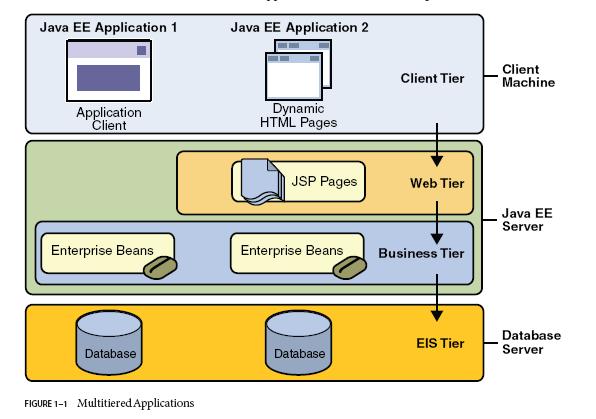
J2EE考点（简答题，问答题）

1. 概述

* J2EE是分布式的，多层的框架，有哪些层？每层有哪些组件？
* **industry standard for enterprise Java computing**
* **Utilize the new, lightweight Java EE 6 Web Profile to create next-generation web applications, and the full power of the Java EE 6 platform for enterprise applications**
* **The aim of the JavaTM Platform, Enterprise Edition (Java EE) platform is to provide developers a powerful set of APIs while reducing development time, reducing application complexity, and improving application performance**
* **@ introduces a simplified programming model**
* **Application logic is divided into components according to function, and the various application components （that make up a Java EE application ）are installed on different machines （depending on the tier in the multitiered Java EE environment to which the application component belongs）.**



* **Client-tier components run on the client machine**
  + **web client/thin client:** dynamic web pages containing various types of markup language (HTML, XML, and so on)
  + **Applets**
  + **application client:** typically has a graphical user interface (GUI) created from the Swing or the Abstract Window Toolkit (AWT) API
  + **The JavaBeansTM Component Architecture**
* **Web-tier components run on the Java EE server**
  + **Java Servlet, JavaServer Faces, and JavaServer PagesTM (JSPTM)**
* **Business-tier components run on the Java EE server**
  + **Enterprise JavaBeansTM (EJBTM)**
* **Enterprise information system (EIS)-tier software runs on the EIS server**
  + enterprise infrastructure systems such as enterprise resource planning (ERP), mainframe transaction processing, database systems, and other legacy information systems
* 是通过容器管理的，容器对每层提供什么服务？
* **The Java EE security model lets you configure a web component or enterprise bean so that system resources are accessed only by authorized users.**
* **The Java EE transaction model lets you specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.**
* **JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access these services.**

**The Java EE remote connectivity model manages low-level communications between clients and enterprise beans. After an enterprise bean is created, a client invokes methods on it as if it were in the same virtual machine**

* J2EE有哪些开发角色？
* **The Java EE product provider**
  + **designs and makes available for purchase the Java EE platform APIs, and other features defined in the Java EE specification.**
  + **typically application server vendors who implement the Java EE platform according to the Java EE 5 Platform specification.**
* **The tool provider**
  + **creates development, assembly, and packaging tools used by component providers, assemblers, and deployers.**
* **Application Component Provider**
  + **creates web components, enterprise beans, applets, or application clients for use in Java EE applications.**
* **Application Assembler**
  + **receives application modules from component providers and assembles them into a Java EE application EAR file.**
* **Application Deployer and Administrator**
  + **configures and deploys the Java EE application, administers the computing and networking infrastructure where Java EE applications run, and oversees the runtime environment.**

1. Servlet

* Servlet生命周期
* **The life cycle of a servlet is controlled by the container in which the servlet has been deployed.**
* **When a request is mapped to a servlet, the container performs the following steps.**
  + **1. If an instance of the servlet does not exist, the web container**
    - **a. Loads the servlet class.**
    - **b. Creates an instance of the servlet class.**
    - **c. Initializes the servlet instance by calling the init method.**
  + **2. Invokes the service method, passing request and response objects.**
* **If the container needs to remove the servlet, it finalizes the servlet by calling the servlet’s destroy method.**
* 容器提供哪些服务

**A web container provides services such as request dispatching, security, concurrency, and life-cycle management. It also gives web components access to APIs such as naming, transactions, and email.**

* URL组成部分

[**http://[*host*]:[*port*][*request-path*]?[*query-string***](http://[host]:[port][request-path]?[query-string)***]***

