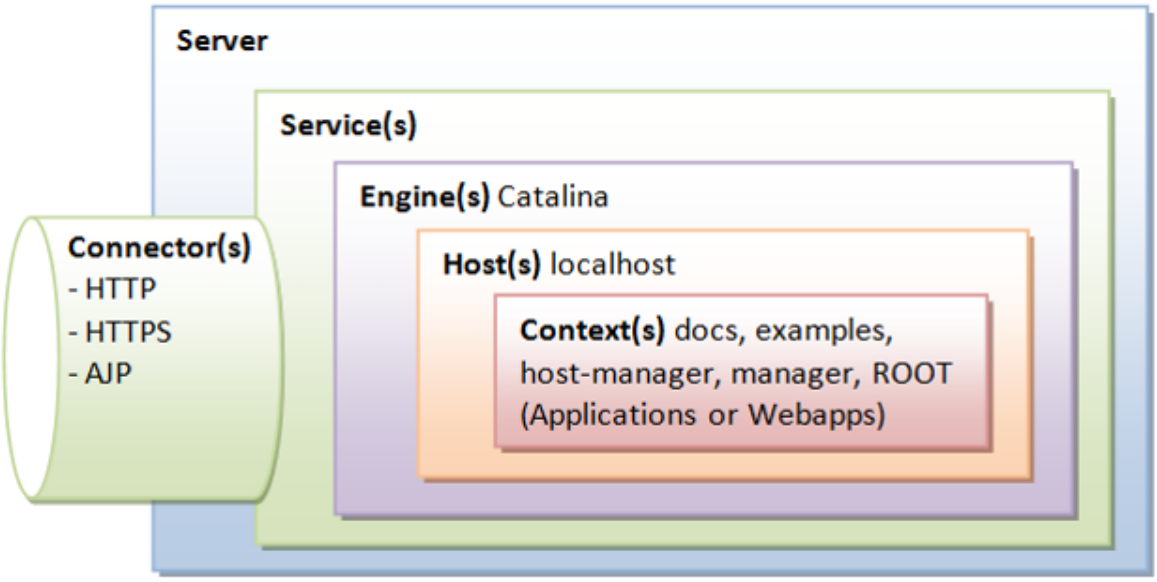
> `Tomcat` 是 Apache 软件基金会的一个项目，实现了对 `JavaServer Page` 的支持，由于其本身具有独立HTTP服务功能，所以 Tomcat 可被视为一个独立的 Web 服务器。Sun 公司创建了第一个 `Servlet 容器`，即 Java Web Server，但 JWS 只是为了演示 Servlet 的相应功能，所以其很不稳定。与此同时，ASF 创建了 JServ 项目，一个能够与 apache 整合起来的 servlet 容器。1999 年，Sun 把 JWS 捐给了 ASF，于是两个项目合二为一，即今天 Tomcat 的前身。第一个tomcat 版本是 Tomcat 3.x 系列，而发布于2001年 Tomcat4.0 则是在此前基础上进行了重新设计和实现，其代码项目被命名为 `Catalina`。
>
> Tomcat 实现了 JACA 2 EE 众多 API 当中的 Servlet、JSP、JMX。是 JAVA 2 EE 平台的一个不完整实现。Tomcat 也是可以作为一个独立的 WEB 服务器。Tomca t也是使用 JAVA 语言开发。

Tomcat核心功能

> `Catalina`：是 Tomcat 的 servlet 容器，实现了Sun Microsystems 对 Servlet 和 JavaServer Pages(JSP) 的规范。
>
> `Coyote`：这允许 Catalina，名义上是一个 Java Servlet 或 JSP 容器，也可以作为一个简单的 Web 服务器，它将本地文件作为 HTTP 文档提供服务。
>
> `Jasper`：JSP 引擎，Jasper 检测到对 JSP 文件的更改并重新编译它们。
>
> `Cluster`：集群负载均衡
>
> `High availability`：高可用性。通过在主端口上升级主服务器时将实时流量请求分派到不同端口上的临时服务器来完成的。在处理高流量Web应用程序的用户请求时非常有用。
>
> `Web application`：用户以及基于系统的Web应用程序增强功能。

