0.1 接口分层架构与调用关系解析

以下基于用户提供的接口文件信息，整理出 **Mybatis Mapper接口、服务接口、实现类、控制器** 的分层逻辑与调用关系，并通过流程图和代码示例说明各层职责。

1. 一、分层架构核心关系图

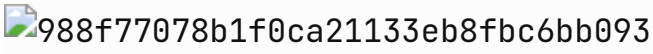

Figure 1

```
┌──────────────┐      ┌──────────────┐      ┌──────────────┐
┌──────────────┐
│  Controller  │  →   │   Service    │  →   │    Impl      │
→   │    Mapper    │
│（请求处理层）│      │（业务接口层）│      │（实现类层）  │
  │（数据操作层）│
└──────────────┘      └──────────────┘      └──────────────┘
└──────────────┘
       ↑                    ↑                    ↑
        ↑
   处理HTTP请求          定义业务逻辑          实现接口方法          执
行SQL/文件操作
```

Fence 1

接口文件	功能	所属目录	备注
Mapper层			

接口文件	功能	所属目录	备注
DictMapper.java	字典映射	mapper	定义字典与数据库字段的映射关系
FileMapper.java	文件元数据操作	mapper	处理文件路径、状态等数据库记录
RoleMapper.java	角色权限映射	mapper	管理角色-权限关联表操作
UserMapper.java	用户数据操作	mapper	用户表的CRUD方法定义
TestFileMapper.java	测试文件操作	mapper	测试环境专用文件接口
Service层			
MenuService.java	菜单服务接口	service	定义动态菜单生成逻辑
UserService.java	用户服务接口	service	用户登录、注册等业务逻辑
AnalyzeService.java	数据分析接口	service	统计与数据分析功能
TranService.java	通信服务接口	service	处理外部系统通信协议
ServiceImpl层			
UserServiceImpl.java	用户服务实现	impl	实现用户相关业务逻辑
MenuServiceImpl.java	菜单服务实现	impl	动态菜单树组装与缓存
CollectServiceImpl.java	数据采集服务实现	impl	实现数据采集与清洗逻辑
Controller层			
MenuController.java	菜单控制器	controller	处理菜单加载HTTP请求
FileController.java	文件操作控制器	controller	管理文件上传/下载接口
UserController.java	用户请求控制器	controller	处理用户登录、注册等HTTP请求
TranController.java	通信请求控制器	controller	处理外部系统通信请求
ExcelController.java	Excel导出控制器	controller	生成并导出Excel报表

Table 1

1.1 目录说明

目录名	说明
mapper	定义数据库实体操作接口（Mybatis Mapper），与SQL语句映射
service	业务逻辑接口层，声明核心功能（如用户管理、数据分析）
impl	服务实现类目录，实现service接口的具体逻辑
controller	控制器层，接收HTTP请求并调用service接口，返回响应结果

Table 2

1.2 关键功能映射

1. 数据操作
    - UserMapper → 用户数据存取
    - FileMapper → 文件元数据管理
2. 业务逻辑
    - UserService → 用户权限校验
    - MenuService → 动态菜单渲染
3. 请求处理

- UserController → 处理/user/*请求
- ExcelController → 导出数据报表

1.3

三、调用关系代码示例

```java
// Controller层调用Service接口
@RestController
public class UserController {
    @Autowired
    private UserService userService;  // 依赖注入Service接口

    @PostMapping("/user/login")
    public ResponseEntity<?> login(@RequestBody UserDTO user) {
        UserVO userVO = userService.login(user);  // 调用Service方法
        return ResponseEntity.ok(userVO);
    }
}

// Service实现类调用Mapper接口
@Service
public class UserServiceImpl implements UserService {
    @Autowired
    private UserMapper userMapper;  // 依赖注入Mapper

    @Override
    public UserVO login(UserDTO user) {
        User userEntity = userMapper.selectByUsername(user.getUsername());  // 调用Mapper
        // 校验密码、生成Token等逻辑
        return convertToVO(userEntity);
    }
}
```