* **The request path is further composed of the following elements:**
  + **Context path: A concatenation of a forward slash (/) with the context root of the servlet’s web application.**
  + **Servlet path: The path section that corresponds to the component alias that activated this request. This path starts with a forward slash (/).**
  + **Path info: The part of the request path that is not part of the context path or the servlet path.**
* Web.xml常用标签
  + **<servlet>**
    - **<servlet-name>\*\*</servlet-name>**
    - **<servlet-class>\*\*</servlet-class>**
  + **</servlet>**
  + **<servlet-mapping>**
    - **<servlet-name>\*\*</servlet-name>**
    - **<url-pattern>\*\*</url-pattern>**

**</servlet-mapping>**

* + **servlet-name的含义**
    - **首先，初始化参数、定制的URL模式以及其他定制通过此注册名而不是类名引用此servlet**
    - **其次,可在URL而不是类名中使用此名称**
      * 可使用http://host:port/webAppPrefix/Hello代替http://host:port/webAppPrefix/servlet.HelloServlet (以Tomcat为例)
* Session两种实现：URL重写，cookies 分别怎么实现
* **1、当用户第一次访问站点→创建一个新的会话对象（Httpsession）， Server分配一个唯一的会话标识号(sessionID)；**
  + **Servlet容器自动处理sessionID的分配**
  + **尽可能长，确保安全**
  + **把sessionID信息放到HttpSession对象中**
* **2、Server创建一个暂时的HTTP cookie**
  + **cookie存储这个sessionID（名:jsessionid）**
  + **Server将cookie添加到HTTP响应中**
  + **Cookie被放置到客户机浏览器中，存储到客户机硬盘**
* **客户浏览器发送包含Cookie的请求；**
* **4、根据客户机浏览器发送的sessionID信息（cookie），Server找到相应的HttpSession对象，跟踪会话**
* **5、在会话超时间隔期间，如果没有接收到新的请求， Server将删除此会话对象**
  + **用户又访问该站点，必须重新注册，确保安全**
* **Cookie被客户禁用时，采用URL重写机制：**
  + **调用reponse.encodeURL(*URL*)方法；**
  + **http://…;jsessionid=….**
* **1、5与Cookie机制相同**
* **2、Server将sessionID放在返回给客户端的URL中；**
* **3、客户浏览器发送的请求将包含sessionID；**
* **4、根据包含请求的sessionID信息（URL），Server找到相应的HttpSession对象，跟踪会话**
* Tomcat组织结构，里面的xml看一下，给一个请求，它如何处理
* **Server> 代表一个服务器，可包含多个Service**
  + **<Service> 可包含一个Engine，多个Connector**
    - **<Connector/> 代表通信接口**
    - **<Engine> 可包含多个Host**
      * **<Host> 可包含多个Context**
        + **<Context>**
        + **</Context>**
      * **</Host>**
    - **</Engine>**
  + **</Service>**
* **</Server>**
* [**http://localhost:8080/HelloServlet/**](http://localhost:8080/HelloServlet/)
* **1.请求被发送到本机端口8080，被Coyote Http/1.1 Connector获得；**
* **2. Connector将该请求交给它所在的Service的Engine来处理，并等待Engine的回应；**
* **3. Engine获得请求，匹配所有虚拟主机；**
* **4. Engine匹配到名为localhost的主机；**
* **5. localhost主机获得请求，匹配所拥有的所有Context；**
* **6. localhost主机匹配到路径为/HelloServlet的Context；**
* cookie，session的应用场景
* 过滤器、监听器的应用场景
* **Applications of filters include authentication, logging, image conversion, data compression, encryption, tokenizing streams, XML transformations, and so on.**
* **代码重用**
* **应用安全策略**
* **日志**
* **为特定目标浏览器传输XML输出**
* **图像转换、加密**
* **动态压缩输出**

**monitor and react to events in a servlet’s life cycle by defining listener objects whose methods get invoked when life-cycle events occur**

1. JSP（不考表达式语言和标准标签库）

* JSP生命周期
* **A JSP page services requests as a servlet. Thus, the life cycle and many of the capabilities of JSP pages (in particular the dynamic aspects) are determined by Java Servlet technology.**

