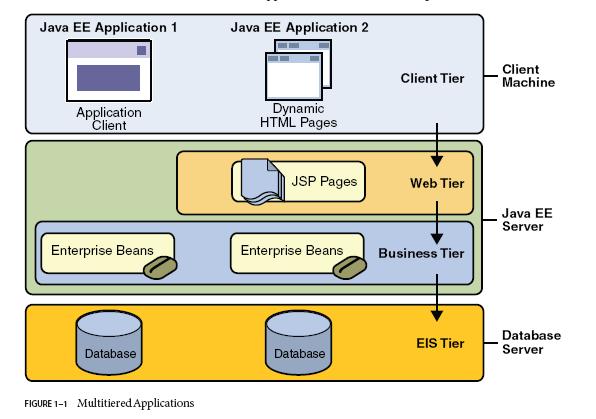
J2EE考点（简答题，问答题）

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？



* **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
* JSP指令有哪些
* JSP脚本元素有哪些
* JSP隐式对象有哪些
* JSP有哪些作用域对象
* JSP动作：include，forward动作
* JSP动作与重定向差别

1. 不考JDBC
2. 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
2. **JNDI：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