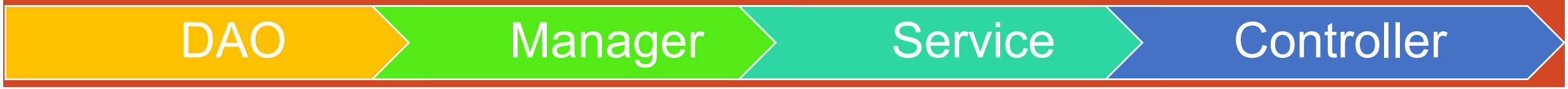
Web 四层结构

Java Web MVC　黄隆

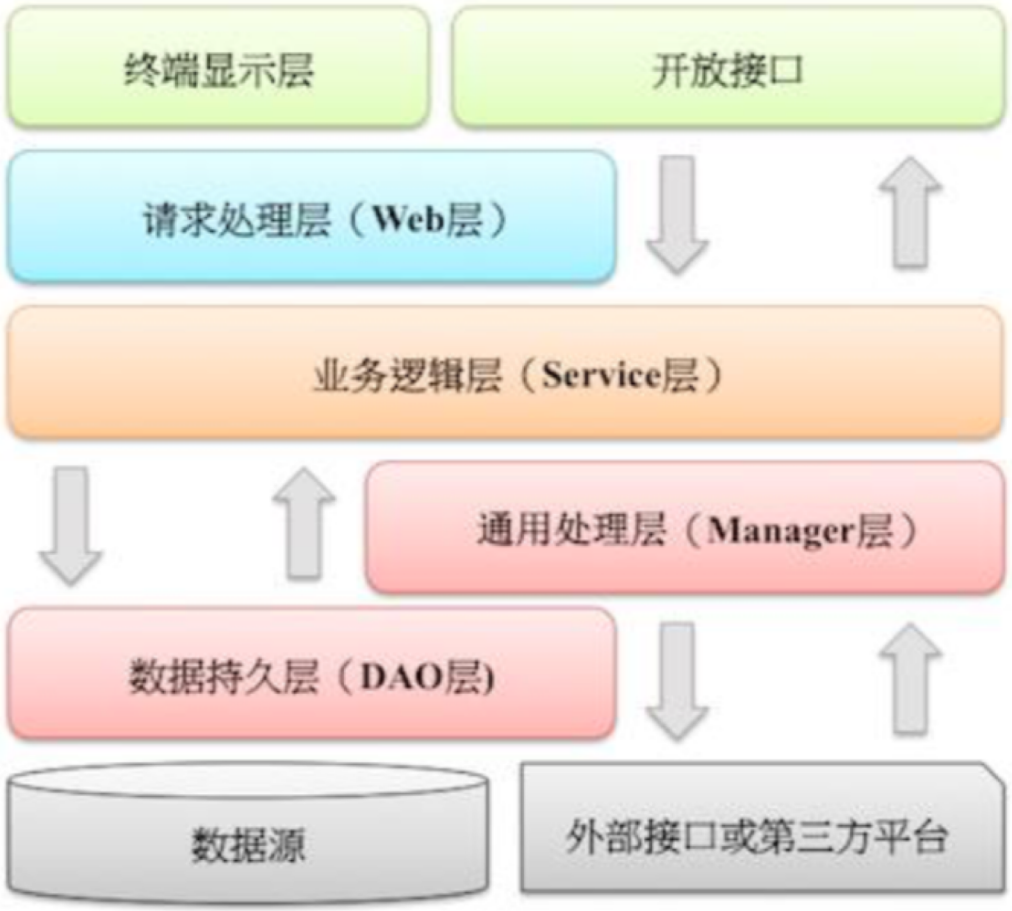


垂直层级

要点

- 面向SOA
- 扩展性强，水平垂直扩展都简单
- 可集中式，可分布式，成长性强
- 不允许横向调用，只能垂直调用

基本架构



参考阿里巴巴java开发手册

开放接口层:可直接封装 Service 方法暴露成 RPC 接口;通过 Web 封装成 http 接口;进行网关安全控制、流量控制等。

终端显示层:各个端的模板渲染并执行显示的层。当前主要是 velocity 渲染，JS 渲染， JSP 渲染，移动端展示等。

Web 层:主要是对访问控制进行转发，各类基本参数校验，或者不复用的业务简单处理等。

Service 层:相对具体的业务逻辑服务层。

Manager 层:通用业务处理层，它有如下特征:

1) 对第三方平台封装的层，预处理返回结果及转化异常信息;
2) 对Service层通用能力的下沉，如缓存方案、中间件通用处理;
3) 与DAO层交互，对多个DAO的组合复用。