**When a request is mapped to a JSP page, the web container first checks whether the JSP page’s servlet is older than the JSP page. If the servlet is older, the web container translates the JSP page into a servlet class and compiles the class. During development, one of the advantages of JSP pages over servlets is that the build process is performed automatically**

* 两种部分组成：静态，动态（JSP元素）
* JSP元素有哪些
* 指令元素，脚本元素，动作元素
* JSP指令有哪些
* **<jsp:include>**
* **<jsp:forward>**
* **<jsp:useBean><jsp:setProperty><jsp:getProperty>**用于访问JavaBean
* **<jsp:param>**以名-值对的形式为其他标签提供附加信息
* **<jsp:plugin><jsp:params><jsp:fallback>**1.产生与客户端游览器相关的HTML标签，有时还需要下载java插件，并插件中执行指定的applet或javabean 2.包含多个<jsp:param>,向Applet或者Javabean提供参数3.给指定的java插件不能启动时显示给用户一段文字 这两者都是前者的一部分，且只能在前者中使用，
* **<jsp:element>**用于动态定义一个XML元素的标签
* **<jsp:attribute>**用于在XMl中定义一个动作属性的值，在element中指定输出元素的属性
* **<jsp:body>**用于定义元素的内容
* **<jsp:text>**用于封装模板数据
* **<jsp:output>**用于输出XML声明和文档类型声明
* **<jsp:invoke><jsp:doBody>**
* **<jsp:root><jsp:declaration><jsp:scriptlet><jsp:expression>**
* JSP脚本元素有哪些

**1.声明**

**2.脚本段**

**3.表达式**

* JSP隐式对象有哪些
* **1.request**
* **2.response**
* **3.session**
* **4.application**
* **5.config**
* **6. pageContext**

**7. Out**

**8. page**

**9. exception**

* JSP有哪些作用域对象

**1.Page范围**

**2.Request范围**

**3.Session范围**

**4.Application范围**

1. MVC

* 控制流程
* **1. Client makes a request (Web browser).**
* **2. Servlet gets the client's request.**
* **3. Servlet determines which program elements (JavaBeans, EJBs or other objects) are required to carry out the specified request.**
* **4. JavaBeans or EJBs perform the business logic operations for the servlet, and encapsulate the results.**
* **5. Servlet selects a presentation template (JSP) for delivering the content back to the client.**
* **6. The JSP generates a specific response by accessing the resultant content available through the JavaBeans.**

1. JMS、Java Mail

* 应用场景
  + **web前端为顾客订货录入系统**
  + **仓库管理系统：接受订单、配送货品、把订单转交发货系统**
  + **发货系统：更新顾客的账号纪录，开始发货**
  + **异步传递消息**
* JMS两种消息域：点对点，发布订阅，他们的应用场景
* 点对点消息传送

1 多个生成方可向一个队列发送消息。生成方可共享连接或使用不同连接，但它们均可访问同一队列

2 多个接收者可使用一个队列中的消息，但每条消息只能由一个接收者使用。

3 接收者可共享连接或使用不同连接，但它们均可访问同一队列

4 发送者和接收者之间不存在时间上的相关性

5 可在运行时动态添加和删除发送者和接收者，即可根据需要扩展或收缩消息传送系统

6 消息在队列中的放置顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级以及使用消息时是否使用选择器等因素

* 发布/ 订阅消息传送

1 多个生成方可向一个主题发布消息。生成方可共享连接或使用不同连接，但它们均可访问同一主题

2 多个订户可使用一个主题中的消息。订户可检索发布到一个主题中所有消息订户可共享连接或使用不同连接，但它们均可访问同一主题。

3 长期订户可能处于活动状态，也可能处于非活动状态。在它们处于非活动状态时，代理会为它们保留消息

4 可在运行时动态添加和删除发布者和订户，这样，即可根据需要扩展或收缩消息传送系统。

消息发布到主题的顺序与发送顺序相同，但它们的使用顺序则取决于消息失效期、消息优先级以及使用消息时是否使用选择器等因素

5 发布者与订户之间存在时间上的相关性：主题订户只能使用在它创建订阅后发布的消息

* JMS消费者：同步，异步 之间的差别
* **消息的传递是异步的，客户机程序在消息到达之前，可以做其他事情。**
* **receive()方法：如果有可用的消息，返回这个消息，否则将一直等待**
* **receiveNoWait()方法：如果有可用的消息，返回这个消息，否则返回NULL**
* **receive(long timeout)方法：根据给定的超时参数制定的时间等待一个消息的到来，如果在这个时间之内有可用的消息，返回这个消息，如果超时后仍没有可用的消息，返回NULL**

