# JAVAWEB

## 1. web基础

### （★）forward 和redirect 的区别?

答：

forward 是容器中控制权的转向，是服务器请求资源，服务器直接访问目标地址的URL，把那个URL 的响应内容读取过来，然后把这些内容再发给浏览器，浏览器根本不知道服务器发送的内容是从哪儿来的，所以它的地址栏中还是原来的地址。redirect 就是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址，一般来说浏览器会用刚才请求的所有参数重新请求，所以session,request 参数都可以获取，并且从浏览器的地址栏中可以看到跳转后的链接地址。前者更加高效，在前者可以满足需要时，尽量使用forward()方法，并且，这样也有助于隐藏实际的链接；在有些情况下，比如，需要跳转到一个其它服务器上的资源，则必须使用sendRedirect()方法。

### （★）说出数据连接池的工作机制是什么?

答：

J2EE 服务器启动时会建立一定数量的池连接，并一直维持不少于此数目的池连接。客户端程序需要连接时，池驱动程序会返回一个未使用的池连接并将其表记为忙。如果当前没有空闲连接，池驱动程序就新建一定数量的连接，新建连接的数量有配置参数决定。当使用的池连接调用完成后，池驱动程序将此连接表记为空闲，其他调用就可以使用这个连接。

### （★）get 和post 的区别？

答：

Form 中的get 和post 方法，在数据传输过程中分别对应了HTTP 协议中的GET 和POST 方法。二者主要区别如下：

1）Get 是用来从服务器上获得数据，而Post 是用来向服务器上传递数据；

2）Get 将表单中数据按照variable=value 的形式，添加到action 所指向的URL 后面，并且两者使用“?”连接，而各个变量之间使用“&”连接；Post 是将表单中的数据放在form 的数据体中，按照变量和值相对应的方式，传递到action所指向URL；

3）Get 是不安全的，因为在传输过程，数据被放在请求的URL 中；Post 的所有操作对用户来说都是不可见的；

4）Get 传输的数据量小，这主要是因为受URL 长度限制；而Post 可以传输大量的数据，所以在上传文件只能使用Post；

5）Get 限制Form 表单的数据集必须为ASCII 字符，而Post 支持整个ISO10646字符集；

6）Get 是Form 的默认方法。

### 4. （★）什么情况下调用doGet()和doPost()？

答：

Jsp 页面中的form 标签里的method 属性为get 时调用doGet()，为post时调用doPost()。

### （★）如何从form 表单中得取checkbox 的值；

答：

可在页面把checkbox 的name 属性取同一个，value 属性取每个条目的id,后台用getParamter(“name”)能取到checkbox 的一组值。

### （★）页面中有一个命名为bank No 的下拉列表，写脚本获取当前选项的索引值。

答：

用java 或javaScript 的处理方式分别如下：

Java： request.getParameter(“bank No”);

javaScript：

var selectItems = document.getElementsByName(“bank No”);

selectItems[0].value;

### （★）常用的web 容器和开发工具；

答：

最常用的容器包括：tomcat、weblogic；开发工具有：eclipse,jbuilder。

### （★）Request 对象的主要方法?

答：

setAttribute(String name,Object)：设置名字为name 的属性值

getAttribute(String name)：返回由name 指定的属性值

getAttributeNames()：返回request 对象所有属性的名字集合(枚举)

getCookies()：返回客户端的所有Cookie 对象，结果是一个Cookie 数组

getCharacterEncoding()：返回请求中的字符编码方式

getContentLength()：返回请求的Body 的长度

getHeader(String name)：获得HTTP 协议定义的文件头信息

getHeaders(String name)：返回指定名的request Header 的所有值(枚举)

getHeaderNames()：返回所有request Header 的名字(枚举)

getInputStream()：返回请求的输入流，用于获得请求中的数据

getMethod()：获得客户端向服务器端传送数据的方法

getParameter(String name)：获得客户端请求中传送的name 指定的参数值

getParameterNames()：获得客户端传送给服务器端的所有参数的名字(枚举)

getParameterValues(String name)：获得有name 指定的参数的所有值

getProtocol()：获取客户端向服务器端传送数据所依据的协议名称

getQueryString()：获得查询字符串

getRequestURI()：获取发出请求字符串的客户端地址

getRemoteAddr()：获取客户端的IP 地址

getRemoteHost()：获取客户端的名字

getSession([Boolean create])：返回和请求相关Session

getServerName()：获取服务器的名字

getServletPath()：获取客户端所请求的脚本文件的路径

getServerPort()：获取服务器的端口号

removeAttribute(String name)：删除请求中的一个属性

### （★）页面间对象传递的方法。

答：

request,session,application,cookie 等

### （★）详细描述MVC。

答：

基于Java 的Web 应用系统采用MVC 架构模式，即model（模型）、view（视图）、control（控制）分离设计；这是目前WEB 应用服务系统的主流设计方向。

Model：即处理业务逻辑的模块，每一种处理一个模块；

View：负责页面显示，显示MODEL 处理结果给用户，主要实现数据到页面转换过程；

Control：负责每个请求的分发，把FORM 数据传递给MODEL 处理，把处理结果的数据传递给VIEW 显示。

### （★）MVC 的各个部分都有那些技术来实现?如何实现?

答：

MVC 是Model－View－Controller 的简写。"Model" 代表的是应用的业务逻辑（通过JavaBean，EJB 组件实现）， "View" 是应用的表示面（由JSP 页面产生），"Controller" 是提供应用的处理过程控制（一般是一个Servlet），通过这种设计模型把应用逻辑，处理过程和显示逻辑分成不同的组件实现。这些组件可以进行交互和重用

### （★）应用服务器有那些？

答：

BEA WebLogic Server, IBM WebSphere Application Server, Oracle9i Application Server, JBoss, Tomcat。

### （★）几种会话跟踪技术？

答：

cookie、URL 重写、设置表单隐藏域。

### （★）BS 与CS 的联系与区别？

答：

C/S 是Client/Server 的缩写，是客户机与服务器结构的应用程序，服务器通常采用高性能的PC、工作站或小型机，并采用大型数据库系统，如Oracle、Sybase、Informix 或SQL Server。客户端需要安装专用的客户端软件。B/Ｓ是Brower/Server 的缩写，是浏览器和服务器结构的应用程序，即Web 应用程序，客户机上只要安装一个浏览器（Browser），如Netscape Navigator 或InternetExplorer，服务器安装Oracle、Sybase、Informix 或SQL Server 等数据库。在这种结构下，用户界面完全通过WWW 浏览器实现，一部分事务逻辑在前端实现，但是主要事务逻辑在服务器端实现。浏览器通过Ｗeb Server 同数据库进行数据交互。

C/S 与B/S 区别：

1)硬件环境不同:

C/S 一般建立在专用的网络上, 小范围里的网络环境, 局域网之间再通过专门服务器提供连接和数据交换服务；B/S 建立在广域网之上的, 不必是专门的网络硬件环境,例与电话上网, 租用设备. 信息自己管理. 有比C/S 更强的适应范围, 一般只要有操作系统和浏览器就行；

2)对安全要求不同：

C/S 一般面向相对固定的用户群, 对信息安全的控制能力很强. 一般高度机密的信息系统采用C/S 结构适宜. 可以通过B/S 发布部分可公开信息；B/S 建立在广域网之上, 对安全的控制能力相对弱, 可能面向不可知的用户；

3)对程序架构不同：

C/S 程序可以更加注重流程, 可以对权限多层次校验, 对系统运行速度可以较少考虑；B/S 对安全以及访问速度的多重的考虑, 建立在需要更加优化的基础之上.比C/S 有更高的要求B/S 结构的程序架构是发展的趋势, 从MS 的.Net 系列的BizTalk 2000 Exchange 2000 等, 全面支持网络的构件搭建的系统. SUN 和IBM 推的JavaBean 构件技术等,使B/S 更加成熟；

4)软件重用不同：

C/S 程序可以不可避免的整体性考虑, 构件的重用性不如在B/S 要求下的构件的重用性好；B/S 对的多重结构,要求构件相对独立的功能. 能够相对较好的重用.就入买来的餐桌可以再利用,而不是做在墙上的石头桌子；

5)系统维护不同：

C/S 程序由于整体性, 必须整体考察, 处理出现的问题以及系统升级. 升级难. 可能是再做一个全新的系统；B/S 构件组成,方面构件个别的更换,实现系统的无缝升级. 系统维护开销减到最小.用户从网上自己下载安装就可以实现升级；

6)处理问题不同：

C/S 程序可以处理用户面固定, 并且在相同区域, 安全要求高需求, 与操作系统相关. 应该都是相同的系统；B/S 建立在广域网上, 面向不同的用户群, 分散地域, 这是C/S 无法作到的. 与操作系统平台关系最小；

7)用户接口不同：

C/S 多是建立的Window 平台上,表现方法有限,对程序员普遍要求较高；B/S 建立在浏览器上, 有更加丰富和生动的表现方式与用户交流. 并且大部分难度减低,减低开发成本；

8)信息流不同：

C/S 程序一般是典型的中央集权的机械式处理, 交互性相对低；B/S 信息流向可变化, B-B B-C B-G 等信息、流向的变化, 更像交易中心。

### （★）过滤器有哪些作用？

答：

可以验证客户是否来自可信的网络，可以对客户提交的数据进行重新编码，可以从系统里获得配置的信息，可以过滤掉客户的某些不应该出现的词汇，可以验证用户是否登录，可以验证客户的浏览器是否支持当前的应用，可以记录系统的日志等等。

### （★）过滤器的用法？（对客户端的请求统一编码和对客户端进行认证）

答：

首先要实现（implements）Filter 接口，同时覆盖Filter 接口的三个方法：

init(FilterConfig config) //用于获得FilterConfig 对象；

doFilter(ServletRequest request, ServletResponse response,

FilterChain chain) //进行过滤处理一些业务；

destroy() //销毁Filter

### （★）简述HttpSession 的作用、使用方法，可用代码说明。（3 分钟）

答：

HttpSession 中可以跟踪并储存用户信息，把值设置到属性中，有2 个方法：setAttribute(),getAttrribute()；

例如：在一个方法中用session.setAttribute(“student”,student);在session 中设置一个属性名为student,值为一个名为student 的对象。而后可在同一session 范围内用getAttribute(“student”)取出该属性，得到student对象

### （★）Web.Xml 的作用？

答：

用于配置web 应用的信息；如listener、filter 及servlet 的配置信息等

### （★）写出熟悉的JSTL 标签。

答：

<c:if>、<c:choose>、<c: when>、<c: otherwise>、<c:forEach>、<c:set>

### （★★）写一个自定义标签；

答：

代码如下：

import javax.servlet.jsp.tagext.\*;

import javax.servlet.jsp.\*;

import java.io.\*;

public class TimeTag extends SimpleTagSupport{

private boolean isServer = true;

public void setServer(boolean isServer){

this.isServer = isServer;

}

public void doTag() throws JspException, IOException{

JspWriter out = getJspContext().getOut();

if(isServer) {

out.println(new java.util.Date());

}else{

out.println("<script language=\"javascript\">");

out.println("document.write(new Date());");

out.println("</script>");

}

}

}

### （★）sendRedirect()和forward()方法有什么区别？

答：

sendRedirect()方法会创建一个新的请求，而forward()方法只是把请求转发到一个新的目标上。重定向(redirect)以后，之前请求作用域范围以内的对象就失效了，因为会产生一个新的请求，而转发(forwarding)以后，之前请求作用域范围以内的对象还是能访问的。一般认为sendRedirect()比forward()要慢。

### （★）什么是cookie？session和cookie有什么区别？

答：

cookie是Web服务器发送给浏览器的一块信息。浏览器会在本地文件中给每一个Web服务器存储cookie。以后浏览器在给特定的Web服务器发请求的时候，同时会发送所有为该服务器存储的cookie。下面列出了session和cookie的区别：

无论客户端浏览器做怎么样的设置，session都应该能正常工作。客户端可以选择禁用cookie，但是，session仍然是能够工作的，因为客户端无法禁用服务端的session。

在存储的数据量方面session和cookies也是不一样的。session能够存储任意的Java对象，cookie只能存储String类型的对象。

### （★）doGet()方法和doPost()方法有什么区别？

答：

doGet：GET方法会把名值对追加在请求的URL后面。因为URL对字符数目有限制，进而限制了用在客户端请求的参数值的数目。并且请求中的参数值是可见的，因此，敏感信息不能用这种方式传递。

doPOST：POST方法通过把请求参数值放在请求体中来克服GET方法的限制，因此，可以发送的参数的数目是没有限制的。最后，通过POST请求传递的敏感信息对外部客户端是不可见的。

### （★）什么是注解?

答：

定义：注解（Annotation），也叫元数据。一种代码级别的说明。它是JDK1.5及以后版本引入的一个特性，与类、接口、枚举是在同一个层次。它可以声明在包、类、字段、方法、局部变量、方法参数等的前面，用来对这些元素进行说明，注释。

作用分类：

①编写文档：通过代码里标识的元数据生成文档【生成文档doc文档】

②代码分析：通过代码里标识的元数据对代码进行分析【使用反射】

③编译检查：通过代码里标识的元数据让编译器能够实现基本的编译检查【Override】

### （★★★）ActionContext是用来干什么的？

答：

ActionContext是Action的上下文。

通过ActionContext这个类来可以获得:request，application，session，Locale、ValueStack,ConversionErrors

1 可以获得servlet的api：request，application，session。

2 进行国际化：获得语言环境 Locale。

3 收集错误信息：ConversionErrors。

4 有个值栈来装action实例与创建的模型驱动的实例:ValueStack。

5 ActionContext是线程安全的-用本地线程实现的。

### 26. （★★）如果我的网站用的 utf-8 编码，为防止乱码出现，都需要注意哪些地方？

答：

1)数据库中库和表都用 utf8 编码

2)java连接 mysql，指定数据库编码为 utf8 ;

3)java文件指定编码为 utf-8

4)网站下所有文件的编码为 utf8

5)html 文 件 指 定 编 码 为 utf-8

### 27. （★★）简述 Tcp 协议的三次握手过程。

答：

TCP 是主机对主机层的传输控制协议，提供可靠的连接服务，采用三次握手确认建立一个连接：

第一次握手：建立连接时，客户端发送 syn 包(syn=j)到服务器，并进入 SYN\_SEND 状态，等待服务器确认；

第二次握手：服务器收到 syn 包，必须确认客户的 SYN（ack=j+1），同时自己也发送一个 SYN 包（syn=k），即 SYN+ACK 包，此时服务器进入 SYN\_RECV 状态；

第三次握手：客户端收到服务器的 SYN＋ACK 包，向服务器发送确认包 ACK(ack=k+1)，此包发送完毕，客户端和服务器进入 ESTABLISHED 状态，完成三次握手。

完成三次握手，客户端与服务器开始传送数据。

### 28. （★★★）SOAP是什么？

答：

SOAP是simple object access protocal的缩写，即简单对象访问协议。 是基于XML和HTTP的一种通信协议。是webservice所使用的一种传输协议，webservice之所以能够做到跨语言和跨平台，主要是因为XML和HTTP都是独立于语言和平台的。Soap的消息分为请求消息和响应消息，一条SOAP 消息就是一个普通的XML 文档，包含下列元素：

1、 必需的 Envelope 元素，可把此XML 文档标识为一条SOAP 消息

2、 可选的 Header 元素，包含头部信息

3、 必需的 Body 元素，包含所有的调用和响应信息

4、 可选的 Fault 元素，提供有关在处理此消息所发生错误的信息

### 29. （★）截止JDK1.8版本,java并发框架支持锁包括?

A.读写锁 B.自旋锁 C.X锁 D.乐观锁 E.排他锁

答:

ABD

### 30. （★）什么是cookie,作用是什么?cookie和session的区别与联系

答:

cookie是存在于客户端（浏览器）。

cookie的使用是由浏览器按照一定的原则在后台自动发送给服务器的。浏览器检查所有存储的cookie，如果某个cookie所声明的作用范围大于等于将要请求的资源所在的位置，则把该cookie附在请求资源的HTTP请求头上发送给服务器。

cookie的内容主要包括：名字，值，过期时间，路径和域。其中路径与域一起构成cookie的作用范围。若不设置过期时间，则表示这个cookie的生命期为浏览器会话期间，关闭浏览器窗口，cookie就消失

Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录

在服务器上。这就是Session。客户端浏览器再次访问时只需要从该Session中查找该客户的状态就可以了。

每个用户访问服务器都会建立一个session，那服务器是怎么标识用户的唯一身份呢？事实上，用户与服务器建立连接的同时，服务器会自动为其分配一个SessionId

总结

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的cookie并进行cookie欺骗，考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能，考虑到减轻服务器性能方面，应当使用cookie。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、可以考虑将登陆信息等重要信息存放为session，其他信息如果需要保留，可以放在cookie中。

### 31.(★)请写出HTTP 4种以上的状态码，并解释各是什么意义?

200 请求成功

300 多重选择。链接列表，用户可以选择某链接 到达目的地。最多允许五个地址

400 因为语法错误，服务器未能理解请求

404 服务器无法找到被请求的页面

500 请求未完成服务器遇到不可预知的情况

更多详情尽在https://blog.csdn.net/w405722907/article/details/72722640

### 32.(★)请写出Spring的几种注解，并说明涵义，越多越好。

@controller 控制器（注入服务）

@service 服务（注入dao）

@repository dao（实现dao访问）

@component （把普通pojo实例化到spring容器中，相当于配置文件中的<bean id=""class=""/>）

### 33.(★★)请列出java常见的开源数据连接池，并对参数做出简单的说明

答：

**C3P0：**

User 数据库账号

Password 数据库密码

jdbcUrldriverClass 链接驱动

acquireIncrement 如果池中数据连接不够时一次增长多少个

initialPoolSize 初始化数据库连接池时连接的数量

maxPoolSize 数据库连接池中的最大的数据库连接数

minPoolSize 数据库连接池中的最小的数据库连接数

**DBCP：**

driverClassName 驱动名

url 链接地址

Username 用户名

Password 密码

initialSize 初试连接数

maxTotal 最大活跃数

maxIdle 最大idle数

minIdle 最小idle数

maxWaitMillis最长等待时间(毫秒)

### 34.(★★)SpringMVC的MVC理解:Java Web三层架构的理解及各自都有哪或开源组件

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。以上的说法太正式.简单明了的说在我们经常用到的接触到的就是这样：

M —>model(模型数据)

V —>view (视图展现)

C —>controller(控制器)

三层架构与MVC的关系:

表现层 ———–>mvc框架

业务层 ———–>service层

持久层 ———–>Dao层框架

细说三层架构与之对应的流行框架

表现层 ———–>mvc框架———>strtus2/SpringMVC|

业务层 ———–>service层 |  Spring容器

持久层 ———–>Dao层框架——>Hibernate/Mybatis|

Spring框架作为一个容器,利用核心功能IOC、AOP控制管理着三层架构,实现整体架构的解耦，也就是市面流行的strtus2/SpringMVC是表现层框架.

