

Linux高级运维

NSD OPERATION

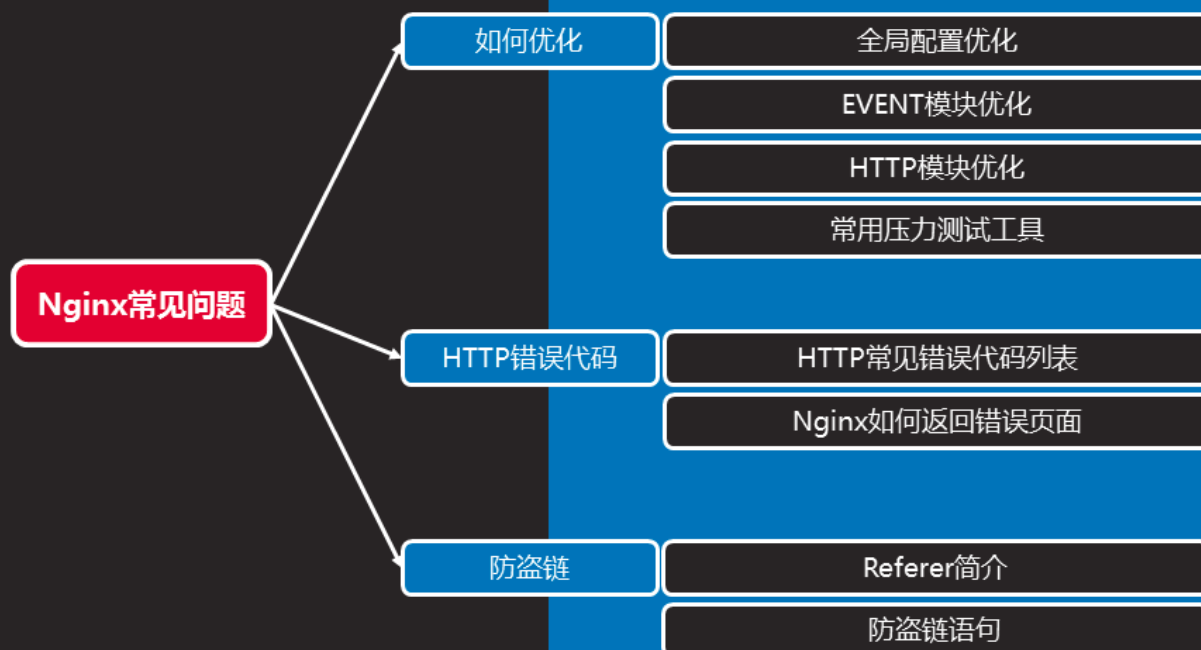
DAY04

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	Nginx常见问题
	10:30 ~ 11:20	
	11:30 ~ 12:20	Tomcat服务器
下午	14:00 ~ 14:50	
	15:00 ~ 15:50	Tomcat高级应用
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑



Nginx常见问题



如何优化

全局配置优化

- 调整进程数量

```
user www www;  
worker_processes 2;  
error_log /var/log/nginx.error_log info;  
# [ debug | info | notice | warn | error | crit ]
```

//与CPU核心数量一致
//定义日志级别

EVENT模块优化

- $\text{max_clients} = \text{worker_processes} * \text{worker_connections}$
- 注意修改系统ulimit限制/etc/security/limits.conf

知识讲解

```
events {  
    worker_connections 10000;           //每个worker最大并发连接数  
    use epoll;  
}
```



HTTP模块优化

知识讲解

```
http {  
    server_tokens off;                //不显示Nginx具体版本号  
    sendfile on;                      //提升Nginx读文件性能  
    tcp_nodelay on;                   //关闭TCP缓延迟发送数据  
    keepalive_timeout 10;             //保持连接的超时时间  
    gzip on;  
    gzip_min_length 1000;  
    gzip_comp_level 4;  
    gzip_types text/plain text/css application/json application/x-javascript text/xml  
    application/xml application/xml+rss text/javascript;  
    client_header_buffer_size 1k;     //默认请求包头信息的缓存  
    large_client_header_buffers 4 4k; //大请求包头部信息的缓存个数与容量  
    //先根据client_header_buffer分配，如果不够，再根据large值分配  
}
```



