一、功能说明

该文档的目的是是介绍一种可实施且通用的数据编解码方案, 以及工具包: tools-mate的扩展用法。

背景

- 某大厂因泄露用户敏感信息被罚数十亿美元
- 安全部门要求对部分字段进行加密存储,为了尽可能减少开发的工作量, 因此设计了编解码功能

二、工具使用

- 添加依赖: pom.xml

```
<!-- 读写分离依赖包 -->
<dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>tools-mate</artifactId>
    <version>4.20.1</version>
</dependency>
```

- 启用工具

> 在启动类上或其他配置类上扫描包: cn.gmlee.tools

```
@SpringBootApplication
@ComponentScan("cn.gmlee.tools")
public class XxxApp {
    public static void main(String[] args) {
        SpringApplication.run(XxxApp.class, args);
    }
}
```

- 实现接口: CodecServer

> 以下代码是默认代码, 如无重写表示不进行编解码处理

```
    /**
     * 编码.
     * <p>
     *      当mapper处理的数据集为对象或集合对象时触发
     * </p>
     *
     * @param fieldsMap 持久化对象反射后的字段集
     * @param obj       持久化对象
     * @param key       字段名称
     * @param field     字段
     */
    default void encode(Map<String, Field> fieldsMap, Object obj, String
key, Field field) {
    }
```

```
    /**
     * 解码.
     * <p>
     *      当mapper处理的数据集为对象或集合对象时触发
     * </p>
     *
     * @param fieldsMap 反射后的字段集
     * @param obj       持久化对象
     * @param key       字段名
     * @param field     字段
     */
    default void decode(Map<String, Field> fieldsMap, Object obj, String
key, Field field) {
    }
```

三、工具原理

逻辑

- 整个工具围绕着mybatis的拦截器Interceptor实现
- 在mapper持久化前进行拦截, 将持久化对象(或Map)中的属性值进行编码
- 在mapper查询的结果集处理后进行拦截, 将接受对象对象(或Map)中的属性值进行解码

代码

- 拦截器类名: DataAuthInterceptor
- 编码: cn.gmlee.tools.mate.interceptor.DataAuthInterceptor:223
- 解码: cn.gmlee.tools.mate.interceptor.DataAuthInterceptor:354

数据

- 此工具没有产生额外数据

四、使用示例

- 启动类

```
@SpringBootTest
@RunWith(SpringJUnit4ClassRunner.class)
public class LoggerTest {

    @Resource
    LoggerMapper loggerMapper;

    @Test
    public void list() throws Exception {
        PageRequest pr = new PageRequest();
        // 保存对象
        loggerMapper.insert(new Logger("127.0.0.1"));
        // 保存Map
        Map entity = new HashMap();
        entity.put("host", "127.0.0.1");
        loggerMapper.insertMap(entity);
        // 查询对象(接收)
        Page<Logger> page = loggerMapper.list(Page.of(pr.current, pr.size),
"127.0.0.1");
        // 查询Map(接收)
        Page<List<Map>> pageMap = loggerMapper.listMap(Page.of(pr.current,
pr.size), "127.0.0.1");
    }

}
```

- 实现类

```
@Bean
public CodecServer dataAuthServer(){
    return new CodecServer() {
        @Override
        public void encode(Map<String, Field> fieldsMap, Object obj, String
key, Field field) {
            //...
        }

        @Override
        public void decode(Map<String, Field> fieldsMap, Object obj, String
key, Field field) {
            //...
        }
    };
}
```