高级系统架构设计师下午试题(Ⅰ)模拟12

试题一

阅读以下关于软件复用技术应用的叙述，根据要求回答问题。

[说明]

随着政府职能的转型，XX行业职能主管部门(国家相关部委、各省、市的相关厅、局、委等)机构日趋精简，但随着国家现代化的发展，业务量反而越来越大。为保证本行业管理工作的质量和效率，实现企业资质审批管理信息化、网络化和电子化，提高了工作效率和质量，规范了业务处理流程，提高管理工作的广度、深度、力度和速度，更有效地为相关企业和业主提供服务，该行业主管部门委托FT软件开发公司开发一个XX行业管理信息系统。该系统由公共信息管理模块、系统管理维护模块、资质管理模块、企业信息管理模块和系统扩展接口模块等5大功能模块组成。

在该项目架构分析会上，FT公司高层领导提出，为了使那些公共功能模块比较容易地被其他电子政务项目所复用，应充分利用领域分析和软件复用的知识，采用基于可复用的软件开发方式，在这些公共模块的实现中保持高度的独立性，即在实现的具体细节上与XX行业国家部委(或者其他的政府机构)无关。

1、[问题1]

特定领域软件架构(Domain Specific Software Architecture，DSSA)是一种有效实现特定领域软件重用的手段。结合你的系统架构设计经验，请用300字以内的文字简要说明基于DSSA的软件设计开发主要包含哪些阶段以及每个阶段的目标。

2、[问题2]

结合你的系统架构设计经验，请用300字以内的文字简要说明该项目5个功能模块在软件复用方面的基本架构思路。

3、[问题3]

软件复用包括两个相关过程：可复用软件(构件)的开发；基于可复用软件(构件)的应用系统构造(集成和组装)。软件构件技术是软件复用的核心技术。结合你的系统架构设计经验，请用200字以内的文字简要说明可复用构件应具备哪些属性，并给予简要的解释。

试题二

阅读以下关于某平安城市工程视频监控系统架构的叙述，根据要求回答问题。

[说明]

某城市为满足治安管理、城市管理、交通管理和应急指挥等需求，决定在城市的所有进出路口、客货运场所、主要道路路口、重要公共场所、商业密集区域，以及治安案件高发区等地进行视频监控，并通过网络建立完善的社会治安视频监控系统，即实施"平安城市工程"，实现视频监控信息资源的整合与共享。

平安城市工程的网络接入如图1所示。所有监控点的摄像机通过运营商提供的线路接入平安城市网络，公安局的监控体系有三级结构，分别为市局、分局和派出所监控中心。运营商传输网络负责所有视频监控信号的传输、存储和转发，由传输设备、网络设备和存储设备等构成。

平安城市工程规范中规定，实时调阅视频流从采集至播放的时间延迟不得大于1s。

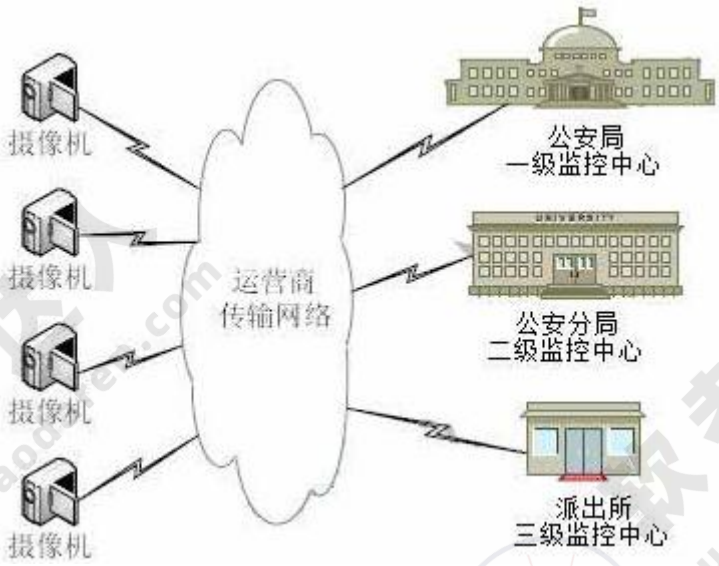


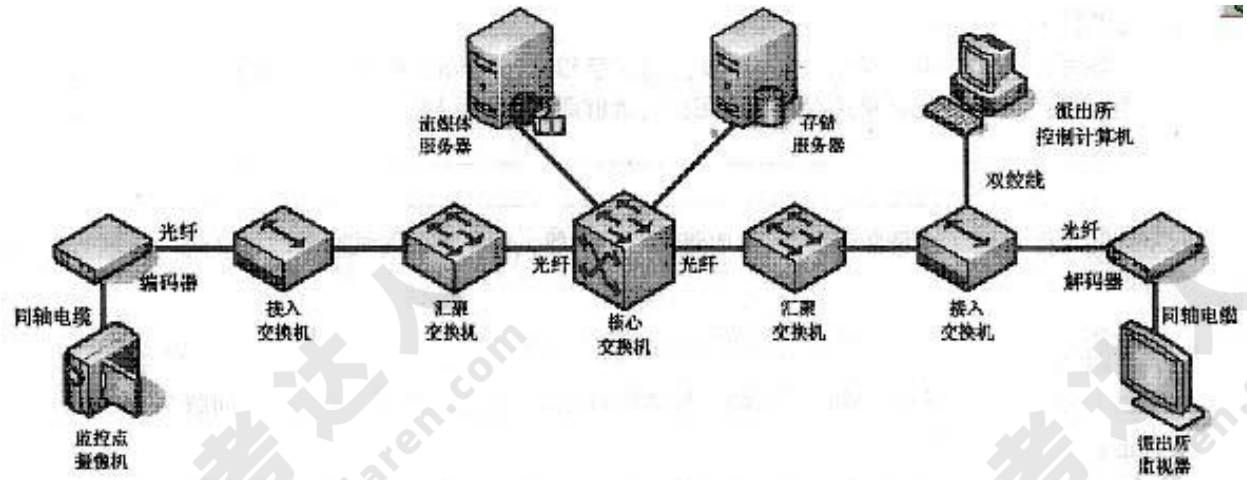
图1

4、[问题1]

图2为某派出所与其管辖的一个监控点之间的设备连接图，表1为图中各设备产生的延迟情况。其中，核心交换机3号插槽上安装8端口GBIC千兆以太网模块WS-X6408A(8 port GIGABITETHERNET)，用于与各行政区汇聚交换机互连；核心交换机4号插槽上安装16端口GBIC千兆以太网模块WS-6516-GBIC(16 port GIGABIT ETHERNET)，负责连接平安城市工程中所有的流媒体服务器、存储服务器等设备，端口1和端口2连接两台流媒体服务器，端口3和端口4连接两台存储服务器。

请计算该派出所与其管辖的一个监控点的实时视频调阅延迟，并指出是否符合平安城市工程规范。若符合规范，请简要说明理由；若不符合规范，在不改变编解码器和流媒体服务器产品的情况下，请给出可能的优化方案。

图2



设备连接图

表1各设备延迟情况

序号	设备	延迟原因	延迟时间(ms)
1	编码器	视频信号模数转换延时	350
2	接入交换机	数据帧转发延时	30
3	汇聚交换机	数据帧转发延时	30
4	核心交换机	数据帧模块间转发延时	10
5	核心交换机	数据帧模块内端口间转发延时	5
6	流媒体服务器	视频流处理及转发延时	80
7	存储服务器	视频存储延时	250

8	存储服务器	视频调阅转发延时	100
9	解码器	视频信号数模转换延时	350
10	各线路	信号传输延时	0(忽略不计)

5、[问题2]

