主要是讲Apache如何连接tomcat的

Server是一个顶级组件

Service仅仅是一个关联组件

一定注意 一个连接器Connector仅仅属于一个引擎

Connector:Engine = N:1

因此在同一个Service中 仅仅有一个Engine

Engine是一个容器类组件 里面可以部署WEB程序

Engine内部可以部署多个host ---- 类似于Apache的虚拟主机

当外部的请求链接到哪一个主机 ---- 就是链接到匹配的虚拟主机 否则就访问默认的虚拟主机 --- 因此定义引擎的时候 要定义一个默认主机

在一个主机内部可以定义context ---- context类似于Apache中的别名 alias

这个是tomcat的层次架构

同一个物理主机上 每一个tomcat运行起来的进程 就是server 就是jvm进程

我们tomcat自身就是Java写的 ---- 所以 tomcat要运行 必须把tomcat自身的类装载到已经运行的jvm虚拟机中 ==== 绝不是tomcat启动了 tomcat进程再去启动一个jvm, 然后tomcat进程来管理这个jvm进程 ----- 这个是错误的

而是tomcat需要借助一个额外的组件 Bootstrap ------ 来完成加载启动jvm 将tomcat自身需要的类都加载进去 最后tomcat自己运行到里面去了 ---- 最后 在这个jvm中 部署了一个Servlet Container

Tomcat实例本身就是一个jvm虚拟机

一个物理主机 每一个tomcat就是jvm进程

======

能不能启动两个tomcat实例？

Tomcat支持的链接器类型有 ajp, http https proxy

每一个连接器都需要一个Java的一个类来实现他们

Tomcat在内部使用Java实现了一个web服务器

apr – apache portable runtime Java可移植运行环境

如果你的tomcat支持apr 那么有两种以上的格式

是一种让Apache跨平台的组件

连接器 ----- 由此 对于连接器 tomcat而言

一个tomcat要运行 必须注册至少一个连接器 否则无法接受外界的请求

同一个物理机：

T1监听在 http 80 ajp 8009

那么另一个tomcat实例T2 和 T1独立的进程 各自是完成的组件 这样 两个不能监听在同一个套接字上

这样 在同一个主机上面 我们只需要定义监听在不同套接字上面的tomcat ----- 就可以同时启动起来 ----- 这个就是多实例的tomcat

Mysql支持多实例么？ -------- 默认监听在3306 ----- 可以多实例 只是使用了不同的端口

---- 还可以做主从

多实例 在某些级别的服务 还是很常见的 虽然不建议这样部署

昨天的脚本只能启动一个tomcat

但是 就需要每次修改Catalina\_home

还有一个变量 就是CATALINA\_BASE 用来指定catalina的基本目录

前端的反向代理 Nginx过来的请求 本地没有 就要代理到后端服务器上 使用的就是http请求 ---- 这样nginx自己没有 就会转发到后端的web应用服务器上面 ------ 这样 对于外部 可以隐藏真实的web应用服务器 --- 提高安全性

注意 每一个web对象都是独立请求的

==🡺每一个图片都是独立的web资源 ---- 所以每一次请求 都是一个独立的web资源的请求 不存在混合的请求

Html就是html img就是img html中的img的src 会发两次请求 一次本身html 一次img

一个是动态资源请求 另一个是静态资源请求

同理 对于客户端服务器 可以启动缓存功能

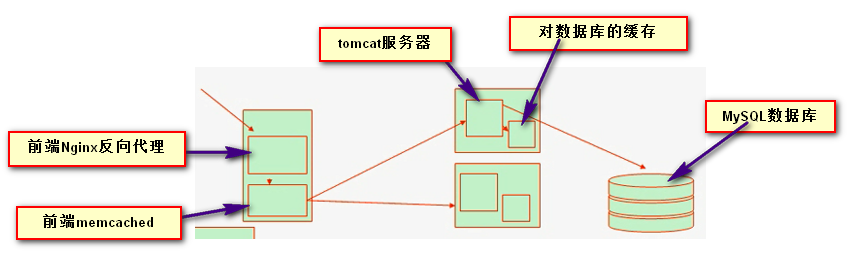
请求图片 本地返回 可以缓存

请求动态信息 第一次 从web应用服务器获取内容 然后缓存 最后返回给客户端

前端内容很快到达 ---- 但是nginx此时即使web服务器 又是反向代理服务器 又要提供缓存 ---- 压力大 ---- 这时候 把缓存提出来 提供专业的缓存服务器

前端是反向代理 中间是缓存 最后是tomcat ----这是三层结构 这个级别至少是中型服务

还可以这样部署



所以 设计的时候 不得不这样干

分层主要是服务器不能够应对这样的压力

架构扩展或者迁移不麻烦

但是 就是业务要终止一段时间

【我们的nginx就是和tomcat部署到一台机器了 --- shuaji】

其实tomcat就是应用程序服务器

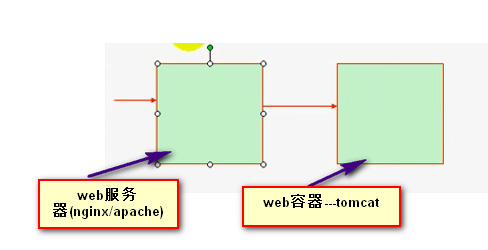
LAMP中 php无法处理应用请求 要和Apache 组合才可以

但是 tomcat自身就可以接受外部请求 tomcat相当于LAMP中的p--php

Tomcat部署的实例部署

------ 如果使用tomcat内部的web服务器进行部署 那就是standalone的部署

------ 同一个物理机 前端是nginx/apache + tomcat ---- 这个也可以分开



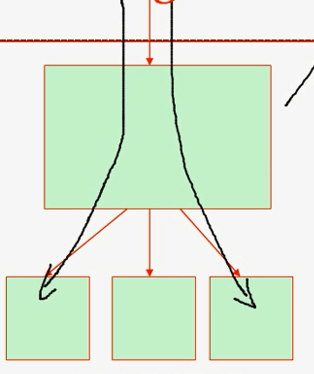
甚至有些场景建议使用Apache 主要是 Apache反向代理的时候 有两类模块 其实有三类模块 可以实现反向代理 ----- 成为Apache的代理模块 或者 Apache的连接器

 jk这种设计用来和tomcat完成通信 jk2的版本 ----- 但是jk2bug太多 所以没人维护 ---- 要用也用jk

早起建议使用jk 或者jk2 现在proxy做了巨大的改进 现在建议使用mod\_proxy

Mod\_proxy对新技术的运用的上超过了mod\_jk

如果后面是电商应用 如果代理 没有控制 会出现session失效、



但是mod\_proxy是可以实现代理绑定 session affinity 说白了就是基于会话的持久性 jk也能实现

session affinity 向后还可以区别是jsp 还是php

由此 我们的Apache有两类模块 mod\_proxy 和mod\_jk

无论是哪一种 代理到后端的时候 一种是http 另一种是ajp---二进制格式 但是效率高

------

Nginx使用upstream可以实现负载均衡

26:00

先不看【这一讲】