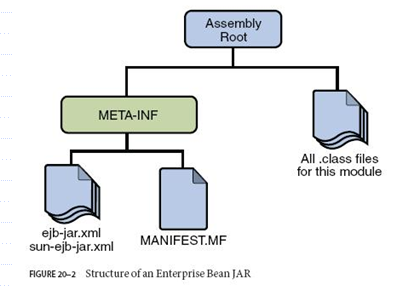
1. J2EE容器提供声明性安全和编程性安全

* ***Declarative security* expresses an application component’s security requirements using *deployment descriptors*.**
* **Deployment descriptors are external to an application, and include information that specifies how security roles and access requirements are mapped into environment-specific security roles, users, and policies.**
* ***Programmatic security* is embedded in an application and is used to make security decisions.**
* **Programmatic security is useful when declarative security alone is not sufficient to express the security model of an application.**

1. EJB

* EJB3.0有哪些EJB类型

**Session Bean**

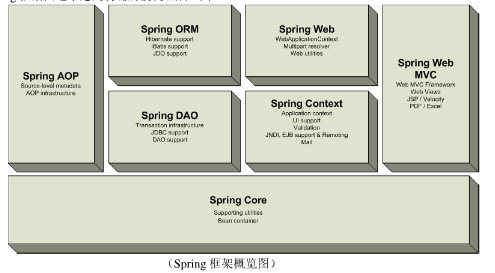
* 目录结构
* 
* 既包括EJB又包括应用模块的目录结构

1. JNDI提供什么服务

**为不同的目录提供一个共同的接口；**

**学习单一的API可以访问所有类型的目录服务器；**

1. structs架构是什么样的结构，M是什么，V是什么，C是什么



1. Hibernate与Java持久性API的比较

* Hibernate工作：配置文件——里面有哪些主要的标签
* **<class>标签：name属性为映射的对象，table属性为映射的表**
* **<id>标签：代表主键，column属性指定表中字段，type属性指定User实例中userID的类型**
* **<property>标签： column属性指定表中字段，type属性指定对象中属性的类型**
* 可能编写什么类

1. **DAO模式：数据访问对象在Java EE中是怎么用到的？**

使用数据访问对象来抽象和封装对数据源的所有访问。数据访问对象负责管理与数据源的连接，来获取和储存其中的数据。

数据访问对象实现与数据源相关的访问机制。 数据源可以是关系型数据库管理系统，可以是像B2B EXCHANGE这样的内部服务，可以是LDAP库，或者也可以是通过CORBA IIOP 或者是低层sockets来访问的商业服务. 依赖于DAO的商业组件只对他的客户端暴露一些非常简单的DAO外部接口. DAO将数据源的实现细节对客户端完全的隐藏了起来. 因为,暴露给客户端的DAO接口在低层数据源的实现发生改变时并不会随着改变,所以这种设计模式使得DAO可以适应不同的数据储存方式类型而不影响客户端和商业组件.最主要的, DAO还在组件和数据源之间扮演着协调者的角色.

以下是DAO设计模式中各个模块的解释：

1 BusinessObject指的是数据客户端,他通常需要去访问数据源以获得数据或储存数据.一个BusinessObject除了访问数据源的servlet或者helper bean之外也可以是会话BEAN,实体BEAN以及一些其他的JAVA对象.

2 DataAccessObject 是这个设计模式的核心部分, DataAccessObject为BusinessObject抽象了底层的数据访问实现的细节,使得访问数据变得透明. BusinessObject还将数据的装载和储存交给了DataAccessObject进行代理.

3 DataSource他表示的是数据源的实现. 一个数据源可以四像关系型数据库管理体统这样的数据库,可以是面向对象型的数据库管理系统,可以是XML文档,也可以是简单文件等等. 当然他也可以是其他的系统,(遗留系统,大型主机),可以是服务(B2B服务,信用卡局服务)或者是像LDAP这样的数据库等.

4 TransferObject他代表的是传递对象,一般用于数据的载体. DataAccessObject使用传递对象来将数据返回给客户端. DataAccessObject也可以使用传递对象来从客户端接受数据来将原先数据库中的数据进行更新.