该平安城市工程视频监控系统可以提供实时监控、存储和随时调看CIF格式(352×288)和D1格式(720×576)分辨率的图像，支持：MPEG-2、MPEG-4和H.264等编码格式。

(1)该城市某行政区内预计共有监控点600个，如果保存的是CIF格式的图像，码流为512kbps，请计算每小时保存该行政区内全部监控点视频流需要多大的存储空间(B或GB.。(请将计算结果保留小数点之后3位数)。

如果保存的是D1格式的图像，码流为2048kbps，请计算每小时保存该行政区内全部监控点视频流需要多大的存储空间(B或GB.。

(2)全部监控视频流信息保存在IP SAN设备S2600中，S2600控制框(双控，220V交流，4GB内存，8xGE iSCSI主机接口，磁盘数量12个/框，最大支持附加7个磁盘扩展框)。假设在本项目中采用SATA1.5TB 7.2KRPM硬盘，在IP SAN配置的RAID组级别为RAID10。

若该视频监控系统实施时，图像格式采用了CIF，码流为512 kbps，请计算保存该行政区内全部监控点30天视频流需要的存储空间(B、GB或TB.，并计算出保存30天视频流至少需要的硬盘数，以及至少需要配置的$2600控制框数量。

6、[问题3]

该平安城市工程视频监控系统的一些关键的应用系统，采用双机冗余热备的方式进行保护。

请用200字以内的文字，说明双机冗余热备方式主要解决的是系统运行中的哪些问题，以及在选择双机冗余热备产品时通常需要考虑哪些问题?

试题三

阅读以下关于设计模式应用的叙述，根据要求回答问题。

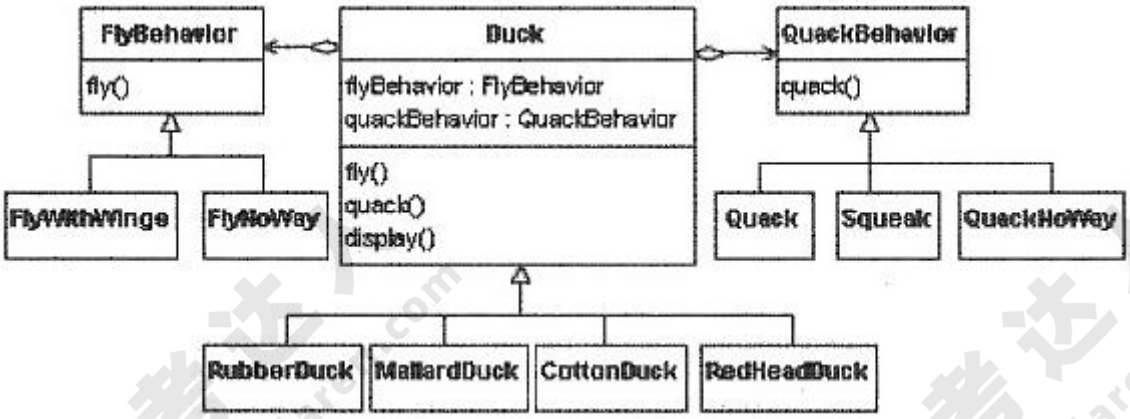
[说明]

某软件公司承接了一项面向儿童的模拟游戏软件的开发任务，该游戏软件主要模拟现实世界中各种鸭子的发声特征、飞行特征和外观特征。游戏软件需要模拟的鸭子种类及其特征如表所示

鸭子种类及其特征			
鸭子种类	发声特征	飞行特征	外观特征
灰鸭(MallardDuck)	发出"嘎嘎"声(Quack)	用翅膀飞行(FlyWithWings)	灰色羽毛
红头鸭(RedHeadDuck)	发出"嘎嘎"声(Quack)	用翅膀飞行(FlyWithWings)	灰色羽毛、头部红色
棉花鸭(CottonDuck)	不发声(QuackNoWay)	不能飞行(FlyNoWay)	白色
橡皮鸭(RubberDuck)	发出橡皮与空气摩擦的声音(Squeak)	不能飞行(FlyNoWay)	黑白橡皮颜色

为支持将来能够模拟更多种类鸭子的特征，该公司架构师采用某种设计模式设计的类图如图1所示。在图1中，类Duck描述了抽象的鸭子，方法fly7、quack7和display7分别表示不同种类的鸭子都具有飞行特征、发声特征和外观特征；类FlyBehavior与QuackBehavior分别用于表示抽象的飞行行为与发声行为。

图1



7、[问题1]

请用350字以内的文字指出该公司架构师所采用的设计模式的具体名称、设计意图及其优缺点。

8、[问题2]

请用400字以内的文字指出该公司架构师所采用的设计模式的适用性，以及图1中需要考虑哪些实现问题?

9、[问题3]

设计模式在力度和抽象层次上各不相同。按设计模式的目的划分，可分为创建型、结构型和行为型3种模式；按设计模式的范围划分，可分为类设计模式和对象设计模式两种。请将下列A～J标记的设计模式填入到下表中的(1)～(5)空缺处。(请用A～J答题)

A. Abstract Factory模式　　B. Adapter模式　　C. Chain of Responsibility模式
D. Decorator模式　　E. Factory Method模式　　F. Flyweight模式
G. Interpreter模式　　H. Iterator模式　　I. Template Method模式
J. Visitor模式

设计模式空间				
		目 的		
		创建型	结构型	行为型
范 围	类	(1)	—	(2)
	对 象	(3)	(4)	(5)

试题四

阅读以下关于体系结构设计的叙述，根据要求回答问题。

[说明]

某大中型电子商务公司的主要业务是在线购物，包括书籍、服装、家电和日用品等。随着公司业务规模不断增大，公司决策层决定重新设计并实现其网上交易系统。PH软件公司承担了该项目软件开发任务，负责系统开发的杜工和赵工分别给出了两种不同的设计方案，分别如图1和图2所示。

图1

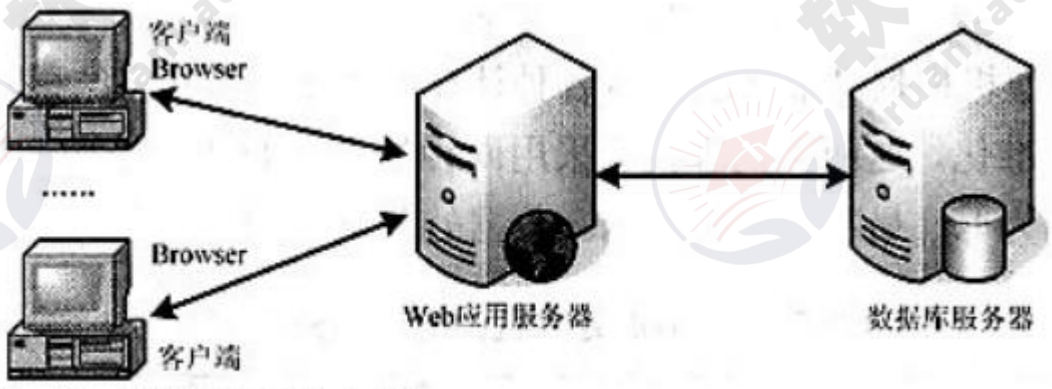
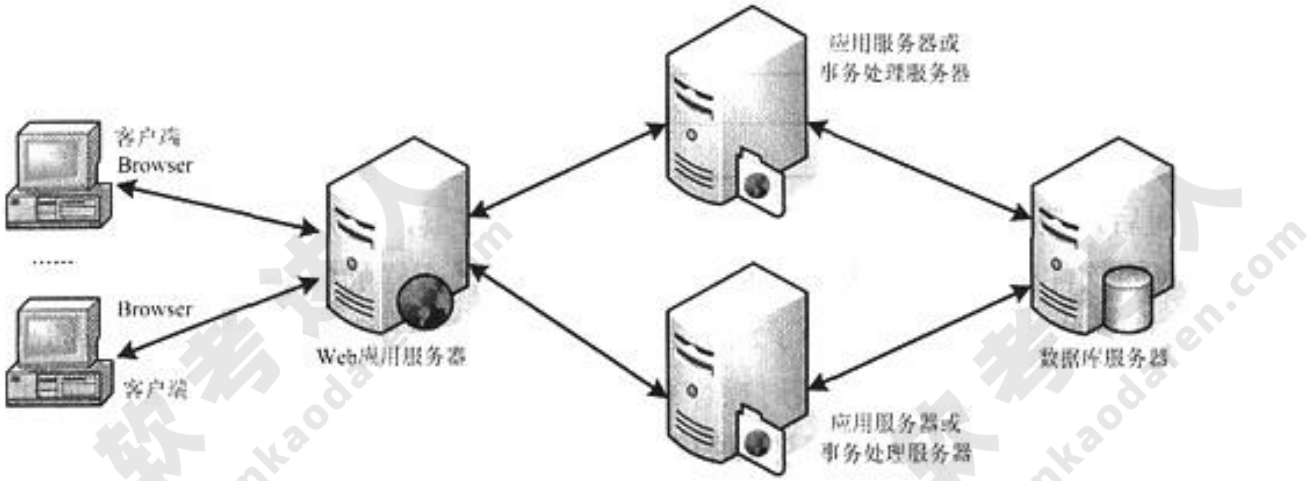


