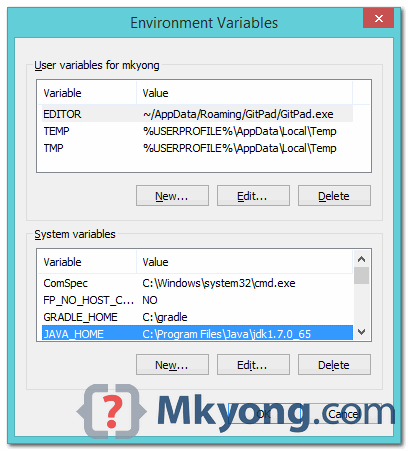
Install DockerToolbox-1.12.6.exe for Windows 8 not never version, otherwise “Docker Quickstart Terminal” gets certificate exceptions

Install Java JDK1.8

1. JDK and JAVA\_HOME

Make sure JDK is installed, and “**JAVA\_HOME**” variable is added as Windows environment variable.



Need to add Spring Tools to Eclipse:

Visit [Maven official website](http://maven.apache.org/download.cgi), download the Maven zip file, for example : apache-maven-3.2.2-bin.zip. Unzip it to the folder you want to install Maven.

Assume you unzip to this folder – C:\Program Files\Apache\maven

Install Maven in Centos:

<https://tecadmin.net/install-apache-maven-on-centos/>

Change settings.xml in /opt/maven/conf/

Add the following to settings.xml:

1. <proxies>
2. <proxy>
3. <id>myproxy</id>
4. <active>true</active>
5. <protocol>http</protocol>
6. <host>proxy.somewhere.com</host>
7. <port>8080</port>
8. <username>proxyuser</username>
9. <password>somepassword</password>
10. <nonProxyHosts>\*.google.com|ibiblio.org</nonProxyHosts>
11. </proxy>
12. </proxies>

Add the following to pom.xml

<repositories>

<repository>

<id>central</id>

<url>http://central.maven.org/maven2/</url>

</repository>

</repositories>

Run in centos: export MAVEN\_OPTS="-Djava.net.preferIPv4Stack=true"

In centos:

<https://github.com/chriscowley/chriscowley.me.uk/wiki/Installing-yum-repo-server-on-CentOS>

install docker centos:

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-centos-7>

install docker-machine in centos (to get docker ip):

<https://docs.docker.com/machine/install-machine/#installing-machine-directly>

Correct Docker Machine install:

curl -L "https://github.com/docker/machine/releases/download/v0.10.0/docker-machine-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-machine

create source folder in Centos:

cd ~

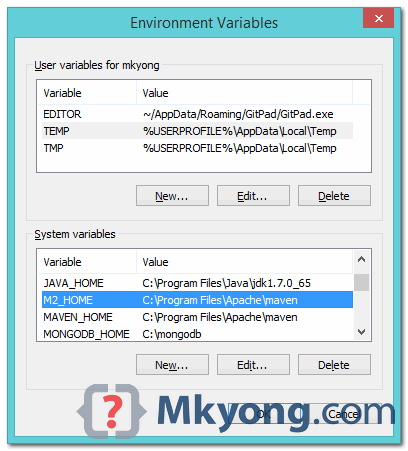
mkdir source

cd ~/source

mkdir directoryinfoservice

3. Add M2\_HOME and MAVEN\_HOME

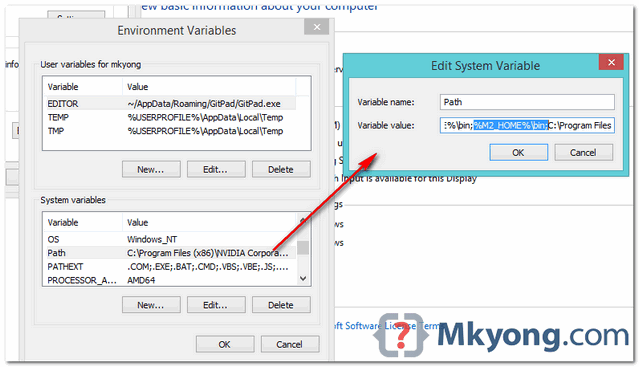
Add both M2\_HOME and MAVEN\_HOME variables in the Windows environment, and point it to your Maven folder.



**M2\_HOME or MAVEN\_HOME**  
Maven document said add M2\_HOME only, but some programs still reference Maven folder with MAVEN\_HOME, so, it’s safer to add both.

4. Add To PATH

Update PATH variable, append Maven bin folder – %M2\_HOME%\bin, so that you can run the Maven’s command everywhere.



5. Verification

Done, to verify it, run mvn –version in the command prompt.

