**Web应用的运行和维护**

1. **Web应用的运行**

本文描述web应用在服务器上启动、访问的原理及过程，主要从jsp+Servlet模式角度阐述。

* 1. .web服务器

Web服务器是指能够为发出请求的浏览器提供文档的程序，应用层使用的是HTTP协议。目前最主流的三个Web服务器是Apache Nginx IIS。

* 1. .web容器

Web容器是一种服务器程序，例如Tomcat默认端口8080就提供了一个相应的服务程序，一个服务器可以有多个容器。

web应用启动做了什么？

web容器读取web.xml文件

- web.xml常用配置参数：

1).context-param(上下文参数)

2).listener(监听器配置参数)

3).filter(过滤器配置参数)

4).servle(servlet页面参数)

5).其他

- 读取顺序：1)>>>2)>>>3)>>>4)(5))(具体对应的执行顺序也是如此)

- 原理及过程

1)应用在服务器上启动，容器按上述顺序自动读取web.xml中的参数

2)容器创建一个上下文对象，即ServletContext，整个web应用共享这个对象

3)容器将读取到的context-param配置参数转化为键值对，封装到ServletContext

4)创建监听器对象(Servlet容器实例化所有监听器，回调无参构造方法，调用init初始化方法)

5)创建过滤器对象(Servlet容器实例化所有过滤器，回调无参构造方法，调用init初始化方法)

注意：此时只是web应用启动时的读取文件以及初始化阶段，客户端没有发送的请求的情况下不会有其他任何操作

③.浏览器到服务器解析？

A.浏览器发送请求到服务器

1.在浏览器地址栏输入请求地址，浏览器解析请求URL，解析出IP地址等信息（域名解析）

2.解析后的URL信息封装成HTTP报文

3.浏览器创建一个socket调用send()方法将HTTP报文发送到服务器(以流的方式)

B.服务器接收到浏览器请求

1.web容器也会创建一个socket并且调用receive()方法接收浏览器发送过来的HTTP报文(携带有IP地址，请求参数等信息)

2.服务器解析请求信息，将请求到的静态或动态资源利用sokcet调用send()方法发送给浏览器

3.浏览器利用sokcet调用receive()方法接收服务器端发送的资源，例如html，jsp等

4.浏览器解析资源，有可能进行再次请求(例如请求css/js等文件资源)，遇到有html或jsp代码将对脚本标签进行渲染

④.jsp？

什么是jsp？

jsp是一种运行在服务器端的脚本语言，是一种软件开发者可以动态响应给客户端的web网页技术标准。

jsp运行原理及过程？

-本质上jsp只是一个带有.jsp后缀的脚本语言文件，并不能直接在jvm(虚拟机)上运行，所以需要将其转换为一个Servlet类，才能通过jvm解释执行：

1)客户端发送请求到服务器，web容器通过jsp引擎将jsp文件转换为一个Servlet类源文件(.java)(如果是第一次访问或jsp文件被修改过，那么将会重新转换为一个新的Servlet类)

2)转换过来的Servlet类进一步被java编译器编译成字节码文件(.class)

3)字节码文件被系统类加载器装载到jvm上，开始初始化、执行

⑤.Servlet？

什么是Servlet？

Java Servlet 是运行在 Web 服务器或应用服务器上的程序,它是作为来自 Web 浏览器或其他 HTTP 客户端的请求和 HTTP 服务器上的数据库或应用程序之间的中间层。

Servlet创建？

1.创建一个类，实现HttpServlet接口（HttpServlet继承于GenericServlet，GenericServlet（实现类）继承于Servlet接口）

2.右键—新建Servlet(选择配置文件还是注解虚拟路径)

3.重写doGet()和doPost()方法处理请求

Servlet运行原理？

1.Servlet没有main方法不能直接独立运行。所以需要依赖于web容器（如tomcat），由这个容器来控制servlet从创建到销毁的过程

2.容器启动的时候，只会读取xml配置文件，如果不发生请求servlet是不会被加载运行的

3.客户端第一次发送请求，服务器接收到请求后，解析请求参数并封装成该请求对应的HttpServletRequest和HttpServletResponse对象以给容器使用