Tomcat组件



> `Server`：一个tomcat实例。Tomcat Instance，运行中的tomcat进程。
>
> `Engine`：tomcat核心组件，用于运行jsp或servlet代码。一个engine内可以有多个主机。
>
> `Connector`：连接器，接受并解析用户请求，将请求映射为Engine中运行的代码，之后将运行结果构建为响应报文。支持http、ajp协议。连接器。可以有多个，接收不同的请求，或者接收不同端口发来的请求。
>
> `Service`：将连接器和引擎关联起来。将连接器关联至engine，一个service只能包含一个engine组件，可以包含一个或多个连接器。
>
> `Host`：类似于httpd中的虚拟主机。只可以基于主机名的 "虚拟主机"。
>
> `context`：主机上下文。
>
> 每一个组件都由一个java"类"实现，这些组件大体可分为以下几个类型：顶级组件：`Server`，服务类组件：`Service`，连接器组件：`connector`，容器类：`Engine`,`Host`,`Context`，被嵌套类：`valve`,`logger`,`realm`,`loader`,`manager`，集群类组件：`listener`,`cluster`。

Tomcat运行模式

> **Standalone**
>
> 单独接受请求。通过内置的Web server (http connector)来接收客户端请求。
>
> **反向代理模型**
>
> 由专用的web服务器httpd或者nginx接收请求，web服务器再将请求代理到tomcat。反向代理服务器可以实现动静分离的效果。
>
> **web和tomcat分离运行模型**
>
> 接收http请求的web服务器和tomcat分离。

Tomcat安装部署

> 部署Tomcat有两种方式，使用系统yum仓库提供的jdk安装包进行安装，另一种方式是使用官方提供的rpm包安装。

```
# 使用yum仓库提的rpm包安装
yum install java-1.8.0-openjdk
yum install tomcat tomcat-admin-webapps.noarch
tomcat-docs-webapp.noarch tomcat-webapps.noarch
# 管理工具：tomcat-admin-webapps.noarch
# Tomcat文档：tomcat-docs-webapp.noarch tomcat
# 示例应用：tomcat-webapps.noarch tomcat
```

> 使用自行下载的rpm包安装。下载安装相应的安装包，安装后默认路径为`/usr/java/`

```
rpm -ivh jdk-8u152-linux-x64.rpm
# 查看安装后生成目录文件
ls -l /usr/java/jdk1.8.0_152
/usr/java/jdk1.8.0_152/bin/      # 二进制程序目录
    /usr/java/jdk1.8.0_152/bin/javac     # java编译器
    /usr/java/jdk1.8.0_152/bin/java      # java虚拟机
/usr/java/jdk1.8.0_152/db/       # 数据库访问相关的脚本或类库
    /usr/java/jdk1.8.0_152/db/lib        # 类库
/usr/java/jdk1.8.0_152/include   # 头文件
/usr/java/jdk1.8.0_152/jre       # 运行时环境
/usr/java/jdk1.8.0_152/lib       # 类库
```

> 配置 JAVA_HOME 环境变量，指向 java 的安装路径。

```
vim /etc/profile.d/java.sh
 export JAVA_HOME=/usr/java/latest
 export PATH=$JAVA_HOME/bin:$PATH
```

> 查看JAVA版本信息，验证是否安装成功。

```
java -version
```

> 安装tomcat，解压tomcat安装包至指定目录。

```
tar -xf apache-tomcat-8.5.23.tar.gz -C /usr/local/
# 创建软链接
ln -sv apache-tomcat-8.5.23/ tomcat
# tomcat的目录结构
# bin：脚本，及启动时用到的类
# conf：配置文件目录
# lib：库文件，Java类库，jar
# logs：日志文件目录
# temp：临时文件目录
# webapps：webapp的默认目录
# work：工作目录，存放编译后的字节码文件
```

> 导出tomcat环境变量

```
vim /etc/profile.d/tomcat.sh
 export CATALINT_HOME=/usr/local/tomcat
 export PATH=$CATALINT_HOME/bin:$PATH
```

> catalina.sh脚本使用

```
run                 当前窗口运行，而不在后台运行
run -security       使用安全管理器启动
start               启动tomcat
stop                停止toncat
stop n              等待n秒后，停止tomcat
stop -force         停止Catalina，等待5秒钟，然后使用kill -KILL（如果仍在运行）
stop n -force       等待n秒后，将强制停止tomcat
configtest          检查server.xml配置文件有没有语法错误
version             查看tomcat版本信息
```

> 创建toncat用户，并将相关文件的属组属主修改为tomcat

```
useradd tomcat
chown -R root:tomcat /usr/local/tomcat/*
chown -R tomcat:tomcat /usr/local/tomcat/{conf,log}
```

> 启动tomcat

```
catalina.sh start
# 访问tomcat 172.18.26.1:8080
```