机器 汇编 C C++

计算机芯片 🡪汇编耦合 ISA32架构

ASCII C

编译依赖于底层的操作系统架构

CGI java跨平台

Java是运行在jvm中的，而非操作系统；虚拟运行环境是针对平台的。Jvm不是跨平台的；这既是适配者模式，添加一层就可以屏蔽掉不兼容性。

Java编译的结果为字节码文件，而非真正的二进制文件

Applet 应用小程序

Java的组成部分：

1. Java程序设计语言
2. Java class文件格式
3. Java api
4. Java vm

Java编译环境：javac 🡪编译结果为class文件，即字节码文件

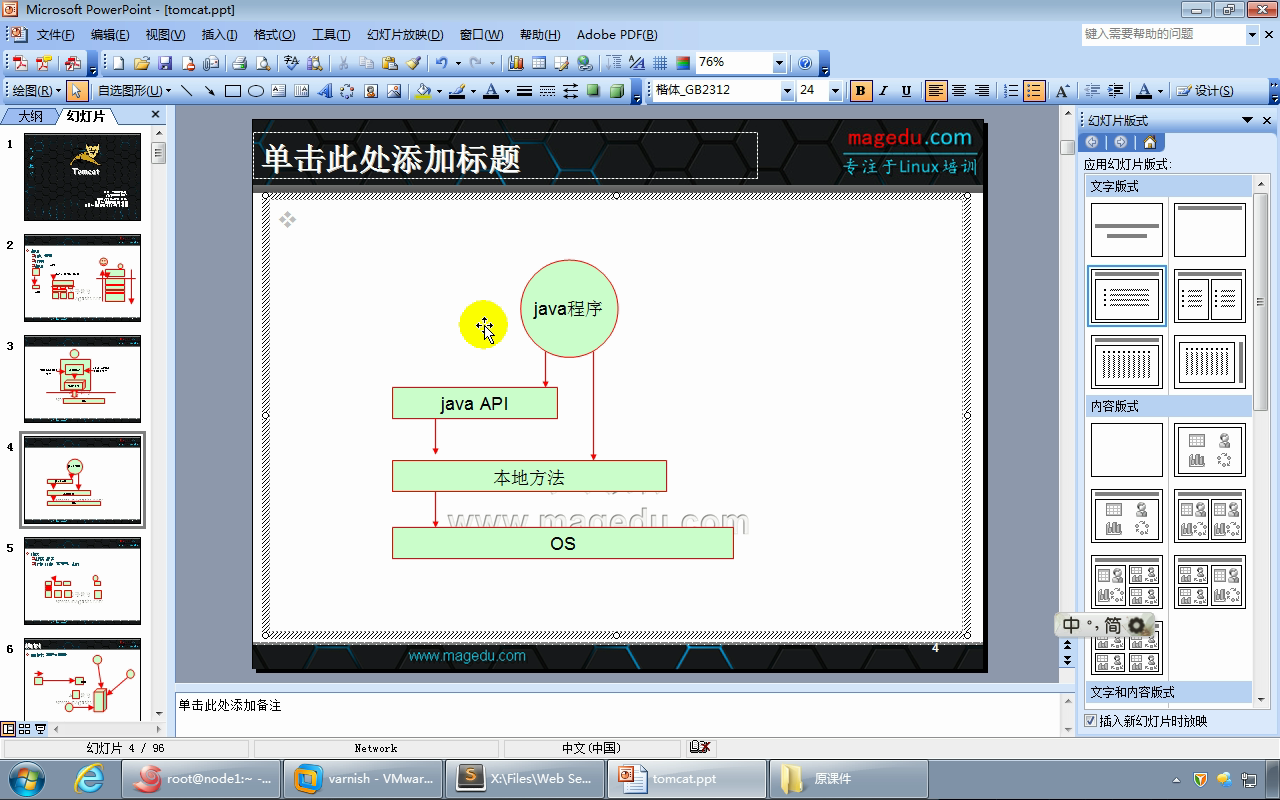
Java的运行环境：jvm 🡪装载类库，类加载器