DAO 层:数据访问层，与底层 MySQL、Oracle、Hbase 等进行数据交互。

外部接口或第三方平台:包括其它部门RPC开放接口，基础平台，其它公司的HTTP接口

DAO层

- DAO层主要是做数据持久层的工作，主要与**数据库**进行交互
- DAO设计的总体规划需要和设计的表，和实现类之间一一对应
- DAO层更多是单一的SQL语句，没有必要进行事务控制，因为事务开销并不便宜
- DAO层的设计首先是设计DAO的接口，和**mybatis mapper.xml** 实现并映射
- DAO层由Manager层调用，异常往上抛
- DAO层日志由数据库连接池输出

数据持久化

Manager层

- Manager层主要负责业务模块的**逻辑应用**设计
- Manager层被Service层调用，调用DAO层，对多个DAO的组合复用
- Manager层进行事务控制是相当必要的，对于多条SQL进行**事务控制**，如果某个SQL执行失败，那么应当对已经执行的SQL语句进行回滚
- Manager层没有必要进行try-catch，直接往上抛异常就可以。如果非要做日志处理，进行try-catch最后往上抛异常（**不建议处理异常**）
- Manager层封装的业务逻辑有利于业务逻辑的**独立性**和**重复利用性**，程序显得非常简洁
- 对第三方平台封装的层，预处理返回结果及转化异常信息； 对Service层通用能力的下沉，如缓存方案、中间件通用处理;

事务处理层

Service层

- Service层面向的是应用人员，应该返回应用人员能读懂的信息(VO)
- Service层必须做异常处理，一般来说会有统一的异常处理方法
- Service层负责具体的**业务模块流程的控制**
- Service层主要调用Manager层里面的接口控制具体的业务流程
- Service层**处理日志**，**参数校验**，调用**外部请求**等业务流程处理，不建议在Manager层处理异常
- Service层可对外开放

具体业务处理

Controller层

- Controller层的方法对应的是某个URL，只做入口控制
- Controller层只做简单的访问转发
- Controller层简化和View层柔和的复杂度