再细说MVC框架与strtus2/SpringMVC的对应关系

model模型:数据库表对应的实体类,用作前后台的数据交互。

view模型:视图的展现,在框架中的返回值(return “index”;)可以直接返回到指定名称的页面

controller模型:接收用户请求,并进行处理.可以接收页面表单数据,页面的表单数据就是以model的属性来传参

### 35.(★★)请描述Struts/Spring/Hibernate/Websphere/Tomcat/DB2/Oracle是做什么的？

**Struts2：**

Struts2是一个基于MVC设计模式的Web应用框架，它本质上相当于一个servlet，在MVC设计模式中，Struts2作为控制器(Controller)来建立模型与视图的数据交互。Struts 2是Struts的下一代产品，是在 struts 1和WebWork的技术基础上进行了合并的全新的Struts 2框架。其全新的Struts 2的体系结构与Struts 1的体系结构差别巨大。Struts 2以WebWork为核心，采用拦截器的机制来处理用户的请求，这样的设计也使得业务逻辑控制器能够与ServletAPI完全脱离开，所以Struts 2可以理解为WebWork的更新产品。虽然从Struts 1到Struts 2有着太大的变化，但是相对于WebWork，Struts 2的变化很小。

**Spring:**

Spring框架是由于软件开发的复杂性而创建的。Spring使用的是基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅仅限于服务器端的开发。从简单性、可测试性和松耦合性角度而言，绝大部分Java应用都可以从Spring中受益。

◆目的：解决企业应用开发的复杂性

◆功能：使用基本的JavaBean代替EJB，并提供了更多的企业应用功能

◆范围：任何Java应用

Spring是一个轻量级控制反转(IoC)和面向切面(AOP)的容器框架。

**Hibernate:**

Hibernate是一个开放源代码的对象关系映射框架，它对JDBC进行了非常轻量级的对象封装，它将POJO与数据库表建立映射关系，是一个全自动的orm框架，hibernate可以自动生成SQL语句，自动执行，使得Java程序员可以随心所欲的使用对象编程思维来操纵数据库。 Hibernate可以应用在任何使用JDBC的场合，既可以在Java的客户端程序使用，也可以在Servlet/JSP的Web应用中使用，最具革命意义的是，Hibernate可以在应用EJB的JaveEE架构中取代CMP，完成数据持久化的重任。

**Tomcat:**

Tomcat是Apache 软件基金会（Apache Software Foundation）的Jakarta 项目中的一个核心项目，由Apache、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现，Tomcat 5支持最新的Servlet 2.4 和JSP 2.0 规范。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

Tomcat 服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache 服务器，可利用它响应HTML（标准通用标记语言下的一个应用）页面的访问请求。实际上Tomcat是Apache 服务器的扩展，但运行时它是独立运行的，所以当你运行tomcat 时，它实际上作为一个与Apache 独立的进程单独运行的。

**DB2:**

IBM DB2 是美国IBM公司开发的一套关系型数据库管理系统，它主要的运行环境为UNIX（包括IBM自家的AIX）、Linux、IBM i（旧称OS/400）、z/OS，以及Windows服务器版本。

DB2主要应用于大型应用系统，具有较好的可伸缩性，可支持从大型机到单用户环境，应用于所有常见的服务器操作系统平台下。 DB2提供了高层次的数据利用性、完整性、安全性、可恢复性，以及小规模到大规模应用程序的执行能力，具有与平台无关的基本功能和SQL命令。DB2采用了数据分级技术，能够使大型机数据很方便地下载到LAN数据库服务器，使得客户机/服务器用户和基于LAN的应用程序可以访问大型机数据，并使数据库本地化及远程连接透明化。 DB2以拥有一个非常完备的查询优化器而著称，其外部连接改善了查询性能，并支持多任务并行查询。 DB2具有很好的网络支持能力，每个子系统可以连接十几万个分布式用户，可同时激活上千个活动线程，对大型分布式应用系统尤为适用。

**ORACLE:**

ORACLE数据库系统是美国ORACLE公司（甲骨文）提供的以分布式数据库为核心的一组软件产品，是目前最流行的客户/服务器(CLIENT/SERVER)或B/S体系结构的数据库之一。比如SilverStream就是基于数据库的一种中间件。ORACLE数据库是目前世界上使用最为广泛的数据库管理系统，作为一个通用的数据库系统，它具有完整的数据管理功能；作为一个关系数据库，它是一个完备关系的产品；作为分布式数据库它实现了分布式处理功能。但它的所有知识，只要在一种机型上学习了ORACLE知识，便能在各种类型的机器上使用它。

Oracle数据库最新版本为Oracle Database 12c。Oracle数据库12c 引入了一个新的多承租方架构，使用该架构可轻松部署和管理数据库云。此外，一些创新特性可最大限度地提高资源使用率和灵活性，如Oracle Multitenant可快速整合多个数据库，而Automatic Data Optimization和Heat Map能以更高的密度压缩数据和对数据分层。这些独一无二的技术进步再加上在可用性、安全性和大数据支持方面的主要增强，使得Oracle数据库12c 成为私有云和公有云部署的理想平台。

### 36.(★★★)使用sparing mybatis\ Hibernate、jsp\struts框架实现一个业务的新增功能，你会创建哪些文件。描述一下如何配置使用

使用struts框架

**创建**

struts-config.xml

**配置**

首先，添加/register 到<action>元素的path属性。ActionServlet使用Web容器转发给它的URI来选择正确的Action类。URI和ActionMapping的path属性匹配。这里，请求给出的路径必须在去除前缀和后缀后和/register匹配前缀或后缀通常是/do/或者.do。我们的练习中，将后缀设置为.do。当URI具有一个.do扩展名，容器就知道将请求转发给ActionServlet。Struts会自动去除扩展名，所以我们在配置时不必加上它们。

下一步添加：registerForm到<action>元素的name属性。<action>元素使用name属性来识别哪个ActionForm将被创建，并将提交的表单组装给他。

然后，添加 app.RegisterAction到<action>元素的type属性。ActionServlet使用这个属性来识别将用来处理请求的Action类。

接下来，在<forward>元素下，添加success到nam属性并且/success.html 到path属性。最后，再在另一个<forward>下添加failure到name属性，/failure.html 到path属性。这些元素将创建ActionForward对象，我们将用它来选择程序的控制流。<forward>元素定义了在RegisterAction中使用的逻辑名称之间的关联

### 37.(★)Session，Cookie区别

答：Cookie与Session的区别

|  |  |
| --- | --- |
| Cookie | Session |
| 存储在客户端 | 存储在服务器端 |
| 两种类型  ※有生命周期  ※无生命周期 | 两种实现方式  ※依赖于Cookie  ※url重写 |
| 父路径不能访问子路径的cookie | 同一个session的窗口共享一个session |
| 典型应用  ※3个月不用再登录  ※购物车 | 典型应用  ※用户登录  ※购物车也可以用session实现 |
| 不可靠 | 可靠 |

### 38.(★★★)aop的设计模式和机制是什么

### 39.(★)简述Cookie的主要属性和作用，以及Cookie与Session的关系？

name字段 ：一个cookie的名称。

value字段 ：一个cookie的值。

domain字段 ：可以访问此cookie的域名

path字段：可以访问此cookie的页面路径。

Size字段 ：此cookie大小。

http字段 ：cookie的httponly属性。若此属性为true，则只有在http请求头

中会带有此cookie的信息，而不能通过document.cookie来访问此cookie。

secure 字段 ：设置是否只能通过https来传递此条cookie

关系：

Cookies是属于Session对象的一种。但有不同，Cookies不会占服务器资源，是存在客服端内存或者一个cookie的文本文件中；而“Session”则会占用服务器资源。所以，尽量不要使用Session，而使用CookiesCookies是属于Session对象的一种。但有不同，Cookies不会占服务器资源，是存在客服端内存或者一个cookie的文本文件中；而“Session”则会占用服务器资源。所以，尽量不要使用Session，而使用Cookies

### 40.(★)GET请求与POST请求有什么不同？

GET请求在URL中传送的参数是有长度限制的，而POST没有。

GET比POST更不安全，因为参数直接暴露在URL上，所以不能用来传递敏感信息。

GET参数通过URL传递，POST放在Request body中。

GET请求参数会被完整保留在浏览器历史记录里，而POST中的参数不会被保留。

GET请求只能进行url编码，而POST支持多种编码方式。

GET请求会被浏览器主动cache，而POST不会，除非手动设置。

GET产生的URL地址可以被Bookmark，而POST不可以。

GET在浏览器回退时是无害的，而POST会再次提交请求。

### 41.(★★)请分别说明用Servlet机制实现页面跳转的方式？

转发：

1、request.getRequestDispatcher("跳转页面URL").forward(request, response);

2、POST 请求；

3、页面跳转后地址栏URL不变；

4、发送一次请求；

5、不能跳到其他网站；

6、传值方式（request.setAttribute("key", value)）

重定向：

1、respose.sendRedirect("跳转页面URL");

2、GET 请求；

3、页面跳转后地址栏URL改变；

4、至少发送两次请求；（request与服务器断开，重新请求URL）

5、可以跳到其他网站；

6、传值方式：url = "...? key = value"

### 42.(★)浏览web网页服务默认的端口号是（ A）

1. 8080
2. 23333
3. 4399
4. 80

### 43.(★)下列是Jsp作用域的通信对象的有（ABC）

1. Session
2. Application
3. PageContext
4. Cookie

### 44.(★)在Spring框架中，面向切面编程（AOP）的目标在于（A）

1. 编写程序时不用关心其依赖组件得到实现
2. 将程序中涉及的公用问题几种解决
3. 封装JDBC访问数据的代码，简化数据访问层的重复代码
4. 实现页面的“无刷新

### 45.(★)在JavaScript里，下列选项中不属于数组方法的是（B）

1. Sort()
2. Length()
3. Concat()
4. Reverse()

### 46.(★)下面关闭名为mydiv层代码正确的是（C）

1. Document.getElementById(mydiv).style.display=”none”
2. Document.getElementById(“mydiv”).style.display=none
3. Document.getElementById(“mydiv”).style.display=”none”
4. Document.getElementById(“mydiv”).style.display==”none”

### 47.(★)当DOM加载完成后要执行的函数，下面哪个是正确的（C）

1. jQuery(expression,[context])
2. jQuery(html,[owerDocument])
3. jQuery(callback)
4. jQuery(elements)

Get与Post的区别

### 48.(★)下面那个不是块段元素？BC

A:div B:url C:input D:h1

### 49.(★)下面CSS优先级算法的说法是？AC

A:优先级就近原则，样式定义最近者为准

B:重叠的定义，后引入的CSS文件覆盖引入CSS文件

C:内联样式表的权值高

D:Class类选择器的权值比HTML标签选择器的权值高

### 50.(★)Bootstrap栅格系统把一行分为（）列C

A:8 B:10 C:12 D:16

### 51.(★)页面中有一个性别单选按钮，请设置“男”为选中状态。正确的是（D）

<input type = “”radio” name = “sex”> 男

<input type = “”radio” name = “sex”> 女

A:$(“sex(0)”).attr(“checked”,true);

B:$(“#sex(0)”).attr(“checked”,true);

C:$(“{ name = sex};radio”).attr(“checked”,true);

D:$(“:radio{name = sex};eq(0)”).attr(“checked”,true);

### 52.(★★)执行下面语句单击按钮<input type=”button” id=”click” value=”点击我”>会有什么效果？(C)

$(document).ready(function(){

$(“#click”). click (function){

alert(“click one time”);

});

$(“#click”).click(function(){

alert(“click two time”);

});

});

A:弹出一次对话框，显示click one time

B: 弹出一次对话框，显示click two time

C: 弹出两次对话框，依次显示click one time，click two time

D:js编译错误

### 53.(★)下面说法错误的有（ACD）

1. 在类方法中可用this来调用本类的类方法
2. 在类方法中调用本类的类方法时可直接调用
3. 在类方法中只能调用本类中的类方法
4. 在类方法中绝对不能调用实例方法

### 54.(★★)请解释Spring Bean的生命周期

Spring Bean的生命周期简单易懂。在一个bean实例被初始化时，需要执行一系列的初始化操作以达到可用的状态。同样的，当一个bean不在被调用时需要进行相关的析构操作，并从bean容器中移除。

Spring bean factory 负责管理在spring容器中被创建的bean的生命周期。Bean的生命周期由两组回调（call back）方法组成。

初始化之后调用的回调方法。

销毁之前调用的回调方法。

Spring框架提供了以下四种方式来管理bean的生命周期事件：

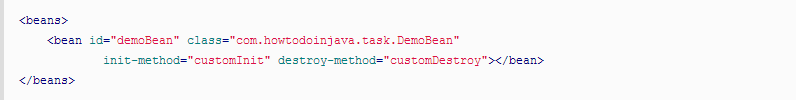
InitializingBean和DisposableBean回调接口

针对特殊行为的其他Aware接口

Bean配置文件中的Custom init()方法和destroy()方法

@PostConstruct和@PreDestroy注解方式

使用customInit()和 customDestroy()方法管理bean生命周期的代码样例如下：



### 55.(★★)Spring框架中都用到了哪些设计模式？

1. 简单工厂

又叫做静态工厂方法（StaticFactory Method）模式，但不属于23种GOF设计模式之一。

简单工厂模式的实质是由一个工厂类根据传入的参数，动态决定应该创建哪一个产品类，Spring中的BeanFactory就是简单工厂模式的体现。

2. 工厂方法（Factory Method）

定义一个用于创建对象的接口，让子类决定实例化哪一个类。Factory Method使一个类的实例化延迟到其子类。Spring中的FactoryBean就是典型的工厂方法模式。

3. 单例（Singleton）

保证一个类仅有一个实例，并提供一个访问它的全局访问点。

Spring中的单例模式完成了后半句话，即提供了全局的访问点BeanFactory。但没有从构造器级别去控制单例，这是因为Spring管理的是是任意的Java对象。

4. 适配器（Adapter）

将一个类的接口转换成客户希望的另外一个接口。Adapter模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

5.包装器（Decorator）

动态地给一个对象添加一些额外的职责。就增加功能来说，Decorator模式相比生成子类更为灵活。

Spring中用到的包装器模式在类名上有两种表现：一种是类名中含有Wrapper，另一种是类名中含有Decorator。基本上都是动态地给一个对象添加一些额外的职责。

6. 代理（Proxy）

为其他对象提供一种代理以控制对这个对象的访问。从结构上来看和Decorator模式类似，但Proxy是控制，更像是一种对功能的限制，而Decorator是增加职责。

Spring的Proxy模式在aop中有体现，比如JdkDynamicAopProxy和Cglib2AopProxy。

7.观察者（Observer）

定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

Spring中Observer模式常用的地方是listener的实现。如ApplicationListener。

8. 策略（Strategy）

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。本模式使得算法可独立于使用它的客户而变化。

9.模板方法（Template Method）

定义一个操作中的算法的骨架，而将一些步骤延迟到子类中。Template Method使得子类可以不改变一个算法的结构即可重定义该算法的某些特定步骤，Template Method模式一般是需要继承的。

### 56.(★★)请举例解释@Autowired注释

@Autowired注解对自动装配何时何处被实现提供了更多细粒度的控制。@Autowired注解可以像@Required注解、构造器一样被用于在bean的设值方法上自动装配bean的属性，一个参数或者带有任意名称或带有多个参数的方法。

比如，可以在设值方法上使用@Autowired注解来替代配置文件中的 <property>元素。当Spring容器在setter方法上找到@Autowired注解时，会尝试用byType 自动装配。

当然我们也可以在构造方法上使用@Autowired 注解。带有@Autowired 注解的构造方法意味着在创建一个bean时将会被自动装配，即便在配置文件中使用<constructor-arg> 元素。



下面是没有构造参数的配置方式：



### 57.(★★)如何向Spring Bean中注入一个java.util.Properties?

第一种方法是使用如下面代码所示的<props> 标签：



也可用”util:”命名空间来从properties文件中创建出一个propertiesbean，然后利用setter方法注入bean的引用。

### 58.(★★★)谈谈如何提高java web server并发性能。

1. 对Collection、Map接口的类对象初始化时要先分配合理的空间大小，同时还要按照自已的实际需求选择合适的对象。
2. 优化循环体

循环是比较重复运行的地方，如果循环次数很大，循环体内不好的代码对效率的影响就会被放大而变的突出。

1. 少用new初始化一个实例

尽量少用new来初始化一个类的实例，当一个对象是用new进行初始化时，其构造函数链的所有构造函数都被调用到，所以new操作符是很消耗系统资源的，new一个对象耗时往往是局部变量赋值耗时的上千倍。同时，当生成对象后，系统还要花时间进行垃圾回收和处理。当new创建对象不可避免时，注意避免多次的使用new初始化一个对象。尽量在使用时再创建该对象，另外，应该尽量重复使用一个对象，而不是声明新的同类对象。一个重用对象的方法是改变对象的值，如可以通过setValue之类的方法改变对象的变量达到重用的目的。

1. 选择合适的方法调用：

在Java中，一切都是对象，如果有方法（Method）调用，处理器先要检查该方法是属于哪个对象，该对象是否有效，对象属于什么类型，然后选择合适的方法并调用。可以减少方法的调用，不影响可读性等情况下，可以把几个小的方法合成一个大的方法。另外，在方法前加上final，private关键字有利于编译器的优化。

1. 异常处理技巧

异常是Java的一种错误处理机制，对程序来说是非常有用的，但是异常对性能不利。抛出异常首先要创建一个新的对象，并进行相关的处理，造成系统的开销，所以异常应该用在错误处理的情况，不应该用来控制程序流程，流程尽量用while，if等处理。在不是很影响代码健壮性的前提下，可以把几个try/catch块合成一个。

1. 尽量使用局部变量

尽量使用局部变量，调用方法时传递的参数以及在调用中创建的临时变量都保存在栈（Stack） 中，速度较快。其他变量，如静态变量、实例变量等，都在堆（Heap）中创建，速度较慢。

1. 同步处理技巧

同步主要出现在多线程的情况，为多线程同时运行时提供对象数据安全的机制，多线程是比较复杂话题，应用多线程也是为了获得性能的提升，应该尽可能减少同步。

另外，如果需要同步的地方，可以减少同步的代码段，如只同步某个方法或函数，而不是整个代码。

1. 尽可能的使用Java自身提供的API

Java的API一般都做了性能的考虑，如果完成相同的功能，优先使用API而不是自己写的代码，如数组复制。

1. 尽量减少I/O操作

输入/输出（I/O）包括很多方面，我们知道，进行I/O操作是很消耗系统资源的。程序中应该尽量少用I/O操作。使用时可以注意： . 合理控制输出函数System.out.println()对于大多时候是有用的，特别是系统调试的时候，但也会产生大量的信息出现在控制台和日志上，同时输出时，有序列化和同步的过程，造成了开销。

特别是在发行版中，要合理的控制输出，可以在项目开发时，设计好一个Debug的工具类，在该类中可以实现输出开关，输出的级别，根据不同的情况进行不同的输出的控制。

1. 尽量使用缓存

读写内存要比读写硬盘上的文件要快很多，应尽可能使用缓冲，以便直接从内存中读取数据。尽可能使用带有Buffer的类代替没有Buffer的类，如可以用BufferedReader 代替Reader，用BufferedWriter代替Writer来进行处理I/O操作。

同样可以用BufferedInputStream代替InputStream都可以获得性能的提高

1. 尽量不使用同步：

Servlet是多线程的，以处理不同的请求，基于前面同步的分析，如果有太多的同步就失去了多线程的优势了。

1. 不用保存太多的信息在HttpSession中

很多时候，存储一些对象在HttpSession中是有必要的，可以加快系统的开发，如网上商店系统会把购物车信息保存在该用户的Session中，但当存储大量的信息或是大的对象在会话中时，是有害的，特别是当系统中用户的访问量很大，对内存的需求就会很高。具体开发时，在这两者之间应作好权衡。

1. 清除SESSION：

通常情况，当达到设定的超时时间时，同时有些Session没有了活动，服务器会释放这些没有活动的Session，.. 不过这种情况下，特别是多用户并访时，系统内存要维护多个的无效Session。当用户退出时，应该手动释放，回收资源，实现如下：..HttpSession theSession = request.getSession();// 获取当前Sessionif(theSession != null){theSession.invalidate(); // 使该Session失效}

1. 缓存Home接口

EJB库使用Enterprise Bean 的客户端通过它的Home接口创建它的实例。客户端能通过JNDI访问它。服务器通过Lookup方法来获取。JNDI是个远程对象，通过RMI方式调用，对它的访问往往是比较费时的。所以，在设计时可以设计一个类专门用来缓存Home接口，在系统初始化时就获得需要的Home接口并缓存，以后的引用只要引用缓存即可。

1. 使用快速度的Jdbc驱动

JDBC API包括两种实现接口形式，一种是纯Java实现的驱动，一种利用ODBC驱动和数据库客户端实现，具体有四种驱动模式：

第一类：JDBC-ODBC桥，再加上ODBC驱动程序。JDBC驱动程序是JDBC-ODBC桥再加上一个ODBC驱动程序。建议第一类驱动程序只用于原型开发，而不要用于正式的运行环境。桥接驱动程序由Sun提供，它的目标是支持传统的数据库系统。Sun为该软件提供关键问题的补丁，但不为该软件的最终用户提供支持。一般地，桥接驱动程序用于已经在ODBC技术上投资的情形，例如已经投资了Windows应用服务器。尽管Sun提供了JDBC-ODBC桥接驱动程序，但由于ODBC会在客户端装载二进制代码和数据库客户端代码，这种技术不适用于高事务性的环境。另外，第一类JDBC驱动程序不支持完整的Java命令集，而是局限于ODBC驱动程序的功能，这种驱动方式也叫胖客户，主要用于低并发请求，大数据量传输的应用。

第二类：本机API，部分是Java的驱动程序。JDBC驱动程序是本机API的部分Java代码的驱动程序，用于把JDBC调用转换成主流数据库API的本机调用。这类驱动程序也存在与第一类驱动程序一样的性能问题，即客户端载入二进制代码的问题，而且它们被绑定了特定的平台。第二类驱动程序要求编写面向特定平台的代码，主流的数据库厂商，例如Oracle和IBM，都为它们的企业数据库平台提供了第二类驱动程序，使用这些驱动程序的开发者必须及时跟进不同数据库厂商针对不同操作系统发行的各个驱动程序版本。另外，由于第二类驱动程序没有使用纯Java的API，把Java应用连接到数据源时，往往必须执行一些额外的配置工作。很多时候，第二类驱动程序不能在体系结构上与大型主机的数据源兼容；即使做到了兼容，效果也是比较差。

第三类：面向数据库中间件的纯Java驱动程序。JDBC驱动程序是面向数据库中间件的纯Java驱动程序，JDBC调用被转换成一种中间件厂商的协议，中间件再把这些调用转换到数据库API。第三类JDBC驱动程序的优点是它以服务器为基础，也就是不再需要客户端的本机代码，这使第三类驱动程序要比第一、二两类快。另外，开发者还可以利用单一的驱动程序连接到多种数据库。

第四类：直接面向数据库的纯Java驱动程序。JDBC驱动程序是直接面向数据库的纯Java驱动程序，即所谓的“瘦”（thin）驱动程序，它把JDBC调用转换成某种直接可被DBMS使用的网络协议，这样，客户机和应用服务器可以直接调用DBMS服务器。对于第四类驱动程序，不同DBMS的驱动程序不同。因此，在一个异构计算环境中，驱动程序的数量可能会比较多。但是，由于第四类驱动程序具有较高的性能，能够直接访问DBMS，所以这一问题就不那么突出了， 这种驱动方式，主要用于高并发，低数据量请求的应用中。

1. 使用Jdbc链接池

为了提高访问数据库的性能，我们还可以使用JDBC 2.0的一些规范和特性，JDBC是占用资源的，在使用数据库连接时可以使用连接池Connection Pooling，避免频繁打开、关闭Connection。而我们知道，获取Connection是比较消耗系统资源的。Connection缓冲池：当一个应用程序关闭一个数据库连接时，这个连接并不真正释放而是被循环利用，建立连接是消耗较大的操作，循环利用连接可以显著的提高性能，因为可以减少新连接的建立。

一个通过DataSource获取缓冲池获得连接，并连接到一个CustomerDB数据源的代码演示如下：

Context ctx = new InitialContext();

DataSource dataSource = (DataSource) ctx.lookup(“jdbc/CustomerDB”);

Connection conn = dataSource.getConnection(“password”,”username”);

1. 缓存DataSorce

一个DataSource对象代表一个实际的数据源。这个数据源可以是从关系数据库到表格形式的文件，完全依赖于它是怎样实现的，一个数据源对象注册到JNDI名字服务后，应用程序就可以从JNDI服务器上取得该对象，并使用之和数据源建立连接。通过上面的例子，我们知道DataSource是从连接池获得连接的一种方式，通过JNDI方式获得，是占用资源的。为了避免再次的JNDI调用，可以系统中缓存要使用的DataSource。

