

Getting started

with



<http://clif.objectweb.org/>

Copyright © 2006-2007 France Telecom

License information : <http://creativecommons.org/licenses/by-nc-sa/3.0>

Table of contents

1. Introduction to ISAC-Plug ins.....	3
1.1. What is an ISAC-Plug ins.....	3
1.2. Technical requirements.....	3
1.3. Ready to use distributions.....	3
1.4. Installation.....	4
2. Define an Ldap injector writing an isac plug-ins.....	8
2.1. Ldap directory overview.....	8
2.2. Write a LDAP Injector Isac Plug-ins.....	9
2.2.1. <i>Create an Isac Plug-ins project</i>	10
2.2.2. <i>Write your Isac-plugins</i>	14
3. ISAC is a Scenario Architecture for CLIF.....	32
3.1. Define an ISAC scenario for Ldap.....	32
3.2. Record your ISAC scenario.....	33
4. 4. Define your test plan.....	49
4.1. Description of the context.....	49
4.2. Create your test plan.....	49
4.3. Add Probes and Injectors to your test plan.....	50
4.4. Deploying and executing your test plan.....	53
5. Clif server.....	56
5.1. Requirements.....	56
5.2. Rationale.....	56
5.3. Running a Registry.....	56
5.4. Configure a clif server.....	58
5.5. Running a clif server.....	59

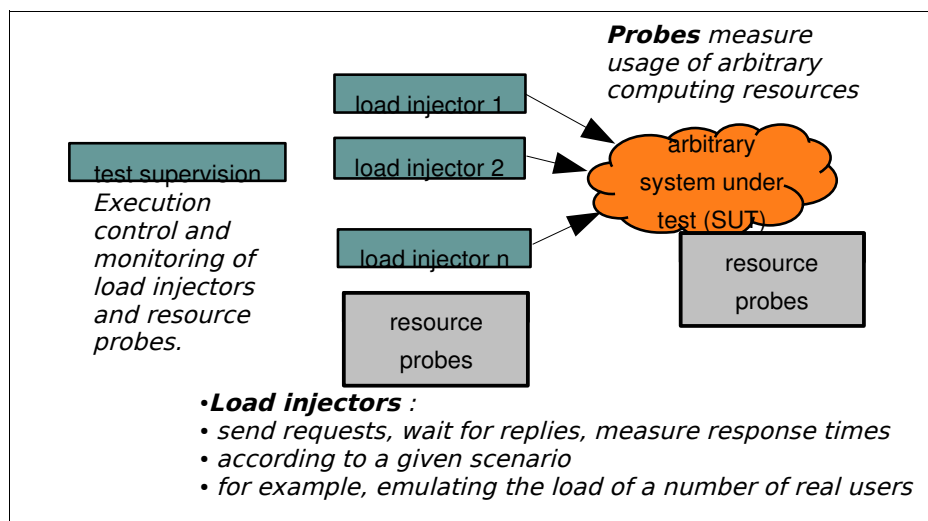
1. Introduction to CLIF

1.1. What is CLIF

CLIF is a component-oriented software framework written in Java, designed for load testing purposes of any kind of target system. By load testing, we mean generating traffic on a System Under Test in order to measure its performance, typically in terms of request response time or throughput, and assess its scalability and limits, while observing the computing resources usage.

Basically, CLIF offers the following features:

- deployment, remote control and monitoring of distributed load injectors;
- deployment, remote control and monitoring of distributed probes;
- final collection of measurements produced by these distributed probes and load injectors.



Analysis tools for these measurements will be provided as soon as possible. For the time being, all measurements are available as CSV (comma separated values)-formatted text files.

Thanks to its component-based framework approach, CLIF is easily customizable and extensible to particular needs, for example, in terms of specific injectors and probes, definition of load generation scenarios, storage of measurements, user (tester) skills, integration to a test management platform, etc. For instance, user interfaces are available as command-line tools, Java Swing-based GUI and Eclipse-based GUI.

1.2. Technical requirements

The CLIF framework and provided load injectors are 100% Java™. CLIF requires a Java runtime environment (JRE) or development kit (JDK), and the Java-based ant utility from Apache.org. The Current CLIF version is known to be working with :

–CLIF user manual and programmer's guide

- Sun JDK 5.0 (also known as J2SDK™ 1.5)
Download from http://java.sun.com/javase/downloads/index_jdk5.jsp
- Apache ant utility version 1.5.4 or greater
download from <http://ant.apache.org/bindownload.cgi>
Make sure ant is using the right JDK!
- Linux 2.4 and 2.6 kernels
- Mac OS X tiger
- Microsoft Windows XP™

System probes for Linux are also 100% Java, while system probes for Windows and Mac OS X are native (C code embedded in Java code via the Java Native Interface).

Since CLIF is written in Java, the only constraint about the SUT is that it must be reachable from a Java Virtual Machine (JVM), either directly or indirectly through some wrapping, gateway or native library.

There are two ways of getting a CLIF runtime environment: either by getting the whole source from the CVS repository, or by getting a ready-to-use binary distribution.

1.3. Clif installation

CLIF's site at OW2 Forge offers several binary distributions, available as zip files (see http://forge.objectweb.org/project/showfiles.php?group_id=57):

- clif
full runtime environment with a Java Swing based GUI and support for CLIF servers.
- server
reduced runtime environment just for running CLIF servers.
- console-Linux
Eclipse-RCP based standalone console for Linux/Intel.
- console-Windows
Eclipse-RCP based standalone console for Windows/Intel.
- console-Macosx
Eclipse-RCP based standalone console for Windows/Intel.
- clif-plugin
CLIF console as an Eclipse plug-in. Refer to section for plug-in installation.
- isac-plugin
ISAC editor as an Eclipse plug-in (requires CLIF console plug-in). Refer to section for plug-in installation.

In this tutorial we will only use the Eclipse RCP console (console-Windows, console-Linux, console-MacOSX).

Download the console archive file depending of your OS :

http://forge.objectweb.org/project/showfiles.php?group_id=57.

console-Linux

<u>1.2.2</u>		2007-08-09		
clif-console-1.2.2-Linux-i386.zip	60,043.9	i386	.zip	

console-MacOSX

<u>1.2.2ppc</u>		2007-08-09		
clif-console-1.2.2-macosx-ppc.zip	59,712.7	PPC	.zip	

console-Windows

<u>1.2.2</u>		2007-08-09		
clif-console-1.2.2-windows-x86.zip	60,300.3	i386	.zip	

Once downloaded you may unzip the archive file wherever you like. Avoid installing a distribution in a directory containing a whitespace character in its path (set-up problems have been reported in some conditions).

To make CLIF work you have to install apache, JDK 5.0 and define path for ANT_HOME and JAVA_HOME (refer to Appendix A to do it).

Once you did, you are able to launch your CLIF console.

Go to the directory where you unzip your archive file and launch the CLIF console :