HTTP模块优化（续1）

- 如果需要处理大量静态文件，需要保持这些文件句柄为打开状态，避免后续再次打开

知识讲解

```
http {  
    open_file_cache      max=2000 inactive=20s;  
        open_file_cache_valid 60s;  
        open_file_cache_min_uses 5;  
        open_file_cache_errors off;  
    //设置服务器最大缓存2000个文件句柄，关闭20秒内无请求的文件句柄  
    //文件句柄的有效时间是60秒，60秒后过期  
    //只有访问次数超过5次会被缓存  
}
```



HTTP模块优化（续2）

- 客户端浏览器缓存数据

知识讲解

```
http {  
    location ~* \.(jpg|jpeg|gif|png|css|js|ico|xml)$ {  
        access_log      off;  
        expires          30d;  
    }  
}
```



常用压力测试工具

知识讲解

- ab
 - ab -c并发数 -n总请求数 URL
- http_load
 - http_load -p 并发数 -s 测试时间 URL
- webbench
 - webbench -c 并发数 -t 测试时间 URL
- siege
 - siege -c 并发数 -r 重复次数 URL



HTTP错误代码



HTTP常见错误代码列表

知识讲解

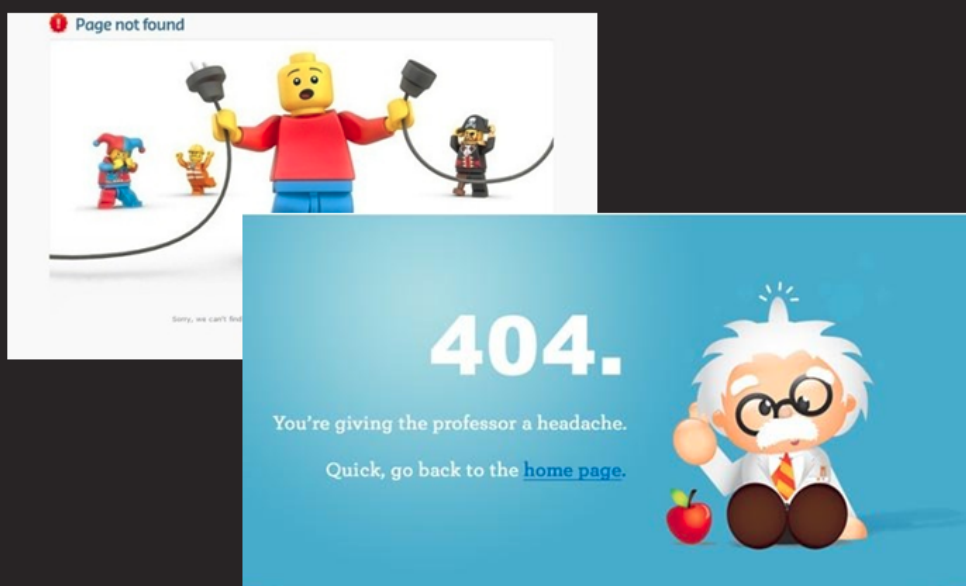
返回码	描述
200	一切正常
400	请求语法错误
401	访问被拒绝（账户或密码错误）
403	资源不可用，通常由于服务器上文件或目录的权限设置导致
403	禁止访问：客户端的IP地址被拒绝
404	无法找到指定位置的资源（Not Found）
414	请求URI头部太长
500	服务器内部错误
502	服务器作为网关或者代理时，为了完成请求访问下一个服务器，但该服务器返回了非法的应答（Bad Gateway）



Nginx如何返回错误页面

- 遇到访问错误时，处理错误代码你还有别的选择

知识讲解



nginx如何返回错误页面（续1）

知识讲解

```
http {  
    fastcgi_intercept_errors on;           //错误页面重定向  
    server {  
        error_page 404 /40x.html;         //自定义错误页面  
            location = /40x.html {  
                root html;  
            }  
        error_page 500 502 503 504 /50x.html;  
            location = /50x.html {  
                root html;  
            }  
    }  
}
```



