations: classpath:/mapper/**/*.xml

#配置日志显示sql
logging:
  level:
    #指定日志前缀
    mybatis: debug
```

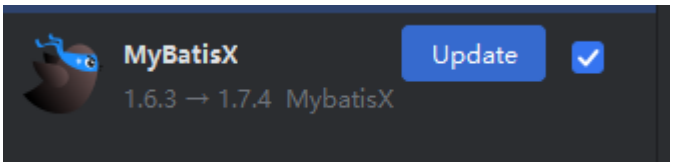
1.3 在Dao层编写的mapper接口，添加@Mapper注解



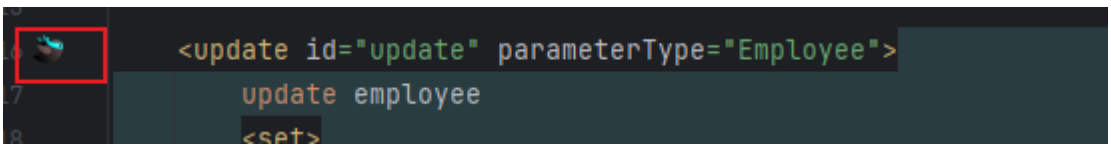
1.4 编写动态sql文件（xx.xml文件）



1.5 安装mybatisx插件



安装之后在mapper和动态SQL文件对应代码左边有一个小鸟图标，方便直接跳转



2.自动映射

当开启驼峰命名自动映射

```yaml
mybatis:
  configuration:
    map-underscore-to-camel-case: true
```

3.手动映射

当自动映射无法满足需求时，可以使用 `<resultMap>` 进行精确的手动映射。

```xml
<resultMap id="唯一标识" type="要映射的Java类型">
    <!-- 主键映射 -->
    <id property="Java属性名" column="数据库列名"/>

    <!-- 普通字段映射 -->
    <result property="Java属性名" column="数据库列名"/>

    <!-- 关联关系映射 -->
    <association property="关联对象属性" javaType="关联对象类型"/>
    <collection property="集合属性" ofType="集合元素类型"/>
</resultMap>
```

举例：

3.1基础字段映射

实体类

```java
public class User {
    private Long userId;            // 与数据库字段名不同
    private String userName;        // 需要手动映射
    private String email;
    private Integer status;
    private LocalDateTime createTime;
    private LocalDateTime updateTime;

    // 构造函数、getter、setter...
}
```

数据库表结构：

```sql
CREATE TABLE t_user (
    id BIGINT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100),
    status TINYINT,
    created_at DATETIME,
    updated_at DATETIME
);
```

```xml
<!-- 基础手动映射示例 -->
<resultMap id="BaseUserMap" type="User">
```

```
    <!-- 主键字段，使用 id 标签 -->
    <id property="userId" column="id"/>

    <!-- 普通字段，使用 result 标签 -->
    <result property="userName" column="name"/>
    <result property="email" column="email"/>
    <result property="status" column="status"/>
    <result property="createTime" column="created_at"/>
    <result property="updateTime" column="updated_at"/>
</resultMap>

<!-- 使用 resultMap -->
<select id="selectUserById" resultMap="BaseUserMap">
    SELECT id, name, email, status, created_at, updated_at
    FROM t_user
    WHERE id = #{id}
</select>
```

说明：

3.2一对一关联映射

3.2.1单次查询嵌套结果映射

举例：

实体类

```
public class User {
    private Long userId;
    private String userName;
    private UserProfile userProfile; // 一对一关联
    // getter/setter...
```

```
}

public class UserProfile {
    private Long profileId;
    private Long userId;
    private String realName;
    private Integer age;
    private String address;
    private String phone;
    // getter/setter...
}
```

手动映射配置：

```
<!-- 用户和用户详情的一对一映射 -->
<resultMap id="UserWithProfileMap" type="User">
    <id property="userId" column="id"/>
    <result property="userName" column="name"/>

    <!-- association: 一对一关联映射 -->
    <association property="userProfile" javaType="UserProfile">
        <id property="profileId" column="profile_id"/>
        <result property="userId" column="id"/> <!-- 注意：这里复用外层查询的id -->
        <result property="realName" column="real_name"/>
        <result property="age" column="age"/>
        <result property="address" column="address"/>
        <result property="phone" column="phone"/>
    </association>
</resultMap>

<!-- 关联查询SQL -->
<select id="selectUserWithProfile" resultMap="UserWithProfileMap">
    SELECT
        u.id, u.name,
        up.id as profile_id, up.real_name, up.age, up.address, up.phone
    FROM t_user u
    LEFT JOIN t_user_profile up ON u.id = up.user_id
    WHERE u.id = #{userId}
</select>
```

3.2.2 多次查询嵌套查询映射

```
<!-- 主结果映射 -->
<resultMap id="UserWithProfileNestedMap" type="User">
    <id property="userId" column="id"/>
    <result property="userName" column="name"/>