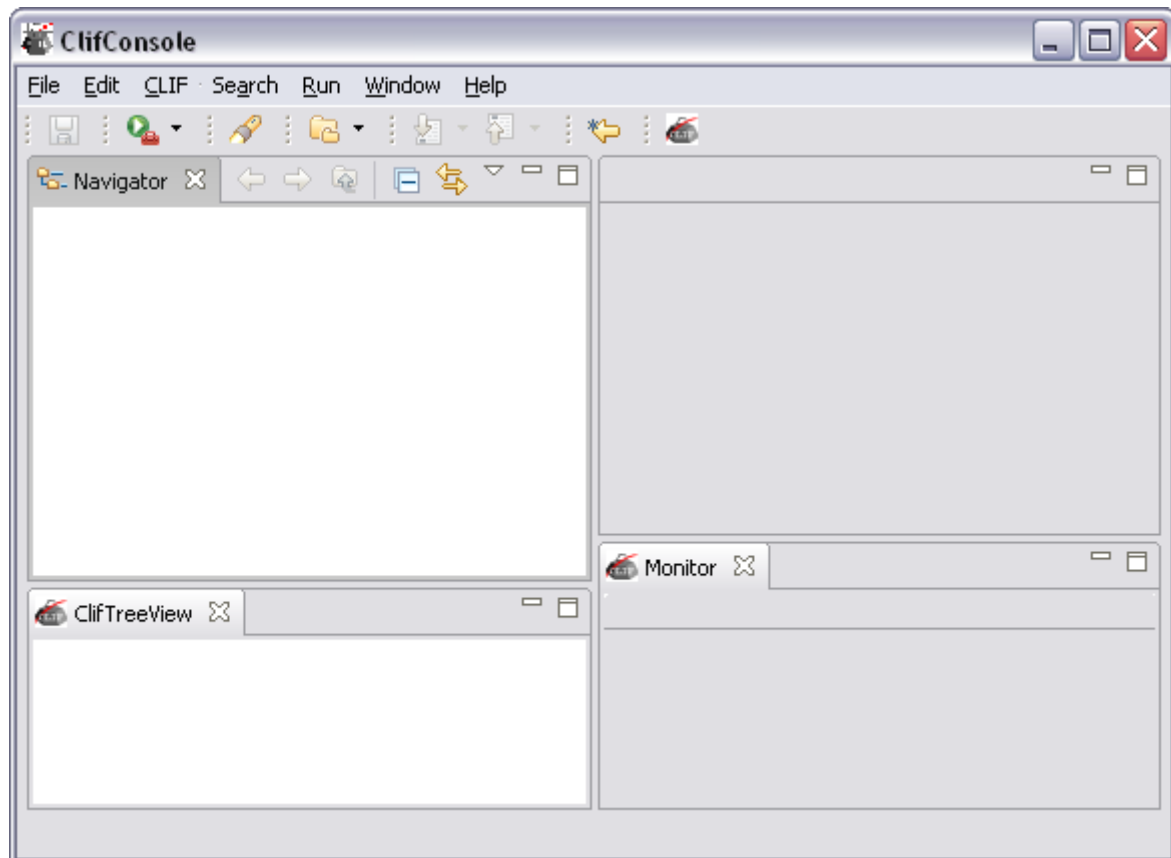
On windows double click on the clif-console.exe file

On Linux and Mac OS X go to the directory where you unzip the archive file and enter the command line : `./clif-console`

```
myLinux@user :~$ cd clif-1.3.0-console/
myLinux@user :~/clif-1.3.0-console$ ./clif-console █
```

You will see the next screenshot :

–CLIF user manual and programmer's guide



2. Define your System under Test (SUT)

Now the next step is to define what is the system that we want to test, what kind of measure we need to validate the load behavior.

2.1. Hello world web server

To be able to inject http request on an Hello world web server you will have to create a simple web server. See the Appendix A – Hello world web server to create your web server.

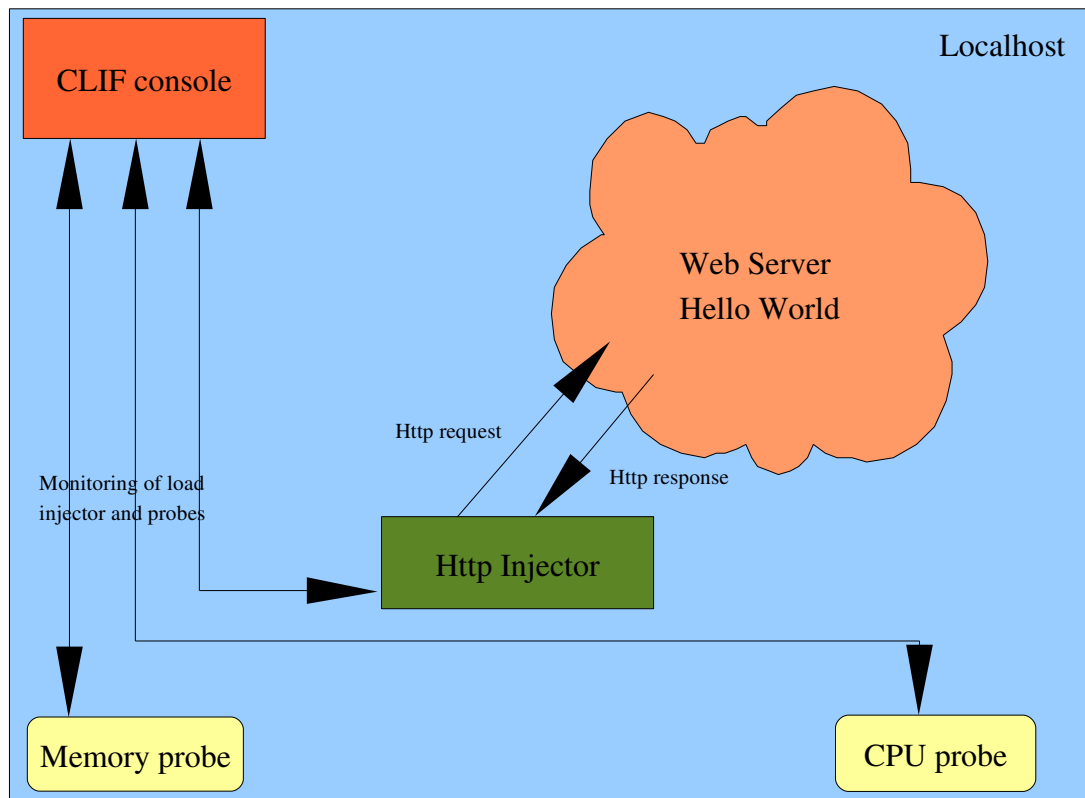
2.2. Define your load test plan

To be able to test the web server we need :

- a cpu probe which gets cpu usage of the web server
- a memory probe which gets memory usage of the web server
- a CLIF injector which injects requests on the web server

In this tutorial, the web server, the load injector and the probes will be on your local host, But it is possible to deploy load injectors and probes on remote host and to monitor them with your local host using the CLIF console.

Here you can see the schema of our ... :



–CLIF user manual and programmer's guide

In this load test we want to determine the maximum of user that can be connected to the web server at the same time.

So to do it we will define a scenario that all virtual users will follow.

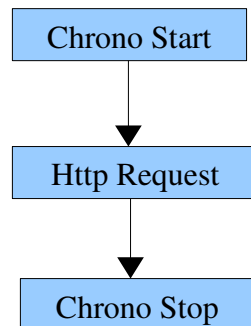
It is a simple scenario, a virtual user will connect to the web server and then disconnect from it.

A scenario will take 1 second.

To rise the requests load we will define a load profile which will create virtual users which will execute the scenario Simultaneously.

So if we have 50 virtual users at the time we can say that the web server treat 50 request by seconds.

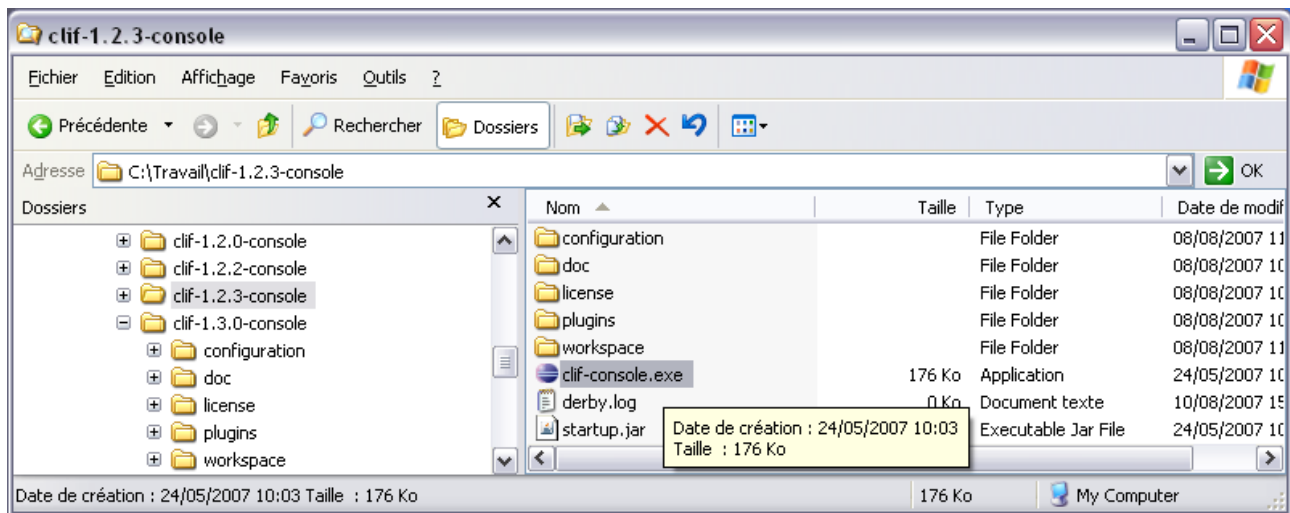
Scenario :