防盗链



Referer简介

知识讲解

- HTTP请求头部信息
 - HTTP Referer是header的一部分，当浏览器向Web服务器发送请求的时候，一般会带上Referer，告诉服务器我是从哪个页面链接过来的，服务器籍此可以获得一些信息用于处理



防盗链语句

知识讲解

```
[root@localhost ~]# cat /usr/local/nginx/conf/nginx.conf
location ~* \.(gif|jpg|png|swf|flv)$ {
    valid_referers none blocked www.tarena.com;
    if ($invalid_referer) {
        rewrite ^/ http://www.tarena.com/403.html;
    }
}
```



防盗链语句（续1）

知识讲解

- none表示没有Referer，也就是直接访问
 - 比如直接在浏览器打开一个图片
- blocked表示有Referer，但内容被防火墙或代理删除
- server_names就是最后的域名
 - 可以使用*.tarena.com来表示二级域名



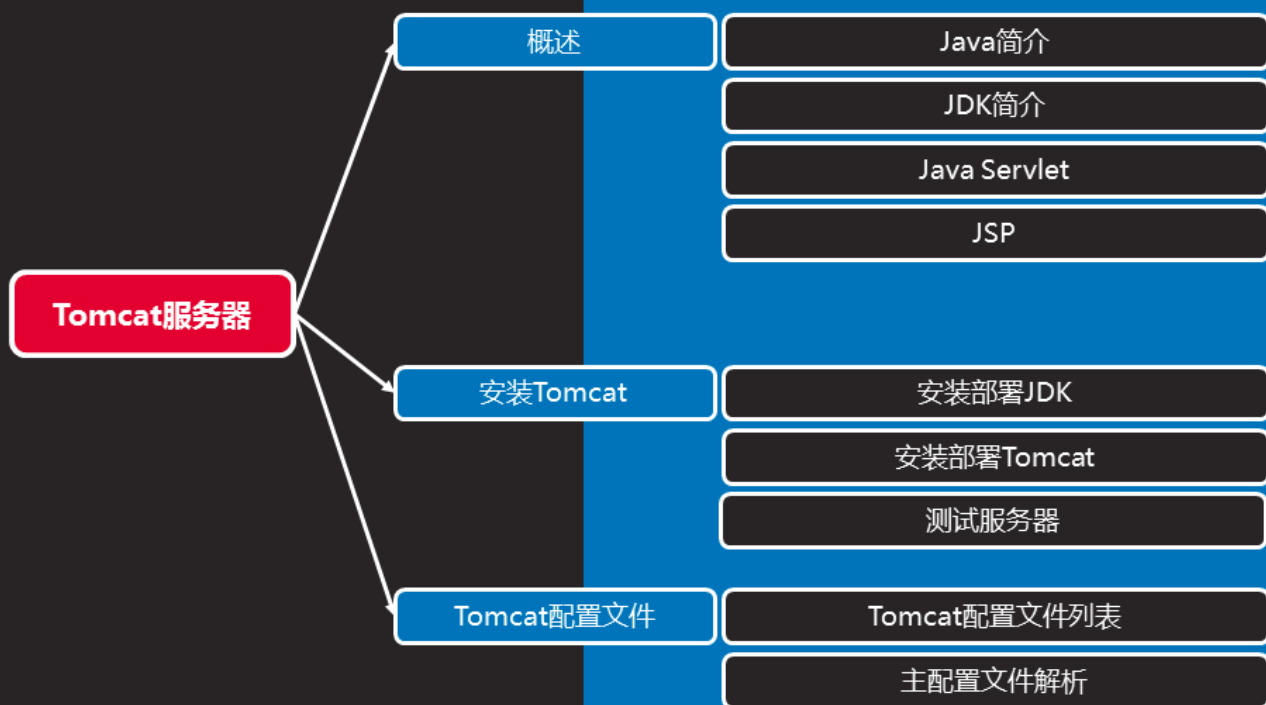
案例1：Nginx常见问题处理

课堂练习

- 不显示Nginx软件版本号
- 浏览网站时提示 “Too many open files” 如何解决
- 如何解决客户端访问头部信息过长的的问题
- 如何让客户端浏览器缓存数据
- 如何自定义返回给客户端的404错误页面



