**Spring Data JPA - Quick Example**

**Country.java**

package com.cognizant.demo.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@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;

}

@Override

public String toString() {

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

}

}

**CountryRepository.java**

package com.cognizant.demo.repository;

import com.cognizant.demo.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.demo.service;

import com.cognizant.demo.model.Country;

import com.cognizant.demo.repository.CountryRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.demo;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication {

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

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

}

}

**application.properties**

spring.application.name=Spring data Gpa Quick Example

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# DB Configuration

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

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

spring.datasource.username=root

spring.datasource.password=1234

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Test**

package com.cognizant.demo;

import com.cognizant.demo.model.Country;

import com.cognizant.demo.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

**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

#

# 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.

# ----------------------------------------------------------------------------

# ----------------------------------------------------------------------------

# Apache Maven Wrapper startup batch script, version 3.3.2

#

# Optional ENV vars

# -----------------

# JAVA\_HOME - location of a JDK home dir, required when download maven via java source

# MVNW\_REPOURL - repo url base for downloading maven distribution

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

# MVNW\_VERBOSE - true: enable verbose log; debug: trace the mvnw script; others: silence the output

# ----------------------------------------------------------------------------

set -euf

[ "${MVNW\_VERBOSE-}" != debug ] || set -x

# OS specific support.

native\_path() { printf %s\\n "$1"; }

case "$(uname)" in

CYGWIN\* | MINGW\*)

[ -z "${JAVA\_HOME-}" ] || JAVA\_HOME="$(cygpath --unix "$JAVA\_HOME")"

native\_path() { cygpath --path --windows "$1"; }

;;

esac

# set JAVACMD and JAVACCMD

set\_java\_home() {

# For Cygwin and MinGW, ensure paths are in Unix format before anything is touched

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"

JAVACCMD="$JAVA\_HOME/jre/sh/javac"

else

JAVACMD="$JAVA\_HOME/bin/java"

JAVACCMD="$JAVA\_HOME/bin/javac"

if [ ! -x "$JAVACMD" ] || [ ! -x "$JAVACCMD" ]; then

echo "The JAVA\_HOME environment variable is not defined correctly, so mvnw cannot run." >&2

echo "JAVA\_HOME is set to \"$JAVA\_HOME\", but \"\$JAVA\_HOME/bin/java\" or \"\$JAVA\_HOME/bin/javac\" does not exist." >&2

return 1

fi

fi

else

JAVACMD="$(

'set' +e

'unset' -f command 2>/dev/null

'command' -v java

)" || :

JAVACCMD="$(

'set' +e

'unset' -f command 2>/dev/null

'command' -v javac

)" || :

if [ ! -x "${JAVACMD-}" ] || [ ! -x "${JAVACCMD-}" ]; then

echo "The java/javac command does not exist in PATH nor is JAVA\_HOME set, so mvnw cannot run." >&2

return 1

fi

fi

}

# hash string like Java String::hashCode

hash\_string() {

str="${1:-}" h=0

while [ -n "$str" ]; do

char="${str%"${str#?}"}"

h=$(((h \* 31 + $(LC\_CTYPE=C printf %d "'$char")) % 4294967296))

str="${str#?}"

done

printf %x\\n $h

}

verbose() { :; }

[ "${MVNW\_VERBOSE-}" != true ] || verbose() { printf %s\\n "${1-}"; }

die() {

printf %s\\n "$1" >&2

exit 1

}

trim() {

# MWRAPPER-139:

# Trims trailing and leading whitespace, carriage returns, tabs, and linefeeds.

# Needed for removing poorly interpreted newline sequences when running in more

# exotic environments such as mingw bash on Windows.

printf "%s" "${1}" | tr -d '[:space:]'

}

# parse distributionUrl and optional distributionSha256Sum, requires .mvn/wrapper/maven-wrapper.properties

while IFS="=" read -r key value; do

case "${key-}" in

distributionUrl) distributionUrl=$(trim "${value-}") ;;

distributionSha256Sum) distributionSha256Sum=$(trim "${value-}") ;;

esac

done <"${0%/\*}/.mvn/wrapper/maven-wrapper.properties"

[ -n "${distributionUrl-}" ] || die "cannot read distributionUrl property in ${0%/\*}/.mvn/wrapper/maven-wrapper.properties"

case "${distributionUrl##\*/}" in

maven-mvnd-\*bin.\*)

