Jenkins

Continuous Delivery

> Continuous Delivery (CD) is a DevOps practice that is used to deploy an application quickly while maintaining a high quality with an automated approach. It is about the way application package is deployed in the Web Server or in the Application Server in environment such as dev, test or staging. Deployment of an application can be done using shell script, batch file, or plugins available in Jenkins. Approach of automated deployment in case of Continuous Delivery and Continuous Deployment will be always same most of the time. In the case of Continuous Delivery, the application package is always production ready

Jenkins with Maven Build

Setup maven

- Go to `Jenkins Dashboard` -> `Manage Jenkins` -> `Global Tool Configuration` > `Maven` > Give a Name `Maven_Local` > Check `Install Automatically` > Install from Apache (specify a version) > `Save`
- You can give a logical name to identify the correct version while configuring a build job

Maven build phases

- Maven itself requires Java installed on your machine.
- You can verify if Maven is installed on your machine by running **mvn -v** in your command line/terminal.
- Maven is based on the `Project Object Model (POM)` configuration, which is stored in the XML file called the same – `pom.xml`.

- It is a structured format that describes the project, it's dependencies, plugins, and goals.
- `pom.xml` file should be present in your project directory
- Below are the Maven Build Phases
- **Validate** : Validate Project is correct & all necessary information is available.
- **Compile** : Compile the Source Code
- **Test** : Test the Compiled Source Code using suitable unit Testing Framework (like JUnit)
- **package** : Take the compiled code and package it.
- **Install** : Install package in Local Repo, for use as a dependency in other project locally.
- **Deploy** : Copy the final package to the remote repository for sharing with other developers.
- The above are always are sequential, if you specify `install`, all the phases before `install` are checked.

pom.xml definitions

- `<modelVersion>` : POM model version (always 4.0.0).
- `<groupId>` : Group or organization that the project belongs to. Often expressed as an inverted domain name.
- `<artifactId>` : Name to be given to the project's library artifact (for example, the name of its JAR or WAR file).
- `<version>` : Version of the project that is being built.
- `<packaging>` : How the project should be packaged. Defaults to `jar` for JAR file packaging. Use `war` for WAR file packaging.

Jenkins Job

- Click on **New Item** then enter an item name, select **Freestyle project**.
- Under **Source Code Management** tab, select Git and then set the Repository URL to point to your GitHub Repository. `https://github.com/YourUserName/repo-name.git`
- Under Build Environment Build Step > Select `Invoke top-level Maven targets` from dropdown > select the Maven Version that we just created, specify `clean install`.
- Under Advanced tab, specifiy the `pom.xml` file relative path location from git repository.
- Click on `Save`

> it will run command `mvn clean install -f pom.xml`

- Click OK and Build a Job and you will see that a `war file` is created.

> `clean` -> Deletes `/var/lib/jenkins/workspace/jenkins-maven-build/java-tomcat-sample/target`

Artifacts Archive

- Go to `Jenkins dashboard` -> `Jenkins project or build job` -> `Post-build Actions` -> `Add post-build action` -> `Archive the artifacts`:
- Enter details for options in `Archive the artifacts` section:
    - For `Files to archive` enter the Path of the `.war` file like : `java-tomcat-sample/target/*.war`
- `Save` the changes and `Build Now`.
- Check the directories as below to validate above information:

```
ls /var/lib/jenkins/jobs
ls /var/lib/jenkins/jobs/<JOB_NAME>
ls /var/lib/jenkins/jobs/<JOB_NAME>/builds/<BUILD_NUMBER>
ls /var/lib/jenkins/workspace/<JOB_NAME>
```

- If you check the directory structure, there will be `archive` directory present under the subsequent build number for which the job is executed with Post build action as `Archive the artifacts`

Jenkins Build and Deploy

- Below steps assume that, you have a Jenkins Server Up and Running on one of the EC2 instance.

