Tomcat&Servlet

讲师：莫凡

# 课程回顾

# 课程介绍

* 1. Tomcat安装;（掌握）
* 2. 手动web项目;（掌握）
* 3. 部署项目;（掌握）
* 4. Servlet入门;（掌握）
* 5. HTTP协议;（了解）

1. 服务器概述【了解】

服务器（英文名称server），也称伺服器。指一个管理资源并为用户提供服务的计算机设备。由于服务器需要响应服务请求，并进行处理，因此一般来说服务器应具备承担服务并且保障服务的能力。服务器的构成包括处理器、硬盘、内存、系统总线等，和通用的计算机架构类似，但是由于需要提供高可靠的服务，因此在处理能力、稳定性、可靠性、安全性、可扩展性、可管理性等方面要求较高。**在网络环境下，根据服务器提供的服务类型不同：**分为**文件服务器，数据库服务器，应用程序服务器，WEB服务器**等。

也就是说服务器分为硬件和软件两种，硬件服务器指的就是电脑，软件服务器指的是在计算机上运行的某个软件【即服务器提供的服务类型】。

根据服务类型不同，我们简单的分为以下三类：**WEB服务器、HTTP服务器、应用程序服务器。**

## WEB服务器&HTTP服务器

Web服务器的基本功能就是提供Web信息浏览服务。它只需支持HTTP协议、HTML文档格式及URL。与客户端的网络浏览器配合。因为Web服务器主要支持的协议就是HTTP，所以通常情况下HTTP服务器和WEB服务器是相等的，说的是一回事。

## 应用程序服务器

应用服务器是指通过各种协议把商业逻辑曝露给客户端的程序。它提供了访问商业逻辑的途径以供客户端应用程序使用。应用服务器使用此商业逻辑就像调用对象的一个方法一样。也就是说应用程序服务器可以执行后台代码。

通俗的讲，**Web服务器传送页面使浏览器可以浏览**，然而**应用程序服务器提供的是客户端应用程序可以调用的方法**。确切一点，你可以说：Web服务器专门处理HTTP请求，但是应用程序服务器是通过很多协议来为应用程序提供商业逻辑。

以Java EE为例，Web服务器主要是处理静态页面处理和作为 Servlet容器，解释和执行servlet/JSP，而应用服务器是运行业务逻辑的，主要是EJB、 JNDI和JMX API等J2EE API方面的，还包含事务处理、数据库连接等功能，所以在企业级应用中，应用服务器提供的功能比WEB服务器强大的多。

## 常见的服务器

Apache、Tomcat都可以属于Web服务器，Weblogic、WebSphere都属于应用服务器。

### Web服务器

Apache：

在Web服务器中，Apache是纯粹的Web服务器，经常与Tomcat配对使用。它对HTML页面具有强大的解释能力，但是不能解释嵌入页面内的服务器端脚本代码（JSP/Servlet）。

Tomcat：

早期的Tomcat是一个嵌入Apache内的JSP/Servlet解释引擎。后来的Tomcat已不再嵌入Apache内，Tomcat进程独立于Apache进程运行。而且**Tomcat已经是一个独立的Servlet和JSP容器**，业务逻辑层代码和界面交互层代码可以分离了。因此，有人把Tomcat叫做轻量级应用服务器。

### 应用服务器

实现了J2EE的各种规范。**WebSphere(IBM)**、**WebLogic(Oracle)**、**JBoss(red hat)**。

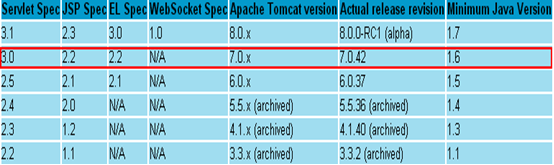
综上：Apache是纯粹的web服务器，而Tomcat因为具有了解释执行服务器端代码的能力，可以称作为**轻量级应用服务器或带有服务器功能的Web服务器**。Weblogic、WebSphere因为能提供强大的J2EE功能，毫无疑问是绝对的应用服务器。对于处于中间位置的Tomcat，它可以配合纯Web服务器Apache一起使用，也可以作为应用服务器的辅助与应用服务器一起部署。

