目录

1、架构：

c/s 架构 (client 客户端-server 服务端)

(胖客户端:要求客户端运行业务；把业务放到服务器端，则是瘦客户端)

典型的 c/s 应用：ftp 工具、QQ、邮件系统、杀毒软件...

1.建立在 tcp/ip 协议之上，有自己的通信规则(建立业务)

2.需要相互配合才能完成一个完整业务逻辑

3.允许多个客户端程序同时接入一个 server 程序(并发)

4.每一个 client(机器)都必须安装客户软件

5.修改了 server 程序，通常 client 程序都要修改(升级)

优点：利用客户端的计算能力，分担服务器的负荷(大型网络游戏就利用这点)

缺点：用户必须安装客户端程序；客户端需要升级(麻烦)

b/s 架构 (browser - web server(cluster 集群)) (极瘦客户端:最低限度地减少客户端程序，只需要 browser(浏览器))

1.基于 http 协议(应用层)

2.几乎所有的业务逻辑处理都在 server 完成

3.支持并发

4.client 要求很少，只需要安装 browser(浏览器)

5.修改 server 之后，client 不需要任何变化

6.server 端开发技术： html/js,xhtml,... php,asp,jsp,servlet

优点：支持高并发访问；不需另外安装软件(只需浏览器)，免去更新的麻烦。

缺点：所有业务都在服务器端完成，服务器负荷大。

2、什么是 servlet?

Servlet 是 sun 公司制订的一种用来扩展 web 服务器功能的组件规范。

Server + Applet =Servlet 意为服务器端的小程序。

如何理解：

a,扩展 web 服务器功能： 指的是标准的 web 服务器只具有与客户端（浏览器）通讯的功能，不能够处理业务、逻辑的请求，需要编写相应的程序来负责处理客户端的请求。

b,组件: 实现了特定规范的可以单独部署的软件模块；组件一般用来实现业务逻辑；组件必须依赖容器来运行。

c,容器: 实现了特定规范的程序，提供组件的运行环境，并且管理组件的生命周期。一般的 web 服务器，比如 tomcat,weblogic,was 都内置有一个 servlet 容器。

3、如何开发一个 servlet?

1)写一个 java 类，实现 Servlet 接口或者继承 HttpServlet 类。

2)编译:(servlet-api.jar)

3)打包: appname(名称任意) WEB-INF classes(放字节码文件) lib(可选，放一些 jar 文件) web.xml(部署描述文件)

4)部署: 将第 3 步生成的文件夹或者使用 jar 命令 将该文件夹压缩所生成的.war 文件 copy toweb 服务器特定的文件夹下面。然后启动服务器。

5)访问: http://ip:port/appname/url-pattern

4、web.xml 的模板(一个 web.xml 中可以配置多个 Servlet)：

```
<?xml version="1.0" encoding="UTF-8"?>
   <web-app version="2.5"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
   http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
<servlet>
 <servlet-name>servlet 的名字 1</servlet-name> //servlet 的逻辑名
 <servlet-class>servlet 类全名 1</servlet-class> //类的完全限定名
</servlet>
<servlet>
 <servlet-name>servlet 的名字 2</servlet-name>
```

```
    <servlet-class>servlet 类全名 2</servlet-class>
  </servlet>
    <servlet-mapping>
    <servlet-name>servlet 的名字 1</servlet-name> //要和 servlet 标签中的相同
    <url-pattern>指定 servlet 相对于应用目录的路径</url-pattern> //servlet 的访问路径</servlet-mapping>
  <servlet-mapping>
    <servlet-name>servlet 的名字 2</servlet-name>
    <url-pattern>指定 servlet 相对于应用目录的路径</url-pattern>
  </servlet-mapping>
 <welcome-file-list>
  <welcome-file>index.jsp</welcome-file>//指定默认的欢迎页面
 </welcome-file-list>
</web-app>
```

5、tomcat 的安装：

1)将/opt/apache-tomcat 压缩文件，解压缩到/homt/soft01 下。 比如，解压缩以后，在 soft01 下，有了一个文件夹叫 apache-tomcat-5.5.29。

2)配置环境变量 cd /home/soft01 vi .bash_profile JAVA_HOME

CATALINA_HOME:/home/soft01/apache-tomcat-5.5.29

PATH:/home/soft01/apache-tomcat-5.5.29/bin . .bash_profile

3)启动服务器 cd apache-tomcat-5.5.29 cd bin sh startup.sh

4)在浏览器地址栏输入: http://localhost:8080

5)关闭服务器 cd apache-tomcat-5.5.29 cd bin sh shutdown.sh

6)tomcat 的几个文件夹的作用。

bin:是一些可执行文件，比如启动和关闭服务器的脚本。

conf:是一些配置文件,比如 server.xml,可以配置 tomcat 的监听端口号等等。

webapps:部署文件夹，将一个应用 copy 到这儿，服务器会自动部署。

work:服务器在运行时，临时生成的一些文件，比如，调用 jsp 所生成的 servlet 源代码及字节码。

6、Servlet 的运行过程（重点）：

在浏览器地址栏输入:

http://ip:port/appname/hello?name=zs

a,浏览器会依据 ip,port 连接服务器，浏览器将/appname/hello?name=zs(请求资源路径)存放到请求数据包(依据 http 协议打包)。

b,Servlet 引擎(web server 当中负责通讯的模块)会创建 Request 对象(一般称为请求对象,Servlet 引擎会将请求数据包中的数据封装到 Request 对象当中，方便 Servlet 获取数据，也就是说，Servlet 不用处

理 http 协议相关的代码），还会创建 Response 对象(一般称为 响应对象，方便 Servlet 将处理之后的结果返回给客户端)。

c,Servlet 引擎依据/appname 找到对应的应用。依据应用所对应的 web.xml，找到 url-pattern 元素。接下来，依据 servlet-class 元素指定的类名，创建 Servlet 实例。

d,Servlet 实例可以通过 Request 对象获取请求参数值。也可以通过 Response 对象输出结果。

7、http 协议(了解)：

http 协议是什么？

超文本传输控制协议,其作用是：定义了浏览器与 web 服务器之间数据传输的过程及数据的格式。

a,数据传输的过程：

浏览器向服务器发送建立连接的请求。

浏览器向服务器发送请求数据。

服务器处理请求数据，返回响应数据。

服务器立即关闭连接。

b,数据的格式: 请求数据包：

1 请求行 请求方式(get/post) 请求资源的路径 协议的版本号

2 消息头 由 w3c 定义的一些关键字，用于浏览器与服务器之间发送一些特定的消息，比如，浏览器可以发送 cookie 消息头，向服

务器发送 cookie 数据。

3 实体内容 如果请求方式是 post 方式，则请求参数会存放在实体内容里面。如果是 get 方式，则请求参数会存放在请求资源路径后面。

响应数据包：

1 状态行

协议的版本

常见的状态码:

