# Web学习

## 记得录屏

## 常用的Web容器

### Tomcat

### Jetty

### Weblogic

### Nginx

### JBoss

### IIS

### WebSphere

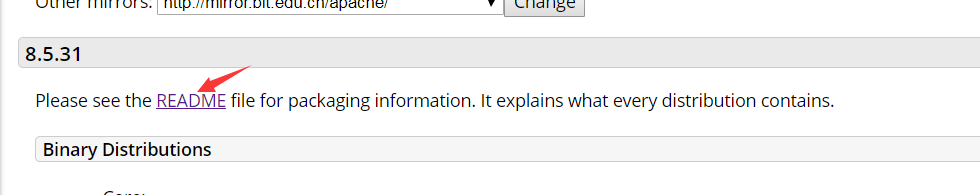
## Tomcat服务

### tomcat的安装（windows）

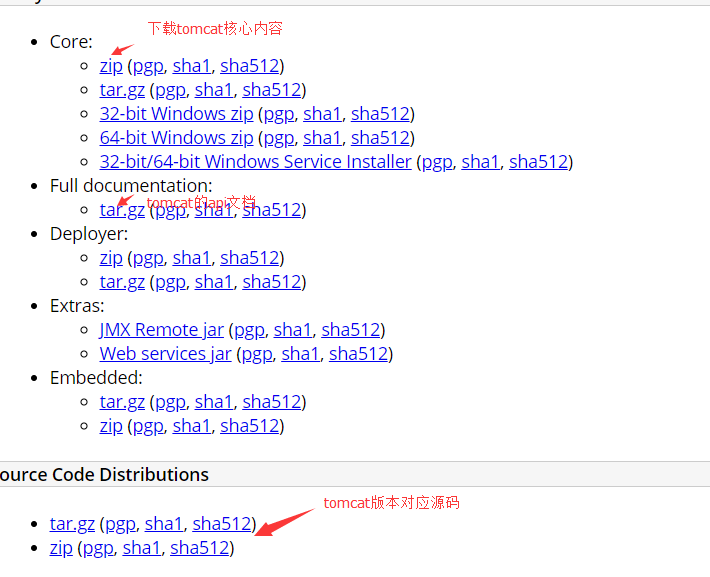
1、确定系统版本选择对应的tomcat版本(X86[32位]，X86\_64[64位的版本])

2、在apache官网（https://tomcat.apache.org/）上面下载对应tomcat版本

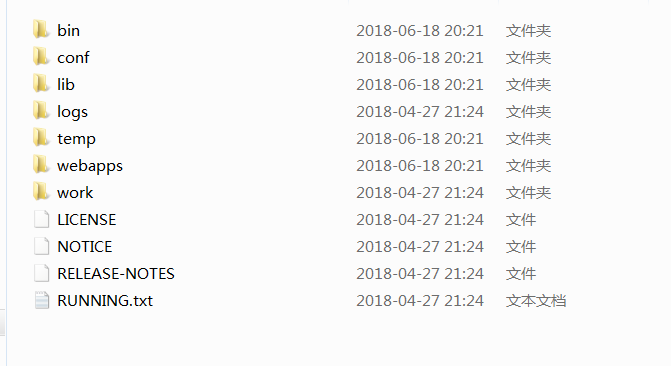
a、确定本地安装的jdk版本是否与当前tomcat版本兼容



b、指定下载的tomcat内容



### tomcat的目录结构说明



bin： tomcat所有可执行文件的存放目录(window，linux系统的操作)