图1 杜工设计的体系结构示意图

图2



公司的架构师和开发者针对这两种设计方案，从服务器负载情况、业务逻辑的分离性、系统可靠性，以及实现简单性等方面进行讨论与评估，综合考虑最终采用了赵工给出的方案。

10、[问题1]

结合你的系统架构设计经验，请分析比较杜工、赵工两种方案的优点和不足，将下表中(1)～(6)空缺处的内容填写完整。

架构方案对比表

体系结构方案 / 评价因素	杜工建议的体系结构方案	赵工建议的体系结构方案
服务器负载	(1)	(2)
业务逻辑的分离性	(3)	(4)
系统的可靠性	采用单台 Web 服务器，整个系统的可靠性较差	(5)
实现简单性	(6)	需要将脚本语言与面向对象编程语言相结合，相对复杂

11、[问题2]

如何架构高性能Web应用系统是PH公司项目组面临的另一个问题。结合你的系统架构设计经验，请用200字以内的文字列举两个主要影响着Web应用系统服务端执行效率的技术因素，并针对每个因素提出相应的解决方案以提高系统性能。

12、[问题3]

REST(REpresentational State Transfer)是从几种基于网络的架构风格衍生出来的一种混合架构风格。采用这种方法设计的Web应用系统能够结合REST风格和面向服务思想的优点。结合你的系统架构设计经验，请用300字以内的文字简要说明与传统的Web服务相比，采用REST服务构建的Web应用具有哪些优势和不足。

试题五

阅读以下关于Java企业级应用系统开发架构的叙述，根据要求回答问题。

[说明]

某软件公司承担了某中小型企业应用软件开发任务，进度要求紧迫。为了按时完成任务，选择合适的企业应用系统开发架构非常重要。项目组在进行方案论证时，项目组成员提出了两种开发思路。

13刘工建议采用J2EE 5.0和EJB 3.0进行开发。理由是J2EE定义了标准的应用开发体系结构和部署环境，EJB是J2EE的基础和核心。J2EE 5.0主要目标是简化开发，相比EJB 2.1，EJB 3.0具有很多改进和提高。

14杜工建议采用Struts、Spring和Hibenate轻量级开源框架相结合的方式。理由是随着Java开源项目阵营的发展壮大，一些基于POJOs(Plan Old Java Objects)的开源框架被广泛地引入到Java企业应用开发中来，与重量级的EJB框架相比，这些轻量级的框架有很多优点。

项目组仔细比较分析了两种方案的特点、优点和不足之处。认为杜工和刘工的建议都合理，但

是从结合当前项目实际情况出发，最后决定采用杜工的建议。

13、[问题1]

Java企业级应用框架一般被划分为3个层次，请用150字以内的文字说明都有哪3个层次?功能分别是什么?

14、[问题2]

请用200字以内的文字叙述Struts、Spring和Hibernate开源框架特点和结合方式。

15、[问题3]

请用200字以内的文字说明基于Struts、Spring和Hibernate的轻量级框架与基于EJB的重量级框架解决问题的侧重点有什么不同?

答案:

试题一

1、领域分析是分析和研究某个应用领域特性的活动，它是识别、收集、组织和描述一个领域相关信息，发现和记录领域中的共性和差异的过程，是系统化、形式化、有效复用的关键。通过领域分析，类似系统的公共特性将被提取，使用于该领域所有公共的、基本的对象，操作也将被标志出来，并且通过定义模型来描述他们之间的关系。领域分析的本质是以复用为目的，对具有若干共同特性的一群应用系统进行分析，对共同的部分开发出一系列公用的组件，对不同的部分进行参数化。

按照Will Tracz的说法，特定领域软件架构(DSSA)就是一个特定的问题领域中由领域模型、参考需求和参考架构等组成的开发基础架构，其目标就是支持一个特定领域中多个应用的生成。DSSA的基本活动包括领域分析、领域设计和领域实现。领域分析的主要目的是获得领域模型，领域模型描述领域中系统之间共同的需求，即领域需求；领域设计的主要目标是获得DSSA，DSSA描述领域模型中表示需求的解决方案；领域实现的主要目标是依据领域模型和DSSA开发、组织可重用信息。

在最高的级别上，DSSA方法共有5个阶段，每个阶段可以进一步划分为一些步骤或子阶段，每个阶段包括一组需要回答的问题、一组需要的输入，以下一组将产生的输出和验证标准。该方法的领域工程过程是并发的、递归的和反复的，或者可以说，它是螺旋型的，完成该过程可能需要对每个阶段经历几遍，每次增加更多的细节。该领域工程过程的5个阶段如下。

(1)定义领域范围：重点是确定领域中包含哪些元素及领域工程过程到何时结束。这一阶段的一个主要输出是领域中的应用需要满足的一系列用户的需求。

(2)定义领域特定的元素：目标是制订领域字典和领域术语的同义词词典。在领域工程过程的前一个阶段产生的高层次块图中增加更多的细节，特别是识别领域中各种应用间的共同性和差异性。

(3)定义领域特定的设计和实现需求约束：目标是描述空间中的特性，不仅要识别约束，并且要记录约束对设计和实现决定造成的后果，还要记录对处理这些问题时产生的所有问题的讨论。

(4)定义领域模型和构架：目标是产生DSSA，并说明构成它的模块或构件的语法和语义。

(5)产生和搜集可复用的产品：目标是为DSSA增加构件，使得它可以被用来产生问题域中的新应用。

从软件实践上看，面对特定领域的系统开发，迫切需要一种好的开发方法来有效地支持软件重用性，而DSSA方法已经在实践中被证明是有效的，能显著加大重用的粒度，从而缩短整体开发周期，在提高产品品质的同时降低开发费用。

2、在电子政务的建设过程中，应充分利用领域分析和软件复用的知识，采用基于可复用的软件开发方式，注重采用领域分析的方法，建立领域模型。例如，在领域分析时使用国际上通用的统一建模语言(UML)作为建模工具。在电子政务的建设过程中，应建立系统工程，建立行业范围内的一些标准，以利于确定复用的方案，使开发的系统可靠性高、成本低。

