Mybatis 插件实现动态设置参数

一、背景

笔者在搭建架构时，通常会利用泛型对 dao 层 和 service 层公共的代码（增删改）进行抽取，但是遇到一个尴尬的问题，就是实体类中的时间设置。

解决办法有很多，简单的方法就是在 web 层接收实体类参数后直接设置时间即可。但是，web 层理论上只是调用 service 层代码而已，设置时间的操作应该放在 service 层来实现，且设置时间又是一个简单的、重复性的操作，因此在网上查阅了一些资料，个人感觉比较友好的方式就是使用 Mybatis 插件。

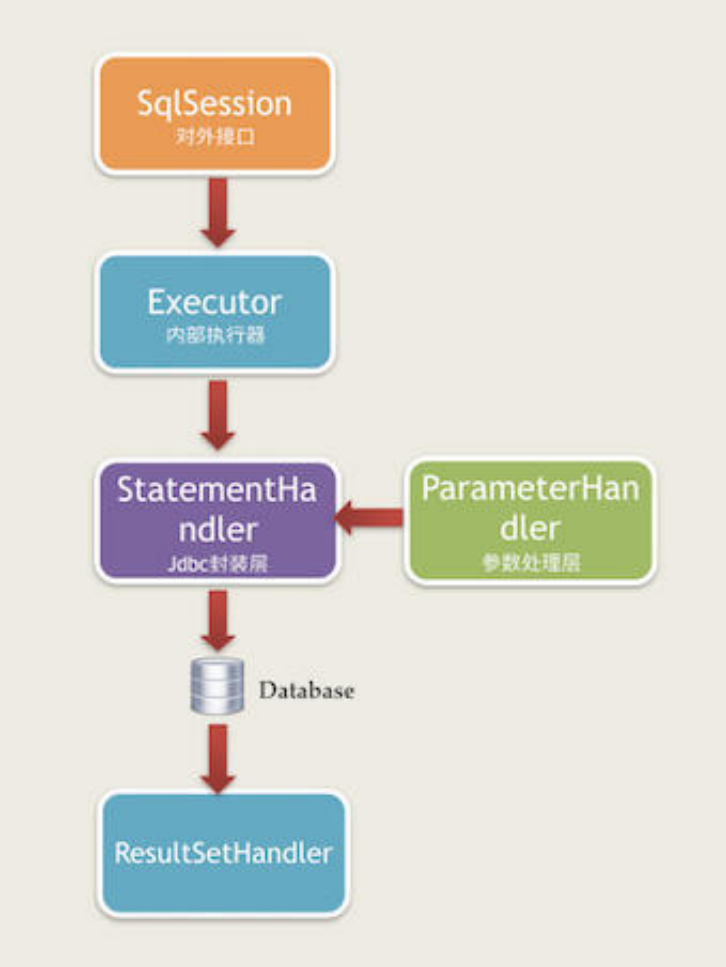
本文介绍使用 Mybatis 插件动态设置参数。

二、Mybatis 插件简单介绍

Mybatis 提供 Interceptor 接口，配合 @Intercepts 注解可以拦截如下 4 个对象的方法调用：

```
Executor (update, query, flushStatements, commit, rollback,
getTransaction, close, isClosed)
ParameterHandler (getParameterObject, setParameters)
ResultSetHandler (handleResultSets, handleOutputParameters)
StatementHandler (prepare, parameterize, batch, update, query)
```

其关系如下图：



解释：

1. Executor 是 Mybatis 的内部执行器，它负责调用 StatementHandler 操作数据库，并把结果集通过 ResultSetHandler 进行自动映射，另外，它还处理了二级缓存的操作。
2. StatementHandler 是 Mybatis 直接和数据库执行 sql 脚本的对象，另外，它也实现了 Mybatis 的一级缓存。
3. ParameterHandler 是 Mybatis 实现 sql 入参设置的对象。
4. ResultSetHandler 是 Mybatis 把 ResultSet 集合映射成 POJO 的接口对象。

本篇不陈述 **Mybatis** 的内部原理，感兴趣的读取请自行查阅相关资料。

三、案例演示

案例主要演示如何编写 Mybatis 自定义插件来实现动态设置时间参数，解决上文提到的问题。

3.1 自定义注解

```
@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.FIELD})
public @interface CreateTime {

    String value() default "";
}

@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.FIELD})
public @interface UpdateTime {

    String value() default "";
}
```

将两个注解添加到实体类的 Date 类型成员变量上。

```
@Data
public class Category {

    private Integer id;

    private String name;

    private String descr;

    @CreateTime
    private Date createTime;

    @UpdateTime
    private Date updateTime;

}
```

3.2 自定义插件

```
/**
 * 自定义 Mybatis 插件，自动设置 createTime 和 updatTime 的值。
 * 拦截 update 操作（添加和修改）
 */
@Intercepts({ @Signature(type = Executor.class, method = "update", args =
{ MappedStatement.class, Object.class }) })
public class CustomInterceptor implements Interceptor {

    private final Logger logger =
LoggerFactory.getLogger(this.getClass());

    @Override
    public Object intercept(Invocation invocation) throws Throwable {

        MappedStatement mappedStatement = (MappedStatement)
invocation.getArgs()[0];

        // 获取 SQL 命令
        SqlCommandType sqlCommandType =
mappedStatement.getSqlCommandType();

        // 获取参数
        Object parameter = invocation.getArgs()[1];

        if (parameter != null) {
                        // 获取成员变量
            Field[] declaredFields =
parameter.getClass().getDeclaredFields();

            for (Field field : declaredFields) {
                if (field.getAnnotation(CreateTime.class) != null) {
                    if (SqlCommandType.INSERT.equals(sqlCommandType)) { //
insert 语句插入 createTime
                        field.setAccessible(true);
                        if (field.get(parameter) == null) {
                            field.set(parameter, new Date());
                        }
                    }
                }
```

```
                if (field.getAnnotation(UpdateTime.class) != null) { //
insert 或 update 语句插入 updateTime
                    if (SqlCommandType.INSERT.equals(sqlCommandType) ||
SqlCommandType.UPDATE.equals(sqlCommandType)) {
                        field.setAccessible(true);
                        if (field.get(parameter) == null) {
                            field.set(parameter, new Date());
                        }
                    }
                }
            }
        }

        return invocation.proceed();
    }

    @Override
    public Object plugin(Object target) {
        return Plugin.wrap(target, this);
    }

    @Override
    public void setProperties(Properties properties) {
    }
}
```

3.3 注册插件

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration
        PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>
    <settings>
        <!-- 获取数据库自增主键值 -->
        <setting name="useGeneratedKeys" value="true"/>
        <!-- 使用列别名替换列名，默认为 true -->
        <setting name="useColumnLabel" value="true"/>
        <!-- 开启驼峰命名转换：Table(create_time) => Entity(createTime) -->
        <setting name="mapUnderscoreToCamelCase" value="true"/>
    </settings>

    <plugins>
        <plugin interceptor="com.extlight.plugin.CustomInterceptor">
        </plugin>
    </plugins>
</configuration>
```

通过这三个步骤就解决问题了。

四、参考资料

《MyBatis 工作流程及插件开发》