# 

# 

# **Assessment**

# **Application Server**

Trainee Name : Aditya Upadhyay

Newers ID : 3978

Mentor Name : Tarun Saxena

College : UPES

**1.What is the difference between an Application Server and a Web Server?**

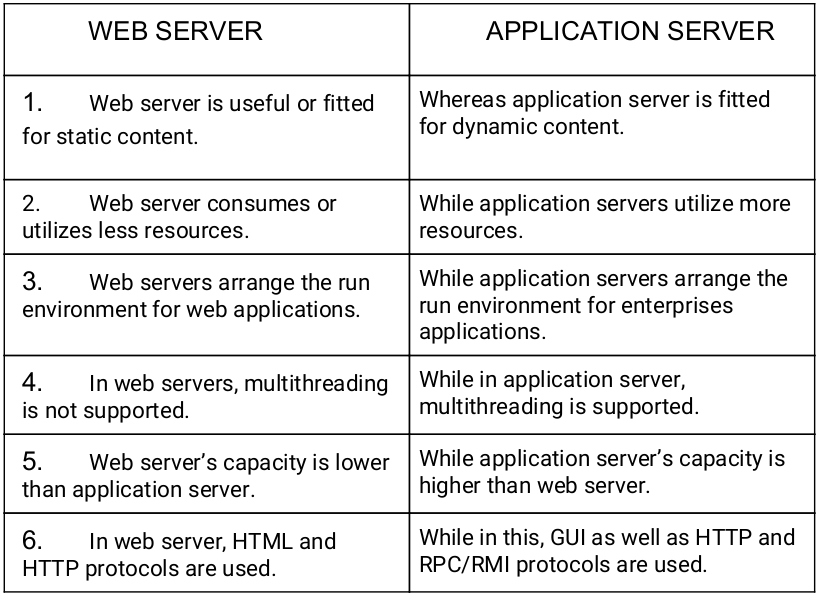
There is not much difference between Application server and web server, one is used to serve HTTP request and other is used to store business logic.

Basically, web server are used before application server, the reasons are

Security: Our business logic is stored in the application server, we want to hide it.

SSL offloading: We don’t have to manage multiple SSL certificate, hence we only configure HTTPS on web server.

Web servers are faster and lightweight and hence are used to serve static content.



2. **What is Catalina?**

Catalina in Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and JavaServer Pages (JSP)

A servlet is a server-side component written in Java programming language that receives requests from clients and sends responses back.

3. **Describe tomcat directory structure.**

/bin : Contains Start/Stop scripts for tomcat for both windows and linux operating system. Also consists of JAR files with classes required to control the tomcat server.

/conf : Contains main configuration files (web.xml and server.xml).

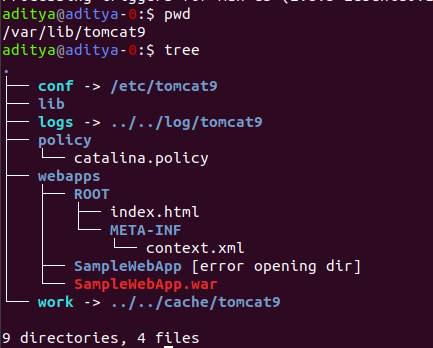
/lib : Contains the Tomcat Java Archive (jar) files, shared across all Tomcat components.

/logs : Contains tomcat log files.

/temp: Temporary file storage.

/webapps : Deployment staging directory, where you place your WAR file.

/work : Tomcat`s working directory where Tomcat places all servlets that are generated from JSPs.



4.**Connect any sample.war to MySQL running on localhost.**

##### **1. MySQL configuration**

Ensure that you follow these instructions as variations can cause problems.

Create a new test user, a new database and a single test table. Your MySQL user must have a password assigned. The driver will fail if you try to connect with an empty password.