3. Define your test plan, scenario and load profile

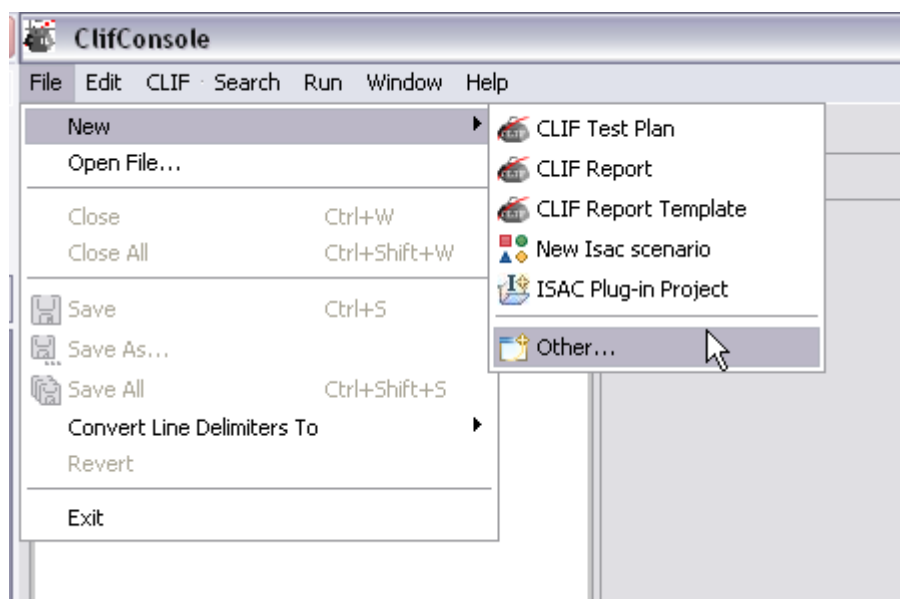
3.1. Create a project

First of all launch your CLIF console clicking on clif-console (.exe for windows) file.



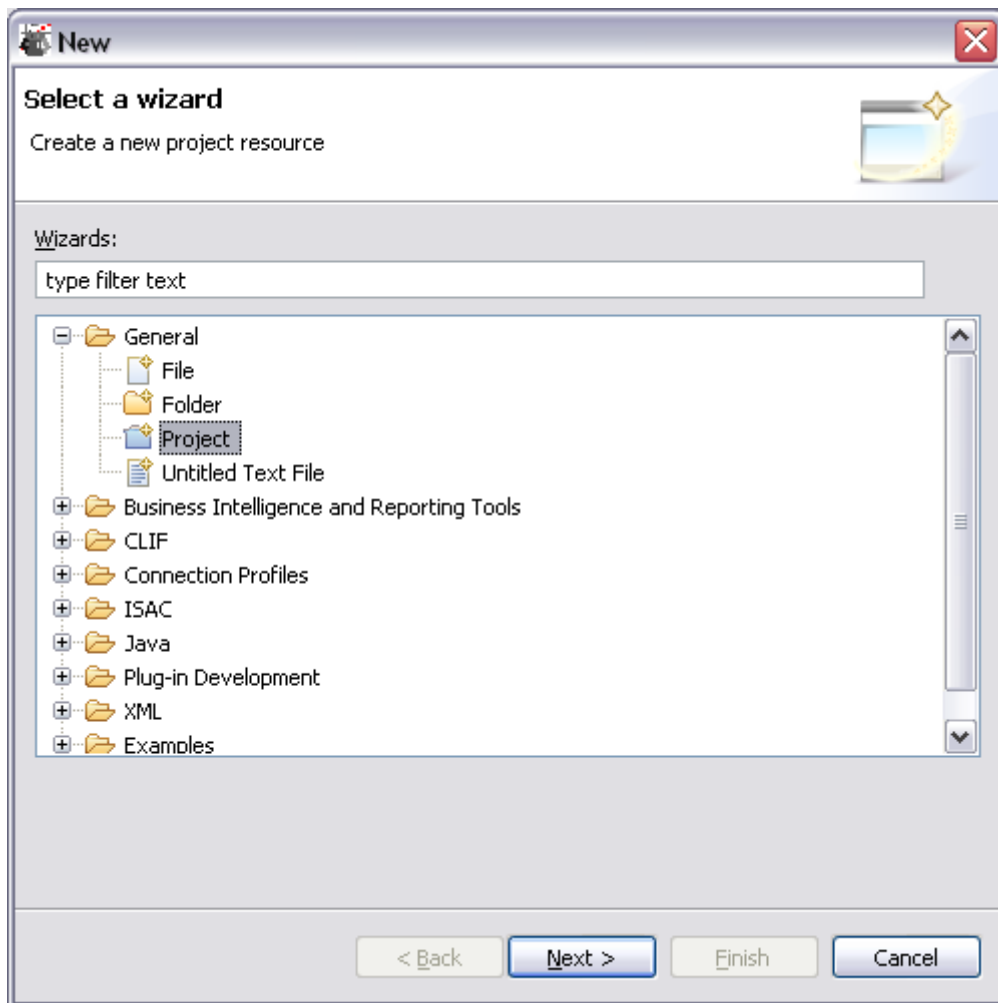
Once your CLIF console launched you have to create a project.

Click on File -> New -> Other...



–CLIF user manual and programmer's guide

Then the following window appears :

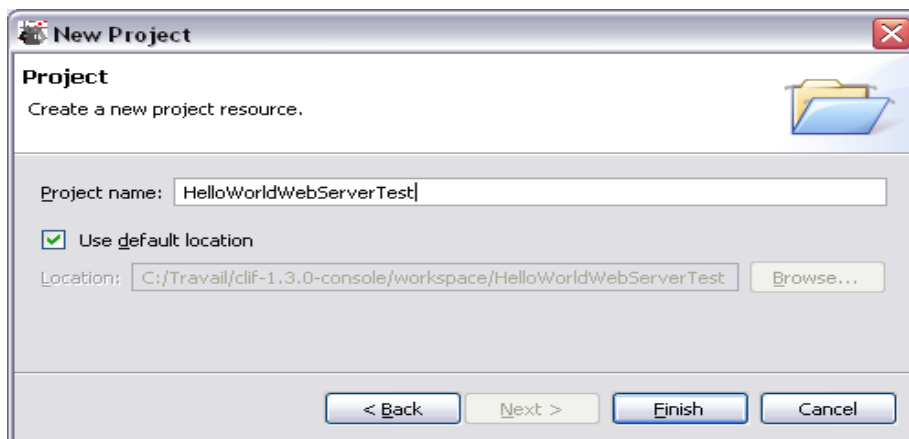



Choose General repertory and Project.

Click on the next button

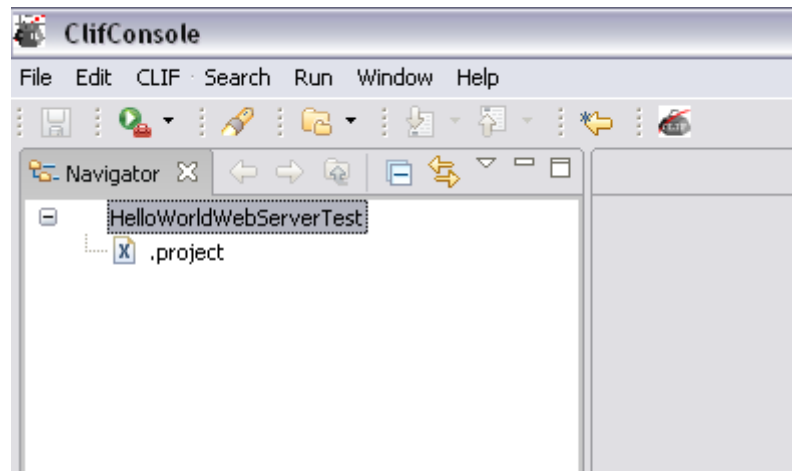


Now set your project name and the location of your project if you don't want to use the default one :



Then click on the finish button : 

