第5章 软件架构

软件架构定义

软件架构为软件系统提供了一个**结构**、**行为**和**属性**的**高级抽象**，由构成系统的**元素的描述**、这些**元素的相互作用**、指导**元素集成的模式**以及这些**模式的约束**组成。软件架构不仅指定了系统的**组织结构**和**拓扑结构**，并且显示了**系统需求和**构成系统的**元素之间的对应关系**，提供了一些**设计决策**的基本原理。

软件架构的发展史

无体系结构设计阶段：以汇编语言进行小规模应用程序开发为特征

萌芽阶段（结构设计主题）：以控制流图和数据流图构成软件结构为特征

初级阶段（类视图、对象视图等不同侧面描述系统的结构模型）：以UML为代表

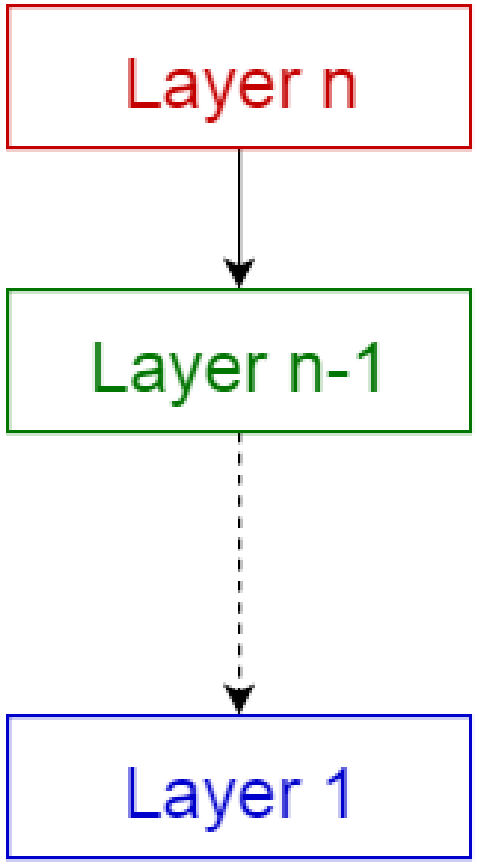
高级阶段（构件化技术和体系结构技术）：以描述系统的高层结构为中心，不关心建模细节，划分了体系结构模型与传统的软件结构的界限。

设计模式与软件架构的区别

1、软件架构是针对系统相对宏观的设计工具，设计模式是针对具体问题相对微观的设计工具，两者的应用**层次**不一样。

2、软件架构总是针对特定应用领域，而设计模式不受限于应用领域。

分层模式（Layered pattern）

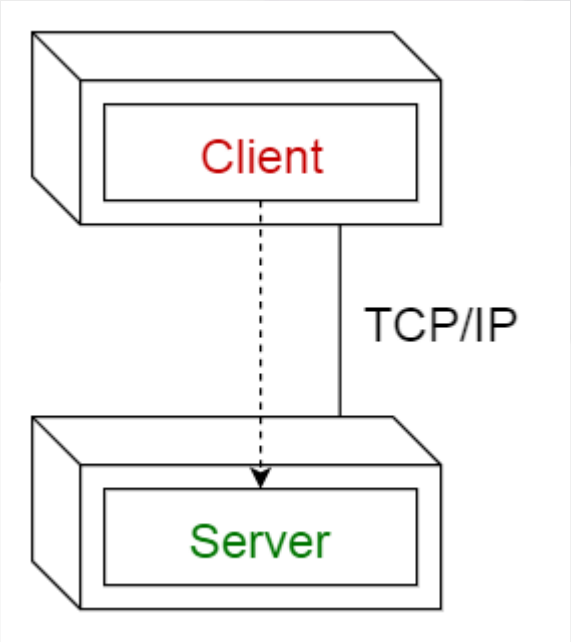


这种模式也称为多层体系架构模式。它可以用来构造可以分解为子任务组的程序，每个子任务都处于一个特定的抽象级别。每个层都为下一个提供更高层次服务。一般信息系统中最常见的是如下所列的4层。