根据题干关键信息，并结合从事系统架构设计的历史经验，大致可以分析该系统5大功能模块中，公共信息管理模块是Internet网络上的一个应用系统，这部分主要包括系统网站相关功能，包括信息发布和浏览、资质等级标准查询、企业资质查询、网上调查、意见箱管理、问题解答，以及下载文

档管理等。系统管理维护模块主要是系统管理员用户用来设置或设定系统的参数，管理和维护系统最基本的数据。主要功能包括用户及权限管理、工作流管理、企业管理、信息发布管理、系统维护、文档管理和基础数据管理等。对于这两个功能模块应该在具体的配置和使用上通过配置文件(或数据文件)把建设方的相关信息加入进来。这样，这些公共功能模块就很容易的被其他电子政务项目所复用。

根据行业经验，资质管理模块通常包括企业资质的网上申请、审批、年审、动态管理、变更、备案、资质证书管理和相关资质申请审批工作处理情况的查询及资质标准和等级的管理和分析统计等。企业信息管理模块主要是对具体企业信息的维护，包括企业相关信息的输入、输出、修改、删除、查询和打印等功能。其中，人员管理还包括人员的调离、年龄控制等。系统扩展接口模块用于本系统与其他信息系统的数据交换，保证了建设方的管理系统与其他行业部门已有或待建信息化系统的无缝链接等。对于这些与建设方密切相关的功能模块，即使在实现上无法完全和建设方相关信息分开，也要尽力降低和建设方的耦合度，以便使得今后复用时的代价最小化。

在具体实现上，应该充分采用在市场上广泛使用的技术与产品。例如，Web服务器、J2EE应用服务器、电子邮件系统、数据库系统、信息门户和系统安全解决方案等，建议使用已有的软件产品而不是自行开发。同时采用CMM模型或者CASE工具等指导所承接的软件系统的开发，以便高质量地完成项目的建设工作，使得所建成的软件系统具有良好的可复用性。

3、构件(component)是指应用系统中可以明确辨识的构成成分。它是软件系统可替换的、物理的组成部分，它封装了实现体(实现某个职能)，并提供了一组接口的实现方法。而可复用构件(Reusablecomponent)是指具有相对独立的功能和可复用价值的构件。软件构件技术是软件复用的核心技术。它是基于面向对象的，以即插即用型软构件概念为中心，通过构件组合来建立应用的体系。其主要研究内容包括构件获取、构件模型、构件描述语言、构件分类与检索、构件复合组装和构件标准化。

可复用构件应具备以下属性。

(1)有用性(Usefulness)，构件必须提供有用的功能。

(2)可用性(Usability)，构件必须易于理解和使用。

(3)质量(Quality)，构件及其变形必须能正确工作。

(4)适应性(Adaptability)，构件应该易于通过参数化等方式在不同语境中进行配置。

(5)可移植性(Portability)，构件应能在不同的硬件运行平台和软件环境中工作。

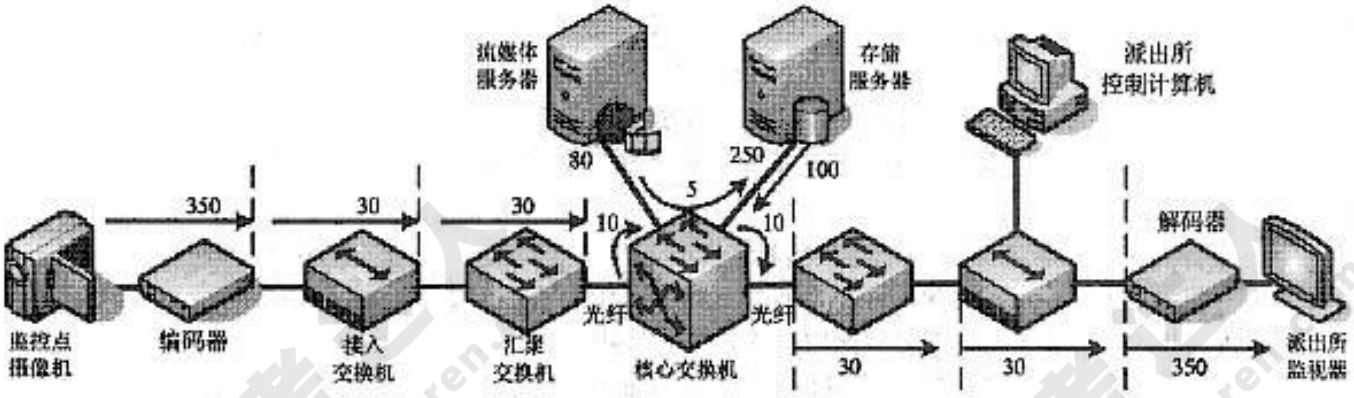
(6)可变性(Variability)，构件除了向复用者提供一些公共"特性"外，还要提供可变的"特性"。针对不同的应用系统，只需对其可变部分进行适当的调节，复用者要根据复用的具体需要，改造构件的可变"特性"，即进行"客户化"工作等。

试题二

4、依题意，基于图2所示的视频监控设备连接图和表1所示的各种设备产生的延迟参数可以做出如下分析，当某派出所的控制计算机发出调阅其管辖的某个监控点实时视频的指令后，该调阅指令将被流媒体服务器接收并进行相关处理，流媒体服务器将向所指定监控点的摄像机发出采集视频流的操作命令；摄像机所采集的模拟视频流信号经编码器进行模/数(A/D)转换并封装成相应的以太数据帧送入视频监控网络的接入层交换机(约延迟350ms)，之后该视频流数据帧将经过接入层交换机和汇聚层交换机的转发(分别延迟了30ms和30ms)，将到达核心交换机3号插槽光纤接口模块；由于流媒体服务器是连接在核心交换机4号插槽的端口1和端口2，因此该视频流数据帧从3号插槽光纤接口模块转发到4号插槽千兆以太接口模块时，将有10ms延迟时间；该视频流数据帧经流媒体服务器进行相关处理后(约延迟80ms)，将由4号插槽的端口1(或端口2)转发给连接在端口3(或端口4)的存储服务器进行数据存储操作，这一过程存在有数据帧模块内端口间5ms转发延时和250ms视频存储延迟时间；当从存储服务器调阅相应的视频流数据时(约延迟100ms)，该视频流数据帧将从4号插槽的端口3(或端口4)转发到3号插槽光纤接口模块(约延迟10ms)，然后分别经汇聚层交换机和接入层交换机转发后(分别延迟了30ms和30ms)，送至解码器进行数据帧的拆封并通过数/模(D/A)转换形成监视器可接收的模拟视频流信号(约延迟350ms)。以上分析过程中，各设备延迟情况的示意图如图3所示。因此某派出所与其管辖的一个监控点的实时视频调阅延迟

t=350+30+30+10+80+5+250+100+10+30+30+350=1275ms=1.275s。

图3



各设备延迟情况分析图

由于平安城市工程规范中规定，实时调阅视频流从采集至播放的时间延迟不得大于1s，由于1.275s＞1.0s，因此图2所产生的实时视频调阅延迟不符合平安城市工程规范。若不能改变编/解码器和流媒体服务器产品，即保留实时视频调阅延迟t中350ms、80ms和350ms 3个参数的情况下，则可以从优化其他转发延时参数入手进行分析，在不考虑经费限制的情况下，可能的优化方案如下。

(1)改进流媒体服务器视频流处理软件的工作模式，即当视频流数据帧经流媒体服务器进行相关处理后，采取并发(或并行)或组播的工作模式将处理后的数据帧一路送至存储服务器进行数据存储操作，同时另一路数据帧直接送至核心交换机的相应输出端口。采取这一优化措施，可能节省约350ms转发延时(即节省250ms视频存储延迟时间和100ms视频调阅转发延时)。

