**Application server (tomcat)**

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

A Web server exclusively handles HTTP requests, whereas an application server serves business logic to application programs through any number of protocols.In most cases, the server exposes this business logic through a component API, such as the EJB (Enterprise JavaBean) component model found on J2EE (Java 2 Platform, Enterprise Edition) application servers. Moreover, the application server manages its own resources. Such gate-keeping duties include security, transaction processing, resource pooling, and messaging.

- **What is Catalina?**

  Catalina is Tomcat's servlet container. Catalina implements Sun Microsystems' specifications for servlet and JavaServer Pages (JSP). In Tomcat, a Realm element represents a "database" of usernames, passwords, and roles (similar to Unix groups) assigned to those users. Different implementations of Realm allow Catalina to be integrated into environments where such authentication information is already being created and maintained, and then use that information to implement Container Managed Security as described in the Servlet Specification.

- **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 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.

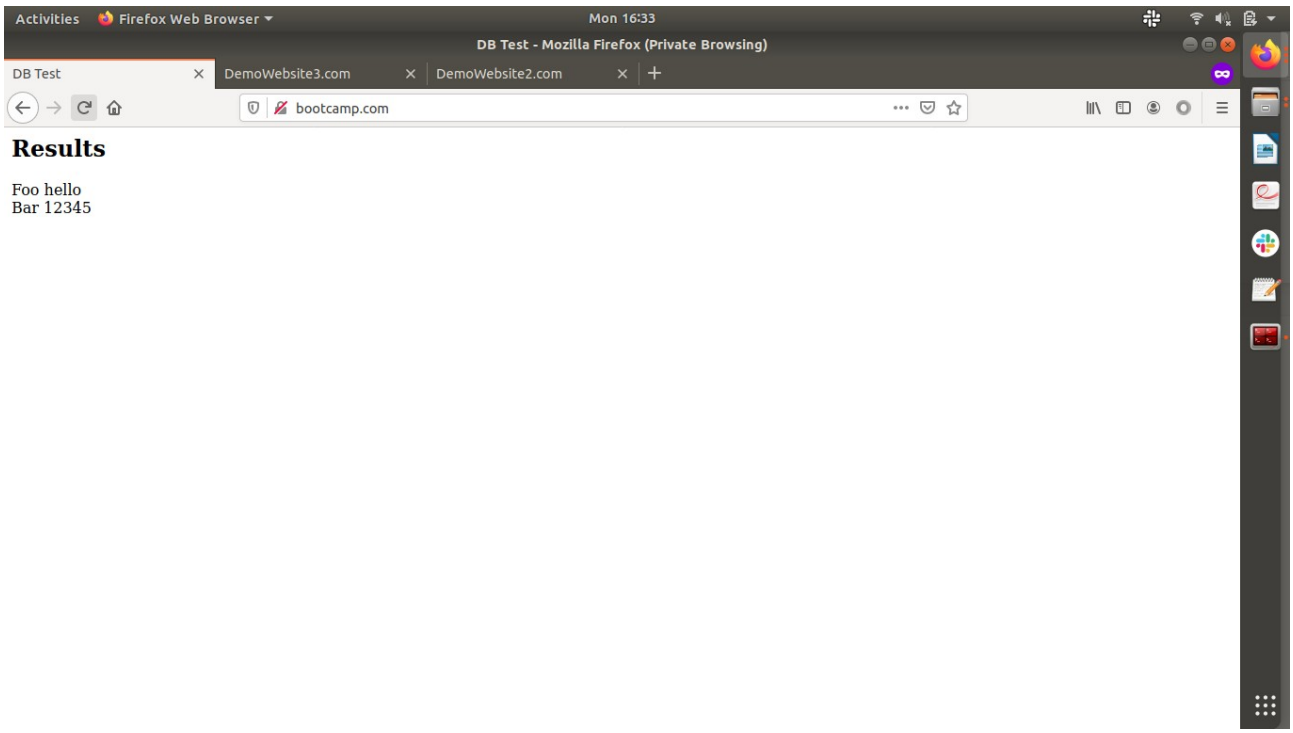
/logs : Contains tomcat log files.

