FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Car Fleet Management System

Java Application Developer’s Guide

2019
FIT CVUT

1.0 Setting up environment

1.1--Installing Java JDK and JRE

-Step 1: Add A Third Party PPA to Ubuntu

The easiest way to install Oracle Java JDK 8 on Ubuntu is via a third party PPA… To add that PPA, run the commands below

```
sudo add-apt-repository ppa:webupd8team/java
```

After running the commands above, you should see a prompt to accept the PPA key onto Ubuntu… accept and continue

-Step 2: Download Oracle Java 8 Installer

Now that the PPA repository has been added to Ubuntu, run the commands below to download Oracle Java 8 installer…. the installer should install the latest Java JDK 8 on your Ubuntu machines.

```
sudo apt update
```

```
sudo apt install oracle-java8-jdk oracle-java8-jre
```

When you run the commands above you'll be prompted to access the license terms of the software… accept and continue..

-Step 3: Configure Oracle JDK8 as Default

Set Oracle JDK8 as default, to do that, install the oracle-java8-set-default package. This will automatically set the JAVA env variable.

```
sudo apt install oracle-java8-set-default
```

The command above will automatically set Java 8 as the default… and that should complete your installation, you can check you java version by running following command.

```
javac -version
```

1.2--Installing Apache Maven

Apache Maven is a free and open source project management and comprehension tool used primarily for Java projects. Maven uses a Project Object Model (POM) which is essentially a XML file containing information about the project, configuration details, the project's dependencies, and so on.

-Step 1: Start by updating the package index

```
sudo apt update
```

-Step 2: Next, install Maven by typing the following command

```
sudo apt install maven
```

-Step 3: Verify the installation by running the mvn -version command

```
mvn -version
```

-Step 4: Setting up environment variables

we'll need to setup the environment variables. To do so open your text editor and create a new file named mavenenv.sh inside of the /etc/profile.d/ directory.

```
sudo nano /etc/profile.d/maven.sh
```

Paste the following configuration:

```
export JAVA_HOME=/usr/lib/jvm/default-java
export M2_HOME=/opt/maven
export MAVEN_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}
```

Save and close the file. This script will be sourced at shell startup.
Make the script executable by typing:

```
sudo chmod +x /etc/profile.d/maven.sh
```

Finally load the environment variables using the following command:

```
source /etc/profile.d/maven.sh
```

1.3--Installing MYSQL Server and Client

MySQL is still a popular database server… however, the default open source database server on the majority of Linux systems is MariaDB… Some webmasters still find MySQL very useful and are sticking with it..

-Step 1: Installing the server and client through apt

```
sudo apt-get update
```

```
sudo apt-get install mysql-server mysql-client
```

Doing the above will install MySQL from the repository with the latest packages for it.

After installing, you may want to run the commands below to secure MySQL.

```
sudo mysql_secure_installation
```

This will prompt you for the current root password you created when installing the software. You can follow the guide below to complete the setup.

```
Enter password for user root: ENTER ROOT PASSWORD
Press y|Y for Yes, any other key for No: n
Change the password for root? n
Remove anonymous users? y
Disallow root login remotely? y
```

Remove test database and access to it? y
Reload privilege tables now? Y

After that, you’re all done.

1.4--Installing GIT

Git is one of the most popular tools used for distributed version control system(VCS). Git is commonly used for source code management (SCM) and has become more used than old VCS systems like SVN.

-Step 1: Install Git

```
sudo apt-get update
sudo apt-get install git-core
```

-Step 2: Confirm Git the installation

With the main installation done, first check to ensure the executable file is setup and accessible. The best way to do this is simply to run git with the version command.

```
git --version
```

-Step 3: Configure Git’s settings

It’s a good idea to setup your user for CVUT FIT gitlab now, to prevent any commit errors later. We’ll setup the user testuser with the e-mail address testuser@fit.cvut.cz.

```
git config --global user.name "testuser"
git config --global user.email "testuser@example.com"
```

2.0 Setting up environment

2.1--Cloning the repository to your local machine.

The main repository is hosted on the gitlab of CVUT FIT accessible through the link

https://gitlab.fit.cvut.cz/rybolzde/cfs.git

To clone the repository to a local folder you should use this command

```
git clone https://gitlab.fit.cvut.cz/rybolzde/cfs.git
```

It will ask you for credentials then it will be cloned to your local disk.

2.2--Setting up the MYSQL database.

Start by running the MYSQL client

```
mysql -u root -p
```

It will ask you for the root password

Then create a database with the name of choice (fleet in our example)

```
create database fleet;
```

Exit the client by typing this command

```
exit;
```

2.3--Populating MYSQL database.

In the root folder of the repository you will find the file **init.sql** you need this file to build the schema of the database in the one you just created in the client

Use this command while being in the root folder

```
mysql -u root -p fleet < init.sql
```

2.3--Connecting the application with the MYSQL database.

Edit **application.properties** file with your favourite text editor and change the values to fit to your mysql database.

3.0 Running the Application

3.1--Running the application from the source code with maven.

Traverse to the root directory of the application where the pom.xml is, run this command to run the application

```
mvn sping-boot:run
```

Maven will start building the application now, after a few minutes the application will be running.

3.2--Accessing the application.

After building the application and running it you will be able to access it on port 8080

For example http://localhost:8080

3.3--Running the application in the background without nohup.

To run the application without stopping you can use the command below

```
nohup mvn spring-boot:run &
```

4.0 Running behind NGINX reverse proxy

4.1--Installing NGINX using apt.

```
sudo apt update
sudo apt install nginx
```

4.2--Configuring NGINX.

Now that the Spring application is running as a service, an NGINX proxy allows opening the application to an unprivileged port and setting up SSL.

Create an NGINX configuration for the reverse proxy:

```
server {
	listen 80;
	listen [::]:80;

	server_name example.com;

	location / {
		proxy_pass http://localhost:8080/;
		proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
		proxy_set_header X-Forwarded-Proto $scheme;
		proxy_set_header X-Forwarded-Port $server_port;
	}
}
```

4.3--Test NGINX configuration.

```
sudo nginx -t
```

4.4--restarting NGINX.

```
sudo service nginx restart
```

4.5--accessing application.

Now changes have taken effect to access it navigate to the public IP address of your server.

5.0 Unit testing with JUnit

There are testing classes included to test under “Test Packages”

To run the test use the command below

mvn clean test

The output should be similar to this if everything is ok

```
$ mvn clean test

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running net.petrikainulainen.junit5.JUnit5ExampleTest
This test method should be run
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.033 sec -
in net.petrikainulainen.junit5.JUnit5ExampleTest

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO]
------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO]
------------------------------------------------------------------------
```