Java class loader：java的类加载器，jvm执行时会分析这些java类之间的依赖关系，并调用java加载器来加载相应的字节码文件。

Jvm就是java代码的执行引擎

类加载器负责加载编译好的class文件，然后由执行引擎负责执行这些文件，执行引擎会与os进行交互。

Java也可以直接与os进行交互



Java语言特性：

编译为class字节码文件

面向对象

面向对象以数据为核心，面向过程以指令为核心。

对象=数据+方法

多线程适用于并行架构，实现高效执行

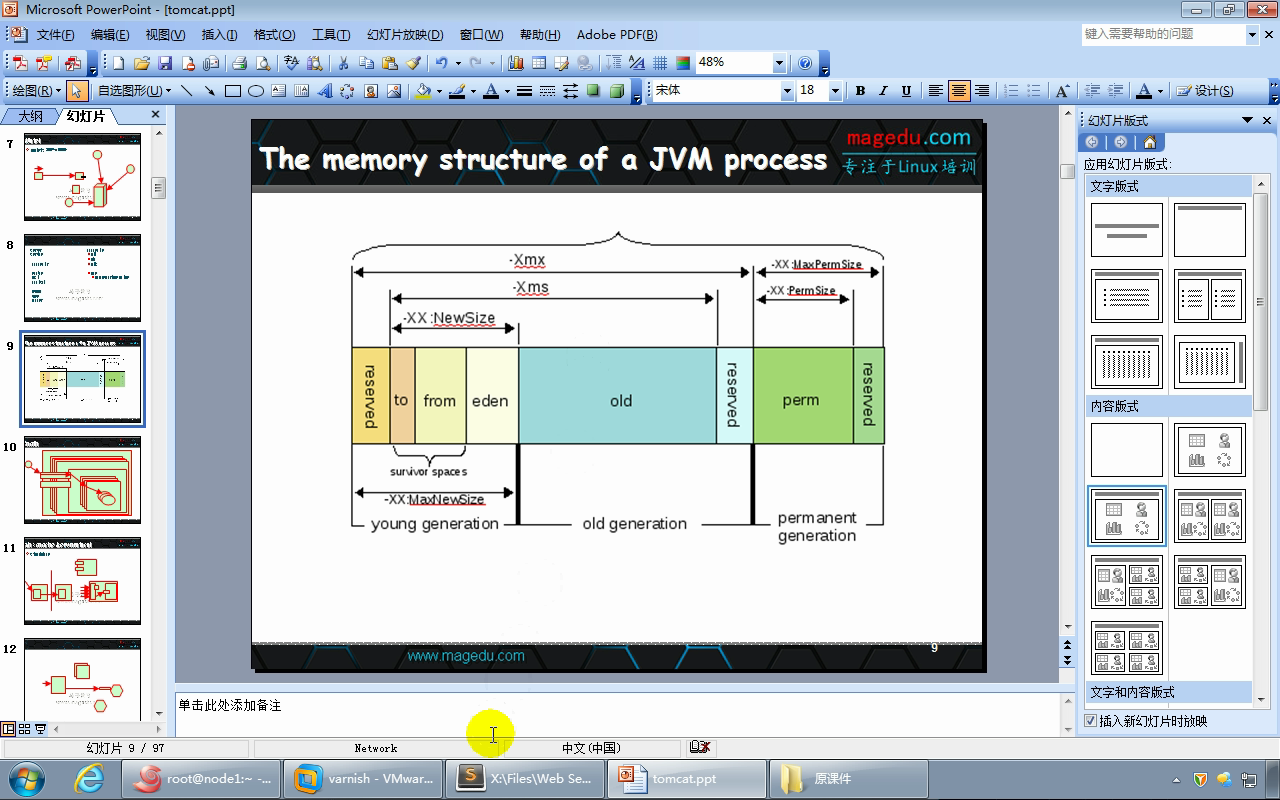
垃圾收集：Gc，后台线程，负责管理java的内存

C语言编程：malloc 和free，程序员自己来管理内存