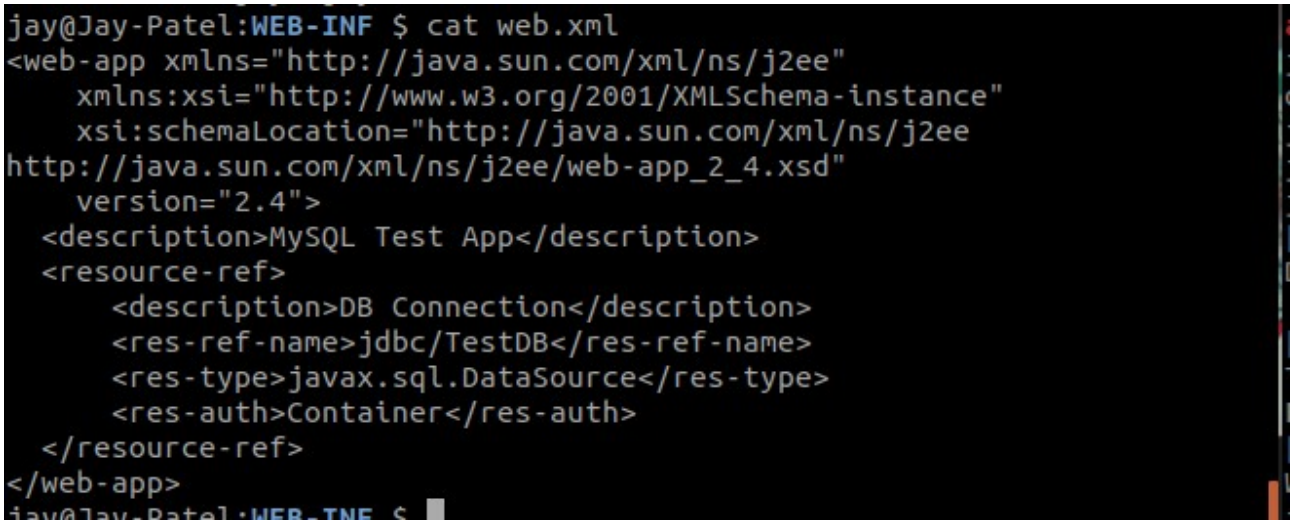
/temp: Temporary file storage.

/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.