Setup Apache Tomcat on Amazon Linux:

- Launch a new EC2 Instance for Webserver Configuration

```
sudo hostnamectl set-hostname tomcat.example.com
sudo yum install java-1.8.0 -y
sudo yum install java-devel
cd /opt/
sudo wget https://archive.apache.org/dist/tomcat/tomcat-9/v9.0.35/bin/apache-tomcat-9.0.35.tar.gz
sudo tar -zvxf apache-tomcat-9.0.35.tar.gz
----------------------------
z - The file is a “gzipped” file
v - Verbose, print the file names as they are extracted one by one
x -  Extract files
f - Use the following tar archive for the operation
----------------------------
sudo ls -ltr /opt/apache-tomcat-9.0.35/bin
sudo cd /opt/apache-tomcat-9.0.35/bin
```

- To Start Apache Tomcat : Run the `./startup.sh` file in `/opt/apache-tomcat-9.0.35/bin`
- We can make the scripts executable and then create a symbolic link for this scripts.

```
```

```
sudo chmod +x /opt/apache-tomcat-9.0.35/bin/startup.sh
sudo chmod +x /opt/apache-tomcat-9.0.35/bin/shutdown.sh
```

- Create symbolic link to these file so that tomcat server start and stop can be executed from any directory.

```
echo $PATH
sudo ln -s /opt/apache-tomcat-9.0.35/bin/startup.sh /usr/bin/tomcatup
sudo ln -s /opt/apache-tomcat-9.0.35/bin/shutdown.sh /usr/bin/tomcatdown
tomcatup
netstat -nltp | grep 8080
```

> If you want to run Apache Tomcat on same Machine where Jenkins is Installed, then change the port of Apache Tomcat in : `/opt/apache-tomcat-9.0.35/conf/server.xml` file to `8090` as below,

```
<Connector port="8090" protocol="HTTP/1.1"
               connectionTimeout="20000"
               redirectPort="8443" />
```

- If above changes are made, execute the command `tomcatdown` and `tomcatup`.
- Create an empty repo and clone it, add project files into the local git folder and commit -> push the local repo to remote github repo using Git Bash.
- Verify the files are available in your github repository

Tomcat War file deployment Configs

- To have access to the dashboard the admin user needs the manager-gui role. Later, we will need to deploy a WAR file using Maven, for this, we need the `manager-script` role too.
- In order for Tomcat to accept remote deployments, we have to add a user with the role `manager-script`. To do so, edit the file `../conf/tomcat-users.xml` and add the following lines:
- In this case : add below configuration in file `/opt/apache-tomcat-9.0.35/conf/tomcat-users.xml`

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="admin" password="admin" roles="manager-gui, manager-script"/>
<user username="deployer" password="deployer" roles="manager-script" />
```

- Edit the RemoteAddrValve under this file `/opt/apache-tomcat-9.0.35/webapps/manager/META-INF/context.xml` to allow all.
- `Before`

```
```

```
        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow="127\.\d+\.\d+\.\d+|::1|0:0:0:0:0:0:0:1" />
```

- `After`