1. 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。

另一个称呼**Servlet容器**：

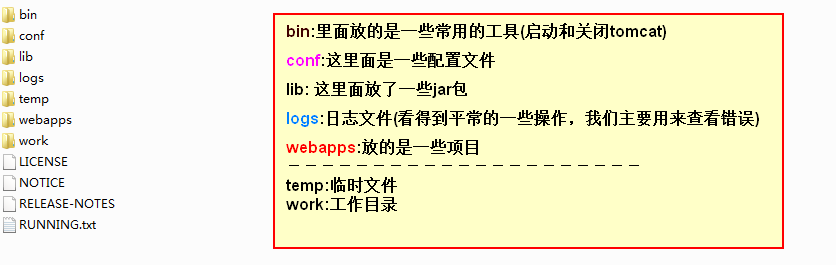
J2EE中实现动态网页开发需要使用到Servlet/JSP，Servlet/JSP类型程序是运行在服务器【Tomcat之类的软件】上。在服务器中负责找到和调用这些Servlet/JSP程序的应用，我们称之为容器。也就是说只要实现了J2EE中Servlet/JSP规范的服务器都可以称呼为Servlet容器，可以理解为是对软件服务器的另一个称呼。



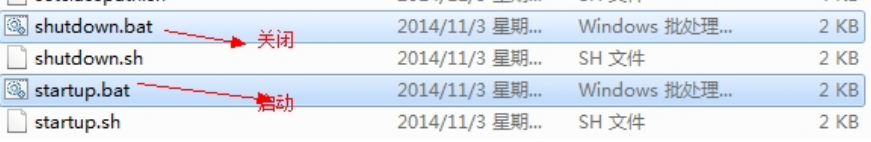
* 1. 安装&访问&配置【重点】

1. 配置JDK环境变量

2. 解压缩TOMCAT的安装包，在不包含中文的路径中



3.Tomcat，浏览器中访问：<http://localhost:8080/>



* + 1. 详解部分

配置JDK环境变量

1.JAVA\_HOME：若Java\_HOME不存在，则”新建”，输入变量名、变量值，变量值末不需要分号

值：D:\Program Files\Java\jdk1.6.0\_29

2.CLASSPATH：环境变量中若不存在，则”新建”，变量值末不需要分号

值：D:\Program Files\Java\jdk1.6.0\_29\lib\dt.jar;D:\Program Files\Java\

jdk1.6.0\_29\lib\tools.jar

3.Path：”编辑”，变量值使用英文分号隔开，变量值末不需要分号

值： D:\Program Files\Java\jdk1.6.0\_29\bin

4.“开始”->“运行”->输入cmd，在命令提示符中输入“java -version”，“java”，“javac”几个命令，出现画面，说明环境变量配置成功

**若JDK配置环境变量之后，仍然不能正常启动Tomcat，可配置Tomcat的环境变量，一般不用**，以TOMCAT7为例：

1.CATALINA\_BASE

值： D:\Program Files\apache-tomcat-7.0.22（Tomcat解压到的目录）

2.CATALINA\_HOME

值： D:\Program Files\apache-tomcat-7.0.22

3.CATALINA\_TMPDIR

值： D:\Program Files\apache-tomcat-7.0.22\temp

4.Path

值：D:\Program Files\apache-tomcat-7.0.22\bin

* + 1. 修改端口

Tomcat默认端口是8080，很多情况下我们都会把这个端口进行修改，修改成80，此时需要在tomcat解压缩目录下的conf目录下的server.xml中修改



Tomcat使用80端口之后，访问时就不用再使用端口号进行访问

* + 1. 访问注意事项

1. 访问项目默认访问index开头的页面，因为Tomcat的web.xml中进行了配置



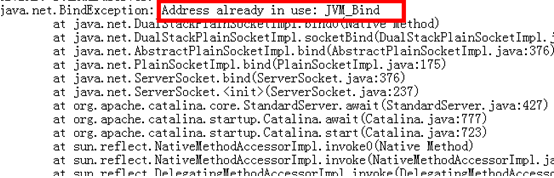
1. 访问之前，一定要启动Tomcat服务器，不启动无法访问



1. 访问页面出现404错误，一般代表访问的URL路径不存在，仔细检查路径



1. 同一个Tomcat只能启动一次，重复启动会报错，因为Tomcat需要监听端口，两次启动会监听同一个端口



1. 不要随便修改Tomcat配置文件中的东西，也不能写中文，若需要在配置文件中写中文，需要将**配置文件的文件格式修改为UTF-8**
   * 1. 其他

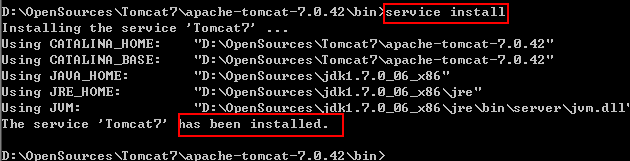


* 1. 第二种安装方式

该种方式以系统服务形式启动Tomcat，这样可以随着系统的启动而启动(一般用在服务上)