1. 即时关闭使用过的资源

互联网应用系统一般是并发的系统，在每次申请和使用完资源后，应该释放供别人使用，使用完成后应该保证彻底的释放。

1. 架构选型

CoreMediaCMS将整个应用分成四成架构，每一层都可以独立于其他层而正常运行，每一层都可以分布式布署，极大的提高了应用系统的稳定性、可扩展性、支持高并发的要求，每一次之前通过中间件Corba进行稳定的传输数据。

1. 开发框架的选型

充分利用开源框架，可以大大提高开发效率。很多初级开发者，都采用DB JavaBean JSP这种初级的开发模式，而现在主要使用Struts、Spring等MVC开发框架。

常用开发框架构选型有：Struts、Spring、Webwork等。

1. 分级存储
2. 数据库数据分级存储：

将经常访问的数据和访问频度低的数据，分别存放到不同的分区，甚至存放到不同的数据库服务器，以便合进分配硬盘I/O及系统I/O。

2）网站内容发布之后，分级存储：

任何一个大型的网站，一般都有海量的内容，为了提高访问效率，应搭建分级存储体系，根据应用的重要性和访问并发要求，将这些内容分级存储，同时将静态内容中的静态页面文件、图片文件、下载文件分不同的Web服务器访问，降低I/O争用，提高访问效率，同时让数据存储、管理、备份更加清晰。

1. 页面静态化

一个大型网站，既有静态内容，也有动态内容。静态内容，直接通过Apache或者Squid访问，效率高，稳定可靠，更多的是受服务器等硬件设备的I/O吞吐量、网络环境及页面代码本身质量限制，不受应用系统及数据库性能限制，这些内容往往访问速度和效率不会有较大的问题。

而动态内容，除了受硬件设备I/O、操作系统I/O及内容、网络环境及页面代码的影响，还要受应用服务器和数据库性能影响，因此，这部份内容，要尽可能作静态化或者伪静态，并采用缓存技术，将其缓存，以减少对应用服务器和数据库服务器的操作次数，提高用户访问效率和稳定性。

1. 缓存策略

对于构建的业务系统，如果有些数据要经常要从数据库中读取，同时，这些数据又不经常变化，这些数据就可以在系统中缓存起来，使用时直接读取缓存，而不用频繁的访问数据库读取数据。缓存工作可以在系统初始化时一次性读取数据，特别是一些只读的数据，当数据更新时更新数据库内容，同时更新缓存的数据值。

## 2.HTML

### 1. （★★★）说明HTML文档中DTD的意义和作用(酷讯)

答:

DTD，文档类型定义，是一种保证 html 文档格式正确的有效方法，在解析网页时，浏览器将使用 DTD 来检查页面的有效性（是否符合规范，元素和标签使用是否正确）并且采取相应的措施。同时它还会影响浏览器的渲染模式（工作模式）。

### 2. （★★）准确的指出以下代码的显示结果（酷讯）

<style> a{ position:relative; height:200px; width:500px; border:solid 1px #000; background:#FFF;

}

#b,#c{position:absolute; width:250px; height:90px;}

#b{top:30px; left:50px; background:#FF0000; z-index:1;}

#c{bottom:30px; right:50px; background:#0000FF;}

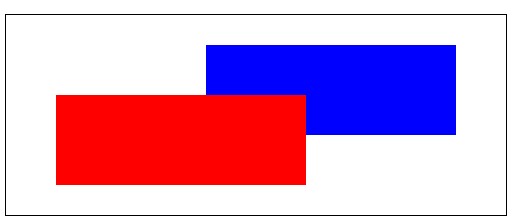
</style>

<div class="a">

<div id="b"></div> <div id="c"></div>

</div>

答:

 结果如图所示：

### 3. （★★）指出以下代码的显示效果

<table border=1 width=500 style="text-align:center;">

<tr>

<td rowspan=2 width=50% height=50>a</td>

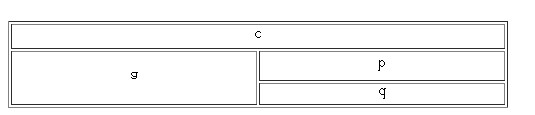
<td width=50% eight=25>d</td>

</tr>

<tr><td width=50% height=25>b</td></tr>

<tr height=25><td colspan=2>c</td></tr>

</table>

答:

### 4. （★）如何显示没有下划线的超链接？（）

A. a {text-decoration:none}

B. a {text-decoration:no underline}

C. a {underline:none}

D. a {decoration:no underline}

答:

A

### 5. （★）请问, div和span, p标签三者间的区别(YG)

答:

从语义化的角度来讲，div和span标签对于它们包含的元素是没有意义的，而p标签则是

一个语义化标签，表示一个段落。所以div 和 span 一般是用来布局。

从显示的角度来讲，span 是内联的，用在一小块的内联 HTML 中，前后不断行，div 元素是块级的，等同于其前后有断行，p 元素是块级的，前后断行，而且还要再隔一行，相当于断两行。

### 6. （★）CSS中class和id的区别是?

答:

id：每个页 ID 是唯一的不能重复。 class：可以设置多个 class 为同一个值。

### 7. （★）实现框架的标签 举例说明

答:

<frameset cols="25%,50%,25%">

<frame src="frame\_a.htm" /> <frame src="frame\_b.htm" />

<frame src="frame\_c.htm" />

</frameset>

### 8. （★）写出以下 html 标签的含义：input form script style table b img

答:

<input /> 标签用于搜集用户信息。根据不同的 type 属性值，输入字段拥有很多种形式。输入字段可以是文本字段、复选框、掩码后的文本控件、单选按钮、按钮等等。

<form> 标签用于为用户输入创建 HTML 表单。表单能够包含 input 元素，比如文本字段、复选框、单选框、提交按钮等等。

<script> 标签用于定义客户端脚本，比如 JavaScript。script 元素既可以包含脚本语句，也可以通过 src 属性指向外部脚本文件。

<style> 标签用于为 HTML 文档定义样式信息。

<table> 标签定义 HTML 表格。简单的 HTML 表格由 table 元素以及一个或多个 tr、 th 或 td 元素组成。

<b>呈现粗体文本效果。

<img /> 元素向网页中嵌入一幅图像。

### 9. （★★）CSS中 link 和@import 的区别是？

答:

(1) link属于HTML标签，而@import是CSS提供的;

(2) 页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;

(3) import只在IE5以上才能识别，而link是HTML标签，无兼容问题;

(4) link方式的样式的权重高于@import的权重.

### 10. （★★） 用 css、html 编写一个两列布局的网页，右侧固定宽度 200px，左侧自适应

答:

方案1:



### 11. （★）看以下代码：

文件名称：forward.jsp

<html>

<head><title> 跳转 </title> </head>

<body>

<jsp:forward page="index.htm"/>

</body>

</html>

如果运行以上jsp文件，地址栏的内容为

A.http://127.0.0.1:8080/myjsp/forward.jsp

B.http://127.0.0.1:8080/myjsp/index.jsp

C.http://127.0.0.1:8080/myjsp/index.htm

D.http://127.0.0.1:8080/myjsp/forward.htm

答:

A

### 12. （★）基于以下 HTML 结构，以下关于 main1.css 和 main2.css 的描述有哪些是正确的？

A.main1.css和main2.css同时开始加载，先加载完成的优先解析

B.如果main1.css和main2.css中有相同的选择器规则，那么main2.css中的规则将覆盖 main1.css的规则

C.main2.css只有在main1.css加载并解析后，才开始加载

D.如果main1.css和main2.css中有相同的选择器规则，那么main2.css中的规则将被忽略

答:

A

### 13. （★）下列html语义化标签中,描述存在错误的是

A.<em>强调文字，<blockquote>长文本引用

B.<strong>强调文字，<h1>文章栏目标题

C.<a>超链接，<p>文章段落

D.<section>定义文档中的节，<nav>导航

E.<ul>有序列表。<ol>无序列表

答:

E

<ul>无序列表,<ol>有序列表

### 14. （★）设置文字颜色应使用属性

A.font-family B.font-color

C.color D.font

答:

C

### 15. （★）以下标签，错误的是

A.text-left用于左对齐

B.text-center水平居中对齐

C.text-vertical垂直居中对齐

D.text-uppercase可以将文字全部大写

答:

C

### 16. （★★）有一个元素，需要在pc端显示而在手机端隐藏，需要使用的方法是？

A.visible-xs-8 hidden-md B.visible-md-8 hidden-xs

C.visible-md-8 hidden-sm D.visible-sm-8 hidden-md

答:

B

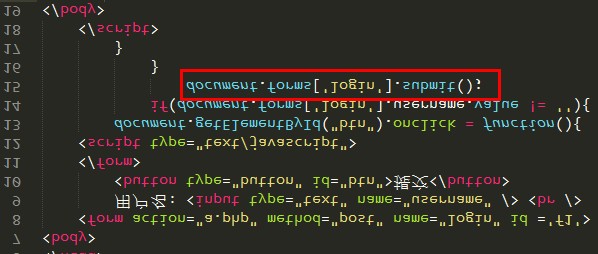
### 17.(★)列出HTML里的input标签的type属性有那些值？

type 属性取值:  
text -- 文字输入域(输入型)  
password -- 也是文字输入域,但是输入的文字以密码符号'\*'显示(输入型)  
file -- 可以输入一个文件路径(输入型)  
checkbox -- 复选框.可以选择零个或多个(选择型)  
radio -- 单选框.只可以选择一个而且必须选择一个(选择型)  
hidden -- 代表隐藏域,可以传送一些隐藏的信息到服务器  
button -- 按钮(点击型)  
image -- 使用图片来显示按钮,使用src属性指定图像的位置(就像img标签的src属性)(点击型)  
submit -- 提交按钮,表单填写完毕可以提交,把信息传送到服务器.可以使用value属性来显示按钮上的文字(点击型)  
reset -- 重置按钮,可以把表单中的信息清空(点击型)

## 3.Javascript

### 1. （★）解释下面语句的意思：document.form["formName"].submit。

答:

获取 name 为 formName 的表单并将该表单数据提交到服务器。但这行代码是不能运行的，有两处错误，一是通过表单名称来获取表单得到时候，应该写成 document.forms[‘formName’]，提交表单事件使用 submit 方法，需要加上括号.

### 2. （★★）简单说一下浏览器本地存储是怎样的

答:

在较高版本的浏览器中，js提供了sessionStorage和globalStorage。在HTML5中提供了localStorage来取代globalStorage。

html5中的Web Storage包括了两种存储方式：sessionStorage和localStorage。

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。

而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

### 3. （★）javascript中document.getElemenById的返回值的类型为（）

A.Array B.Object C.String D.Function

答:

B

### 4. （★）下面给定的代码中， myFunc() 返回什么？

var foo=’foo’;

var bar=’bar’;

function myFunc() {

return foo+bar;

}

1. undefined B.foobar C.foo+bar D.NaN

答:

C

### 5. （★）哪一个是javascript中array的正确写法？

A.var txt = new Array("tim","kim","jim")

B.var txt = new Array="tim","kim","jim"

C.var txt = new Array:1=("tim")2=("kim")3=("jim")

D.var txt = new Array(1:"tim",2:"kim",3:"jim")

答:

A

### 6. （★）下面程序的显示结果是？

A.hi B.hi hello C.hello D.不显示

答:

A

### 7. （★）请阅读以下代码：问题：span标签的width和height分别为多少？

A.width = 0px，height = 0px

B.width = 400px，height = 200px

C.width = 100px，height = 50px

D.width = 0px，height = 200px

答:

D

### 8. （★）JS 表单弹出对话框函数是?获得输入焦点函数是?

答:

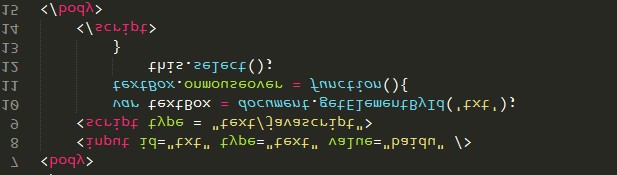
弹出对话框函数：alert(), prompt(), confirm() 获得输入焦点函数：focus()

### 9. （★）JS 的转向函数是?怎么引入一个外部 JS 文件?

答:

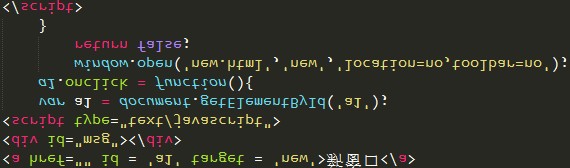
转向使用 window.location.href = "" 引入外部 js 使用 <script src=""/>

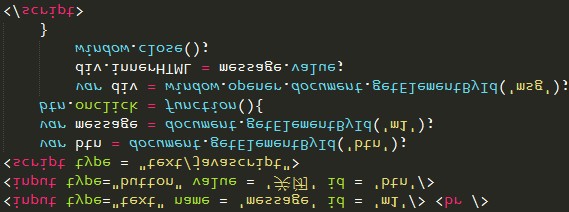
### 10. （★）有下面语句：<input id="txt" type="text" value="baidu" /> 编写代码，当鼠标划过文本框，自动选中文本框中的内容。

答:

### 11. （★★）设计一个网页，使得打开它时弹出一个全屏的窗口，该窗口中有一个文本框和一个按钮。用户在文本框中输入信息后点击按钮就可以把窗口关闭，而输入的信息却在主网页中显示。

答:

主窗口：

弹出窗口：

### 12. （★）判断以下代码是否正确，如果有错，请指出错误，如果正确，请指出运行结果

var arr = new Array(

new Array(1,2,3,4),

new Array("abc", "def", "xyz"),

);

for(i = 0; i < arr.length; i++) { document.write(arr[0])

}

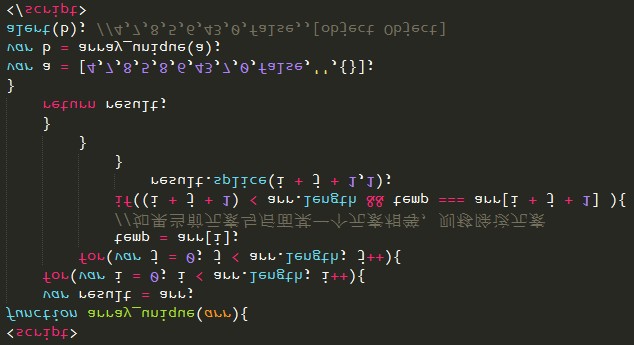
答:

怀疑这个代码写错了，第二个 new array 后面的逗号是多余的，如果此处没问题，则代码是正确的，输出的结果是 1,2,3,4 1,2,3,4

注：arr 是一个二维数组，该数组有两个元素，第一个元素是数组[1,2,3,4]，第二个元素也是一个数组为["abc", "def", "xyz"]，for 循环语句执行了两次，但都是输出第一个元素，即数组[1,2,3,4]。

### 13. （★★）用 JavaScript 写一个函数,其功能是可以删除数组中重复的元素。

答:



### 14. （★）以下哪条 JavaScript 语句会产生运行错误：（ ）

var obj = ( );

var obj = [ ];

var obj = { };

D. var obj = //;

答：

A

### 15. （★）请选择结果为真的表达式：（ ）

null instanceof Object

null === undefined

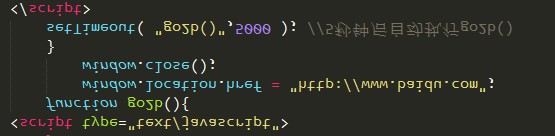
null == undefined

NaN == NaN

答：

C

### 16. （★★）假设 a.html和 b.html 在同一个文件夹下面，用 JavaScript 实现当打开 a.html 五秒钟后，自动跳转到 b.html。

答：

### 17. （★★）请使用 JavaScript 写出三种产生一个 image 标签的方法（提示：从方法、对象、HTML 角度考虑）

答：

(1). var img = new Image();

(2). var img = document.createElement("image")

(3). img.innerHTML = "<img src='xxx.jpg' />"

### 18. （★★）js 中网页前进和后退的代码

答：

前进: history.forward();或者 history.go(1); 后退: history.back (); 或者 history.go(-1);

实现点击按钮弹出窗口的代码 (YG)

document.getElementById( 'button' ).onclick = function(){ window.open ('page.html');

}

其中，button 是按钮的 id，page.html 是要弹出的窗口页面。

JavaScript 包括那些基本数据类型？（小米）

JavaScript 中包括 5 种基本数据类型，分别是 Number，String，Boolean，Null 和 Undefined。

### 21.（★）javascript 常用的方面；

答：

常用于数据输入校验和页面特殊效果等。

### 22.（★★）javascript 的优缺点和内置对象；

答：

1）优点：简单易用，与Java 有类似的语法，可以使用任何文本编辑工具编写，只需要浏览器就可执行程序，并且事先不用编译，逐行执行，无需进行严格的变量声明，而且内置大量现成对象，编写少量程序可以完成目标；

2）缺点：不适合开发大型应用程序；

3）Javascript 有11 种内置对象：Array、String、Date、Math、Boolean、Number、Function、Global、Error、RegExp、Object

### 23.(★★)比较一下Java和JavaSciprt

答：JavaScript 与Java是两个公司开发的不同的两个产品。Java 是原Sun 公司推出的面向对象的程序设计语言，特别适合于互联网应用程序开发；而JavaScript是Netscape公司的产品，为了扩展Netscape浏览器的功能而开发的一种可以嵌入Web页面中运行的基于对象和事件驱动的解释性语言，它的前身是LiveScript；而Java 的前身是Oak语言。

下面对两种语言间的异同作如下比较：

1）基于对象和面向对象：Java是一种真正的面向对象的语言，即使是开发简单的程序，必须设计对象；JavaScript是种脚本语言，它可以用来制作与网络无关的，与用户交互作用的复杂软件。它是一种基于对象（Object-Based）和事件驱动（Event-Driven）的编程语言。因而它本身提供了非常丰富的内部对象供设计人员使用；

2）解释和编译：Java 的源代码在执行之前，必须经过编译；JavaScript 是一种解释性编程语言，其源代码不需经过编译，由浏览器解释执行；

3）强类型变量和类型弱变量：Java采用强类型变量检查，即所有变量在编译之前必须作声明；JavaScript中变量声明，采用其弱类型。即变量在使用前不需作声明，而是解释器在运行时检查其数据类型；

4）代码格式不一样。

## 4.JSP

### 1. （★）JSP 中动态INCLUDE 与静态INCLUDE 的区别？

答：

动态INCLUDE 用jsp:include 动作实现<jsp:include page="included.jsp"flush="true" />它总是会检查所含文件中的变化，适合用于包含动态页面，并且可以带参数；静态INCLUDE 用include 伪码实现,它不会检查所含文件的变化，适用于包含静态页面<%@ include file="included.htm" %>

### 2. （★）JSP 的内置对象及方法？

答：

request 表示HttpServletRequest 对象。它包含了有关浏览器请求的信息，并且提供了几个用于获取cookie, header 和session 数据的有用的方法。response 表示HttpServletResponse 对象，并提供了几个用于设置送回浏览器的响应的方法（如cookies,头信息等）。out 对象是javax.jsp.JspWriter 的一个实例，并提供了几个方法使你能用于向浏览器回送输出结果。pageContext 表示一个javax.servlet.jsp.PageContext 对象。它是用于方便存取各种范围的名字空间、servlet 相关的对象的API，并且包装了通用的servlet 相关功能的方法。session 表示一个请求的javax.servlet.http.HttpSession 对象。Session可以存贮用户的状态信息。application 表示一个javax.servle.ServletContext 对象。这有助于查找有关servlet 引擎和servlet 环境的信息。config 表示一个javax.servlet.ServletConfig 对象。该对象用于存取servlet 实例的初始化参数。page 表示从该页面产生的一个servlet 实例。

### 3. （★）JSP 的常用指令？

答：

<%@page language=”java” contenType=”text/html;charset=gb2312”

session=”true” buffer=”64kb” autoFlush=”true” isThreadSafe=”true”

info=”text” errorPage=”error.jsp” isErrorPage=”true” isELIgnored=”

true” pageEncoding=”gb2312” import=”java.sql.\*”%>

isErrorPage：是否能使用Exception 对象；isELIgnored：是否忽略EL 表达式；

<%@include file=”filename”%>

<%@taglib prefix=”c”uri=”http://……”%>

### （★）jsp 有哪些动作?作用分别是什么?

答：

JSP 共有以下6 种基本动作：

jsp:include：在页面被请求的时候引入一个文件；

jsp:useBean：寻找或者实例化一个JavaBean。；

jsp:setProperty：设置JavaBean 的属性。；

jsp:getProperty：输出某个JavaBean 的属性；

jsp:forward：把请求转到一个新的页面；

jsp:plugin：根据浏览器类型为Java 插件生成OBJECT 或EMBED 标记。

### （★）jsp 有哪些内置对象?作用分别是什么?

答：

JSP 共有以下9 种基本内置组件（可与ASP 的6 种内部组件相对应）：

request：用户端请求，此请求会包含来自GET/POST 请求的参数；

response：网页传回用户端的回应；

pageContext：网页的属性是在这里管理；

session：与请求有关的会话期；

application：servlet 正在执行的内容；

out：用来传送回应的输出；

config：servlet 的构架部件；

page：JSP 网页本身；

exception：针对错误网页，未捕捉的例外。

### 6. （★）JSP 和Servlet 有哪些相同点和不同点，他们之间的联系是什么？

答：

JSP 是Servlet 技术的扩展，本质上是Servlet 的简易方式，更强调应用的外表表达。JSP 编译后是"类servlet"。Servlet 和JSP 最主要的不同点在于，Servlet 的应用逻辑是在Java 文件中，并且完全从表示层中的HTML 里分离开来。而JSP 的情况是Java 和HTML 可以组合成一个扩展名为.jsp 的文件。JSP 侧重于视图，Servlet 主要用于控制逻辑。