```
        <Valve className="org.apache.catalina.valves.RemoteAddrValve"
         allow=".*" />
```

- Restart the tomcat server using `tomcatdown` and `tomcatup`

Apache Tomcat Terminology

- **Document root** : This is the top-level directory of a web application where all the application resources are located, like JSP files, HTML pages, Java classes, and images.
- **Context path** : This refers to the location that's relative to the server's address and represents the name of the web application. For e.g, If the WAR File is kept under the `$CATALINA_HOME\webapps\myapp` directory, it'll be accessed by the URL http://TOMCAT_IP:PORT/myapp, and its context path will be `/myapp`.
- **WAR** – Web Archive. It's the extension of a file that packages a web application directory hierarchy in ZIP format.

Jenkins Plugin installation

- To install the Plugin `Deploy to container` navigate to `Manage Jenkins` > `Manage Plugins`, search `Deploy to container` under `Available` tab.

Jenkins Job to deploy war file

- Click on **New Item** then enter an item name, select **Freestyle project**.
- Select the GitHub project checkbox and set the Project URL to point to your GitHub Repository. `https://github.com/YourUserName/`
- Under Source Code Management Section : Provide the Github Repository URL of the Maven Project, keep the branch as `master`.
- Go to Jenkins Project -> Configure -> Under Build Environment Build Step > Select `Invoke top-level Maven targets` from dropdown > select the `Maven Version` that is configured > Enter `clean install`
- Under `Post-build Actions`, from the `Add post-build action` dropdown button select the option `Deploy war/ear to a container`
- Enter details of the War file that will be created as:
    - For `WAR/EAR files` you can use wild cards, e.g. `**/*.war`.

- The `context path` is the context path part of the URL under which your application will be published in Tomcat.
- Select the appropriate Tomcat version from the Container dropdown box (note that you can also deploy to Glassfish or JBoss using this Jenkins plugin).
- Under the `Credentials`, `Add` username and password value that is entered in the `tomcat-users.xml` file. Specify the ID of the credentials as `tomcat_creds`. This will be used later in Pipeline Script.
- The Tomcat URL is the base URL through which your Tomcat instance can be reached (e.g `http://172.31.67.85:8080`)

> Make Sure network is open on specific port by checking the Security Group attached to EC2 Instance.

- `Save` the changes and `Build Now`.
- Once Jenkins Job is build, if there is a Success for deploy, verify the deployment files on Tomcat Server under `/opt/apache-tomcat-9.0.35/webapps/` path.
- Access the Application in Browser with specific Home Page present in `src/webapp` i.e `<TOMCAT_SERVER_IP>:<TOMCAT_PORT>/<CONTEXT_PATH>/index.jsp`
- Make some changes in the code on the github configured branch in the Jenkins Job and build the Job again to verify the Artifact Deployment on Tomcat Path.

Java Database Connection

- The Java Project Object Model file i.e `pom.xml` file contains `dependency` specified as below:

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.24</version>
</dependency>
```

- During the Java Build, the above mentioned dependency packages i.e jar files are downloaded from Maven Website Portal and added inside the `lib` path inside the Artifact `.war` file.
- The `mysql-connector-java-8.0.24.jar` file can be found under `/opt/apache-tomcat-9.0.35/webapps/java-tomcat-sample-deploy/WEB-INF/lib/mysql-connector-java-8.0.24.jar`
- Jar files contain .class files already compiled and can be used as import packages in .java programs.
- To view the content of the Jar files use command as : `jar tf FILENAME.jar`
  - `jar tf /opt/apache-tomcat-9.0.35/webapps/java-tomcat-sample-deploy/WEB-INF/lib/mysql-connector-java-8.0.24.jar`
- This will display all the .class files present.
- The `mysql-connector-java-8.0.24.jar` file is used in the `java-tomcat-sample/src/main/java/dao/GetDao.java` file to import methods and use objects to make connection to mysql database.