(2)将核心交换机4号插槽的GBIC千兆以太网模块升级(或更换)成至少拥有4个千兆光纤接口和16个千兆以太端口的模块，由这4个千兆光纤接口分别连接图2中的两台汇聚交换机。若4号插槽中有空余的千兆光纤接口，则可将两台汇聚交换机上连光纤由原来插接在3号插槽变更为插接在4号插槽。采取这一优化措施，将节省10ms转发延时，即节省图2中两次的核心交换机数据帧模块间的10ms转发延时，但同时也引入了两次的数据帧模块内端口间的5ms转发延时。

(3)在光纤传输距离允许且汇聚交换机有多余光纤接口的情况下，可将编码器和解码器直接连接至各自的汇聚交换机。采取这一优化措施，若编/解码器均直接上连至汇聚交换机，则将节省60ms转发延时；若只将一侧的编码器(或解码器)直接上连至汇聚交换机，则将节省30ms转发延时。

(4)优化存储服务器对视频流存储和调阅转发的处理机制(例如，采用RAID10或RAID0技术，把连续的数据分散到多个磁盘上存取)，或者升级存储服务器的相关硬件(如CPU、内存和硬盘等)，再或者更换成具有更高数据存取性能的存储服务器产品，从而减少视频流存储延时和调阅转发延时。

5、(1)分辨率是数字监控产品中一项重要的技术指标，它在很大程度上决定了产品的性能(清晰度、存储量和带宽)和价格。目前监控行业中主要使用QCIF(176×144)、CIF(352×288)、HALF D1(704×288)和D1(704×576)等几种分辨率，CIF是主流的录像分辨率格式。

通常，监控图像硬盘存储容量的计算主要取决于"码流"这一参数，与图像格式的分辨率大小没有直接关系。CIF、QCIF、DCIF和D1等都有对应的码流范围，如果只是一味地将码流参数调高，图像质量也不会有明显的变化。依题意，若该行政区内视频监控系统保存的是CIF格式图像，码流为512 kbps，则每小时保存行政区内600个监控点视频流需要138.24×10⁹B(128.746 GB)的存储空间。具体计算过程如下。

$$M_1 = 512\times10^3(\text{bps})\times3600(\text{s})\times600(\text{个}) = 1105.92\times10^9(\text{bit}) \div 8 = 138.24\times10^9\,\text{Byte}$$

$$= \frac{138.24\times10^9\,\text{Byte}}{2^{10}\times2^{10}\times2^{10}} \approx 128.746\,\text{GB}$$

若监控系统保存的是D1格式的图像，码流为2048kbps，该码流为CIF格式图像码流512kbps的4倍，则每小时保存行政区内600个监控点视频流需要552.960×10⁹B(即138.24×10⁹×4 B)或514.984GB(即128.746×4 GB)的存储空间。

(2)若该视频监控系统实施时，图像格式采用CIF格式，码流为512kbps，则保存行政区内600个监控点30天视频流需要99.5328×10¹²B(或92697.144 GB，或90.525 TB)的存储空间。具体计算过程如下。

$$M_2 = 138.24\times10^9\times24\times30\text{Byte} = 99532.8\times10^9\text{Byte} = \frac{99532.8\times10^9\text{Byte}}{2^{10}\times2^{10}\times2^{10}\times2^{10}} \approx 90.525\ \text{TB}$$

廉价磁盘冗余阵列(RAID)是利用一台磁盘阵列控制器来管理和控制一组磁盘驱动器，组成一个高度可靠的、快速的大容量磁盘系统。RAID级别是指磁盘阵列中硬盘的组合方式，不同级别的RAID为用户提供的磁盘阵列在性能上和安全性的表现上也有不同。RAID0也称为Stripe(条带化)，它把连续的数据分散到多个磁盘上存取，代表了所有RAID级别中最高的存储性能。其磁盘利用率为100%，但它不提供数据冗余。RAID1具有磁盘镜像和磁盘双工功能，可利用并行读/写特性，将数据块同时写入主盘和镜像盘，故比传统的镜像盘速度快，但其磁盘利用率只有50%。

RAID10是建立在RAID0和RAID1基础上的高可靠性与高性能的组合，即利用了RAID0极高的读写效率和RAID1较高的数据保护和恢复能力。但RAID10的磁盘利用率只有50%。

依题意，该行政区内视频监控系统全部监控视频流信息保存在IP SAN设备S2600中，采用存储容量为1.5 TB的SATA硬盘，且采用RAID10配置(磁盘利用率只有50%)，因此保存30天视频流至少需要的硬盘数为121块。具体计算过程如下。

$$N_1 = \frac{90.525(TB)}{1.5(TB)\times50\%} = 120.7$$

，将计算结果向上取整数为121。

由于IP SAN设备S2600中，每个控制框仅支持12个磁盘，因此至少需要配置的S2600控制框数为11个(即121/12≈10.083，将计算结果向上取整数为11)。

6、容错技术是指在一定程度上容忍故障的技术，也称为故障掩盖技术(Fault Masking)。容错主要依靠冗余设计来实现，以增加资源换取可靠性。由于资源的不同，冗余技术分为硬件冗余、软件冗余、时间冗余和信息冗余。也可以是元器件级、部件级和系统级的冗余设计。

采用双机冗余热备方式，当本地某个系统发生故障时，系统能够自动快速地切换到正常的系统，通过本地故障恢复确保系统持续提供服务。在这种方案中，需采用双机热备份软件，用于提高服务器的可靠性。可选用离线数据备份及灾难恢复软件，保证数据的可靠性。还需要用到磁带机、磁带库和磁盘阵列等硬件设备。

双机热备份方式的两台服务器都处于热机状态，如果一台服务器坏了，另一台服务器可以将所有的业务接管过来。它有两种工作方式：Online方式—两台服务器都在工作，分别担负不同的任务，均衡负载。缺点是成本大，管理难；Standby方式—备份机不工作，只是监测作业机的工作状况。缺点是服务器之间的切换时间较长。

目前有许多不同厂家提供双机冗余热备的产品。在选择双机冗余热备产品时，通常需要考虑的因素有：①双机热备产品适用的规模；②支持的操作系统；③支持的数据库系统；④对正常业务系统的性能影响；⑤提供的图形用户界面(GUI)管理工具功能易用性；⑥能够完全实现多应用多级切换(应用级切换)，适用于多种应用并存的系统，某一应用的切换可以不对其他应用产生影响；⑦集中管理配置能力；⑧远程监控和管理能力；⑨切换速度；⑩磁盘管理方面的功能等。