状态码	描述
404:	找不到资源
500:	系统出错（应用程序出错）
200:	正确

2 消息头

3 实体内容

服务器返回的数据

8、get/post 请求：

1)哪一些是 get 请求：

a,在浏览器地址栏直接输入一个地址。

b,点击链接地址。click

c,表单默认的提交方式。<form action="abc.do">

2)哪一些是 post 请求： 表单设置了 method="post"。

3)get 方式的特点: 因为请求参数都添加在请求资源路径后面，所以，添加的参数大小有限制。并且，请求参数会在浏览器地址栏显示，不安全。 浏览器会缓存 get 方式所获取的资源。

4)post 方式的特点: 请求参数都添加到了实体内容里面，添加参数大小理论上没有限制。因为请求参数不会在浏览器地址栏显示，相对安全。 get 请求一般用于向服务器请求资源。 Post 请求一般用于向服务器提交数据。

9、表单处理：

如何获取表单中的数据：

String request.getParameter(String paraName);

需要注意：

paraName 对应的参数名不存在，返回 null。

String[] request.getParameterValues(String paraName);

Map request.getParameterMap();返回所有的请求参数与请求参数值对。

10、中文问题（1）：

a,如果是一个静态页面（html），里面有中文，一定要设置

<meta http-equiv="Conent-Type"content="text/html;charset=utf-8">,

并且要保证保存该页面的编码也是 utf-8。 当然，charset 也可以 gbk,gb2312。其作用是，模拟 Conent-Type 消息头，告诉浏览器当前返回的页面的编码是什么。

b,在 Servlet 类的 service 方法里：

request.setCharacterEncoding("utf-8");其作用是，告诉服务器，以指定的编码格式去解码。如果 Servlet 还要将结果输出到客户端（浏览器）还需要:

response.setContentType("text/html;charset=utf-8"); 其作用是：第一，告诉服务器，输出数据到 PrintWriter 时，所采用的编码格式；第二，生成一个消息头,告诉浏览器，以指定的编码来显示返回的数据。

c,数据库

useUnicode=true&characterEncoding=utf8

11、重定向（重点）：

1)什么是重定向？ 服务器向浏览器发送一个状态码 302 及一个消息头 location(location 的值是一个 地址)，浏览器会立即向 location 所指定的地址发送一个新的请求。我们把这样一种 机制叫重定向。

2)编程： response.sendRedirect(String url);

3)需要注意的问题在重定向之前，不能够有任何的输出；如果 response 缓存当中有数据，在重定向 之前，会自动清空。

4)重定向的特点：

a,地址任意

b,浏览器地址栏地址会变化（即变化为跳转之后的地址）。

12、系统异常处理：

程序在执行过程当中，发生了不可恢复的错误（程序不能够处理，需要人工参与），这种类型的错误称为系统异常。

处理方式：

step1 throws 系统异常。

step2 在 web.xml 中，配置一个错误处理页面。

```
<error-page>
    <error-code>500</error-code>
    <location>/error.html</location>
</error-page>
```

13、Servlet 容器如何处理请求资源地址：

比如 http://ip:port/appname/abc

step1, 依据 appname，找到对应的应用；然后，依据 web.xml 中的 urlpattern 与/abc 匹配。

step2,匹配过程:

a,精确匹配

b,通配符匹配:即使用"*","*"代表任意的字符串。

c,后缀匹配: 以"*."开头,后面加上任意的字符串。比如"*.do"。匹配任意以.do 结尾的请求。

step3 如果匹配不成功，此时，会查找对应的资源，,认为这是一种静态资

源，若找到，则返回，找不到，返回 404。

14、servlet 的生命周期及核心的类与接口（重点）：

1、生命周期是什么？

Servlet 容器如何去创建 Servlet 实例、分配资源、调用其方法、销毁实例的整个过程。

2、Servlet 的生命周期分为四个阶段：

1)创建 Servlet 对象第一次请求到来时，会创建 Servlet 对象（默认）。

2）调用 Servlet 对象的 init()方法，初始化 Servlet 的信息，init()方法只会在创建后立即被调用一次；而不是每一个用户请求都会调用该方法；

3）相应请求，调用 Service()或者是 doGet,doPost()方法来处理请求，这些方法试运行在多线程的状态下。

4)在长时间没有调用或者是服务器关闭,会销毁 Servlet 对象，同时在销毁 servlet 对象之前调用 destory()方法。

3、通过 web.xml 配置 servlet 对象的创建时机：

在 <Servlet></Servlet> 标签中使用 <load-on-startup> 数字 </load-onstartup>，表示服务器启动时创建，并依照数字的大小按顺序创建，小数字先加载。

4、通过 web.xml 配置 servlet 初始化参数：

通过在<Servlet></Servlet>标签中使用

```
<init-param>
    <param-name>…</param-name>
    <param-value>…</param-value>
</init-param>
```

标签配置初始化参数，并可以通过 ServletConfig 对象调用 getInitParameter(String name)方法来得到初始化参数；当没有要取的参数时，会返回 null。

5、生命周期相关的类与接口：

a,Servlet 接口：

初始化方法：init(ServletConfig config);

销毁 Servlet 实例之前，执行该方法：destroy();

处理请求：service(ServletRequest req,ServletResponse res);

b,GenericServlet 抽象类：

实现了 init(),destroy()方法。

c,HttpServlet 抽象类： 继承 GenericServlet，实现了 service() 方法 service(HttpServletRequest req, HttpServletResponse res);该方法

依据请求方式：如果是 get 请求，会调用 doGet(req,res)方法，否则，调用 doPost(req,res)方法。所以开发一个 Servlet，可以选择 override HttpServlet 的 service() 方法，或者 override HttpServlet 的 doGet(),doPost()方法。

d,HttpServletRequest 是 ServletRequest 的子接口，HttpServletResponse 是 ServletResponse 的子接口。

15、JSP （Java Server Page）：

1)什么是 jsp

java server page:java 服务器端页面技术；sun 制订的一种服务器端动态页面生成技术规范，jsp 页面，主要由 html 元素加上少量的 java 代码构成；该页面文件要以.jsp 为后缀保存，不需要开发人员去编译。jsp 开发动态页面，不再需要使用 out.println()语句，所以，相对 Servlet，要简单得多。jsp 其本质，是用来简化 servlet 中有关表示逻辑生成的一种技术。jsp 从某种意义上讲，其实就是 servlet。

2)jsp 的组成部分

a,html(包括 html,css,js),直接写在.jsp 文件当中即可。

b,java 代码:

java 代码片断:<% java 代码 %>

jsp 表达式:<%= %>

c,指令

告诉 jsp 引擎，在将.jsp 源文件转换成.java 源文件时，做一些额外的处理。比如，导包、告诉 jsp 引擎，.jsp 源文件的编码是什么等。

语法:<%@ …%>

1.<%@page%>指令:

import 属性:导包

<%@ page import="java.util.*,java.text.*"%>

也可以

<%@ page import="java.util.*"%>

<%@ page import="java.text.*"%>

pageEncoding 属性：告诉 jsp 引擎，.jsp 源文件的编码是什么。

contentType 属性：两个作用，一是生成 contentType 消息头，另外一个，输出到 response 对象上时，使用的编码格式。

2. <%@ include file =" relativeURLspec "%>

直接引入内容，这些内容会直接嵌入到页面中。

d,隐含对象

在.jsp 源文件中，不用声明，就可以直接使用的对象。叫隐含对象。因为.jsp 源文件在转换成.java 源文件时，jsp 引擎会自动添加隐含对象的创建过程的相关代码。所以，可以直接使用 out、request、response 等。

3)jsp 是如何执行的？

jsp 引擎(容器当中的一个特定模块)将.jsp 源文件转换成.java 源

文件(该 java 文件其实是一个 Servlet)。然后编译成.class 文件，最后执行 service()方法。所以，可以这样认为，jsp 的实质就是一个 Servlet。

4).jsp 源文件如何转换成.java 源文件。

a,html 转换到 servlet 的 service()方法里，使用 out.write()输出。

b, <% %>java 代码转换到 servlet 的 service()方法里，照搬。

c,<%=%> jsp 表达式转换到 servlet 的 service()方法里，使用 out.print()输出。

16、转发（重点）：

1）什么是转发?