### 7. （★）jsp 的四种范围？

答：

a.page 是代表与一个页面相关的对象和属性。一个页面由一个编译好的Java servlet 类（可以带有任何的include 指令，但是没有include 动作）表示。这既包括servlet 又包括被编译成servlet 的JSP 页面b.request 是代表与Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个Web 组件（由于forward 指令和include 动作的关系）c.session 是代表与用于某个Web 客户机的一个用户体验相关的对象和属性。一个Web 会话可以也经常会跨越多个客户机请求d.application 是代表与整个Web 应用程序相关的对象和属性。这实质上是跨越整个Web 应用程序，包括多个页面、请求和会话的一个全局作用域。

### 8. （★）JSP 和Servlet 中的请求转发分别如何实现？

答：

JSP 中的请求转发可利用forward 动作实现：<jsp:forward />；Serlvet 中实现请求转发的方式为：

getServletContext().getRequestDispatcher(path).forward(req,res)

### 9. （★）介绍在JSP 中如何使用JavaBeans？

答：

在JSP 中使用JavaBean 常用的动作有：

1）<jsp:useBean />：用来创建和查找bean 对象；

2）<jsp:setProperty />：用来设置bean 的属性，即调用其setXxx()方法；

3）<jsp:getProperty />：用来获得bean 的属性，即调用其getXxx()方法

### 10. （★★） JSP 标签的作用？如何定义？

答：

作用：

分离jsp 页面的内容和逻辑；业务逻辑开发者可以创建自定义标签；

封装业务逻辑；可重用并且易维护；易于手工修改、易于工具维护；

提供简洁的语法；

定义:

写标签处理器；写tld 文件；将标签处理器和tld 文件放到同一个包里面；

把jsp 页面和标签库配置部署在一起

### 11. （★★）什么是JSP动作(JSP action)？

答：

JSP动作以XML语法的结构来控制Servlet引擎的行为。当JSP页面被请求的时候，JSP动作会被执行。它们可以被动态的插入到文件中，重用JavaBean组件，转发用户到其他的页面，或者是给Java插件产生HTML代码。下面列出了可用的动作：

jsp:include-当JSP页面被请求的时候包含一个文件。

jsp:useBean-找出或者是初始化Javabean。

jsp:setProperty-设置JavaBean的属性。

jsp:getProperty-获取JavaBean的属性。

jsp:forward-把请求转发到新的页面。

jsp:plugin-产生特定浏览器的代码。

### 12. （★）什么是JSP指令(Directive)？JSP中有哪些不同类型的指令？

答：

Directive是当JSP页面被编译成Servlet的时候，JSP引擎要处理的指令。Directive用来设置页面级别的指令，从外部文件插入数据，指定自定义的标签库。Directive是定义在 <%@ 和 %>之间的。下面列出了不同类型的Directive：

包含指令(Include directive)：用来包含文件和合并文件内容到当前的页面。

页面指令(Page directive)：用来定义JSP页面中特定的属性，比如错误页面和缓冲区。

Taglib指令： 用来声明页面中使用的自定义的标签库

### 13. （★★）JSP有什么优点？

答：

下面列出了使用JSP的优点：

JSP页面是被动态编译成Servlet的，因此，开发者可以很容易的更新展现代码。

JSP页面可以被预编译。

JSP页面可以很容易的和静态模板结合，包括：HTML或者XML，也可以很容易的和产生动态内容的代码结合起来。

开发者可以提供让页面设计者以类XML格式来访问的自定义的JSP标签库。

开发者可以在组件层做逻辑上的改变，而不需要编辑单独使用了应用层逻辑的页面。

### 14. （★）jsp有哪些内置对象？作用分别是什么？（至少三个）

答：

1、request对象 客户端请求，此请求会包含来自GET/POST请求的参数通过它才能了解到客户的需求，然后做出响应。

2、response对象 响应客户请求的有关信息

3、session对象 它指的是客户端与服务器的一次会话，从客户端连到服务器的一个 WebApplication开始，直到客户端与服务 器断开连接为止。

4、out对象 它是JspWriter类的实例,是向客户端输出内容常用的对象

5、page对象 它是指向当前JSP页面本身，有点象类中的this指针，它是 Java.lang.Object类的实例

6、application对象 它实现了用户间数据的共享，可存放全局变量。它开始于服务器的启动，直到服务器的关闭

7、exception对象 它是一个例外对象，当一个页面在运行过程中发生了例外，就产生这个对象。

8、pageContext对象 它提供了对JSP页面内所有的对象及名字空间的访问

9、config对象 它是在一个Servlet初始化时，JSP引擎向它传递信息用的

### 15. （★）下面有关jsp中静态include和动态include的区别，说法错误的是？

A.动态INCLUDE:用jsp:include动作实现

B.静态INCLUDE:用include伪码实现，定不会检查所含文件的变化，适用于包含静态页面<%@ include file="included.htm" %>

C.静态include的结果是把其他jsp引入当前jsp，两者合为一体;动态include的结构是两者独立，直到输出时才合并

D.静态include和动态include都可以允许变量同名的冲突.页面设置也可以借用主文件的

答: D

### 16.(★)下面那一个不能做JSP的服务器 D

A. IBM WebSphere B. BEA WebLogic C. Tomcat D.pws

## 5.Servlet

### 1. （★）如何知道是哪一个客户端的机器正在请求你的Servlet？

答：

ServletRequest类可以找出客户端机器的IP地址或者是主机名。

getRemoteAddr()方法获取客户端主机的IP地址，getRemoteHost()可以获取主机名。

### （★★）解释下Servlet的生命周期。

答：

对每一个客户端的请求，Servlet引擎载入Servlet，调用它的init()方法，完成Servlet的初始化。然后，Servlet对象通过为每一个请求单独调用service()方法来处理所有随后来自客户端的请求，最后，调用Servlet(译者注：这里应该是Servlet而不是server)的destroy()方法把Servlet删除掉。

### （★）什么是Servlet？

答：

Servlet是用来处理客户端请求并产生动态网页内容的Java类。Servlet主要是用来处理或者是存储HTML表单提交的数据，产生动态内容，在无状态的HTTP协议下管理状态信息。

### （★★）servlet中转发和重定向的区别?

答：

Servlet中页面的跳转有两种方式：转发和重定向。

1、转发和重定向的区别

①转发是服务器行为，重定向是客户端行为。

②转发是浏览器发生了一次请求，重定向至少是两次请求。

③转发地址栏中的url不会发生改变，重定向地址栏中的url会发生改变。

④转发因为是一次请求，两次跳转之间传输信息不会丢失，而重定向两次跳转会丢失传输信息。

⑤转发只能将请求跳转到同一个web应用中的页面，重定向可以指向任何页面，包括同一个web应用中的页面，同一个站点中的页面，其他站点的页面。

2、传递给HttpServlet

Response.sendRedirect 方法的相对URL以“/”开头，它是相对于整个WEB站点的根目录；如果创建RequestDispatcher 对象时指定的相对URL以“/”开头，它是相对于当前WEB应用程序的根目录。

### 5. （★）下面有关servlet service描述错误的是？

A.不管是post还是get方法提交过来的连接，都会在service中处理

B.doGet/doPost 则是在 javax.servlet.GenericServlet 中实现的

C.service()是在javax.servlet.Servlet接口中定义的

D.service判断请求类型，决定是调用doGet还是doPost方法

答: B

### 6. （★）下列有关Servlet的生命周期，说法不正确的是？

A.在创建自己的Servlet时候，应该在初始化方法init()方法中创建Servlet实例

B.在Servlet生命周期的服务阶段，执行service()方法，根据用户请求的方法，执行相应的doGet()或是doPost()方法

C.在销毁阶段，执行destroy()方法后会释放Servlet 占用的资源

D.destroy()方法仅执行一次，即在服务器停止且卸载Servlet时执行该方法

答: A

### 7. （★）下面有关servlet中init,service,destroy方法描述错误的是？

A.init()方法是servlet生命的起点。一旦加载了某个servlet，服务器将立即调用它的init()方法

B.service()方法处理客户机发出的所有请求

C.destroy()方法标志servlet生命周期的结束

D.servlet在多线程下使用了同步机制，因此，在并发编程下servlet是线程安全的

答: D

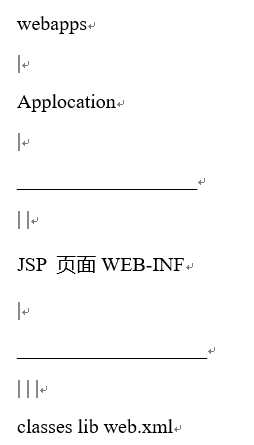
### 8. （★） 如何实现servlet 的单线程模式？

答：<%@page isThreadSafe=”false”%>

### 9. （★） 画出Servlet 2.2 以上Web Application 的基本目录结构（2 分钟）

答：

目录结构如下图所示：



### 10. （★）Servlet 执行时一般实现哪几个方法？

答：

public void init(ServletConfig config)

public ServletConfig getServletConfig()

public String getServletInfo()

public void service(ServletRequest request,ServletResponse

response)

public void destroy()

### 11. （★★）说出Servlet 的生命周期，并说出Servlet 和CGI 的区别?

答：

Web 容器加载Servlet 并将其实例化后，Servlet 生命周期开始，容器运行其init 方法进行Servlet 的初始化，请求到达时运行其service 方法，service方法自动派遣运行与请求对应的doXXX 方法（doGet，doPost）等，当服务器决定将实例销毁的时候调用其destroy 方法。与cgi 的区别在于servlet 处于服务器进程中，它通过多线程方式运行其service 方法，一个实例可以服务于多个请求，并且其实例一般不会销毁，而CGI 对每个请求都产生新的进程，服务完成后就销毁，所以效率上低于servlet。

### 12. （★★）Servlet 的基本架构

答：

public class ServletName extends HttpServlet {

public void doPost(HttpServletRequest request,HttpServletResponse response)

throws ServletException, IOException {}

public void doGet(HttpServletRequest request,HttpServletResponse response)

throws ServletException, IOException {}

}

### 13. （★★）简述Servlet的生命周期中的方法及作用。

答：

Servlet生命周期分为三个阶段：

　　(1)，初始化阶段 ：调用init()方法

　　(2)，响应客户请求阶段：调用service()方法

　　(3)，终止阶段：调用destroy()方法

生命周期：Servlet是一种可以在Servlet容器中运行的组件，那么理所当然就应该有一个从创建到销毁的过程，这个过程我们可以称之为Servlet生命周期。Servlet的生命周期可以分为加载、实例化、初始化、处理客户请求和卸载五个阶段，体现在方法上主要是init（）、service（）和destroy（）三个方法。生命周期的具体说明如下：

Servlet容器完成加载Servlet类和实例化一个Servlet对象；

init（）方法完成初始化工作，该方法由Servlet容器调用完成；

service（）方法处理客户端请求，并返回响应结果；

destroy（）方法在Servlet容器卸载Servlet之前被调用，释放一些资源。

### 14. （★★）简述forward和redirect的区别。

答：

(1).从地址栏显示来说

forward是服务器请求资源,服务器直接访问目标地址的URL,把那个URL的响应内容读取过来,然后把这些内容再发给浏览器.浏览器根本不知道服务器发送的内容从哪里来的,所以它的地址栏还是原来的地址.

redirect是服务端根据逻辑,发送一个状态码,告诉浏览器重新去请求那个地址.所以地址栏显示的是新的URL.

(2).从数据共享来说

forward:转发页面和转发到的页面可以共享request里面的数据.

redirect:不能共享数据.

(3).从运用地方来说

forward:一般用于用户登陆的时候,根据角色转发到相应的模块.

redirect:一般用于用户注销登陆时返回主页面和跳转到其它的网站等.

(4).从效率来说

forward:高.

redirect:低.

### 15. （★★）JSP和Servlet有哪些相同点和不同点，他们之间的联系是什么?

答：

(1).jsp经编译后就变成了Servlet.(JSP的本质就是Servlet，JVM只能识别java的类，不能识别JSP的代码,Web容器将JSP的代码编译成JVM能够识别的java类)

(2).jsp更擅长表现于页面显示,servlet更擅长于逻辑控制.

(3).Servlet中没有内置对象，Jsp中的内置对象都是必须通过HttpServletRequest对象，HttpServletResponse对象以及HttpServlet对象得到.

Jsp是Servlet的一种简化，使用Jsp只需要完成程序员需要输出到客户端的内容，Jsp中的Java脚本如何镶嵌到一个类中，由Jsp容器完成。而Servlet则是个完整的Java类，这个类的Service方法用于生成对客户端的响应。

联系： JSP是Servlet技术的扩展，本质上就是Servlet的简易方式。JSP编译后是“类servlet”。Servlet和JSP最主要的不同点在于，Servlet的应用逻辑是在Java文件中，并且完全从表示层中的HTML里分离开来。而JSP的情况是Java和HTML可以组合成一个扩展名为.jsp的文件。JSP侧重于视图，Servlet主要用于控制逻辑。

### 16.(★)在Servlet的生命周期中，容器只调用一次的方法是：CD

A.service B.getServletCongfig C.init D. destroy

### 17.(★)在JavaEE中，HTTPServetRequest对象由什么（B）创建的？

1. 由servlet容器创建，对于每个HTTP,Servlet容器都会创建一个HttpServletRequest对象
2. 由javaWeb应用组件的Servlet或Jsp最贱负责创建，当Servlet或Jsp最贱相应HTTp请求时，先创建HTTPServletRequest对象
3. 通过编码形式手动创建，以传递请求数据
4. 以上都不对

### 18.(★)在javax.servlet.http.\* API中，HttpServlet的（BC）方法用来处理客户端的请求。

1. Init()
2. doPost()
3. doGet()
4. destroy()

### 19.(★★)说明用Servlet机制实现页面跳转的方式？

Servlet跳转页面的方法:

跳转分为两部分，一部分发生在servlet，另一部分发生在JSP，JSP就是servlet，但是两者还是有部分差异。

Servlet：（跳转发生在doGet,doPost等方法里）

（1）redirect方式

response.sendRedirect(“/index.jsp”);页面的路径是相对路径。sendRedirect可以将页面跳转到任何页面，不一定局限于web应用中，如：response.sendRedirect（”http://www.baidu.com“）；

跳转后浏览器地址发生变化，只能在url总带parameter或者放在session中，无法使用request.setAttribute来传递值。

（2）forward方式

request.getRequestDispatcher(“/index.jsp”).forward(request,response);页面的路径是对路径。forward方式只能跳转到本web应用中的页面上。跳转后地址栏不会发生变化。使用这种方式跳转，传值可以使用三种方法：url中带parameter，session，request.setAttribute

JSP：

（1）response.sendRedirect();

和servlet的response.sendRedirect()方式一样

跳转后浏览器地址栏会变化，如果要跳转到不同主机下，跳转后，此语句后面的语句，如同新开了线程，但是对response的 操作已经没有意义了；如果要跳到相同主机下，此语句后面的语句执行完成后才会跳转；

（2）response.setHeader(“Location”,”“);

跳转后浏览器地址栏发生变化，此语句后面的语句执行完成后才能发生调转

（3）

跳转后浏览器地址栏不变，但是只能跳转到当前主机下，此语句后面的执行语句执行完成后才会跳转

请求转发RequestDispatcher.forward方法和响应（重定向）转发HttpServletResponse.sendRedirect方法

### 20.(★★)JavaWeb应用程序中，给定一个Servlet的doGet方法中的代码片段如下所示，那么在MyServlet中可以使用（D）语句把属性id的值取出来

Request.setAttribute(“id”,”3456-789-3353-77897”);

Response.sendRedirect(“http://lcoahost:8080/servlet/UserServlet”);

1. String str=request.getAttribute(“id”);
2. String str=(String)request.getAttribute(“id”);
3. Object str=request.getAttriubte(“id”);
4. 无法取得

### 21.(★★)什么是Servlet链（Servlet Chaining）?

Servlet链是把一个Servlet的输出发送给另一个Servlet的方法。第二个Servlet的输出可以发送给第三个Servlet，依次类推。链条上最后一个Servlet负责把响应发送给客户端。

## 6.struts1/struts2

### 1. （★）struts2如何解决重复提交问题?

答：

token 在struts配置文件中增加token拦截器。

### 2. （★★）struts2 国际化

答：

在struts2中是使用了拦截器来实现国际化。

struts2的国际化分为三部分：

1.Action范围，这种范围是用Action的实现类的类名来作为前缀\_\_语言名称\_国家地区名（大写）.properties.

2.包范围，包范围是用package作为前缀\_\_语言名称\_国家地区名（大写）.properties。

3.全局范围，全局范围的前缀默认的是application，可以通过xml配置文件配置常量(使用constant标签注册i18l拦截器)来指定前缀，前缀\_\_语言名称\_国家地区名（大写）.properties。

国际化实现的原理:通过i18n拦截器获得请求参数request——locale的值(zh或者是en)并把当前的值转化为locale对象，把locale对象存在ww\_trans\_i18n\_locale的session作用域中，根据struts2的请求处理流程，拦截器只会在进入action的时候才会调用，所有我们一般把对jsp的访问转化为对action的访问。

### 3. （★）什么是拦截器 说说Struts2用拦截器来干什么 说出6个拦截器来。

答：

在访问类的某个方法或者属性之前执行，拦截的是Action的请求，进行拦截然后在方法的执行前或者之后加入某些操作，

如：国际化，权限，效验等与源代码无关的操作。

国际化，表单重复提交，类型转换，自动装配，数据封装,异常处理，模型驱动，请求参数，处理类型转换错误，日志拦截器。

### 4. （★）在Struts框架中如果要使用Validation作验证的话，需要使用以下哪个Form？

A.ActionForm

B.ValidatorActionForm

C.ValidatorForm

D.DynaValidatorActionForm

答: D

### 5. （★）下面有关struts1和struts2的区别，描述错误的是？

A.Struts1要求Action类继承一个抽象基类。Struts 2 Action类可以实现一个Action接口

B.Struts1 Action对象为每一个请求产生一个实例。Struts2 Action是单例模式并且必须是线程安全的

C.Struts1 Action 依赖于Servlet API，Struts 2 Action不依赖于容器，允许Action脱离容器单独被测试

D.Struts1 整合了JSTL，Struts2可以使用JSTL，但是也支持OGNL

答: B

### （★）struts 的入口类？

答：

是ActionServlet，所有的struts 请求都经由该类转发处理。

### （★）STRUTS 的应用(如STRUTS 架构)？

答：

Struts 是采用Java Servlet/JavaServer Pages 技术开发Web 应用程序的开放源码的framework。采用Struts 能开发出基于MVC(Model-View-Controller)设计模式的应用构架。Struts 有如下的主要功能：

1)包含一个controller servlet，能将用户的请求发送到相应的Action 对象；

2)JSP 自由tag 库，并且在controller servlet 中提供关联支持，帮助开发人员创建交互式表单应用；

3)提供了一系列实用对象：XML 处理、通过Java reflection APIs 自动处理JavaBeans 属性、国际化的提示和消息

### 8. （★）说出struts 中的标签。

答：

<bean:message /> <html:errors />

<bean:include /> <html:messages />

<bean:define /> <html:html>

<bean:write /> <html:form>

<bean:resource /> <html:link>

<bean:cokkie /> <html:text>

<bean:heder /> <logic:present />

<bean:parameter /> <logic:equal />

<bean:size />

## 7.hibernate

### 1.（★★）hibernate 延迟加载?

答：

延迟加载（lazy load）也叫懒加载，它是hibernate为提高程序执行效率而提供的一种机制，即只有真正使用该对象的数据时才会创建。

Hibernate中主要通过代理（proxy）机制来实现延迟加载的。具体过程：Hibernate从数据库获取某一个对象数据时、获取某一个对象的集合属性值时，或获取某一个对象所关联的另一对象时，由于没有使用该对象的数据（除标识符值外），Hibernate并不从数据库加载真正的数据，而只是为该对象创建一个代理对象来代表这个对象，这个对象上的所有属性都为默认值；只有在真正需要使用该对象的数据时才创建这个真实对象，真正从数据库中加载它的数据。这样在某些情况下，就可以提高查询效率。

Hibernate中默认采用延迟加载的情况主要由以下几种：

（1）当调用Session上的load()方法加载一个实体时，会采用延迟加载。

（2）当Session加载某个实体时，会对这个实体中的集合属性值采用延迟加载。

（3）当Session加载某个实体时，会对这个实体所单端关联的另一个实体对象采用延迟加载。

### 2. （★★）Hibernate中get和load有什么不同之处?

答：

把get和load放到一起进行对比是Hibernate面试时最常问到的问题，这是因为只有正确理解get()和load()这二者后才有可能高效地使用Hibernate。get和load的最大区别是，如果在缓存中没有找到相应的对象，get将会直接访问数据库并返回一个完全初始化好的对象，而这个过程有可能会涉及到多个数据库调用；而load方法在缓存中没有发现对象的情况下，只会返回一个代理对象，只有在对象getId()之外的其它方法被调用时才会真正去访问数据库，这样就能在某些情况下大幅度提高性能。

### 3. （★★）hibernate的三种状态之间如何转换

答：

当对象由瞬时状态(Transient)一save()时，就变成了持久化状态；

当我们在Session里存储对象的时候，实际是在Session的Map里存了一份， 也就是它的缓存里放了一份，然后，又到数据库里存了一份，在缓存里这一份叫持久对象(Persistent)。 Session 一 Close()了,它的缓存也都关闭了,整个Session也就失效了,这个时候，这个对象变成了游离状态(Detached)，但数据库中还是存在的。

