**CODE:**

# 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

#

# http://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.

wrapperVersion=3.3.2

distributionType=only-script

distributionUrl=https://repo.maven.apache.org/maven2/org/apache/maven/apache-maven/3.9.10/apache-maven-3.9.10-bin.zip

package com.cognizant.ormservice.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

// Getters and setters

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

// toString()

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}package com.cognizant.ormservice.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.cognizant.ormservice.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

package com.cognizant.ormservice.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

package com.cognizant.ormservice.service;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormservice.model.Country;

import com.cognizant.ormservice.repository.CountryRepository;

import com.cognizant.ormservice.service.exception.CountryNotFoundException;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

Optional<Country> result = countryRepository.findById(code);

if (!result.isPresent()) {

throw new CountryNotFoundException("Country with code " + code + " not found.");

}

return result.get();

}

}

package com.cognizant.ormservice;

import com.cognizant.ormservice.model.Country;

import com.cognizant.ormservice.service.CountryService;

import com.cognizant.ormservice.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import jakarta.annotation.PostConstruct;

@SpringBootApplication

public class OrmserviceApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmserviceApplication.class);

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmserviceApplication.class, args);

}

@PostConstruct

public void testFindCountryByCode() {

LOGGER.info("Start");

try {

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

}

<!DOCTYPE html>

<html>

<head>

<title>ORM Service</title>

</head>

<body>

<h1>Welcome to Spring Boot ORM Service!</h1>

</body>

</html>

# Database connection

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=

# JDBC Driver

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

# JPA & Hibernate settings

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

package com.cognizant.ormservice;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class OrmserviceApplicationTests {

@Test

void contextLoads() {

}

}

<# : batch portion

@REM ----------------------------------------------------------------------------

@REM Licensed to the Apache Software Foundation (ASF) under one

@REM or more contributor license agreements. See the NOTICE file

@REM distributed with this work for additional information

@REM regarding copyright ownership. The ASF licenses this file

@REM to you under the Apache License, Version 2.0 (the

@REM "License"); you may not use this file except in compliance

@REM with the License. You may obtain a copy of the License at

@REM

@REM http://www.apache.org/licenses/LICENSE-2.0

@REM

@REM Unless required by applicable law or agreed to in writing,

@REM software distributed under the License is distributed on an

@REM "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

@REM KIND, either express or implied. See the License for the

@REM specific language governing permissions and limitations

@REM under the License.

@REM ----------------------------------------------------------------------------

@REM ----------------------------------------------------------------------------

@REM Apache Maven Wrapper startup batch script, version 3.3.2

@REM

@REM Optional ENV vars

@REM MVNW\_REPOURL - repo url base for downloading maven distribution

@REM MVNW\_USERNAME/MVNW\_PASSWORD - user and password for downloading maven

@REM MVNW\_VERBOSE - true: enable verbose log; others: silence the output

@REM ----------------------------------------------------------------------------

@IF "%\_\_MVNW\_ARG0\_NAME\_\_%"=="" (SET \_\_MVNW\_ARG0\_NAME\_\_=%~nx0)

@SET \_\_MVNW\_CMD\_\_=

@SET \_\_MVNW\_ERROR\_\_=

@SET \_\_MVNW\_PSMODULEP\_SAVE=%PSModulePath%

@SET PSModulePath=

@FOR /F "usebackq tokens=1\* delims==" %%A IN (`powershell -noprofile "& {$scriptDir='%~dp0'; $script='%\_\_MVNW\_ARG0\_NAME\_\_%'; icm -ScriptBlock ([Scriptblock]::Create((Get-Content -Raw '%~f0'))) -NoNewScope}"`) DO @(

IF "%%A"=="MVN\_CMD" (set \_\_MVNW\_CMD\_\_=%%B) ELSE IF "%%B"=="" (echo %%A) ELSE (echo %%A=%%B)

)

@SET PSModulePath=%\_\_MVNW\_PSMODULEP\_SAVE%

@SET \_\_MVNW\_PSMODULEP\_SAVE=

@SET \_\_MVNW\_ARG0\_NAME\_\_=

@SET MVNW\_USERNAME=

@SET MVNW\_PASSWORD=

@IF NOT "%\_\_MVNW\_CMD\_\_%"=="" (%\_\_MVNW\_CMD\_\_% %\*)

@echo Cannot start maven from wrapper >&2 && exit /b 1

@GOTO :EOF

: end batch / begin powershell #>

$ErrorActionPreference = "Stop"

if ($env:MVNW\_VERBOSE -eq "true") {

$VerbosePreference = "Continue"