Tomcat服务器



概述

Java简介

知识讲解

- Java
 - java是一种跨平台的、面向对象的程序设计语言，Java技术具有卓越的通用性、高效性、**平台移植性**和安全性。
- Java体系
 - Java SE（标准版）
 - Java EE（企业版）
 - Java ME（移动版）



JDK简介

知识讲解

- JDK(Java Development Kit)是Sun针对Java开发者推出的**Java语言的软件开发工具包**
- JDK已经成为使用最广泛的Java SDK
- JDK是整个Java的核心
 - 包括了Java运行环境
 - Java工具（如编译、排错、打包等工具）
 - Java基础的类库



JDK简介（续1）

知识讲解

- JRE（Java Runtime Environment，Java运行环境），运行JAVA程序所必须的环境的集合，包含JVM标准实现及Java核心类库
- JRE包括
 - Java虚拟机（jvm）
 - Java核心类库和支持文件
 - 不包含开发工具(JDK)--编译器、调试器和其它工具
- JRE是JDK的子集



Java Servlet

知识讲解

- Servlet是一种扩展Web服务器功能的组件规范
- 它能够以一种可移植的方法来提供动态的、面向用户的内容，处理用户请求



Java Servlet (续1)

知识讲解

- 常见Servlet容器
 - IBM websphere
 - Oracle weblogic
 - Apache tomcat
 - RedHat Jboss
- 开发者一般主要开发的是Servlet容器中的Servlet代码



JSP

知识讲解

- JSP (Java Server Page)
 - SUN推出的类似于ASP的**镶嵌型的JSP**，把**JSP TAG镶嵌到HTML语句中**，大大简化和方便了网页的设计和修改



安装Tomcat

安装部署JDK

- 安装jdk1.8

知识讲解

```
[root@svr5 ~]# yum -y install java-1.8.0-openjdk  
[root@svr5 ~]# yum -y install java-1.8.0-openjdk-headless
```



安装部署Tomcat

- 安装Tomcat

知识讲解

```
[root@svr5 ~]# tar -xzf apache-tomcat-8.0.30.tar.gz
[root@svr5 ~]# mv apache-tomcat-8.0.30 /usr/local/tomcat
[root@svr5 ~]# ls /usr/local/tomcat
bin/                //主程序目录
lib/                //库文件目录
logs/               //日志目录
temp/               //临时目录
work/               //自动编译目录jsp代码转换servlet
conf/               //配置文件目录
webapps/             //页面目录
```



安装部署Tomcat (续1)

- 启动Tomcat

知识讲解

```
[root@svr5 ~]# /usr/local/tomcat/bin/startup.sh
Using CATALINA_BASE: /usr/local/tomcat
Using CATALINA_HOME: /usr/local/tomcat
Using CATALINA_TMPDIR: /usr/local/tomcat/temp
Using JRE_HOME: /usr/java/default
Using CLASSPATH:
/usr/local/tomcat/bin/bootstrap.jar:/usr/local/tomcat/bin/tomcat-juli.jar
Tomcat started.
```



测试服务器

知识讲解

- 生产测试页面

```
[root@svr5 ~]# # vim /usr/local/tomcat/webapps/ROOT/test.jsp
<html>
<body>
<center>
Now time is: <%=new java.util.Date()%>
</center>
</body>
</html>
```

- 客户端测试

- firefox http://localhost:8080
- firefox http://localhost:8080/test.jsp



Tomcat配置文件

Tomcat配置文件列表

知识讲解

- server.xml
 - 主配置文件
- context.xml
 - 定义会话管理器、JDBC等
- tomcat-users.xml
 - 用户认证的账号和密码配置文件



主配置文件解析

知识讲解

- server.xml配置文件框架

```
<?xml version='1.0' encoding='utf-8'?>
<Server port="8005" shutdown="SHUTDOWN">
  <Service name="Catalina">
    <Connector port="8080" protocol="HTTP/1.1"
      connectionTimeout="20000"
      redirectPort="8443" />
    <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
    <Engine name="Catalina" defaultHost="localhost">
      <Host name="localhost" appBase="webapps"
        unpackWARs="true" autoDeploy="true">
      </Host>
    </Engine>
  </Service>
</Server>
```