MVN\_CMD=mvnd.sh \_MVNW\_REPO\_PATTERN=/maven/mvnd/

case "${PROCESSOR\_ARCHITECTURE-}${PROCESSOR\_ARCHITEW6432-}:$(uname -a)" in

\*AMD64:CYGWIN\* | \*AMD64:MINGW\*) distributionPlatform=windows-amd64 ;;

:Darwin\*x86\_64) distributionPlatform=darwin-amd64 ;;

:Darwin\*arm64) distributionPlatform=darwin-aarch64 ;;

:Linux\*x86\_64\*) distributionPlatform=linux-amd64 ;;

\*)

echo "Cannot detect native platform for mvnd on $(uname)-$(uname -m), use pure java version" >&2

distributionPlatform=linux-amd64

;;

esac

distributionUrl="${distributionUrl%-bin.\*}-$distributionPlatform.zip"

;;

maven-mvnd-\*) MVN\_CMD=mvnd.sh \_MVNW\_REPO\_PATTERN=/maven/mvnd/ ;;

\*) MVN\_CMD="mvn${0##\*/mvnw}" \_MVNW\_REPO\_PATTERN=/org/apache/maven/ ;;

esac

# apply MVNW\_REPOURL and calculate MAVEN\_HOME

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

[ -z "${MVNW\_REPOURL-}" ] || distributionUrl="$MVNW\_REPOURL$\_MVNW\_REPO\_PATTERN${distributionUrl#\*"$\_MVNW\_REPO\_PATTERN"}"

distributionUrlName="${distributionUrl##\*/}"

distributionUrlNameMain="${distributionUrlName%.\*}"

distributionUrlNameMain="${distributionUrlNameMain%-bin}"

MAVEN\_USER\_HOME="${MAVEN\_USER\_HOME:-${HOME}/.m2}"

MAVEN\_HOME="${MAVEN\_USER\_HOME}/wrapper/dists/${distributionUrlNameMain-}/$(hash\_string "$distributionUrl")"

exec\_maven() {

unset MVNW\_VERBOSE MVNW\_USERNAME MVNW\_PASSWORD MVNW\_REPOURL || :

exec "$MAVEN\_HOME/bin/$MVN\_CMD" "$@" || die "cannot exec $MAVEN\_HOME/bin/$MVN\_CMD"

}

if [ -d "$MAVEN\_HOME" ]; then

verbose "found existing MAVEN\_HOME at $MAVEN\_HOME"

exec\_maven "$@"

fi

case "${distributionUrl-}" in

\*?-bin.zip | \*?maven-mvnd-?\*-?\*.zip) ;;

\*) die "distributionUrl is not valid, must match \*-bin.zip or maven-mvnd-\*.zip, but found '${distributionUrl-}'" ;;

esac

# prepare tmp dir

if TMP\_DOWNLOAD\_DIR="$(mktemp -d)" && [ -d "$TMP\_DOWNLOAD\_DIR" ]; then

clean() { rm -rf -- "$TMP\_DOWNLOAD\_DIR"; }

trap clean HUP INT TERM EXIT

else

die "cannot create temp dir"

fi

mkdir -p -- "${MAVEN\_HOME%/\*}"

# Download and Install Apache Maven

verbose "Couldn't find MAVEN\_HOME, downloading and installing it ..."

verbose "Downloading from: $distributionUrl"

verbose "Downloading to: $TMP\_DOWNLOAD\_DIR/$distributionUrlName"

# select .zip or .tar.gz

if ! command -v unzip >/dev/null; then

distributionUrl="${distributionUrl%.zip}.tar.gz"

distributionUrlName="${distributionUrl##\*/}"

fi

# verbose opt

\_\_MVNW\_QUIET\_WGET=--quiet \_\_MVNW\_QUIET\_CURL=--silent \_\_MVNW\_QUIET\_UNZIP=-q \_\_MVNW\_QUIET\_TAR=''

[ "${MVNW\_VERBOSE-}" != true ] || \_\_MVNW\_QUIET\_WGET='' \_\_MVNW\_QUIET\_CURL='' \_\_MVNW\_QUIET\_UNZIP='' \_\_MVNW\_QUIET\_TAR=v