widnows =>startup.bat,shutdown.bat

linux=> starup.sh,shutdown.sh

catalina.sh, catalina.bat 都是设置tomcat内存参数

conf： tomcat配置文件目录

lib： tomcat运行需要的依赖jar

logs： tomcat运行操作过程中产生的日志薪

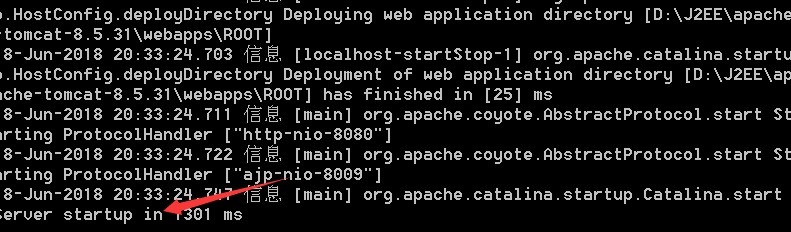
temp: 临时文件存储目录

webapps: web项目部署的目录

work: jsp文件运行编译一些文件存放的目录

### 测试tomcat能否正常运行

1、启动tomcat服务器=>bin/startup.bat



启动异常排除：

闪退: 1、本地没有安装jdk，2、本地安装的jdk版本和tomcat服务不兼容，3、端口冲突。

可以查看tomcat目录下面logs/catlina.log文件

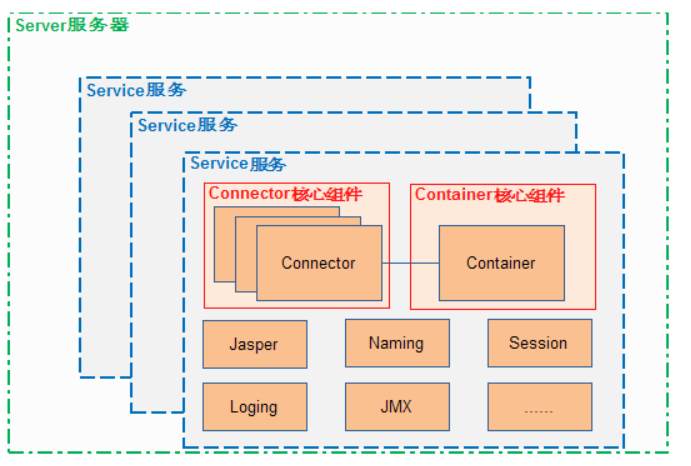
<http://localhost:8080>

2、在tomcat的webapps中发布一个项目

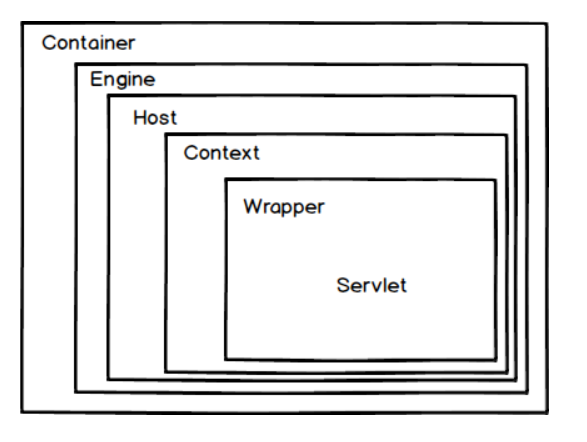
a、 在webapps中创建一个myapp目录下面创建一个简单的html页面

b、直接启动tomcat服务器访问localhost:8080/myapp/my.html

### 总体架构

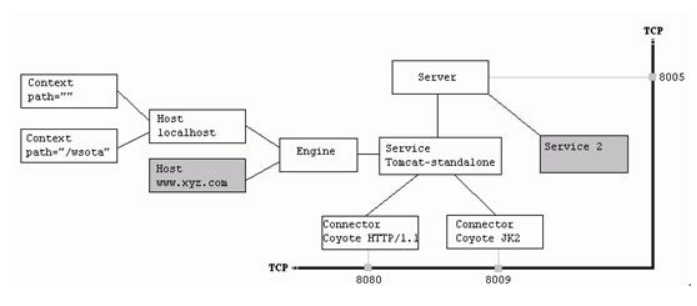


#### 主要的组件



1. Engine：没有父容器，一个 Engine代表一个完整的 Servlet 引擎，它接收来自Connector的请求，并决定传给哪个Host来处理，Host处理完请求后，将结果返回给Engine，Engine再将结果返回给Connector。
2. Host：Engine可以包含多个Host，每个Host代表一个虚拟主机，这个虚拟主机的作用就是运行多个应用，它负责安装和展开这些应用，并且标识这个应用以便能够区分它们，每个虚拟主机对应的一个域名，不同Host容器接受处理对应不同域名的请求。
3. Context：Host可以包含多个Context，Context是Servlet规范的实现，它提供了Servlet的基本环境，一个Context代表一个运行在Host上的Web应用
4. Wrapper: Context可以包含多个Wrapper, Wrapper 代表一个 Servlet，它负责管理一个 Servlet，包括的 Servlet 的装载、初始化、执行以及资源回收。Wrapper 是最底层的容器，它没有子容器了，所以调用它的 addChild 将会报错。

#### tomcat的执行过程



假设来自客户的请求为：http://localhost:8080/test/index.jsp

1. 请求被发送到本机端口8080，被在那里侦听的Coyote HTTP/1.1 Connector获得
2. Connector把该请求交给它所在的Service的Engine来处理，并等待Engine的回应
3. Engine获得请求localhost:8080/test/index.jsp，匹配它所有虚拟主机Host
4. Engine匹配到名为localhost的Host（即使匹配不到也把请求交给该Host处理，因为该Host被定义为该Engine的默认主机）
5. localhost Host获得请求/test/index.jsp，匹配它所拥有的所有Context