主配置文件解析（续1）

知识讲解

- server.xml配置文件框架

Server是Tomcat实例的顶层元素，一个tomcat实例

Service是一个集合，它由一个或者多个Connector以及一个Engine组成

Connector负责接受用户请求和向客户返回响应结果

Engine负责处理所有Connector所获得的客户请求，它处理在同一个Service中所有Connector元素接收到的客户请求。它匹配请求和自己的虚拟主机，并将请求发给对应的<Host>处理，默认的主机是localhost

一个<Engine>元素可以包含多个<Host>元素，每个<Host>的元素定义一个虚拟主机，它包含一个或多个web应用



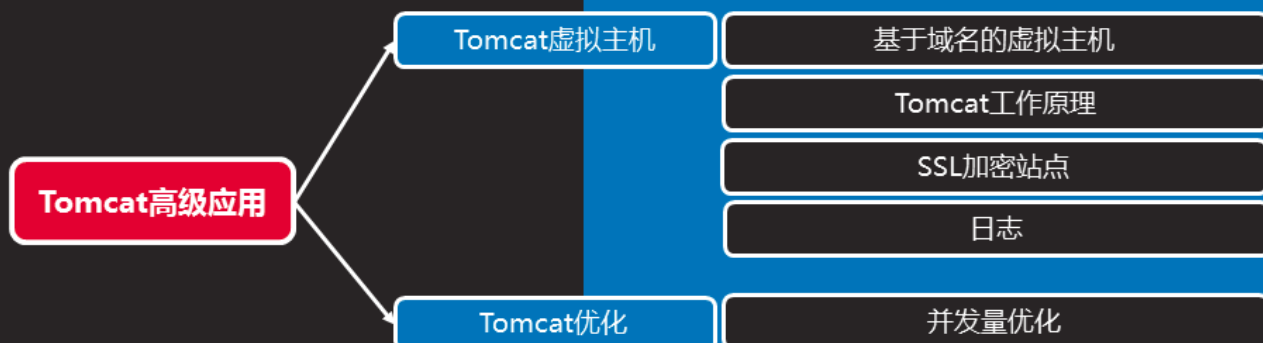
案例2：安装部署Tomcat服务器

课堂练习

- 安装部署JDK环境
- 安装部署Tomcat服务器
- 创建JSP测试页面，文件名为test.jsp，显示服务器当前时间



Tomcat高级应用



Tomcat虚拟主机

基于域名的虚拟主机

知识讲解

- 修改server.xml文档，添加host虚拟主机

```
#vim /usr/local/tomcat/conf/server.xml
... ..
<Host name="www.a.com" appBase="a" unpackWARS="true"
autoDeploy="true">
</Host>
<Host name="www.b.com" appBase="b" unpackWARS="true"
autoDeploy="true">
</Host>
#mkdir -p /usr/local/tomcat/{a,b}/ROOT
#echo "A" > /usr/local/tomcat/a/ROOT/index.html
#echo "B" > /usr/local/tomcat/b/ROOT/index.html
#/usr/local/tomcat/bin/shutdown.sh
#/usr/local/tomcat/bin/startup.sh
```



基于域名的虚拟主机（续1）

知识讲解

- 修改server.xml文档，给host添加context

```
#vim /usr/local/tomcat/conf/server.xml
... ..
<Host name="www.a.com" appBase="a" unpackWARS="true"
autoDeploy="true">
<Context path="" docBase="base" reloadable="true"/>
</Host>
<Host name="www.b.com" appBase="b" unpackWARS="true"
autoDeploy="true"> </Host>
#mkdir -p /usr/local/tomcat/a/base
#echo "base" > /usr/local/tomcat/a/base/index.html
#/usr/local/tomcat/bin/shutdown.sh
#/usr/local/tomcat/bin/startup.sh
```

