10个企业级软件架构设计模式

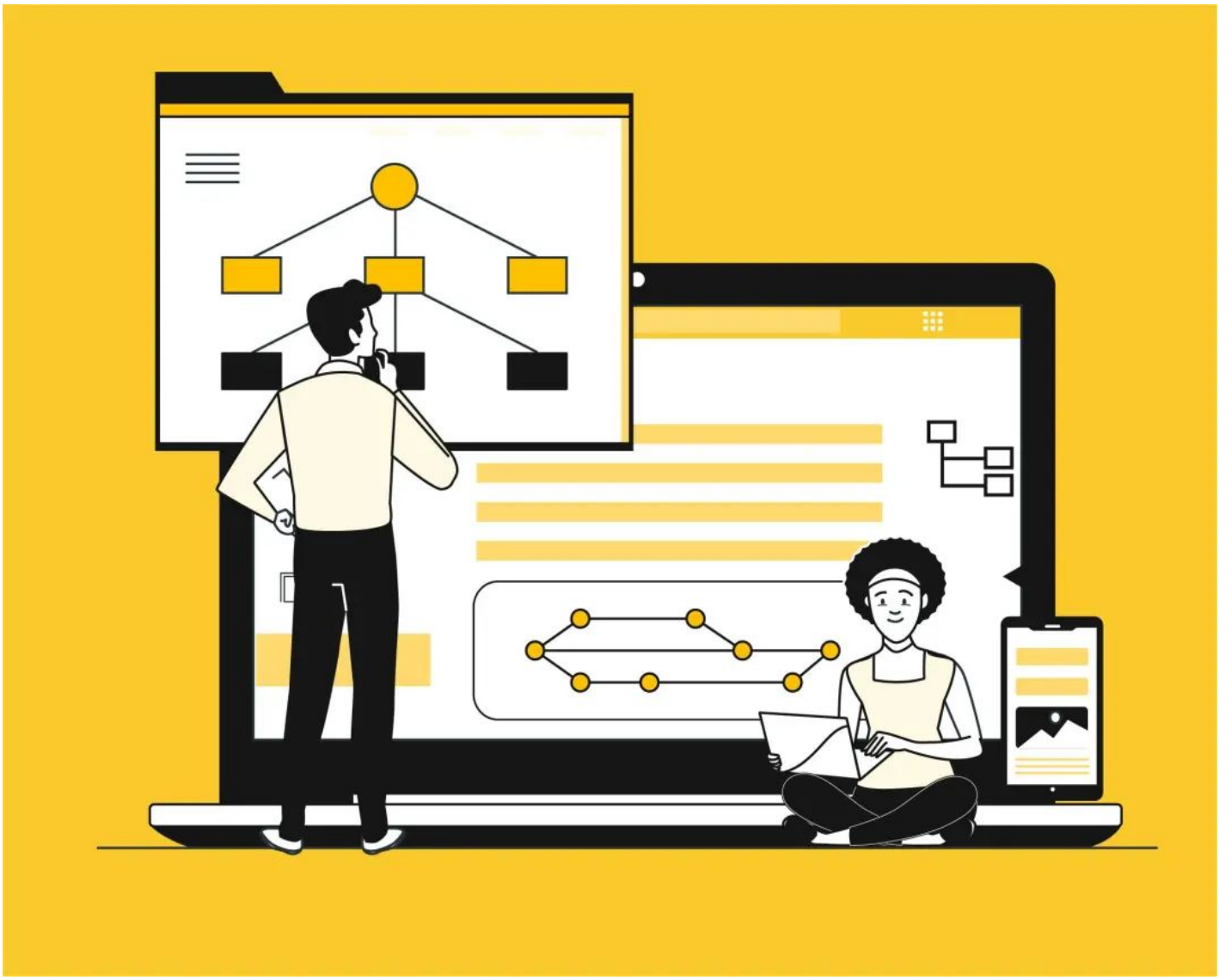
原创 Java学研大本营 已于 2023-11-13 15:50:07 修改 阅读量218 收藏 1 点赞数 5 版权