一个 web 组件（servlet/jsp）将未完成的处理交给另外一个 web 组件继续完成。转发所涉及的各个 web 组件可以共享 request 和 response 对象。

2）编程：

step1 绑订数据到 request 对象上。

request.setAttribute(String name,Object obj);

request.removeAttribute(String name);

Object request.getAttribute(String name); //如果绑订名不存在，则返回 null。

step2 获得转发器

RequestDispatcher rd =request.getRequestDispatcher(String url);

step3 转发

rd.forward(request,response);

servlet:负责业务逻辑处理（包括数据访问）。

jsp：负责生成界面。

3）需要注意的问题：

在转发之前，response 缓存的数据会被清空。

17、转发与重定向的区别：

(1)转发的目的地只能是同一个应用内部的各个组件之间；而重定向的目的地是任意的。如果想实现组件之间的跳转，如果是同一个应用的内部，应该用转发（因为效率更高）；如果组件之间要共享 request 对象，也应该用转发。

（request/response 对象的生存时间：在一次请求与响应期间存在，超过该范围，容器会销毁该对象）。

(2)浏览器地址的地址：转发是不变的，而重定向是变的。

(3)转发的各个组件可以共享 request 对象，而重定向不行。

(4)转发是一件事情未做完，而重定向是一件事情已经做完。

18、状态管理：

1）什么是状态管理？

客户端访问服务器时，需不需要将客户端的相关数据（对服务

器的操作，所涉及的 数据）记录下来。如果不记录，就是无状态的，反之，则是有状态的。所谓状态管理，就是将客户端的相关数据记录下来。

2）如何进行状态管：

http 协议是一个无状态协议，http 协议有这样一个特点：客户端发起一次请求，服务器响应之后，立即关闭连接，当客户端再次发请求时，需要重新建立一个新的连接。也就是说，服务器端不会记录是哪一个客户端发起了请求；这种方式，不能够进行状态管理。如果要实现状态管理，则需要使用以下两种方式来进行状态管理：

a,客户端保存状态

当客户端（一般是浏览器）访问服务器时，服务器将相关的数据以 cookie 的形式发送给浏览器，浏览器将 cookie 中的数据保存下来。这样，当浏览器下次访问服务器时，会自动将 cookie 中的数据发送给服务器。

b,服务器端保存状态 将用户的状态记录在服务器端，即 session 技术。

19、cookie 技术（重点）：

1)什么是 cookie？

cookie 是一种在客户端（浏览器）维护用户状态的状态管理技术。浏览器在访问服务器时，服务器发送一个消息头(set-cookie)及对

应的值给浏览器，浏览器会将该消息头对应的值保存到内存或者硬盘上。当浏览器下次访问服务器时，会将这些值再发送给服务器。通过这种方式来维护用户的状态（也就是说，将用户的状态写到了消息头对应的值里面）。

2)如何创建 cookie？

Cookie cookie = new Cookie(String name,String value);

response.addCookie(cookie); cookie 的值只能是 ascii 字符，需要将中文转化成 ascii 表示；保存 cookie 时，使用 URLEncoder.encode();查询 cookie 时 URLDecoder.decode();

3) 如何查询 cookie？

Cookie[] request.getCookies();如果没有 cookie，则返回 null。

Cookie.getName():cookie 名字 Cookie.getValue():cookie 的值

4)cookie 的生存时间：

默认情况下，cookie 会被浏览器保存到内存里，当浏览器关闭，cookie 会被删除。可以通过调用 Cookie.setMaxAge(int seconds):设置 cookie 生存时间

seconds < 0： 默认情况

seconds > 0: cookie 会被保存到硬盘上，浏览器关闭，cookie 有可能还存在。

seconds = 0: 删除 cookie，如：

```
Cookie c = new Cookie("username","");
c.setMaxAge(0);
```

response.addCookie(c);

5)cookie 的路径问题：

浏览器在向服务器发送请求之前，先比较 cookie 的路径与要访问的服务器的地址。只有符合要求的服务器地址（即要访问的地址是 cookie 的路径的子路径或者相等），才会携带对应的 cookie。

cookie 的路径，在默认情况下，与保存 cookie 的组件的路径是一致的。可以通过 Cookie.setPath()来设置 cookie 的路径。

Cookie.setPath("/appname"):此时，该 cookie 可以被 appname 包含的所有组件访问。 Cookie.setPath("/"):此时，该 cookie 可以被部署在同一台服务器上的所有应用中的组件访问。

比如：

如下图的服务器文件结构

+ appname

| + a

| | *saveCookie.jsp

| | *getCooki e2.jsp

| | + b

| | | *getCookie3.jsp

| *getCookie1.jsp

保存 someKey 的组件是 saveCookie.jsp,此时 cookie 的路径是 /appname/a 访问 someKey 的组件是 getCookie1.jsp,此时，地址是 /appname,因为/appname/a 是/appname 的子路径，所以，浏览器不

会发送 cookie 给服务器。访问 someKey 的组件是 getCookie2.jsp,此时地址是/appname/a,因为/appname/a 与/appname/a 一致，会发送 cookie。访问 someKey 的组件是 getCookie3.jsp 此时，地址是/appname/a/b,也会发送 cookie。 6)cookie 的限制：

a,用户可以禁止 cookie。

b,cookie 的数量也有限制（浏览器保存的 cookie 的数量，比如 ie,大约 300）。

c,cookie 的大小有限制（大约是 4k）。

d,cookie 的值只能是字符串，并且需要考虑编码问题。

7)cookie 的常见使用：

a,记住用户的使用习惯；

b,自动登录；

c,购物车（与 session 一起实现）。

20、session 技术（重点）：

1)什么是 session?

在服务器端维护用户状态的状态管理技术。浏览器在访问服务器时，服务器会创建一个对象（session 对象），然后，在默认情况下，服务器会将 session 对象对应的 sessionId(每一个 session 对象都有唯一的 sessionId)以 cookie 机制发送给浏览器。浏览器下次访问服务器时，会将 sessionId 发送给服务器，服务器依据 sessionId 查找对应的

session 对象。通过这种方式，来维护用户的状态（可以这样理解，相当于每一个用户在服务器端有一个与之对应的 session 对象，用户的状态就保存在该对象里）。

2)如何创建 session?

