maven installation

1.1 Large programs

We have already learned how to write small programs, so now we will learn how to write large ones. As you know, the larger and more complex the program, the more money is paid for its development :) And let's start with a little background ...

As programs grow in size, developers face two new challenges:

- A large number of people are working on the same program.
- There is no such person who would know the entire code of the program.

Very often, situations began to arise when a programmer fixed a bug in one place of the program and at the same time broke something in another. The release documentation even has this joke:

List of changes:

- Fixed old bugs :)
- Added new ones :(

Then they came up with two approaches to solving this problem: technical and managerial.

The technical approach was that programs were broken into parts: **libraries and modules** . Each such module was a small brick from which large projects were then built. Libraries are such universal components that can be used in different programs.

The managerial approach was even more interesting - they limited the number of people who could work on one project/library. Empirically, they even came up with a rule: the team should be so large that "it could be fed with two pizzas . " This usually means that if there are more than **8 people** working on a project , then it needs to be split into two projects.

It has become popular in the Java developer community to write libraries for all occasions and make them publicly available. Thus, Java programmers could not write the same code again (which was often raw and contained bugs), but use ready-made and **proven solutions** .

An additional incentive was the fact that the Java language gained great popularity when writing server-side solutions (it worked on the backend). First, server software has higher requirements for reliability, and using time-tested libraries is always preferable to writing your own code.

Secondly, servers have practically no limits on the size of the code. The developer of a mobile application tries to cram it into 10 megabytes, a desktop application - into 100 megabytes. And a Java backend developer can cram several tens of gigabytes of libraries into a project and no one will say a word to him :)

By the way, this is not a joke. You can easily come across a backend project with several dozen modules and a couple of hundred libraries. But it has become extremely difficult to describe (and change!) build scripts for such projects.

And then Maven appeared.

1.2 Introduction to Maven

Maven is a special "framework" for project build management. It standardizes 3 things:

- Description of the project;
- Project build scripts;
- Dependencies between libraries.

Maven's predecessor was **Ant** , and its successor is **Gradle** . But it was Maven that developed and perfected the three listed standards, and also regulated their interaction. It was he who brought the work of Java communities to a new level. Let's look at it in more detail.

Technically, Maven is a special program / service, the main purpose of which is **to manage the process of building** projects. It can be simply downloaded as an archive and unpacked to any directory. You don't need a special installer for this.

She does not have a graphical interface - **all commands are given to her using the console** . To make it even more comfortable to work with it, it is recommended to register special environment variables in your OS.

Maven also has a special **repository** (directory / folder) where it stores the libraries that it uses when building projects. You will need to select some folder on the disk and assign it as a repository.

Another interesting thing is the presence of a global Maven repository for all libraries, but we will talk about this a little later.

1.3 Download and install Maven

Maven has an official site maven.apache.org . There is a lot of documentation on the project, so if you have any difficulties or additional questions - come in, do not be shy.

Also on the downloads page (https://maven.apache.org/download.cgi) you can download the maven archive (apache-maven-3.8.5-bin.zip). The unpacked archive will take somewhere around 10 MB, although the local maven repository will eventually require several hundred megabytes of memory.

Maven is written in Java and requires a JRE of at least version 7, as well as defined JAVA_HOME environment variables.

Just create a folder for Maven on your computer, for example, d:\devtools , and extract the archive with Maven into it. As a result, you should get a folder like d:\devtools\maven\bin , where the main binaries of the project will be located.

1.4 Environment variables

After that, you need to add the path to the bin folder from the unpacked archive to the PATH environment variable.

To set the environment variable in Windows 10, you need to go to Control Panel - System - Advanced system settings. Then click "Environment Variables", find PATH and select "Edit", then add the path d:\devtools\maven\bin to the end of the line. Pay attention, the path must lead exactly to the bin folder.

On a Unix-based OS, the environment variable can be added with a console command:

```
export PATH=/opt/apache-maven-3.8.5/bin:$PATH
```

If you did everything correctly, then in the console you need to type the command: "mvn -v". In response, you will see something like:

```
C:\Users\Zapp>mvn -v
Apache Maven 3.0.5 (r01de14724cdef164cd33c7c8c2fe155faf9602da; 2013-02-19 15:51:28+0200)
Maven home: T:\apache-maven-3.0.5\bin\..
Java version: 1.8.0_65, vendor: Oracle Corporation
Java home: C:\Program Files\Java\jdk1.8.0_65\jre
Default locale: en_US, platform encoding: Cp1251
OS name: "windows 7", version: "6.1", arch: "amd64", family: "dos"
```

1.5 Local Maven Repository

You can also set a special folder where Maven will store jar libraries that it will use when building projects. This folder is called **the local maven repository** .

If no such folder is specified, Maven will create it in the current user's home directory. My directory is: C:\Users\Zapp\.m2

The folder has a rather specific name “.m2”. Although it does not scare Linux users - there it is a fairly common approach to naming various "repositories" and / or any other storage of service information.

Important! Do not place Maven in system folders, as it will need write permissions to these folders during operation, which may be of unhealthy interest to the antivirus or operating system.

Maven prior to version 3.5 required an environment variable called M2_HOME, but this is no longer necessary.

You can read more about configuring Maven at the link: https://maven.apache.org/configure.html