

一、Tomcat 服务器（很熟悉）

1、Web 开发概述

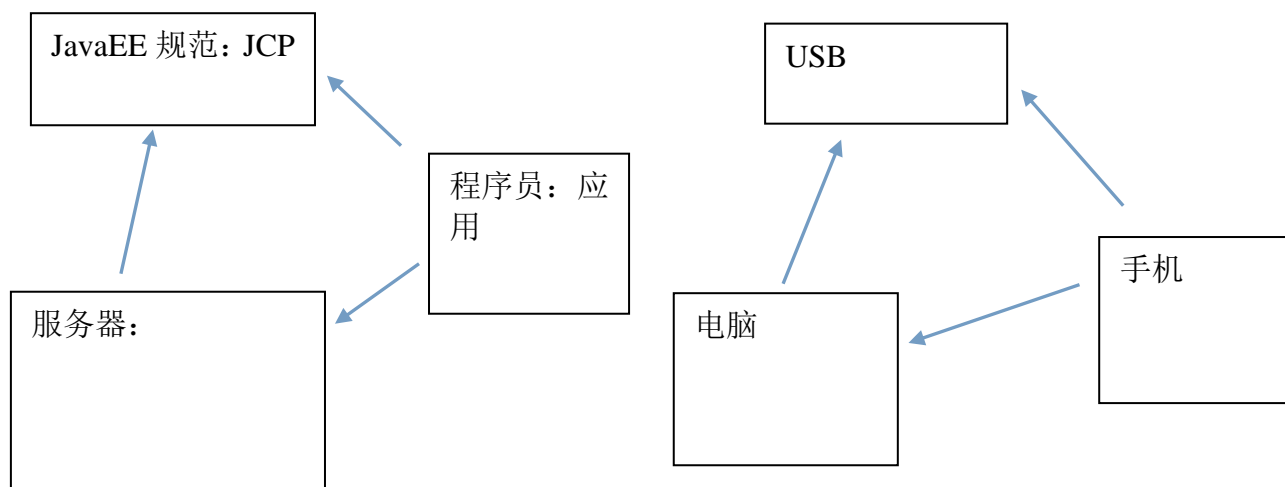
javaSE:

javaEE:13 种

javaME:

JavaEE 规范: 13 种技术的总称。Servlet/Jsp JDBC JNDI JTA...

Tomcat:Servlet/Jsp 容器,轻量级服务器。



2、Tomcat 下载

Tomcat 官方站点: <http://tomcat.apache.org>

3、Tomcat 的安装与配置

3.1、安装:

直接解压到指定目录即可。(注: 目录不要太深; 目录不要有中文或空格)

3.2、启动服务器:

F:\apache-tomcat-7.0.52\bin\startup.bat

启动服务器的前题:

配置 JAVA_HOME:

进入到 F:\apache-tomcat-7.0.52\bin>startup.bat 才能执行命令

配置 CATALINA_HOME:

F:\apache-tomcat-7.0.52

startup.bat 启动命令

shutdown.bat 停止命令

如果 tomcat 端口被占用，解决办法

> 修改 F:\apache-tomcat-7.0.52\conf\server.xml

第 70 行： <Connector port="8081"/>

> 关闭端口号对应的进程：

cmd-->netstat -ano -->找到pid-->任务管理器-->显示pid-->关闭进

程

3.3、测试服务器是否启动成功

http://localhost:8080

3.4、Tomcat 的主要目录：（重要）

bin	可执行文件
conf	配置文件
lib	tomcat运行时需要用的jar文件
logs	tomcat运行产生的日志文件
temp	运行时产生临时文件
webapps	*** 存放我们编写的應用
work	tomcat的工作目录
LICENSE	
NOTICE	
RELEASE-NOTES	
RUNNING.txt	