方式一： HttpSession session = request.getSession();

方式二: HttpSession session = request.getSession(boolean flag);

当 flag 为 true 时:

当请求到达服务器时，服务器会检查请求中是否包含 sessionId,如果没有，则创建一个 session 对象。如果有，依据 sessionId 查找对应的 session 对象，如果找到，则返回，如果找不到，则创建一个新的 session 对象。

当 flag 为 false 时:

当请求到达服务器时，服务器会检查请求中是否包含 sessionId,如果没有，返回 null。如果有，依据 sessionId 查找对应的 session 对象，如果找到，则返回。

request.getSession()与 request.getSession(true)完全一样。

3)session 的几个常用方法：

session.setAttribute(String name,Object obj);//向 session 中设置属性

Object session.getAttribute(String name);//找不到，返回 null。

session.removeAttribute(String name);

String session.getId();//获得 sessionId

4)session 对象的生存时间：

对于不同的服务器，session 对象会有不同的默认生存时间，比如 tomcat，默认是 30 分钟。超过指定时间，不去访问 session 对象，服务器会删除 session 对象。 可以设置最大不活动时间 session.setMaxInactiveInterval(int seconds);也可以使用配置文件，设置超时限制。对于整个服务器设置：在 tomcat_home/conf/web.xml

<session-config>

<session-timeout>30</sessiontimeout>

</session-config> 也可以，针对单个应用来设置：在 WEB-INF/web.xml

<session-config>

<session-timeout>30</session-timeout>

</session-config>

5)删除 session 对象： session.invalidate();

6)session 的常见使用(参照课上代码复习)：

a、session 验证：

step1 在登录成功以后，向 session 绑订相应的数据:session.setAttribute();

step2 在需要受保护的页面或者资源，添加判断:session.getAttribute()；如果为 null，说明没有登录，跳转到登录页面。

b、验证码：（一般使用 servlet 生成）

step1 设置消息头:response.setContentType("image/jpeg");

step2 使用 java.awt 提供的 java 类生成图片;

step3 压缩图片并输出

step4 使用验证码；在生成验证码时，在 session 中保存该验证码，并以图片的形式将验证码发送给客户端（即浏览器）；客户端输入的验证码与=session 中保存的验证码作比较。

step5 客户端 jsp 中点击“看不清，换一个”请求要每次都不一样，请求的后面要加 Math.random()或 new Date()等手段。

c、购物车：

1)编码过程：

step1 创建购物车购物车、商品实体；

step2 完成实体对应的 DAO 层编码；

step3 在 servlet 中把 session 中对购物车进行业务操作；

step4 在 jsp 中把 session 中购物车里的信息打到页面中和发出请求给后台的 servlet。

2)序列图(基本用力流程图)：

21、URL 重写实现 session 技术：

当用户禁止 cookie 之后，如果实现 session 机制（如果实现 sessionId 的跟踪），需要使用 url 重写机制：

简单地说，所谓 url 重写，指的是在 url 地址后面添加 sessionId；具体来讲，要访问某个组件（该组件如果需要使用 session 机制），不

能够直接在浏览器地址栏输入该组件的地址，而应该使用服务器生成的该组件的地址（该地址中包含了 sessionId）。当使用服务器生成的地址去访问某个组件时，会将 sessionId 传递给服务器，服务器通过 sessionId 找到对应的 session 对象。

使用到的方法：

在链接，表单提交中使用：

response.encodeURL(String url); 在重定向时，使用：

response.encodeRedirectURL(String url);

比如:

response.sendRedirect(response.encodeRedirectURL("xxxxx"));

22、路径问题：

1)路径问题所涉及的四种情况：

a,链接

b,表单提交

c,重定向

d,转发

2)相对路径: 路径不以"/"开头。

3)绝对路径 路径以"/"开头。

对于 a,链接 b,表单提交 c,重定向三种情况，从应用名开始。 对于 d,转发从应用名后开始。 通过 String request.getContextPath():返回

/appname，实现绝对路径。

23、过滤器：

1)什么是过滤器：

Servlet 规范当中定义的一种特殊的类，该类的作用：对容器的调用过程进行拦截；在过滤器类当中，可以实现一些通用的逻辑，比如进行编码处理、权限管理、安全处理、session 验证等等。

2)编程实现：

step1 写一个 java 类，实现 Filter 接口。

step2 在 doFilter 方法里，实现过滤的逻辑。

step3 配置(web.xml)。

3)过滤器的优先级：

<filter-mapping>的先后顺序决定了过滤器调用的先后顺序。

4)给过滤器添加配置参数：

通过在<filter></ filter >标签中使用

<init-param>

<param-name>…</param-name>

<param-value>…</param-value>

</init-param> 可以给过滤器添加相应的参数；然后，使用 FilterConfig.getInitParameter(name)读取。

5)过滤器的优点

a,将一类相同或相似的操作集中封装到过滤器当中，方便代码的维护。

b,可以实现代码的"可插拔性"。即增加或者减少某个模块，只需要添加相应的类及配置文件。

24、监听器：

1)什么是监听器？

Servlet 规范当中定义的一种特殊的类，其作用是，监听两类事件：

a,当容器创建或者销毁 ServletContext,HttpSession,ServletRequest 三种对象时，产生的事件。

b,对 ServletContext,HttpSession,ServletRequest 执行 setAttribute(),removeAttribute()产生的事件。

2)ServletContext 接口：

a,当容器启动时，容器会为每一个已经部署的应用，创建一个符合 ServletContext 接口的实例，该实例称为 Servlet 上下文。该实例会一直存在，除非容器关闭或者对应的应用被卸载。

b,如何获得 Servlet 上下文:

方式一：使用 GenericServlet 提供的方法：getServletContext();

方式二：session.getServletContext();

方式三:

ServletConfig.getServletContext();

c, ServletContext 作用:

1、 绑订数据:

setAttribute(String name Object obj);

removeAttribute(String name);

Object getAttribute(String name);

2、读取全局参数: 全局参数，指的是，在 web.xml 当中，使用 Context-param 配置的参数； String getInitParameter(String paraName);依据逻辑路径返回物理路径 String getRealPath(String path);

3) 监听器编程：

step1 写一个 java 类，实现相应的监听器接口（如：

ServletContextListener、HttpSessionListener）。

step2 在接口方法里，实现监听逻辑。

step3 配置(web.xml)；比如，要统计在线人数。

25、上传文件

step1 在表单中，添加 enctype 属性,即

<form action="" method="post" enctype="multipart/form-data">

enctype="multipart/form-data",其作用是，浏览器以原始的二进制流向服务器发送数据。在服务器端，不能用 request.getParameter()