文章标签： 设计模式

> 本文介绍10个企业级 软件架构设计 模式，附有图解，帮助你更好理解。

微信搜索关注《Java学研大本营》

你是否好奇过大型企业级规模系统是如何设计的？在开发主要软件之前，我们需要选择一个合适的架构来提供所需的功能和质量属性。因此，我们应该了解不同架构的各自的特点以便进行选择。



什么是架构模式 ？

根据 维基百科 ，“架构模式是一种在特定上下文中解决软件架构中常见问题的通用的、可重复使用的解决方案。架构模式类似于软件设计模式，但范围更广泛。”

在本文中将简要解释以下10种常见的架构模式及其用途、优缺点。希望对读者有所帮助！

- 分层模式
- 客户端-服务器模式
- 主从模式
- 管道-过滤器模式

- 代理模式
- 对等模式
- 事件总线模式
- 模型-视图-控制器模式
- 黑板模式
- 解释器模式

1 分层模式

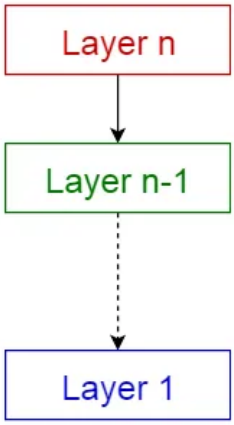
这种模式可用于构建能够分解为多个子任务组的程序，每个子任务组都处于特定的抽象层级，每个层级为下一个更高层级提供服务。即每个层级只需要关注其相邻的上下层级，可以减少系统的复杂度、提高系统的可维护

通常情况下，分层模式会把一个系统划分为四个层级：

- 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（也称为持久层）

用途：

- 通用桌面应用程序。
- 电子商务Web应用程序。

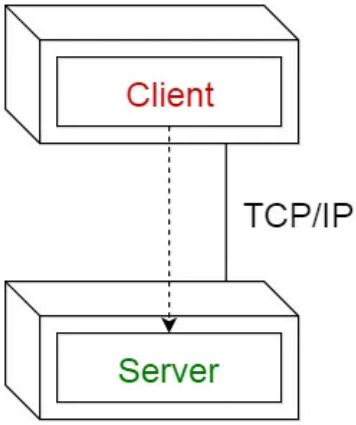


2 客户端-服务器模式

这种模式由两个组件组成：服务器和多个客户端。服务器组件向多个客户端组件提供服务，客户端向服务器请求服务。客户端向服务器发送请求，服务器响应这些请求并向客户端提供相应的服务。同时，服务器会持续监听客户端的请求，以便及时响应客户端的需求。

用途：

- 在线应用程序，如电子邮件、文档共享和银行业务。

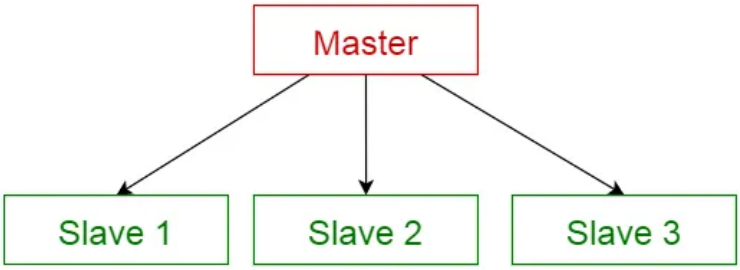


3 主从模式

这种模式由两个组件组成：主组件和从组件。主组件将任务分配给相同的从组件，并从从组件返回的结果计算最终结果。

用途：

- 在数据库复制中，主数据库被认为是权威源，而从数据库则与之同步。
- 计算机系统中连接到总线的外围设备（主设备和从设备）。

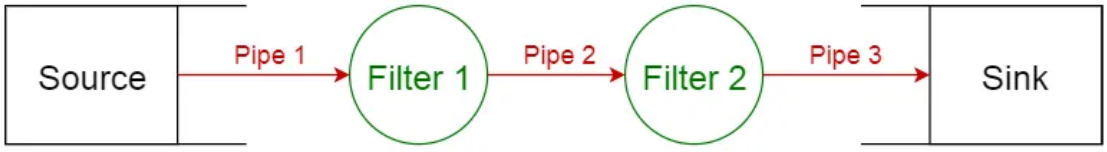


4 管道-过滤器模式

该模式可用于构建生成和处理数据流的系统。每个处理步骤都封装在过滤器组件中，数据在不同的过滤器之间通过管道传递。这些管道可用于缓冲或同步目的。

用途：

- 编译器。连续的过滤器执行词法分析、语法分析、语义分析和代码生成。
- 生物信息学中的工作流程。



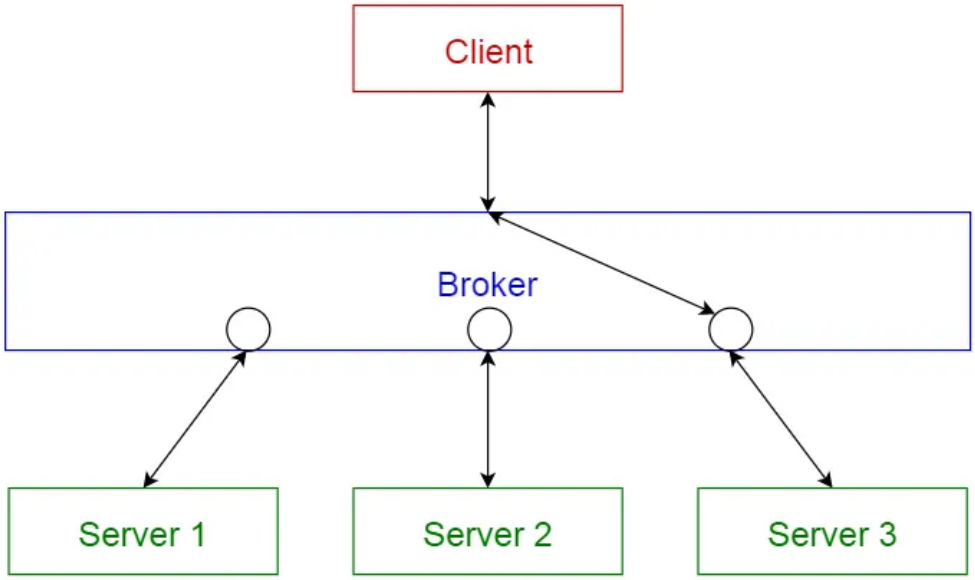
5 代理模式

该模式用于构建分布式系统的解耦组件，这些组件可以通过远程服务调用相互交互，代理组件负责协调组件之间的通信。

通常情况下，服务端将自己的能力（服务和特性）发布到一个代理中。当客户端需要请求服务时，它会从代理处请求服务。代理会根据客户端的需求，从其注册表中找到合适的服务并将客户端重定向到该服务。这种方式，代理可以隐藏服务的实际位置和实现细节，从而提高系统的可扩展性和安全性。

用途：

- 消息代理软件，如Apache ActiveMQ、Apache Kafka、RabbitMQ和JBoss Messaging。

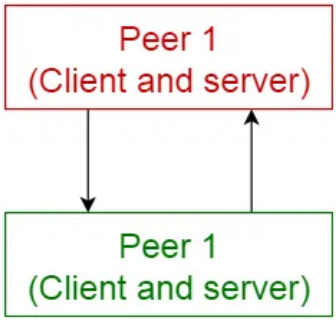


6 对等模式

在该模式中，个体组件被称为对等方。对等方可以同时作为客户端，向其他对等方请求服务，也可以作为服务器，向其他对等方提供服务。一个对等方可以作为客户端、服务器或两者兼而有之，并且可以动态地改变其角色。