动态链接：对象的引用计数器

Android:java+jvm+linux

Ios:Object-c+FreeBSD

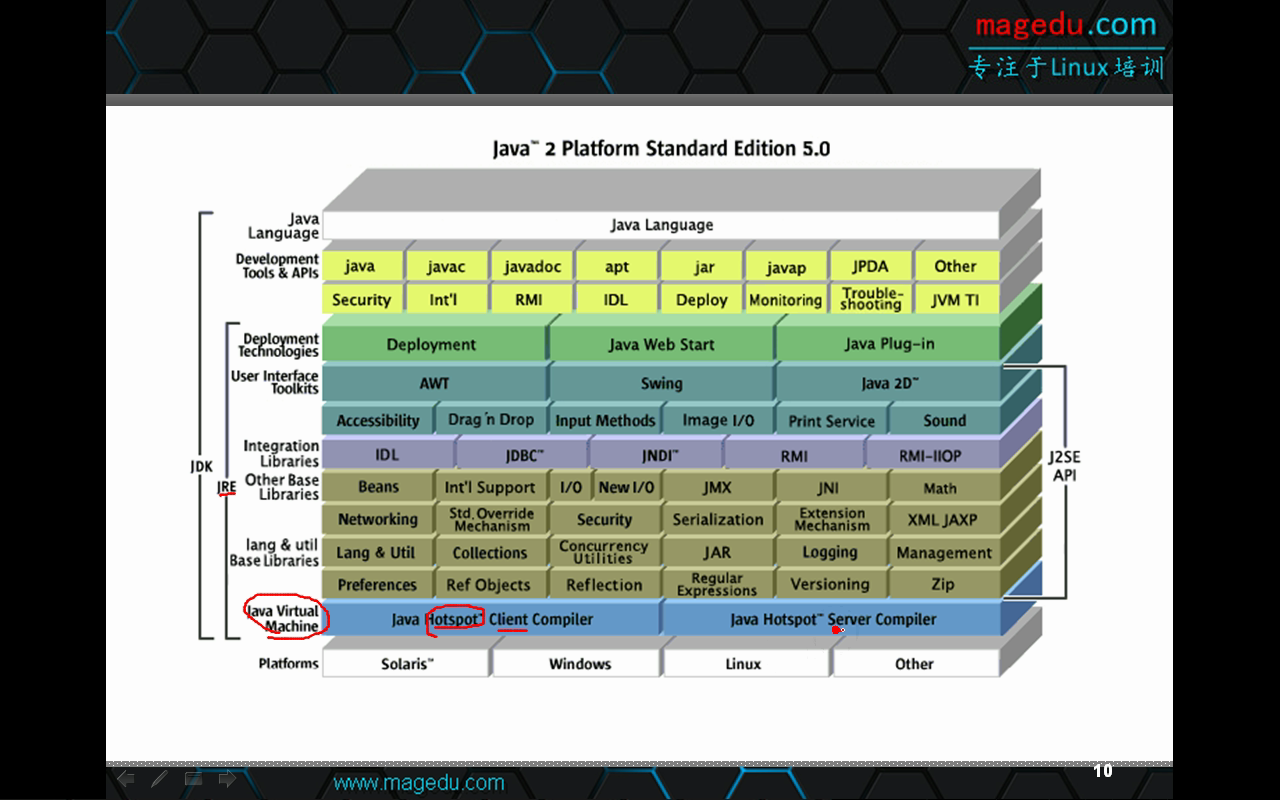


垃圾收集器的老生代、新生代

Jvm=java类加载器+java字节码执行引擎

JRE: jav running environment，通常由jvm和一些基础库程序库以及其他组件组成。

JDK：java程序开发包



方法区

堆

栈

PC：程序计数器

本地方法栈



Servlet

Servlet容器

Servlet缺陷：

Html必须硬编码，即直接写在程序中，然后输出到前端

Jsp的出现：将java程序对象嵌入到html中，从而解决了servlet的缺陷