**mysql> GRANT ALL PRIVILEGES ON \*.\* TO javauser@localhost**

-> IDENTIFIED BY 'javadude' WITH GRANT OPTION;

mysql> create database javatest;

mysql> use javatest;

mysql> create table testdata (

-> id int not null auto\_increment primary key,

-> foo varchar(25),

-> bar int);

**mysql> insert into testdata values(null, 'hello', 12345);**

Query OK, 1 row affected (0.00 sec)

mysql> select \* from testdata;

+----+-------+-------+

| ID | FOO | BAR |

+----+-------+-------+

| 1 | hello | 12345 |

+----+-------+-------+

1 row in set (0.00 sec)

mysql>

##### **2. Context configuration**

Configure the JNDI DataSource in Tomcat by adding a declaration for your resource to your Context.

**<Context>**

<!-- maxActive: Maximum number of database connections in pool. Make sure you

configure your mysqld max\_connections large enough to handle

all of your db connections. Set to -1 for no limit.

-->

<!-- maxIdle: Maximum number of idle database connections to retain in pool.

Set to -1 for no limit. See also the DBCP 1.x documentation on this

and the minEvictableIdleTimeMillis configuration parameter.

-->

<!-- maxWait: Maximum time to wait for a database connection to become available

in ms, in this example 10 seconds. An Exception is thrown if

this timeout is exceeded. Set to -1 to wait indefinitely.

-->

<!-- username and password: MySQL username and password for database connections -->

<!-- driverClassName: Class name for the old mm.mysql JDBC driver is

org.gjt.mm.mysql.Driver - we recommend using Connector/J though.

Class name for the official MySQL Connector/J driver is com.mysql.jdbc.Driver.

-->

<!-- url: The JDBC connection url for connecting to your MySQL database.

-->

<Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"

maxActive="100" maxIdle="30" maxWait="10000"

username="javauser" password="javadude" driverClassName="com.mysql.jdbc.Driver"

url="jdbc:mysql://localhost:3306/javatest"/>

</Context>

##### **3. web.xml configuration**

Now create a WEB-INF/web.xml for this test application.

**<web-app xmlns="http://java.sun.com/xml/ns/j2ee"**

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee

http://java.sun.com/xml/ns/j2ee/web-app\_2\_4.xsd"

version="2.4">

<description>MySQL Test App</description>

<resource-ref>

<description>DB Connection</description>

<res-ref-name>jdbc/TestDB</res-ref-name>

<res-type>javax.sql.DataSource</res-type>

<res-auth>Container</res-auth>

</resource-ref>

</web-app>

##### **4. Test code**

Now create a simple test.jsp page for use later.

<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<sql:query var="rs" dataSource="jdbc/TestDB">

