Frontend

To install the frontend, generate the war of “lsim-frontend” project using maven (`mvn clean package`). The war will be generated on target folder.

There's one file that have to be edited on this war, located at “/WEB-INF/classes/conf/config.properties” . This file defines the propety `tmp_folder` that must point to a folder on the server where temporary files will be stored, the property `data_folder` that must point to a folder on the server where the register and the results will be stored and the property `cron`, which can be true to clean the tmp folder periodically or false otherwise. It is important that both folders (tmp and data) have the enough permissions. There's also “jdbc.properties” that contains the configuration of the database if we want to track statistics of usage. The tables needed are described in the file “createdb.sql” of the “lsim-project”.

The “index.html” file needed to acces the frontend is located in the project “lsim-web-frontend”. This project can also generate a war executing `mvn clean package.`

When we have both wars, we only have to deploy them to the webapps folder of the tomcat / jetty.

The last step to configure the frontend is to copy “configAPI.properties” file from “lsim-frontend/src/main/resources” to tomcat root folder. This file contains the codes api configuration:

```
rmiport=9010
host=compute-0-1
name=CoDeS
```

Dispatchers and resource notifier

The dispatcher and resource notifier can be generated as an executable jar containing all libraries. On “lsim-resource-pool” folder execute `mvn clean install`. On “lsim-dispatcher” folder and “lsim-resource-notifier” execute `mvn assembly:assembly`. This will generate a jar in both target folders named something like -with-dependencies.jar.

Copy both jars to the destination folder where you want to have the dispatcher. In this folder we also need the following files, which have to be edited:

1. config.properties: Configuration of resource notifier
```
# broker host
broker_address=compute-0-1
# broker port
broker_port=55007
# refresh interval of nodes file in seconds
interval=30
# reliabilty of this node
reliability=50
#port used to restart dispatcher
restart_port=29002
#dispatcher jar name (default: lsim-dispatcher.jar)
dispatcher_jar=lsim-dispatcher.jar
```

2. configDispRun.properites: Configuration of the dispatcher
```
# max number of applications
numApp=10
# dispatcher listening port
port=29001
# Class user to retrieve files (change it only if you know what are you
doing)
Storage=dispatcher.storage.StorageRemoteFiles
# temporary folder where instances will execute
folder=/state/partition1/mpeerez/lsim/applications
# ports to be used for each instance
appPorts=29003-29012
```

3. startDispatcher.sh: script to start the dispatcher (located in “lsim-resource-notifier”)
```
#!/bin/bash
# only need to specify resource-notifier jar, dispatcher is executed as
a process.
# remember to configure dispatcher_jar property of config.properties
file
# restartToken is the string used to restart the dispatcher using the
remote restart tool
java -jar asd-resource-notifier-1.0-dep.jar restartToken &>
resourcenotifier.log &
```

Resource pool

The project that contain the resource pool is “lsim-resource-pool”. Generate the jar with `mvn clean install` and then copy it to the resource pool folder.

Copy the “lsim-codes-xx.jar” (located in “lsim-codes/target”) to the resource pool folder. Moreover, copy the “codes-1.0M.jar” (you can find it in “lsim-libraries.zip” of the LSim tutorial) in a “lib” folder that you have to create on the resource pool folder.

Configuration files needed, which have to be edited:

1. config.properties (located in “lsim-resource-pool”)
   ```
   #filepath where resources will be stored
   nodes_file=nodes.txt
   # listening port
   port=55007
   # host where resource pool is installed and listening
   host=sd.uoc.edu
   # refresh interval of nodes file in seconds
   interval=30
   ```

2. configAPI.properties: codes configuration (located in “lsim-codes”)
   ```
   rmiport=9010
   host=compute-0-1
   name=CoDeS
   ```

3. startCodes.sh (it doesn't exist, it has to be created)
   ```
   java -cp lib/*:asd-resource-pool-0.0.1-SNAPSHOT.jar
   edu.uoc.lsim.asd.pool.ResourcePoolManagement &>resourcepool.log &
   java -cp lib/*:lsim-codes-0.1.4.jar
   distributed.codes.DistributedNodesRun &>error.log &
   ```