



软件架构设计-架构风格对比

序号	架构风格名	常考关键字及实例	简介
1	数据流-批处理	传统编译器，每个阶段产生的结果作为下一个阶段的输入，区别在于整体。	一个接一个，以整体为单位。
2	数据流-管道-过滤器		一个接一个，前一个输出是后一个输入。
3	调用/返回-主程序/子程序		显示调用，主程序直接调用子程序。
4	调用/返回-面向对象	无	对象是构件，通过对象调用封装的方法和属性。
5	调用/返回-层次结构		分层，每层最多影响其上下两层，有调用关系。
6	独立构件-进程通信		进程间独立的消息传递，同步异步。
7	独立构件-事件驱动(隐式调用)	事件触发推动动作，如程序语言的语法高亮、语法错误提示。	不直接调用，通过事件驱动。
8	虚拟机-解释器	自定义流程，按流程执行，规则随时改变，灵活定义，业务灵活组合。 机器人。	解释自定义的规则，解释引擎、存储区、数据结构。
9	虚拟机-规则系统		规则集、规则解释器、选择器和工作内存，用于DSS和人工智能、专家系统。
10	仓库数据库	现代编译器的集成开发环境IDE,以数据为中心。 又称为数据共享风格	中央共享数据源，独立处理单元。
11	仓库超文本		网状链接，多用于互联网。
12	仓库-黑板		语音识别、知识推理等问题复杂、解空间很大、求解过程不确定的这一类软件系统，黑板、知识源、控制。
13	闭环过程控制	汽车巡航定速，空调温度调节，设定参数，并不断调整。	发出控制命令并接受反馈，循环往复达到平衡。
14	C2风格	构件和连接件、顶部和底部。	通过连接件绑定在一起按照一组规则运作的并行构件网络



软件架构设计-案例分析常见的架构风格

架构风格	主要特点	主要优点	主要缺点	适合领域
管道-过滤器	过滤器相对独立	功能模块复用；可维护性和可扩展性较强；具有并发性；模块独立性高	不适于交互性强的应用，对于存在关系的数据流必须进行协调	系统可划分清晰的模块；模块相对独立；有清晰的模块接口。
面向对象	力争实现问题空间和软件系统空间结构的一致性。	高度模块性；实现封装；代码共享灵活；易维护；可扩充性好	增加了对象之间的依赖关系	多种领域
事件驱动	系统由若干子系统构成且称为一个整体；系统有统一的目标；子系统有主从之分；每一个子系统有自己的事件收集和处理机制	适合描写系统组；容易实现并发处理和多任务；可扩展性好；具有类层次结构；简化代码；	因为树型结构所以削弱了对系统计算的控制能力；各个对象的逻辑关系复杂	一个系统对外部的表现可以从它对事件的处理表征出来。
分层风格	各个层次的组件形成不同功能级别的虚拟机；多层相互协同工作，而且实现透明	支持系统设计过程中的逐级抽象；可扩展性好；支持软件复用	不同层次之间耦合度高的系统很难实现	适合功能层次的抽象和相互之间低耦合的系统
数据共享风格	采用两个常用构件中央数据单元和一些相对独立的组件集合	中央数据单元实现了数据的集中，以数据为中心	适合于特定领域	适合于专家系统等人工智能领域问题的求解
解释器风格	系统核心是虚拟机	可以用多种操作来解释一个句子，灵活应对自定义场景	适合于特定领域	适合于模式匹配系统与语言编译器
闭环控制风格	通过不断的测量被控对象，认识和掌握被控对象；将控制理论引入体系结构构建	将控制理论引入到计算机软件体系结构中。	适合于特定领域	该系统中一定存在有目标的作用、信息处理闭环控制过程。



架构风格真题详解

阅读以下关于软件架构风格的说明，在答题纸上回答问题1和问题2。

【说明】某软件公司为其新推出的字处理软件设计了一种脚本语言，专门用于开发该字处理软件的附加功能插件。为了提高该语言的编程效率，公司组织软件工具开发部门为脚本语言研制一套集成开发环境。软件工具开发部门根据字处理软件的特点，对集成开发环境进行了需求分析，总结出以下3项核心需求：