### 安装服务

1. 使用cmd工具,并进入Tomcat根/bin目录
2. 执行service install命令





### 删除服务

1. 使用cmd工具,并进入Tomcat根/bin目录
2. 执行service remove命令

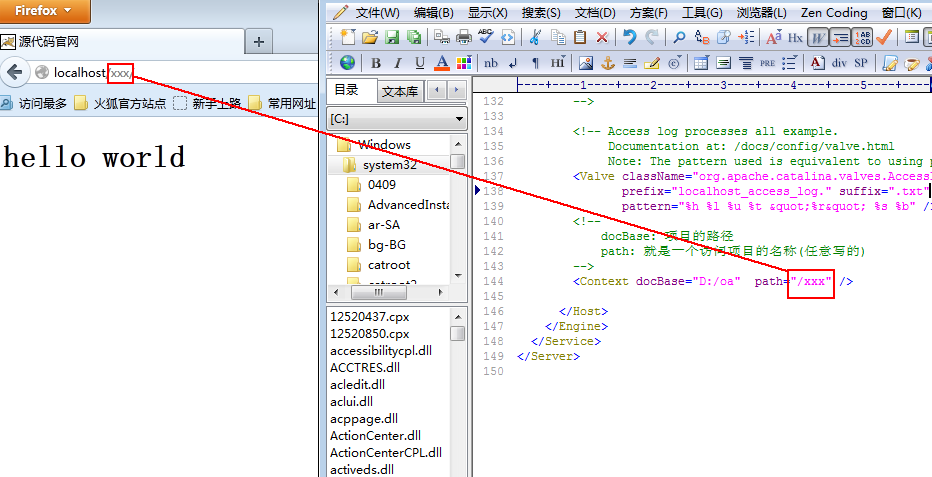


* 1. eclipse安装插件启动

直接找到eclispe的安装路径中的plugins文件夹，将准备好的插件jar包复制拷贝到该文件夹



1. 部署项目【重点】
   1. 手动部署项目
2. 直接将完整项目拷贝到webapps文件夹下
3. 在Server.xml中进行配置
4. 在server.xml文件中的Host节点中添加Context节点



* 1. Context中docBase属性取值：被部署项目所在的路径
     1. 该路径包含项目名，则URL访问时不用带项目名
     2. 该路径不包含项目名，则URL访问时需要带上项目名
  2. Context中path属性取值：URL访问的地址，若不写则不使用路径