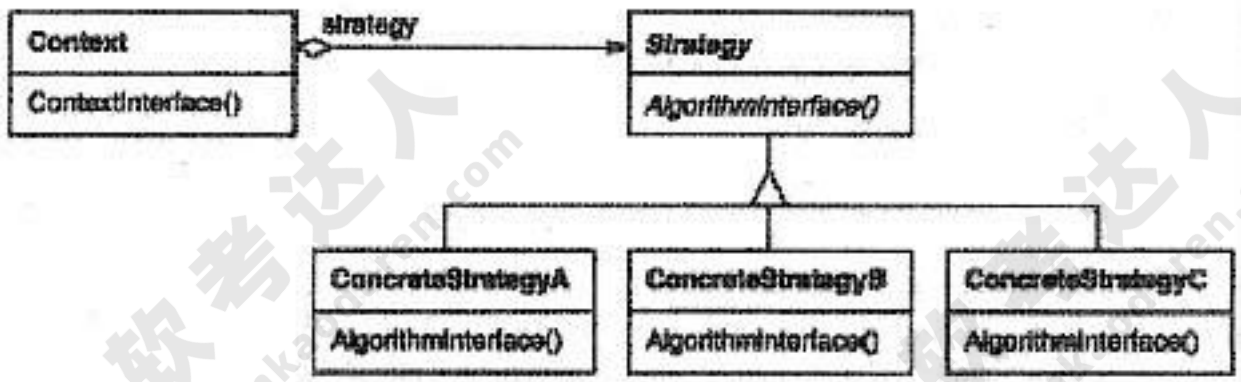
试题三

7、依题意，在图1中，Duck为抽象类，描述了抽象的鸭子，方法fly()、quack()和display()分别表示不同种类的鸭子都具有飞行特征、发声特征和外观特征；而类RubberDuck、MallardDuck、CottonDuck和RedHeadDuck分别描述具体的鸭子种类；类Fly Behavior与QuackBehavior为抽象类，分别用于表示抽象的飞行行为与发声行为；类Fly NoWay与Fly WithWings分别描述不能飞行的行为和用翅膀飞行的行为；类Quack、Squeak与QuackNoWay分别描述发出“嘎嘎”声的行为、发出橡皮与空气摩擦声的行为和不发声的行为。鉴于不同的鸭子种类只是在行为方面有所区别，且为支持将来能够模拟更多种类鸭子的特征，该公司架构师最有可能采用策略(Strategy)设计模式来设计如图1所示的模拟鸭子游戏软件。

Strategy模式定义了一组能够用来表示可能行为集合的类。这些行为可以在应用程序中使用，来修改应用程序功能。Strategy(策略)模式的设计意图是，定义一系列的算法，把它们一个个封装起来，并且使它们可相互替换，使得算法可独立于使用它的客户而变化。具体而言，该模式是一种定义一系列算法的方法，从概念上看，所有这些算法完成的都是相同的工作，只是实现不同，它可以以

相同的方式调用所有的算法，减少了各种算法类与使用算法类之间的耦合。Strategy模式的一般结构如图2所示。

图2



Strategy 模式的一般结构图

Strategy模式具有以下一些优点和缺点。

(1)另一种子类化方法。Strategy类层次为Context(上下文)定义了一系列的可供重用的算法或行为。继承有助于析取出这些算法中的公共功能。可以直接生成一个Context类的子类，从而给它以不同的行为。但这会将行为强制编制到Context中，而将算法的实现与Context的实现混合起来，从而使Context难以理解、难以维护和难以扩展，而且还不能动态地改变算法。最后得到一堆相关的类，它们之间的唯一差别是它们所使用的算法或行为。将算法封装在独立的Strategy类中使得架构师可以独立于Context而改变它，使它易于切换、理解和扩展。

(2)在类自身中定义了每一个行为，从而减少了一些条件语句；Strategy模式提供了用条件语句选择所需行为以外的另一种选择。当不同的行为堆砌在一个类中时，很难避免使用条件语句来选择合适的行为。将行为封装在一个个独立的Strategy类中消除了这些条件语句。

(3)更容易扩展模型，即在不对应用程序进行代码修改的情况下，使该模式具有新的行为。

(4)客户必须了解不同的Strategy。该模式有一个潜在的缺点，就是一个客户要选择一个合适的Strategy就必须知道这些Strategy到底有何不同。此时可能不得不向客户暴露具体的实现问题。因此仅当这些不同行为变成与客户相关的行为时，才需要使用Strategy模式。

(5)Strategy和Context之间的通信开销。无论各个ConcreteStrategy(具体策略)实现的算法是简单还是复杂，它们都共享Strategy定义的接口。因此很可能某些ConcreteStrategy不会都用到所有通过这个接口传递给它们的信息；简单的ConcreteStrategy可能不使用其中的任何信息。这就意味着有时Context．会创建和初始化一些永远不会用到的参数。如果存在这样问题，那么将需要在Strategy和Context之间进行更加紧密的耦合。

(6)增加了对象的数目。Strategy增加了一个应用中的对象的数目。有时可以将Strategy实现为可供各Context共享的无状态的对象来减少这一开销。任何其余的状态都由Context维护。Context在每一次对Strategy对象的请求中都将这个状态传递过去。共享的Strategy不应在各次调用之间维护状态。

8、在以下情况中，应该使用Strategy模式。

(1)许多相关类只是在行为方面有所区别。"策略"提供了一种用多个行为中的一个行为来配置一个类的方法。

(2)需要使用一个算法的不同变体。例如，定义了一些反映不同的空间或时间权衡的算法，当这些变体实现为一个算法的类层次时，可以使用策略模式。

(3)算法使用客户端未知的数据，可使用策略模式以避免暴露复杂的、与算法相关的数据结构。

(4)一个类定义了多种行为，并且这些行为在这个类的操作中以多个条件语句的形式出现。将相关的条件分支移入它们各自的Strategy类中以代替这些条件语句。

依题意，在图1中，需要考虑以下一些Strategy模式实现问题。

(1)定义类Duck和类FlyBehavior(或类QuackBehavior)接口。这些接口必须使得类FlyWithwings、类FlyNoWay、类Quack、类Squeak和类QuackNoWay等能够有效地访问它所需

要的类Duck中的任何数据，反之亦然。一种解决办法是让类Duck将数据放在参数中传递给类FlyBehavior(或类QuackBehavior)操作，也就是说，将数据发送给类FlyBehavior(或类QuackBehavior)。这使得类FlyBehavior(或类QuackBehavior)和类Duck解耦。但从另一个角度考虑，类Duck也可能发送一些类FlyBehavior(或类QuackBehavior)不需要的数据。

另一种解决办法是让类Duck将自身作为一个参数传递给类FlyBehavior(或类QuackBehavior)，该类FlyBehavior(或类QuackBehavior)再显式地向类Duck请求数据；或者类FlyBehavior(或类QuackBehavior)可以存储对它的类Duck的一个引用，这样根本不再需要传递任何东西。这两种情况下，类FlyBehavior(或类QuackBehavior)都可以请求到它所需要的数据。但要求类Duck必须对它的数据定义一个更为精细的接口，这将使得类FlyBehavior(或类QuackBehavior)和类Duck更加紧密地耦合在一起。

(2)将类FlyBehavior(或类QuackBehavior)作为模板参数。例如，在C++中，可利用模板机制用一个Strategy来配置一个类。然而这种技术仅当下面条件满足时才可以使用：可以在编译时选择Strategy；它无须在运行时改变。在这种情况下，要被配置的类(如类Duck)被定义为以一个Strategy类作为一个参数的模板类。使用模板不再需要定义给类FlyBehavior(或类QuackBehavior)定义接口的抽象类。把类FlyBehavior(或类QuackBehavior)作为一个模板参数也使得可以将一个类FlyBehavior(或类QuackBehavior)和它的类Duck静态地绑定在一起，从而提高效率。

(3)尽量使类FlyBehavior(或类QuackBehavior)成为可选的对象。即使在不使用额外的FlyBehavior(或类OuackBehayior)对象的情况下，类Duck也还有意义，那么它还可以被简化。类Duck在访问类FlyBehavior(或类QuackBehavior)前先检查它是否存在，如果有，那么就使用它；如果没有，那么类Duck执行默认的行为。这种方法的好处是客户根本不需要处理FlyBehavior(或类QuackBehavior)对象，除非它们不喜欢默认的行为。

9、设计模式主要用于得到简洁灵活的系统设计，GoF的书中共有23个设计模式，这些模式可以按两个准则来分类：一是按设计模式的目的划分，可分为创建型、结构型和行为型3种模式；二是按设计模式的范围划分，即根据设计模式是作用于类还是作用于对象来划分，可分为类设计模式和对象设计模式，如表所示。