# normalize http auth

case "${MVNW\_PASSWORD:+has-password}" in

'') MVNW\_USERNAME='' MVNW\_PASSWORD='' ;;

has-password) [ -n "${MVNW\_USERNAME-}" ] || MVNW\_USERNAME='' MVNW\_PASSWORD='' ;;

esac

if [ -z "${MVNW\_USERNAME-}" ] && command -v wget >/dev/null; then

verbose "Found wget ... using wget"

wget ${\_\_MVNW\_QUIET\_WGET:+"$\_\_MVNW\_QUIET\_WGET"} "$distributionUrl" -O "$TMP\_DOWNLOAD\_DIR/$distributionUrlName" || die "wget: Failed to fetch $distributionUrl"

elif [ -z "${MVNW\_USERNAME-}" ] && command -v curl >/dev/null; then

verbose "Found curl ... using curl"

curl ${\_\_MVNW\_QUIET\_CURL:+"$\_\_MVNW\_QUIET\_CURL"} -f -L -o "$TMP\_DOWNLOAD\_DIR/$distributionUrlName" "$distributionUrl" || die "curl: Failed to fetch $distributionUrl"

elif set\_java\_home; then

verbose "Falling back to use Java to download"

javaSource="$TMP\_DOWNLOAD\_DIR/Downloader.java"

targetZip="$TMP\_DOWNLOAD\_DIR/$distributionUrlName"

cat >"$javaSource" <<-END

public class Downloader extends java.net.Authenticator

{

protected java.net.PasswordAuthentication getPasswordAuthentication()

{

return new java.net.PasswordAuthentication( System.getenv( "MVNW\_USERNAME" ), System.getenv( "MVNW\_PASSWORD" ).toCharArray() );

}

public static void main( String[] args ) throws Exception

{

setDefault( new Downloader() );

java.nio.file.Files.copy( java.net.URI.create( args[0] ).toURL().openStream(), java.nio.file.Paths.get( args[1] ).toAbsolutePath().normalize() );

}

}

END

# For Cygwin/MinGW, switch paths to Windows format before running javac and java

verbose " - Compiling Downloader.java ..."

"$(native\_path "$JAVACCMD")" "$(native\_path "$javaSource")" || die "Failed to compile Downloader.java"

verbose " - Running Downloader.java ..."

"$(native\_path "$JAVACMD")" -cp "$(native\_path "$TMP\_DOWNLOAD\_DIR")" Downloader "$distributionUrl" "$(native\_path "$targetZip")"

fi

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

if [ -n "${distributionSha256Sum-}" ]; then

distributionSha256Result=false

if [ "$MVN\_CMD" = mvnd.sh ]; then

echo "Checksum validation is not supported for maven-mvnd." >&2

echo "Please disable validation by removing 'distributionSha256Sum' from your maven-wrapper.properties." >&2

exit 1

elif command -v sha256sum >/dev/null; then

if echo "$distributionSha256Sum $TMP\_DOWNLOAD\_DIR/$distributionUrlName" | sha256sum -c >/dev/null 2>&1; then

distributionSha256Result=true

fi

elif command -v shasum >/dev/null; then

if echo "$distributionSha256Sum $TMP\_DOWNLOAD\_DIR/$distributionUrlName" | shasum -a 256 -c >/dev/null 2>&1; then

distributionSha256Result=true

fi

else

echo "Checksum validation was requested but neither 'sha256sum' or 'shasum' are available." >&2

echo "Please install either command, or disable validation by removing 'distributionSha256Sum' from your maven-wrapper.properties." >&2

exit 1

fi

if [ $distributionSha256Result = false ]; then

echo "Error: Failed to validate Maven distribution SHA-256, your Maven distribution might be compromised." >&2

echo "If you updated your Maven version, you need to update the specified distributionSha256Sum property." >&2

exit 1

fi

fi

# unzip and move

if command -v unzip >/dev/null; then

unzip ${\_\_MVNW\_QUIET\_UNZIP:+"$\_\_MVNW\_QUIET\_UNZIP"} "$TMP\_DOWNLOAD\_DIR/$distributionUrlName" -d "$TMP\_DOWNLOAD\_DIR" || die "failed to unzip"

else

tar xzf${\_\_MVNW\_QUIET\_TAR:+"$\_\_MVNW\_QUIET\_TAR"} "$TMP\_DOWNLOAD\_DIR/$distributionUrlName" -C "$TMP\_DOWNLOAD\_DIR" || die "failed to untar"

fi

printf %s\\n "$distributionUrl" >"$TMP\_DOWNLOAD\_DIR/$distributionUrlNameMain/mvnw.url"

mv -- "$TMP\_DOWNLOAD\_DIR/$distributionUrlNameMain" "$MAVEN\_HOME" || [ -d "$MAVEN\_HOME" ] || die "fail to move MAVEN\_HOME"

clean || :

exec\_maven "$@"

**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>3.5.3</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.cognizant</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Spring data Gpa Quick Example</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</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>