1. 在tomcat解压目录下conf\Catalina\localhost中进行配置

在该路径下创建一个XML文件，XML中写入

<Context docBase=”被部署的项目路径” />

此时，Context节点中不需要包含path属性，XML文件名相当于path属性名

1. Eclipse手动创建WEB项目
   1. WEB项目结构

在Tomcat的webapps中存储的是部署的项目，在服务器中的项目文件夹结构如下：

**项目名【文件夹，一般称之为根】**

**|**

**|---------------网页文件、JS、CSS**

**|**

**|---------------WEB-INF【文件夹】**

**|---------------lib【文件夹】**

**|---------------classes【文件夹】**

**|---------------web.xml【配置文件】**

注意：**WEB-INF文件夹里面的程序是不可见的，必需通过我们的程序访问**

web.xml文件可以从Tomcat的自带项目中拷贝

lib文件夹：用于存放web项目中引用的第三方jar包

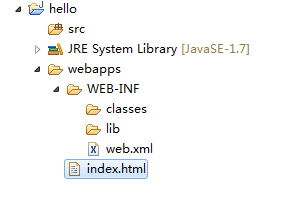
classes文件夹：用于保存以后编译后的.class文件

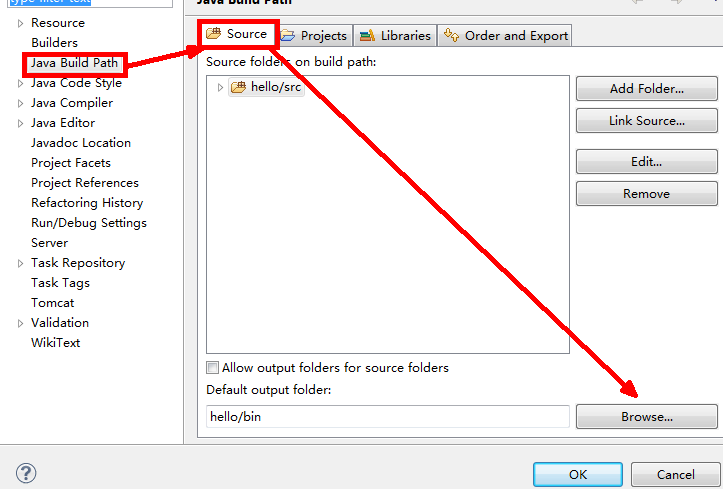
**思考&讨论：**

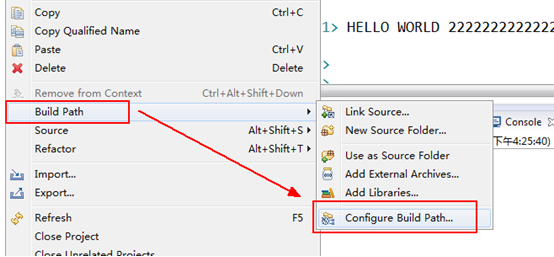
**能否手动创建一个符合上述结构的项目，且在Tomcat中访问成功？**

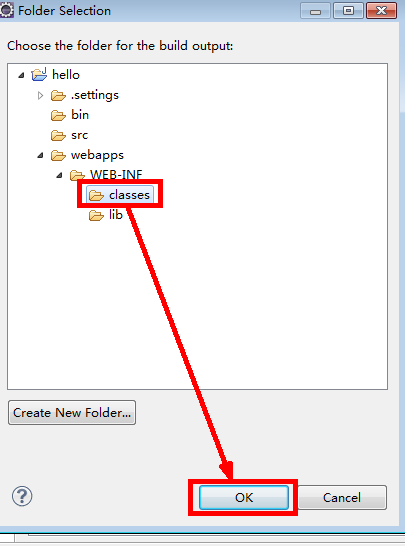
* 1. 手动创建web项目

1. 创建一个普通的JAVA项目
2. 在JAVA项目中添加WebContent文件夹，与src同级
3. WebContent文件夹中根据WEB项目结构分别添加网页文件、WEB-INF文件夹
4. WEB-INF文件夹中添加lib文件夹、classes文件夹、web.xml文件
5. 修改该项目.class文件最终的编译文件夹为WEB-INF文件夹下的classes文件夹









1. Servlet入门
   1. 概述

Servlet 是运行在 Web 服务器中的小型 Java 程序。Servlet 通常通过 HTTP（超文本传输协议）接收和响应来自 Web 客户端的请求。

要实现此接口，可以编写一个扩展 javax.servlet.GenericServlet的一般Servlet，或者编写一个扩展 javax.servlet.http.HttpServlet的 HTTPServlet。

此接口定义了初始化Servlet的方法、为请求提供服务的方法和从服务器移除Servlet的方法。这些方法称为生命周期方法，它们是按以下顺序调用的：

1. 构造Servlet，然后使用init方法将其初始化。

2. 处理来自客户端的对service方法的所有调用。

3. 从服务中取出Servlet，然后使用destroy方法销毁它，最后进行垃圾回收并终止它。

除了生命周期方法之外，此接口还提供了getServletConfig方法和 getServletInfo方法，Servlet可使用前一种方法获得任何启动信息，而后一种方法允许servlet返回有关其自身的基本信息，比如作者、版本和版权。

思考&讨论：

上面这段话透露出了几个信息？讨论分析

1. 什么是Servlet？Servlet作用？

2. Servlet实现的方式

3. Servlet的生命周期

4. Servlet中的getServletConfig

5. Servlet中的getServletInfo

* 1. Servlet实现

servlet就是SUN公司提供的一个动态网页开发技术。是一个运行在WEB服务器上的小型Java程序，用于接收和响应客户端请求，使用HTTP协议。

实现&运行步骤：

1. 定义类，实现servlet接口或者继承GenericServlet或者继承HTTPServlet

2. 在web.xml中针对实现的servlet进行配置

3. 浏览器上访问servlet

Servlet :接口

|

GenericServlet :通用的Servlet

|

HttpServlet :HttpServlet

\* web.xml进行配置

实现Servlet可以通过实现Servlet接口，也可以继承GenericServlet类或者继承HttpServlet类，那么这三者有什么区别呢？

1. 基于Servlet接口的实现需要实现接口中所有方法

2. GenericServlet类是Servlet接口的实现类，但是不带有任何协议

3. HttpServlet类同样是Servlet接口的实现类，但实现了Http协议

* + 1. Servlet接口

API

Servlet接口：

void init(ServletConfig config)