(1)集成开发环境需要提供对脚本语言的编辑、语法检查、解释、执行和调试等功能的支持，并要实现各种功能的灵活组合、配置与替换。

(2)集成开发环境需要提供一组可视化的编程界面，用户通过对界面元素拖拽和代码填充的方式就可以完成功能插件核心业务流程的编写与组织。

(3)在代码调试功能方面，集成开发环境需要实现在脚本语言编辑界面中的代码自动定位功能。具体来说，在调试过程中，编辑界面需要响应调试断点命中事件，并自动跳转到当前断点处所对应的代码。

针对上述需求，软件工具开发部门对集成开发环境的架构进行分析与设计，王工认为该集成开发环境应该采用管道-过滤器的架构风格实现，李工则认为该集成开发环境应该采用以数据存储为中心的架构风格来实现。公司组织专家对王工和李工的方案进行了评审，最终采用了李工的方案。

【问题1】(12分)请用200字以内的文字解释什么是软件架构风格，并从集成开发环境与用户的交互方式、集成开发环境的扩展性、集成开发环境的数据管理三个方面说明为什么最终采用了李工的设计方案。

【问题2】(13分)在对软件系统架构进行设计时，要对架构需求进行分析，针对特定需求选择最为合适的架构风格，因此实际的软件系统通常会混合多种软件架构风格。请对核心需求进行分析，说明为了满足需求(2)和(3)，分别应采用何种架构风格，并概要说明采用相应架构风格后的架构设计过程。



架构风格真题详解

试题答案

【问题1】软件架构风格是指描述特定软件系统组织方式的惯用模式。组织方式描述了系统的组成构件和这些构件的组织方式，惯用模式则反映众多系统共有的结构和语义。

从集成开发环境与用户的交互方式看，用户通常采用交互式的方式对脚本语言进行编辑、解释执行与调试。在这种情况下，采用以数据存储为中心的架构风格能够很好地支持交互式数据处理，而管道-过滤器架构风格则对用户的交互式数据处理支持有限。

从集成开发环境的扩展性来看，系统核心需求要求实现各种编辑、语法检查、解释执行等多种功能的灵活组织、配置与替换。在这种情况下，采用以数据存储为中心的架构风格，以数据格式解耦各种功能之间的依赖关系，并可以灵活定义功能之间的逻辑顺序。管道-过滤器架构风格同样以数据格式解耦数据处理过程之间的依赖关系，但其在数据处理逻辑关系的灵活定义方面较差。

从集成开发环境的数据管理来看，集成开发环境需要支持脚本语言、语法树(用于检查语法错误)、可视化模型、调试信息等多种数据类型，并需要支持数据格式的转换。以数据存储为中心的架构将数据存储统一的中心存储器中，中心存储器能够表示多种数据格式，并能够为数据格式转换提供各种支持。管道-过滤器架构风格通常只能支持有限度的数据格式，并且在数据格式转换方面的灵活性较差。



架构风格真题详解

【问题2】

为了满足需求(2)，应该采用解释器架构风格。具体来说，需要：

- ①为可视化编程元素及其拖拽关系定义某种语言，并描述其语法与语义；
- ②编写解释器对该语言进行解释；
- ③生成对应的脚本语言程序。

为了满足需求(3)，应该采用隐式调用架构风格。具体来说，首先需要定义“断点在调试过程中命中”这一事件，并实现当断点命中后的屏幕定位函数。集成开发环境维护一个事件注册表结构，将该事件与屏幕定位函数关联起来形成注册表中的一个记录项。在调试过程中，集成开发环境负责监听各种事件，当“断点在调试过程中命中”这一事件发生时，集成开发环境查找事件注册表，找到并调用屏幕定位函数，从而实现脚本语言编辑界面与调试代码的自动定位。

软件架构设计-C/S架构



两层C/S架构：客户端和服务端都有处理功能，现在已经不常用，原因有：开发成本较高、客户端程序设计复杂、信息内容和形式单一、用户界面风格不一、软件移植困难、软件维护和升级困难、新技术不能轻易应用、安全性问题、服务器端压力大难以复用。



三层C/S架构：**将处理功能独立出来**，表示层和数据层都变得简单。表示层在客户机上，功能层在应用服务器上，数据层在数据库服务器上。即将两层C/S架构中的数据从服务器中独立出来了。其优点下面四点：