Jsp分拣器，将存在嵌入在html文档的代码转换为servlet代码。

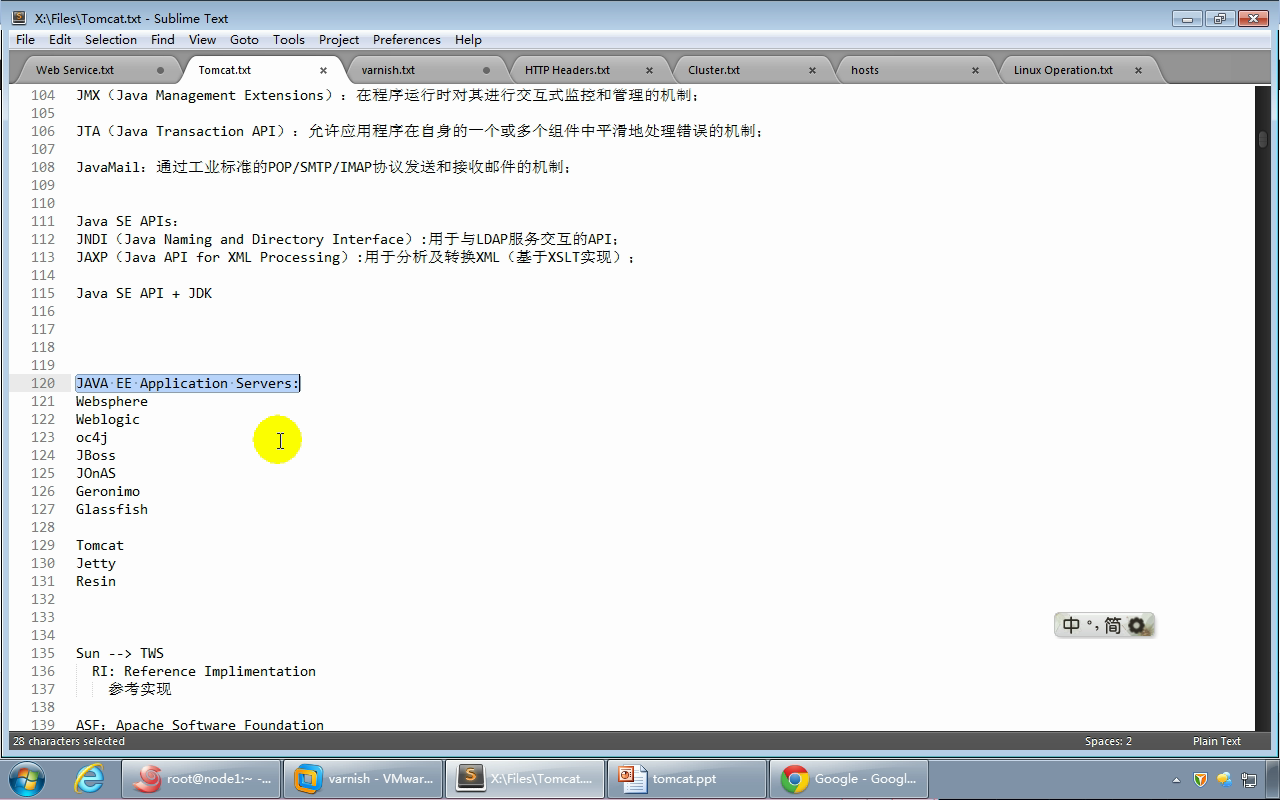
**任何软件工程遇到的问题都可以通过增加一个中间层来解决**

容器🡪servlet类+jsp类+web服务器

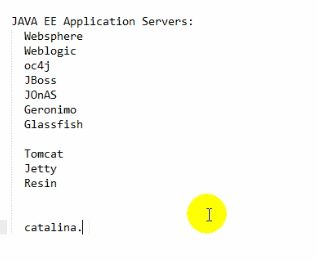
# 08-Java核心技术之安装tomcat



Javaee 应用服务器



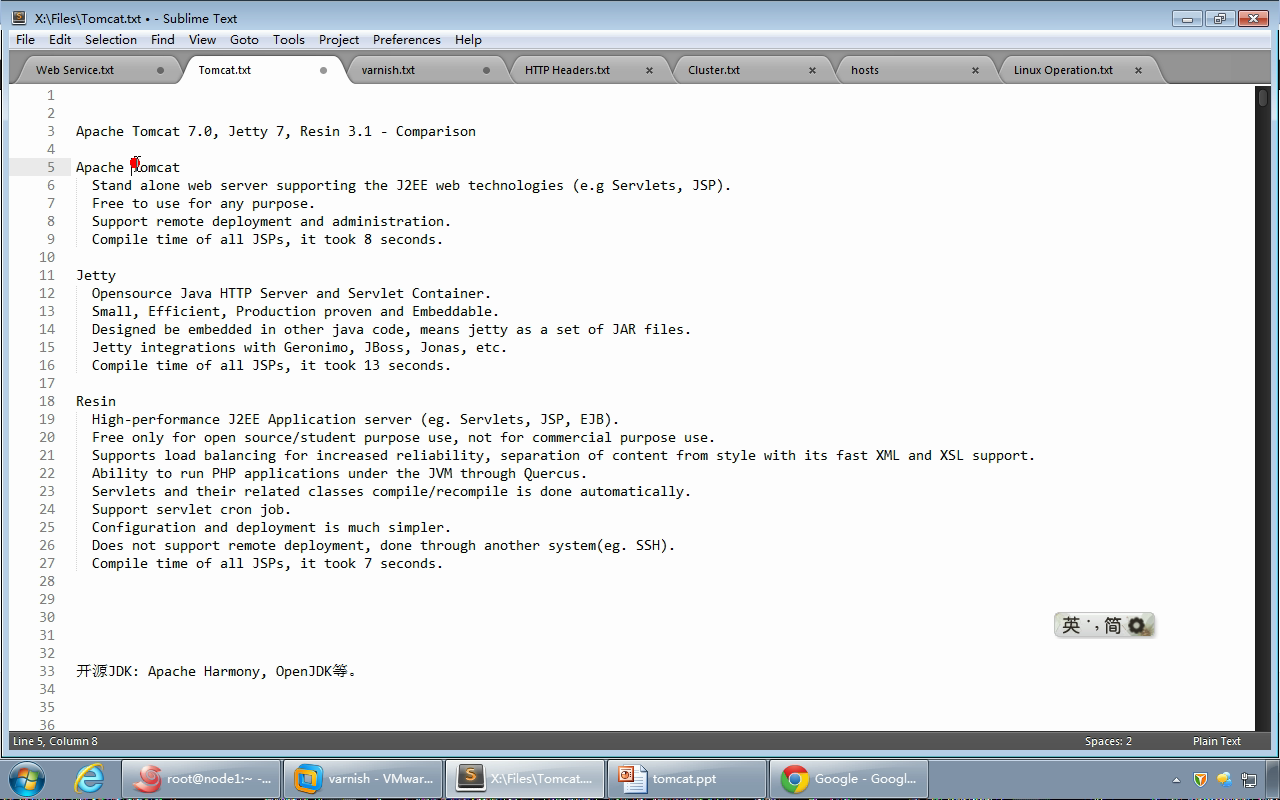
Websphere weblogic

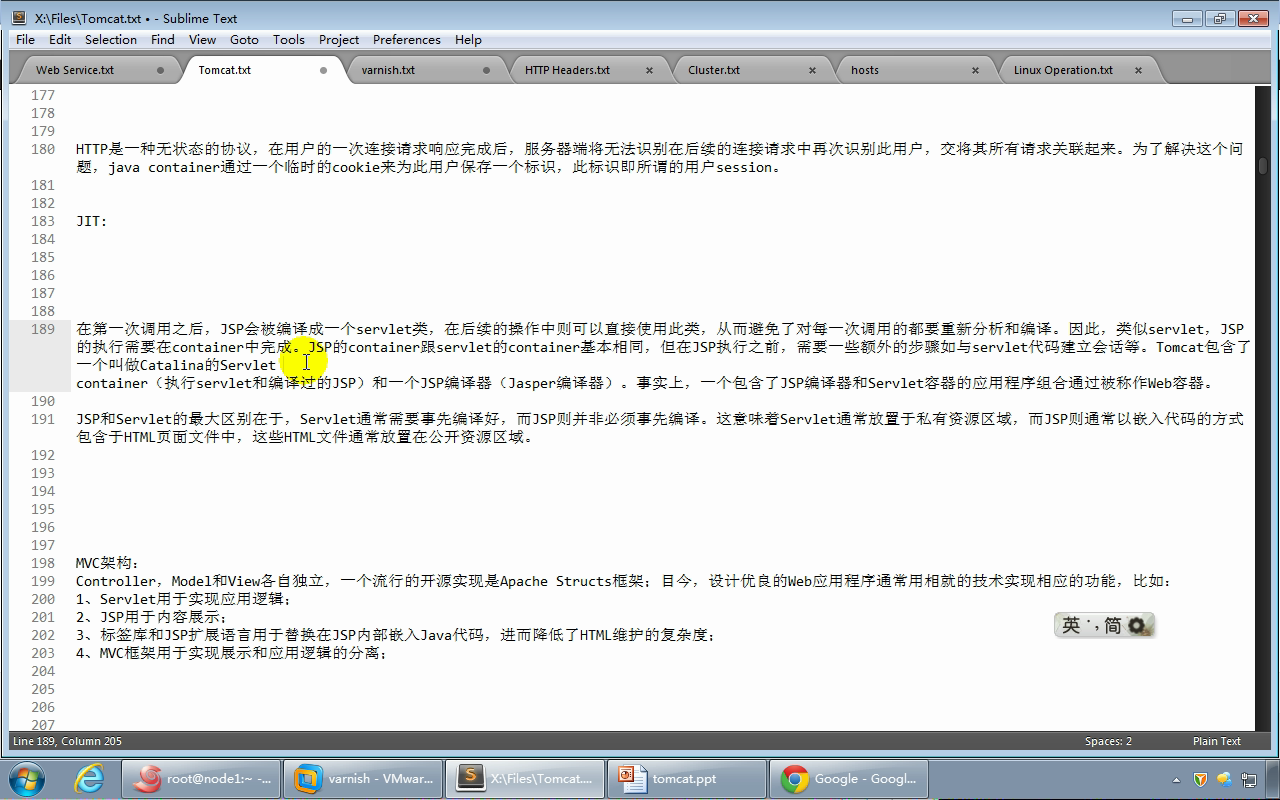


Tomcat仅仅实现了servlet、jsp等

