**论软件系统架构风格**

历年真题

**2015**

2. 论软件系统架构风格

系统架构风格（System Architecture Style）是描述某一特定应用领域中系统组织方式的

惯用模式.架构风格定义了一个词汇表和一组约束，词汇表中包含一些构件和连接件类型，

而这组约束指出系统是如何将这些构件和连接件组合起来的口软件系统架构风格反映了领

域中众多软件系统所共有的结构和语义特性，并指导如何将各个模块和子系统有效地组织成

一个完整的系统。软件系统架构风格的共有部分可以使得不同系统共享同一个实现代码，系

统能够按照常用的、规范化的方式来组织，便于不同设计者很容易地理解系统架构。

请以“软件系统架构风格” 论题，依次从以下三个方面进行论述：

1．概要叙述你参与分析和开发的软件系统开发项目以及你所担任的主要工作。

2．分析软件系统开发中常用的软件系统架构风格有哪些？详细阐述每种风格的具体含

义。

3．详细说明在你所参与的软件系统开发项目中，采用了哪种软件系统架构风格，具体

实施效果如何.

写作要点：

一、简要描述所参与分析和开发的软件系统开发项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、分析在软件系统开发中常用的软件系统架构风格，详细阐述每种风格的具体含义。

软件系统开发中常用的软件系统架构风格主要包括：

1.管道和过滤器风格

在管道/过滤器风格的软件体系结构中，每个构件都有一组输入和输出，构件读输入的数据流，经过内部处理，然后产生输出数据流。这个过程通常通过对输入流的变换及增量计算来完成，所以在输入被完全消费之前，输出便产生了。因此，这里的构件被称为过滤器，这种风格的连接件就像是数据流传输的管道，将一个过滤器的输出传到另一过滤器的输入。此风格特别重要的过滤器必须是独立的实体，它不能与其他的过滤器共享数据，而且一个过滤器不知道它上游和下游的标识。一个管道/过滤器网络输出的正确性并不依赖于过滤器进行增量计算过程的顺序。

2. 数据抽象和面向对象风格

抽象数据类型概念对软件系统有着重要作用，目前软件界已普遍转向使用面向对象系统。这种风格建立在数据抽象和面向对象的基础上，数据的表示方法和它们的相应操作封装在一个抽象数据类型或对象中。这种风格的构件是对象，或者说是抽象数据类型的实例。对象是一种被称作管理者的构件，因为它负责保持资源的完整性。对象是通过函数和过程的调用来交互的。

3. 基于事件的隐式调用风格

基于事件的隐式调用风格的思想是构件不直接调用一个过程，而是触发或广播一个或多个事件。系统中的其他构件中的过程在一个或多个事件中注册，当一个事件被触发，系统自动调用在这个事件中注册的所有过程，这样，一个事件的触发就导致了另一模块中的过程的调用。基于事件的隐式调用风格的主要特点是事件的触发者并不知道哪些构件会被这些事件影响。这样不能假定构件的处理顺序，甚至不知道哪些过程会被调用，因此，许多隐式调用的系统也包含显式调用作为构件交互的补充形式。

4. 层次系统风格

层次系统组织成一个层次结构，每一层为上层服务，并作为下层客户。在一些层次系统中，除了一些精心挑选的输出函数外，内部的层只对相邻的层可见。这样的系统中构件在一些层实现了虚拟机(在另一些层次系统中层是部分不透明的)。连接件通过决定层间如何交互的协议来定义，拓扑约束包括对相邻层间交互的约束。这种风格支持基于可增加抽象层的设计。这样，允许将一个复杂问题分解成一个增量步骤序列的实现t由于每一层最多只影响两层，同时只要给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强大的支持。

5. 仓库风格

在仓库风格中，有两种不同的构件：中央数据结构说明当前状态，独立构件在中央数据存储上执行，仓库与构件间的相互作用在系统中会有大的变化。控制原则的选取产生两个主要的子类。若输入流中某类时间触发进程执行的选择，则仓库是一传统型数据库；另一方面，若中央数据结构的当前状态触发进程执行的选择，则仓库是一黑板系统。

6. C2风格

C2体系结构风格可以概括为:通过连接件绑定在一起的按照一组规则运作的并行构件网络。C2风格中的系统组织规则如下：系统中的构件和连接件都有一个顶部和一个底部；构件的顶部应连接到某连接件的底部，构件的底部则应连接到某连接件的顶部，而构件与构件之间的直接连接是不允许的；一个连接件可以和任意数目的其他构件和连接件连接；当两个连接件进行直接连接时，必须由其中一个的底部到另一个的顶部。

三、针对作者实际参与的软件系统开发项目，说明所采用的软件架构风格，并描述该架构风格所产生的实际应用效果。

我的范文

根据业务需求以及既有系统的实际情况, 我们在此次架构设计中采用了层次结构、管道过滤器、事件驱动、数据库系统这四种架构风格. 下面我将具体介绍每种架构风格的选择过程以及使用效果.

