Continuous Integration

I.   Information

Source of the Java example available at http://www.javaworld.com/javaworld/jw-12-2005/jw-1205-maven.html .
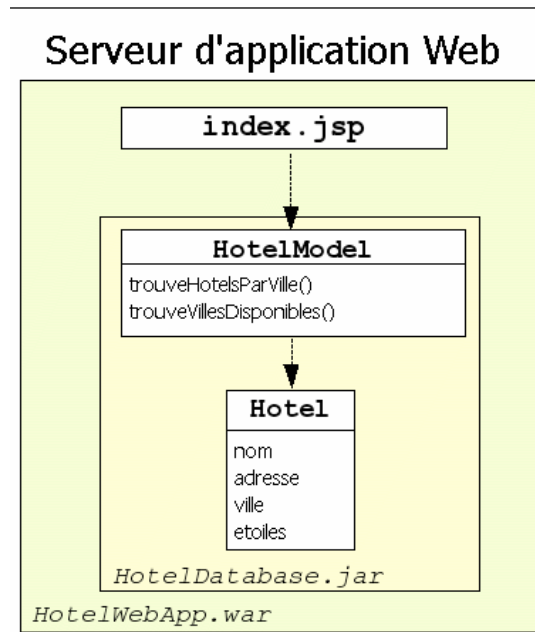
This example uses Maven as build tool.

To run the example, go into the HotelWebApp module and type :

```
$ mvn jetty:run
```

The web application will be deployed on http://localhost:8080

II.   Set up the CI infrastructure

- Create a VM on Windows Azure
    - choose the Ubuntu 14.04 template
    - DNS name: name that will be used to reach your Jenkins instance.
        ex: http://toto.cloudapp.net
    - Once the VM created, add a rule to open the firewall
        from 80 (HTTP) to 8080 (default port used by Jenkins).
- Install Jenkins (see https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu)

```
wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-
key add -
sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install openjdk-7-jdk jenkins
```

Check that Jenkins is running by browsing: http://xxx.cloudapp.net

III. Configure Jenkins

You can configure Jenkins by following the link in the left menu. Choose "Configure the system".

A. Global security configuration

- Prevent anonymous actions on your Jenkins
  - Manage users
    - create an *admin* account
  - Enable security
    - Jenkins' own user database (disable "Allow users to sign up")
    - Logged-in users can do anything (ensure an admin user is created user before applying this option)
- disable builds on master
  - Manage Jenkins/ Manage nodes/master/configure :
    number of executors = 0

B. Maven configuration

We will use the automated installer even if you already have Maven installed!
It will show you that jenkins is able to install libraries on remote machines you will use to run jobs.

C. Email server (SMTP)

The SMTP server should be set to xxxx. Test the service.

IV. Add a jenkins plugin

A great thing with Jenkins is that it is easily expandable. So, you can easily write a Jenkins plugin. That's why there are more than 900 plugins available!

A. Green bullets

Simple (and intuitive) things are often the best. A common semantic is to use the green color when all is fine and the red color when it goes wrong.

B. Git plugin

Jenkins does not support git by default ☹. We need to install the git plugin to enable it.

C. Docker plugin

Install Docker and configure it with Jenkins:
https://iww.inria.fr/tech-zone/using-docker-to-run-jenkins-jobs/

V. Create your own repository

In order to simulate developments made by a team, you will fork an existing repository on GitHub (demarey/HotelApp) .

On your computer, you will get a working copy from the forked repository to be able to work on the Hotels code.

```
$ git clone git@github.com:yourGitHubAccount/HotelApp.git hotels
```

VI.   Creating a new job

Create a new job (link in the left menu) named Hotel. Choose the maven build type.
Write a small description of the job and set the number of builds to keep to 2.

You now need to define where to find the code (the forked GitHub repository).
WARNING : to avoid to declare SSH keys, use the https protocol for the project URL.
A good strategy for code retrieving is to emulate a clean checkout to reduce the
bandwidth consumption.

Continuous Integration is useful if tests are run after each code integration (a commit on
the source code repository). Simply, there are 2 ways to obtain this result:
   • Poll the SCM (for example every 5 minutes). But this approach implies an
     overload of the SCM server and is not very efficient.
   • The best way is to define a post-hook commit that will warn Jenkins of a new
     commit and trigger a new build.
We will see later how to set up a post-commit hook. At this time, don't configure any
build trigger.

Endly, configure the build notification : your email adress (should be the developers
mailing-list in a « real » situation).

VII.   Run a job

Your job is now configured! Let us go to see if it works!

Simply click on the   icon beside the job on the dashboard.
Then, to visualize the result or the current state of a job, open the console associated to
this build: Click on the job, then on the build number you want to visualize, endly on the

console link   in the left menu.
Check the log output and catch following steps:
   • Automated Maven installation by Jenkins (for the first run)
   • Code checkout
   • Maven modules discovery
   • Build, tests
If the build succeeds, you will see a blue bullet on the dashboard. Why blue ? I don't have
the answer. We will address this issue later.

VIII.   Post-commit hook

In order to trigger the build automatically after each commit, we need to setup a post
commit hook.
Go to GitHub and select your project. Then click on Settings (right menu), Webhooks &
Services (left menu) , click on 'Add service' and select 'Jenkins (Git plugin)'.
Use the following URL:
`http://xxx.cloudapp.net`
WARNING : Don't forget to activate the polling (without Schedule) on the Jenkins job
configuration.

IX. Some development

A. Add a new Hotel

In the HotelModel class, add a new hotel named « Hotel Cigogne », located at « Grand Place », with an empty town and two stars.

```
new Hotel("Hotel Cigogne","Grand place","",2)
```

You are in a hurry and you don't want to loose time: you commit without test it before! Bad idea ☹, but let us see.

```
$ git commit —m « adding a new hotel »
$ git push
```

Now check if a build is triggered on the Jenkins server. What is the result?

B. Fix a problem

As you can see, there is a problem in a test.
Find the test in error from the Jenkins server.

X. Code coverage

Install the Cobertura plugin from Jenkins.
Set the job build goal to :

```
cobertura:cobertura
```

Activate Cobertura coverage report publication in the job configuration and set the *cobertura xml report pattern* to:

```
**/target/site/cobertura/coverage.xml
```


XI. C++ build with Jenkins

A. CMake plugin installation

Install the CMake plugin from Jenkins.

B. Configure a new job

We will use an already defined C++ project.
Create a new job (free-syle project) named cpp-project.
Configure the source code repository to:

```
svn://scm.gforge.inria.fr/svnroot/ecoleadt11/trunk/practicalClass/tests/C++
```

Then, add a CMake build step. Finally, trigger a build manually and check the output.