获得数据。只能使用 InputStreamrequest.getInputStream();通过分析该流，来获得数据。同时，浏览器段可以使用 <input type="file" name="xxx"/>来接收要上传的文件。

step2 一般在服务器，不会直接使用 request.getInputStream()来获得流，并分析流。而 是使用相应的工具包来分析。

比如，apache 提供的 common-fileuploader.jar（同时需要 common-io.jar）。

服务器端 apache 技术编程实现：

step1 为解析类提供解析配置：

DiskFileItemFactory dfif =new DiskFileItemFactory();（默认解析配置）

step2 创建解析器：

ServletFileUpload sfu = new ServletFileUpload(dfif);

sfu.setFileSizeMax(1024*50);（设置最大上传文件大小）

step3 获取上传文件：

List<FileItem> items = sfu.parseRequest(request);（获得解析文件）

step4 对上传文件进行处理。

26、Servlet 线程安全：

1）为什么 Servlet 会有线程安全问题？

当客户端请求到达服务器时，服务器会启动一个线程，该线程会调用 Servlet 对象的 service()方法来处理请求。因为在服务器当中，某个 Servlet 类型只会有一个对应的对象，所以，就有可能多个线程在某一时刻同时访问某个 Servlet 对象；这样就会出现线程安全问题。

2）如何解决线程安全问题：

a,加锁，即对 service()方法或者 service()方法里的代码块加锁；使用 synchronized 关键字。

b,实现 SingleThreadModel 接口；不建议使用，因为每当一个请求到达时，容器会为每一个线程分配一个对应的实例。因为创建的实例过多，服务器压力比较大。

c,对于 Servlet，不要去修改其属性；可以把想要修改的属性放在 session、ServletContext 中。

27、中文问题（2）：

1)表单处理（参考之前的知识，上文的中文问题(1)）。

2)链接地址包含中文：

w3c 规定，链接地址，不能出现一些特殊的字符，比如中文，一些拉丁字母等。如果包 含了这样的字符，浏览器会对其进行编码。firefox,ie 都采用 utf-8 进行编码；服务器默认采 用 iso-8859-1 解码 。对于 tomcat ，可 以修改 tomcat_home/conf/server.xml 文件 添加 URIEncoding="utf-8"

3)链接地址包含中文参数：

浏览器会对链接地址中的中文参数进行编码，具体采用哪一种编码，要看打开该页面的编码是什么。如果链接地址包含中文，可以使用 URLEncoder.encode()对中文参数进行编码。

28、JSP 总结（主要组成部分）：

1)html(包括 css,js)

2)java 代码

a, <% %> java 代码片断

b, <%= %> jsp 表达式:

c, <%!= %> jsp 声明：声明的可以是成员变量，方法或内部类；声明中的变量会作为 servlet 的属性，而方法会作为 servlet 对应的方法。

3)指令

语法:<%@ 指令名 属性=”属性值”%>

a,page：

pageEncoding: 设置页面显示的字符集。

contentType: 设置文件格式和编码方式。

import: 导入需要用到的类（可导入多个包，用“,”分割）。

session:true(缺省)/false,当值为 true 时，表示生成的 Servlet 源代码当中，会包含 HttpSession session = request.getSession()。也就是，

可以使用 session 隐含对象。

errorPage:指定错误处理页面。

isErrorPage:true/false(缺省)当前页面是否为错误处理页面。

isELIgnored:true/false。是否忽略 el 表达式。

extends 本页面生成的 servlet 所继承的类。

language 本页面使用的语言,默认是 java ,通常不写。 buffer 本页面的输出是否支持缓冲（缓冲大小）。 autoFlush 缓冲区是否自动刷新。

b,include：

file:将 file 所指定的文件源代码插入到指令所在的位置。

c,taglib：

（导入标签库）：

uri: 标签库引用路径。

prefix:标签前缀名。

4)隐含对象

生命周期和可见范围不同，范围如下：

pageContext -->request -->session-->application

生命周期和可见范围越来越长

5)活动元素(action element)

a.<jsp:forward page="">转发,page 属性指向转发的页面。

b.<jsp:include>包含,与指令不同，<%@include file=""%>是在.jsp 源文件转换成.java 源文件时，将 file 指定的源文件插入到指令所在

的位置。而<jsp:include page=""/>是在 jsp 实例已经运行了，然后去调用 page 所定的 jsp；如果要向被包含的文件传参，应该使用<jsp:include/>。

c.<jsp:param>用于传参。

d.<jsp:useBean id="" scope="" class="">jsp 引擎会检查 scope 指定的范围有没有一个绑订名 为 id 所指定的名称的对象；如果没有，则依据 class 所指定的类名创建一个对象，并绑订到 scope 所指定的范围。

<jsp:getProperty>访问指定的 bean 的属性值。

<jsp:setProperty>用于给指定的 bean 的属性赋值

6)注释

JSP 注释：<%-- … --%>，翻译阶段消失。

JAVA 注释：//、/**/、/***/，编译阶段消失。

HTML 注释：<!-- … -->, 不会消失。采用<!---->,如果注释的内容是 java 代码，会执行。

29、jstl 和 el 表达式：

1)jstl 是什么?

java standard tag lib:java 标准标签库；sun 公司制订的一种特殊标记，使用该标记可以避 免在 jsp 页面当中直接编写 java 代码；使用标签，可以使得 jsp 文件更容易维护。

2)el 表达式是什么？

最早用于标签技术，是一套计算规则，其计算的结果给标签的属性赋值。现在， el 表 达式也可以将其结果直接输出到页面上。

3)el 表达式的语法

${el 表达式}

a,访问 bean 的属性： ${user.name}:jsp 引擎会依次从 pageContext,request,session,application 这四个隐含对象当中查找绑订名为"user"的对象，即执行类似:request.getAttribute("user")；如果找到了，则不会向下继续查找；找到后，会执行 getName()方法，并输出；如果找不到，输出""。

${pageScope.user.name}:指定从 pageContext 中查找，如果找不到，不再从其它隐含对象 中查找了；此外，还可以是 requestScope,sessionScope,applicationScope 。另外，还可以使用 ${user["name"]}或者${user[str]}；这种写法，允许属性是一个变量。

b,获得请求参数值：

${param.name}:等价 request.getParameter("name");

${paramValus.interest}:等价 request.getParameterValues("interest");

c,计算表达式的值，直接输出：

算术运算: +,-,*,/,%,对于"+",不能进行字符串的连接操作。

关系运算：

= =可以用 eq 表示；

!= 可以用 ne 表示；

< 可以用 lt 表示；

> 可以用 gt 表示；

<= 可以用 le 表示；

>= 可以用 ge 表示。

逻辑运算：

&&,||,!，也可以用相应的字符串来表示，比如 "&&" 可以用"and"表示。

empty 运算：

以下三种情况，结果为 true：

空字符串、集合内容为空、找不到对象、找到了 null。

d,计算表达式的值,用于给标签的属性赋值（如：嵌套 jstl 标签同时应用）。

4)jstl 使用步骤

step1 导入标签相关的 jar 包（如果使用 javaee5.0 以上的版本，则不用导入）。

step2 在 jsp 文件中，使用 taglib 指令引入标签。

step3 使用标签。

5)核心标签

a,<c:if test="" var="" scope="">

test 的属性值如果为 true，则执行标签体的内容；test 属性一般使用 el 表达式进行计算； 可以将 test 属性值绑订到 scope 指定的范围，绑订名由 var 指定。