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

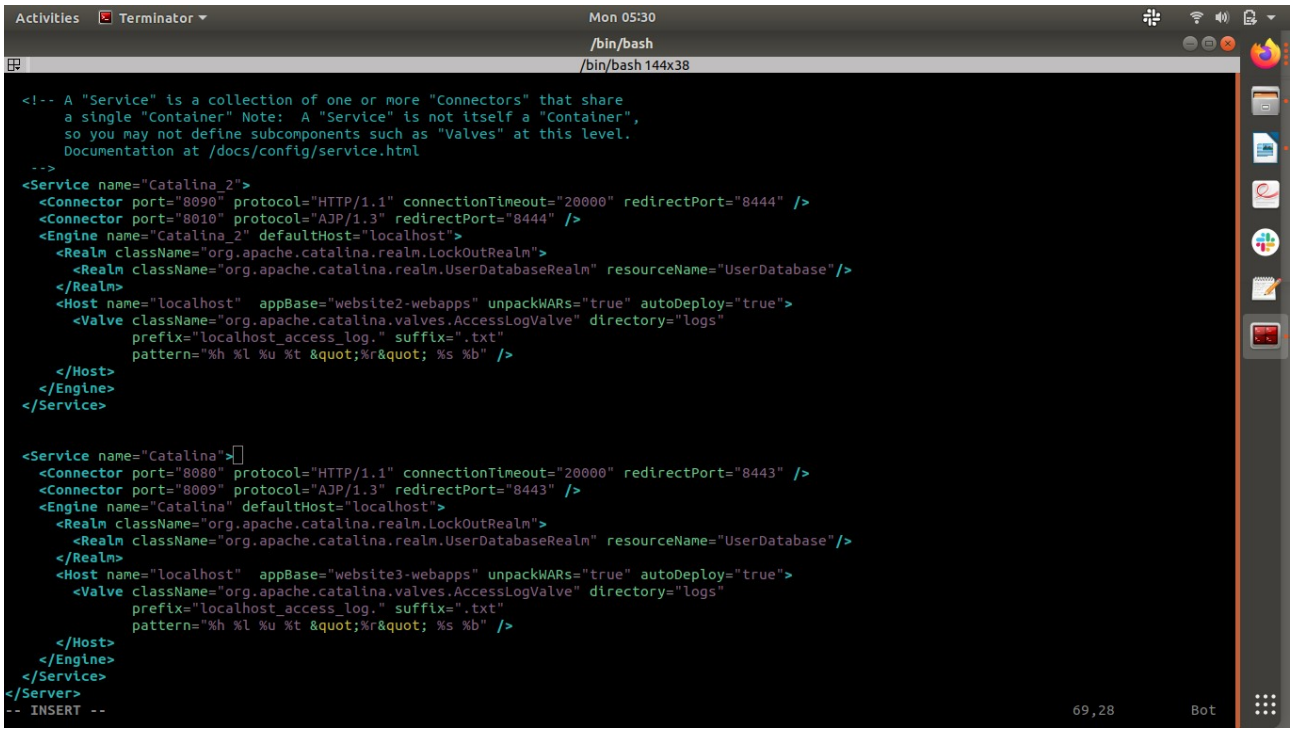

Activities
Firefox Web Browser
Mon 16:33
DB Test - Mozilla Firefox (Private Browsing)
DB Test
DemoWebsite3.com
DemoWebsite2.com
bootcamp.com
Results
Foo hello
Bar 12345

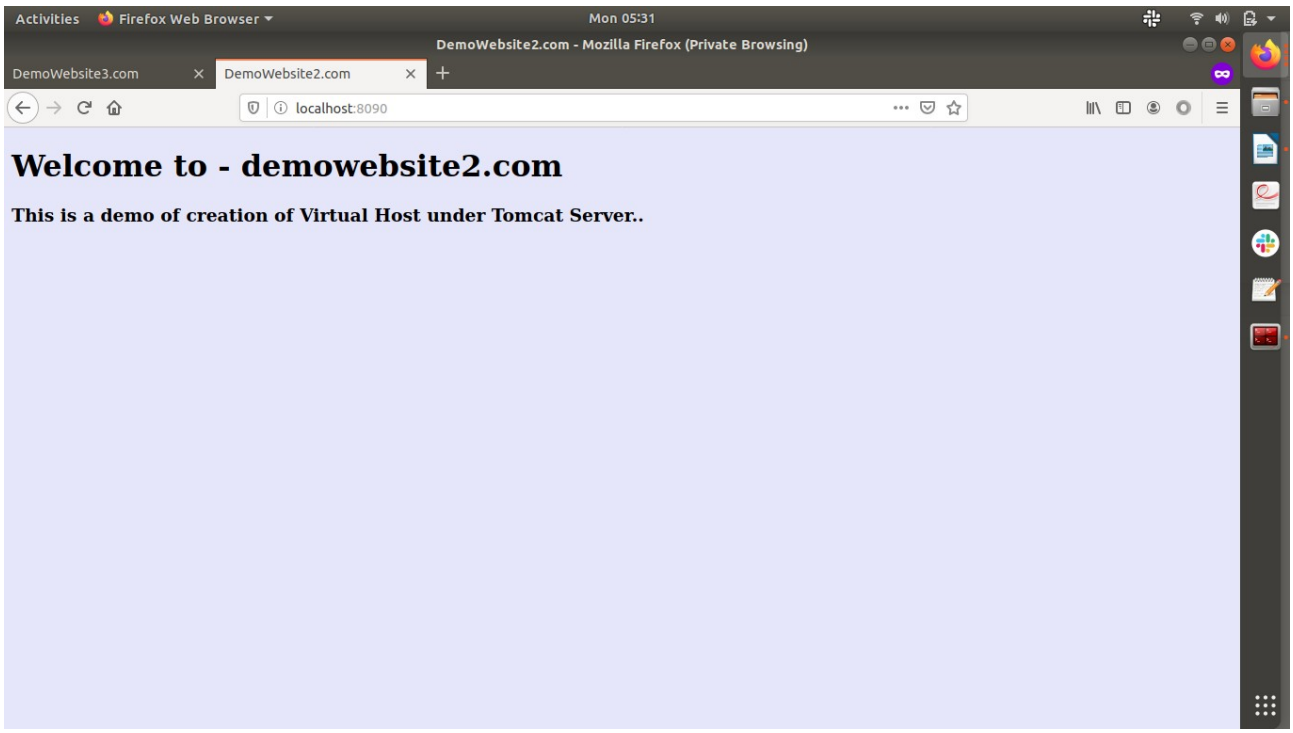

jay@Jay-Patel:WEB-INF $ cat web.xml
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
version="2.4">
<description>MySQL Test App</description>
<resource-ref>
<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>
jay@Jay-Patel:WEB-INF $