4、标准的 JavaWeb 应用的目录结构（很重要：记住）

应用：

MyApp

1.html

Css

myStyle.css

Js

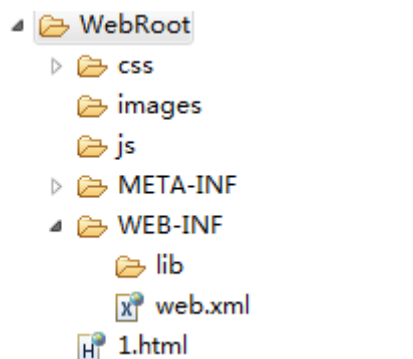
My.js

WEB-INF：注意：固定写法。此目录下的文件不能被外部直接访问。

classes：我们编写的程序代码。.class 文件(class 中的类层次与包的层次相同)

lib：应用需要用的 jar 文件

web.xml：应用的配置信息



5、部署应用到 Tomcat 服务器

> 开放目录部署方式

把应用直接复制到 F:\apache-tomcat-7.0.52\webapps 下。

> 把应用打成 war 包。

打 war 包命令：jar -cvf MyApp.war .

把 war 包直接复制到 F:\apache-tomcat-7.0.52\webapps 下，应用自动解压

注：webapps 目录下有几个目录就代表有几个应用。

Tomcat 服务器与 MyEclipse 集成(一定要掌握)

- 1、Tomcat 集成
- 2、创建 web 应用
- 3、部署

URL: 统一资源定位符（网址）

URI: 统一资源标识符

http://localhost:8080/day08_02/1.html

- **http**→协议
- **localhost:8080**→主机 IP（端口号）
- **day08_02/1.html**→URI（当前应用的资源路径）

6、深入熟悉 Tomcat 服务器（了解）

6.1、虚拟目录：

方式一：（不建议使用，因为需要重启服务器）

在 tomcat 配置文件 server.xml 中，设置虚拟目录：

虚拟目录映射：

```
<Host name="localhost" path="/myApp" appBase="webapps"
      unpackWARs="true" autoDeploy="true">
```

- docBase:代表的是应用的真实路径。
- path: 网络访问的虚拟目录名
- unpackWARS: 自动解压
- autoDeploy: 自动部署

注：配置完成后需要重启服务器。

访问时访问的是虚拟目录，访问以上网址的 url 为：

<http://localhost:8080/myAPP>

若 path 改为 path="/aaa"，访问时 url 为：

<http://localhost:8080/aaa>

方式二：

新建 myAPP.xml 文件，文件名就代表应用的虚拟目录名。

将文件放在 F:\apache-tomcat-7.0.52\conf\Catalina\localhost 目录下

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<Context docBase="D:\myAPP"/>
```

➤ 文件名 myAPP 为访问的虚拟目录；

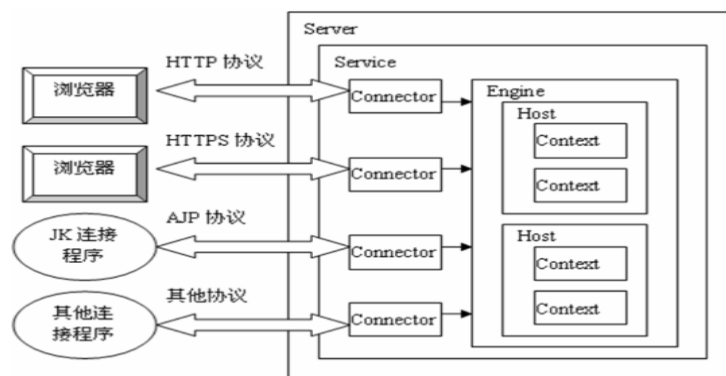
➤ docBase 为访问的真实目录；

注：配置完成后不需要重启。

访问：<http://localhost:8080/myAPP>

6.2、Tomcat 体系结构

- Server 表示整个 tomcat 服务器；
- 一个服务器对应多个服务，每个服务由 Connector 配置，可以配置不同的端口和协议；
- 服务会调用 tomcat 引擎(engine)对请求做出处理，引擎管理 host。
- 一个 host 表示一个主机，负责调用对应的程序，处理具体的请求，发回相应。



7、配置默认端口、默认应用、默认主页

A、把 server.xml 中 <Connector port="80" protocol="HTTP/1.1" connectionTimeout="20000" redirectPort="8443" />

浏览器 http 默认端口：80

B、默认应用：

默认应用：把配置的虚拟目录的配置文件名改为 ROOT.xml 即可。

C、默认主页

修改当前应用 web.xml,添加以下内容：

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app 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_3_0.xsd"
  version="3.0">

  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>1.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

</web-app>
```