- 表示层(也称为UI层)
- 应用层(也称为服务层)
- 业务逻辑层(也称为领域层)
- 数据访问层(也称为持久化层)

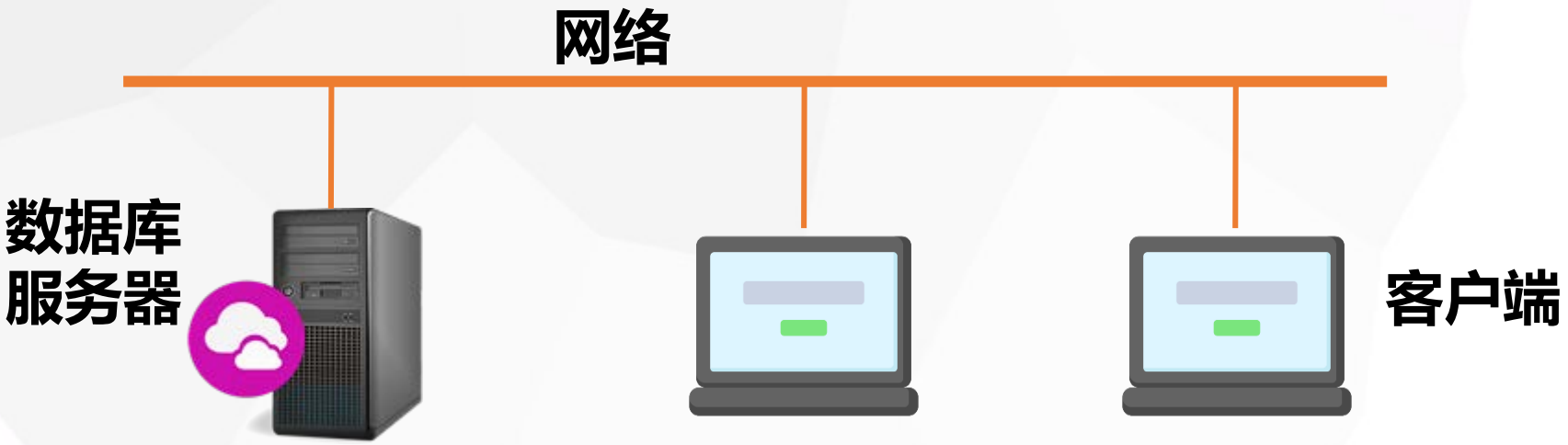
应用领域：桌面应用和商业Web应用

客户机/服务器模式（Client-server pattern）



这种模式由两部分组成：一个服务器和多个客户端。服务器组件将为多个客户端组件提供服务。客户端从服务器请求服务，服务器为这些客户端提供相关服务。此外，服务器持续侦听客户机请求。