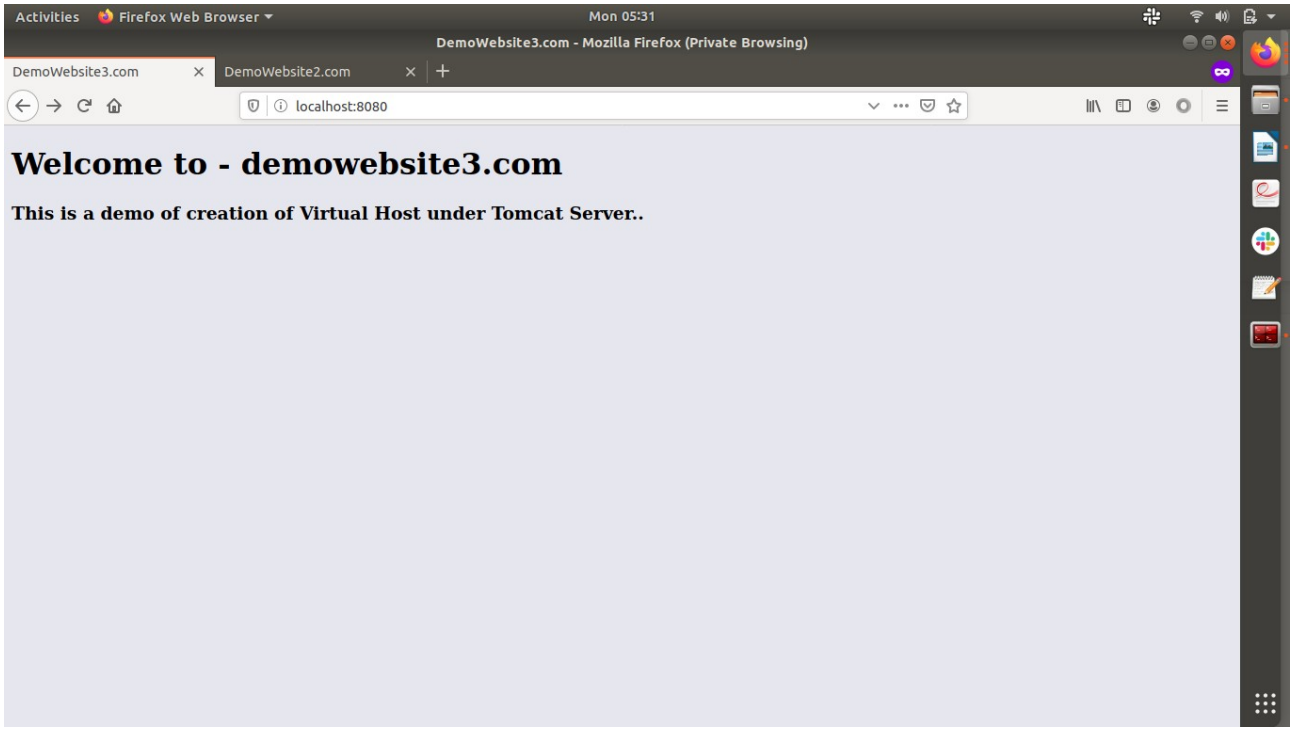
```
jay@Jay-Patel:ROOT $ cat index.jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<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>
jay@Jay-Patel:ROOT $
```

```
 <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSour
ce"
               maxActive="100" maxIdle="30" maxWait="10000"
               username="javauser" password="javadude" driverClassName=
"com.mysql.jdbc.Driver"
               url="jdbc:mysql://localhost:3306/javatest"/>
</Context>
```