例：

```
<c:if test="${!(empty user.age)}">
    <h1>hello</h1>
</c:if>
```

b, <c:choose> <c:when> <c:otherwise>

<c:when>,<c:otherwise>必须写在<c:choose>里面；

<c:when>有一个属性 test，当 test 值为 true,执行标签体的内容 ；

<c:when>标签可以出现多次 ，表示多种不同的情况 ；

<c:otherwise>只允许出现一次，当各种条件均不满足，执行该标签体的内容。

例：

```
<c:choose>
    <c:when test="${param.age<18}">
    <h1>you is a child<h1>
    </c:when>
    <c:otherwise>
    <h1>you is a young person</h1>
</c:otherwise>
</c:choose>
```

c,<c:forEach>

用于选代集合；items(可以使用 el 表达式)用于指定要选代的集

合；jsp 引擎会从集合中每次取出一个对象，存放到 pageContext 里（相当于 pageContext.setAttribute("user",user)） varStatus 指向一个对象（该对象封装了迭代的相关信息）；比如: status.index 表示迭代的下 标(从 0 开始),status.count 表示是第几次迭代。

例：

```
<c:forEach var="book" items="${store.books}" varStatus="status">
    <h1>${book.parice}</h1>
</c:forEach>
```

类似 for 循环：

```
<c:forEach begin="1" end="5" step="1">
    <h1>hello</h1>
</c:forEach>
```

d, <c:out value="" default="" escapeXml="">

主要用于输出 el 表达式的值；default 表示，当查找不到对应的对象时，输出的缺省值。 escapeXml 为 true，表示一些特殊字符，比如"<",">","&",""等使用相应的实体来代替。

e, <c:remove var="" scope="">

解除绑订，注意范围。

f, <c:set var="" value="" scope="">

将对象绑订到某个范围。

g, <c:catch var="">

处理异常,var 指定绑订名，范围只能是 pageContext。

例:

```
<c:catch var="msg">
    <%Integer.parseInt("123");%>
</c:catch>
<c:if test="${!empty msg}" var="rs" scope="request">
    error:${msg}
</c:if>
```

h, <c:url value="">

作用 1：当用户禁止 cookie 以后，会自动在地址后添加 sessionId。

作用 2：生成绝对地址。

i, <c:import url="引入内容的 url ">

相当于<jsp:include/>。

j, <c:redirect url="">

重定向，并且，当用户禁止 cookie；以后，会自动在地址后添加 sessionId。

6)el 函数

el 函数用于对 el 表达式的结果进行计算，jsp 引擎会将其计算结果输出。

如：${fn:length("abc")}，取得字符串、集合等的长度。

7)如何自定义 el 函数

step1 写一个 java 类，定义 public static 方法，每一个方法对应一个函数。

step2 在.tld 文件当中，描述该函数；可以参考 fn.tld 文件编写。该文件可以放在两个地 方：放在 WEB-INF 或是 WEB-INF 下的子文件夹均可；也可以将 tld 文件打到.jar 包当中（此 时，tld 文件必须放到 META-INF 下）。

step3 使用 taglib 指令导入函数。

例：xxx.tld 中的配置

```
<function>
    <name>reverse</name><!—jsp 中使用的函数名-->
    <function-class>jsp2.examples.el.Functions</function-class><
!--函数所在的 java 类-->
    <function-signature>java.lang.String reverse( java.lang.String )
</function-signature>
<!--函数原型，也就是函数的返回值类型，函数名，参数表，注意要
写类型的全名-->
</function>
```

8)自定义标签

step1 写一个 java 类，继承 SimpleTagSupport 类；标签类的属性与标签的属性对应，并 且有相应的 get/set 方法。

step2 override doTag()方法，在该方法当中，实现相应的处理逻辑。一般在 doTag()方法 里，使用

```
PageContext pc = (PageContext)getJspContext();
```

获得 pageContext 对象，通过该对象， 可以查找到其它隐含

对象。

step3 在.tld 文件中，描述标签。<body-content>元素：

JSP:简单标签不支持该值（1.2 版本时的），表示在标签体里，可以使用 java 代码。 empty:该标签没有标签体。

scriptless:标签有标签体，但是标签体不能够出现 java 代码。

step4 使用 taglib 引入标签，便可以使用自定义的标签了。

例：xxx.tld 中的配置

```
<tag>
    <name>right</name><!—jsp 中使用的标签名-->
    <tag-class>mytag.RightTag</tag-class><!--签名所在的 java 类
-->
    <bodycontent>scriptless</body-content><!—设置标签体-->
  <attribute>
    <name>username</name><!—设置标签属性名-->
    <required>true</required><!—设置标签属性是否必须设置-->
    <rtexprvalue>true</rtexprvalue><!—标签属性赋值的时候是
否可以使用表达式->
  </attribute>
</tag>
```

30、中文问题（数据库乱码）

1、mysql 建库时设置默认编码：

GBK: CREATE DATABASE test2 DEFAULT CHARACTER SET gbk COLLATE gbk_chinese_ci;

UTF8: CREATE DATABASE test2 DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;

2、若是已经建好的数据库可使用：

alter database opensource default character set 'utf8';

3、建表时设置默认编码：

create table t_pic(id bigint primary key auto_increment,picName varchar(100), userId bigint) ENGINE=InnoDB DEFAULT CHARSET=utf8;

4、若是已经建好的数据表可使用：

alter table t_books character SET 'utf8';

5、查看时更改终端编码：

GBK: set names gbk

UTF8: set names utf8

6、连接数据库的 url：

URL 后加：?useUnicode=true&characterEncoding=utf8

31、MVC

1、什么是 mvc?

MVC 是一种软件架构思想，是将一个软件的组成部分划分成模型(model)、视图(view) 和控制器(controller)。其中，模型负责业务逻辑(主要包括业务数据的加工处理规则，另外， 还有，为保证处理所需要的一些基础服务，比如事务、安全、日志等等)的处理。视图负责 展示模型处理之后的结果，并且提供相应的用户界面或接口，也就是说，视图负责表示逻辑。 控制器负责协调模型和视图。协调指的是，控制器将视图与模型解藕，这样做的好处是，视 图或者模型发生改变，不会相互影响。

*模型 负责业务逻辑(model)

*视图 负责表示逻辑(view)

*控制器 负责流程控制逻辑(controller)

2、使用 mvc 的好处：

业务数据的加工规则以及保障业务逻辑能够正常执行所添加的一些基础服务，比如事务、安全、日志、性能等等。 业务数据的展现以及用户操作的界面。

视图向控制器发送请求，控制器依据一定的规则，调用对应的模型来处理请求；模型处理的结果发送给控制器，控制器选择合适的视图，生成相应的界面，提供给用户。

1)一个模型可以使用多种不同的视图来展现其处理之后的结果。因为模型只负责返回数 据（也就是说，返回的数据是与具体的显示方式无关的）。

2)模型开发完之后，可以立即测试（比如，在开发一个 web 应

用时，如果业务逻辑写在 servlet 里面，则为了测试业务逻辑是否正确，需要部署整个应用，并且启动服务器才能测试，如果，将业务逻辑写在一个 java 类（即模型）里，则，可以直接测试该 java 类，不 必启动服务器）。