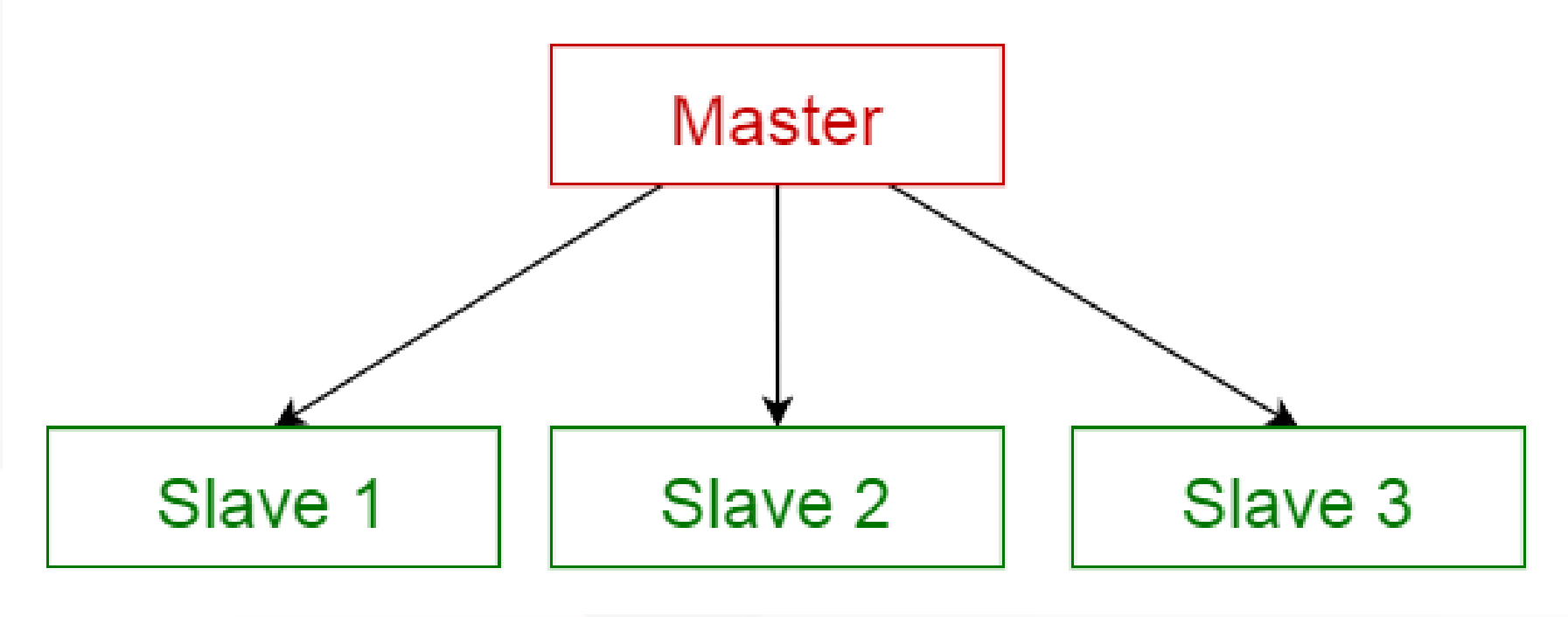
应用领域：电子邮件、即时通讯、银行等在线应用



主-从模式（Master-slave pattern）

这种模式由两方组成：主设备和从设备。主设备组件在相同的从设备组件中分配工作，并计算最终结果，这些结果是由从设备返回的结果。

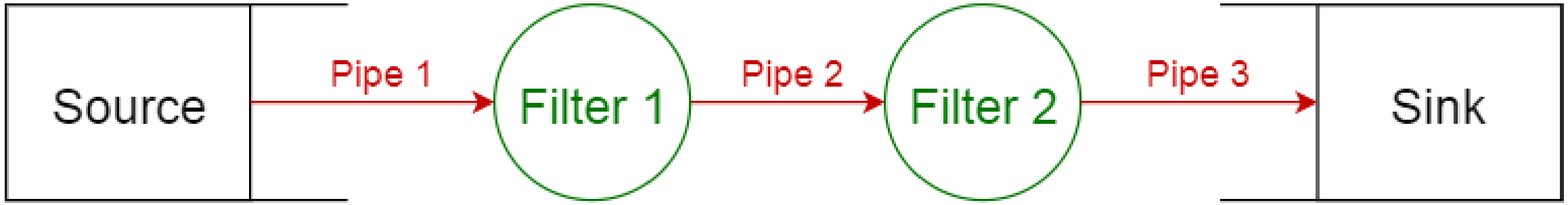
应用领域：分布式计算系统、数据库备份管理



管道-过滤器模式（Pipe-filter pattern）