- **各层在逻辑上保持相对独立**，整个系统的逻辑结构更为清晰，能提高系统和软件的可维护性和可扩展性；
- 允许灵活有效的选用相应的平台和硬件系统，具有良好的可升级性和开放性；
- 各层可以并行开发，各层也可以选择各自最适合的开发语言；
- 功能层有效的隔离表示层与数据层，为严格的安全管理奠定了坚实的基础，**整个系统的管理层次也更加合理和可控制**。



三层C/S架构**设计的关键在于各层之间的通信效率**，要慎重考虑三层间的通信方法、通信频度和数据量，否则即使分配给各层的硬件能力很强，性能也不高。



软件架构设计-C/S架构



三层B/S架构： 是三层C/S架构的变种，将客户端变为用户客户端上的浏览器，将应用服务器变为网络上的WEB服务器，又称为0客户端架构，虽然不用开发客户端，但有很多缺点：

- 使用浏览器作为客户端的话安全性难以控制；
- 在数据查询等响应速度上，要远远低于C/S架构，因为C/S架构有部分数据存储在本地；
- 数据提交一般以页面为单位，数据的动态交互性不强。

混合架构风格



- 内外有别模型：企业内部使用C/S,外部人员访问使用B/S。
- 查改有别模型：采用B/S查询，采用C/S修改。
- 混合架构实现困难，且成本高。



软件架构设计-RIA



富互联网应用RIA：弥补三层B/S架构存在的问题，**RIA是一种用户接口**，比用HTML实现的接口更加健壮，且有可视化内容，本质还是网站模式，其优点如下：

- **RIA结合了C/S架构反应速度快、交互性强的优点与B/S架构传播范围广及容易传播的特性；**
- **RIA简化并改进了B/S架构的用户交互；**
- **数据能够被缓存在客户端**，从而可以实现一个比基于HTML的响应速度更快且数据往返于服务器的次数更少的用户界面。
- **本质还是O客户端**，借助于高速网速实现必要插件在本地的快速缓存，增强页面对动态页面的支持能力，典型的如小程序。



软件架构设计-MVC架构



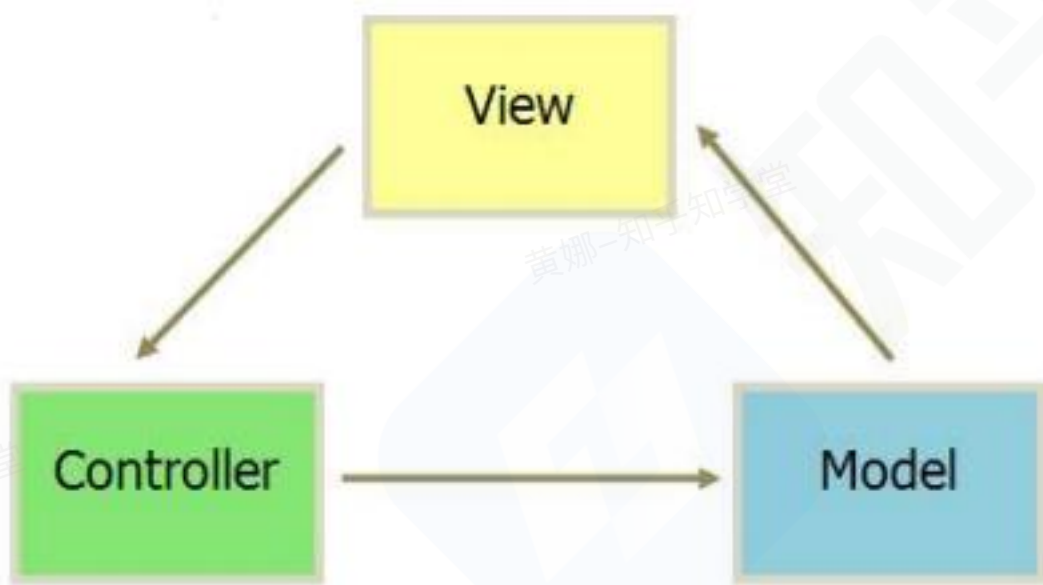
控制器(Controller)：是应用程序中**处理用户交互的部分**。通常控制器负责从视图读取数据，控制用户输入,并向模型发送数据。



模型(Model)：是应用程序中用于**处理应用程序数据逻辑的部分**。通常模型对象负责在数据库中存取数据。模型表示业务数据和业务逻辑。



视图(View)：是应用程序中**处理数据显示的部分**。通常视图是依据模型数据创建的。**是用户看到并与之交互的界面**。视图向用户显示相关的数据，并能接收用户的输入数据，但是它并不进行任何实际的业务处理：