Installing Mysql Server on EC2

- Navigate to Apache Tomcat Server and execute below commands.

```
sudo wget https://dev.mysql.com/get/mysql57-community-release-el7-11.noarch.rpm
sudo yum localinstall mysql57-community-release-el7-11.noarch.rpm
sudo yum install mysql-community-server

######
================================================================================
================================================================================
================================================================================
=======
 Package                                                          Arch
Version                                                       Repository
Size
================================================================================
================================================================================
================================================================================
=======
Installing:
 mysql-community-libs                                             x86_64
5.7.36-1.el7                                                  mysql57-
community                                          2.4 M
     replacing  mariadb-libs.x86_64 1:5.5.68-1.amzn2
 mysql-community-libs-compat                                      x86_64
5.7.36-1.el7                                                  mysql57-
community                                          1.2 M
     replacing  mariadb-libs.x86_64 1:5.5.68-1.amzn2
 mysql-community-server                                           x86_64
5.7.36-1.el7                                                  mysql57-
community                                          174 M
Installing for dependencies:
 mysql-community-client                                           x86_64
5.7.36-1.el7                                                  mysql57-
community                                           25 M
 mysql-community-common                                           x86_64
5.7.36-1.el7                                                  mysql57-
community                                          310 k
 ncurses-compat-libs                                              x86_64
6.0-8.20170212.amzn2.1.3                                      amzn2-core
308 k

######

sudo systemctl start mysqld.service
sudo systemctl status mysqld.service
netstat -nltp

# get the default database login password
sudo grep 'temporary password' /var/log/mysqld.log
2022-01-12T19:52:31.672795Z 1 [Note] A temporary password is generated for
root@localhost: d/YSB<iD0ud2

# Change the DB password
```

```
mysql_secure_installation

##############################
Securing the MySQL server deployment.

Enter password for user root:

The existing password for the user account root has expired. Please set a new
password.

New password:

Re-enter new password:
The 'validate_password' plugin is installed on the server.
The subsequent steps will run with the existing configuration
of the plugin.
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : y

New password:

Re-enter new password:

Estimated strength of the password: 100
Do you wish to continue with the password provided?(Press y|Y for Yes, any other
key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : n

 ... skipping.
By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) :
y
 - Dropping test database...
```

```
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
##############################

# Use below command to login to mysql shell
mysql -u root -p

mysql> show databases;
mysql> create database db;
mysql> use db;

# Create employees table
CREATE TABLE employees (
    emp_no      INT             NOT NULL,
    birth_date  DATE            NOT NULL,
    first_name  VARCHAR(14)     NOT NULL,
    last_name   VARCHAR(16)     NOT NULL,
    gender      ENUM ('M','F')  NOT NULL,
    hire_date   DATE            NOT NULL,
    PRIMARY KEY (emp_no)
);

show tables;
desc employees;

INSERT INTO `employees` VALUES (10001,'1953-09-02','Georgi','Facello','M','1986-
06-26'),
(10002,'1964-06-02','Bezalel','Simmel','F','1985-11-21'),
(10003,'1959-12-03','Parto','Bamford','M','1986-08-28'),
(10004,'1954-05-01','Chirstian','Koblick','M','1986-12-01'),
(10005,'1955-01-21','Kyoichi','Maliniak','M','1989-09-12'),
(10006,'1953-04-20','Anneke','Preusig','F','1989-06-02'),
(10007,'1957-05-23','Tzvetan','Zielinski','F','1989-02-10'),
(10008,'1958-02-19','Saniya','Kalloufi','M','1994-09-15'),
(10009,'1952-04-19','Sumant','Peac','F','1985-02-18'),
(10010,'1963-06-01','Duangkaew','Piveteau','F','1989-08-24'),
(10011,'1953-11-07','Mary','Sluis','F','1990-01-22'),
(10012,'1960-10-04','Patricio','Bridgland','M','1992-12-18'),
(10013,'1963-06-07','Eberhardt','Terkki','M','1985-10-20'),
(10014,'1956-02-12','Berni','Genin','M','1987-03-11'),
(10015,'1959-08-19','Guoxiang','Nooteboom','M','1987-07-02'),
(10016,'1961-05-02','Kazuhito','Cappelletti','M','1995-01-27'),
(10017,'1958-07-06','Cristinel','Bouloucos','F','1993-08-03'),
(10018,'1954-06-19','Kazuhide','Peha','F','1987-04-03'),
```