4.根据URL匹配容器中的servlet实例，若没有容器将创建和实例化一个servlet，若有则进入下一步（实例化之前类加载器将字节码文件加载过来）

5.容器调用servlet实例的init()方法完成初始化工作

6.调用service方法处理请求，判断请求方式是get还是post，再调用业务逻辑层的方法完成数据交互

7.将请求资源响应给客户端

8.服务器关闭（或者更新servlet文件），servlet实例调用destroy()方法，servlet实例被销毁

Servlet生命周期？

1.当servlet实例调用init()方法实例化的时候，servlet的生命周期开始

2.当servlet实例调用destroy()方法的时候，servlet被销毁，生命周期结束

注意：

- 1.servlet实例只会被加载一次，init()方法、destroy()方法也只会被调用一次

- 2.只有创建的对象会每请求一次就会创建一对（requset和response）

- 3.如果需要在实例被创建和加载的时候就完成初始化，则可以如下设置参数

@requires\_authorization

<servlet>

<servlet-name>类名</servlet-name>

<servlet-class>包名.类名</servlet-class>

<load-on-startup>优先级别（数字）</load-on-startup>

</servlet>

4.对于某些servlet没有元素，但又想提前初始化则可以进行上述配置，否则需要配置参数

5.get和post请求

a. get请求只能以字符串形式传递参数名和值，直接跟在URL后，安全性 差;post请求在请求头中，较为安全，并且传递数据没有限制

b. get传输数据量较小；post传输数据量大

c. get请求服务器接收效果不好，不能判断是否完成发送；post方式是先发送长度，可以判断是否发送完成

**二、Web应用的维护**

Web服务器的日常维护是网管的一项重要工作，主要工作有：入侵检测、数据备份、服务器优化、常见故障处理以及日志安排等一系统日常维护，服务器管理工作必须规范严谨。

①、入侵检测和数据备份

（一）入侵检测工作

作为服务器的日常管理，入侵检测是一项非常重要的工作，在平常的检测过程中，主要包含日常的服务器安全例行检查和遭到入侵时的入侵检查，也就是分为在入侵进行时的安全检查和在入侵前后的安全检查。系统的安全性遵循木桶原理，木桶原理指的是：一个木桶由许多块木板组成，如果组成木桶的这些木板长短不一，那么这个木桶的最大容量不取决于长的木板，而取决于最短的那块木板。应用到安全方面也就是说系统的安全性取决于系统中最脆弱的地方，这些地方是日常的安全检测的重点所在。

日常的安全检测

日常安全检测主要针对系统的安全性，工作主要按照以下步骤进行：

1、查看服务器状态：

打开进程管理器，查看服务器性能，观察CPU和内存使用状况。查看是否有CPU和内存占用过高等异常情况。

2、检查当前进程情况

切换“任务管理器”到进程，查找有无可疑的应用程序或后台进程在运行。用进程管理器查看进程时里面会有一项taskmgr，这个是进程管理器自身的进程。如果正在运行windows更新会有一项wuauclt.exe进程。对于拿不准的进程或者说不知道是服务器上哪个应用程序开启的进程，可以在网络上搜索一下该进程名加以确定[进程知识库：http://www.dofile.com/]。通常的后门如果有进程的话，一般会取一个与系统进程类似的名称，如svch0st.exe，此时要仔细辨别[通常迷惑手段是变字母o为数字0，变字母l为数字1]