MVC分层的好处：

- MVC分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注一个方面。例如，您可以在不依赖业务逻辑的情况下专注于视图设计。同时也让应用程序的测试更加容易。
- MVC分层同时也简化了分组开发。不同的开发人员可同时开发视图、控制器逻辑和业务逻辑。

软件架构设计-MVP架构

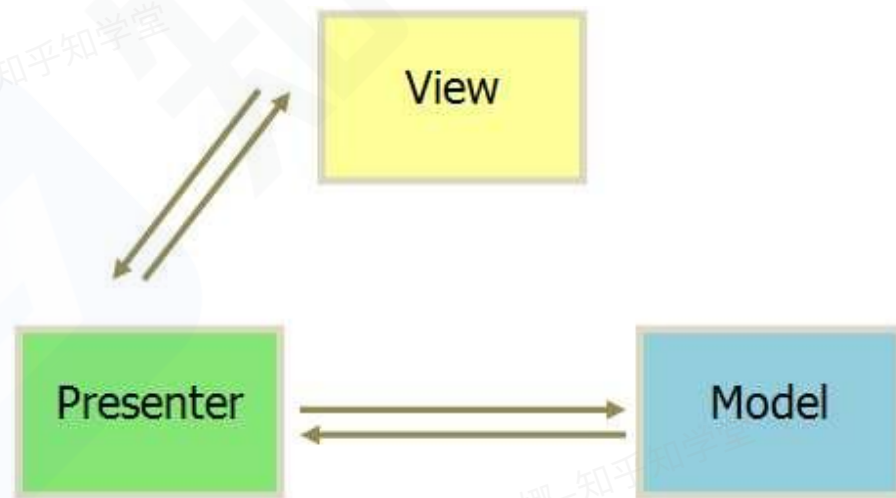


MVP是把MVC中的**Controller**换成了**Presenter(呈现)**，目的就是完全切断**View**跟**Model**之间的联系，由**Presenter**充当桥梁，做到**View-Model**之间通信的完全隔离。



MVP特点:

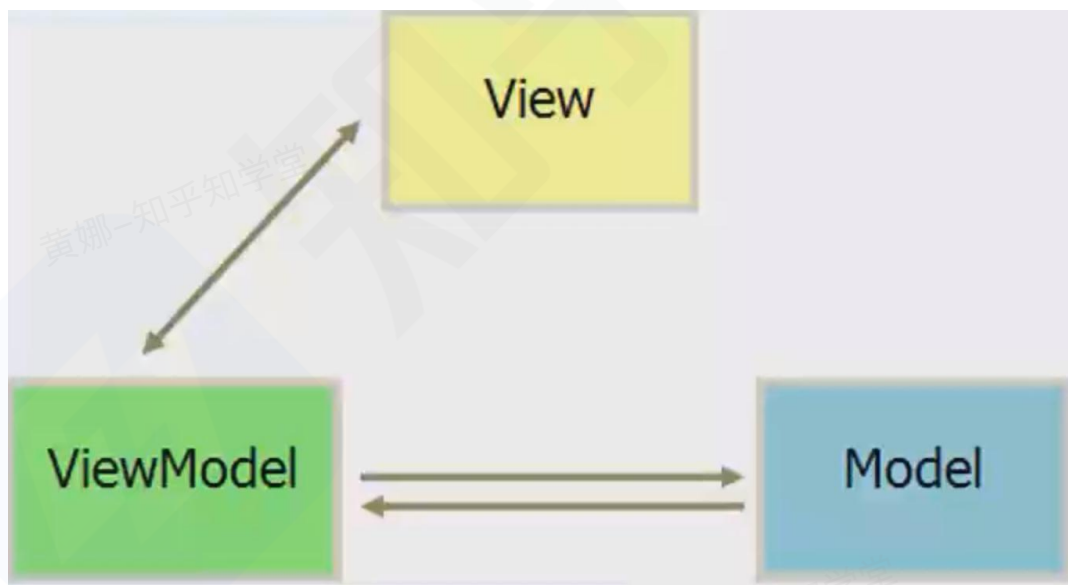
- M、V、P之间双向通信。
- View 与Model不通信，都通过Presenter传递。Presenter完全把Model和View进行了分离，主要的程序逻辑在Presenter里实现。
- View非常薄，不部署任何业务逻辑，称为“被动视图”(PassiveView)，即没有任何主动性，而Presenter非常厚，所有逻辑都部署在那里。
- Presenter与具体的View是没有直接关联的，而是通过定义好的接口进行交互，从而使得在变更View时候可以保持Presenter的不变，这样就可以重用。