设计模式空间				
		目的		
		创建型	结构型	行为型
范围	类	Factory Method	Adapter(类)	Interpreter Template Method
	对象	Abstract Factory Builder Prototype Singleton	Adapter(对象) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

创建型模式是对对象实例化过程的抽象，它通过采用抽象类所定义的接口，封装了系统中对象如何创建及组合等信息。该模式允许在系统中创建对象，而不需要在代码中标识特定类的类型，这样用户就不需要编写大量复杂的代码来初始化对象。它是通过该类的子类来创建对象的。但是，这可能会限制在系统内创建对象的类型或数目。创建型模式主要有Factory Method(工厂方法)、Abstract Factory(抽象工厂)、Builder(构建器)、Prototype(原型)和Singleton(单独)等模式。

结构型模式主要用于如何组合已有的类和对象以获得更大的结构，一般借鉴封装、代理或继承等概念将一个或多个类或对象进行组合和封装，以提供统一的外部视图或新的功能。该模式允许在不

重写代码或自定义代码的情况下创建系统，从而使系统具有增强的重复使用性和应用性能。该模式控制了应用程序大部分之间的关系，将以不同的方式影响应用程序。结构型模式主要有Adapter(适配器)、Bridge(桥接)、Composite(组成)、Decorator(装饰)、Fagade(外观)、Flyweight(享元)和Proxy(代理)等。

行为型模式主要用于对象之间的职责及其提供的服务的分配，它不仅描述对象或类的模式，还描述它们之间的通信模式，特别是描述一组对等的对象怎样相互协作以完成其中任意一个对象都无法单独完成的任务。该模式可以影响一个系统的状态和行为流。通过优化状态和行为流转换及修改的方式，可以简化、优化并且提高应用程序的可维护性。行为型模式主要有Interpreter(解释器)、TemplateMethod(模板方法)、Chain of Responsibility(职责链)、Command(命令)、Iterator(迭代器)、Mediator(中介者)、Memento(备忘录)、Observer(观察者)、State(状态)、Strategy(策略)和Visitor(访问者)等。

试题四

10、本问题主要考查体系结构设计需要注意的问题。根据图1和图2给出的拓扑结构可知，图1给出的体系结构代表了一种典型的基于数据库服务器的动态内容发布结构，这种结构在服务器端设置了一台Web服务器和一台数据库服务器。在这种体系架构下，用户工作界面是通过WWW浏览器来实现，极少部分事务逻辑在前端(Browser)实现，但是主要事务逻辑在Web应用服务器端(Server)实现。Web服务器通过应用程序的支持(通常采用ASP、JSP等脚本语言，比较简单)，就可以给用户提供动态的信息服务，通过定制页面模板，添加到后台数据库中的信息可以及时发布给客户。这种将浏览器与Web服务器、应用服务器三者分开的架构风格，可跨平台操作，具有良好的开放性和可扩充性，便于系统升级、扩展和维护。但是，在这种架构下，Web服务器需要同时负责业务逻辑的处理和数据库访问，负载很较重；业务逻辑代码和其他程序代码全部在Web服务器中，不能做到业务逻辑代码与其他代码分离，且其中任何一个环节出错，都会导致Web服务器宕机，系统可靠性较差。

图2给出的是一种分布式的Web应用架构。与图1的架构风格相比，在Web服务器和后台数据库服务器之间增加了一层应用服务器。这是一种比较先进的架构模式，通过Web服务器处理用户请求，应用服务器处理业务逻辑与数据库访问，负载较为均衡。但是这种架构模式需要将脚本语言与面向对象编程语言相结合，相对复杂。由于增加了中间层应用服务器，因此可以将业务逻辑和数据库连接等放置到中间层上，不但减轻了Web应用服务器的负担，而且还可以做到业务逻辑代码与其他程序代码之间的分离。多个应用服务器的存在也可以提高访问性能，并增加系统的可靠性。

整理以上分析内容，可得到如下表所示的两种架构方案各自的优点和不足。

完整的架构方案对比表

体系结构 / 方案 / 评价因素	拟采用的体系结构方案	题干建议的体系结构方案
服务器负载	Web 服务器需要同时处理业务逻辑与数据库访问，负担较重	Web 服务器处理用户请求，应用服务器处理业务逻辑与数据库访问，负载较为均衡
业务逻辑的分离性	业务逻辑与数据库访问都位于 Web 服务器中，业务与逻辑没有分离	采用多个应用服务器专门进行业务逻辑处理，做到业务逻辑与其他代码分离
系统的可靠性	采用单台 Web 服务器，整个系统的可靠性较差	采用多台应用服务器，系统的可靠性较高
实现简单性	主要采用 JSP、ASP 等脚本语言实现系统，比较简单	需要将脚本语言与面向对象编程语言相结合，相对复杂

11、本问题主要考查Web应用系统的性能优化问题。性能是Web应用系统的一个重要质量属性。主要有如下一些重要的技术因素影响着Web应用系统服务端的执行效率。

(1)数据库的连接与销毁。可以采用数据池的方式缓存数据库链接，实现数据库链接复用，提高系统的数据访问效率。

(2)构件或中间件的加载与卸载。可以采用分布式对象池的方式缓存创建开销大的对象，实现对象复用，提高效率。

(3)线程的创建与销毁。可以采用线程池的方式缓存已经创建的线程，提高系统的反应速度。

12、REST(REpresentational State Transfer)是从几种基于网络的架构风格衍生出来的一种混合架构风格，它是目前因特网的核心架构风格。REST风格中的特点是客户端/服务器、无状态、

缓存、统一接口、分层系统和按需代码。REST组件通过以一种数据格式转移资源的表述进行通信，可以基于接收者的能力和期待的内容，以及资源的性质动态地选择不同的表述。

基于REST服务(RESTful Service)的Web应用系统设计任务主要包括识别并设计REST风格的服务和采用面向服务的思想进行REST服务集成。采用这种方法设计的Web应用系统能够结合REST风格和面向服务思想的优点，近年来受到了广泛的关注。

与传统的Web服务相比，REST服务主要有以下几点优势。

(1)REST服务基于W3C/IETF的标准与规范(包括HTTP、XML、URI和MIME等)，其实现技术简单且成熟。

(2)REST服务基于URI和超链接技术，不需要通过集中式的服务信息仓库即可发现服务资源。

(3)REST服务支持缓存，具有无状态的特性，这些使得REST服务能够支持大量客户端，构建的应用系统具有较强的伸缩性。

(4)REST服务基于轻量级的Web框架，仅仅需要基本的开发工具支持，构建过程简单且成本较低。

(5)REST服务的测试相对简单，采用浏览器即可完成服务功能测试。

与传统的Web服务相比，REST服务主要存在如下不足。

(1)REST服务倡导的REST风格与实际实现尚存在一定差距。例如高层REST服务倡导使用GET、PUT、POST和DELETE所有4个统一接口，在REST实现部分通常只能采用GET和POST接口，因为大多数的代理和防火墙会屏蔽其他接口；并且XHTML表单中只能使用GET和POST接口。

(2)REST服务要求所有的输入参数都必须在URI中传递，这样会产生对参数容量大小的限制(目前的大小是4KB)。如果超出该数量，会导致HTTP协议错误(错误代码414：Request—URI too long)。

(3)在URI中表达复杂类型的参数比较困难，且目前对uRj中的参数不存在一种公认的编组(marshalling)和解编(un-marshalling)方法。

试题五

13、这是一道要求读者掌握Java企业应用框架层次结构及其各层功能的简答题。本题所涉及的知识点如下。

(1)Java企业应用框架一般被划分为表现层、业务逻辑组件层和持久层等3个逻辑层次。

(2)其中，表现层用来建立应用系统的界面，对应视图(View)。该层集中于为从客户端发来的请求服务的对象及其行为，用于展现数据；负责View组件实现模式、组件在View显示粒度、页面跳转，以及事件触发等功能。例如，表现层采用JSF(Java Server Face)，JSF的开发流程的核心是事件驱动，组件和标签的封装程度非常高，很多典型应用已经不需要开发者去处理HTTP，整个过程是通过IoC(依赖注入)来实现的，即可以帮助对客户端请求进行先期及后期的处理等。