Jboss无非对tomcat做了二次封装，增加了自己的一些管理组件

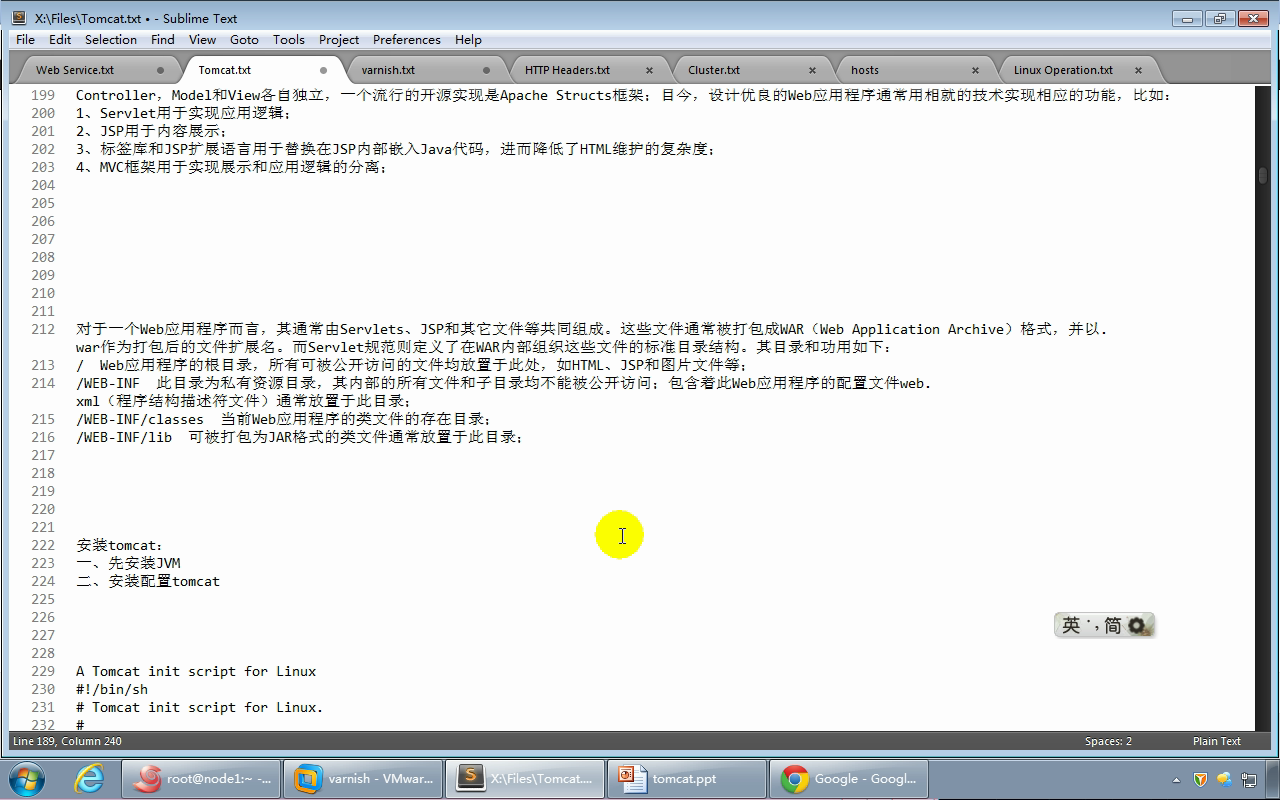
Resin商业





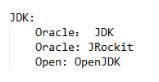
# 09-Java核心技术之安装tomcat-2

Jsp只会被编译一次

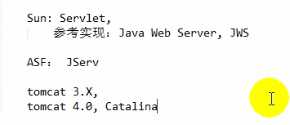


Tomcat和jetty就是由java编写，需要jdk环境

Jdk有多种：



# 10-Java技术回顾



Tomcat只提供了jsp和java servlet

Jsp由jasper转换为servlet代码，然后jdk运行servlet再将servlet转换为html

# 11-tomcat架构组件详解\_rec\_rec

Tomcat server是由组件组成



顶级组件：server ,位于整个配置的最外层

容器类：可包含其他组件的组件

连接器组件：连接用户请求至tomcat

被嵌套的组件：位于一个容器中，不能包含其他组件

容器类：

Engine 核心容器，catalina引擎，负责通过connector接收用户请求，分析并处理，转发到host

Host 类似于httpd中的虚拟主机，支持基于FQDN的虚拟主机

Context 最内层的容器类组件，一个context代表一个web应用程序，配置context的主要目的，指定对应的webapp根目录，还能为webapp指定额外的属性，如部署方式等。