首先是**层次架构风格**. 考虑到不同用户对交通数据需求的不同, 我采用了B/S和C/S架构并存的混合层次架构风格. 在交通管理部门角度, 采用了C/S架构, 即数据库服务器+客户端应用程序. 主要是考虑到其作为数据的责任方, 不但需要查询多种数据, 还需要对相关数据进行维护操作, 如修改道路信息、新增路段限行公告、修改交通设施信息等, 对交互性和性能要求较高; 另外, 为更好地支持动态页面, 集成数据库处理功能, 增强数据交互性, 简化OLTP应用, 且考虑到交通管理部门工作人员使用终端的网络与系统服务器处于一个局域网内网中, 因此采用二层C/S架构; 在公众出行者用户角度, 采用了B/S架构, 即数据库服务器+web服务器+客户端浏览器. 用户通过公网对交通数据进行查询与简单的交互, 使用浏览器或者其它移动终端就能得到服务和功能, 该架构具有良好的扩展性, 具有更好的易扩展性和安全性.

其次是数据流(**管道过滤器**)风格. 在处理车辆GPS数据的过程中, 我采用了数据流风格.平台的功能之一是对原始的车辆GPS数据进行处理, 生成车辆轨迹数据, 并将车辆轨迹以可视化线矢量数据形式予以展示. GPS数据量非常大, 且从原始数据得到最终数据需要经过多个步骤的处理, 因此我设计了多个构件, 即过滤器, 每个过滤器的功能不同, 包括删除研究区域外数据、变换数据时间和经纬度格式、删除重复和异常数据、对数据按车辆进行分组排序、生成标准线矢量数据等. 每个过滤器读入GPS数据流, 经过内部处理, 产生输出数据流, 并通过连接件(管道)传给下一个过滤器. 该风格通过构件实现功能的解耦, 便于需求变更和功能修改, 逻辑清晰, 易于维护. 很好的完成了海量GPS数据的处理需求.

然后是**事件驱动**风格. 在路况更新频率的控制中, 我采用了事件驱动风格架构. 在本平台中, 路况依据道路车辆通行速度分为畅通、缓行和拥堵三个级别, 而计算路况等级需要融合视频监控数据、雷达数据、GPS数据等多源数据, 运算性能消耗较大, 因此在路况更新频率方面, 采用了默认状态下(无人工干预)每15分钟更新一次, 当有用户提交”立即更新路网”的请求时, 系统立即执行路网更新运算模块, 以提供给用户实时数据. 具体的实现方法是先定义一个”用户要求立即更新路网”的事件, 以及”更新路网状态”的构件过程, 并将该过程注册到事件中, 当事件发生时, 系统自动调用”更新路网状态”的构件过程. 该架构风格为路网更新构件的维护和演化带来了方便.

最后是**数据库系统**风格. 在道路路段、交通相关用地等多属性数据的存储上, 采用了数据库系统风格. 以道路路段为例, 路段包含路段线型空间矢量数据、车道数、道路名称、道路等级等众多属性, 这些属性由多个部门的不同人员共同维护, 并实现共享. 另一方面, 公众出行者的用户视图也需要获取到最新版本的路段数据. 通过采用数据库系统, 由中央共享数据源保存了当前交通数据的状态, 多个独立处理元素对数据进行操作, 实现了数据共享的同时, 还保证了数据高速访问、数据一致性和完整性.