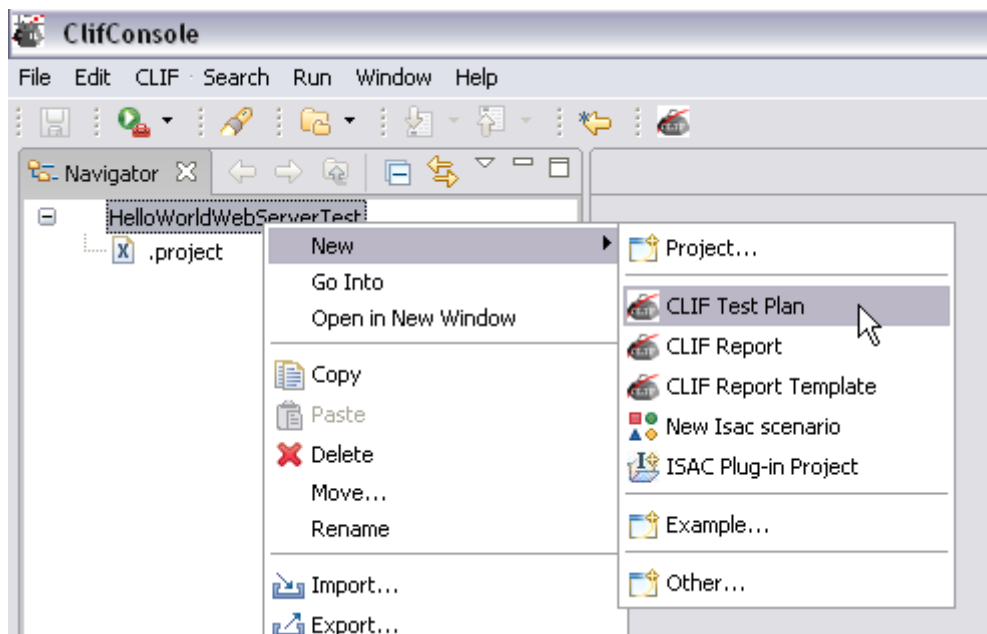
Now you can see your project in the CLIF console :



3.2. Test plan

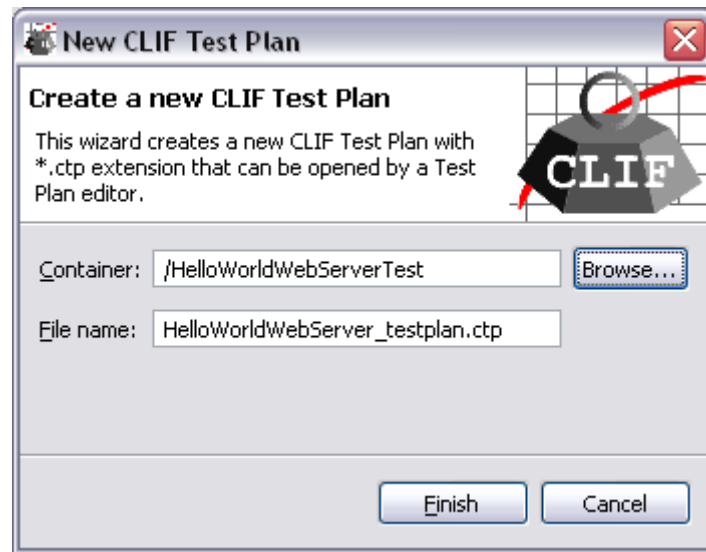
To create your test plan you have to do a right click on your project name

Then choose New -> CLIF Test Plan



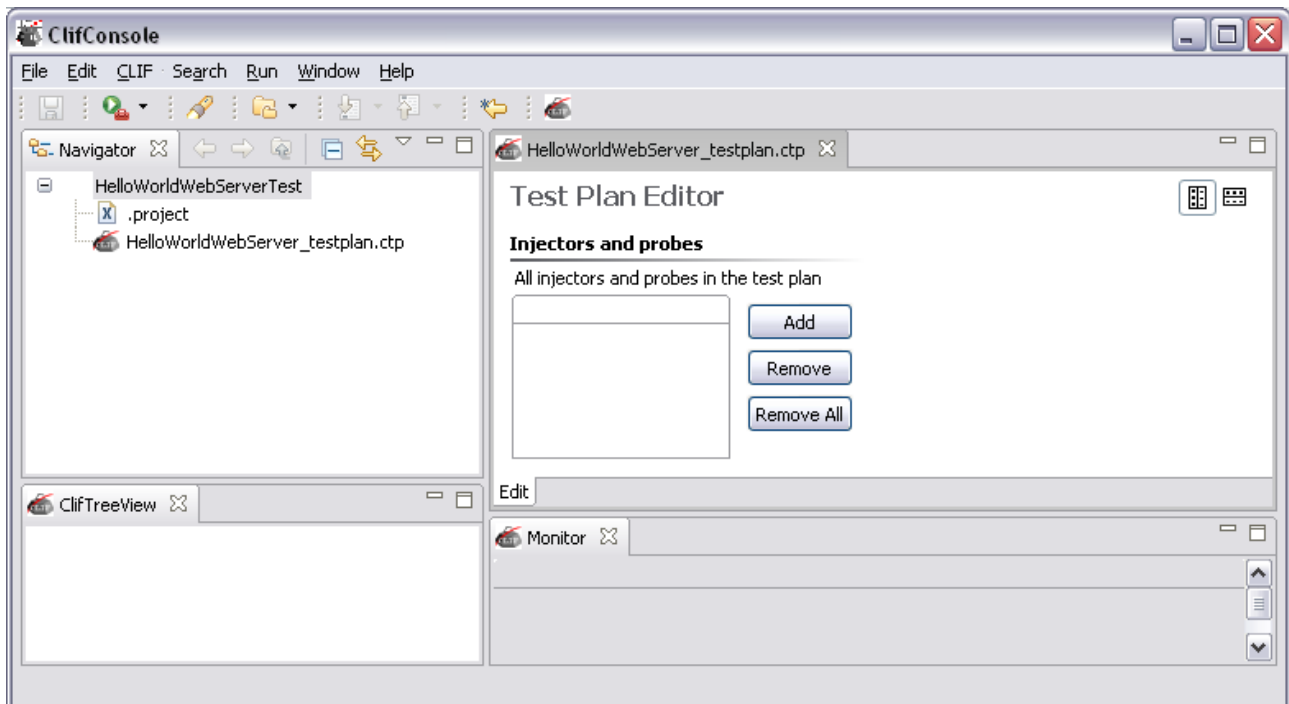
Now you have to choose the project container (let the default one) and your test plan name :

–CLIF user manual and programmer's guide




Once you did it, you can click on the finish button

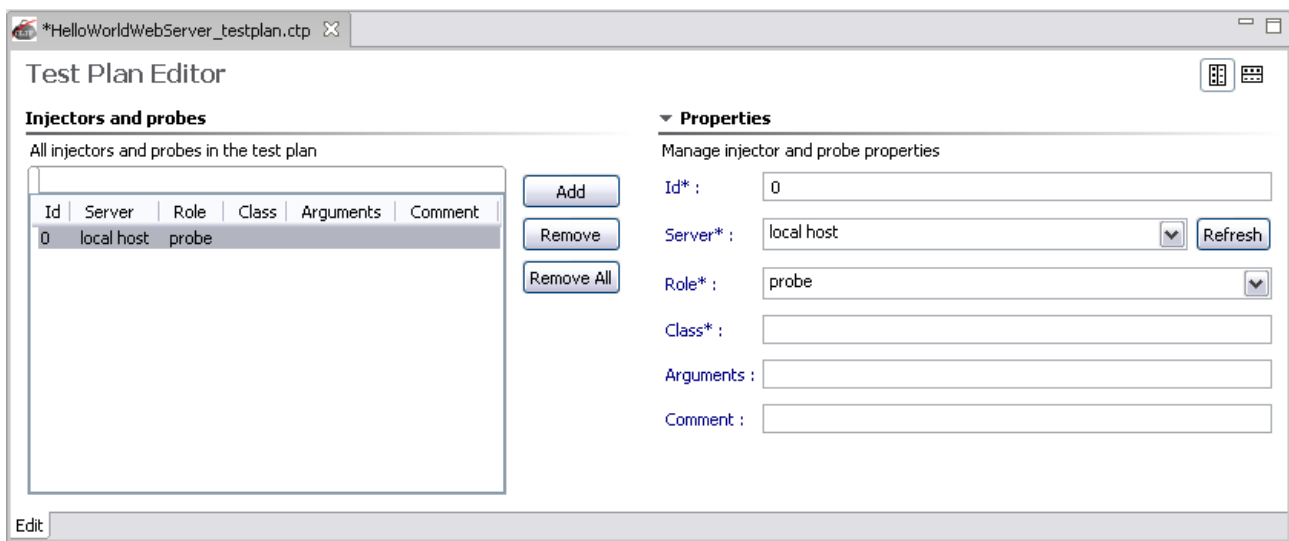
Then your testplan file (HelloWorldWebServer_testplan.ctp) and the test plan editor appears in the clif-console, respectively in the navigator a the left of the console and in a new tabulation at the right of the console.