//appBase定义基础目录，基础目录下可以有很多项目，默认项目ROOT
//docBase定义首页路径，默认为ROOT



基于域名的虚拟主机（续2）

- 修改server.xml文档，给host添加context

```
#vim /usr/local/tomcat/conf/server.xml
```

```
... ..
```

```
<Host name="www.a.com" appBase="a" unpackWARs="true"
autoDeploy="true">
```

```
<Context path="/test" docBase="/var/www/html/" />
```

```
</Host>
```

```
<Host name="www.b.com" appBase="b" unpackWARs="true"
autoDeploy="true"> </Host>
```

```
#echo "test" > /var/www/html/index.html
```

```
#/usr/local/tomcat/bin/shutdown.sh
```

```
#/usr/local/tomcat/bin/startup.sh
```

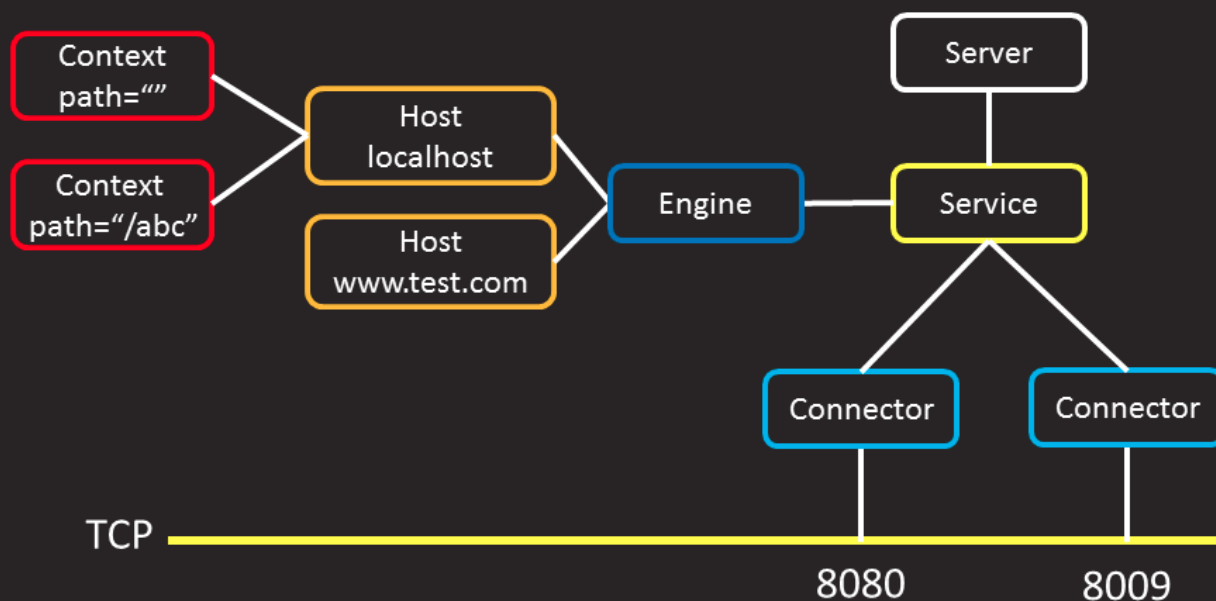
//path指定用户访问的URL,docBase指定页面存储的位置

验证：http://IP:8080/test/

知识讲解



Tomcat工作原理



知识讲解



SSL加密站点

知识讲解

- 生产私钥证书文件

```
# keytool -genkeypair -alias tomcat -keyalg RSA -keystore  
/usr/local/tomcat/keystore
```

- 修改server.xml配置文件

```
#vim /usr/local/tomcat/conf/server.xml
```

```
... ..
```

```
<Connector port="8443"
```

```
protocol="org.apache.coyote.http11.Http11NioProtocol"
```

```
maxThreads="150" SSLEnabled="true" scheme="https" secure="true"
```

```
keystoreFile="/usr/local/tomcat/keystore" keystorePass="123456"
```

```
clientAuth="false" sslProtocol="TLS" />
```



SSL加密站点（续1）

- 客户端测试

```
# firefox https://localhost:8443
```

```
# firefox https://www.a.com:8443
```

知识讲解



日志

- 虚拟主机创建独立日志文件

知识讲解

```
#vim /usr/local/tomcat/conf/server.xml
<Host name="www.a.com" appBase="a" unpackWARs="true"
autoDeploy="true">
  <Context path="" docBase="base" />
  <Valve className="org.apache.catalina.valves.AccessLogValve"
    prefix="wwwacom_access_log." suffix=".txt"
    pattern="common"/>
</Host>
```



