链滴

mybatis 自动映射对象为 json

作者：TWanGT

原文链接：https://ld246.com/article/1598193631657

来源网站：链滴

一般我们关系型数据库存储的字段都是从不同单一维度描述这个对象, 而随着业务的复杂和数据维度增加, 我们有时候需要直接将一些简单维度(只作展示, 不涉及条件查询以及基本不修改)的集合封装成json格式放入一个大字段中(避免联合查询的额外扫表的开销)

typeHandler的方式做对象映射

本文只介绍简单的使用方式, 具体原理和详细解析请跳转以下链接

参考资料

https://www.codenong.com/js92a4cfdcc700/

https://juejin.im/post/6844903997271179277

1. 自定义定义映射器

对象映射器

```
public abstract class BaseMybatisObject2JsonHandler<T> extends BaseTypeHandler<T> {


    @Override
    public void setNonNullParameter(PreparedStatement ps, int i, Object parameter,
                                    JdbcType jdbcType) throws SQLException {
        ps.setString(i, JSON.toJSONString(parameter));
    }

    @Override
    public T getNullableResult(ResultSet rs, String columnName)
            throws SQLException {
        String data = rs.getString(columnName);
        return StringUtils.isBlank(data) ? null : JSON.parseObject(data, (Class<T>) getRawType());
    }

    @Override
    public T getNullableResult(ResultSet rs, int columnIndex) throws SQLException {
        String data = rs.getString(columnIndex);
        return StringUtils.isBlank(data) ? null : JSON.parseObject(data, (Class<T>) getRawType());
    }

    @Override
    public T getNullableResult(CallableStatement cs, int columnIndex)
            throws SQLException {
        String data = cs.getString(columnIndex);
        return StringUtils.isBlank(data) ? null : JSON.parseObject(data, (Class<T>) getRawType());
    }
}
```

列表映射器

```
public abstract class BaseMybatisList2JsonHandler<T> extends BaseTypeHandler<List<T>> {

    @Override
    public void setNonNullParameter(PreparedStatement ps, int i, List<T> parameter, JdbcType
jdbcType) throws SQLException {
        ps.setString(i, JSON.toJSONString(parameter));
    }

    @Override
    public List<T> getNullableResult(ResultSet rs, String columnName)
            throws SQLException {
        String data = rs.getString(columnName);
        return StringUtils.isBlank(data) ? null : JSON.parseArray(data, (Class<T>) getRawType());
    }

    @Override
    public List<T> getNullableResult(ResultSet rs, int columnIndex) throws SQLException {
        String data = rs.getString(columnIndex);
        return StringUtils.isBlank(data) ? null : JSON.parseArray(data, (Class<T>) getRawType());
    }

    @Override
    public List<T> getNullableResult(CallableStatement cs, int columnIndex)
            throws SQLException {
        String data = cs.getString(columnIndex);
        return StringUtils.isBlank(data) ? null : JSON.parseArray(data, (Class<T>) getRawType());
    }
}
```

需要进行映射的继承注册一下即可

```
@MappedTypes({Snippet.class})
@MappedJdbcTypes(value = JdbcType.VARCHAR, includeNullJdbcType = true)
public class InstructionBoTypeHandler extends BaseMybatisObject2JsonHandler<InstructionB
> {

}
```

- @MappedTypes: 对应的java对象
- @MappedJdbcTypes: 对应的数据库字段类型
- includeNullJdbcType: 是否包含一个null的Jdbc类型(这个类型在框架获取typeHandler时候用到,细看跳转文章)

2. 配置xxxmapper.xml文件

java对象内容

```
public class Snippet implements Serializable {
```

```
    private static final long serialVersionUID = 1L;

    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;
    private String name;
    private Integer homeId;
    @TableField(typeHandler = FastjsonTypeHandler.class)
    private InstructionBo commandSet;
    private Integer revision;
    private Integer state;
    private String createUser;
    @TableField(fill = FieldFill.INSERT)
    private LocalDateTime createTime;
    private String updateUser;
    @TableField(fill = FieldFill.INSERT_UPDATE)
    private LocalDateTime updateTime;
}
```

xml配置

```
<!-- 通用查询映射结果 -->
    <resultMap id="BaseResultMap" type="com.smart.life.userserver.domain.po.Snippet">
        <id column="id" property="id" />
        <result column="name" property="name" />
        <result column="home_id" property="homeId" />
        <result column="command_set" property="commandSet" javaType="com.smart.life.use
server.domain.po.Snippet" typeHandler="com.smart.life.userserver.helper.type.InstructionBoT
peHandler"/>
        <result column="revision" property="revision" />
        <result column="state" property="state" />
        <result column="create_user" property="createUser" />
        <result column="create_time" property="createTime" />
        <result column="update_user" property="updateUser" />
        <result column="update_time" property="updateTime" />
    </resultMap>
```

3. properties配置

mybatis-spring-boot-starter方式

```
## application.properties里配置
mybatis.typeHandlersPackage={BarTypeHandler所在包路径}
```

mybatis-spring 方式

构造一个SqlSessionFactoryBean对象，并调用其setTypeHandlersPackage方法设置类型处理器扫路径

mybatis-spring 方式

```
## application.properties里配置
```

mybatis-plus.type-aliases-package=com.smart.life.userserver.helper.type

错误

一直报下面这个错

org.mybatis.spring.MyBatisSystemException: nested exception is org.apache.ibatis.exceptions
PersistenceException:
Error updating database. Cause: java.lang.IllegalStateException: Type handler was null on
parameter mapping for property 'commandSet'. It was either not specified and/or could not
e found for the javaType (com.smart.life.common.domain.Bo.InstructionBo) : jdbcType (null) c
mbination.
The error may exist in com/smart/life/userserver/mapper/SnippetMapper.java (best gues
)
The error may involve com.smart.life.userserver.mapper.SnippetMapper.insert
The error occurred while executing an update
Cause: java.lang.IllegalStateException: Type handler was null on parameter mapping for p
operty 'commandSet'. It was either not specified and/or could not be found for the javaType (
om.smart.life.common.domain.Bo.InstructionBo) : jdbcType (null) combination.

mybatis-plus大法好

使用mybatis-plus的童鞋可以非常方便的使用转换器, 因为mybatis-plus都已经实现好了, 只需要指他用哪种json

在需要进行json序列化与反序列化的字段上加上如下注解

根据自己使用使用的json包选择对应的注解

```
@TableField(typeHandler = FastjsonTypeHandler.class)
@TableField(typeHandler = JacksonTypeHandler.class)
```

xml 中配置

需要json化的字段加上 typeHandler="com.baomidou.mybatisplus.extension.handlers.FastjsonT
peHandler"

```
<result column="command_set" property="commandSet" typeHandler="com.baomidou.my
atisplus.extension.handlers.FastjsonTypeHandler"/>
```