**Project Title:** **Landing Page with Contact Form and Spring Boot Backend**

**Technologies used:** **Frontend (HTML/CSS/JS)  
Backend (Spring Boot/Java)  
Database (H2)**

**File name: ContactController.java**

**Code:**

package com.example.contactform.controller;

import com.example.contactform.model.Contact;

import com.example.contactform.service.ContactService;

import jakarta.validation.Valid;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/contact")

@CrossOrigin(origins = "\*")

public class ContactController {

    private final ContactService service;

    public ContactController(ContactService service) {

        this.service = service;

    }

    @PostMapping

    public ResponseEntity<String> submitForm(@Valid @RequestBody Contact contact) {

        if (!service.isNameAvailable(contact.getName())) {

            return ResponseEntity.badRequest().body("Name already exists");

        }

        service.saveContact(contact);

        return ResponseEntity.ok("Form submitted successfully");

    }

}

**File name:Contact.java**

package com.example.contactform.model;

import jakarta.persistence.\*;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotBlank;

import jakarta.validation.constraints.Pattern;

import lombok.Data;

@Data

@Entity

public class Contact {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    @NotBlank(message = "Name is required")

    @Pattern(regexp = "^[A-Za-z ]+$", message = "Name must contain only letters and spaces")

    @Column(unique = true)

    private String name;

    @NotBlank(message = "Email is required")

    @Email(message = "Invalid email format")

    private String email;

    @NotBlank(message = "Message is required")

    @Column(length = 1000)

    private String message;

}

**File name:ContactRequest.java**

package com.example.contactform.model;

import jakarta.validation.constraints.Email;

import jakarta.validation.constraints.NotEmpty;

import jakarta.validation.constraints.Size;

public class ContactRequest {

    @NotEmpty(message = "Name is required")

    private String name;

    @Email(message = "Invalid email format")

    private String email;

    @Size(min = 10, message = "Message must be at least 10 characters long")

    private String message;

    // Getters and Setters

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getMessage() {

        return message;

    }

    public void setMessage(String message) {

        this.message = message;

    }

}

**File Name: ContactRepository.java**

package com.example.contactform.repository;

import com.example.contactform.model.Contact;

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

import java.util.Optional;

public interface ContactRepository extends JpaRepository<Contact, Long> {

    Optional<Contact> findByNameIgnoreCase(String name);

}

**File Name:ContactService.java**

package com.example.contactform.service;

import com.example.contactform.model.Contact;

import com.example.contactform.repository.ContactRepository;

import org.springframework.stereotype.Service;

@Service

public class ContactService {

    private final ContactRepository repository;

    public ContactService(ContactRepository repository) {

        this.repository = repository;

    }

    public boolean isNameAvailable(String name) {

        return repository.findByNameIgnoreCase(name).isEmpty();

    }

    public Contact saveContact(Contact contact) {

        return repository.save(contact);

    }

}

**File Name:** **ContactformApplication.java**

package com.example.contactform;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ContactformApplication {

    public static void main(String[] args) {

        SpringApplication.run(ContactformApplication.class, args);

    }

}

**application.properties**

# H2 in-memory database

spring.datasource.url=jdbc:h2:mem:contactdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

# Server port

server.port=8080

**File name: ContactformApplicationTests.class**

// Source code is decompiled from a .class file using FernFlower decompiler (from Intellij IDEA).

package com.example.contactform;

import org.junit.jupiter.api.Test;

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

@SpringBootTest

class ContactformApplicationTests {

   ContactformApplicationTests() {

   }

   @Test

   void contextLoads() {

   }

}

**File name: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 3.3.3

#

# 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:]'

}

scriptDir="$(dirname "$0")"

scriptName="$(basename "$0")"

# 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 <"$scriptDir/.mvn/wrapper/maven-wrapper.properties"

[ -n "${distributionUrl-}" ] || die "cannot read distributionUrl property in $scriptDir/.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${scriptName#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

# Find the actual extracted directory name (handles snapshots where filename != directory name)

actualDistributionDir=""