Now we have to define the injectors and the probes that we want to use in the load test of our web server.

To add a probe or an injector click on the add  button

Now you can set some parameters which will define you probes or injectors :



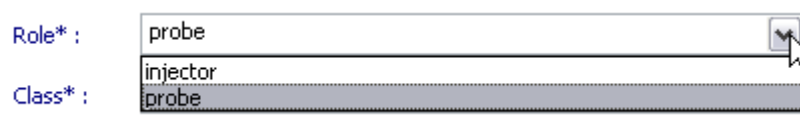
You can choose to change the default value of the id but be careful with it, because the id must be unique.

As we want to deploy probes and injectors on our local host we let the default value of the server name.

Then we have to choose the kind of "Role" we want to add.

We have the choice between :

- probe
- injector



The value of the Class text field will depends of the Role that we have chosen before.

- If you chose injector the Class value is : IsacRunner
- If you chose probe you can choose several values depending of the kind of probe you want to deploy :
 - cpu
 - memory
 - jvm

In the argument text field you have to set the arguments that the probes or the injectors need.

For the injectors it is the name of the isac scenario (filename.xis).

For the probes (cpu, memory, jvm) it is first the polling period (ms) and then the execution duration.

Finally you can add a comment for each probe or injector added.

Notice that for each type of probes and for injectors you have a different tabulation.

So concerning the load test of our web server we will have :

- For the injector :

memory	cpu	injector				
Id	Server	Role	Class	Arguments	Comment	
0	local host	injector	IsacRunner	HelloWorldWebServerTest_scenario.xis	Injecteur	

- For the cpu probe :

Injectors and probes

All injectors and probes in the test plan


injector	cpu	memory				
Id	Server	Role	Class	Arguments	Comment	
1	local host	probe	cpu	1000 600	Cpu probe	

- For the memory probe :

Injectors and probes

All injectors and probes in the test plan

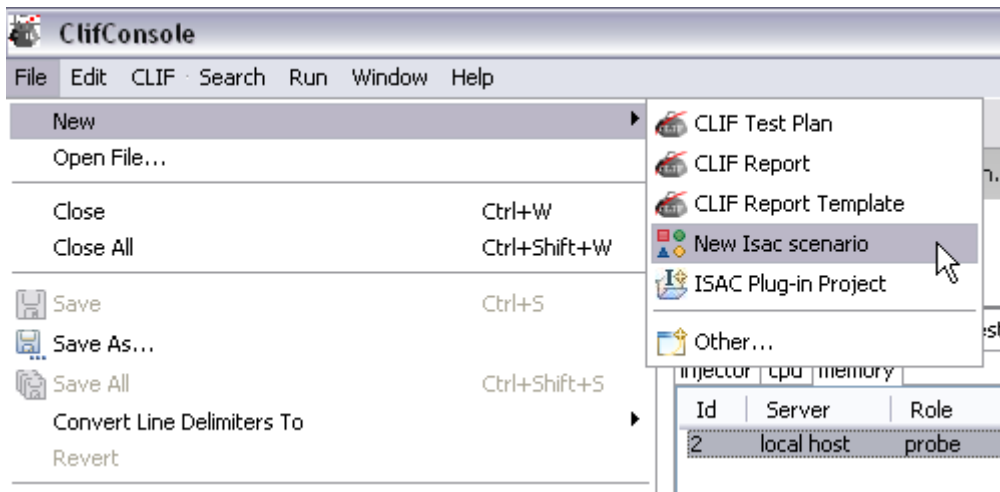
injector	cpu	memory				
Id	Server	Role	Class	Arguments	Comment	
2	local host	probe	memory	1000 600	Memory probe	

Don't forget to save your work (Ctrl+s or )

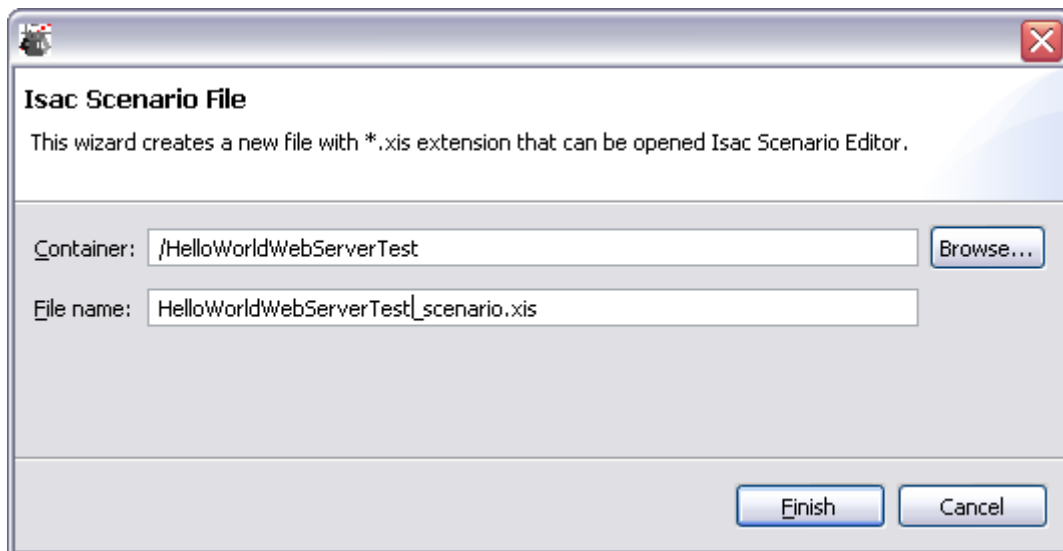
3.3. Scenario

To create your scenario, you should use the ISAC scenario wizard.

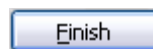
Click on File -> New -> New Isac Scenario



Now you have to set the container (let the default one) and the scenario name :

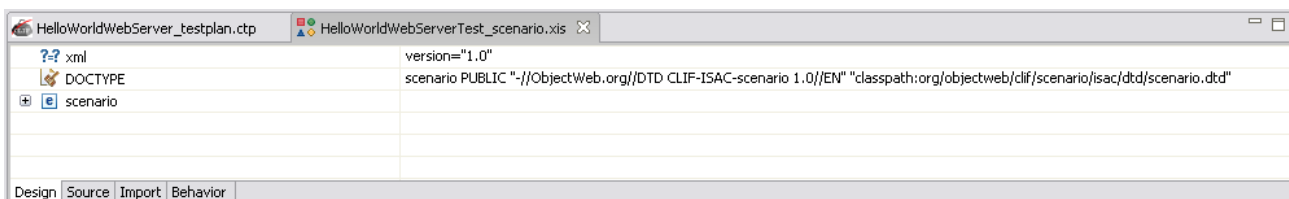
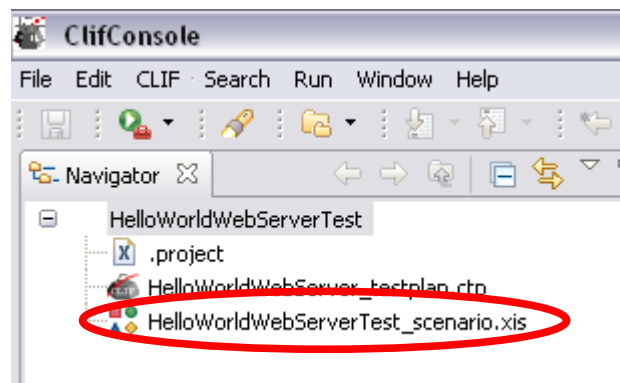


Then you can click on the finish button



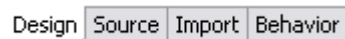
In the navigator at the left of your CLIF console you can see now your scenario file :

–CLIF user manual and programmer's guide



At the right of your CLIF console the content of scenario file appears.