Service和Controller都可对外开放

小结

每一层分担职责都比较均匀

分工明确-固定层做固定事

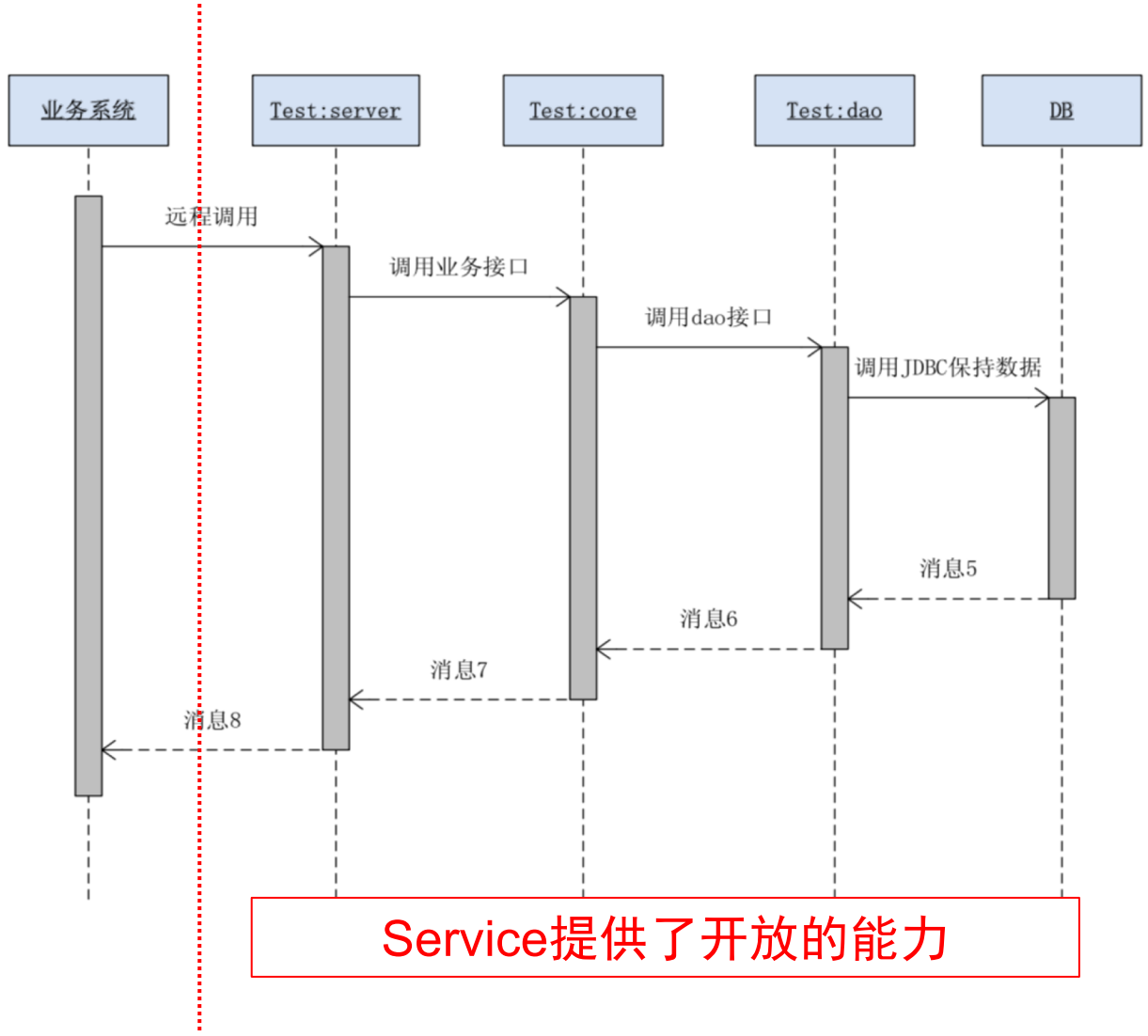
问题定位简单，容易解决问题

分层模型-参考

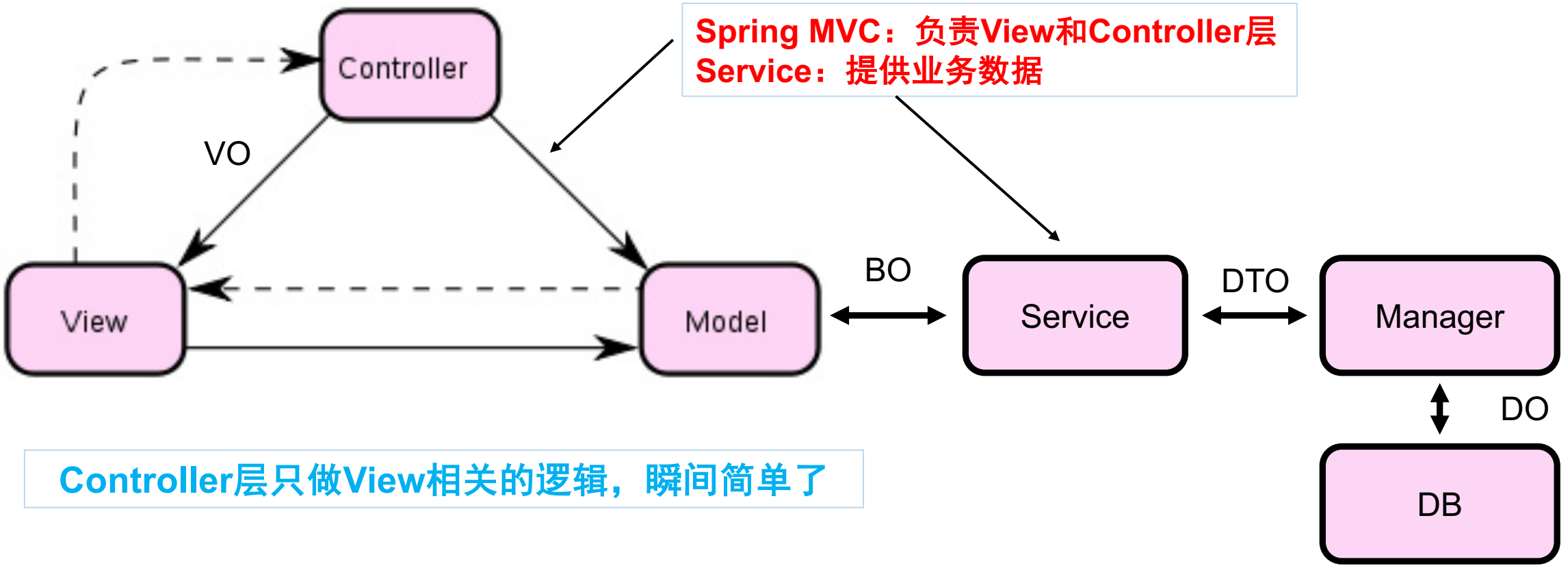
- **DO**(Data Object):与数据库表结构一一对应，通过 DAO 层向上传输数据源对象。
- **DTO**(Data Transfer Object):数据传输对象，Service 或 Manager 向外传输的对象。
- **BO**(Business Object):业务对象。由 Service 层输出的封装业务逻辑的对象。
- **AO**(Application Object):应用对象。在Web层与Service层之间抽象的复用对象模型， 极为贴近展示层，复用度不高。
- **VO**(View Object):显示层对象，通常是 Web 向模板渲染引擎层传输的对象。
- **Query**:数据查询对象，各层接收上层的查询请求。注意超过 2 个参数的查询封装，禁止使用 Map 类来传输