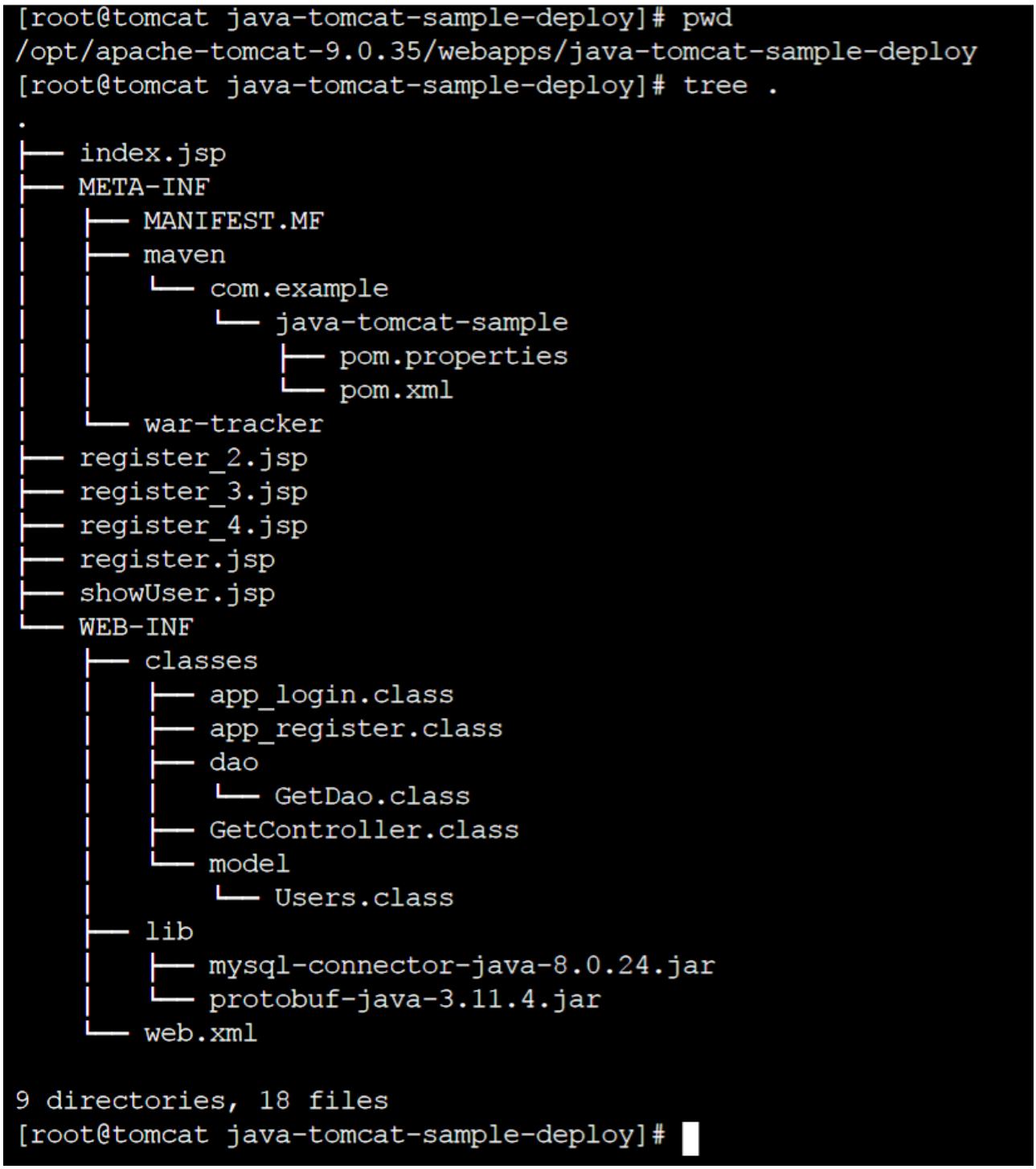
```
(10019,'1953-01-23','Lillian','Haddadi','M','1999-04-30'),
(10020,'1952-12-24','Mayuko','Warwick','M','1991-01-26');

# Create Title Table
CREATE TABLE titles (
    emp_no      INT             NOT NULL,
    title       VARCHAR(50)     NOT NULL,
    from_date   DATE            NOT NULL,
    to_date     DATE,
    # FOREIGN KEY (emp_no) REFERENCES employees (emp_no) ON DELETE CASCADE,
    PRIMARY KEY (emp_no,title, from_date)
);

INSERT INTO `titles` VALUES (10001,'Senior Engineer','1986-06-26','9999-01-01'),
(10002,'Staff','1996-08-03','9999-01-01'),
(10003,'Senior Engineer','1995-12-03','9999-01-01'),
(10004,'Engineer','1986-12-01','1995-12-01'),
(10004,'Senior Engineer','1995-12-01','9999-01-01'),
(10005,'Senior Staff','1996-09-12','9999-01-01'),
(10005,'Staff','1989-09-12','1996-09-12'),
(10006,'Senior Engineer','1990-08-05','9999-01-01'),
(10007,'Senior Staff','1996-02-11','9999-01-01'),
(10007,'Staff','1989-02-10','1996-02-11'),
(10008,'Assistant Engineer','1998-03-11','2000-07-31'),
(10009,'Assistant Engineer','1985-02-18','1990-02-18'),
(10009,'Engineer','1990-02-18','1995-02-18'),
(10009,'Senior Engineer','1995-02-18','9999-01-01'),
(10010,'Engineer','1996-11-24','9999-01-01'),
(10011,'Staff','1990-01-22','1996-11-09'),
(10012,'Engineer','1992-12-18','2000-12-18'),
(10012,'Senior Engineer','2000-12-18','9999-01-01'),
(10013,'Senior Staff','1985-10-20','9999-01-01'),
(10014,'Engineer','1993-12-29','9999-01-01'),
(10015,'Senior Staff','1992-09-19','1993-08-22'),
(10016,'Staff','1998-02-11','9999-01-01'),
(10017,'Senior Staff','2000-08-03','9999-01-01'),
(10017,'Staff','1993-08-03','2000-08-03'),
(10018,'Engineer','1987-04-03','1995-04-03'),
(10018,'Senior Engineer','1995-04-03','9999-01-01'),
(10019,'Staff','1999-04-30','9999-01-01'),
(10020,'Engineer','1997-12-30','9999-01-01');

SELECT employees.emp_no,employees.first_name, employees.last_name,
employees.hire_date, titles.title, titles.from_date, titles.to_date FROM employees
left JOIN titles ON employees.emp_no=titles.emp_no;
```