(3)业务逻辑组件层用来开发应用逻辑，对应控制器(Controller)。例如，业务逻辑组件层采用EJB3.0的Session Bean。EJB 3.0允许开发者使用耦合松散的组件来开发应用，实现一个EJB所有使用的类和接口都减少了。

该层集中于支持由表现层发起的(某些情况下也可能由持久层直接发起)业务数据的逻辑处理。例如，隐藏业务对象的复杂性，集中工作流的处理；分离表现层与持久层，并为服务提供外观和代理接口等。

(4)持久层是实现持久化存储，对应模型(Model)。例如，采用EJB 3.0实体Bean持久化模型，吸收了Hibernate的一些思想采用O/R MappinR模式。

该层集中于支持外部资源通信。例如，与数据库交互数据；抽象数据源，提供透明的数据访问；帮助进行EJB组件中的异步处理等。

14、这是一道要求读者掌握Struts，Spring和Hibenate轻量级开源框架特点和结合方式的简答题。本题所涉及的知识点如下。

(1)由于EJB容器能够很好的处理系统性能、事务机制、安全访问权限，以及分布式运算等问题，基于EJB框架进行开发能保证企业应用平滑发展，而不是发展到一种规模就重新更换一套软件系统，且可以保证开发人员将大部分精力集中在业务逻辑的开发上。采用EJB框架开发的企业应用具有必须继承或依赖EJB容器的特点。EJB充分考虑到了顶级大型项目的需求，使用它几乎能解决企业级应用

涉及的所有问题，相应的基于EJB框架也是一个功能复杂的重量级框架。

(2)在基于POJOs轻量级框架上开发的应用程序无须依赖于EJB容器可独立运行，对应于Java企业应用3个层次的轻量级框架技术分别都得到了一定的发展。Struts框架+Spring框架+Hiberhate框架实现了表现层、业务逻辑组件层和持久层的结合。

(3)目前比较流行的开源表现层框架主要有Struts和Tapestry。其中，Struts是基于模型—视图—控制器(MVC)模式的开源框架，主要用于企业应用中表示层的实现。借助于Struts，开发人员可以把主要精力集中在业务处理上，简化遵循MVC设计模式的Web应用开发工作，很好地实现代码重用，提高开发效率。

Straats框架包括：①模型(Model)。在Straats中模型是一个Action类，开发者通过其实现商业逻辑，同时用户请求通过控制器向Action的转发过程是基于由Struts-config. xml文件描述的配置信息的。②视图(View)。视图是由与控制器配合工作的一整套JSP定制标签库构成，利用它们可以快速建立应用系统的界面。③控制器(Contoller)，本质上是一个Servlet，将客户端请求转发到相应的Action类。④一堆用来做XML文件解析的工具包。

Struts应用框架由于出现时间早，因此使用相对广泛，很容易找到很多现成的开源功能标签以供使用，以及样例程序可供参考。但是它的组件在页面中显示的粗粒度，以及框架类的限制在很多情况下会表现得过于死板，给表示层的开发会带来一些额外的代码开销。

Tapestry与之不同的是，它是基于组件，而不是面向脚本语言(比如JSP和Velocity)的，组件是由一个定义文件(以XML的格式)、一个HTML模板和一个Java类构成的。

(4)Spring是业务组件层轻量级框架。Spring框架是一个基于IoC(依赖注入)和AOP(面向方面编程)的构架。用户可以通过Spring来利用普通Java对象(POJO)编程，使用依赖注入解析POJO间的依赖性，然后使用面向方面编程(AOP)将服务与它们相关联。采用依赖注入使得它可以很容易地实现Bean的装配，提供了简洁的AOP并据此实现事务管理等，但是它不具备处理应用分布式的能力。Spring的核心要点是支持不绑定到特定J2EE服务的可重用业务和数据访问对象。这样的对象可以在不同的J2EE环境(Web或EJB)、独立应用程序和测试环境之间重用。

Spring框架处于应用服务器和服务库的上方，服务整合的代码属于框架，并暴露于应用开发者。它与应用服务器整合的能力相对EJB 3.0要弱。但是Spting框架模块的可分离配置体现了它优于EJB 3.0的灵活性。

(5)持久层框架主要有Hibernate和各种JDO产品，以及iBATIS等。其中，Hiberhate是一个开源的O/R Mapping框架，它对JDBC进行了非常轻量级的对象封装，可以应用在任何使用JDBC的场合，可以在应用EJB的J2EE框架中取代CMP，完成数据持久化的重任。相对而言，Hibernate基本优势表现在：使用Java反射机制而不是字节码增强程序来实现透明性；使用简单；映射的灵活性很出色，它支持各种关系数据库，从一对一(1:1)到多对多(m:n)的各种复杂关系。其缺点是限制所使用的对象模型(例如，一个持久性类不能映射到多个表)。

iBATIS是一个简易的SQL Map工具，它是将手工编写的在XML。配置文件中的SQL语句映射成Java对象。使用iBATIS提供的O/R Mapping机制，对业务逻辑实现人员而言，面对的是纯粹的Java对象，这一层与通过Hibernate实现O/R. Mapping而言基本一致，而对于具体的数据操作，HiberTlate会自动生成SOL语句，而iBATIS则要求开发者编写具体的SQL语句。相对Hibernate等"全自动"O/R Mapping机制而言，iBATIS以SQL开发的工作量和数据库移植性上的让步，为系统设计提供了更大的自由空间。作为"全自动"ORM实现的一种有益补充，iBATIS的出现显得别具意义。

15、这是一道要求读者对比轻量级框架与重量级框架解决问题的侧重点的综合理解题。本题所涉及的知识点如下。

(1)设计与性能是实际框架选择的两个基本点，善于平衡才是框架选择的主要宗旨。轻量级框架和重量级框架解决问题的侧重点是不同的。

(2)轻量级框架侧重于减小开发的复杂度，相应的它的处理能力便有所减弱(如事务功能弱、不具备分布式处理能力)，比较适用于开发中小型企业应用。采用轻量级框架后，一方面因为采用基于POJOs的方法进行开发，使应用不依赖于任何容器，这可以提高开发调试效率；另一方面轻量级框架多数是开源项目，开源社区提供了良好的设计和许多快速构建工具，以及大量现成可供参考的开源代码，这有利于项目的快速开发。例如，目前Tomcat+Spring+Hibernate已经成为许多开发者开发

J2EE中小型企业应用偏爱的一种架构选择。

(3)作为重量级框架的EJB框架则强调高可伸缩性，适用于开发大型企业应用。在EJB体系结构中，一切与基础结构服务相关的问题和底层分配问题都由应用程序容器或服务器来处理，且EJB容器通过减少数据库访问次数及分布式处理等方式提供了专门的系统性能解决方案，能够充分解决系统性能问题。

(4)轻量级框架的产生并非是对重量级框架的否定，在某种程度上可以说二者是互补的。轻量级框架旨在开发具有更强大、功能更完备的企业应用；而EJB3.0规范则在努力简化J2EE的使用，以使得EJB不仅仅是擅长处理大型企业系统，也利用开发中小型系统，这也是EJB轻量化的一种努力。对于大型企业的应用及将来可能涉及能力扩展的中小型应用，结合使用轻量级框架和重量级框架也不失为一种较好的解决方案。