

Name : Chandana Chowdary Potu

Project : CI/CD Deployment for Springboot Application

Project Objective : As a Full Stack Developer, you have to build a CI/CD pipeline to demonstrate continuous deployment and host the application on AWS EC2 instance.

SOURCE CODE

Src/main/java/com/SpringTest/SpringApplication.java:

```
package com.SpringTest;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringJenkinsApplication {

    public static Logger log =
LoggerFactory.getLogger(SpringJenkinsApplication.class);

    public void init() {
        log.info("Spring Boot Application Started.....");
    }

    public static void main(String[] args) {

        log.info("Application Executed .....");
        SpringApplication.run(SpringJenkinsApplication.class, args);
    }
}
```

Src/test/java/com/SpringTest/SpringApplicationTest.java:

```

package com.SpringTest;

import org.junit.jupiter.api.Test;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class SpringJenkinsApplicationTests {

    public static Logger log =
        LoggerFactory.getLogger(SpringJenkinsApplication.class);

    @Test
    void contextLoads() {

        log.info("Spring Test Case Executing.....");

    }

}

```

META-INF/maven/com.SpringTest/Testing-Spring-Jenkins/pom.properties:

```

#Generated by Maven Integration for Eclipse
#Tue May 10 13:03:45 IST 2022
m2e.projectLocation=C:\\Users\\bh\\Desktop\\phase 5 project\\CI-CD-Deployment-for-  

Springboot-Application
m2e.projectName=Spring-Jenkins
groupId=com.SpringTest
artifactId=Testing-Spring-Jenkins
version=0.0.1-SNAPSHOT

```

META-INF/maven/com.SpringTest/Testing-Spring-Jenkins/pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.SpringTest</groupId>
  <artifactId>Testing-Spring-Jenkins</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Spring-Jenkins</name>
  <description>Spring Boot -Jenkins</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

```

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>
    </project>

```

MANIFEST.MF:

Manifest-Version: 1.0
 Build-Jdk-Spec: 13
 Implementation-Title: Spring-Jenkins
 Implementation-Version: 0.0.1-SNAPSHOT
 Created-By: Maven Integration for Eclipse

Maven-archiver/pom.properties:

artifactId=Testing-Spring-Jenkins
 groupId=com.SpringTest
 version=0.0.1-SNAPSHOT

mvnw:

```

#!/bin/sh
# -----
# Licensed to the Apache Software Foundation (ASF) under one
# or more contributor license agreements. See the NOTICE file
# distributed with this work for additional information
# regarding copyright ownership. The ASF licenses this file
# to you under the Apache License, Version 2.0 (the
# "License"); you may not use this file except in compliance
# with the License. You may obtain a copy of the License at
#
# https://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing,
# software distributed under the License is distributed on an
# "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
# KIND, either express or implied. See the License for the
# specific language governing permissions and limitations
# under the License.
# -----

```

```

# -----
# Maven Start Up Batch script
#
# Required ENV vars:
# -----
# JAVA_HOME - location of a JDK home dir
#
# Optional ENV vars
# -----
# M2_HOME - location of maven2's installed home dir
# MAVEN_OPTS - parameters passed to the Java VM when running Maven
#   e.g. to debug Maven itself, use
#   set MAVEN_OPTS=-Xdebug -
#   Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=8000
# MAVEN_SKIP_RC - flag to disable loading of mavenrc files
# -----

if [ -z "$MAVEN_SKIP_RC" ] ; then

    if [ -f /etc/mavenrc ] ; then
        . /etc/mavenrc
    fi

    if [ -f "$HOME/.mavenrc" ] ; then
        . "$HOME/.mavenrc"
    fi

fi

# OS specific support. $var _must_ be set to either true or false.
cygwin=false;
darwin=false;
mingw=false
case "`uname`" in
    CYGWIN*) cygwin=true ;;
    MINGW*) mingw=true;;
    Darwin*) darwin=true
        # Use /usr/libexec/java_home if available, otherwise fall back to /Library/Java/Home
        # See https://developer.apple.com/library/mac/qa/qa1170/\_index.html
        if [ -z "$JAVA_HOME" ] ; then
            if [ -x "/usr/libexec/java_home" ] ; then
                export JAVA_HOME="/usr/libexec/java_home"
            else
                export JAVA_HOME="/Library/Java/Home"
            fi
        fi
    ;;
esac