Service 将连接器关联至engine

因此一个service内部可以有多个connector,但只能有一个engine

顶级组件：server，表示一个运行与jvm中的tomcat实例；

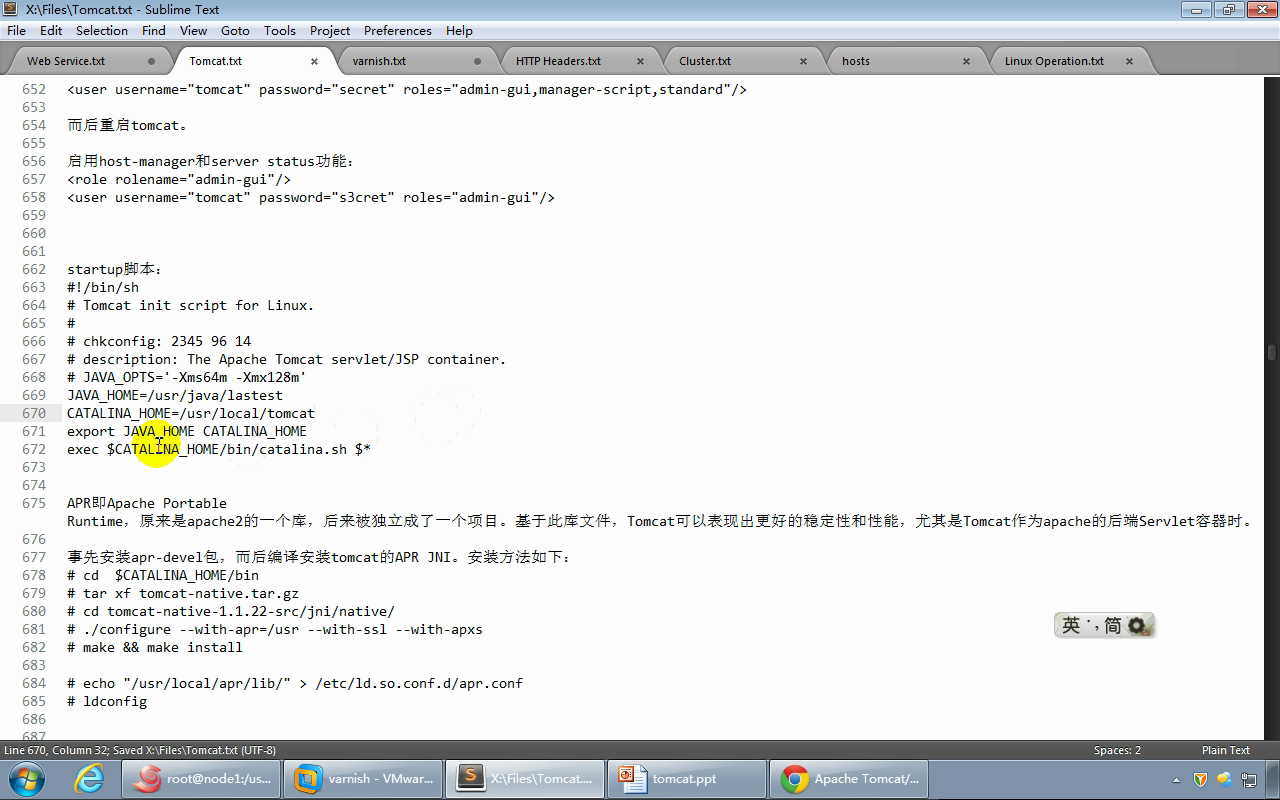
嵌套类组件：valve：拦截请求，并在将其转至对应的webapp之前进行某种处理操作，如access log valve,remote address filter valve：基于ip做访问控制