}

# calculate distributionUrl, requires .mvn/wrapper/maven-wrapper.properties

$distributionUrl = (Get-Content -Raw "$scriptDir/.mvn/wrapper/maven-wrapper.properties" | ConvertFrom-StringData).distributionUrl

if (!$distributionUrl) {

Write-Error "cannot read distributionUrl property in $scriptDir/.mvn/wrapper/maven-wrapper.properties"

}

switch -wildcard -casesensitive ( $($distributionUrl -replace '^.\*/','') ) {

"maven-mvnd-\*" {

$USE\_MVND = $true

$distributionUrl = $distributionUrl -replace '-bin\.[^.]\*$',"-windows-amd64.zip"

$MVN\_CMD = "mvnd.cmd"

break

}

default {

$USE\_MVND = $false

$MVN\_CMD = $script -replace '^mvnw','mvn'

break

}

}

# apply MVNW\_REPOURL and calculate MAVEN\_HOME

# maven home pattern: ~/.m2/wrapper/dists/{apache-maven-<version>,maven-mvnd-<version>-<platform>}/<hash>

if ($env:MVNW\_REPOURL) {

$MVNW\_REPO\_PATTERN = if ($USE\_MVND) { "/org/apache/maven/" } else { "/maven/mvnd/" }

$distributionUrl = "$env:MVNW\_REPOURL$MVNW\_REPO\_PATTERN$($distributionUrl -replace '^.\*'+$MVNW\_REPO\_PATTERN,'')"

}

$distributionUrlName = $distributionUrl -replace '^.\*/',''

$distributionUrlNameMain = $distributionUrlName -replace '\.[^.]\*$','' -replace '-bin$',''

$MAVEN\_HOME\_PARENT = "$HOME/.m2/wrapper/dists/$distributionUrlNameMain"

if ($env:MAVEN\_USER\_HOME) {

$MAVEN\_HOME\_PARENT = "$env:MAVEN\_USER\_HOME/wrapper/dists/$distributionUrlNameMain"

}

$MAVEN\_HOME\_NAME = ([System.Security.Cryptography.MD5]::Create().ComputeHash([byte[]][char[]]$distributionUrl) | ForEach-Object {$\_.ToString("x2")}) -join ''

$MAVEN\_HOME = "$MAVEN\_HOME\_PARENT/$MAVEN\_HOME\_NAME"

if (Test-Path -Path "$MAVEN\_HOME" -PathType Container) {

Write-Verbose "found existing MAVEN\_HOME at $MAVEN\_HOME"

Write-Output "MVN\_CMD=$MAVEN\_HOME/bin/$MVN\_CMD"

exit $?

}

if (! $distributionUrlNameMain -or ($distributionUrlName -eq $distributionUrlNameMain)) {

Write-Error "distributionUrl is not valid, must end with \*-bin.zip, but found $distributionUrl"

}

# prepare tmp dir

$TMP\_DOWNLOAD\_DIR\_HOLDER = New-TemporaryFile

$TMP\_DOWNLOAD\_DIR = New-Item -Itemtype Directory -Path "$TMP\_DOWNLOAD\_DIR\_HOLDER.dir"

$TMP\_DOWNLOAD\_DIR\_HOLDER.Delete() | Out-Null

trap {

if ($TMP\_DOWNLOAD\_DIR.Exists) {

try { Remove-Item $TMP\_DOWNLOAD\_DIR -Recurse -Force | Out-Null }

catch { Write-Warning "Cannot remove $TMP\_DOWNLOAD\_DIR" }

}

}

New-Item -Itemtype Directory -Path "$MAVEN\_HOME\_PARENT" -Force | Out-Null

# Download and Install Apache Maven

Write-Verbose "Couldn't find MAVEN\_HOME, downloading and installing it ..."

Write-Verbose "Downloading from: $distributionUrl"

Write-Verbose "Downloading to: $TMP\_DOWNLOAD\_DIR/$distributionUrlName"

$webclient = New-Object System.Net.WebClient

if ($env:MVNW\_USERNAME -and $env:MVNW\_PASSWORD) {

$webclient.Credentials = New-Object System.Net.NetworkCredential($env:MVNW\_USERNAME, $env:MVNW\_PASSWORD)

}

[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12

$webclient.DownloadFile($distributionUrl, "$TMP\_DOWNLOAD\_DIR/$distributionUrlName") | Out-Null

# If specified, validate the SHA-256 sum of the Maven distribution zip file

$distributionSha256Sum = (Get-Content -Raw "$scriptDir/.mvn/wrapper/maven-wrapper.properties" | ConvertFrom-StringData).distributionSha256Sum

if ($distributionSha256Sum) {

if ($USE\_MVND) {

Write-Error "Checksum validation is not supported for maven-mvnd. `nPlease disable validation by removing 'distributionSha256Sum' from your maven-wrapper.properties."