软件架构设计-MVVM架构

 MVVM: MVVM模式和MVC模式类似, 主要目的是分离视图(View)和模型(Model), 实现双向绑定, 有几点优点:

- 低耦合, 视图(View)可以独立于Model变化和修改, 一个ViewModel可以绑定到不同的"View"上, 当View变化的时候Model可以不变, 当Model变化的时候View也可以不变。
- 可重用性, 可以把一些视图逻辑放在一个ViewModel里面, 让很多view重用这段视图逻辑。
- 独立开发, 开发人员可以专注于业务逻辑和数据的开发(ViewModel), 设计人员可以专注于页面设计。
- 可测试, 界面向来是比较难于测试的, 而现在测试可以针对ViewModel来写。





层次架构真题详解

阅读以下关于软件系统设计的叙述，在答题纸上回答问题1至问题3。【说明】

某文化产业集团委托软件公司开发一套文化用品商城系统，业务涉及文化用品销售、定制、竞拍和点评等板块，以提升商城的信息化建设水平。该软件公司组织项目组完成了需求调研，现已进入到系统架构设计阶段。考虑到系统需求对架构设计决策的影响，项目组先列出了可能影响系统架构设计的部分需求如下：

- (a) 用户界面支持用户的个性化定制；
- (b) 系统需要支持当前主流的标准和服务，特别是通信协议和平台接口；
- (c) 用户操作的响应时间应不大于3秒，竞拍板块不大于1秒；
- (d) 系统具有故障诊断和快速恢复能力；
- (e) 用户密码需要加密传输；
- (f) 系统需要支持不低于2G的数据缓存；
- (g) 用户操作停滞时间超过一定时限需要重新登录验证；
- (h) 系统支持用户选择汉语、英语或法语三种语言之一进行操作。

项目组提出了两种系统架构设计方案：瘦客户端c/s架构和胖客户端c/s架构，经过对上述需求逐条分析和讨论，最终决定采用瘦客户端c/s架构进行设计。



层次架构真题详解

【问题1】(8分)

在系统架构设计中，决定系统架构设计的非功能性需求主要有四类：操作性需求、性能需求、安全性需求和文化需求。请简要说明四类需求的含义。

【问题2】(8分)

根据表的分类，将题干所给出的系统需求(a)~(h)分别填入(1)~(4)。需求分类如下

需求类别	系统需求
操作性需求	(1)
性能需求	(2)
安全性需求	(3)
文化需求	(4)

【问题3】(9分)

请说明瘦客户端c/s架构能够满足题干中给出的哪些系统需求(只需要回答出三个系统需求)。



层次架构真题详解

参考答案:

【问题1】

系统性能需求: 指响应时间、吞吐量、准确性、有效性、资源利用率等与系统完成任务效率相关的指标。可靠性、可用性等指标归为此类。

安全性需求: 系统向合法用户提供服务并阻止非授权用户使用服务方面的系统需求。

操作性需求: 与用户操作使用系统相关的一些需求。

文化需求: 带有文化背景因素的系统需求。

【问题2】

(1)a,b

(2)c,d,f

(3)e,g

(4)h

【问题3】

(a)瘦的把业务逻辑从客户端放到了服务上。

(b)胖和瘦无明显差异

(c)胖客户端, 在客户端的运算能力强一些。瘦客户端可以在服务端用集群做支持

(d)瘦客户端将业务逻辑迁移到应用服务器上, 所以故障只要修复服务器上的内容, 而胖客户端要更新所有客户端, 工作量大, 所以此情况下瘦客户端有优势。

(e)胖客户端的后端是数据库, 没有业务逻辑, 此时要做加密传输没有基础, 但瘦客户端可以做到。

(f)胖客户端做到2G数据缓存很容易, 而瘦客户端不现实。

(g)瘦客户端与胖客户端均可做到。

(h)瘦客户端与胖客户端均可做到。