- Once above DB and Tables are created, validate the details of the `DB Hostname, Database, DB UserName, DB Password values` in file `java-tomcat-sample/src/main/java/dao/GetDao.java`
- Modify the above DB Password that is set in the above file and build and deploy artifact WAR File again.

- Execute the Jenkins Job to build above artifact and deploy the same on the Webserver Context Path.
- A directory with name of the `.war` file is present on the `webapps` path.
- 

- To view the content of `.war` file use : `jar tf java-tomcat-sample-deploy.war`

```
[root@tomcat webapps]# pwd
/opt/apache-tomcat-9.0.35/webapps
[root@tomcat webapps]# jar tf java-tomcat-sample-deploy.war
META-INF/
META-INF/MANIFEST.MF
WEB-INF/
WEB-INF/classes/
WEB-INF/classes/dao/
WEB-INF/classes/model/
WEB-INF/lib/
WEB-INF/web.xml
WEB-INF/classes/app_register.class
WEB-INF/classes/app_login.class
WEB-INF/classes/dao/GetDao.class
WEB-INF/classes/model/Users.class
WEB-INF/classes/GetController.class
WEB-INF/lib/mysql-connector-java-8.0.24.jar
WEB-INF/lib/protobuf-java-3.11.4.jar
register.jsp
register_2.jsp
register_3.jsp
register_4.jsp
index.jsp
showUser.jsp
META-INF/maven/
META-INF/maven/com.example/
META-INF/maven/com.example/java-tomcat-sample/
META-INF/maven/com.example/java-tomcat-sample/pom.xml
META-INF/maven/com.example/java-tomcat-sample/pom.properties
[root@tomcat webapps]#
```

- Also, validate the details retured in the WebPage from Database.