6. Host匹配到路径为/test的Context（如果匹配不到就把该请求交给路径名为""的Context去处理）
7. path="/test"的Context获得请求/index.jsp，在它的mapping table中寻找对应的servlet
8. Context匹配到URL PATTERN为\*.jsp的servlet，对应于JspServlet类
9. 构造HttpServletRequest对象和HttpServletResponse对象，作为参数调用JspServlet的doGet或doPost方法
10. Context把执行完了之后的HttpServletResponse对象返回给Host
11. Host把HttpServletResponse对象返回给Engine
12. Engine把HttpServletResponse对象返回给Connector

13. Connector把HttpServletResponse对象返回给客户browser

# Servlet

普通java application 🡺 控制台执行的。

Servlet定义： 运行在服务端的java应用程序。

## Servlet基础

从现在开始都一直使用idea开发工具。（<http://xdouble.cn:8888/>破解）

### 1、怎么创建一个Servlet

a、创建一个普通的java类继承HttpServlet类

b、重写HttpServlet类中的doPost，doGet方法

客户端常用的请求方式:

post: 一般是表单提交

get: 地址栏请求、超链接

### 2、Servlet怎么配置

a、完成客户端请求与Servlet处理类的映射

b、在WEB-INF/web.xml文件中配置servlet和servlet-mapping。



### 3、Servlet怎么运行

通过实例运行。

### 4、Servlet执行的生命周期和流程

a、servlet执行的生命周期：

实例化->初始化（初始化一些参数）->调用service方法->[doGet，doPost处理]->销毁方法[服务器正常关闭或者重启才能够触发]

b、流程步骤:

1、通过客户端请求的路径找到web.xml文件中的url-pattern配置

2、通过url-pattern对应name找到servlet节点中配置的servlet-class中内容。

3、tomcat容器的warpper组件通过反射方法创建Servlet实例。

4、通过请求的方式获取到对应请求处理方法。

5、通过反射的方式调用对应处理方法，把warpper组件中构建的HttpServletRequest，HttpServletResponse对象传入对应处理方法中完成调用。

6、通过HttpServletResponse把处理的结果响应到客户端。

### 5 Servlet与九大内置对象

|  |  |
| --- | --- |
| JSP对象 | 怎样获得 |
| Out | Resp.getWriter |
| Request | Service方法中的req参数 |
| Response | Service方法中的resp参数 |
| Session | Req.getSession()函数 |
| Application | getServletContext()函数 |
| Exception | Throwable |
| Page | this |
| pageContext | PageContext |
| Config | getServletConfig函数 |