此模式可用于构造生成和处理数据流的系统。每个处理步骤都封装在一个过滤器组件内。要处理的数据是通过管道传递的。这些管道可以用于缓冲或用于同步。

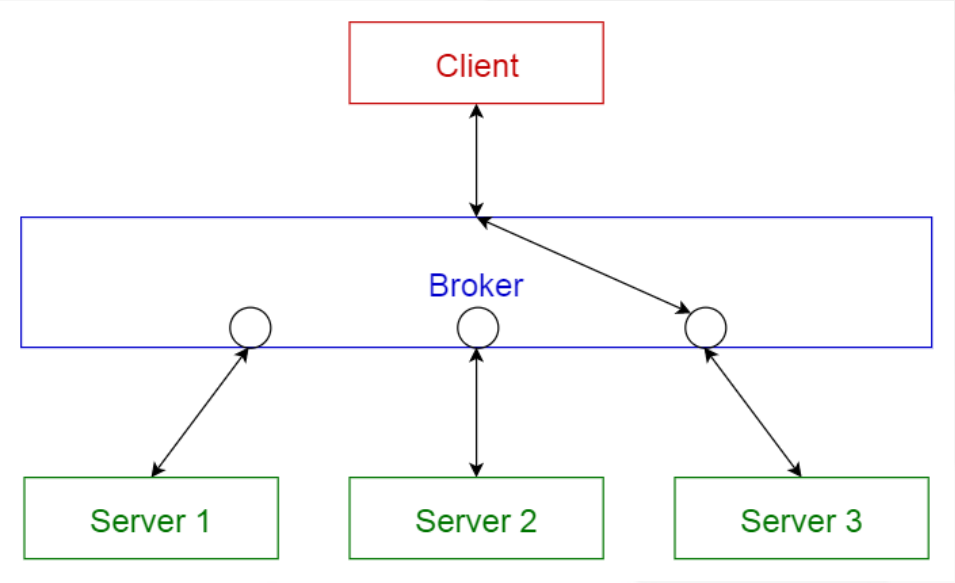
应用领域：编译器，连续的过滤器执行词法分析、解析、语义分析和代码生成



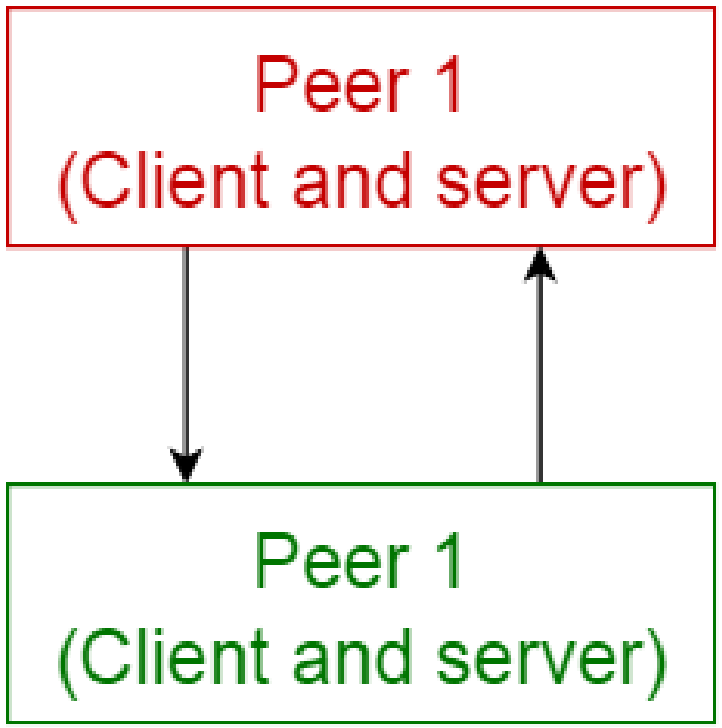
代理模式（Broker pattern）

此模式用于构造具有解耦组件的分布式系统。这些组件可以通过远程服务调用彼此交互。代理组件负责组件之间的通信协调。服务器将其功能(服务和特征)发布给代理。客户端从代理请求服务，然后代理将客户端重定向到其注册中心的适当服务。