C:\Users\mkyong>mvn -version

Apache Maven 3.2.2 (45f7c06d68e745d05611f7fd14efb6594181933e; 2014-06-17T21:51:42+08:00)

Maven home: C:\Program Files\Apache\maven

Java version: 1.7.0\_65, vendor: Oracle Corporation

Java home: C:\Program Files\Java\jdk1.7.0\_65\jre

Default locale: en\_US, platform encoding: Cp1252

OS name: "windows 8.1", version: "6.3", arch: "amd64", family: "windows"

C:\Users\mkyong>

View Video To Dockerize the Spring Boot Application:

<https://www.youtube.com/watch?v=Ojs9I-gnkc8>

Create new docker file in the root of the project called “Dockerfile” without an extension by just using the “Add file” functionality.

Add the following to the Dockerfile:

FROM openjdk:8

🡪 says which base image this docker imager will be created from

EXPOSE 8080

🡪says that the container listens on this port at runtime

ADD /target/directory-info-service.jar directory-info-service.jar

* ADD[“<src>”,”<dest>”]. Copies files URLs from <src> and adds them to the filesystem of the container at the path specified in <dest>
* This is done to copy the project’s compiled JAR file which mavern creates to the docker image

ENTRYPOINT ["java","-jar","directory-info-service.jar"]

🡪The ENTRYPOINT allows us to configure a container which will run as an executable

Install Mavern in windows:

<https://www.mkyong.com/maven/how-to-install-maven-in-windows/>

Then open “Docker Quickstart Terminal”

* Go to the source code directory (cd /c/Users/Guillaume/workspace/directory-info-service/)
* Run: mvn clean
  + This does a clean and downloads the required dependencies
* Run: mvn install
  + This installs the decencies for the project, builds the project and creates the directory-info-service.jar file which contains all **of the source code and references**
* Run: docker build -f <Dockerfile name> -t <Docker image name (has to be lowercase)> . (**notice the space dot at the end**)
* Thus run: docker build -f Dockerfile -t directoryinfoservice .
  + This builds the docker image
* Run: docker images
  + Gets a list of docker images
  + You should see the directoryinfoservice image in the list if it was successfully built
* Run: docker run -p <publish exposed port> <dockerimage>
* Thus Run: docker run -p 8080:8080 directoryinfoservice
  + This will run the docker image and expose port 8080 on the docker image as port 8080 to the “outside world”
* Press Ctrl + C (to allow you to type and see the input)
* Run: docker-machine ls
  + This will give you the IP address of the docker machine (192.168.99.100:2376 on my machine)
* Open a web browser and run: http://<docker machine ip>:8080/svc/v1/directoryinfo/getDirectoryInfo/?directory=C:/temp

**In Centos, run docker without detaching:**

docker run -p 8080:8080 -d springframeworkguru/masteringthymeleaf

**docker run -d centos tail -f /dev/null**

Instead of running with docker run -i -t image your-command, using -d is recommended because you can run your container with just one command and you don’t need to detach terminal of container by hitting Ctrl + P + Q.

However, there is a problem with -d option. **Your container immediately stops unless the commands are not running on foreground**.  
Docker requires your command to keep running in the foreground. Otherwise, it thinks that your applications stops and shutdown the container.

The problem is that some application does not run in the foreground. How can we make it easier?

In this situation, you can add tail -f /dev/null to your command.  
By doing this, even if your main command runs in the background, your container doesn’t stop because tail is keep running in the foreground.

**How to call Directory Info Service:**

**Enable Curl:**

From linux:

Curl -X GET http://<Docker Image IP Address>:<Docker Image Port>/svc/v1/directoryinfo/getDirectoryInfo/?directory=/tmp

From Windows:

1. Open a web browser
2. Go URL: http://<Docker Image IP Address>:<Docker Image Port>/svc/v1/directoryinfo/getDirectoryInfo/?directory=/tmp

For Linux Build And Run:

1. Enable wget if not enable/installed:
   1. yum install wget -y
2. Install JDK 1.8 (**This should only be done if Maven/mvn cannot run because JDK 1.8 is missing**):
   1. mkdir /opt/java && cd /opt/java
   2. For 32-bit Systems
      1. cd /opt/java
      2. wget --no-cookies --no-check-certificate --header "Cookie: gpw\_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u45-b14/jdk-8u45-linux-i586.tar.gz"
   3. For 64-bit Systems
      1. cd /opt/java
      2. wget --no-cookies --no-check-certificate --header "Cookie: gpw\_e24=http%3A%2F%2Fwww.oracle.com%2F; oraclelicense=accept-securebackup-cookie" <http://download.oracle.com/otn-pub/java/jdk/8u45-b14/jdk-8u45-linux-x64.tar.gz>
      3. Once file has been downloaded, you may extract the tarball using tar command as shown below:
         1. # tar -zxvf jdk-8u45-linux-i586.tar.gz [For 32-bit Systems]
         2. # tar -zxvf jdk-8u45-linux-x64.tar.gz [For 64-bit Systems]
   4. Next, move to the extracted directory and use command update-alternatives to tell system where java and its executables are installed.
      1. # cd jdk1.8.0\_45/
      2. # update-alternatives --install /usr/bin/java java /opt/java/jdk1.8.0\_45/bin/java 100
      3. # update-alternatives --config java
   5. Tell system to update javac alternatives as:
      1. # update-alternatives --install /usr/bin/javac javac /opt/java/jdk1.8.0\_45/bin/javac 100
      2. # update-alternatives --config javac
   6. Similarly, update jar alternatives as:
      1. # update-alternatives --install /usr/bin/jar jar /opt/java/jdk1.8.0\_45/bin/jar 100
      2. # update-alternatives --config jar
   7. Setting up Java Environment Variables.
      1. # export JAVA\_HOME=/opt/java/jdk1.8.0\_45/
      2. # export JRE\_HOME=/opt/java/jdk1.8.0\_45/jre
      3. # export PATH=$PATH:/opt/java/jdk1.8.0\_45/bin:/opt/java/jdk1.8.0\_45/jre/bin
   8. Now You may verify the Java version again, to confirm.
      1. # java –version
3. Install Apache Maven:
   1. You can add maven to the yum libraries like this:
      1. wget http://repos.fedorapeople.org/repos/dchen/apache-maven/epel-apache-maven.repo -O /etc/yum.repos.d/epel-apache-maven.repo
   2. Now you can install maven like this:
      1. yum install apache-maven
   3. Check version:
      1. mvn –version
4. Install Docker:
   1. Update the package database:
      1. sudo yum check-update
   2. Add the official Docker repository, download the latest version of Docker, and install it:
      1. curl -fsSL https://get.docker.com/ | sh
   3. Start the docker service:
      1. sudo service docker start
   4. After installation has completed, start the Docker daemon:
      1. sudo systemctl start docker
   5. Verify that it's running:
      1. sudo systemctl status docker
   6. Lastly, make sure it starts at every server reboot:
      1. sudo systemctl enable docker
5. Copy the source code to the centos machine:
   1. Install Git:
      1. sudo yum install git
   2. create a folder for the source code:
      1. cd ~
      2. mkdir /sourcecode
   3. go the the sourcecode directory:
      1. cd /sourcecode
   4. get the source code from git:
      1. The first time (Clone):
         1. git clone <https://github.com/Evolve-IT/directory-info-service.git>
      2. If it has already been cloned (For updates from git), do a pull:
         1. git pull <https://github.com/Evolve-IT/directory-info-service.git> master
6. Build the project using Maven:
   1. go to the directory-info-service folder:
      1. cd /sourcecode/directory-info-service/directory-info-service
   2. Run: mvn clean
      1. This does a clean and downloads the required dependencies
   3. Run: mvn install
      1. This installs the decencies for the project, builds the project and creates the directory-info-service.jar file which contains all of the source code and references
7. Build and run the docker image:
   1. Run: docker build -f <Dockerfile name> -t <Docker image name (has to be lowercase)> . (**notice the space dot at the end**)
      1. Thus run: docker build -f Dockerfile -t directoryinfoservice .
      2. This builds the docker image
   2. Run: docker images
      1. Gets a list of docker images
      2. You should see the directoryinfoservice image in the list if it was successfully built
   3. Run: docker run -p <publish exposed port> -i -t <dockerimage>
      1. Thus Run: docker run -p 8080:8080 -i -t directoryinfoservice
      2. This will run the docker image and expose port 8080 on the docker image as port 8080 to the “outside world”
   4. Press Ctrl + P +Ctrl + Q (to allow you to type and see the input if it is still attached to the docker container)
8. Test the directoryinfoservice running in the docker container:
   1. Get the ip address of the docker container:
      1. Run: docker ps
         1. This will show a list of docker containers
         2. Use the CONTAINER\_ID which is displayed of the directoryinfoservice to get the ip address below.
      2. $ docker inspect --format '{{ .NetworkSettings.IPAddress }}' $(docker ps -q)
         1. This should display the docker container’s IP Address
         2. If this doesn’t work, use one of the commands on this page: <http://networkstatic.net/10-examples-of-how-to-get-docker-container-ip-address/>
   2. If Curl is not enabled:
      1. $ yum install curl
   3. Send a request to the directoryinfoservice:
      1. For Json Output:
         1. $ curl -X GET http://<Docker Image IP Address>:<Docker Image Port>/svc/v1/directoryinfo/getDirectoryInfoJson/?directory=/tmp
         2. You should see the json returned by the rest service
         3. An object with the following structure will be returned
      2. For Xml Output:
         1. $ curl -X GET http://<Docker Image IP Address>:<Docker Image Port>/svc/v1/directoryinfo/getDirectoryInfoXml/?directory=/tmp
         2. You should see the xml returned by the rest service
      3. The data returned by both methods will be in the following format: