典型的软件体系结构风格

管道-过滤器体系结构风格

管道-过滤器体系结构风格为处理数据流的软件系统架构提供了一种参考结构.它是由过滤器和管道组成的。每个处理步骤都被封装在一个过滤器组件中，数据流通过相邻过滤器之间的管道进行传输。每个过滤器可以被单独修改，功能单一，并且它们之间的顺序可以进行配置。

构件类型:过滤器(数据处理构件)

连接件类型:管道(过滤器间的连接件)

特点:

- 功能特性
  - 处理或者转换输入数据流
  - 对数据流的处理可以容易地分成几个独立的处理步骤
- 非功能特性
  - 系统的升级要求可以通过替换/增加/重组过滤器实现.有时甚至由使用者完成操作.(系统运行时配置)
  - 不同的处理步骤不共享信息(过滤器构件间松耦合)

过滤器是数据流的处理功能单元。根据相邻过滤器间交互方式，可以分为以下两种类型:

- 被动过滤器
  - 过滤器从上一个相邻过滤器采用拉入方式读取数据
  - 过滤器采用压出方式输出数据给下一个相邻过滤器
- 主动过滤器
  - 过滤器同时支持以上两种方式的读写数据。　　管道是实现过滤器构件间交互的连接件
- 如果管道连接两个主动过滤器，那么管道需要进行缓冲和同步.
- 管道可以实现数据在过滤器之间的数据转换，将一个过滤器的输出数据格式转换为其后接的过滤器的数据输入格式.

优点:

- 高内聚低耦合
- 通过过滤器的增加/移除/重组可实现数据流处理系统的灵活性/可扩展性
- 过滤器构件具有可重用性
- 有利于系统的维护与更新
- 可支持局部步骤的并行吹了以提高效率

缺点:

- 增量式处理数据，存在效率问题
- 数据格式转换的问题：数据转换额外开销
- 不适合交互式应用系统

基于事件的隐式调用体系结构风格

基于事件的系统结构风格中，构件间通过事件机制交互。系统中的事件处理者构件可以通过事件注册来订阅它所关心的事件相关联。当某一事件发生时，系统会通知所有与这个事件相关联的事件处理者构件，即一个事件的激发导致了事件处理者构件与事件源构件间的隐式交互。

构件类型:

- 事件源构件
- 事件处理者构件
- 事件分发者构件

连接件类型:

- 事件对象(Event)

适用场景:

- 松耦合的异步运行系统

优点:

- 系统具有很好的灵活性，系统易于伸缩扩展

缺点:

- 系统控制权的问题
- 数据的交换问题

分层体系结构风格

适用场景:

- 一个需要分解的大系统
- 系统规格说明描述了高层任务，并希望可移植性
- 系统的外部边界要求系统附带功能接口
- 高层任务到平台的映射不是直接的
- 系统需要满足以下非功能特性
  - 后期源代码的改动不应该影响到整个系统
  - 系统的各个部分应该可以替换。构件应该可以有不同的实现而不影响其他的构件
  - 提高构件的内聚性
  - 复杂构件的进一步分解
  - 设计和开发系统时，工作界限必须清楚

模式:

- 将系统分成适当层次，按适当次序放置.
- 从最低层抽象层次开始，以梯状把抽象层次n放在n-1层顶部。直到功能顶部
- 第n层提供的绝大多数服务由第n-1层提供的服务组成
- 每个层次是一个独立的组件。它的责任是：提供了由上层使用的服务，并且委派给下一层次。
- 不允许较高层次直接越级访问较低层次

层构件内部子构件端口保护：为每个层构件指定一个单一的端口>br>

- 为保证层构件具有良好的封装性，第N层对于第N+1层是黑盒
- 可以考虑设计一个flat接口，并且把这个接口封装在一个facade(外观)对象中.
- 灰盒：上层能够知道下层由哪些组件组合而成，但是不知道每个组件内部结构
- 应该尽量使用黑盒的方式，除非有一些特殊的原因：比如效率问题

构件：各层中包含的构件
连接件:层间的交互协议

优点:

- 层构件的封装性，可重用性，可替换性
- 系统的局部依赖特性

缺点:

- 层构件间的依赖性，特别是当低层构件的修改影响高层构件的时候，可能引起底层之上的多个层构件的修改.
- 效率问题:顶层构件到底层构件之间需要进行层层的参数传递/转换等

仓库风格

以数据为中心的体系结构风格
构件：独立构件,仲裁者
连接件:仓库
适用场景:

- 以数据为中心的分布式系统

优点:

- 可扩展性
- 可维护性
- 安全性
- 并行处理性

缺点:

- 单一失败点

Date:2017.6.15