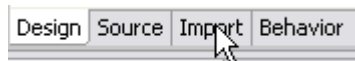
In the left bottom corner of this view which displayed the file content there are some tabulations.



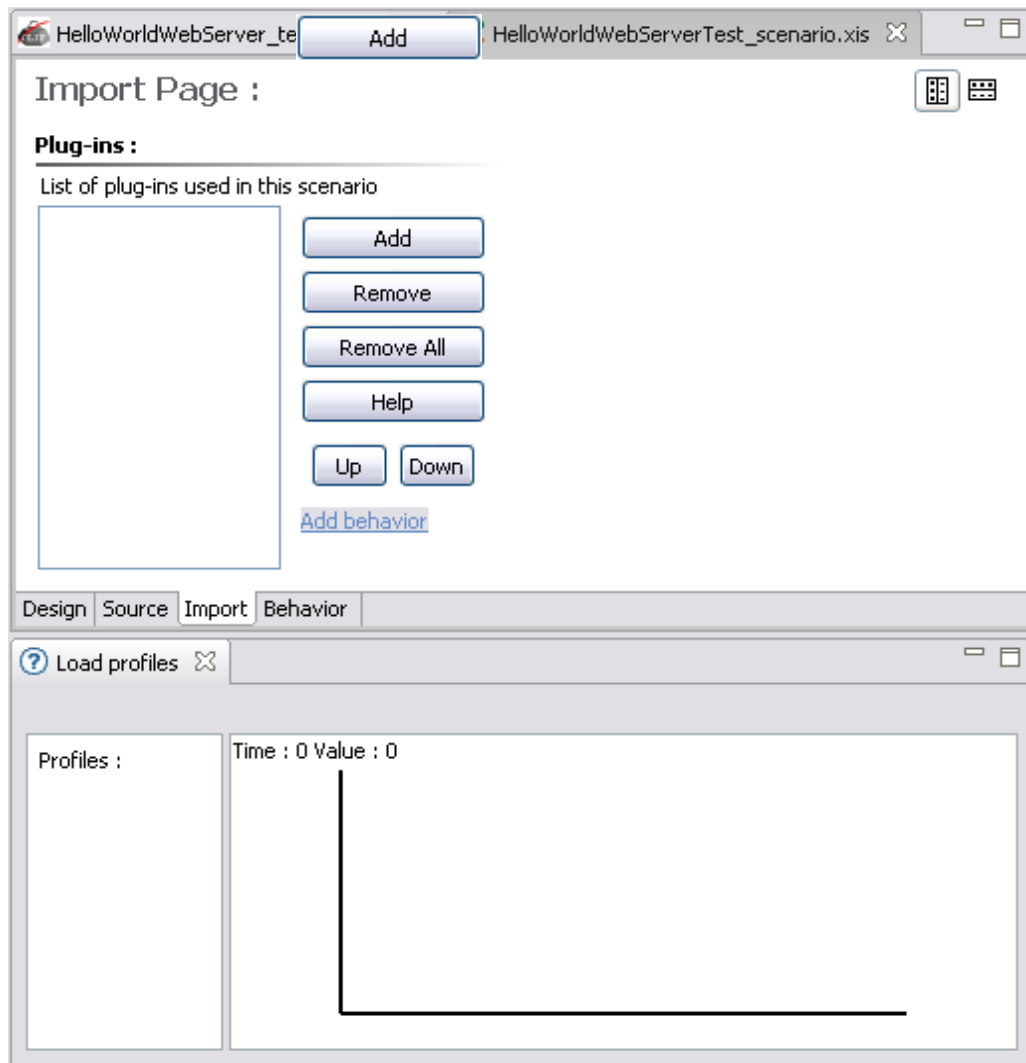
To define your scenario we need to use two existing ISAC plug-in :

- HttpInjector_1.0 : this plug-in provides methods to send http request
- ConstantTimer : this plug-in provides methods to fix the scenario duration

Now click on the Import tabulation

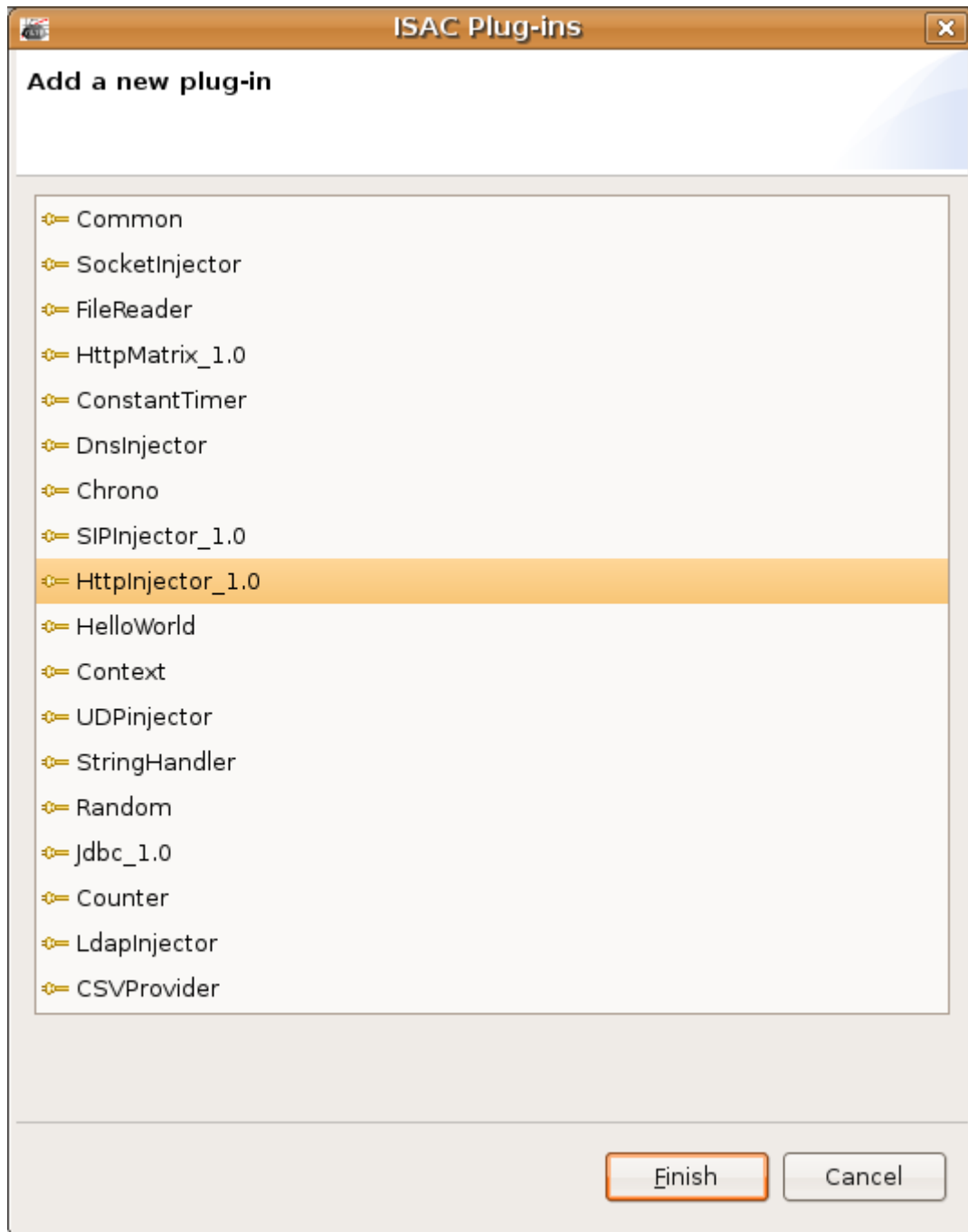


The following view appears :