}

Import-Module $PSHOME\Modules\Microsoft.PowerShell.Utility -Function Get-FileHash

if ((Get-FileHash "$TMP\_DOWNLOAD\_DIR/$distributionUrlName" -Algorithm SHA256).Hash.ToLower() -ne $distributionSha256Sum) {

Write-Error "Error: Failed to validate Maven distribution SHA-256, your Maven distribution might be compromised. If you updated your Maven version, you need to update the specified distributionSha256Sum property."

}

}

# unzip and move

Expand-Archive "$TMP\_DOWNLOAD\_DIR/$distributionUrlName" -DestinationPath "$TMP\_DOWNLOAD\_DIR" | Out-Null

Rename-Item -Path "$TMP\_DOWNLOAD\_DIR/$distributionUrlNameMain" -NewName $MAVEN\_HOME\_NAME | Out-Null

try {

Move-Item -Path "$TMP\_DOWNLOAD\_DIR/$MAVEN\_HOME\_NAME" -Destination $MAVEN\_HOME\_PARENT | Out-Null

} catch {

if (! (Test-Path -Path "$MAVEN\_HOME" -PathType Container)) {

Write-Error "fail to move MAVEN\_HOME"

}

} finally {

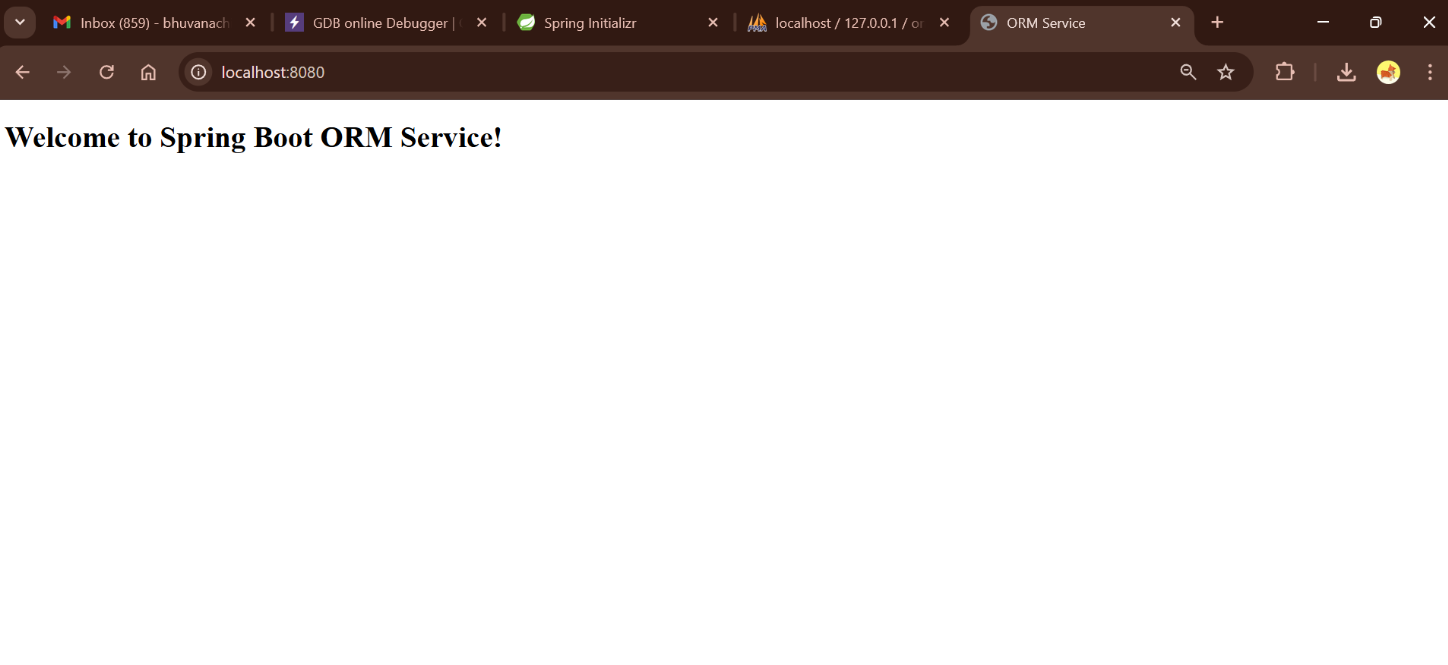
try { Remove-Item $TMP\_DOWNLOAD\_DIR -Recurse -Force | Out-Null }

catch { Write-Warning "Cannot remove $TMP\_DOWNLOAD\_DIR" }

}

Write-Output "MVN\_CMD=$MAVEN\_HOME/bin/$MVN\_CMD"

**OUTPUT:**

****