/\*

\* Copyright 2007-present the original author or authors.

\*

\* Licensed 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.

\*/

import java.net.\*;

import java.io.\*;

import java.nio.channels.\*;

import java.util.Properties;

public class MavenWrapperDownloader {

private static final String WRAPPER\_VERSION = "0.5.6";

/\*\*

\* Default URL to download the maven-wrapper.jar from, if no 'downloadUrl' is provided.

\*/

private static final String DEFAULT\_DOWNLOAD\_URL = "https://repo.maven.apache.org/maven2/io/takari/maven-wrapper/"

+ WRAPPER\_VERSION + "/maven-wrapper-" + WRAPPER\_VERSION + ".jar";

/\*\*

\* Path to the maven-wrapper.properties file, which might contain a downloadUrl property to

\* use instead of the default one.

\*/

private static final String MAVEN\_WRAPPER\_PROPERTIES\_PATH =

".mvn/wrapper/maven-wrapper.properties";

/\*\*

\* Path where the maven-wrapper.jar will be saved to.

\*/

private static final String MAVEN\_WRAPPER\_JAR\_PATH =

".mvn/wrapper/maven-wrapper.jar";

/\*\*

\* Name of the property which should be used to override the default download url for the wrapper.

\*/

private static final String PROPERTY\_NAME\_WRAPPER\_URL = "wrapperUrl";

public static void main(String args[]) {

System.out.println("- Downloader started");

File baseDirectory = new File(args[0]);

System.out.println("- Using base directory: " + baseDirectory.getAbsolutePath());

// If the maven-wrapper.properties exists, read it and check if it contains a custom

// wrapperUrl parameter.

File mavenWrapperPropertyFile = new File(baseDirectory, MAVEN\_WRAPPER\_PROPERTIES\_PATH);

String url = DEFAULT\_DOWNLOAD\_URL;

if(mavenWrapperPropertyFile.exists()) {

FileInputStream mavenWrapperPropertyFileInputStream = null;

try {

mavenWrapperPropertyFileInputStream = new FileInputStream(mavenWrapperPropertyFile);

Properties mavenWrapperProperties = new Properties();

mavenWrapperProperties.load(mavenWrapperPropertyFileInputStream);

url = mavenWrapperProperties.getProperty(PROPERTY\_NAME\_WRAPPER\_URL, url);

} catch (IOException e) {

System.out.println("- ERROR loading '" + MAVEN\_WRAPPER\_PROPERTIES\_PATH + "'");

} finally {

try {

if(mavenWrapperPropertyFileInputStream != null) {

mavenWrapperPropertyFileInputStream.close();

}

} catch (IOException e) {

// Ignore ...

}

}

}

System.out.println("- Downloading from: " + url);

File outputFile = new File(baseDirectory.getAbsolutePath(), MAVEN\_WRAPPER\_JAR\_PATH);

if(!outputFile.getParentFile().exists()) {

if(!outputFile.getParentFile().mkdirs()) {

System.out.println(

"- ERROR creating output directory '" + outputFile.getParentFile().getAbsolutePath() + "'");

}

}

System.out.println("- Downloading to: " + outputFile.getAbsolutePath());

try {

downloadFileFromURL(url, outputFile);

System.out.println("Done");

System.exit(0);

} catch (Throwable e) {

System.out.println("- Error downloading");

e.printStackTrace();

System.exit(1);

}

}

private static void downloadFileFromURL(String urlString, File destination) throws Exception {

if (System.getenv("MVNW\_USERNAME") != null && System.getenv("MVNW\_PASSWORD") != null) {

String username = System.getenv("MVNW\_USERNAME");

char[] password = System.getenv("MVNW\_PASSWORD").toCharArray();

Authenticator.setDefault(new Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(username, password);

}

});

}

URL website = new URL(urlString);

ReadableByteChannel rbc;

rbc = Channels.newChannel(website.openStream());

FileOutputStream fos = new FileOutputStream(destination);

fos.getChannel().transferFrom(rbc, 0, Long.MAX\_VALUE);

fos.close();

rbc.close();

}

}

package com.project.Feedback.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.MediaType;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.ResponseBody;

import org.springframework.web.bind.annotation.RestController;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.services.FeedbackService;

@RestController

public class FeedbackController {

@Autowired

FeedbackService feedbackService;

@GetMapping("/feedback")

public Iterable<Feedback> getAllFeedbacks(){

return feedbackService.GetAllFeedback();

}

@PostMapping(path="/feedback", consumes= {MediaType.APPLICATION\_JSON\_VALUE})

public Feedback addNewFeedback(@RequestBody Feedback fb) {

Feedback newFb = new Feedback(fb.getComments(), fb.getRating(), fb.getUser());

feedbackService.addNewFeedback(newFb);

return newFb;

}

}

package com.project.Feedback.controllers;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PostMapping;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.services.FeedbackService;

@Controller

public class TestFormController {

@Autowired

FeedbackService feedbackService;

@GetMapping("/test\_form")

public String showTestForm(ModelMap model) {

model.addAttribute("test", new Feedback());

return "testformjsp";

}

@PostMapping("/test\_form")

public String submitTestForm(@ModelAttribute("testUser") Feedback fb, ModelMap m) {

feedbackService.addNewFeedback(fb);

m.addAttribute("test", fb);

return "post";