3)方便分工协作。

4)代码好维护。

3、在 web 应用当中，如何使用 mvc？

model:使用 java 类(javabean)或者被容器管理的 javabean(比如 ejb,spring 容器中的 javabean)来封装。（也可以使用其它技术，比如 FreeMarker 模板技术等等）。

controller：使用 Servlet/Filter；所有的请求都发送给控制器，控制器依据请求的内容调 用不同的 model 来处理（当然，也可以调用同一个 model 的不同方法）；控制器依据 model 返回的结果，来选择不同的视图展示结果数据。

4、mvc 的缺点

为了代码的可维护性，mvc 增加了大量的类，也就是增加了整个系统的代码量；采用 mvc 设计的系统，需要良好的设计。所以，对于大型的应用系统，如果需要代码的良好的可维护性与扩展性、方便测试，应用使用 mvc 模式来开发。反之，则不一定要使用 mvc。

32、Ajax

1、什么是 ajax?

asynchronous javascript and xml:异步的 javascript 和 xml。为了解决传统的 web 应用当中 “请求-处理-等待-响应”的弊端而创建的技术，其实质是：使用 javascript 调用浏览器内置 的一个对象（XmlHttpRequest)异步向服务器发送请求 ，服务器返回 xml 或者 text 给 XmlHttpRequest，然后，javascript 使用服务器返回的数据更新页面。在整个过程当中，页面 没有任何的刷新。

2、XmlHttpRequest 对象:

1)如何创建：

w3c 没有标准化这个类型，要区分浏览器:

```
var xmlHttpRequest = null;
if((typeof XMLHttpRequest)!= 'undefined') {
    xmlHttpRequest = new XMLHttpRequest();
}else {
    xmlHttpRequest = new ActiveXObject('Microsoft.XMLHttp');
}
```

2)该对象的常用属性：

a, readyState: xmlHttpRequest 对象，在与服务器通讯过程当中，会经历不同的状态，每 当状态发生改变，可以通过该属性获得状态值。

0-未初始化，请求没有发出（在调用 open() 之前）。

1-请求已经建立但还没有发出（调用 send() 之前）。

2-请求已经发出正在处理之中（这里通常可以从响应得到内容头部）。

3-请求已经处理，响应中通常有部分数据可用，但是服务器还没有完成响应。

4-响应已完成，可以访问服务器响应并使用它。

b,status:获取服务器返回的状态码(比如 200,500)； (下面是常见的状态码)。

200 OK 一切正常，对 GET 和 POST 请求的应答文档跟在后面。

202 Accepted 已经接受请求，但处理尚未完成。

300 Multiple Choices 客户请求的文档可以在多个位置找到，这些位置已经在返回的文档内列出。如果服务器要提出优先选择，则应该 Location 应答头指明。

404 Not Found 无法找到指定位置的资源。

500 Internal Server Error 服务器遇到了意料不到的情况，不能完成客户的请求。

503 Service Unavailable 服务器由于维护或者负载过重未能应答。

例如，Servlet 可能在数据库连接池已满的情况下返回 503。服务器返回 503 时可以提供一个 Retry-After 头 。

504 Gateway Timeout 由作为代理或网关的服务器使用，表示不能及时地从远程服务器获得应答。

505 HTTP Version Not Supported 服务器不支持请求中所指明的 HTTP 版本。

c,responseText:获得服务器返回的处理结果,该结果以文本的形式返回。

d,responseXml:获得服务器返回的处理结果,该结果以符合 dom 规范的对象返回。

e,onreadystatechange:注册一个监听器（绑订一个回调函数或者说是绑订事件处理代码）。作用是，当 readyState 的值发生改变，会产生相应的事件，该事件会触发回调函数。一般在 该回调函数当中，使用服务器返回的数据更新页面。

3、如何使用 ajax 编程？

step1 获得 XmlHttpRequest 对象。

```
var xhr = ...
```

step2 使用该对象，向服务器发送请求。

方式一：get 请求

```
xhr.open("get","valiusername.do?name=zs",true);
//f1 函数用来处理服务器返回的数据。
xhr.onreadystatechange=f1;
xhr.send(null);
```

方式二：post 请求

xhr.open("post","abc.do",true);

xmr.setRequestHeader("Content-Type","application/x-www-form urlencoded");

xhr.onreadystatechange=f1; xhr.send("name=zs");

step3 在服务器端，编写代码，用来处理请求。一般不需要返回完整的页面，只需要返 回部分的数据，一般是普通的文本。

step4 在回调函数当中，处理服务器返回的结果。

```
function f1(){
    if(xhr.readyState == 4){
        var txt = xhr.responseText;使用 javascript 更新页面(dom 操作)。
    }
}
```

例：

javascript 实现 ajax-get 请求：

```
function valiusername(){
var xhr = getXmlHttpRequest();
var url = 'valiusername.do?username=' + $F('username') + '&age=22';
xhr.open('get',encodeURI(url),true);//对 url 进行编码，否则发送中文会乱码。
xhr.onreadystatechange=function(){
```

```
        If(xhr.readyState == 4){
            if(xhr.status == 200){
                var txt = xhr.responseText;
                $('username_msg').innerHTML = txt;
            }else{
                $('username_msg').innerHTML = '系统错误';
            }
            }else{
                $('username_msg').innerHTML = '正在验证...';
            }
        };
                xhr.send(null);
    }
```

后台 servlet 中的核心代码：

```
    PrintWriter out = response.getWriter();
    String username = request.getParameter("username");
    System.out.println("username:" + username); String xml = "";
    if(username == null || username.equals("")){
    xml +="用户名为空";
    }else{
        if(username.equals("zs")){
        xml +="用户名已经存在";
```

```
        }else{
            xml +="用户名可以使用";
        }
    }
        out.println(xml);
        System.out.println(xml);
}
out.close();
```

4、prototype 框架的简单使用:

$('id'):相当于 document.getElementById('id');

$F('id'):相当于 document.getElementById('id').value;

strip():除掉空格 var str = "abc ";str.strip();

5、ajax 应用中的中文问题：

当采用 get 方式向服务器发送请求时，ie 采用 gb2312/gbk 对发送的数据进行编码；而 firefox 采用 utf-8 来对发送的数据进行编码。可以使用 js 内置的一个函数 encodeURI 对发送的数据进行编码；该函数使用 utf-8 来编 码的。

33、JSON

1)什么是 JSON?

javascript object notation,它是一种轻量级的数据交换格式（相

对于 xml 而言，因为 JSON 的语法更简单，解析更容易）。json 一般用于客户端运行的 javascript 程序与服务器端程序进 行数据交换。

2)json 的本语法

a,创建对象：

{属性名 1:属性值 1,...属性名 n:属性值 n}

其中，属性名必须用引号括起来，属性值可以是 5 种基本类型，包括 string,number,boolean,null,undefined 如果是 string,也必须用引号括起来如果属性值是一个对象，也可以这样写：{属性名:{属性名：属性值}}

b,创建数组

[{属性名 1:属性值 1,...属性名 n:属性值 n},...{属性名 1:属性值 1,...属性名 n:属性值 n}]