3、检查系统帐号

打开计算机管理，展开本地用户和组选项，查看组选项，查看administrators组是否添加有新帐号，检查是否有克隆帐号。

4、查看当前端口开放情况

使用activeport，查看当前的端口连接情况，尤其是注意与外部连接着的端口情况，看是否有未经允许的端口与外界在通信。如有，立即关闭该端口并记录下该端口对应的程序并记录，将该程序转移到其他目录下存放以便后来分析。打开计算机管理==》软件环境==》正在运行任务[在此处可以查看进程管理器中看不到的隐藏进程]，查看当前运行的程序，如果有不明程序，记录下该程序的位置，打开任务管理器结束该进程，对于采用了守护进程的后门等程序可尝试结束进程树，如仍然无法结束，在注册表中搜索该程序名，删除掉相关键值，切换到安全模式下删除掉相关的程序文件。

5、检查系统服务

运行services.msc，检查处于已启动状态的服务，查看是否有新加的未知服务并确定服务的用途。对于不清楚的服务打开该服务的属性，查看该服务所对应的可执行文件是什么，如果确定该文件是系统内的正常使用的文件，可粗略放过。查看是否有其他正常开放服务依存在该服务上，如果有，可以粗略的放过。如果无法确定该执行文件是否是系统内正常文件并且没有其他正常开放服务依存在该服务上，可暂时停止掉该服务，然后测试下各种应用是否正常。对于一些后门由于采用了hook系统API技术，添加的服务项目在服务管理器中是无法看到的，这时需要打开注册表中的HKEY\_LOCAL\_MACHINE—SYSTEM—CurrentControlSet—Services项进行查找，通过查看各服务的名称、对应的执行文件来确定是否是后门、木马程序等。

6、查看相关日志

运行eventvwr.msc，粗略检查系统中的相关日志记录。在查看时在对应的日志记录上点右键选“属性”，在“筛选器”中设置一个日志筛选器，只选择错误、警告，查看日志的来源和具体描述信息。对于出现的错误如能在服务器常见故障排除中找到解决办法则依照该办法处理该问题，如果无解决办法则记录下该问题，详细记录下事件来源、ID号和具体描述信息，以便找到问题解决的办法。

7、检查系统文件

主要检查系统盘的exe和dll文件，建议系统安装完毕之后用dir \*.exe /s >1.txt将C盘所有的exe文件列表保存下来，然后每次检查的时候再用该命令生成一份当时的列表，用fc比较两个文件，同样如此针对dll文件做相关检查。需要注意的是打补丁或者安装软件后重新生成一次原始列表。检查相关系统文件是否被替换或系统中是否被安装了木马后门等恶意程序。必要时可运行一次杀毒程序对系统盘进行一次扫描处理。

8、检查安全策略是否更改

打开本地连接的属性，查看“常规”中是否只勾选了“TCP/IP协议”，打开“TCP/IP”协议设置，点“高级”==》“选项”，查看“IP安全机制”是否是设定的IP策略，查看“TCP/IP”筛选允许的端口有没有被更改。打开“管理工具”=》“本地安全策略”，查看目前使用的IP安全策略是否发生更改。

9、检查目录权限

重点查看系统目录和重要的应用程序权限是否被更改。需要查看的目录有c: c:winnt C:winntsystem32 c:winntsystem32/inetsrv

c:winntsystem32/inetsrvdata c:documents and Settings然后再检查serv-u安装目录，查看这些目录的权限是否做过变动。检查system32

下的一些重要文件是否更改过权限，包括：cmd，net，ftp，tftp，cacls等文件。

10、检查启动项

主要检查当前的开机自启动程序。可以使用AReporter来检查开机自启动的程序。

（二）数据备份和数据恢复

数据备份工作大致如下：

1、每月备份一次系统数据。

2、备份系统后的两周单独备份一次应用程序数据，主要包括IIS、serv-u、数据库等数据。

3、确保备份数据的安全，并分类放置这些数据备份。因基本上采用的都是全备份方法，对于数据的保留周期可以只保留该次备份和上次备份数据两份即可。

数据恢复工作：

1、系统崩溃或遇到其他不可恢复系统正常状态情况时，先对上次系统备份后发生的一些更改事件如应用程序、安全策略等的设置做好备份，恢复完系统后再恢复这些更改。

2、应用程序等出错采用最近一次的备份数据恢复相关内容。