用途：

- 文件共享网络，如Gnutella和G2。
- 多媒体协议，如P2PTV和PDTP。
- 基于加密货币的产品，如Bitcoin和Blockchain。

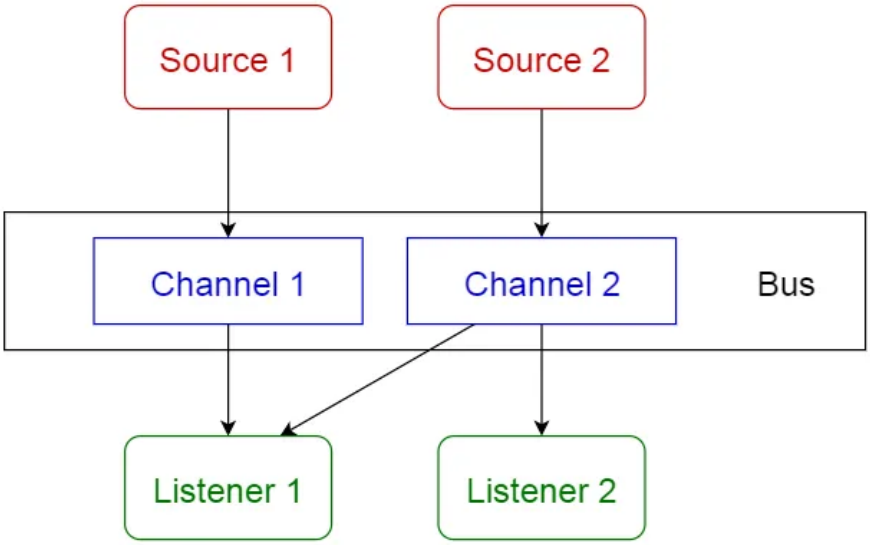


7 事件总线模式

这种模式主要用于处理事件，由四个主要组件组成：事件源、事件监听器、通道和事件总线。事件源负责发布消息并将其发送到事件总线上的特定通道上。监听器订阅特定通道，并在事件总线上等待特定的消息。当事件源发布消息到与其相应的通道上时，事件总线会将该消息广播给所有已订阅该通道的监听器。这样，监听器就可以接收并处理事件源发布的消息。通过这种方式，事件总线模式可以帮助构建具有高度解耦的、基于事件驱动的系统。