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





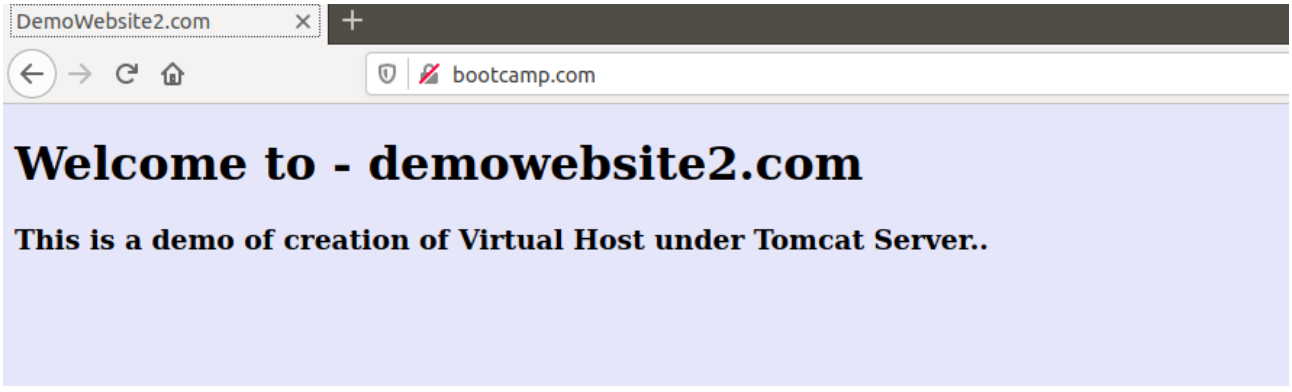



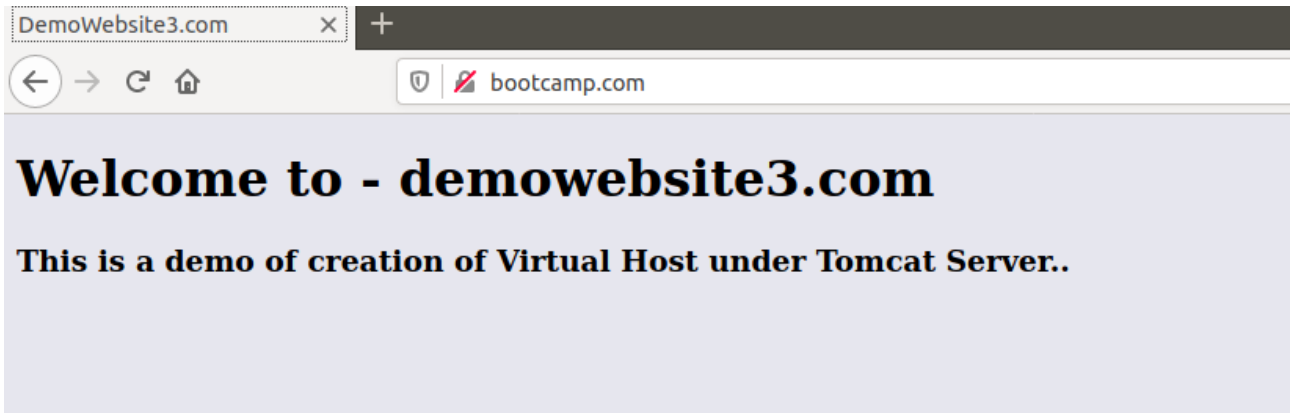

```
jay@Jay-Patel:~ $ ./ajping localhost:8010
Reply from localhost: 7 bytes in 0.002 seconds
Reply from localhost: 7 bytes in 0.002 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.003 seconds
^C
```

```
jay@Jay-Patel:~ $ ./ajping localhost:8009
Reply from localhost: 7 bytes in 0.002 seconds
Reply from localhost: 7 bytes in 0.002 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.004 seconds
Reply from localhost: 7 bytes in 0.002 seconds
Reply from localhost: 7 bytes in 0.004 seconds
```

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

```
upstream jay {
   # hash $request_uri;
    server 127.0.0.1:8080;
    server 127.0.0.1:8090;
}

server {
        listen 127.1.1.4:80;
        server_name bootcamp.com;

        location / {
                proxy_pass http://jay;

        }
}
~
```






- **Setup self signed certificate on that nginx for bootcamp.com.**
- **What is the difference between proxy_pass & proxy_pass reverse?**

If the server actually handling a request does a redirect to a different URL on that server, the ProxyPassReverse directive rewrites the URL in terms of the reverse proxy server.

ProxyPass is the main proxy configuration directive. In this case, it specifies that everything under the root URL ( / ) should be mapped to the backend server at the given address