当游离状态(Detached)update()时，又变为了持久状态(Persistent)。

当持久状态(Persistent)delete()时，又变为了瞬时状态(Transient), 此时，数据库中没有与之对应的记录。

### 4. （★★）Hibernate缓存一般分类有哪些?缓存的作用分为几类?

答：

（1）一级缓存。Session缓存称为一级缓存。由于Session对象的生命周期通常对应一个数据库事务，因此它的缓存是事务范围的缓存。

一级缓存是必需的，Session默认带有，而且不能卸载。在一级缓存中，持久化类的每个实例都具有唯一的OID。

（2）二级缓存。SessionFactory缓存分为内置缓存和外置缓存。内置缓存是Hibernate自带的，不可拆卸，是只读缓存，用来存放映射元数据和预定义SQL语句。外置缓存是一个可配置的缓存插件，默认SessionFactory不会启用这个缓存插件。外置缓存中的数据是数据库数据的复制。SessionFactory对象的生命周期和应用程序的整个进程对应。二级缓存是可选的，可以在每个类或每个集合的粒度上配置二级缓存。

（3）查询缓存。它是Hibernate为查询结果提供的，依赖于二级缓存。

### 5. （★★）Hibernate缓存的作用分为几类?

答：

（1）事务范围。每个事务都有自己的缓存，缓存内数据不会被多个事务并发访问。例如hibernate的一级缓存，事务是不能跨多个Session的，Session内数据只能被当前事务访问，因此它属于事务范围的缓存。

（2）进程范围。进程内的所有事物共享缓存，进程结束，缓存结束生命周期，例如hibernate的二级缓存，SessionFactory对象的生命周期对应应用程序的整个进程，因此它属于进程范围的缓存。

（3）集群范围。缓存被一个或多个机器上的多个进程共享。Hibernate的二级缓存也可以作为集群范围的缓存。

### 6. （★）在 hibernate 开发中，关于 POJO 类对象的状态说法正确的是（）。

A.自由状态（Transient）：实体在内存中自由存在，与数据库中的记录无关

B.持久状态（Persistent）：实体处于由Hibernate框架所管理的状态，对应了数据库中的一条记录，同时与某个session实例发生了关联

C.游离状态（Detached）：在session 关闭之后,可以使游离状态处于持久状态的对象。

D.不能将对象从游离状态转换成持久态

答: ABC

### 7. （★）下面有个hibernate延迟加载，说法错误的是？

A.Hibernate2延迟加载实现：a)实体对象 b)集合（Collection）

B.Hibernate3 提供了属性的延迟加载功能

C.get支持延迟加载，load不支持延迟加

D.hibernate使用Java反射机制，而不是字节码增强程序来实现透明性

答: C

### 8.(★)在Hibernate 的异常处理机制中，如果Session 抛出异常，应当立即回滚数据库事务，调用(D)方法，丢弃该Session实例。

A. Session.Flush()

B. Session.Load

C. Session.Delete()

D. Session.Close()

### 9.(★★★)什么是JPA，JPA和Hibernate之间的关系是什么？

JPA本身是一种规范，它的本质是一种ORM规范（不是ORM框架，因为JPA并未提供ORM实现，只是制定了规范）因为JPA是一种规范，所以，只是提供了一些相关的接口，但是接口并不能直接使用，JPA底层需要某种JPA实现，JPA现在就是Hibernate功能的一个子集

Hibernate 从3.2开始，就开始兼容JPA。Hibernate3.2获得了Sun TCK的 JPA(Java  Persistence API) 兼容认证。JPA和Hibernate之间的关系，可以简单的理解为JPA是标准接口，Hibernate是实现，并不是对标关系，借用下图可以看清楚他们之间的关系，Hibernate属于遵循JPA规范的一种实现，但是JPA是Hibernate遵循的规范之一，Hibernate还有其他实现的规范，所以它们的关系更像是JPA是一种做面条的规范，而Hibernate是一种遵循做面条的规范的汤面，他不仅遵循了做面条的规范，同时也会遵循做汤和调料的其他规范，他们之间并不是吃面条和吃米饭的关系

## 8.OGNL

### 1.（★★★）OGNL是什么?你在项目中如何使用它

答：

OGNL是：对象图形导航语言。

原ognl是单个对象的，在struts2的应用是多个对象的，struts2的类型转换也是通过ognl来实现的。

Struts2中默认的表达式语言就是ognl，struts2的取值设值都是通过ognl来实现的,struts2要依靠它的标签才可以使用ognl。

相对于jstl或者其他表达式语言，它有如下优势：

1 能够调用对象实例的方法

2 能够调用类的静态方法

3 操作集合对象

4 访问Ognl上下文

取值时ognl有三个参数：1 表达式，2 ognl的上下文(map)，3 对象实例(值栈)。

如果表达式带#号是从上下文中(map)拿值，否则是从值栈中拿值。

设值时有四个参数：1 表达式，2 ognl的上下文(map)，3 对象实例，4 设入的值

如果表达式带#号是把值设入上下文中(map)，否则是把值设入值栈中。

在struts2中 ognl的实现关系:ActionConetxt。

ognl 3个常用的符号 # $ %

# 1 构造map，list集合。

2 取ognl上下文的值。

3 用来过滤集合。

$ 1 在校验框架中取资源文件中的值。

2 可以在配置文件中传递参数。

% 使用百分号可以进行转义。

## 9.Ajax

### 1.（★★★）什么是 Ajax？Ajax 的原理是什么？Ajax 的核心技术是什么？Ajax 的优缺点是什么？

答：

Ajax 是 Asynchronous JavaScript and XML 的缩写，是 JavaScript、XML、CSS、DOM 等多个技术的组合。

Ajax 的工作原理是一个页面的指定位置可以加载另一个页面所有的输出内容，这样就实现了一个静态页面也能获取到数据库中的返回数据信息了。所以 Ajax 技术实现了一个静态网页在不刷新整个页面的情况下与服务器通信，减少了用户等待时间，同时也从而降低了网络流量，增强了客户体验的友好程度。

Ajax 的核心技术是 XMLHttpRequest，它是 JavaScript 中的一个对象。

Ajax 的优点是：

(1).减轻了服务器端负担，将一部分以前由服务器负担的工作转移到客户端执行，利用客户端闲置的资源进行处理；

(2).在只局部刷新的情况下更新页面，增加了页面反应速度，使用户体验更友好。

Ajax 的缺点是不利于 SEO 推广优化，因为搜索引擎无法直接访问到 Ajax 请求的内容。

### 2.(★)请用jquery语言写出ajax请求或者post请求代码

答：

$.ajax({

type:'post'

,url:'XX.html'

,dataType:'json'

,success:function(data){

},

});

## 10.mybatis

### 1. （★★）mybatis当实体类中的属性名和表中的字段名不一样

答：

解决办法①：

通过在查询的sql语句中定义字段名的别名，让字段名的别名和实体类的属性名一致

<select id=”selectorder” parametertype=”int” resultetype=”me.gacl.domain.order”>

select order\_id id, order\_no orderno ,order\_price price form orders where order\_id=#{id};

</select>

解决办法②：

通过<resultMap>来映射字段名和实体类属性名的一一对应的关系

<resultMap type=”me.gacl.domain.order” id=”orderresultmap”>

<!–用id属性来映射主键字段–>

<id property=”id” column=”order\_id”>

<!–用result属性来映射非主键字段–>

<result property = “orderno” column =”order\_no”/>

<result property=”price” column=”order\_price” />

</reslutMap>

### 2. （★★）mybatis在mapper中如何传递多个参数?

答：

java的反射机制并不能让框架获取到参数的名字(方法签名中只有参数类型,

可以说是为了优化,也可以说设计就是如此,总之名字无意义), 所以mybatis默认的命名为: param1,param2……

如果想给他们指定名称,可以使用 @param 注解:

01.import org.apache.ibatis.annotations.param;

02.public interface usermapper {

03. user selectuser(@param(“username”) string username,

04. @param(“hashedpassword”) string hashedpassword);

05.}

然后,就可以在xml像下面这样使用(推荐封装为一个map,作为单个参数传递给mapper):

01.<select id=”selectuser” resulttype=”user”>

02. select id, username, hashedpassword

03. from some\_table

04. where username = #{username}

05. and hashedpassword = #{hashedpassword}

06.</select>

### 3. （★★★）什么是MyBatis的接口绑定,有什么好处

答：

接口映射就是在IBatis中任意定义接口,然后把接口里面的方法和SQL语句绑定,

我们直接调用接口方法就可以,这样比起原来了SqlSession提供的方法我们可以有更加灵活的选择和设置.

### 4. （★★）MyBatis实现一对多有几种方式,怎么操作的

答：

有联合查询和嵌套查询,联合查询是几个表联合查询,只查询一次,通过在resultMap里面配置collection节点配置一对多的类就可以完成;

嵌套查询是先查一个表,根据这个表里面的

结果的外键id,去再另外一个表里面查询数据,也是通过配置collection,但另外一个表的查询通过select节点配置

### 5. （★★★）MyBatis里面的动态Sql是怎么设定的?用什么语法?

答：

MyBatis里面的动态Sql一般是通过if节点来实现,通过OGNL语法来实现,但是如果要写的完整,必须配合where,trim节点,where节点是判断包含节点有内容就插入where,否则不插入,trim节点是用来判断如果动态语句是以and 或or开始,那么会自动把这个and或者or取掉

### 6. （★★★）讲下MyBatis的缓存

答：

MyBatis的缓存分为一级缓存和二级缓存,

一级缓存放在session里面,默认就有,二级缓存放在它的命名空间里,默认是打开的,使用二级缓存属性类需要实现Serializable序列化接口(可用来保存对象的状态),可在它的映射文件中配置<cache/>

### 7. （★★★）简单介绍下你对mybatis的理解？

答：

1. mybatis配置

2. SqlMapConfig.xml，此文件作为mybatis的全局配置文件，配置了mybatis的运行环境等信息。

3. mapper.xml文件即sql映射文件，文件中配置了操作数据库的sql语句。此文件需要在SqlMapConfig.xml中加载。

4. 通过mybatis环境等配置信息构造SqlSessionFactory即会话工厂

5. 由会话工厂创建sqlSession即会话，操作数据库需要通过sqlSession进行。

6. mybatis底层自定义了Executor执行器接口操作数据库，Executor接口有两个实现，一个是基本执行器、一个是缓存执行器。

7. Mapped Statement也是mybatis一个底层封装对象，它包装了mybatis配置信息及sql映射信息等。mapper.xml文件中一个sql对应一个Mapped Statement对象，sql的id即是Mapped statement的id。

8. Mapped Statement对sql执行输入参数进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql前将输入的java对象映射至sql中，输入参数映射就是jdbc编程中对preparedStatement设置参数。

9. Mapped Statement对sql执行输出结果进行定义，包括HashMap、基本类型、pojo，Executor通过Mapped Statement在执行sql后将输出结果映射至java对象中，输出结果映射过程相当于jdbc编程中对结果的解析处理过程

### （★★★）高并发量网站解决方案

答：

一个小型的网站，可以使用最简单的html静态页面就实现了，配合一些图片达到美化效果，所有的页面均存放在一个目录下，这样的网站对系统架构、性能的要求都很简单。随着互联网业务的不断丰富，网站相关的技术经过这些年的发展，已经细分到很细的方方面面，尤其对于大型网站来说，所采用的技术更是涉及面非常广，从硬件到软件、编程语言、数据库、WebServer、防火墙等各个领域都有了很高的要求，已经不是原来简单的html静态网站所能比拟的。

　　大型网站，比如门户网站，在面对大量用户访问、高并发请求方面，基本的解决方案集中在这样几个环节：使用高性能的服务器、高性能的数据库、高效率的编程语言、还有高性能的Web容器。这几个解决思路在一定程度上意味着更大的投入。

### （★★）HTML静态化

答：

其实大家都知道，效率最高、消耗最小的就是纯静态化的html页面，所以我们尽可能使我们的网站上的页面采用静态页面来实现，这个最简单的方法其实也是最有效的方法。但是对于大量内容并且频繁更新的网站，我们无法全部手动去挨个实现，于是出现了我们常见的信息发布系统CMS，像我们常访问的各个门户站点的新闻频道，甚至他们的其他频道，都是通过信息发布系统来管理和实现的，信息发布系统可以实现最简单的信息录入自动生成静态页面，还能具备频道管理、权限管理、自动抓取等功能，对于一个大型网站来说，拥有一套高效、可管理的CMS是必不可少的。

　　除了门户和信息发布类型的网站，对于交互性要求很高的社区类型网站来说，尽可能的静态化也是提高性能的必要手段，将社区内的帖子、文章进行实时的静态化、有更新的时候再重新静态化也是大量使用的策略，像Mop的大杂烩就是使用了这样的策略，网易社区等也是如此。

　　同时，html静态化也是某些缓存策略使用的手段，对于系统中频繁使用数据库查询但是内容更新很小的应用，可以考虑使用html静态化来实现。比如论坛中论坛的公用设置信息，这些信息目前的主流论坛都可以进行后台管理并且存储在数据库中，这些信息其实大量被前台程序调用，但是更新频率很小，可以考虑将这部分内容进行后台更新的时候进行静态化，这样避免了大量的数据库访问请求。

### 10.（★★）图片服务器分离

答：

大家知道，对于Web服务器来说，不管是Apache、IIS还是其他容器，图片是最消耗资源的，于是我们有必要将图片与页面进行分离，这是基本上大型网站都会采用的策略，他们都有独立的、甚至很多台的图片服务器。这样的架构可以降低提供页面访问请求的服务器系统压力，并且可以保证系统不会因为图片问题而崩溃。

　　在应用服务器和图片服务器上，可以进行不同的配置优化，比如apache在配置ContentType的时候可以尽量少支持、尽可能少的LoadModule，保证更高的系统消耗和执行效率。

### 11.（★★）数据库集群、库表散列

答：

大型网站都有复杂的应用，这些应用必须使用数据库，那么在面对大量访问的时候，数据库的瓶颈很快就能显现出来，这时一台数据库将很快无法满足应用，于是我们需要使用数据库集群或者库表散列。

　　在数据库集群方面，很多数据库都有自己的解决方案，Oracle、Sybase等都有很好的方案，常用的MySQL提供的Master/Slave也是类似的方案，您使用了什么样的DB，就参考相应的解决方案来实施即可。

　　上面提到的数据库集群由于在架构、成本、扩张性方面都会受到所采用DB类型的限制，于是我们需要从应用程序的角度来考虑改善系统架构，库表散列是常用并且最有效的解决方案。

　　我们在应用程序中安装业务和应用或者功能模块将数据库进行分离，不同的模块对应不同的数据库或者表，再按照一定的策略对某个页面或者功能进行更小的数据库散列，比如用户表，按照用户ID进行表散列，这样就能够低成本的提升系统的性能并且有很好的扩展性。

　　sohu的论坛就是采用了这样的架构，将论坛的用户、设置、帖子等信息进行数据库分离，然后对帖子、用户按照板块和ID进行散列数据库和表，最终可以在配置文件中进行简单的配置便能让系统随时增加一台低成本的数据库进来补充系统性能。

### 12.（★★）缓存

答：

缓存一词搞技术的都接触过，很多地方用到缓存。网站架构和网站开发中的缓存也是非常重要。这里先讲述最基本的两种缓存。高级和分布式的缓存在后面讲述。

　　架构方面的缓存，对Apache比较熟悉的人都能知道Apache提供了自己的缓存模块，也可以使用外加的Squid模块进行缓存，这两种方式均可以有效的提高Apache的访问响应能力。

　　网站程序开发方面的缓存，Linux上提供的Memory Cache是常用的缓存接口，可以在web开发中使用，比如用Java开发的时候就可以调用MemoryCache对一些数据进行缓存和通讯共享，一些大型社区使用了这样的架构。另外，在使用web语言开发的时候，各种语言基本都有自己的缓存模块和方法，PHP有Pear的Cache模块，Java就更多了，.net不是很熟悉，相信也肯定有。

### 13.（★★）镜像

答：

镜像是大型网站常采用的提高性能和数据安全性的方式，镜像的技术可以解决不同网络接入商和地域带来的用户访问速度差异，比如ChinaNet和EduNet之间的差异就促使了很多网站在教育网内搭建镜像站点，数据进行定时更新或者实时更新。在镜像的细节技术方面，这里不阐述太深，有很多专业的现成的解决架构和产品可选。也有廉价的通过软件实现的思路，比如Linux上的rsync等工具。

### 14.（★★）负载均衡

答：

负载均衡将是大型网站解决高负荷访问和大量并发请求采用的高端解决办法。

负载均衡技术发展了多年，有很多专业的服务提供商和产品可以选择，我个人接触过一些解决方法，其中有两个架构可以给大家做参考。

（1）、硬件四层交换

　　第四层交换使用第三层和第四层信息包的报头信息，根据应用区间识别业务流，将整个区间段的业务流分配到合适的应用服务器进行处理。

　　第四层交换功能就像是虚IP，指向物理服务器。它传输的业务服从的协议多种多样，有HTTP、FTP、NFS、Telnet或其他协议。这些业务在物理服务器基础上，需要复杂的载量平衡算法。在IP世界，业务类型由终端TCP或UDP端口地址来决定，在第四层交换中的应用区间则由源端和终端IP地址、TCP和UDP端口共同决定。

　　在硬件四层交换产品领域，有一些知名的产品可以选择，比如Alteon、F5等，这些产品很昂贵，但是物有所值，能够提供非常优秀的性能和很灵活的管理能力。“Yahoo中国”当初接近2000台服务器，只使用了三、四台Alteon就搞定了。

(2)、软件四层交换

　　大家知道了硬件四层交换机的原理后，基于OSI模型来实现的软件四层交换也就应运而生，这样的解决方案实现的原理一致，不过性能稍差。但是满足一定量的压力还是游刃有余的，有人说软件实现方式其实更灵活，处理能力完全看你配置的熟悉能力。

　　软件四层交换我们可以使用Linux上常用的LVS来解决，LVS就是Linux Virtual Server，他提供了基于心跳线heartbeat的实时灾难应对解决方案，提高系统的强壮性，同时可供了灵活的虚拟VIP配置和管理功能，可以同时满足多种应用需求，这对于分布式的系统来说必不可少。

　　一个典型的使用负载均衡的策略就是，在软件或者硬件四层交换的基础上搭建squid集群，这种思路在很多大型网站包括搜索引擎上被采用，这样的架构低成本、高性能还有很强的扩张性，随时往架构里面增减节点都非常容易。

　　对于大型网站来说，前面提到的每个方法可能都会被同时使用到，这里介绍得比较浅显，具体实现过程中很多细节还需要大家慢慢熟悉和体会。有时一个很小的squid参数或者apache参数设置，对于系统性能的影响就会很大。

### 15.（★★）最新：CDN加速技术

答：

什么是CDN？

　　 CDN的全称是内容分发网络。其目的是通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的网络“边缘”，使用户可以就近取得所需的内容，提高用户访问网站的响应速度。

　　CDN有别于镜像，因为它比镜像更智能，或者可以做这样一个比喻：CDN=更智能的镜像+缓存+流量导流。因而，CDN可以明显提高Internet网络中信息流动的效率。从技术上全面解决由于网络带宽小、用户访问量大、网点分布不均等问题，提高用户访问网站的响应速度。

CDN的类型特点

　 　CDN的实现分为三类：镜像、高速缓存、专线。

　　镜像站点（Mirror Site），是最常见的，它让内容直接发布，适用于静态和准动态的数据同步。但是购买和维护新服务器的费用较高，还必须在各个地区设置镜像服务器，配备专业技术人员进行管理与维护。对于大型网站来说，更新所用的带宽成本也大大提高了。

　　高速缓存，成本较低，适用于静态内容。Internet的统计表明，超过80%的用户经常访问的是20%的网站的内容，在这个规律下，缓存服务器可以处理大部分客户的静态请求，而原始的服务器只需处理约20%左右的非缓存请求和动态请求，于是大大加快了客户请求的响应时间，并降低了原始服务器的负载。

　　CDN服务一般会在全国范围内的关键节点上放置缓存服务器。

　　专线，让用户直接访问数据源，可以实现数据的动态同步。

CDN的实例

　　举个例子来说，当某用户访问网站时，网站会利用全球负载均衡技术，将用户的访问指向到距离用户最近的正常工作的缓存服务器上，直接响应用户的请求。

　　当用户访问已经使用了CDN服务的网站时，其解析过程与传统解析方式的最大区别就在于网站的授权域名服务器不是以传统的轮询方式来响应本地DNS的解析请求，而是充分考虑用户发起请求的地点和当时网络的情况，来决定把用户的请求定向到离用户最近同时负载相对较轻的节点缓存服务器上。

　　通过用户定位算法和服务器健康检测算法综合后的数据，可以将用户的请求就近定向到分布在网络“边缘”的缓存服务器上，保证用户的访问能得到更及时可靠的响应。

　　由于大量的用户访问都由分布在网络边缘的CDN节点缓存服务器直接响应了，这就不仅提高了用户的访问质量，同时有效地降低了源服务器的负载压力。

附：某CDN服务商的服务说明

采用GCDN加速方式

　　采用了GCDN加速方式以后，系统会在浏览用户和您的服务器之间增加一台GCDN服务器。浏览用户访问您的服务器时，一般静态数据，如图片、多媒体资料等数据将直接从GCDN服务器读取，使得从主服务器上读取静态数据的交换量大大减少。

　　为VIP型虚拟主机而特加的VPN高速压缩通道，使用高速压缩的电信<==>网通、电信<==>国际（HK）、网通<==>国际（HK）等跨网专线通道，智能多线，自动获取最快路径，极速的动态实时并发响应速度，实现了网站的动态脚本实时同步，对动态网站有一个更加明显的加速效果。

　　每个网络运营商（电信、网通、铁通、教育网）均有您服务器的GCDN服务器，无论浏览用户是来自何处，GCDN都能让您的服务器展现最快的速度！另外，我们将对您的数据进行实时备份，让您的数据更安全