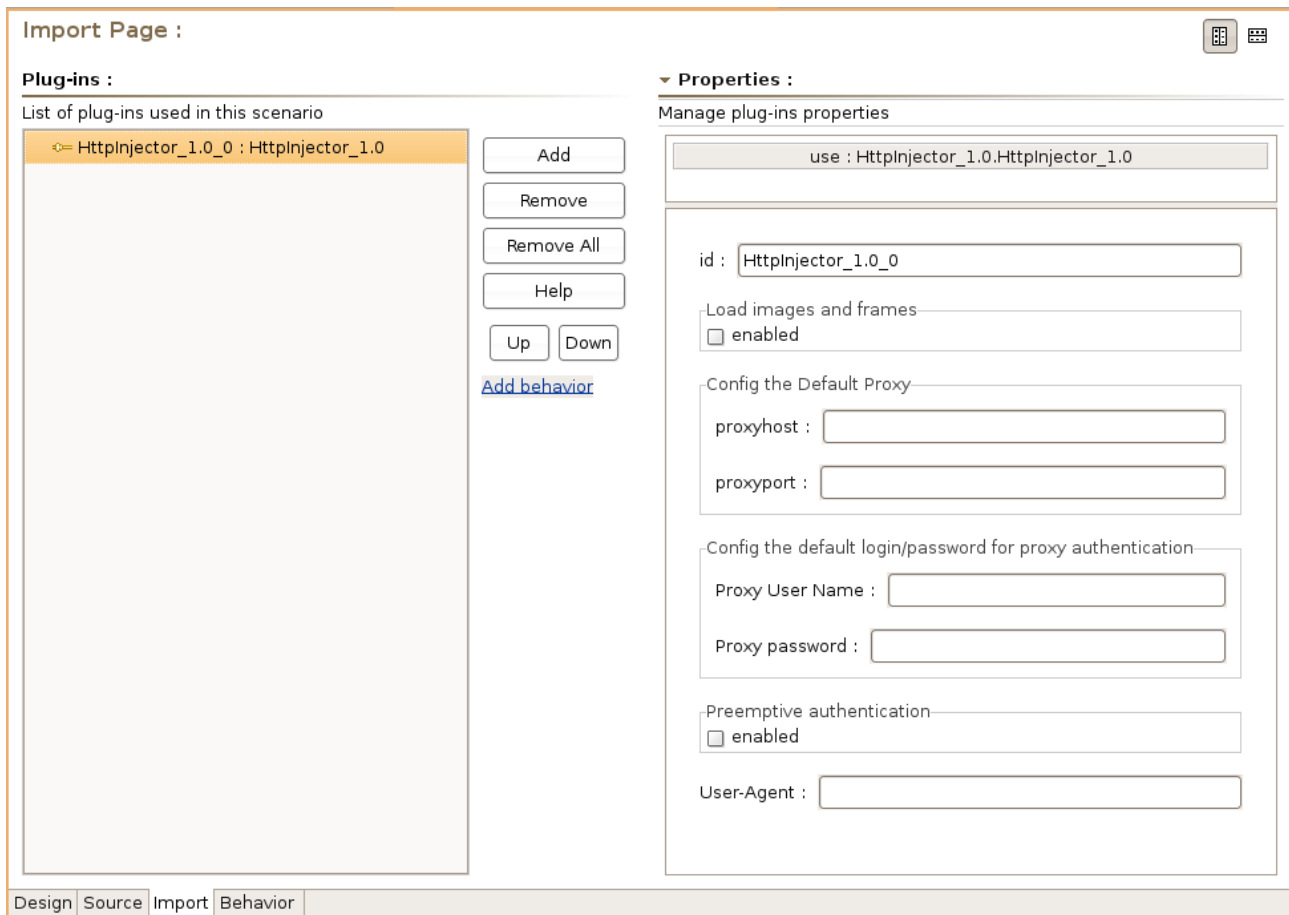


Now click on the add button to import the needed plug-ins.

Select the HttpInjector_1.0 plug-in and click on the finish button



Now you can see the list of the imported plug-ins with your HttpInjector_1.0 plug-in and if you click on it, its properties appear in the right side of the view.

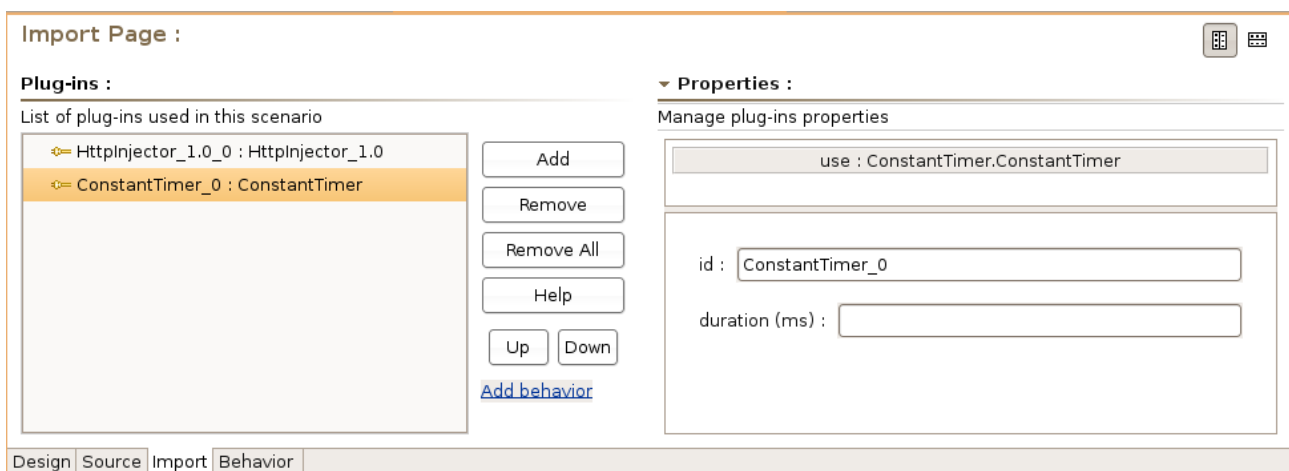


Now add the HttpInjector_1.0 plugin :

For this plug-in you have some parameters to set if you want to use it and to send requests on your web server.

As our web server is running on our local host we don't need to set these properties but if you test web server on your network you may need to set some of these plug-in properties.

And now you can import the last plug-in (ConstantTimer) :



To be able to manage the virtual user simultaneous requesting the web server, the duration properties will be set to 1000 ms (1 second), Indeed, the constant timer will add a sleep time period after sending the request. This sleep time will be variable it will depend on the duration of the request action but the total duration of the scenario will be 1000 ms.

▼ Properties :

Manage plug-ins properties

use : ConstantTimer.ConstantTimer

id : ConstantTimer_0
duration (ms) : 1000

From now all the needed plug-ins are imported :

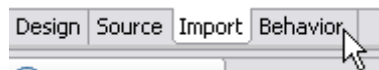
Plug-ins :

List of plug-ins used in this scenario

⚙️ HttpInjector_1.0_0 : HttpInjector_1.0
⚙️ ConstantTimer_0 : ConstantTimer

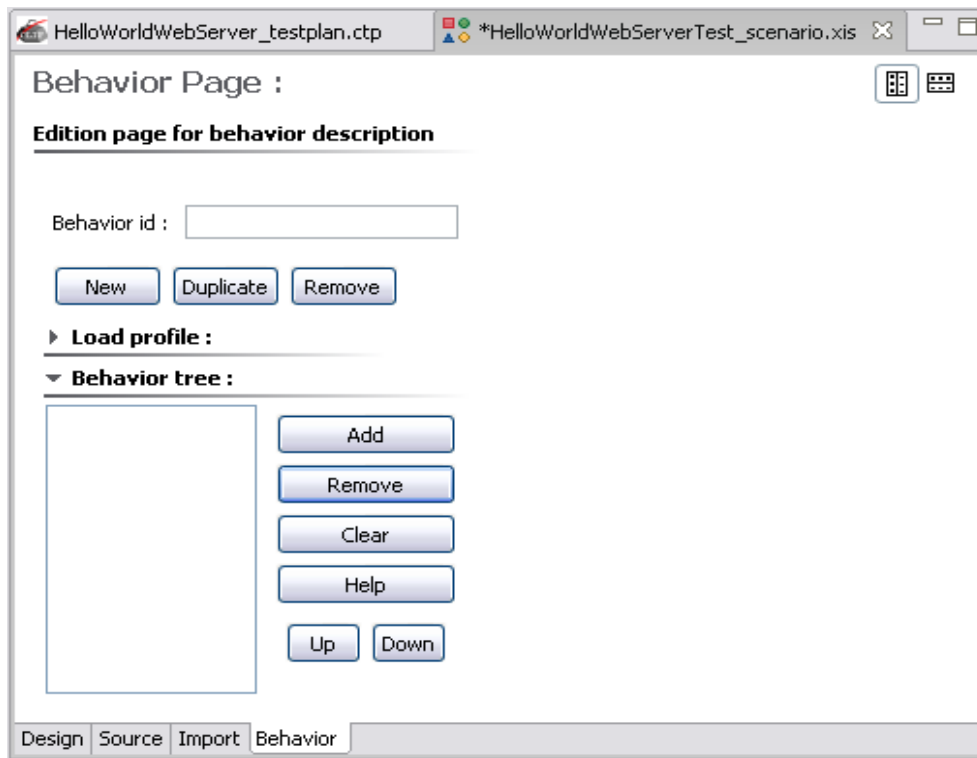
Now we have to define the behavior of our scenario.