select id, foo, bar from testdata

</sql:query>

<html>

<head>

<title>DB Test</title>

</head>

<body>

<h2>Results</h2>

<c:forEach var="row" items="${rs.rows}">

Foo ${row.foo}<br/>

Bar ${row.bar}<br/>

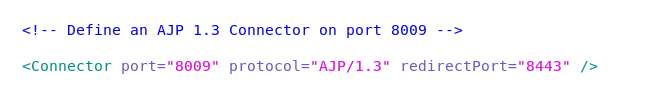
</c:forEach>

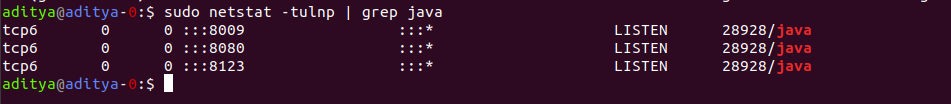
</body>

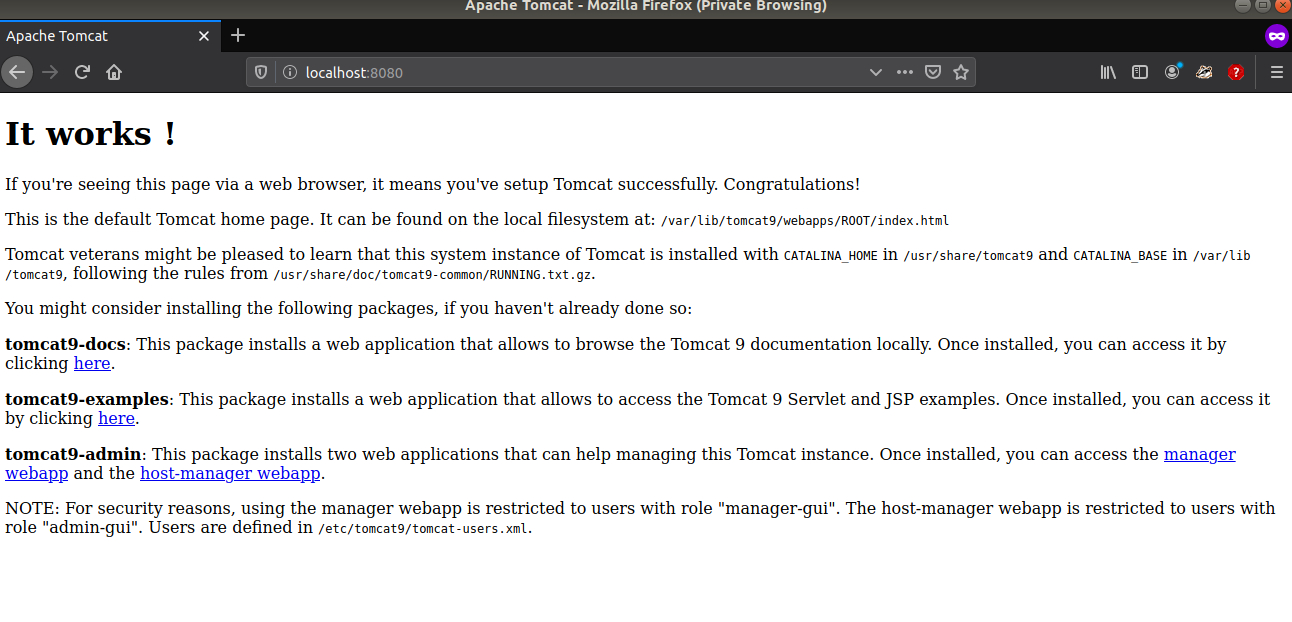
</html>

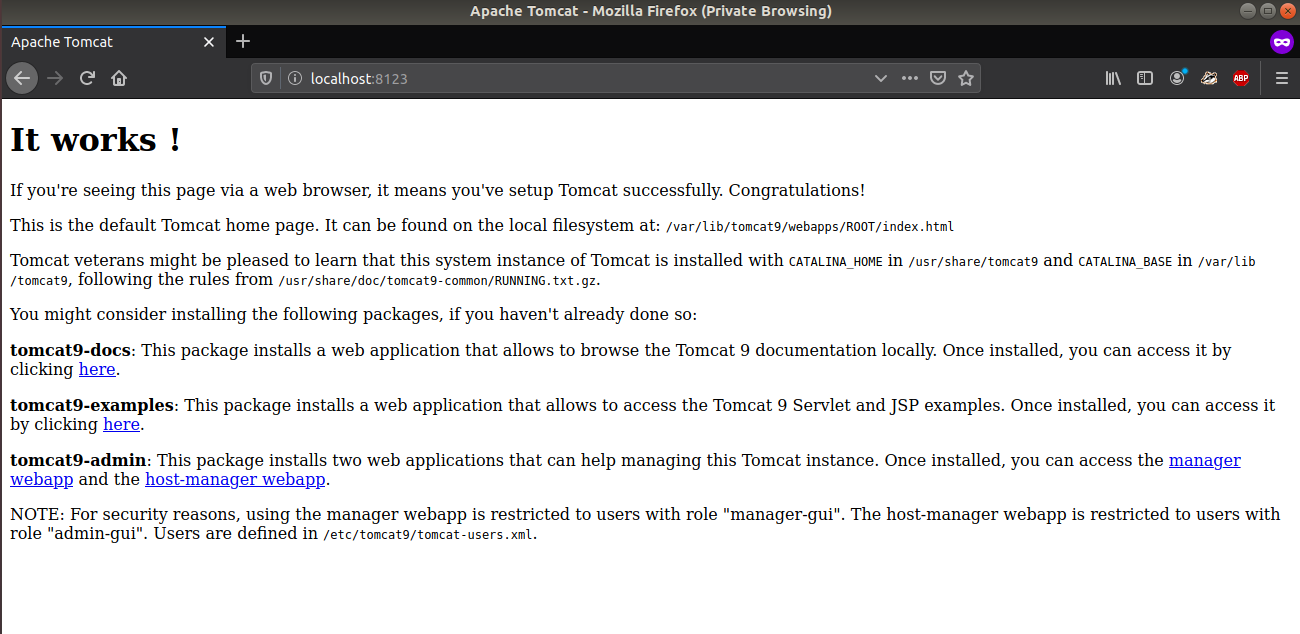
5.**Run multiple services on different ports with different connectors (AJP/HTTP) on same tomcat installation.**



