/\*

\* 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();

}

}