## 11.webservice

### 1. （★★）什么是webservice(用你的话描述webservice)?在什么时候用webservice（webservice能给我们解决什么样的问题）？

答：

一句话概括：WebService是一种跨编程语言和跨操作系统平台的远程调用技术。

所谓跨编程语言和跨操作平台，就是说服务端程序采用Java编写，客户端程序则可以采用其他编程语言编写，反之亦然！跨操作系统平台则是指服务端程序和客户端程序可以在不同的操作系统上运行。

所谓远程调用，就是一台计算机a上的一个程序可以调用到另外一台计算机b上的一个对象的方法。譬如从天气预报系统中获取某个城市的天气数据在自己系统中进行展示；从证券交易系统中获取某只股票的交易信息在自己的系统中进行展示；又譬如一个商城系统中能够展示快递的跟踪信息，而这些信息就是通过webservice从具体的快递公司的系统中获取的数据。

其实可以从多个角度来理解WebService，从表面上看，WebService就是一个应用程序向外界暴露出一个能通过Web进行调用的API，也就是说能用编程的方法通过Web来调用这个应用程序。我们把调用这个WebService的应用程序叫做客户端，而把提供这个WebService的应用程序叫做服务端。从深层次看，WebService是建立可互操作的分布式应用程序的新平台，是一个平台，是一套标准。它定义了应用程序如何在Web上实现互操作性，你可以用任何你喜欢的语言，在任何你喜欢的平台上写Web service ，只要我们可以通过Web service标准对这些服务进行查询和访问。

### 2. （★★）你的系统中是否有使用到webservice开发，具体是怎么实现的？

答：

如果你觉得自己掌握的不够好，对自己不够自信的可以回答为“我的系统中没有使用到webservice的开发，但是我掌握webservice开发的概念和流程”，然后可以给他讲讲相关的概念，这样可以绕过这个问题，因为并不是所有的系统都会涉及到webservice的开发。

另一种回答即是先给他介绍一种webservice开发框架，比如CXF，然后告诉他你做的是服务端开发还是客户端开发，如果你说你做的事服务端开发，那么你就告诉他怎么定义的webservice，使用了哪些注解，怎么跟spring进行的整合，怎么发布的服务等等；如果你告诉他你做的事客户端的开发，那么你可以告诉他你怎么生成的本地代码，然后又怎么通过本地代码去调用的webservice服务。

### 3. （★★）怎么理解UDDI？

答：

UDDI是Universal Description Discovery and Integration的缩写，即统一描述、发现和整合规范。用来注册和查找服务，把web services收集和存储起来，这样当别人访问这些信息的时候就从UDDI中查找，看有没有这个信息存在。

### 4. （★★）Webservice的SEI指什么？

答：

WebService EndPoint Interface（webservice终端[Server端]接口）

就是 WebService 服务器端用来处理请求的接口

### 5. （★★）说说你知道的webservice框架，他们都有什么特点？

答：

Webservice常用框架有JWS、Axis2、XFire以及CXF。

下面分别介绍一个这几种Web Service框架的基本概念

1、JWS是Java语言对WebService服务的一种实现，用来开发和发布服务。而从服务本身的角度来看JWS服务是没有语言界限的。但是Java语言为Java开发者提供便捷发布和调用WebService服务的一种途径。

2、Axis2是Apache下的一个重量级WebService框架，准确说它是一个Web Services / SOAP / WSDL 的引擎，是WebService框架的集大成者，它能不但能制作和发布WebService，而且可以生成Java和其他语言版WebService客户端和服务端代码。这是它的优势所在。但是，这也不可避免的导致了Axis2的复杂性，使用过的开发者都知道，它所依赖的包数量和大小都是很惊人的，打包部署发布都比较麻烦，不能很好的与现有应用整合为一体。但是如果你要开发Java之外别的语言客户端，Axis2提供的丰富工具将是你不二的选择。

3、XFire是一个高性能的WebService框架，在Java6之前，它的知名度甚至超过了Apache的Axis2，XFire的优点是开发方便，与现有的Web整合很好，可以融为一体，并且开发也很方便。但是对Java之外的语言，没有提供相关的代码工具。XFire后来被Apache收购了，原因是它太优秀了，收购后，随着Java6 JWS的兴起，开源的WebService引擎已经不再被看好，渐渐的都败落了。

4、CXF是Apache旗下一个重磅的SOA简易框架，它实现了ESB（企业服务总线）。CXF来自于XFire项目，经过改造后形成的，就像目前的Struts2来自WebWork一样。可以看出XFire的命运会和WebWork的命运一样，最终会淡出人们的视线。CXF不但是一个优秀的Web Services / SOAP / WSDL 引擎，也是一个不错的ESB总线，为SOA的实施提供了一种选择方案，当然他不是最好的，它仅仅实现了SOA架构的一部分。

注：对于Axis2与CXF之间的关系，一个是Axis2出现的时间较早，而CXF的追赶速度快。

如何抉择：

1、如果应用程序需要多语言的支持，Axis2应当是首选了；

2、如果应用程序是遵循 spring哲学路线的话，Apache CXF是一种更好的选择，特别对嵌入式的 Web Services来说；

3、如果应用程序没有新的特性需要的话，就仍是用原来项目所用的框架，比如 Axis1，XFire，Celtrix或BEA 等等厂家自己的Web Services 实现，就别劳民伤财了。

### 6. （★★）WSDL是什么，有什么作用？

答：

WSDL是web service definition language的缩写，即web service的定义（描述）语言。

怎样向别人介绍你的 web service 有什么功能，以及每个函数调用时的参数呢？你可能会自己写一套文档，你甚至可能会口头上告诉需要使用你的web service 的人。这些非正式的方法至少都有一个严重的问题：当程序员坐到电脑前，想要使用你的web service 的时候，他们的工具（如Visual Studio）无法给他们提供任何帮助，因为这些工具根本就不了解你的web service。解决方法是：用机器能阅读的方式提供一个正式的描述文档。web service描述语言 (WSDL)就是这样一个基于 XML的语言，用于描述 web service及其函数、参数和返回值。因为是基于XML 的，所以WSDL 既是机器可阅读的，又是人可阅读的，这将是一个很大的好处。一些最新的开发工具既能根据你的web service 生成WSDL 文档，又能导入WSDL 文档，生成调用相应web service 的代码。

Webservice服务发布之后，通过浏览器访问发布的+?wsdl即可获得wsdl文档。

### 7. （★★）WSDL文档主要有那几部分组成，分别有什么作用？

答：

一个WSDL文档的根元素是definitions元素，WSDL文档包含7个重要的元素：types, import, message, portType, operations, binding和service元素。

1、 definitions元素中一般包括若干个XML命名空间；

2、 Types元素用作一个容器，定义了自定义的特殊数据类型，在声明消息部分（有效负载）的时候，messages定义使用了types元素中定义的数据类型与元素；

3、 Import元素可以让当前的文档使用其他WSDL文档中指定命名空间中的定义；

4、 Message元素描述了Web服务的有效负载。相当于函数调用中的参数和返回值；

5、 PortType元素定义了Web服务的抽象接口，它可以由一个或者多个operation元素，每个operation元素定义了一个RPC样式或者文档样式的Web服务方法；

6、 Operation元素要用一个或者多个messages消息来定义它的输入、输出以及错误；

7、 Binding元素将一个抽象的portType映射到一组具体的协议（SOAP或者HTTP）、消息传递样式（RPC或者document）以及编码样式（literal或者SOAP encoding）；

8、 Service元素包含一个或者多个Port元素

每一个Port元素对应一个不同的Web服务，port将一个URL赋予一个特定的binding，通过location实现。

可以使两个或者多个port元素将不同的URL赋给相同的binding。

## 12.web性能优化

### 1. （★★）合理使用缓存使用

答：

提高性能最好最快的办法当然是通过缓存来改善，对于任何一个web开发者都应该善用缓存。Asp.net下的缓存机制十分强大，用好缓存机制可以让我们极大的改善web应用的性能。

1.页面缓存

2.部分页面缓存

3.使用DataSource缓存  
4.Cache对象

### 2. （★★）避免数据库频繁连接

答：

1.能采用SQL或直接存储过程一次执行的尽量不要用代码多次执行

2、及时关闭数据库连接

### 3. （★★）适当采取配置文件文件存频繁使用文件

答：

1.对不经常更改并且数据量小的可采用xml或者配置文件设置

### 4. （★★）资源文件上传大小验证

答：

1.严格验证上传图片大小

2、严格控制上传Flash动画和视频大小

### 5. （★★）尽量避开访问高峰期，进行数据作业和数据服务

答：

1.执行定时任务尽量避开访问高峰期

2、对应固定报表2可以采取预定格式，避开高峰自动提取

### 6. （★★）数据查询采用真分页

答：

1.需要多少数据取多少数据

### 7. （★★）建设页面跳转覆盖

答：

1.尽量采取弹框或切换选项方式展示数据，避免来回刷新列表重新获大量数据

### 8. （★★）取数据不要查询全部字段

答：

1.查询数据尽量不要SELECT \*

### 9. （★★）数据库优化

**答：**

**1、** 查询出的数据量过大（可以采用多次查询，其他的方法降低数据量），尽量采取分页查询数据

2、锁或者死锁(这也是查询慢最常见的问题，是程序设计的缺陷)

3、返回了不必要的行和列

用OR的字句可以分解成多个查询，并且通过UNION链接多个查询。它们的速度只与是否使用索引有关，如果查询需要用到联合索引，用UNION all执行的效率更高。

4、如果是使用like进行查询的话，简单的使用index是不行的，但是全文索引，耗空间。 like 'a%' 使用索引 like '%a' 不使用索引用 like '%a%' 查询时，查询耗时和字段值总长度成正比,所以不能用CHAR类型，而是VARCHAR。对于字段的值很长的建全文索引。

5、尽量将数据的处理工作放在服务器上，减少网络的开销，如使用存储过程。存储过程是编译、优化过，并且被组织到一个执行规划里，且存储在数据库中的SQL语句（存储过程是数据库服务器端的一段程序），是控制流语言的集合，速度当然快。

6、将需要查询的结果预先计算好放在表中，查询的时候再Select。这在SQL7.0以前是最重要的手段。例如计算商品购买小计计算。

7、没有必要时不要用DISTINCT和ORDER BY，这些动作可以改在客户端执行。它们增加了额外的开销。这同UNION和UNION ALL一样的道理。

8、一次更新多条记录比分多次更新每次一条快,就是说批处理好

9、用临时表，尽量用结果集和Table类性的变量来代替它,Table 类型的变量比临时表好

10、数据库设计：数据库内所有表结构均添加索引

调整原因：

近日数据库压力很大，经查有些大数据量表的查询速度很慢，导致数据库服务器CPU一直持续90%-100%，将这些表添加索引后，CPU很快变正常。

根据查询条件,建立索引,优化索引、优化访问方式，限制结果集的数据量。注意填充因子要适当（最好是使用默认值0）。索引应该尽量小，使用字节数小的列建索引好（参照索引的创建）,不要对有限的几个值的字段建单一索引如性别字段

11、 将大数据表做分库、分区处理:

具体操作如下：

1）、将大数据表与主数据库分离，单独新建一个数据库，然后将这些表做分区；

2）、将数据插入到消息队列内，后台利用windows计划任务执行（5分钟执行一次）C#控制台程序将消息队列内的数据批量（消息队列内有50000条记录，一次性插入到数据表内）插入到相应的数据表内；

调整原因：

例如：用户访问日志，每次用户访问一个页面的时候我们之前的操作是直接将数据插入数据库，这样做对数据库的访问及操作太大，严重影响其他数据插入、查询的效率，利用分库、分区、消息队列完成此操作的好处是用户访问页面的时候不直接对数据库操作，而是在消息队列内积累一定数量的数据后批量插入数据库，只执行一次数据库操作，而且因为数据库分离的原因，对其他的查询及插入不会有影响；

### 10. （★★）分布式架构-独立站点开发

答：

模块化结构化开发，实现多资源分站点，数据分库，为后期实现分布式部署做准备，主要分为以下几部分：

web站点：

1.web前端站点

2.图片、文件资源站点

3.管理端站点  
4.数据接口站点

数据库：

1.业务数据库

2.访问信息数据库、日志

前期访问量和数据量较小可采取单台或小数目台数服务器部署，后期大数据量采取多web站点多数据服务器方式进行部署。

### 11. （★★）分布式部署-CDN分发式网络

答：

CDN的全称是Content Delivery Network，即内容分发网络。其目的是通过在现有的Internet中增加一层新的网络架构，将网站的内容发布到最接近用户的网络"边缘"，使用户可以就近取得所需的内容，解决Internet网络拥塞状况，提高用户访问网站的响应速度。从技术上全面解决由于网络带宽小、用户访问量大、网点分布不均等原因，解决用户访问网站的响应速度慢的根本原因。

该项为收费项目。

### 12. （★★）分布式部署-软负载均衡

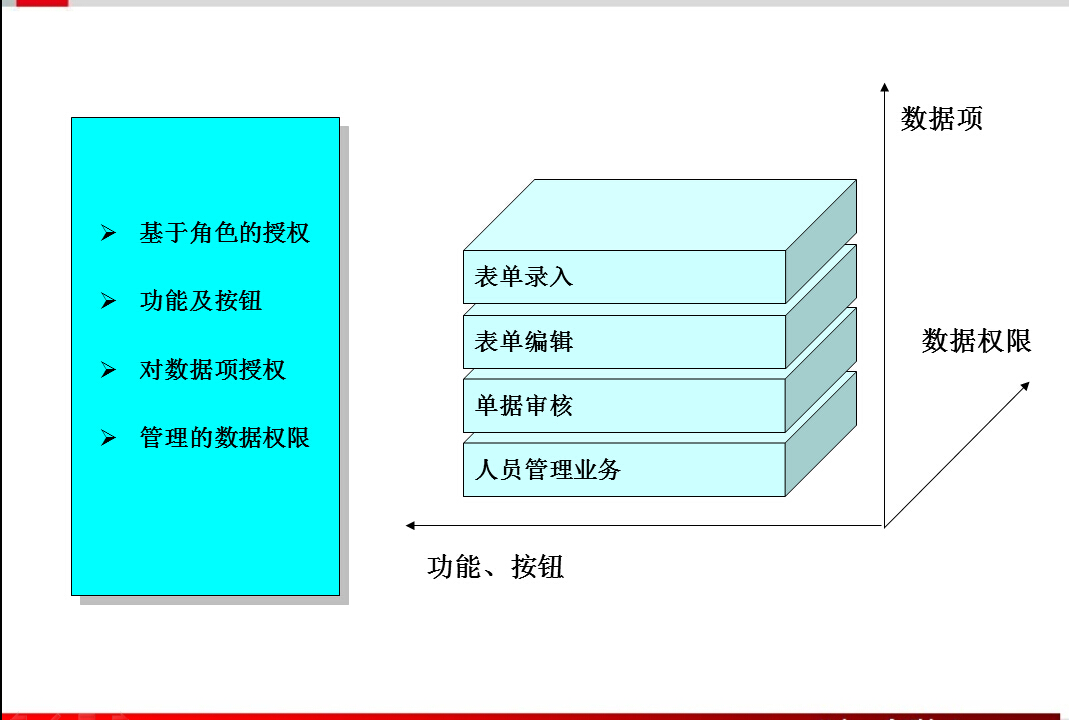
答：

采用nginx进行分流，nginx为轻量级的http服务与反向代理服务器软件，由于其并发能力较强，并且体积很小，所以被称为轻量级http服务软件。  
 nginx的特色功能有：  
  (1).URL rewrite：URL重写  
  (2).reverse proxy：反向代理  
  (3). 做缓存服务器  
  (4). 实现对web服务的负载均衡

(5). 安装第三方插件，实现健康状态监测  
  (6).其他功能

### 13. （★★）权限管理

答：

从模块、表单、数据审核、功能按钮全面数据安全验证及管理。

### 14. （★★）ip验证

答：

数据接口访问进行IP校验

### 15. （★★）登录、操作日志、程序安全日志

答：

系统所有用户登录、操作全部日志记录。

程序安全日志操作可查看我之前写过[[LogHelper 日志记录帮助类](http://blog.csdn.net/fuyifang/article/details/40399225)]。

### 16. （★★）SQL注入校验过滤

答：

a、表单控件js前端校验，特殊字符过滤

b、采用Global.asax的Application\_BeginRequest事件过滤敏感字符。

c、request请求过滤

特殊字符过滤可查看我之前写过[[采用Global.asax的Application\_BeginRequest事件过滤敏感字符](http://blog.csdn.net/fuyifang/article/details/40374809)]。

### 17. （★★）验证规范

答：

a、前端js代码验证

b、后端程序代码验证

c、数据库约束

### 18. （★★）动态验证码

答：

a、邮件动态验证码验证

b、短信动态验证码验证

### 19. （★★）验证码

答：

登录、注册或相关表单采取输入验证码，避免恶意攻击

### 20. （★★★）系统性能定义

答：

让我们先来说说如何什么是系统性能。这个定义非常关键，如果我们不清楚什么是系统性能，那么我们将无法定位之。我见过很多朋友会觉得这很容易，但是仔细一问，其实他们并没有一个比较系统的方法，所以，在这里我想告诉大家如何系统地来定位性能。 总体来说，系统性能就是两个事：

1. Throughput ，吞吐量。也就是每秒钟可以处理的请求数，任务数。
2. Latency， 系统延迟。也就是系统在处理一个请求或一个任务时的延迟。

一般来说，一个系统的性能受到这两个条件的约束，缺一不可。比如，我的系统可以顶得住一百万的并发，但是系统的延迟是2分钟以上，那么，这个一百万的负载毫无意义。系统延迟很短，但是吞吐量很低，同样没有意义。所以，一个好的系统的性能测试必然受到这两个条件的同时作用。 有经验的朋友一定知道，这两个东西的一些关系：

* Throughput越大，Latency会越差。因为请求量过大，系统太繁忙，所以响应速度自然会低。

Latency越好，能支持的Throughput就会越高。因为Latency短说明处理速度快，于是就可以处理更多的请求。

### 21. （★★★）系统性能测试

答：

经过上述的说明，我们知道要测试系统的性能，需要我们收集系统的Throughput和Latency这两个值。

* 首先，需要定义Latency这个值，比如说，对于网站系统响应时间必需是5秒以内（对于某些实时系统可能需要定义的更短，比如5ms以内，这个更根据不同的业务来定义）
* 其次，开发性能测试工具，一个工具用来制造高强度的Throughput，另一个工具用来测量Latency。对于第一个工具，你可以参考一下“[十个免费的Web压力测试工具](http://coolshell.cn/articles/2589.html)”，关于如何测量Latency，你可以在代码中测量，但是这样会影响程序的执行，而且只能测试到程序内部的Latency，真正的Latency是整个系统都算上，包括操作系统和网络的延时，你可以使用Wireshark来抓网络包来测量。这两个工具具体怎么做，这个还请大家自己思考去了。
* 最后，开始性能测试。你需要不断地提升测试的Throughput，然后观察系统的负载情况，如果系统顶得住，那就观察Latency的值。这样，你就可以找到系统的最大负载，并且你可以知道系统的响应延时是多少。

再多说一些，

* 关于Latency，如果吞吐量很少，这个值估计会非常稳定，当吞吐量越来越大时，系统的Latency会出现非常剧烈的抖动，所以，我们在测量Latency的时候，我们需要注意到Latency的分布，也就是说，有百分之几的在我们允许的范围，有百分之几的超出了，有百分之几的完全不可接受。也许，平均下来的Latency达标了，但是其中仅有50%的达到了我们可接受的范围。那也没有意义。
* 关于性能测试，我们还需要定义一个时间段。比如：在某个吞吐量上持续15分钟。因为当负载到达的时候，系统会变得不稳定，当过了一两分钟后，系统才会稳定。另外，也有可能是，你的系统在这个负载下前几分钟还表现正常，然后就不稳定了，甚至垮了。所以，需要这么一段时间。这个值，我们叫做峰值极限。
* 性能测试还需要做Soak Test，也就是在某个吞吐量下，系统可以持续跑一周甚至更长。这个值，我们叫做系统的正常运行的负载极限。

性能测试有很多很复要的东西，比如：burst test等。 这里不能一一详述，这里只说了一些和性能调优相关的东西。总之，性能测试是一细活和累活。

### 22. （★★★）定位性能瓶颈

答：

有了上面的铺垫，我们就可以测试到到系统的性能了，再调优之前，我们先来说说如何找到性能的瓶颈。我见过很多朋友会觉得这很容易，但是仔细一问，其实他们并没有一个比较系统的方法。

#### 3.1）（★★★）查看操作系统负载

答：

首先，当我们系统有问题的时候，我们不要急于去调查我们代码，这个毫无意义。我们首要需要看的是操作系统的报告。看看操作系统的CPU利用率，看看内存使用率，看看操作系统的IO，还有网络的IO，网络链接数，等等。Windows下的perfmon是一个很不错的工具，Linux下也有很多相关的命令和工具，比如：[SystemTap](http://sourceware.org/systemtap/)，[LatencyTOP](https://latencytop.org/)，vmstat, sar, iostat, top, tcpdump等等 。通过观察这些数据，我们就可以知道我们的软件的性能基本上出在哪里。比如：

1）先看CPU利用率，如果CPU利用率不高，但是系统的Throughput和Latency上不去了，这说明我们的程序并没有忙于计算，而是忙于别的一些事，比如IO。（另外，CPU的利用率还要看内核态的和用户态的，内核态的一上去了，整个系统的性能就下来了。而对于多核CPU来说，CPU 0 是相当关键的，如果CPU 0的负载高，那么会影响其它核的性能，因为CPU各核间是需要有调度的，这靠CPU0完成）

2）然后，我们可以看一下IO大不大，IO和CPU一般是反着来的，CPU利用率高则IO不大，IO大则CPU就小。关于IO，我们要看三个事，一个是磁盘文件IO，一个是驱动程序的IO（如：网卡），一个是内存换页率。这三个事都会影响系统性能。

3）然后，查看一下网络带宽使用情况，在Linux下，你可以使用iftop, iptraf, ntop, tcpdump这些命令来查看。或是用Wireshark来查看。