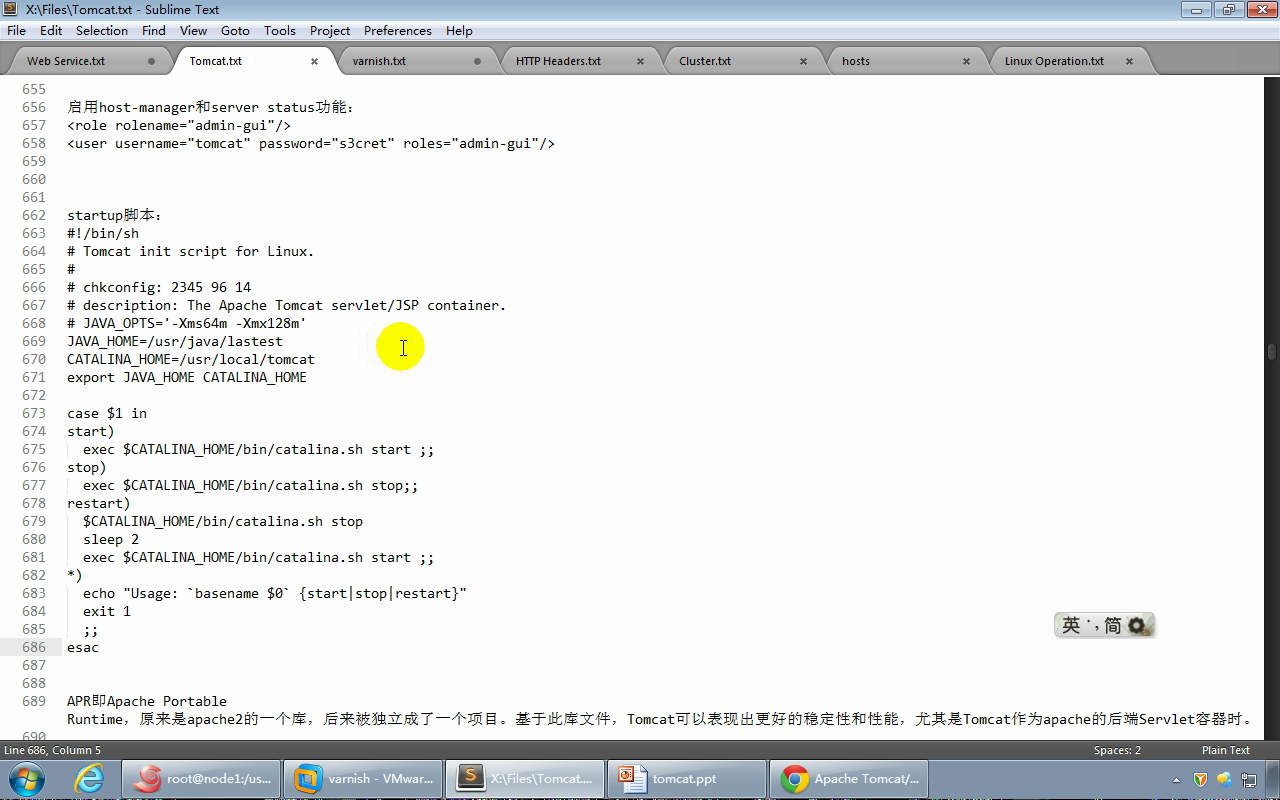
Logger:日志记录器，用于记录组件内部的状态信息

Realm:关联一个用户认证库

# 13-tomcat的运行方式













Maxthread是一个经常需要调整的属性

AcceptCount请求等待队列长度

# 14-tomcat的运行方式-2



Context组件

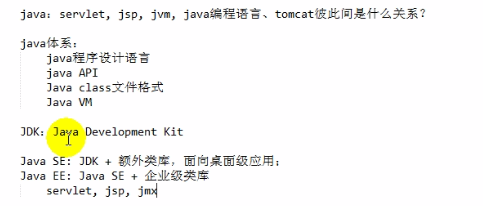


# 15-tomcat配置详解之应用程序目录结构

配置文件在启动时会被读入内存

# 16-tomcat配置详解之安装\_rec\_rec

# 17-tomcat配置详解-3

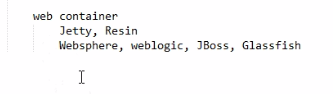




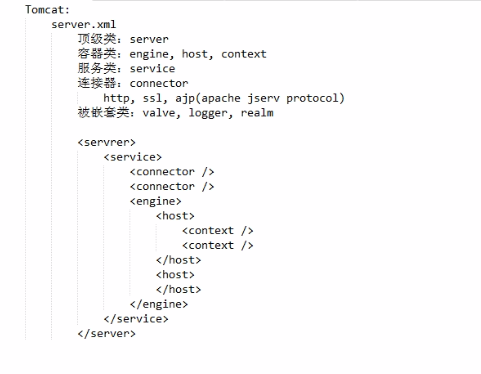
Tomcat就是一个web container

# 18-tomcat配置详解-4

其他web container



Tomcat 核心配置文件：server.xml，配置了各个组件及其之前的关系





# 19-tomcat配置详解-5

Tomcat的配置文件：

Server.xml 核心配置文件

Context.xml 为部署与此tomcat实例上的所有web应用程序提供默认的配置文件，每个webapp都可以使用独有的context.xml，通常放置于webapp目录的META-INF子目录中，常用于定义会话管理器，realm以及jdbc等，此目录不允许从外部访问。

Web.xml 为部署与此tomcat实例上的所有web应用程序提供默认部署描述符，通常用于为webapp提供基本的servlet定义和MIME映射表等。

Tomcat-users.xml 用户配置

Catalia.policy 安全策略配置文件

Catalina.properties java属性定义文件，用于设定加载器路径，安全包列表和一些调整性能相关的参数信息

Logging.properties 定义日志相关的配置文件

# 20-tomcat配置详解之知识点回顾

Tomcat 应用程序的部署：



部署方式：

静态方式：在tomcat启动之前进行web部署

动态部署：在不打断tomcat运行的前提下，通过tomcat manager或其他的命令行工具进行部署

TCD：第三方部署工具

大多静态部署，停止部分服务器

Wbapp体系结构：

Webapp有特定的组织格式，是一种层次性目录结构：通常包含了servlets代码文件，jsp文件，类文件，部署描述符文件等，一般会打包成归档格式

/： web程序的根目录

/WEB-INF 此webapp的私有资源目录，外部不能访问，通web.xml和context.xml均放置在此目录

/WEB-INF/classes 此webapp自有的类

/WEB-INF/lib

/META-INF 非标准，不是必须的

Webapp的归档格式：

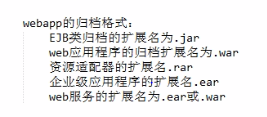
EJB类的归档扩展名为.jar

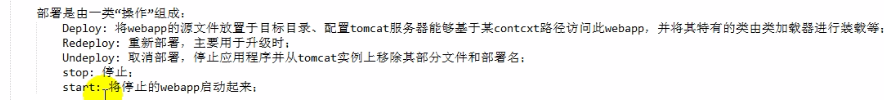
Web应用程序的归档扩展为.war

资源适配器的扩展名为 .rar

企业级应用程序的扩展名为.ear

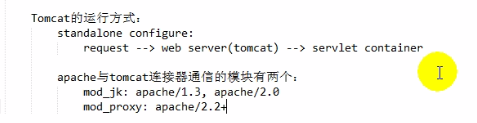
Web服务扩展名为ear或者war





部署方式





# 21-tomcat配置详解之各组件详解

//==================20171023未完待续