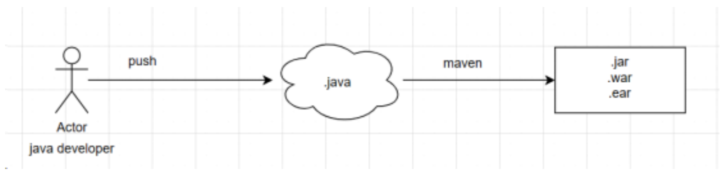
Maven-Day-1

21 November 2023 14:29

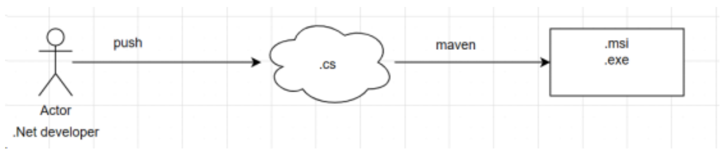
- Today our topic is on **BUILD TOOL**

- **Why do we need Build tool?**
  - Build tool used to convert the source code files into binary packages.
  - 

    Suppose you are a java developer, you write a code in .java files & push the code into GitHub. The source later converted into binaries packages like
    - .jar
    - .war
    - .ear

  - **Which build tool used to convert the .java source code files into .jar/.war/.ear binaries?**
    maven

  - These binary files only we deploy into our DEV/QA/PROD environments.

  - 

    Same way if you are .Net developer, you write code in .cs files & push the code to GithHub. These sources code will be converted into binaries like
    - .msi
    - .exe
    - .dll

  - **Which build tool used to convert the .cs source code files into .msi/.exe binaries?**
    msbuild

  - Like this different build tools based on the different technologies
    - Android --> Gradle
    - Nodejs --> npm

  - In this course we talk about maven as build tool since we have most of applications on Java

- **How to Install maven on Linux machine**

  - **Install java-11**
    - List java-11 related packages are available
      yum list | grep -i java-11*

    - Install the java-11
      yum install java-11* -y

    - Check the java version
      java -version

  - **Download the maven binary**
    - Go to maven official site & get latest maven installer link
      wget https://dlcdn.apache.org/maven/maven-3/3.9.5/binaries/apache-maven-3.9.5-bin.tar.gz

    - Extract the maven binary
      tar -xzvf apache-maven-3.9.5-bin.tar.gz

    - Move the extracted maven folder as /opt/maven
      mv apache-maven-3.9.5 /opt/maven

  - Set the environment variables **maven_home** & **path at user-level**
    - Update the **maven_home** in ~/.bashrc or ~/.bashprofile
      export maven_home=/opt/maven

    - Update the **PATH** variable
      export PATH=$PATH:$maven_home/bin

(OR)

- Set the environment variables **maven_home** & **path at system-level**
  - Update the **maven_home in /etc/profile**
    export maven_home=/opt/maven

  - Update the **PATH** variable
    export PATH=$PATH:$maven_home/bin

- **Check the installed maven version**
  mvn --version

► **How to Install maven on Windows machine - Home work**

- **Install jdk-20**
  Download: https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe
  Update: JAVA_HOME & PATH System variables

- Download & extract maven binary
  https://dlcdn.apache.org/maven/maven-3/3.9.5/binaries/apache-maven-3.9.5-bin.zip
  Extract to C:\maven

- Update the MAVEN_HOME & PATH variables at system level

► Maven is not only a **build tool** but it is also **project management tool**.
- What is project management tool?
  - It's used to create project structure.

  - Just check any java projects in GitHub, they proper directory structure.
    - Let me do search of java project in GitHub
    - If you see calculator repository, it's having proper directory structure src/main/com, src/main/test
    - If you see one more online book store repository, It's also having directory structure in proper way

  - To create this directory structure manually it's
    - Time taking process
    - Chances for not creating project correctly

  - This problems can be avoided with help maven.

- **Now who will use maven as project management tool?**
  DEVELOPER
  DEVELOPER - Will create project structure & will push into GitHub.

- So DEVELOPERS will use the maven as **project management tool (To create projects) & Build Tools(To compile the files & generate binaries locally)**

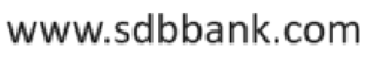
► Now let's work on maven as a project management tool practically
- Assume you are java developer & have requirement to create application for **sdbbank** in java.

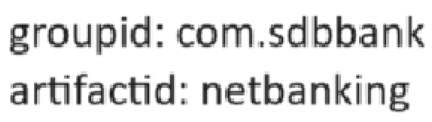
- DEVELOPER will create hotel-booking project structure using maven
  - open cmd & run the command **mvn archetype:generate**

  - It will display list of templates available & prompt for template number to enter
    - Each template will used for different purpose & creates different structure
    - Developers are responsible to choose correct number
    - As of now we go with default one & which is sample maven project.

  - Now the template also will different versions, because in background maven team continiously enhancing this templates. We will choose latest one.