优点与缺点:  
DAO设计模式带来的好处.

1 透明化:  
商业对象可以在完全不知道数据源如何具体实现的情况下来使用数据源. 访问数据源是透明的,因为实现细节已经被隐藏进了DAO.

2 迁移简单化:  
DAO层的出现,使得应用程序向不同的数据库实现进行迁移变的容易.商业对象可以对底层数据实现一无所知.这样,迁移只涉及到了对DAO层的修改.

3 减少在商业对象中的编程难度.  
由于DAO管理着所有的数据访问细节,因而大大简化了在商业对象和其他使用DAO的数据客户端里的代码.所有的实现细节相关的代码比如(SQL 语句)都包含在DAO而不在商业对象中. 这样使得代码变的更加健壮而且大大提高了开发效率.

4 将所有的数据访问都单独集中到一层中去.  
因为所有的数据访问操作现在都已经被DAO所代理,所以这个单独的数据访问层可以被看作可以是将数据访问实现和其余应用程序相互隔离的一层. 这样的集中,使得应用程序可以更加容易的来维护和管理.

缺点：  
1 对容器管理持久化无用  
由于EJB容器使用CMP(容器管理持久化)来管理实体BEAN. 容器会自动的为持久化储存访问提供服务.应用程序使用容器管理的实体BEAN则不需要DAO层的参与.因为应用程序服务器本身就可以透明的提供这些功能.然而,DAO在组合式CMP和BMP需要的场合下还是有用的.

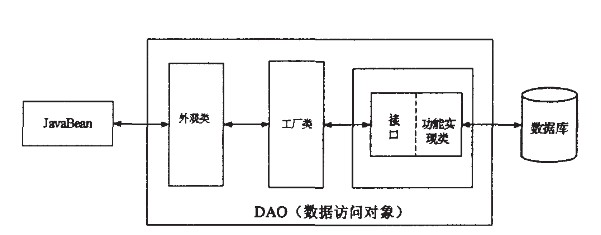
2 增加了多余的层.  
由于DAO在数据客户端和数据源之外多创建了一层对象,因而,需要对他进行设计和实现,来均衡这个设计模式的利弊. 但是,一般来说,采用此设计模式还是利大于弊的.

3 需要对类的相互继承关系进行设计.  
当使用工厂策略的时候,具体工厂类的继承关系和由这些工厂类生成的产品需要进行设计和实现. 我们需要仔细考虑这些多付出的工作是否真的可以产生出来更高的灵活性. 使用这个策略会使设计变的更加复杂,然而,你可以先从工厂方法模式开始来实现这个策略,然后在需要的情况下再转向抽象工厂

（下面是我一个个字码出来的。。居然不可以复制。。）

DAO是Data Access Object数据访问接口，处于业务逻辑与数据库资源中间。DAO模式是标准J2EE设计模式之一。这个模式把数据访问逻辑从业务逻辑中分离出来放入独立的接口中。在需要数据的时候，通过DAO来访问数据库。也就是说，JavaBean保留自己的业务逻辑方法，在需要数据的时候，通过DAO来访问数据库。在要求修改数据访问的时候，只要更新DAO的对象就可以了。

一个典型的DAO实现有以下组件：一个DAO外观（façade类），一个DAO工厂（factory类），一个DAO接口，一个实现了DAO接口的具体类，数据传输对象（有时称为值对象）。



这种设计模式的优点：

* 透明性好，业务对象可以再不知道数据源实现细节的情况下访问数据。由于一切数据访问细节被数据访问对象所隐藏，所以这种访问过程是透明的。
* 可移植性好，在应用系统中添加数据访问对象，可以使得前者能够很方便地移植到另外一种数据库实现上。业务对象与数据实现是隔离的，所以在移植过程中，仅仅对数据访问对象进行一些变化即可。
* 减少业务对象的代码复杂度。由于数据访问对象可以管理所有的数据访问复杂细节，这也就简化了业务模块和其他数据客户的代码。同时也提高了应用系统的整体可读性和开发率。
* 集中处理所有数据访问。由于所有的数据访问操作都移交给数据访问对象，这样应用系统其他部分就与数据访问实现隔离开来，而全部相关操作都与数据访问对象集中处理，这样也使得相关操作更加容易被维护和管理。

缺陷：

* 把DAO对象和SQL语言紧密的耦合在一起；

增加了程序编码的工作量。