}

// TODO: Implement form submission

// TODO: call RestTemplate and make json request to localhost.../feedback

}

//RestTemplate restTemplate = new RestTemplate();

//URL testForm = new URL("http://localhost:8090/feedbacks/{feedback}");

//ResponseEntity<String> response = restTemplate.getForEntity(testForm + "/7", String.class);

//ObjectMapper mapper = new ObjectMapper();

//JsonNode root = mapper.readTree(response.getBody());

//JsonNode name = root.path("name");

//model.addAttribute(name);

//String result = restTemplate.getForObject("http://localhost:8090/feedbacks/{feedback}", String.class, 7);

package com.project.Feedback.entities;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.validation.constraints.NotNull;

import lombok.Data;

@Entity

@Data

public class Feedback {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name="id")

@NotNull

private Integer id;

@Column(name="comments")

private String comments;

@Column(name="rating")

@NotNull

private int rating;

@Column(name="user")

private String user;

public Feedback() {

super();

}

public Feedback(String comments, Integer rating, String user) {

this.comments = comments;

this.rating = rating;

this.user = user;

}

/\*

\* Needed the setters and getters to be able to add name and comments otherwise

\* they are nulls when entering the SQL DB

\*/

public String getComments() {

return comments;

}

public void setComments(String comments) {

this.comments = comments;

}

public Integer getRating() {

return rating;

}

public void setRating(Integer rating) {

this.rating = rating;

}

public String getUser() {

return user;

}

public void setUser(String user) {

this.user = user;

}

@Override

public String toString() {

return "Feedback [id=" + id + ", comments=" + comments + ", rating=" + rating + ", user=" + user + "]";

}

}

package com.project.Feedback.repositories;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.project.Feedback.entities.Feedback;

@Repository

public interface FeedbackRepository extends CrudRepository<Feedback, Integer> {

public Feedback findByUser(String feedback);

}

package com.project.Feedback.services;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.repositories.FeedbackRepository;

@Service

public class FeedbackService {

@Autowired

FeedbackRepository feedbackRepo;

public Iterable<Feedback> GetAllFeedback() {

return feedbackRepo.findAll();

}

public Feedback addNewFeedback(Feedback fb) {

return feedbackRepo.save(fb);

}

}

package com.project.Feedback;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FeedbackApplication {

public static void main(String[] args) {

SpringApplication.run(FeedbackApplication.class, args);

}

}

package com.project.Feedback;

import static org.junit.jupiter.api.Assertions.assertEquals;

import javax.persistence.EntityManager;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import com.project.Feedback.entities.Feedback;

import com.project.Feedback.repositories.FeedbackRepository;

@SpringBootTest

class FeedbackApplicationTests {

@Autowired

EntityManager entityManager;

@Autowired

FeedbackRepository feedbackRepo;

// @Autowired

@Test

void shouldFindByUser() {

Feedback testFeedback = new Feedback("Dummy Test", 5, "dummy");

entityManager.persist(testFeedback);

entityManager.flush();

Feedback cmp = feedbackRepo.findByUser(testFeedback.getUser());

assertEquals(cmp.getUser(), testFeedback.getUser());

}

}

#!/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} $MAVEN\_CONFIG "$@"

<?xml version="1.0" encoding="UTF-8"?>

[<project xsi:schemaLocation="**http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd**" xmlns:xsi="**http://www.w3.org/2001/XMLSchema-instance**" xmlns="**http://maven.apache.org/POM/4.0.0**">](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<modelVersion>4.0.0</modelVersion>[<parent>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-parent</artifactId><version>2.4.3</version><relativePath/>

<!-- lookup parent from repository -->

</parent><groupId>com.project</groupId><artifactId>Feedback</artifactId><version>0.0.1-SNAPSHOT</version><name>Feedback</name><description>Create a Spring Boot project that will capture user feedback using a REST endpoint. The REST resource will take in parameters using HTTP POST. The feedback data will be then added to a database table.</description>[<properties>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<java.version>1.8</java.version></properties>[<dependencies><dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-data-jpa</artifactId></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-data-rest</artifactId></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-jersey</artifactId></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-web</artifactId></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-devtools</artifactId><scope>runtime</scope><optional>true</optional></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>mysql</groupId><artifactId>mysql-connector-java</artifactId><scope>runtime</scope></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-starter-test</artifactId><scope>test</scope></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.projectlombok</groupId><artifactId>lombok</artifactId><optional>true</optional></dependency>

<!-- this Dependency helps make sure that pathing works correct-->

[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.apache.tomcat.embed</groupId><artifactId>tomcat-embed-jasper</artifactId><scope>provided</scope></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>javax.xml.bind</groupId><artifactId>jaxb-api</artifactId></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.javassist</groupId><artifactId>javassist</artifactId><version>3.25.0-GA</version></dependency>[<dependency>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>javax.servlet</groupId><artifactId>jstl</artifactId><version>1.2</version></dependency></dependencies>[<build><plugins><plugin>](file:///F:\Practice_Assignments\Phase3_Practice_Assignments-main\Feedback\pom.xml)<groupId>org.springframework.boot</groupId><artifactId>spring-boot-maven-plugin</artifactId></plugin></plugins></build></project>