二、HTTP 协议

1、HTTP 协议概述

> HTTP 是 HyperText Transfer Protocol(超文本传输协议)的简写，传输 HTML 文件。

> 用于定义 WEB 浏览器与 WEB 服务器之间交换数据的过程及数据本身的格式。

2、请求部分

```
POST /day08_02/1.html HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml, image/gif, image/png
Referer: http://localhost:8080/day08_02/1.html
Accept-Language: zh-CN
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.0.4506.2; .NET CLR 3.5.30729; .NET CLR 3.0.30729)
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: localhost:8080
Content-Length: 17
Connection: Keep-Alive
Cache-Control: no-cache

uName=tom&pwd=123
```

1、请求消息行

GET /day08_02/1.html HTTP/1.1

请求方式：Get（默认） POST DELETE HEAD 等

GET：明文传输 不安全，数据量有限，不超过 1kb

GET /day08_02/1.html?uName=tom&pwd=123 HTTP/1.1

POST: 明文传输，安全。数据量没有限制。

URI: 统一资源标识符。去协议和 IP 地址。

协议/版本：

2、请求消息头

从第 2 行到空行处，都叫消息头

Accept:浏览器可接受的 **MIME** 类型

告诉服务器客户端能接收什么样类型的文件。

Accept-Charset: 浏览器通过这个头告诉服务器，它支持哪种字符集

Accept-Encoding:浏览器能够进行解码的数据编码方式，比如 **gzip**

Accept-Language:浏览器所希望的语言种类，当服务器能够提供一种以上的语言版本时要用到。可以在浏览器中进行设置。

Host:初始 URL 中的主机和端口

Referer:包含一个 **URL**，用户从该 **URL** 代表的页面出发访问当前请求的页面

Content-Type:内容类型

告诉服务器浏览器传输数据的 **MIME** 类型，文件传输的类型

`application/x-www-form-urlencoded`

If-Modified-Since: Wed, 02 Feb 2011 12:04:56 GMT 利用这个头与服务器的文件进行比对，如果一致，则从缓存中直接读取文件。

User-Agent:浏览器类型。

Content-Length:表示请求消息正文的长度

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（在讲会话时解析）

Date: Date: Mon, 22 Aug 2011 01:55:39 GMT 请求时间 GMT

3、消息正文：当请求方式是 POST 方式时，才能看见消息正文

`uName=tom&pwd=123`

3、响应部分

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: W/"188-1434960808873"
Last-Modified: Mon, 22 Jun 2015 08:13:28 GMT
Content-Type: text/html
Content-Length: 188
Date: Mon, 22 Jun 2015 08:22:41 GMT
```

```
<html>
<head>
```

服务器给浏览器发送的附加信息

1、响应消息行

第一行:

HTTP/1.1 200 OK

协议/版本 响应状态码 对响应码的描述 (一切正常)

响应状态码:

常用的就 40 多个。

200(正常) 一切正常

302/307(临时重定向)

304(未修改)

表示客户机缓存的版本是最新的, 客户机可以继续使用它, 无需到服务器请求。

404(找不到) 服务器上不存在客户机所请求的资源。

500(服务器内部错误)

2、响应消息头

Location: <http://www.it315.org/index.jsp> 指示新的资源的位置

通常和 302/307 一起使用, 完成请求重定向

Server:apache tomcat 指示服务器的类型

Content-Encoding: gzip 服务器发送的数据采用的编码类型

Content-Length: 80 告诉浏览器正文的长度

Content-Language: zh-cn 服务发送的文本的语言

Content-Type: text/html; charset=GB2312 服务器发送的内容的 MIME 类型

Last-Modified: Tue, 11 Jul 2000 18:23:51 GMT 文件的最后修改时间

Refresh: 1;url=<http://www.it315.org> 指示客户端刷新频率。单位是秒

Content-Disposition: attachment; filename=aaa.zip 指示客户端下载文件

Set-Cookie:SS=Q0=5Lb_nQ; path=/search 服务器端发送的 Cookie

Expires: -1

Cache-Control: no-cache (1.1)

Pragma: no-cache (1.0) 表示告诉客户端不要使用缓存

Connection: close/Keep-Alive

Date: Tue, 11 Jul 2000 18:23:51 GMT

3、响应正文

和网页右键“查看源码”看到的内容一样。