应用领域：消息代理软件（Apache ActiveMQ）



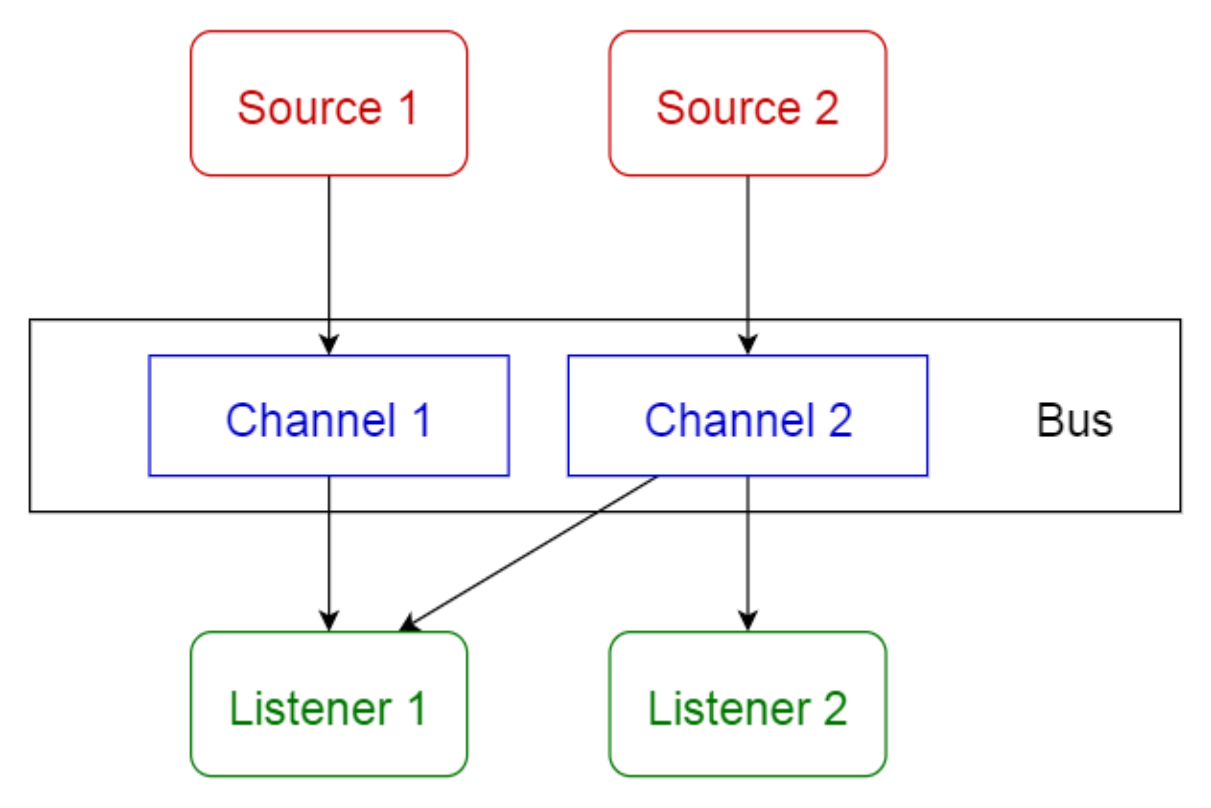
点对点模式（Peer-to-peer pattern）



在这种模式中，单个组件被称为对等点。对等点可以作为客户端，从其他对等点请求服务，作为服务器，为其他对等点提供服务。对等点可以充当客户端或服务器或两者的角色，并且可以随时间动态地更改其角色。

应用领域：文件共享、多媒体协议（P2PTV）、区块链

事件总线模式（Event-bus pattern）

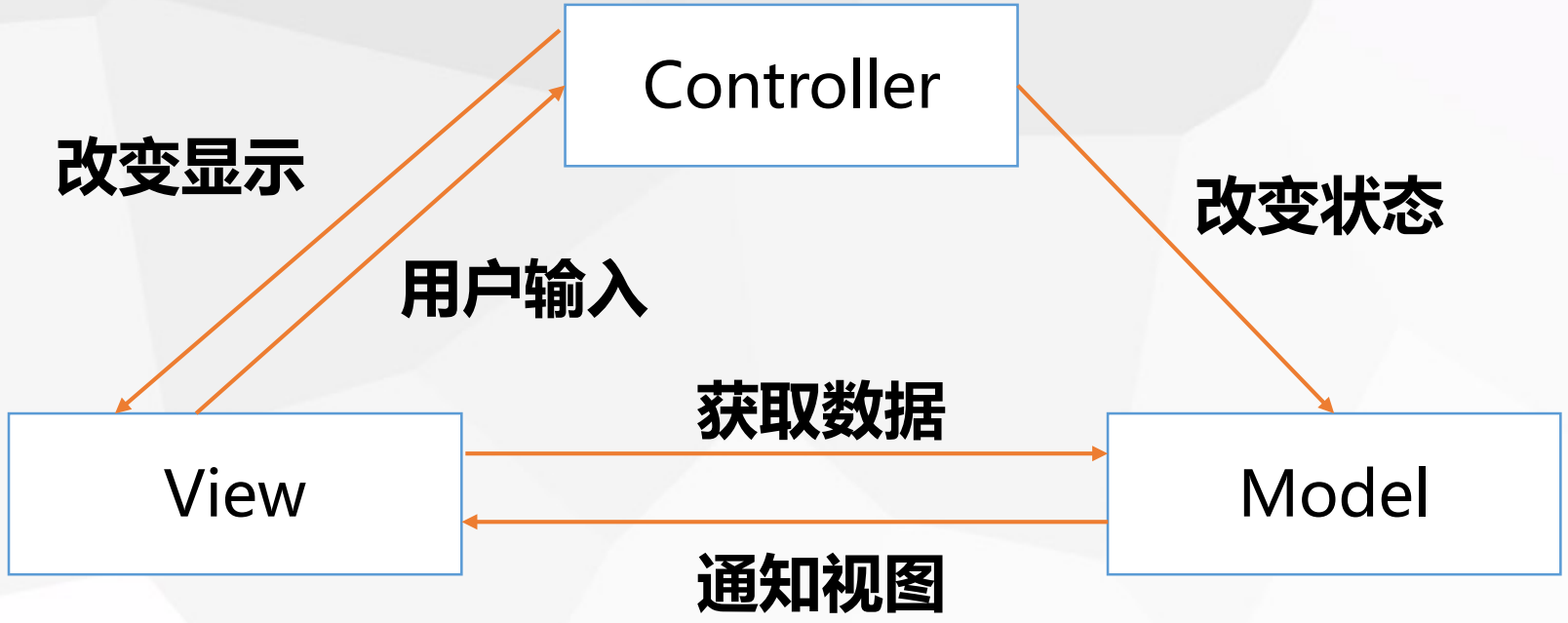


这种模式主要是处理事件，包括4个主要组件：事件源、事件监听器、通道和事件总线。消息源将消息发布到事件总线上的特定通道上。侦听器订阅特定的通道。侦听器会被通知消息，这些消息被发布到它们之前订阅的一个通道上。

应用领域：安卓开发、通知服务

MVC架构模式

MVC模式把用户界面交互分拆到不同的三种角色中，使应用程序被分成三个核心部件：Model（模型）、View（视图）和Controller（控制器）



MVC架构模式

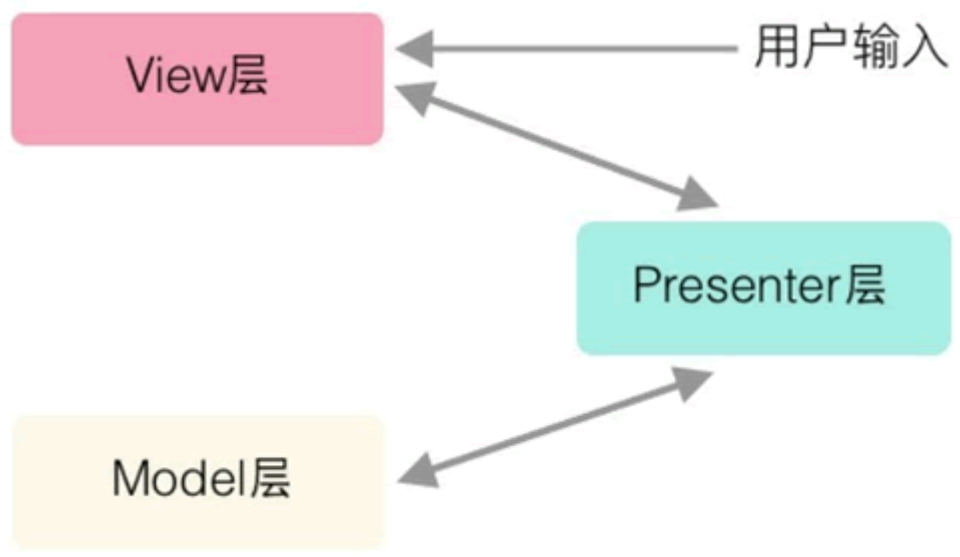
模型持有所有**数据、状态和程序逻辑**，独立于视图和控制器，模型与数据格式无关，能为多个视图提供数据。