历年真题

**2. 论企业应用系统的分层架构风格**

软件架构风格是描述一类特定应用领域中系统组织方式的惯用模式，反映了领域中诸多

系统所共有的结构特征和语义特征，并指导如何将各个模块和子系统有效组织成一个完整的

系统。分层架构是一种常见的软件架构风格，能够有效简化设计，使得设计的系统结构清晰，

便于提高复用能力和产品维护能力。

由于大量企业应用系统都由界面呈现、业务逻辑、数据存储三类功能构成，因此广泛采用分层架构风格进行系统设计。

请围绕“ 企业应用系统的分层架构风格” 论题，依次从以下三个方面进行论述。

1. 概要叙述你参与管理和开发的企业应用系统建设项目以及你在其中所承担的主要工

作。

2．请结合项目实际情况，指出应用系统都有哪些层次以及每个层次的主要功能。

3．请结合项目实际情况，指出设计每个层次时需要注意的问题及相应的解决方案。

一、简要描述所参与管理和开发的企业应用系统建设项目，并明确指出在其中承担的主要任务和开展的主要工作。

二、需要结合项目实际情况指出所开发的应用系统的总体架构，特别是架构的层次关系。分层架构设计是一种常见的架构设计方法，能够有效简化设计，使设计fit系统结构清晰，便于提高复用能力和产品维护能力。一般来说，企业应用系统的架构可以分为表现层、中间层和持久层三个层次。

1. 表现层。表现层主要负责接收用户的请求，对用户的输入、输出进行检查与控制，处理客户端的一些动作，包括控制页面跳转等，并向用户呈现最终的结果信息。表现层主要采用MVC结构来实现。控制器负责接收用户的请求，并决定应该调用哪个模型来处理；然后，模型根据用户请求调用中间层进行相应的业务逻辑处理，并返回数据;最后，控制器调用相应的视图来格式化模型返回的数据，并通过视图呈现给用户。

2. 中间层(业务逻辑层)。中间层主要包括业务逻辑层组件、业务逻辑层工作流、业务逻辑层实体和业务逻辑层框架四个方面。

a业务逻辑层组件分为接口和实现类两个部分，接口用于定义业务逻辑组件必须实现的方法。通常按模块来设计业务逻辑组件，每个模块设计为一个业务逻辑组件，并且每个业务逻辑组件以多个DAO组件作为基础，从而实现对外提供系统的业务逻辑服务。

b业务逻辑层工作流能够实现在多个参与者之间按照某种预定义的规则传递文档、信息或任务的过程自动进行，从而实现某个预期的业务目标，或者促进此目标的实现。

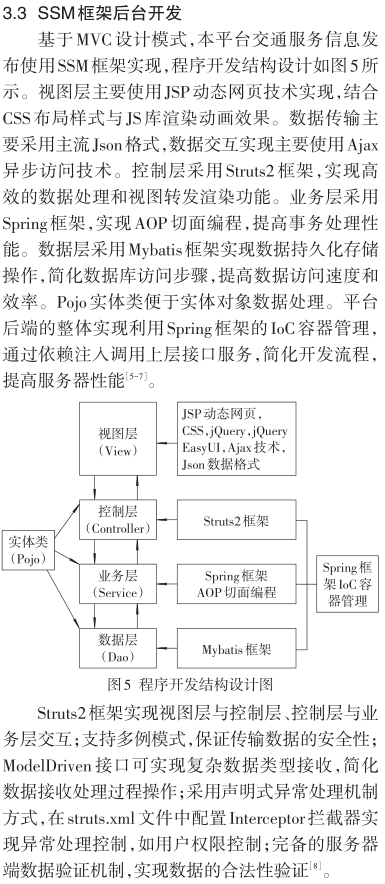
c业务逻辑层实体提供对业务数据及相关功能的状态编程访问，业务逻辑层实体数据可以使用具有复杂架构的数据来构建，这种数据通常来自数据库中的多个相关表。业务逻辑层实体数据可以作为业务过程的部分I/O参数传递，业务逻辑层的实体是可序列化的，以保持它们的当前状态。

业务逻辑层是实现系统功能的核心组件，采用容器的形式，便于系统功能的幵发、代码重用和管理。

3. 持久层。持久层主要负责数据的持久化存储，主要负责将业务数据存储在文件、数据库等持久化存储介质中。持久层的主要功能是为业务逻辑提供透明的数据访问、持久化、加载等能力。

三、考生需要结合项目实际情况，举例说明在设计表现层、中间层和持久层时需要考虑的主要问题，例如：在表现层设计时需要考虑MVC模型中的模型、视图和控制器分别对应哪些组件；在中间层设计时需要考虑框架与业务组件之间的关系；在持久层设计时需要考虑如何支持对多种类型数据的透明访问。





Spring是一个轻量级的控制反转（IoC）和面向

切面（AOP）的容器框架。容器初始化时加载控制

层、业务层、数据层的JavaBean对象，层与层之间通

过依赖注入的方式访问相应对象，可通过配置方式

或注解方式实现各层之间类对象的实例化，实现类

对象的属性和方法调用。同时Spring框架提供切

面编程应用，实现业务逻辑处理与数据库操作事务

处理的分离，降低系统的开发难度。