参考阿里巴巴java开发手册

调用链


业务系统
Test:server
Test:core
Test:dao
DB
远程调用
调用业务接口
调用dao接口
调用JDBC保持数据
消息5
消息6
消息7
消息8
Service提供了开放的能力

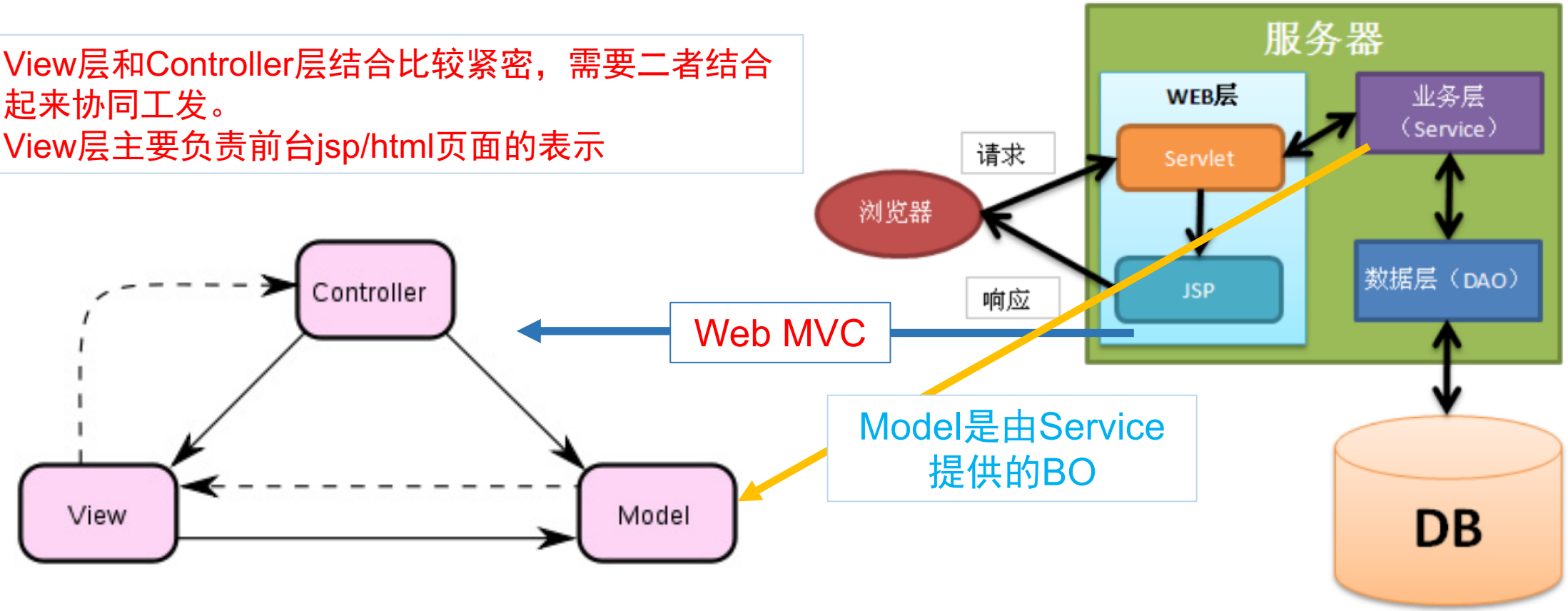
经典MVC



Controller层只做View相关的逻辑，瞬间简单了

经典Java Web框架

View层和Controller层结合比较紧密，需要二者结合起来协同工发。
View层主要负责前台jsp/html页面的表示



示例-Service

```
package com.mycompany.app.user;

public interface UserSevice {
    void addUser();
   void deleteUser();

}
```

Sevice接口声明

```
package com.mycompany.app.user.impl;
import com.mycompany.app.user.UserSevice;
public class DefaultUserAO implements UserSevice {

public void addUser() {
    // TODO Auto-generated method stub

}
public void deleteUser() {

    // TODO Auto-generated method stub }

}
```

Sevice接口实现

示例-Manager

```
package com.mycompany.app.user;

public interface UserManage {

    void addUser();

    void deleteUser();

}
```

Manager接口声明

```
package com.mycompany.app.user.impl;
import com.mycompany.app.user.UserManage;
public class DefaultUserManage implements UserManage {

 @Override

 public void addUser() {
    // TODO Auto-generated method stub

 }

 @Override

 public void deleteUser() {
    // TODO Auto-generated method stub

 } }
```

Manager接口实现