4）如果CPU不高，IO不高，内存使用不高，网络带宽使用不高。但是系统的性能上不去。这说明你的程序有问题，比如，你的程序被阻塞了。可能是因为等那个锁，可能是因为等某个资源，或者是在切换上下文。

通过了解操作系统的性能，我们才知道性能的问题，比如：带宽不够，内存不够，TCP缓冲区不够，等等，很多时候，不需要调整程序的，只需要调整一下硬件或操作系统的配置就可以了。

#### 3.2）（★★★）使用Profiler测试

答：

接下来，我们需要使用性能检测工具，也就是使用某个Profiler来差看一下我们程序的运行性能。如：Java的JProfiler/TPTP/CodePro Profiler，GNU的gprof，IBM的PurifyPlus，Intel的VTune，AMD的CodeAnalyst，还有Linux下的OProfile/perf，后面两个可以让你对你的代码优化到CPU的微指令级别，如果你关心CPU的L1/L2的缓存调优，那么你需要考虑一下使用VTune。 使用这些Profiler工具，可以让你程序中各个模块函数甚至指令的很多东西，如：运行的时间 ，调用的次数，CPU的利用率，等等。这些东西对我们来说非常有用。

我们重点观察运行时间最多，调用次数最多的那些函数和指令。这里注意一下，对于调用次数多但是时间很短的函数，你可能只需要轻微优化一下，你的性能就上去了（比如：某函数一秒种被调用100万次，你想想如果你让这个函数提高0.01毫秒的时间 ，这会给你带来多大的性能）

使用Profiler有个问题我们需要注意一下，因为Profiler会让你的程序运行的性能变低，像PurifyPlus这样的工具会在你的代码中插入很多代码，会导致你的程序运行效率变低，从而没发测试出在高吞吐量下的系统的性能，对此，一般有两个方法来定位系统瓶颈：

1）在你的代码中自己做统计，使用微秒级的计时器和函数调用计算器，每隔10秒把统计log到文件中。

2）分段注释你的代码块，让一些函数空转，做Hard Code的Mock，然后再测试一下系统的Throughput和Latency是否有质的变化，如果有，那么被注释的函数就是性能瓶颈，再在这个函数体内注释代码，直到找到最耗性能的语句。

最后再说一点，对于性能测试，不同的Throughput会出现不同的测试结果，不同的测试数据也会有不同的测试结果。所以，用于性能测试的数据非常重要，性能测试中，我们需要观测试不同Throughput的结果。

### 23. （★★★）常见的系统瓶颈

答：

下面这些东西是我所经历过的一些问题，也许并不全，也许并不对，大家可以补充指正，我纯属抛砖引玉。关于系统架构方面的性能调优，大家可移步看一下《[由12306.cn谈谈网站性能技术](http://coolshell.cn/articles/6470.html)》，关于Web方面的一些性能调优的东西，大家可以看看《[Web开发中需要了解的东西](http://coolshell.cn/articles/6043.html)》一文中的性能一章。我在这里就不再说设计和架构上的东西了。

一般来说，性能优化也就是下面的几个策略：

* 用空间换时间。各种cache如CPU L1/L2/RAM到硬盘，都是用空间来换时间的策略。这样策略基本上是把计算的过程一步一步的保存或缓存下来，这样就不用每次用的时候都要再计算一遍，比如数据缓冲，CDN，等。这样的策略还表现为冗余数据，比如数据镜象，负载均衡什么的。
* 用时间换空间。有时候，少量的空间可能性能会更好，比如网络传输，如果有一些压缩数据的算法（如前些天说的“[Huffman 编码压缩算法](http://coolshell.cn/articles/7459.html)” 和 “[rsync 的核心算法](http://coolshell.cn/articles/7425.html)”），这样的算法其实很耗时，但是因为瓶颈在网络传输，所以用时间来换空间反而能省时间。
* 简化代码。最高效的程序就是不执行任何代码的程序，所以，代码越少性能就越高。关于代码级优化的技术大学里的教科书有很多示例了。如：减少循环的层数，减少递归，在循环中少声明变量，少做分配和释放内存的操作，尽量把循环体内的表达式抽到循环外，条件表达的中的多个条件判断的次序，尽量在程序启动时把一些东西准备好，注意函数调用的开销（栈上开销），注意面向对象语言中临时对象的开销，小心使用异常（不要用异常来检查一些可接受可忽略并经常发生的错误），…… 等等，等等，这连东西需要我们非常了解编程语言和常用的库。
* 并行处理。如果CPU只有一个核，你要玩多进程，多线程，对于计算密集型的软件会反而更慢（因为操作系统调度和切换开销很大），CPU的核多了才能真正体现出多进程多线程的优势。并行处理需要我们的程序有Scalability，不能水平或垂直扩展的程序无法进行并行处理。从架构上来说，这表再为——是否可以做到不改代码只是加加机器就可以完成性能提升？

总之，根据2：8原则来说，20%的代码耗了你80%的性能，找到那20%的代码，你就可以优化那80%的性能。 下面的一些东西都是我的一些经验，我只例举了一些最有价值的性能调优的的方法，供你参考，也欢迎补充。

4.1）算法调优。算法非常重要，好的算法会有更好的性能。举几个我经历过的项目的例子，大家可以感觉一下。

* 一个是过滤算法，系统需要对收到的请求做过滤，我们把可以被filter in/out的东西配置在了一个文件中，原有的过滤算法是遍历过滤配置，后来，我们找到了一种方法可以对这个过滤配置进行排序，这样就可以用二分折半的方法来过滤，系统性能增加了50%。
* 一个是哈希算法。计算哈希算法的函数并不高效，一方面是计算太费时，另一方面是碰撞太高，碰撞高了就跟单向链表一个性能（可参看[Hash Collision DoS 问题](http://coolshell.cn/articles/6424.html)）。我们知道，算法都是和需要处理的数据很有关系的，就算是被大家所嘲笑的“冒泡排序”在某些情况下（大多数数据是排好序的）其效率会高于所有的排序算法。哈希算法也一样，广为人知的哈希算法都是用英文字典做测试，但是我们的业务在数据有其特殊性，所以，对于还需要根据自己的数据来挑选适合的哈希算法。对于我以前的一个项目，公司内某牛人给我发来了一个哈希算法，结果让我们的系统性能上升了150%。（关于各种哈希算法，你一定要看看[StackExchange上的这篇关于各种hash算法的文章](http://programmers.stackexchange.com/questions/49550/which-hashing-algorithm-is-best-for-uniqueness-and-speed/145633#145633) ）
* 分而治之和预处理。以前有一个程序为了生成月报表，每次都需要计算很长的时间，有时候需要花将近一整天的时间。于是我们把我们找到了一种方法可以把这个算法发成增量式的，也就是说我每天都把当天的数据计算好了后和前一天的报表合并，这样可以大大的节省计算时间，每天的数据计算量只需要20分钟，但是如果我要算整个月的，系统则需要10个小时以上（SQL语句在大数据量面前性能成级数性下降）。这种分而治之的思路在大数据面前对性能有很帮助，就像merge排序一样。SQL语句和数据库的性能优化也是这一策略，如：使用嵌套式的Select而不是笛卡尔积的Select，使用视图，等等。

4.2）代码调优。从我的经验上来说，代码上的调优有下面这几点：

* 字符串操作。这是最费系统性能的事了，无论是strcpy, strcat还是strlen，最需要注意的是字符串子串匹配。所以，能用整型最好用整型。举几个例子，第一个例子是N年前做银行的时候，我的同事喜欢把日期存成字符串（如：2012-05-29 08:30:02），我勒个去，一个select  where between语句相当耗时。另一个例子是，我以前有个同事把一些状态码用字符串来处理，他的理由是，这样可以在界面上直接显示，后来性能调优的时候，我把这些状态码全改成整型，然后用位操作查状态，因为有一个每秒钟被调用了150K次的函数里面有三处需要检查状态，经过改善以后，整个系统的性能上升了30%左右。还有一个例子是，我以前从事的某个产品编程规范中有一条是要在每个函数中把函数名定义出来，如：const char fname[]=”functionName()”, 这是为了好打日志，但是为什么不声明成 static类型的呢？
* 多线程调优。有人说，thread is evil，这个对于系统性能在某些时候是个问题。因为多线程瓶颈就在于互斥和同步的锁上，以及线程上下文切换的成本，怎么样的少用锁或不用锁是根本（比如：[多版本并发控制(MVCC)在分布式系统中的应用](http://coolshell.cn/articles/6790.html) 中说的乐观锁可以解决性能问题），此外，还有读写锁也可以解决大多数是读操作的并发的性能问题。这里多说一点在C++中，我们可能会使用线程安全的智能指针AutoPtr或是别的一些容器，只要是线程安全的，其不管三七二十一都要上锁，上锁是个成本很高的操作，使用AutoPtr会让我们的系统性能下降得很快，如果你可以保证不会有线程并发问题，那么你应该不要用AutoPtr。我记得我上次我们同事去掉智能指针的引用计数，让系统性能提升了50%以上。对于Java对象的引用计数，如果我猜的没错的话，到处都是锁，所以，Java的性能问题一直是个问题。另外，线程不是越多越好，线程间的调度和上下文切换也是很夸张的事，尽可能的在一个线程里干，尽可能的不要同步线程。这会让你有很多的性能。
* 内存分配。不要小看程序的内存分配。malloc/realloc/calloc这样的系统调非常耗时，尤其是当内存出现碎片的时候。我以前的公司出过这样一个问题——在用户的站点上，我们的程序有一天不响应了，用GDB跟进去一看，系统hang在了malloc操作上，20秒都没有返回，重启一些系统就好了。这就是内存碎片的问题。这就是为什么很多人抱怨STL有严重的内存碎片的问题，因为太多的小内存的分配释放了。有很多人会以为用内存池可以解决这个问题，但是实际上他们只是重新发明了Runtime-C或操作系统的内存管理机制，完全于事无补。当然解决内存碎片的问题还是通过内存池，具体来说是一系列不同尺寸的内存池（这个留给大家自己去思考）。当然，少进行动态内存分配是最好的。说到内存池就需要说一下池化技术。比如线程池，连接池等。池化技术对于一些短作业来说（如http服务） 相当相当的有效。这项技术可以减少链接建立，线程创建的开销，从而提高性能。
* 异步操作。我们知道Unix下的文件操作是有block和non-block的方式的，像有些系统调用也是block式的，如：Socket下的select，Windows下的WaitforObject之类的，如果我们的程序是同步操作，那么会非常影响性能，我们可以改成异步的，但是改成异步的方式会让你的程序变复杂。异步方式一般要通过队列，要注间队列的性能问题，另外，异步下的状态通知通常是个问题，比如消息事件通知方式，有callback方式，等，这些方式同样可能会影响你的性能。但是通常来说，异步操作会让性能的吞吐率有很大提升（Throughput），但是会牺牲系统的响应时间（latency）。这需要业务上支持。
* 语言和代码库。我们要熟悉语言以及所使用的函数库或类库的性能。比如：STL中的很多容器分配了内存后，那怕你删除元素，内存也不会回收，其会造成内存泄露的假像，并可能造成内存碎片问题。再如，STL某些容器的size()==0  和 empty()是不一样的，因为，size()是O(n)复杂度，empty()是O(1)的复杂度，这个要小心。Java中的JVM调优需要使用的这些参数：-Xms -Xmx -Xmn -XX:SurvivorRatio -XX:MaxTenuringThreshold，还需要注意JVM的GC，GC的霸气大家都知道，尤其是full GC（还整理内存碎片），他就像“恐龙特级克赛号”一样，他运行的时候，整个世界的时间都停止了。

4.3）网络调优

关于网络调优，尤其是TCP Tuning（你可以以这两个关键词在网上找到很多文章），这里面有很多很多东西可以说。看看Linux下TCP/IP的那么多参数就知道了（顺便说一下，你也许不喜欢Linux，但是你不能否认Linux给我们了很多可以进行内核调优的权力）。强烈建议大家看看《[TCP/IP 详解 卷1:协议](http://book.douban.com/subject/1088054/)》这本书。我在这里只讲一些概念上的东西。

A） TCP调优

我们知道TCP链接是有很多开销的，一个是会占用文件描述符，另一个是会开缓存，一般来说一个系统可以支持的TCP链接数是有限的，我们需要清楚地认识到TCP链接对系统的开销是很大的。正是因为TCP是耗资源的，所以，很多攻击都是让你系统上出现大量的TCP链接，把你的系统资源耗尽。比如著名的SYNC Flood攻击。

所以，我们要注意配置KeepAlive参数，这个参数的意思是定义一个时间，如果链接上没有数据传输，系统会在这个时间发一个包，如果没有收到回应，那么TCP就认为链接断了，然后就会把链接关闭，这样可以回收系统资源开销。（注：HTTP层上也有KeepAlive参数）对于像HTTP这样的短链接，设置一个1-2分钟的keepalive非常重要。这可以在一定程度上防止DoS攻击。有下面几个参数（下面这些参数的值仅供参考）：

|  |  |
| --- | --- |
| 1  2  3 | net.ipv4.tcp\_keepalive\_probes = 5  net.ipv4.tcp\_keepalive\_intvl = 20  net.ipv4.tcp\_fin\_timeout = 30 |

对于TCP的TIME\_WAIT这个状态，主动关闭的一方进入TIME\_WAIT状态，TIME\_WAIT状态将持续2个MSL(Max Segment Lifetime)，默认为4分钟，TIME\_WAIT状态下的资源不能回收。有大量的TIME\_WAIT链接的情况一般是在HTTP服务器上。对此，有两个参数需要注意，

|  |  |
| --- | --- |
| 1  2 | net.ipv4.tcp\_tw\_reuse=1  net.ipv4.tcp\_tw\_recycle=1 |

前者表示重用TIME\_WAIT，后者表示回收TIME\_WAIT的资源。

TCP还有一个重要的概念叫RWIN（TCP Receive Window Size），这个东西的意思是，我一个TCP链接在没有向Sender发出ack时可以接收到的最大的数据包。为什么这个很重要？因为如果Sender没有收到Receiver发过来ack，Sender就会停止发送数据并会等一段时间，如果超时，那么就会重传。这就是为什么TCP链接是可靠链接的原因。重传还不是最严重的，如果有丢包发生的话，TCP的带宽使用率会马上受到影响（会盲目减半），再丢包，再减半，然后如果不丢包了，就逐步恢复。相关参数如下：

|  |  |
| --- | --- |
| 1  2  3  4 | net.core.wmem\_default = [8388608](http://www.cnblogs.com/cbscan/articles/tel:8388608)  net.core.rmem\_default = [8388608](http://www.cnblogs.com/cbscan/articles/tel:8388608)  net.core.rmem\_max = [16777216](http://www.cnblogs.com/cbscan/articles/tel:16777216)  net.core.wmem\_max = [16777216](http://www.cnblogs.com/cbscan/articles/tel:16777216) |

一般来说，理论上的RWIN应该设置成：吞吐量  \* 回路时间。Sender端的buffer应该和RWIN有一样的大小，因为Sender端发送完数据后要等Receiver端确认，如果网络延时很大，buffer过小了，确认的次数就会多，于是性能就不高，对网络的利用率也就不高了。也就是说，对于延迟大的网络，我们需要大的buffer，这样可以少一点ack，多一些数据，对于响应快一点的网络，可以少一些buffer。因为，如果有丢包（没有收到ack），buffer过大可能会有问题，因为这会让TCP重传所有的数据，反而影响网络性能。（当然，网络差的情况下，就别玩什么高性能了） 所以，高性能的网络重要的是要让网络丢包率非常非常地小（基本上是用在LAN里），如果网络基本是可信的，这样用大一点的buffer会有更好的网络传输性能（来来回回太多太影响性能了）。

另外，我们想一想，如果网络质量非常好，基本不丢包，而业务上我们不怕偶尔丢几个包，如果是这样的话，那么，我们为什么不用速度更快的UDP呢？你想过这个问题了吗？

B）UDP调优

说到UDP的调优，有一些事我想重点说一样，那就是MTU——最大传输单元（其实这对TCP也一样，因为这是链路层上的东西）。所谓最大传输单元，你可以想像成是公路上的公交车，假设一个公交车可以最多坐70人，带宽就像是公路的车道数一样，如果一条路上最多可以容下100辆公交车，那意味着我最多可以运送7000人，但是如果公交车坐不满，比如平均每辆车只有20人，那么我只运送了2000人，于是我公路资源（带宽资源）就被浪费了。 所以，我们对于一个UDP的包，我们要尽量地让他大到MTU的最大尺寸再往网络上传，这样可以最大化带宽利用率。对于这个MTU，以太网是1500字节，光纤是4352字节，802.11无线网是7981。但是，当我们用TCP/UDP发包的时候，我们的有效负载Payload要低于这个值，因为IP协议会加上20个字节，UDP会加上8个字节（TCP加的更多），所以，一般来说，你的一个UDP包的最大应该是1500-8-20=1472，这是你的数据的大小。当然，如果你用光纤的话， 这个值就可以更大一些。（顺便说一下，对于某些NB的千光以态网网卡来说，在网卡上，网卡硬件如果发现你的包的大小超过了MTU，其会帮你做fragment，到了目标端又会帮你做重组，这就不需要你在程序中处理了）

再多说一下，使用Socket编程的时候，你可以使用setsockopt() 设置 SO\_SNDBUF/SO\_RCVBUF 的大小，TTL和KeepAlive这些关键的设置，当然，还有很多，具体你可以查看一下Socket的手册。

最后说一点，UDP还有一个最大的好处是multi-cast多播，这个技术对于你需要在内网里通知多台结点时非常方便和高效。而且，多播这种技术对于机会的水平扩展（需要增加机器来侦听多播信息）也很有利。

C）网卡调优

对于网卡，我们也是可以调优的，这对于千兆以及网网卡非常必要，在Linux下，我们可以用ifconfig查看网上的统计信息，如果我们看到overrun上有数据，我们就可能需要调整一下txqueuelen的尺寸（一般默认为1000），我们可以调大一些，如：ifconfig eth0 txqueuelen 5000。Linux下还有一个命令叫：ethtool可以用于设置网卡的缓冲区大小。在Windows下，我们可以在网卡适配器中的高级选项卡中调整相关的参数（如：Receive Buffers, Transmit Buffer等，不同的网卡有不同的参数）。把Buffer调大对于需要大数据量的网络传输非常有效。

D）其它网络性能

关于多路复用技术，也就是用一个线程来管理所有的TCP链接，有三个系统调用要重点注意：一个是select，这个系统调用只支持上限1024个链接，第二个是poll，其可以突破1024的限制，但是select和poll本质上是使用的轮询机制，轮询机制在链接多的时候性能很差，因主是O(n)的算法，所以，epoll出现了，epoll是操作系统内核支持的，仅当在链接活跃时，操作系统才会callback，这是由操作系统通知触发的，但其只有Linux Kernel 2.6以后才支持（准确说是2.5.44中引入的），当然，如果所有的链接都是活跃的，过多的使用epoll\_ctl可能会比轮询的方式还影响性能，不过影响的不大。

另外，关于一些和DNS Lookup的系统调用要小心，比如：gethostbyaddr/gethostbyname，这个函数可能会相当的费时，因为其要到网络上去找域名，因为DNS的递归查询，会导致严重超时，而又不能通过设置什么参数来设置time out，对此你可以通过配置hosts文件来加快速度，或是自己在内存中管理对应表，在程序启动时查好，而不要在运行时每次都查。另外，在多线程下面，gethostbyname会一个更严重的问题，就是如果有一个线程的gethostbyname发生阻塞，其它线程都会在gethostbyname处发生阻塞，这个比较变态，要小心。（你可以试试GNU的gethostbyname\_r()，这个的性能要好一些） 这种到网上找信息的东西很多，比如，如果你的Linux使用了NIS，或是NFS，某些用户或文件相关的系统调用就很慢，所以要小心。

4.4）系统调优

A）I/O模型

前面说到过select/poll/epoll这三个系统调用，我们都知道，Unix/Linux下把所有的设备都当成文件来进行I/O，所以，那三个操作更应该算是I/O相关的系统调用。说到  I/O模型，这对于我们的I/O性能相当重要，我们知道，Unix/Linux经典的I/O方式是（关于Linux下的I/O模型，大家可以读一下这篇文章《[使用异步I/O大大提高性能](http://www.ibm.com/developerworks/cn/linux/l-async/)》）：

第一种，同步阻塞式I/O，这个不说了。

第二种，同步无阻塞方式。其通过fctnl设置 O\_NONBLOCK 来完成。

第三种，对于select/poll/epoll这三个是I/O不阻塞，但是在事件上阻塞，算是：I/O异步，事件同步的调用。

第四种，AIO方式。这种I/O 模型是一种处理与 I/O 并行的模型。I/O请求会立即返回，说明请求已经成功发起了。在后台完成I/O操作时，向应用程序发起通知，通知有两种方式：一种是产生一个信号，另一种是执行一个基于线程的回调函数来完成这次 I/O 处理过程。

第四种因为没有任何的阻塞，无论是I/O上，还是事件通知上，所以，其可以让你充分地利用CPU，比起第二种同步无阻塞好处就是，第二种要你一遍一遍地去轮询。Nginx之所所以高效，是其使用了epoll和AIO的方式来进行I/O的。