用途：

- Android开发。
- 通知服务。



8 模型-视图-控制器模式

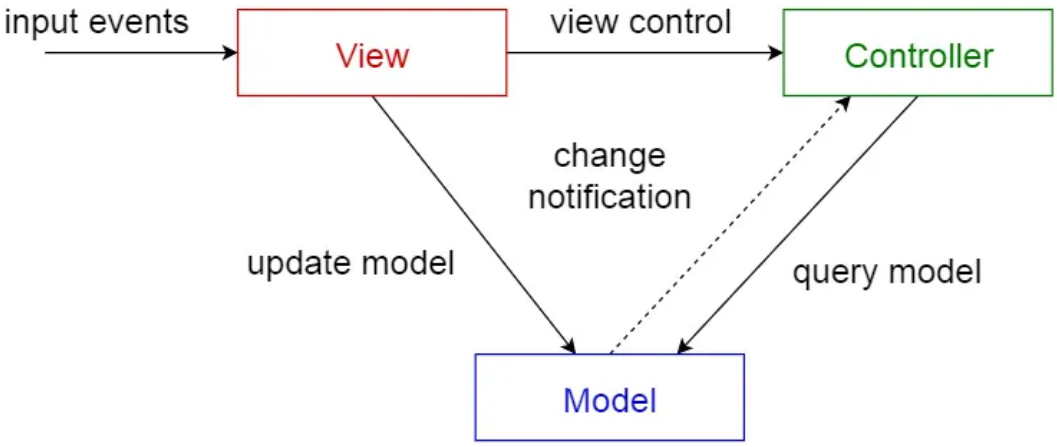
该模式也称为MVC模式，把交互式应用程序分为三个部分：

- 模型 - 包含核心功能和数据。
- 视图 - 将信息显示给用户（可以定义多个视图）。
- 控制器 - 处理用户输入。

这样做是为了把信息的内部表示与向用户呈现和接受信息的方式分离开来。它解耦了组件并允许有效的代码重用。

用途：

- 用于主要编程语言中的World Wide Web应用程序的架构。
- Web框架，如Django和Rails。



9 黑板模式

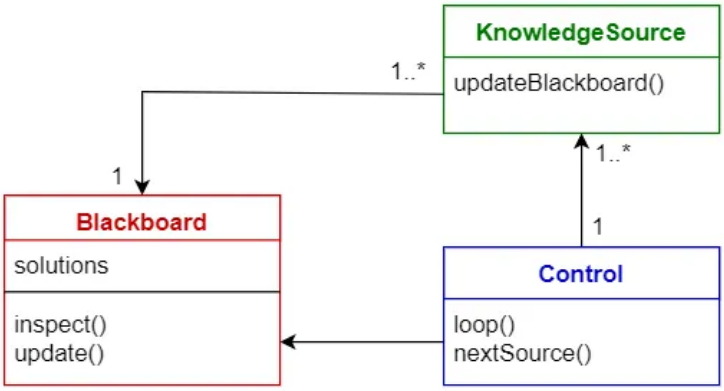
该模式适用于没有确定性解决策略的问题。黑板模式由三个主要组件组成：

- 黑板 - 包含解决空间中的对象的结构化全局内存。
- 知识源 - 具有自己的表示的专业模块。
- 控制器 - 选择、配置和执行模块。

所有组件都可以访问黑板，组件可以生成新的数据对象，并将其添加到黑板上。组件在黑板上查找特定类型的数据，并可以通过与现有知识源的模式匹配来找到这些数据。

用途：

- 语音识别。
- 车辆识别和跟踪。
- 蛋白质结构识别。
- 声纳信号解释。

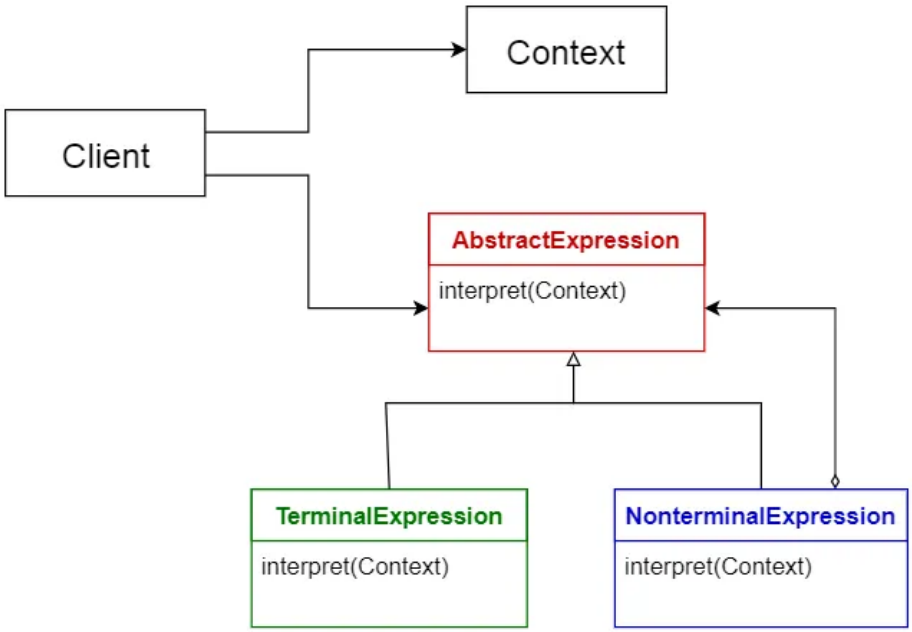


10 解释器模式

这种模式用于设计解释特定语言编写的程序的组件。它主要指定如何评估程序中的行（句子或表达式），这些行是用特定语言编写的。基本思想是为语言中的每个符号都定义一个类，这些类可以根据符号的不同来解释和执行相应的操作。通过这种方式，解释器模式可以帮助构建出灵活、易于扩展的程序，从而提高程序的可维护性和可重用性。

用途：

- 数据库查询语言，如SQL。
- 用于描述通信协议的语言。



推荐书单