初始化创建一个Servlet，首次访问Servlet时创建或者web.xml中配置创建

void service(ServletRequest req,ServletResponse resp)

访问Servlet时执行的方法

void destroy()

销毁Servlet，仅当服务器正常关闭时调用此方法销毁

ServletConfig getServletConfig()

返回Servlet中的配置信息相关

String getServletInfo()

实现servlet接口

public class ServletDemo1 implements Servlet{

@Override

/\*\*

\* 为用户处理请求和响应的方法.

\*/

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException {

res.getWriter().println("Hello Servlet...");

}

...

}

web.xml中配置servlet

<servlet>

<!-- Servlet的名称 -->

<servlet-name>test1</servlet-name>

<!-- Servlet的全路径 -->

<servlet-class>com.itsource.servlet.ServletDemo1</servlet-class>

</servlet>

<!-- Servlet的映射 -->

<servlet-mapping>

<!-- Servlet的名称，注意不要和tomcat的web.xml冲突 -->

<servlet-name>test1</servlet-name>

<!-- Servlet的访问路径 -->

<url-pattern>/ServletDemo1</url-pattern>

</servlet-mapping>

访问servlet

http://localhost:8080/demo/ServletDemo1

1. Http协议【了解】

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。所有的WWW文件都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收HTML页面的方法。1960年美国人Ted Nelson构思了一种通过计算机处理文本信息的方法，并称之为超文本（hypertext）,这成为了HTTP超文本传输协议标准架构的发展根基。Ted Nelson组织协调万维网协会（World Wide Web Consortium）和互联网工程工作小组（Internet Engineering Task Force ）共同合作研究，最终发布了一系列的RFC，其中著名的RFC 2616定义了HTTP 1.1。

* 1. HTTP1.0和HTTP1.1

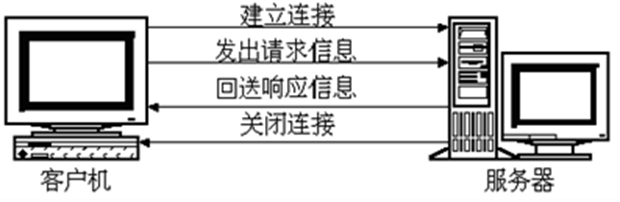
浏览器给服务器发送数据：**一次请求(request)**

服务器给浏览器反馈数据：**一次响应(response)**

在一个TCP连接上可以传送多个HTTP请求和响应，多个请求和响应过程可以重叠进行，增加了更多的请求头和响应头。

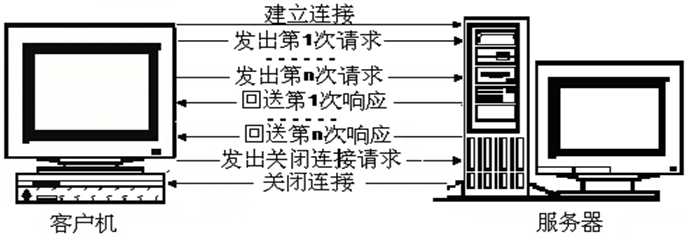
HTTP1.0：**一次请求，响应之后连接断开**

HTTP 1.0规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个TCP连接，服务器完成请求处理后立即断开TCP连接，服务器不跟踪每个客户也不记录过去的请求。



HTTP1.1：一次请求，响应之后，并不会立刻关闭，会保持连接

HTTP 1.1的持续连接，也需要增加新的请求头来帮助实现，例如，Connection请求头的值为Keep-Alive时，客户端通知服务器返回本次请求结果后保持连接；Connection请求头的值为close时，客户端通知服务器返回本次请求结果后关闭连接。