再说一下Windows下的I/O模型，

a）一个是WriteFile系统调用，这个系统调用可以是同步阻塞的，也可以是同步无阻塞的，关于看文件是不是以Overlapped打开的。关于同步无阻塞，需要设置其最后一个参数Overlapped，微软叫Overlapped I/O，你需要WaitForSingleObject才能知道有没有写完成。这个系统调用的性能可想而知。

b）另一个叫WriteFileEx的系统调用，其可以实现异步I/O，并可以让你传入一个callback函数，等I/O结束后回调之， 但是这个回调的过程Windows是把callback函数放到了APC（[Asynchronous Procedure Calls](http://msdn.microsoft.com/en-us/library/windows/desktop/ms681951(v=vs.85).aspx)）的队列中，然后，只用当应用程序当前线程成为可被通知状态（Alterable）时，才会被回调。只有当你的线程使用了这几个函数时[WaitForSingleObjectEx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687036(v=vs.85).aspx), [WaitForMultipleObjectsEx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms687028(v=vs.85).aspx), [MsgWaitForMultipleObjectsEx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms684245(v=vs.85).aspx), [SignalObjectAndWait](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686293(v=vs.85).aspx) 和 [SleepEx](http://msdn.microsoft.com/en-us/library/windows/desktop/ms686307(v=vs.85).aspx)，线程才会成为Alterable状态。可见，这个模型，还是有wait，所以性能也不高。

c）然后是IOCP – IO Completion Port，IOCP会把I/O的结果放在一个队列中，但是，侦听这个队列的不是主线程，而是专门来干这个事的一个或多个线程去干（老的平台要你自己创建线程，新的平台是你可以创建一个线程池）。IOCP是一个线程池模型。这个和Linux下的AIO模型比较相似，但是实现方式和使用方式完全不一样。

当然，真正提高I/O性能方式是把和外设的I/O的次数降到最低，最好没有，所以，对于读来说，内存cache通常可以从质上提升性能，因为内存比外设快太多了。对于写来说，cache住要写的数据，少写几次，但是cache带来的问题就是实时性的问题，也就是latency会变大，我们需要在写的次数上和相应上做权衡。

B）多核CPU调优

关于CPU的多核技术，我们知道，CPU0是很关键的，如果0号CPU被用得过狠的话，别的CPU性能也会下降，因为CPU0是有调整功能的，所以，我们不能任由操作系统负载均衡，因为我们自己更了解自己的程序，所以，我们可以手动地为其分配CPU核，而不会过多地占用CPU0，或是让我们关键进程和一堆别的进程挤在一起。

* 对于Windows来说，我们可以通过“任务管理器”中的“进程”而中右键菜单中的“设置相关性……”（Set Affinity…）来设置并限制这个进程能被运行在哪些核上。
* 对于Linux来说，可以使用taskset命令来设置（你可以通过安装schedutils来安装这个命令：apt-get install schedutils）

多核CPU还有一个技术叫[NUMA](http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access)技术（Non-Uniform Memory Access）。传统的多核运算是使用SMP(Symmetric Multi-Processor )模式，多个处理器共享一个集中的存储器和I/O总线。于是就会出现一致存储器访问的问题，一致性通常意味着性能问题。NUMA模式下，处理器被划分成多个node， 每个node有自己的本地存储器空间。关于NUMA的一些技术细节，你可以查看一下这篇文章《[Linux 的 NUMA 技术](http://www.ibm.com/developerworks/cn/linux/l-numa/index.html)》，在Linux下，对NUMA调优的命令是：numactl 。如下面的命令：（指定命令“myprogram arg1 arg2”运行在node 0 上，其内存分配在node 0 和 1上）

|  |  |
| --- | --- |
| 1 | numactl --cpubind=0 --membind=0,1 myprogram arg1 arg2 |

当然，上面这个命令并不好，因为内存跨越了两个node，这非常不好。最好的方式是只让程序访问和自己运行一样的node，如：

|  |  |
| --- | --- |
| 1 | $ numactl --membind 1 --cpunodebind 1 --localalloc myapplication |

C）文件系统调优

关于文件系统，因为文件系统也是有cache的，所以，为了让文件系统有最大的性能。首要的事情就是分配足够大的内存，这个非常关键，在Linux下可以使用free命令来查看 free/used/buffers/cached，理想来说，buffers和cached应该有40%左右。然后是一个快速的硬盘控制器，SCSI会好很多。最快的是Intel SSD 固态硬盘，速度超快，但是写次数有限。

接下来，我们就可以调优文件系统配置了，对于Linux的Ext3/4来说，几乎在所有情况下都有所帮助的一个参数是关闭文件系统访问时间，在/etc/fstab下看看你的文件系统 有没有noatime参数（一般来说应该有），还有一个是dealloc，它可以让系统在最后时刻决定写入文件发生时使用哪个块，可优化这个写入程序。还要注间一下三种日志模式：data=journal、data=ordered和data=writeback。默认设置data=ordered提供性能和防护之间的最佳平衡。

当然，对于这些来说，ext4的默认设置基本上是最佳优化了。

这里介绍一个Linux下的查看I/O的命令—— iotop，可以让你看到各进程的磁盘读写的负载情况。

其它还有一些关于NFS、XFS的调优，大家可以上google搜索一些相关优化的文章看看。关于各文件系统，大家可以看一下这篇文章——《[Linux日志文件系统及性能分析](http://www.ibm.com/developerworks/cn/linux/l-jfs/)》

4.5）数据库调优

数据库调优并不是我的强项，我就仅用我非常有限的知识说上一些吧。注意，下面的这些东西并不一定正确，因为在不同的业务场景，不同的数据库设计下可能会得到完全相反的结论，所以，我仅在这里做一些一般性的说明，具体问题还要具体分析。

A）数据库引擎调优

我对数据库引擎不是熟，但是有几个事情我觉得是一定要去了解的。

* 数据库的锁的方式。这个非常非常地重要。并发情况下，锁是非常非常影响性能的。各种隔离级别，行锁，表锁，页锁，读写锁，事务锁，以及各种写优先还是读优先机制。性能最高的是不要锁，所以，分库分表，冗余数据，减少一致性事务处理，可以有效地提高性能。NoSQL就是牺牲了一致性和事务处理，并冗余数据，从而达到了分布式和高性能。
* 数据库的存储机制。不但要搞清楚各种类型字段是怎么存储的，更重要的是数据库的数据存储方式，是怎么分区的，是怎么管理的，比如Oracle的数据文件，表空间，段，等等。了解清楚这个机制可以减轻很多的I/O负载。比如：MySQL下使用show engines;可以看到各种存储引擎的支持。不同的存储引擎有不同的侧重点，针对不同的业务或数据库设计会让你有不同的性能。
* 数据库的分布式策略。最简单的就是复制或镜像，需要了解分布式的一致性算法，或是主主同步，主从同步。通过了解这种技术的机理可以做到数据库级别的水平扩展。

B）SQL语句优化

关于SQL语句的优化，首先也是要使用工具，比如：[MySQL SQL Query Analyzer](http://www.mysql.com/products/enterprise/query.html)，[Oracle SQL Performance Analyzer](http://www.oracle-base.com/articles/11g/sql-performance-analyzer-11gr1.php)，或是微软[SQL Query Analyzer](http://msdn.microsoft.com/en-us/library/aa216945(v=sql.80).aspx)，基本上来说，所有的RMDB都会有这样的工具，来让你查看你的应用中的SQL的性能问题。 还可以使用explain来看看SQL语句最终Execution Plan会是什么样的。

还有一点很重要，数据库的各种操作需要大量的内存，所以服务器的内存要够，优其应对那些多表查询的SQL语句，那是相当的耗内存。

下面我根据我有限的数据库SQL的知识说几个会有性能问题的SQL：

* 全表检索。比如：select \* from user where lastname = “xxxx”，这样的SQL语句基本上是全表查找，线性复杂度O(n)，记录数越多，性能也越差（如：100条记录的查找要50ms，一百万条记录需要5分钟）。对于这种情况，我们可以有两种方法提高性能：一种方法是分表，把记录数降下来，另一种方法是建索引（为lastname建索引）。索引就像是key-value的数据结构一样，key就是where后面的字段，value就是物理行号，对索引的搜索复杂度是基本上是O(log(n)) ——用B-Tree实现索引（如：100条记录的查找要50ms，一百万条记录需要100ms）。
* 索引。对于索引字段，最好不要在字段上做计算、类型转换、函数、空值判断、字段连接操作，这些操作都会破坏索引原本的性能。当然，索引一般都出现在Where或是Order by字句中，所以对Where和Order by子句中的子段最好不要进行计算操作，或是加上什么NOT之类的，或是使用什么函数。
* 多表查询。关系型数据库最多的操作就是多表查询，多表查询主要有三个关键字，EXISTS，IN和JOIN（关于各种join，可以参看[图解SQL的Join](http://coolshell.cn/articles/3463.html)一文）。基本来说，现代的数据引擎对SQL语句优化得都挺好的，JOIN和IN/EXISTS在结果上有些不同，但性能基本上都差不多。有人说，EXISTS的性能要好于IN，IN的性能要好于JOIN，我各人觉得，这个还要看你的数据、schema和SQL语句的复杂度，对于一般的简单的情况来说，都差不多，所以千万不要使用过多的嵌套，千万不要让你的SQL太复杂，宁可使用几个简单的SQL也不要使用一个巨大无比的嵌套N级的SQL。还有人说，如果两个表的数据量差不多，Exists的性能可能会高于In，In可能会高于Join，如果这两个表一大一小，那么子查询中，Exists用大表，In则用小表。这个，我没有验证过，放在这里让大家讨论吧。另，有一篇关于SQL Server的文章大家可以看看《[IN vs JOIN vs EXISTS](http://explainextended.com/2009/06/16/in-vs-join-vs-exists/)》
* JOIN操作。有人说，Join表的顺序会影响性能，只要Join的结果集是一样，性能和join的次序无关。因为后台的数据库引擎会帮我们优化的。Join有三种实现算法，嵌套循环，排序归并，和Hash式的Join。（MySQL只支持第一种）
  + 嵌套循环，就好像是我们常见的多重嵌套循环。注意，前面的索引说过，数据库的索引查找算法用的是B-Tree，这是O(log(n))的算法，所以，整个算法复法度应该是O(log(n)) \* O(log(m)) 这样的。
  + Hash式的Join，主要解决嵌套循环的O(log(n))的复杂，使用一个临时的hash表来标记。
  + 排序归并，意思是两个表按照查询字段排好序，然后再合并。当然，索引字段一般是排好序的。

还是那句话，具体要看什么样的数据，什么样的SQL语句，你才知道用哪种方法是最好的。

* 部分结果集。我们知道MySQL里的Limit关键字，Oracle里的rownum，SQL Server里的Top都是在限制前几条的返回结果。这给了我们数据库引擎很多可以调优的空间。一般来说，返回top n的记录数据需要我们使用order by，注意在这里我们需要为order by的字段建立索引。有了被建索引的order by后，会让我们的select语句的性能不会被记录数的所影响。使用这个技术，一般来说我们前台会以分页方式来显现数据，Mysql用的是OFFSET，SQL Server用的是FETCH NEXT，这种Fetch的方式其实并不好是线性复杂度，所以，如果我们能够知道order by字段的第二页的起始值，我们就可以在where语句里直接使用>=的表达式来select，这种技术叫seek，而不是fetch，seek的性能比fetch要高很多。
* 字符串。正如我前面所说的，字符串操作对性能上有非常大的恶梦，所以，能用数据的情况就用数字，比如：时间，工号，等。
* 全文检索。千万不要用Like之类的东西来做全文检索，如果要玩全文检索，可以尝试使用[Sphinx](http://sphinxsearch.com/)。
* 其它。
  + 不要select \*，而是明确指出各个字段，如果有多个表，一定要在字段名前加上表名，不要让引擎去算。
  + 不要用Having，因为其要遍历所有的记录。性能差得不能再差。
  + 尽可能地使用UNION ALL  取代  UNION。
  + 索引过多，insert和delete就会越慢。而update如果update多数索引，也会慢，但是如果只update一个，则只会影响一个索引表。
  + 等等。

关于SQL语句的优化，网上有很多文章， 不同的数据库引擎有不同的优化技巧，正如本站以前转发的《[MySQL性能优化的最佳20+条经验](http://coolshell.cn/articles/1846.html)》

先写这么多吧，欢迎大家指正补充。

关于性能优化这是一个比较大的话题，在《[由12306.cn谈谈网站性能技术](http://coolshell.cn/articles/6470.html)》中我从业务和设计上说过一些可用的技术以及那些技术的优缺点，今天，想从一些技术细节上谈谈性能优化，主要是一些代码级别的技术和方法。**本文的东西是我的一些经验和知识，并不一定全对，希望大家指正和补充**。

在开始这篇文章之前，大家可以移步去看一下酷壳以前发表的《[代码优化概要](http://coolshell.cn/articles/2967.html)》，这篇文章基本上告诉你——**要进行优化，先得找到性能瓶颈**！ 但是在讲如何定位系统性能瓶劲之前，请让我讲一下系统性能的定义和测试，因为没有这两件事，后面的定位和优化无从谈起。

### 24. （★★★）系统性能定义

让我们先来说说如何什么是系统性能。这个定义非常关键，如果我们不清楚什么是系统性能，那么我们将无法定位之。我见过很多朋友会觉得这很容易，但是仔细一问，其实他们并没有一个比较系统的方法，所以，在这里我想告诉大家如何系统地来定位性能。 总体来说，系统性能就是两个事：

1. **Throughput** ，吞吐量。也就是每秒钟可以处理的请求数，任务数。
2. **Latency**， 系统延迟。也就是系统在处理一个请求或一个任务时的延迟。

一般来说，一个系统的性能受到这两个条件的约束，缺一不可。比如，我的系统可以顶得住一百万的并发，但是系统的延迟是2分钟以上，那么，这个一百万的负载毫无意义。系统延迟很短，但是吞吐量很低，同样没有意义。所以，一个好的系统的性能测试必然受到这两个条件的同时作用。 有经验的朋友一定知道，这两个东西的一些关系：

* **Throughput越大，Latency会越差。**因为请求量过大，系统太繁忙，所以响应速度自然会低。
* **Latency越好，能支持的Throughput就会越高。**因为Latency短说明处理速度快，于是就可以处理更多的请求。

### 25. （★★★）系统性能测试

经过上述的说明，我们知道要测试系统的性能，需要我们收集系统的Throughput和Latency这两个值。

* 首先，**需要定义Latency这个值**，比如说，对于网站系统响应时间必需是5秒以内（对于某些实时系统可能需要定义的更短，比如5ms以内，这个更根据不同的业务来定义）
* 其次，**开发性能测试工具**，一个工具用来制造高强度的Throughput，另一个工具用来测量Latency。对于第一个工具，你可以参考一下“[十个免费的Web压力测试工具](http://coolshell.cn/articles/2589.html)”，关于如何测量Latency，你可以在代码中测量，但是这样会影响程序的执行，而且只能测试到程序内部的Latency，真正的Latency是整个系统都算上，包括操作系统和网络的延时，你可以使用Wireshark来抓网络包来测量。这两个工具具体怎么做，这个还请大家自己思考去了。
* 最后，**开始性能测试**。你需要不断地提升测试的Throughput，然后观察系统的负载情况，如果系统顶得住，那就观察Latency的值。这样，你就可以找到系统的最大负载，并且你可以知道系统的响应延时是多少。

再多说一些，

* 关于Latency，如果吞吐量很少，这个值估计会非常稳定，当吞吐量越来越大时，系统的Latency会出现非常剧烈的抖动，所以，我们在测量Latency的时候，我们需要注意到Latency的分布，也就是说，有百分之几的在我们允许的范围，有百分之几的超出了，有百分之几的完全不可接受。也许，平均下来的Latency达标了，但是其中仅有50%的达到了我们可接受的范围。那也没有意义。
* 关于性能测试，我们还需要定义一个时间段。比如：在某个吞吐量上持续15分钟。因为当负载到达的时候，系统会变得不稳定，当过了一两分钟后，系统才会稳定。另外，也有可能是，你的系统在这个负载下前几分钟还表现正常，然后就不稳定了，甚至垮了。所以，需要这么一段时间。这个值，我们叫做峰值极限。
* 性能测试还需要做Soak Test，也就是在某个吞吐量下，系统可以持续跑一周甚至更长。这个值，我们叫做系统的正常运行的负载极限。

性能测试有很多很复要的东西，比如：burst test等。 这里不能一一详述，这里只说了一些和性能调优相关的东西。总之，性能测试是一细活和累活。

## 13.Javaweb杂项

### （★★）编写一个方法。求一个字符串的字节长度？

答：

<script language="javascript" type="text/javascript">

new function(s)

{

if(!arguments.length||!s) return null;

if(""==s) return 0;

var l=0;

for(var i=0;i<s.length;i++)

{

   if(s.charCodeAt(i)>255) l+=2;

   else l++;

}

alert(l);

}("你");

</script>

// GBK字符集实际长度计算

function getStrLeng(str){

    var realLength = 0;

    var len = str.length;

    var charCode = -1;

    for(var i = 0; i < len; i++){

        charCode = str.charCodeAt(i);

        if (charCode >= 0 && charCode <= 128) {

            realLength += 1;

        }else{

            // 如果是中文则长度加2

            realLength += 2;

        }

    }

    return realLength;

}

// UTF8字符集实际长度计算

function getStrLeng(str){

    var realLength = 0;

    var len = str.length;

    var charCode = -1;

    for(var i = 0; i < len; i++){

        charCode = str.charCodeAt(i);

        if (charCode >= 0 && charCode <= 128) {

            realLength += 1;

        }else{

            // 如果是中文则长度加3

            realLength += 3;

        }

    }

    return realLength;

}

### 2. （★★）用JavaScript实现用正则表达式验证，某个字符串是否合法的6位数字的邮编的函数。

答：

var reg = /\d{6}/

### 3. （★★）请使用jQuery将页面上的所有元素边框设置为2px宽的虚线？

答：

<!DOCTYPE HTML>

<html>

<head>

<title>yugi</title>

<meta charset=UTF-8 />

<style type="text/css">

</style>

<script type="text/javascript" src="jquery-1.8.0.min.js"></script>

<script type="text/javascript">

jQuery (function ($)

{

$ ("\*").css ("border", "2px dashed");

})

</script>

</head>

<body>

</body>

</html>

### 4. （★★）如何设定jQuery异步调用还是同步调用？

答：

(1) 异步请求：

    1.1 $.ajax

       $.ajax({

                url : 'your url',

                data:{name:value},

                cache : false,

                async : true,

                type : "POST",

                dataType : 'json/xml/html',

                success : function (result){

                    do something....

                }

            });

    (2) 同步请求

    2.1 $.ajax

       $.ajax({

                url : 'your url',

                data:{name:value},

                cache : false,

                async : false,

                type : "POST",

                dataType : 'json/xml/html',

                success : function (result){

                    do something....

                }

            });

    2.2 $.post

      $.post(

                'your url',

                {name:value},

                function(data) {

                    do something...

                },

            'json/xml/html'

            );

### 5. （★★）说出3条以上firefox和IE的浏览器脚本兼容问题？

答：

IE/Firefox中JS脚本常见兼容问题

1、document.form.item/document.ID

IE中 document.formName.item(”itemName”)

FF中 document.formName.elements["elementName"]

建议使用document.getElementById(itemName)

2, 下标运算

IE中 document.forms(”formName”)

FF中 document.forms["formName"]

IE中 document.getElementsByName(”inputName”)(1)

FF中 document.getElementsByName(”inputName”)[1]

建议使用[]作为下标运算

3,window.event

IE中 可以使用window.event

FF中 event 只能在事件发生的现场使用,可以改写成 事件=函数(event);

function somemethod(evt) {

evt = evt ? evt : (window.event ? window.event : null);

alert(evt);

}

例如: <input onclick=somemethod(event)>

4,event.x 与 event.y 问题

IE中,event 对象有 x, y 属性

FF中,可用event.clientX ,event.clientY替代(IE也有该属性)

也可用:mX = event.x ? event.x : event.pageX;

5,操作frame

IE中 可以用window.FrameName取得该frame，FF中不行

FF中 可以用window.top.document.getElementById(”frameId”)来访问frame

注:IE 和FF都可以通过window.top.document.getElementById(”Frame”).src = ’somefile.htm’来切换frame的 内容,也都可以通过window.top.frameName.location = ’somefile.htm’来切换frame的内容

6,调用showModalDialog

IE中可以用showModalDialog一个子窗口,并获得返回值.

FF中没有showModalDialog,但可以用window.open来实现.

例如:

在Main.cfm文件中有如下代码:

function ShowItemList(Obj){

if (document.all){//IE

var ReturnValue=window.showModalDialog(”ItemList.cfm?Id=341″,”self”,”dialogwidth:500px;status:false”);

if (typeOf(ReturnValue)!=’undefined’){

Obj.value=ReturnValue;

}

}

else{

var subwin=window.open(Item.cfm?id=341,’newWin’,’modal=yes,width=500px’);

}

}

function ReturnValue(ReturnValue){

Obj=document.getElementById(’elementname’);

Obj.value=ReturnValue;

}

如果需要得到返回值.需要用到window.open的参数modal=yes,而且必须在子窗口中向母窗口传值(window.opener).

例如在subwin.cfm中加入以下代码:

function ReturnThisValue(){

window.opener.ReturnValue(document.getElementById(’SelectedItem’).value);

}

7,其它

a, 在FF中，自定义的属性必须使用getAttribute()

b, FF中没有 parentElement parement.children 而用parentNode parentNode.childNodes,可以使用 node.getElementsByTagName()

c, FF不支持onpropertychange事件

d, FF中 createElement不支持HTML代码,可以考虑用:document.write(esHTML);

e, IE 中innerText ,在FF中可以用textContent

f, FF下用class代替IE下的className

g, 如果FF调用obj.focus(); 报错,请尝试改为:window.setTimeout( function(){ obj.focus(); }, 0);

h, firefox不支持 FILTER

i, IE: blur发生在focus后,FF: blur发生在focus前调用

j. IE:event.keyCode=====FF：event.which

function catcathevent(evt)

{

evt = evt ? evt : (window.event ? window.event : null);

if (document.all){

var keyvalue=evt.keyCode;

}

else{

var keyvalue=evt.which;

}

k. 设置容器位置 left、top及长宽,必须加上’px’,如 $(’obj’).style.left = ‘50px’;

l.在FF地址栏中输入about:config，会出现火狐的参数配置设置