    <!-- 嵌套查询：通过 select 属性引用另一个查询 -->
    <association property="userProfile" column="id"
                 select="selectUserProfileByUserId"/>
</resultMap>

<!-- 主查询 -->
```

```xml
<select id="selectUserWithProfileNested" resultMap="UserWithProfileNestedMap">
    SELECT id, name
    FROM t_user
    WHERE id = #{userId}
</select>

<!-- 嵌套查询 -->
<select id="selectUserProfileByUserId" resultType="UserProfile">
    SELECT
        id as profileId,
        user_id as userId,
        real_name as realName,
        age, address, phone
    FROM t_user_profile`
    WHERE user_id = #{userId}
</select>
```

3.3一对多集合映射

实体类:

```java
public class User {
    private Long userId;
    private String userName;
    private List<Order> orders; // 一对多关联
    // getter/setter...
}

public class Order {
    private Long orderId;
    private Long userId;
    private String orderNumber;
    private BigDecimal amount;
    private LocalDateTime orderTime;
    // getter/setter...
}
```

```xml
<!-- 用户和订单的一对多映射 -->
<resultMap id="UserWithOrdersMap" type="User">
    <id property="userId" column="id"/>
    <result property="userName" column="name"/>

    <!-- collection: 一对多关联映射 -->
    <collection property="orders" ofType="Order">
        <id property="orderId" column="order_id"/>
        <result property="userId" column="id"/> <!-- 注意: 这里复用外层查询的id -->
        <result property="orderNumber" column="order_number"/>
        <result property="amount" column="amount"/>
        <result property="orderTime" column="order_time"/>
    </collection>
</resultMap>
```

```xml
<!-- 关联查询SQL -->
<select id="selectUserWithOrders" resultMap="UserWithOrdersMap">
    SELECT
        u.id, u.name,
        o.id as order_id, o.order_number, o.amount, o.order_time
    FROM t_user u
    LEFT JOIN t_order o ON u.id = o.user_id
    WHERE u.id = #{userId}
</select>
```

3.4复杂的多层嵌套

```java
public class User {
    private Long userId;
    private String userName;
    private List<Order> orders;
}

public class Order {
    private Long orderId;
    private String orderNumber;
    private List<OrderItem> orderItems;
}

public class OrderItem {
    private Long itemId;
    private Long productId;
    private Integer quantity;
    private Product product; // 关联商品
}

public class Product {
    private Long productId;
    private String productName;
    private BigDecimal price;
}
```

```xml
<!-- 完整的多层嵌套映射 -->
<resultMap id="CompleteUserMap" type="User">
    <id property="userId" column="user_id"/>
    <result property="userName" column="user_name"/>

    <!-- 第一层嵌套：订单集合 -->
    <collection property="orders" ofType="Order" resultMap="OrderWithItemsMap"/>
</resultMap>

<resultMap id="OrderWithItemsMap" type="Order">
    <id property="orderId" column="order_id"/>
    <result property="orderNumber" column="order_number"/>

    <!-- 第二层嵌套：订单项集合 -->
```

```
    <collection property="orderItems" ofType="OrderItem" resultMap="OrderItemMap"/>
</resultMap>

<resultMap id="OrderItemMap" type="OrderItem">
    <id property="itemId" column="item_id"/>
    <result property="quantity" column="quantity"/>

    <!-- 第三层嵌套：商品信息 -->
    <association property="product" javaType="Product">
        <id property="productId" column="product_id"/>
        <result property="productName" column="product_name"/>
        <result property="price" column="price"/>
    </association>
</resultMap>

<!-- 复杂查询SQL -->
<select id="selectUserWithOrderDetails" resultMap="CompleteUserMap">
    SELECT
        u.id as user_id, u.name as user_name,
        o.id as order_id, o.order_number,
        oi.id as item_id, oi.quantity,
        p.id as product_id, p.name as product_name, p.price
    FROM t_user u
    LEFT JOIN t_order o ON u.id = o.user_id
    LEFT JOIN t_order_item oi ON o.id = oi.order_id
    LEFT JOIN t_product p ON oi.product_id = p.id
    WHERE u.id = #{userId}
    ORDER BY o.order_time DESC, oi.id
</select>
```

3.5.注意

1.可以 自动映射与手动映射结合

```
<!-- 混合映射：autoMapping="true" 开启自动映射 -->
<resultMap id="HybridUserMap" type="User" autoMapping="true">
    <!-- 只手动映射名称不匹配的字段 -->
    <id property="userId" column="id"/>
    <result property="userName" column="name"/>
    <result property="createTime" column="created_at"/>

    <!-- 关联对象也可以使用自动映射 -->
    <association property="userProfile" javaType="UserProfile" autoMapping="true">
        <id property="profileId" column="profile_id"/>
        <!-- 只覆盖需要特殊处理的字段 -->
        <result property="realName" column="real_name"/>
    </association>
</resultMap>
```

##