* 1. 浏览器中查看请求和响应【了解】
     1. GET请求

请求包含：请求行、请求头、请求体【可省略】

GET请求信息

GET请求没有请求体

GET /demo/demo1-http/demo2.html?name=aaa&age=23 HTTP/1.1

Accept: text/html, application/xhtml+xml, \*/\*

X-HttpWatch-RID: 59176-10011

Referer: http://localhost:8080/demo/demo1-http/demo1.html

Accept-Language: zh-CN

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko

Accept-Encoding: gzip, deflate

Host: localhost:8080

DNT: 1

Connection: Keep-Alive

* + 1. POST请求信息

POST有请求体

POST /demo/demo1-http/demo2.html HTTP/1.1

Accept: text/html, application/xhtml+xml, \*/\*

X-HttpWatch-RID: 59176-10031

Referer: http://localhost:8080/demo/demo1-http/demo1.html

Accept-Language: zh-CN

User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; rv:11.0) like Gecko

Content-Type: application/x-www-form-urlencoded

Accept-Encoding: gzip, deflate

Host: localhost:8080

Content-Length: 15

DNT: 1

Connection: Keep-Alive

Cache-Control: no-cache

name=bbb&age=38 //请求体部分

* + 1. 请求消息头含义

Accept：

浏览器可接受的MIME类型（Tomcat安装目录/conf/web.xml中查找），MIME类型就是设定某种扩展名的文件用一种应用程序来打开的方式类型，当该扩展名文件被访问的时候，浏览器会自动使用指定应用程序来打开

Accept-Charset：

告知服务器，客户端支持哪种字符集

Accept-Encoding：

浏览器能够进行解码的数据编码方式

Accept-Language：

浏览器支持的语言。

Referer：

当前页面由哪个页面访问过来的。

Content-Type：

通知服务器，请求正文的MIME类型。

取值：application/x-www-form-urlencoded默认值

对应的是form表单的enctype属性

Content-Length：

请求正文的长度

If-Modified-Since:

通知服务器，缓存的文件的最后修改时间。

User-Agent:

通知服务器，浏览器类型.

Connection:

表示是否需要持久连接。如果服务器看到这里的值为“Keep -Alive”，或者看到请求使用的是HTTP 1.1（HTTP 1.1默认进行持久连接

Cookie:

这是最重要的请求头信息之一（会话有关）

响应

响应包含：状态行、响应头、响应体

响应信息

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Accept-Ranges: bytes

ETag: W/"145-1461807615933"

Last-Modified: Thu, 28 Apr 2016 01:40:15 GMT

Content-Type: text/html

Content-Length: 145

Date: Thu, 28 Apr 2016 01:43:52 GMT

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Insert title here</title>

</head>

<body>

<h1>响应部分</h1>

</body>

</html>

* + 1. 响应消息头含义

Location：

制定转发的地址。需与302/307响应码一同使用

Server：

告知客户端服务器使用的容器类型

Content-Encoding：

告知客户端服务器发送的数据所采用的压缩格式

Content-Length：

告知客户端正文的长度

Content-Type：

告知客户端正文的MIME类型

Conent-Type:text/html;charset=UTF-8

Refresh：

定期刷新。还可以刷新到其他资源

Refresh:3;URL=otherurl：表示3秒后刷新到otherurl这个页面

Content-Disposition：

指示客户端以下载的方式保存文件。

Content-Disposition：attachment;filename=2.jpg

Expires：

网页的有效时间。单位是毫秒(等于-1时表示页面立即过期)

Cache-Control：no-cache

Pragma：no-cache

控制客户端不要缓存

Set-Cookie:SS=Q0=5Lb\_nQ; path=/search服务器端发送的Cookie（会话有关）

get与post的区别

1. post相对安全 get不安全(会在地址栏显示)

2. post没有限制 get的总字符数是有限的

3. post无法保留提交书签(不利于搜索引擎抓取)

4. 超连接的时候用get（用于查询居多）表单提交的时候我们会使用post(用于更新居多)

* 1. 状态行

2XX：成功

3XX：重定向

4XX：请求错误

5XX/6XX：服务器错误

1. 课程总结
   1. 重点
   2. 难点
   3. 如何掌握？
      * 1. 勤加练习
        2. 学会看说明手册
   4. 排错技巧（技巧）
2. 课后练习
   * + 1. 手动创建web项目
       2. 搭建服务器
       3. server.xml中部署项目
       4. 搭建javaweb小项目环境
3. 面试题
4. 扩展知识或课外阅读推荐（可选）
   1. 扩展知识
   2. 课外阅读