视图用来**呈现模型**，是用户看到并与之交互的界面，通常直接从模型中取得其需要显示的状态与数据，对于相同的信息可以有多个不同的显示形式或视图。

控制器位于视图和模型中间，**将输入进行解析并反馈给模型**，通常一个视图具有一个控制器。控制器接收用户的输入并调用模型和视图去完成用户的请求。

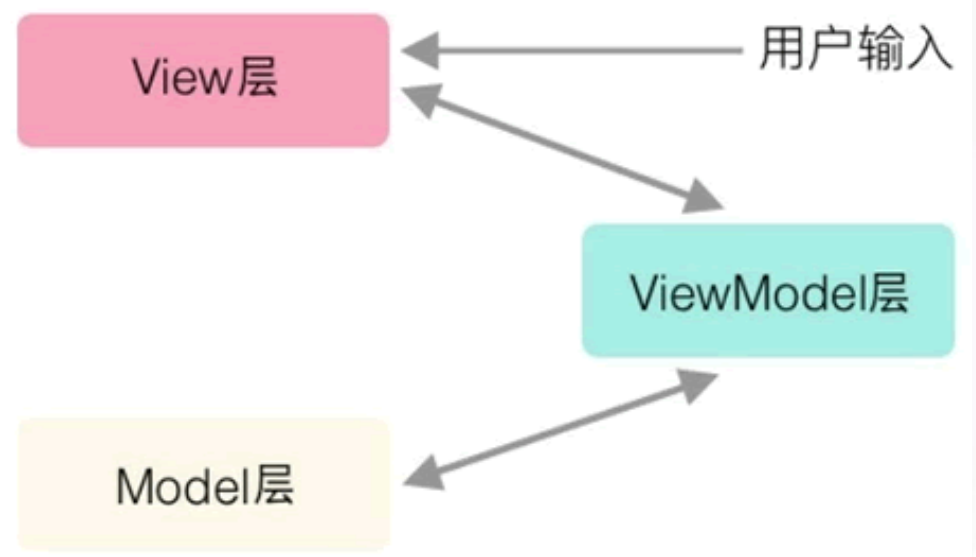
另外两种MV构架

Model-View-Presenter



View层和Model层解耦，复用性上比MVC模型好

Model-View-Presenter

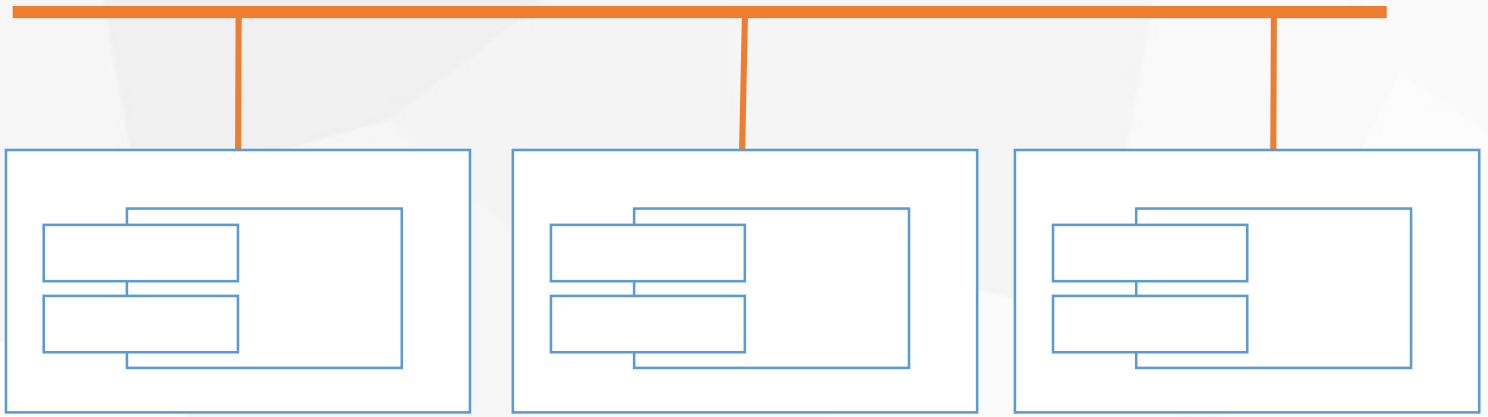


实现View层和Model层数据同步自动化

基于构件的模式

构件是组成软件的基本单位，它包含以下内容：

- 构件是可复用的、自包含的、独立于具体应用的软件对象模块；
- 对构件的访问只能通过其接口进行；
- 构件不直接与别的构件通信。



面向服务的软件架构：SOA

SOA（Service-Oriented Architecture）是一种进行系统开发的体系结构。在SOA构架的系统中，应用程序的功能是由一些**松耦合**，并且具有统一结构定义方式的组件（**服务，Service**）组合构建起来的。

面向服务的软件架构：SOA

SOA技术规范与Web Services

XML标准：规定了服务之间以及服务内部数据交换的格式和结构

SOAP（Simple Object Access Protocol，简单对象访问协议）：用于规范Web服务标准，实现异构程序与平台之间的数据交换。包括三个部分：①封套定义消息内容和处理框架；②一套编码规则用来表示应用程序定义的数据类型的实例；③表示远程过程调用和应答的协定。

WSDL（Web Services Description Language，服务描述语言）：是基于 XML 的用于描述 Web Services 以及如何访问 Web Services 的语言。

面向服务的软件架构：SOA

SOA技术规范与Web Services

UDDI（Universal Discovery Description and Integration，通用服务发现和集成协议）：是一种目录服务，企业可以使用它对 Web services 进行注册和搜索。

面向服务的软件架构：SOA

Rest架构

REST软件架构将网络上所有的资源进行唯一定位，利用支持HTTP的TCP/IP协议来确定互联网资源，无论是图片、文件还是视频等。

REST软件架构遵循**CRUD**原则，该原则规定对于资源需要4种行为：创建（Create）、获取（Read）、更新（Update）和销毁（Delete）。