3)如何将符合 json 语法的字符串转变成 js 对象:

可以使用 prototype 框架中的一个函数 evalJSON()。在 jQuery 框架中，也有一个函数 parseJSON()。

4)如何将 java 对象或者数组、集合转变成 json 字符串对于对象：

JSONObject obj = JSONObject.fromObject(Object);

obj.toString();

对于数组或者集合

JSONArray arr = JSONArray.fromObject(List); arr.toString();

5)在 ajax 应用当中，如何使用 json？

6)在 json 当中，如何处理 java 中的 Date 类型

step1 写一个转换器(写一个 java 类，实现 JsonValueProcessor 接口)。

step2 创建 JsonConfig 对象 JsonConfig config = new JsonConfig();

config.registerJsonValueProcessor(Date.class,dp);

step3 JSONObject.fromObject(obj,config)。

34、jQuery

1)jQuery 基础

a,jQuery 是什么？

jQuery 是一个开源的 js 框架，其基本设计思想，是将查找到的 dom 节点转换成一个 jQuery 对象，然后，通过操作 jQuery 对象提供的属性或者方法，来间接实现对 dom 节点的操作。

b,jQuery 编程的基本步骤：

step1 通过 jQuery 提供的选择器查找到相应的节点。

step2 对 jQuery 对象，调用相应的属性或者方法，来完成对 dom 节点的操作。

c,jQuery 对象如何转变成 dom 对象

方式一： $obj.get(0);

方式二： $obj.get()[0];

d,dom 对象转变成 jQuery 对象$(obj) 或者 jQuery(obj)；因为

$其实是 jQuery 函数的别名。e,如何让 jQuery 框架与其它框架共享,如何避免$冲突。

```
var $a = jQuery.noConflict();//将 jQuery 取别名
```

2)jQuery 选择器

jQuery 选择器模仿 css 选择器语法，来查找节点，css 选择器的作用，查找到节点后， 施加样式，而 jQuery 选择器的作用，是查找到节点后，施加行为。

结合课上老师的 chm 文档学习。

3)jQuery 的 dom 操作

a)、查询：利用选择器查找到节点，使用：

html():输出或者设置节点之间的 html;

text():输出或者设置节点之间的文本;

attr():输出或者设置节点的属性。

另外：

下拉列表,也可以使用 val()获得值单选、多选框要使用循环来获得选项值即使用 each()函数：语法结构:

```
var $obj = $(选择器);
$obj.each(function(index){
    //index 表示遍历的序号，从 0 开始。
    //$(this) 表示每一次遍历所取得的 jQuery 对象。
    // this 表示每一次遍历所取得的 dom 节点
});
```

b)、创建

$(html);

c)、插入节点

append():向每个匹配的元素内部追加内容

prepend():向每个匹配的元素内部前置内容

after():在每个匹配的元素之后插入内容

before():在每个匹配的元素之前插入内容

d)、删除节点

remove()

remove(selector)

empty():清空节点

e)、复制节点

clone()

clone(true): 复制同时将行为复制(注册在节点上的事件处理代码)

f)、属性操作

读取：attr('');

设置：attr('','')

或者一次设置多个 attr({"":"","":""}); 删除：removeAttr('')

g)、样式操作

获取和设置: attr("class",""),attr("style",""); 追加:addClass('')

移除:removeClass('')

或者 removeClass('s1 s2')

或者 removeClass()//会删除所有样式

切换样式：toggleClass,有该样式，则删除，否则，添加。 是否有某个样式 hasClass('')

读取 css('')

设置 css('','')或者

css({'':'','':''})//设置多个样式

h)、遍历节点

children():只考虑子节点，不考虑其它后代节点。 next():下一个兄弟

prev():前一个兄弟

siblings():所有兄弟

find(expr):依据表达式查找节点

4)事件处理机制

a)、事件绑订

bind(type,fn)

b)、绑订方式的简写形式

click(function(){});

c)、合成事件

hover(enter,leave): 模拟光标悬停事件

toggle(fn1,fn2...)：模拟鼠标连续单击事件

d)、事件冒泡

（1）获得事件对象

click(function(event){});

（2）停止冒泡

event.stopPropagation()

（3）停止默认行为

event.preventDefault()

e)、事件对象的属性

event.target:返回事件源（是 dom 对象） event.pageX/pageY

5)动画

a)、show() hide()

b)、show("slow"/"normal"/"fast"/100)

show("slow"/"normal"/"fast"/100) 100 是毫秒值

c)、fadeIn() fadeOut():淡入淡出

d)、slideUp() slideDown() :改变元素的高度

e)、animate(params,speed,callback)

6)操作类数组的方法：

说明：jquery 操作数组的方法，$()操作返回的如果是一个数组，可以使用如下方法来操作:

each(fn(i)):循环遍历每一个元素。

this 代表被迭代的 dom 对象。

$(this)代表被迭代的 jquery 对象。

eq(index)：返回 index+1 位置处的 jquery 对象。

index(obj)：返回下标，其中 obj 可以是 dom 对象或者 jquery 对象。

length：个数、长度。

get()：返回 dom 对象组成的数组。

get(index)：返回 index+1 个 dom 对象。

7)jQuery 对 ajax 的支持

a)、序列化元素：

serialize():将 jquery 对象包含的表单或者表单控件转换成查询字符串 serializeArray()：转换为一个数组，每个数组元素形如{name:fieldName,value:fieldVal}的 对象。

b)、三个方法

（1）load(url),将服务器响应插入当前 jQuery 对象匹配的 dom 元素之内。一般用于从服 务器获取静态的数据 (比如.html 文件),不支持返回 javascript。

（2）$.get(url,[data],[callback],[type])

url:服务器端要访问的组件的地址

data: 向服务器传递的参数 ，参数的格式必须是：

{"name":"zs","age",22}

callback:回调函数，其格式 function(data,statusText)

data:服务器返回给客户端的数据

statusText:服务器返回的状态

type:服务器返回的数据的类型包括如下几种：

xml:返回 xml 文档对象

html:返回的是一个 html

script:返回的是一个 js 脚本

json:返回的是一个 json 字符串，jQuery 会将该字符串自动转换成 js 对象。

text:返回的是一个文本

（3）$.post()格式同上。

（4）$.ajax(options):options 是一个形如{key1:value1,key2,value2...}的 js 对象，用于指定发送请求的选项。选项参数如下：

url(string):请求地址

type(string):GET/POST

data(object/string):发送到服务器的数据格式是 {"":"","",""}

dataType(string) :预期服务器返回的数据类型，一般有

xml、html、script、json、text。

success(function): 请求成功后调用的回调函数，有两个参数：

function(data,textStatus),

其中，data 是服务器返回的数据，可以是 html,text,jsonObj,xmlDoc；

textStatus 描述状态的字符串。

error(function):请求失败时调用的函数，有三个参数

function(XmlHttpRequest,textStatus,errorThrown) 如果 textStatus 取到错误消息，就用 errorThrown。