# First try the expected directory name (for regular distributions)

if [ -d "$TMP\_DOWNLOAD\_DIR/$distributionUrlNameMain" ]; then

  if [ -f "$TMP\_DOWNLOAD\_DIR/$distributionUrlNameMain/bin/$MVN\_CMD" ]; then

    actualDistributionDir="$distributionUrlNameMain"

  fi

fi

# If not found, search for any directory with the Maven executable (for snapshots)

if [ -z "$actualDistributionDir" ]; then

  # enable globbing to iterate over items

  set +f

  for dir in "$TMP\_DOWNLOAD\_DIR"/\*; do

    if [ -d "$dir" ]; then

      if [ -f "$dir/bin/$MVN\_CMD" ]; then

        actualDistributionDir="$(basename "$dir")"

        break

      fi

    fi

  done

  set -f

fi

if [ -z "$actualDistributionDir" ]; then

  verbose "Contents of $TMP\_DOWNLOAD\_DIR:"

  verbose "$(ls -la "$TMP\_DOWNLOAD\_DIR")"

  die "Could not find Maven distribution directory in extracted archive"

fi

verbose "Found extracted Maven distribution directory: $actualDistributionDir"

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

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

clean || :

exec\_maven "$@"

**File name:mvnw.cmd**

<# : 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 3.3.3

@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 -eq $False) { "/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$',''

$MAVEN\_M2\_PATH = "$HOME/.m2"

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

  $MAVEN\_M2\_PATH = "$env:MAVEN\_USER\_HOME"

}

if (-not (Test-Path -Path $MAVEN\_M2\_PATH)) {

    New-Item -Path $MAVEN\_M2\_PATH -ItemType Directory | Out-Null

}

$MAVEN\_WRAPPER\_DISTS = $null

if ((Get-Item $MAVEN\_M2\_PATH).Target[0] -eq $null) {

  $MAVEN\_WRAPPER\_DISTS = "$MAVEN\_M2\_PATH/wrapper/dists"

} else {

  $MAVEN\_WRAPPER\_DISTS = (Get-Item $MAVEN\_M2\_PATH).Target[0] + "/wrapper/dists"

}

$MAVEN\_HOME\_PARENT = "$MAVEN\_WRAPPER\_DISTS/$distributionUrlNameMain"

$MAVEN\_HOME\_NAME = ([System.Security.Cryptography.SHA256]::Create().ComputeHash([byte[]][char[]]$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

# Find the actual extracted directory name (handles snapshots where filename != directory name)

$actualDistributionDir = ""

# First try the expected directory name (for regular distributions)

$expectedPath = Join-Path "$TMP\_DOWNLOAD\_DIR" "$distributionUrlNameMain"

$expectedMvnPath = Join-Path "$expectedPath" "bin/$MVN\_CMD"

if ((Test-Path -Path $expectedPath -PathType Container) -and (Test-Path -Path $expectedMvnPath -PathType Leaf)) {

  $actualDistributionDir = $distributionUrlNameMain

}

# If not found, search for any directory with the Maven executable (for snapshots)

if (!$actualDistributionDir) {

  Get-ChildItem -Path "$TMP\_DOWNLOAD\_DIR" -Directory | ForEach-Object {

    $testPath = Join-Path $\_.FullName "bin/$MVN\_CMD"

    if (Test-Path -Path $testPath -PathType Leaf) {

      $actualDistributionDir = $\_.Name

    }

  }

}

if (!$actualDistributionDir) {

  Write-Error "Could not find Maven distribution directory in extracted archive"

}

Write-Verbose "Found extracted Maven distribution directory: $actualDistributionDir"

Rename-Item -Path "$TMP\_DOWNLOAD\_DIR/$actualDistributionDir" -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"

**File name: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>3.5.5</version>

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

    </parent>

    <groupId>com.example</groupId>

    <artifactId>contactform</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>contactform</name>