```

```

if [ -z "$JAVA_HOME" ] ; then
  if [ -r /etc/gentoo-release ] ; then
    JAVA_HOME=`java-config --jre-home`
  fi
fi

if [ -z "$M2_HOME" ] ; then
  ## resolve links - $0 may be a link to maven's home
  PRG="$0"

  # need this for relative symlinks
  while [ -h "$PRG" ] ; do
    ls=`ls -ld "$PRG"`
    link=`expr "$ls" : '.*-> \(.*)$'`
    if expr "$link" : '/.*' > /dev/null; then
      PRG="$link"
    else
      PRG="`dirname "$PRG"`/$link"
    fi
  done

  saveddir=`pwd`

  M2_HOME=`dirname "$PRG"`/..

  # make it fully qualified
  M2_HOME=`cd "$M2_HOME" && pwd`

  cd "$saveddir"
  # echo Using m2 at $M2_HOME
fi

# For Cygwin, ensure paths are in UNIX format before anything is touched
if $cygwin ; then
  [ -n "$M2_HOME" ] &&
  M2_HOME=`cygpath --unix "$M2_HOME"`
  [ -n "$JAVA_HOME" ] &&
  JAVA_HOME=`cygpath --unix "$JAVA_HOME"`
  [ -n "$CLASSPATH" ] &&
  CLASSPATH=`cygpath --path --unix "$CLASSPATH"`
fi

# For Mingw, ensure paths are in UNIX format before anything is touched
if $mingw ; then
  [ -n "$M2_HOME" ] &&
  M2_HOME="`(cd "$M2_HOME"; pwd)`"
  [ -n "$JAVA_HOME" ] &&
  JAVA_HOME="`(cd "$JAVA_HOME"; pwd)`"
fi

```

```

if [ -z "$JAVA_HOME" ]; then
    javaExecutable=""
    which javac`
    if [ -n "$javaExecutable" ] && ! [ "`expr \"$javaExecutable\" : \"([^\"]*)\" = \"no\" ]"; then
        # readlink(1) is not available as standard on Solaris 10.
        readLink=`which readlink`
        if [ ! `expr "$readLink" : \"([^\"]*)\" = \"no\" ] ]; then
            if $darwin ; then
                javaHome=""
                dirname \"$javaExecutable\"
                javaExecutable=""
                cd \"$javaHome\" && pwd -P`/javac"
            else
                javaExecutable=""
                readlink -f \"$javaExecutable\"
            fi
            javaHome=""
            dirname \"$javaExecutable\"
            javaHome=""
            expr "$javaHome" : \"(.*)/bin\"
            JAVA_HOME="$javaHome"
            export JAVA_HOME
        fi
    fi
fi

```

```

if [ -z "$JAVACMD" ]; then
    if [ -n "$JAVA_HOME" ]; then
        if [ -x "$JAVA_HOME/jre/sh/java" ]; then
            # IBM's JDK on AIX uses strange locations for the executables
            JAVACMD="$JAVA_HOME/jre/sh/java"
        else
            JAVACMD="$JAVA_HOME/bin/java"
        fi
    else
        JAVACMD=""
        which java`
    fi
fi

```

```

if [ ! -x "$JAVACMD" ]; then
    echo "Error: JAVA_HOME is not defined correctly." >&2
    echo " We cannot execute $JAVACMD" >&2
    exit 1
fi

```

```

if [ -z "$JAVA_HOME" ]; then
    echo "Warning: JAVA_HOME environment variable is not set."
fi

```

CLASSWORLDS_LAUNCHER=org.codehaus.plexus.classworlds.launcher.Launcher

```

# traverses directory structure from process work directory to filesystem root
# first directory with .mvn subdirectory is considered project base directory
find_maven_basedir() {

```

```

    if [ -z "$1" ]

```

```

then
    echo "Path not specified to find_maven_basedir"
    return 1
fi

basedir="$1"
wdir="$1"
while [ "$wdir" != '/' ] ; do
    if [ -d "$wdir"/.mvn ] ; then
        basedir=$wdir
        break
    fi
    # workaround for JBEAP-8937 (on Solaris 10/Sparc)
    if [ -d "${wdir}" ] ; then
        wdir=`cd "$wdir/.."; pwd`
    fi
    # end of workaround
done
echo "${basedir}"
}

# concatenates all lines of a file
concat_lines() {
    if [ -f "$1" ] ; then
        echo "$(tr -s '\n' ' ' < "$1")"
    fi
}

BASE_DIR=`find_maven_basedir "$(pwd)"`
if [ -z "$BASE_DIR" ] ; then
    exit 1;
fi

#####
#####
# Extension to allow automatically downloading the maven-wrapper.jar from Maven-central
# This allows using the maven wrapper in projects that prohibit checking in binary data.
#####
#####
if [ -r "$BASE_DIR/.mvn/wrapper/maven-wrapper.jar" ] ; then
    if [ "$MVNW_VERBOSE" = true ] ; then
        echo "Found .mvn/wrapper/maven-wrapper.jar"
    fi
else
    if [ "$MVNW_VERBOSE" = true ] ; then
        echo "Couldn't find .mvn/wrapper/maven-wrapper.jar, downloading it ..."
    fi
    if [ -n "$MVNW_REPOURL" ] ; then
        jarUrl="$MVNW_REPOURL/io/takari/maven-wrapper/0.5.6/maven-wrapper-0.5.6.jar"
    else

```



```

jarUrl="https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/0.5.6/maven-
wrapper-0.5.6.jar"
fi
while IFS="=" read key value; do
    case "$key" in (wrapperUrl) jarUrl="$value"; break ;;
    esac
done < "$BASE_DIR/.mvn/wrapper/maven-wrapper.properties"
if [ "$MVNW_VERBOSE" = true ]; then
    echo "Downloading from: $jarUrl"
fi
wrapperJarPath="$BASE_DIR/.mvn/wrapper/maven-wrapper.jar"
if $cygwin; then
    wrapperJarPath=`cygpath --path --windows "$wrapperJarPath"`
fi

if command -v wget > /dev/null; then
    if [ "$MVNW_VERBOSE" = true ]; then
        echo "Found wget ... using wget"
    fi
    if [ -z "$MVNW_USERNAME" ] || [ -z "$MVNW_PASSWORD" ]; then
        wget "$jarUrl" -O "$wrapperJarPath"
    else
        wget --http-user=$MVNW_USERNAME --http-password=$MVNW_PASSWORD
"$jarUrl" -O "$wrapperJarPath"
    fi
elif command -v curl > /dev/null; then
    if [ "$MVNW_VERBOSE" = true ]; then
        echo "Found curl ... using curl"
    fi
    if [ -z "$MVNW_USERNAME" ] || [ -z "$MVNW_PASSWORD" ]; then
        curl -o "$wrapperJarPath" "$jarUrl" -f
    else
        curl --user $MVNW_USERNAME:$MVNW_PASSWORD -o "$wrapperJarPath"
"$jarUrl" -f
    fi
else
    if [ "$MVNW_VERBOSE" = true ]; then
        echo "Falling back to using Java to download"
    fi
    javaClass="$BASE_DIR/.mvn/wrapper/MavenWrapperDownloader.java"
    # For Cygwin, switch paths to Windows format before running javac
    if $cygwin; then
        javaClass=`cygpath --path --windows "$javaClass"`
    fi
    if [ -e "$javaClass" ]; then
        if [ ! -e "$BASE_DIR/.mvn/wrapper/MavenWrapperDownloader.class" ]; then
            if [ "$MVNW_VERBOSE" = true ]; then
                echo " - Compiling MavenWrapperDownloader.java ..."
            fi

```

```

    # Compiling the Java class
    (" $JAVA_HOME/bin/javac" "$javaClass")
  fi
  if [ -e "$BASE_DIR/.mvn/wrapper/MavenWrapperDownloader.class" ]; then
    # Running the downloader
    if [ "$MVNW_VERBOSE" = true ]; then
      echo " - Running MavenWrapperDownloader.java ..."
    fi
    (" $JAVA_HOME/bin/java" -cp .mvn/wrapper MavenWrapperDownloader
"$MAVEN_PROJECTBASEDIR")
  fi
fi
fi
fi
#####
#####
# End of extension
#####
#####

export MAVEN_PROJECTBASEDIR=${MAVEN_BASEDIR:-"$BASE_DIR"}
if [ "$MVNW_VERBOSE" = true ]; then
  echo $MAVEN_PROJECTBASEDIR
fi
MAVEN_OPTS="$(concat_lines "$MAVEN_PROJECTBASEDIR/.mvn/jvm.config")
$MAVEN_OPTS"

# For Cygwin, switch paths to Windows format before running java
if $cygwin; then
  [ -n "$M2_HOME" ] &&
  M2_HOME=`cygpath --path --windows "$M2_HOME"`
  [ -n "$JAVA_HOME" ] &&
  JAVA_HOME=`cygpath --path --windows "$JAVA_HOME"`
  [ -n "$CLASSPATH" ] &&
  CLASSPATH=`cygpath --path --windows "$CLASSPATH"`
  [ -n "$MAVEN_PROJECTBASEDIR" ] &&
  MAVEN_PROJECTBASEDIR=`cygpath --path --windows
"$MAVEN_PROJECTBASEDIR"`
fi

# Provide a "standardized" way to retrieve the CLI args that will
# work with both Windows and non-Windows executions.
MAVEN_CMD_LINE_ARGS="$MAVEN_CONFIG $@"
export MAVEN_CMD_LINE_ARGS

WRAPPER_LAUNCHER=org.apache.maven.wrapper.MavenWrapperMain

exec "$JAVACMD" \
  $MAVEN_OPTS \
  -classpath "$MAVEN_PROJECTBASEDIR/.mvn/wrapper/maven-wrapper.jar" \

```

```
"-Dmaven.home=${M2_HOME}" "-
Dmaven.multiModuleProjectDirectory=${MAVEN_PROJECTBASEDIR}" \
${WRAPPER_LAUNCHER} $MAVEN_CONFIG "$@"
```

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.5.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.SpringTest</groupId>
  <artifactId>Testing-Spring-Jenkins</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Spring-Jenkins</name>
  <description>Spring Boot -Jenkins</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-thymeleaf</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

</project>
```