  - **Groupid**: Normally sdbbank website url is like www.sdbbank.com
    In this the groupid is **com.sdbbank**
    Similarly for www.facebook.com the groupid is **com.facebook**

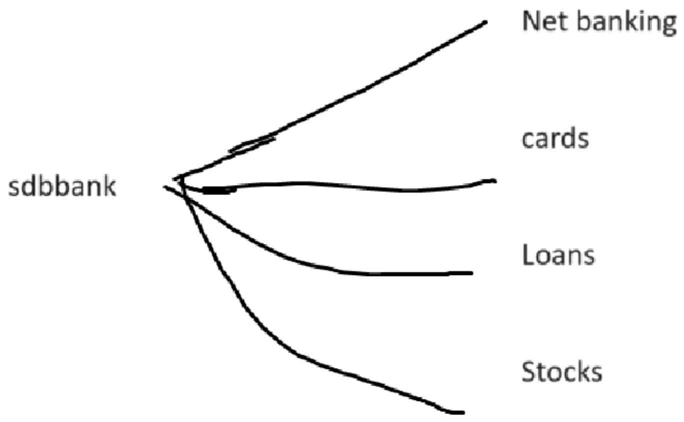

www.sdbbank.com


groupid: com.sdbbank
artifactid: netbanking

www.sdbbank.com

groupid: com.sdbbank
artifactid: netbanking

- **Artifactid**: What kind of operations banks will do?



  - ▢ Net banking
  - ▢ Cards
  - ▢ Loans
  - ▢ Stocks

  Out of these areas we can take anyone as an artifactId

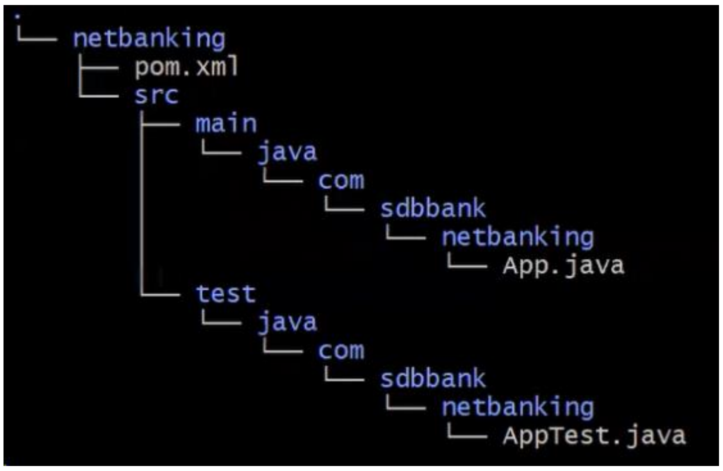
- **Version**: we can choose as snapshot since our project is on development phase.

- **Package** we can as groupid.artifactid

○ Now the project created with name netbanking.

○ Let's check the directory structure
- Install tree command & check directory structure
  yum install tree -y
  tree

- Here we can see

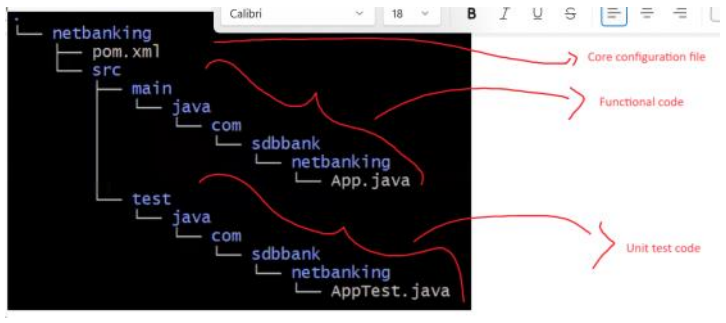


```
.
└── netbanking
    ├── pom.xml
    └── src
        ├── main
        │   └── java
        │       └── com
        │           └── sdbbank
        │               └── netbanking
        │                   └── App.java
        └── test
            └── java
                └── com
                    └── sdbbank
                        └── netbanking
                            └── AppTest.java
```

  - ▢ pom.xml
  - ▢ src/main/java/com/sdbbank/netbanking/App.java
  - ▢ src/test/java/com/sdbbank/netbanking/AppTest.java

- The files that are comes under the src/main --> Related to functional code
  - ▢ What is functional code?
    The code written to develop the net banking application.

- The files that comes under the src/test --> Test code in order to validate the functional code. This code we can call as Unit Test code.



```
└── netbanking
    ├── pom.xml
    └── src
        ├── main
        │   └── java
        │       └── com
        │           └── sdbbank
        │               └── netbanking
        │                   └── App.java
        └── test
            └── java
                └── com
                    └── sdbbank
                        └── netbanking
                            └── AppTest.java
```

  - Why here developers are writing test code? Normally QA has to do testing correct?
    Basic level of functional code can be validated instead of checking with QA engineers.

- **pom.xml** Core configuration file of the maven project & it's considered as heart of the java project.
  - groupId
  - artifactId
  - version

  - **properties:**
    - Properties section used to **define parameters & reuse those parameters throughout the pom.xml file**
      - **<java.version>1.7</java.version>**

    - <maven.compiler.source>${java.version}<\maven.compiler.source>
      <maven.compiler.target>${java.version}<\maven.compiler.target>

      - Normally we can install java7/java8/java11/java17 versions in same machine.

      - Whenever developer write java files we have to tell maven, based on what java version source code files are developed.

      - Here with these two parameters maven compiler will treat,
        - consider the source code is developed based on java7
        - Generated byte code byte code compatible with Java7

  - **Dependencies:**
    - **Dependencies are** external modules/libraries that you are project depends.
    - Here junit dependency used to run the unit test on functional code during build process.

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

    - We can get above snippets by looking the maven official sites.

  - **Plugins**
    - Plugins will provide additional functionalities to enable tasks compile the code, test, package & deploy as part of Maven build Lifecycle.

    - I will explain Maven build Lifecycle in sometime & later you will come to know these plugin needed.

  - Hope you are clear about maven project directory structure.