    <description>Demo project for Spring Boot</description>

    <url/>

    <licenses>

        <license/>

    </licenses>

    <developers>

        <developer/>

    </developers>

    <scm>

        <connection/>

        <developerConnection/>

        <tag/>

        <url/>

    </scm>

    <properties>

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

    </properties>

    <dependencies>

    <!-- Web (already exists) -->

    <dependency>

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

        <artifactId>spring-boot-starter-web</artifactId>

    </dependency>

    <!-- DevTools (already exists) -->

    <dependency>

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

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

        <scope>runtime</scope>

        <optional>true</optional>

    </dependency>

    <!-- Testing (already exists) -->

    <dependency>

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

        <artifactId>spring-boot-starter-test</artifactId>

        <scope>test</scope>

    </dependency>

    <!-- JPA for database -->

    <dependency>

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

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

    </dependency>

    <!-- H2 Database (in-memory) -->

    <dependency>

        <groupId>com.h2database</groupId>

        <artifactId>h2</artifactId>

        <scope>runtime</scope>

    </dependency>

    <!-- Validation for email, name, message -->

    <dependency>

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

        <artifactId>spring-boot-starter-validation</artifactId>

    </dependency>

    <!-- Optional: Lombok to reduce boilerplate -->

    <dependency>

        <groupId>org.projectlombok</groupId>

        <artifactId>lombok</artifactId>

        <optional>true</optional>

    </dependency>

</dependencies>

    <build>

        <plugins>

            <plugin>

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

                <artifactId>spring-boot-maven-plugin</artifactId>

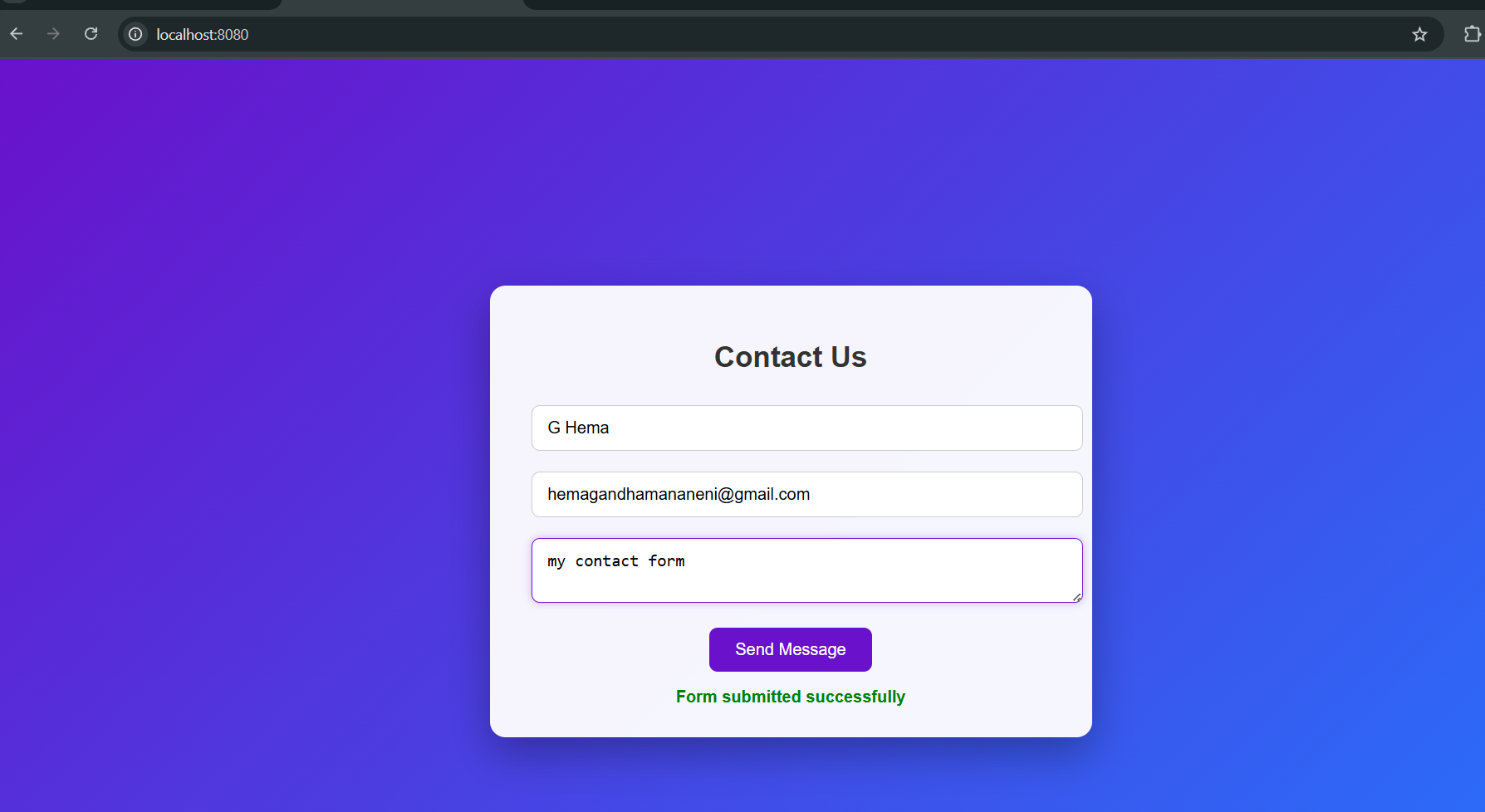
            </plugin>

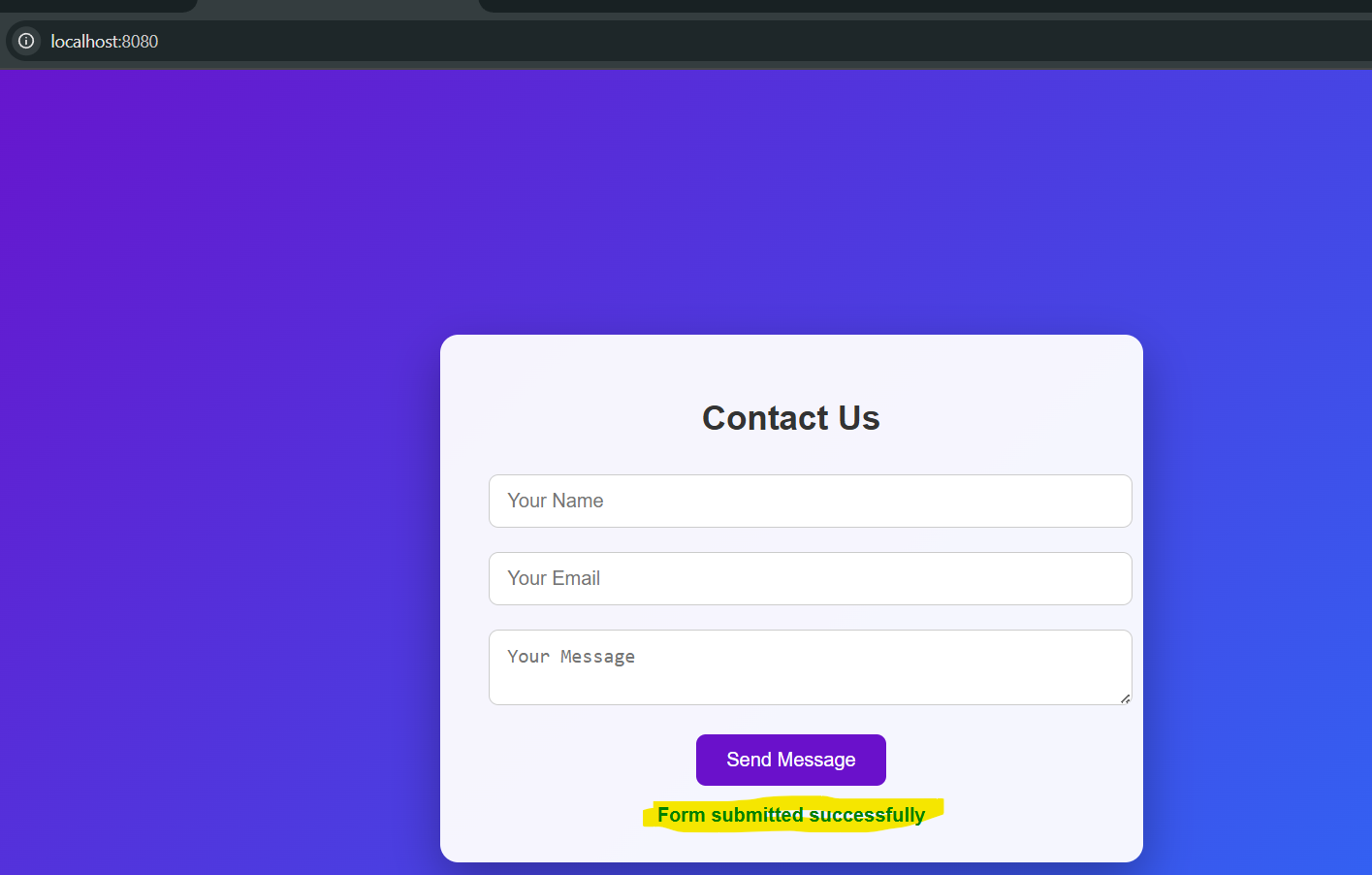
        </plugins>

    </build>

</project>

**Output:**

****

**After giving input  
**