Tomcat优化

并发量优化

- 修改Connector属性

知识讲解

```
#vim /usr/local/tomcat/bin/catalina.sh
<Connector port="8080"
    protocol="HTTP/1.1"
    maxThreads="1000"      //客户请求最大线程数
    minSpareThreads="100"
    maxSpareThreads="1000"//
    enableLookups="false"
    URIEncoding="utf-8"
    acceptCount="1000"     //监听端口队列最大数
/>
```



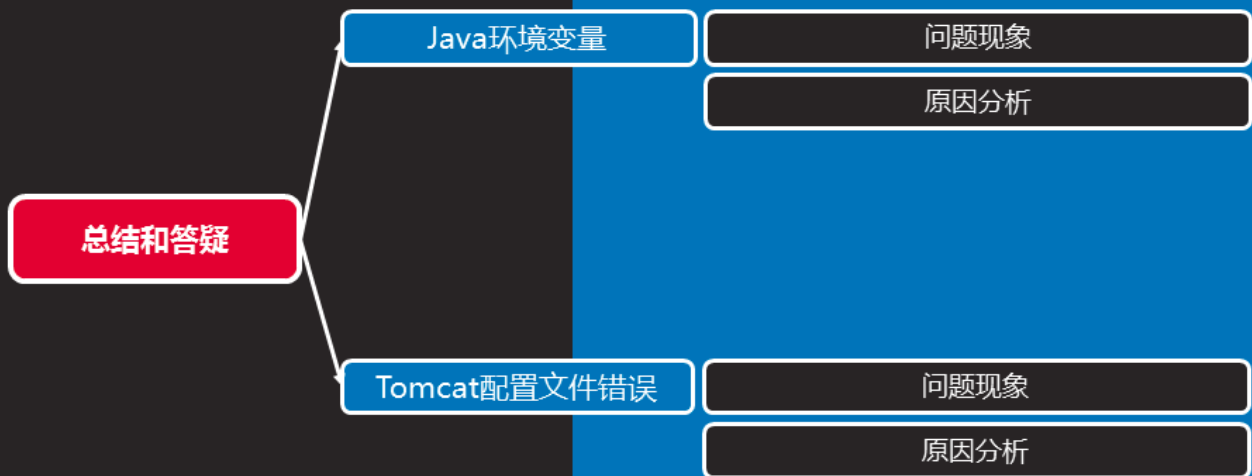
案例3：使用Tomcat部署虚拟主机

- 设置基于域名的虚拟主机，要求如下：
 - 域名：www.test.com www.tomcat.com
 - 访问页面时支持SSL加密通讯
 - 私钥、证书存储路径为/usr/local/tomcat/conf/cert
 - 每个虚拟主机都拥有独立的访问日志文件

课堂练习



总结和答疑



Java环境变量

问题现象

- 故障错误信息

```
[root@svr5 ~]# /usr/local/tomcat/bin/catalina.sh start
```

```
Neither the JAVA_HOME nor the JRE_HOME environment variable is defined
```

知识讲解



原因分析

- 分析故障信息
 - Tomcat启动时无法找到java
- 分析故障原因
 - 未安装jdk
 - 或者安装jdk后没有设置正确的环境变量
 - 使用命令java -version查看版本

知识讲解



Tomcat配置文件错误

问题现象

- 故障错误信息

```
[root@svr5 nginx-1.8.0]# vim /usr/local/tomcat/conf/server.xml
```

该文件语法严格，容易出错

原因分析

知识讲解

- 分析故障
 - 大小写错误，如<host> </Host>
 - 关键词不匹配，如<Host>无结束的</Host>
 - 位置错误，如将<Host>写到<Engine>的上面
 - 默认的localhost站点名称被修改
- 分析故障原因
 - Tomcat严格区分大小写
 - <Host>和</Host>为一对，不可缺少
 - 每个容器仅可以包含有效的信息，注意位置问题
 - 修改localhost站点名称后，访问服务时会找不到默认站点