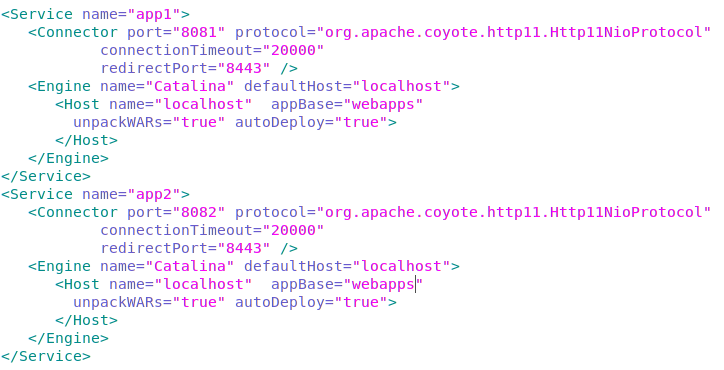








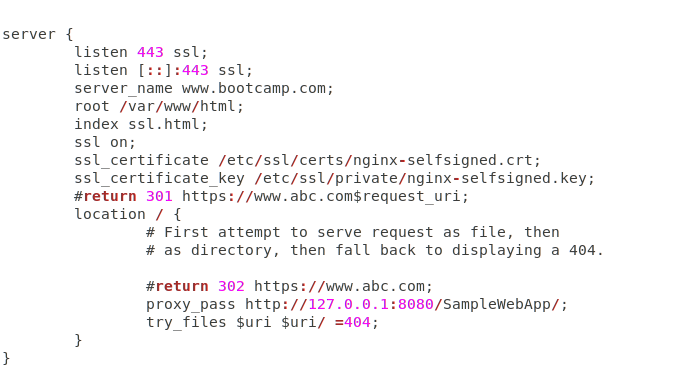
Method 2:

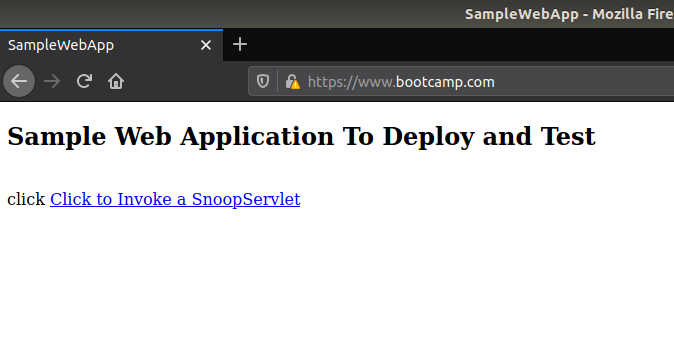


**Use nginx as reverse proxy for tomcat application.**



**Setup self signed certificate on that nginx for bootcamp.com.**





**What is the difference between proxy\_pass & proxy\_pass reverse?**

In forward proxy client identity is hidden when they request(ex. VPN)

In reverse proxy server identity is hidden, (ex. Client will hit nginx and it will serve content from tomcat)