So click on the Behavior tabulation



We arrive then on the Behavior

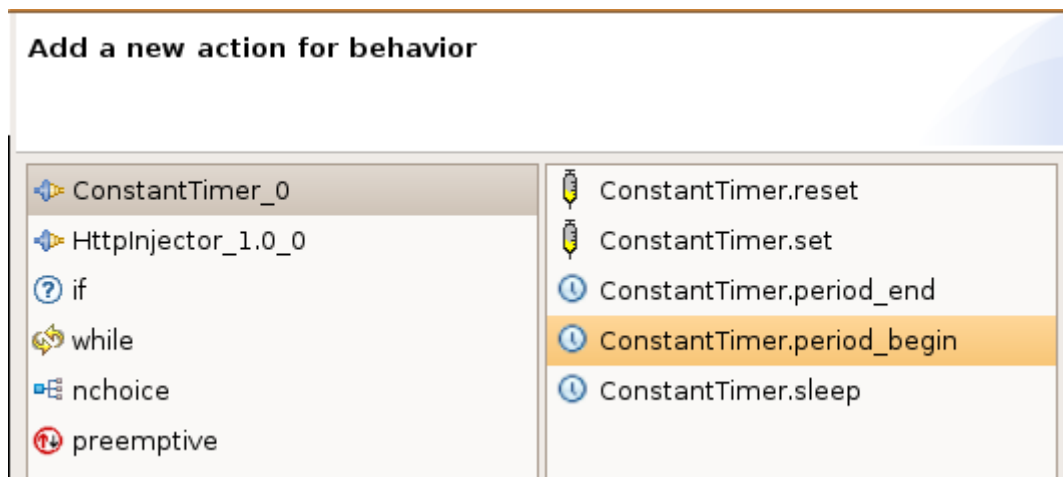
page :



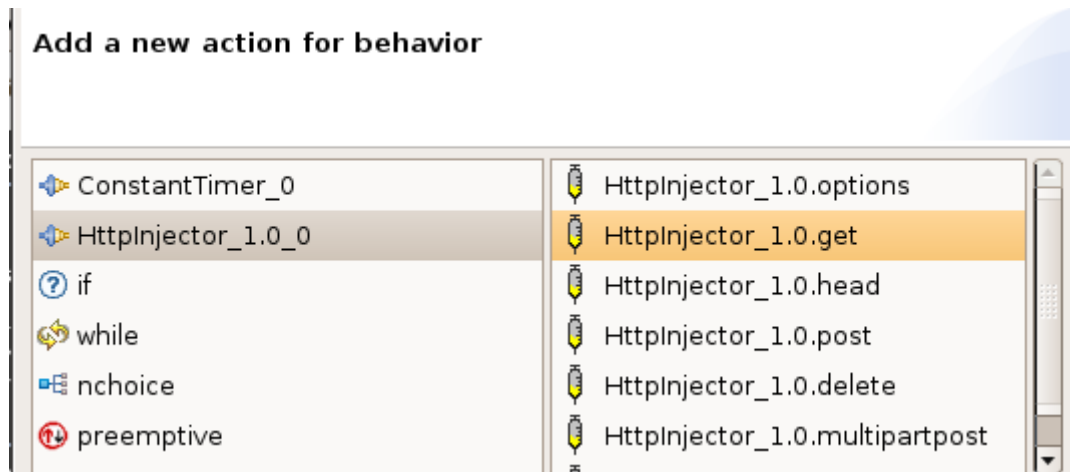
By the same way we will import plug-ins methods to define our scenario.

Click on the add button : 

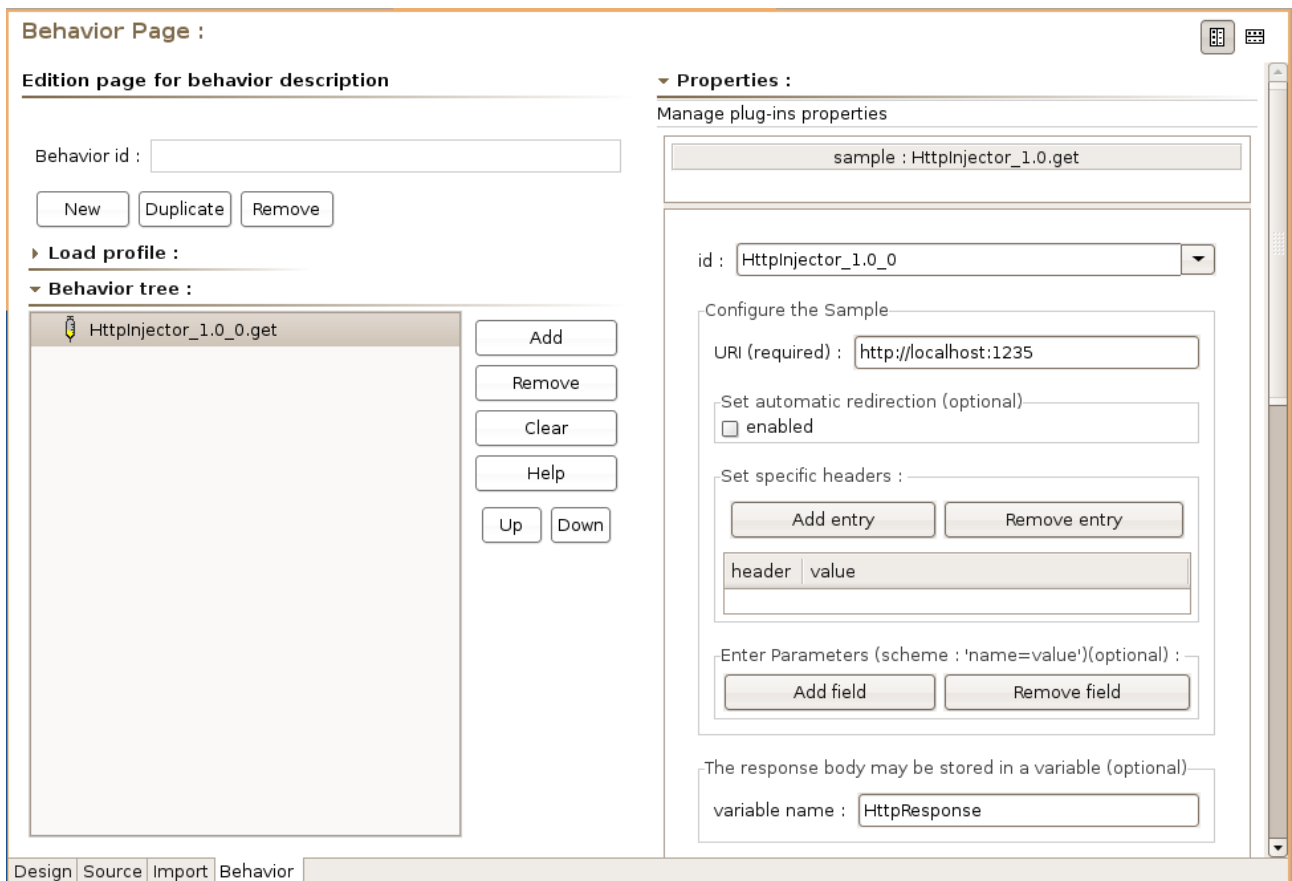
At the beginning of our scenario we initialize the constant timer by adding the `period_begin` method.



Then we add the `Get` method of `HttpInjector_1.0` by clicking on the add button.

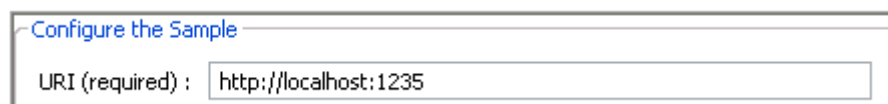


Once you added this method you have to set the GET method properties :

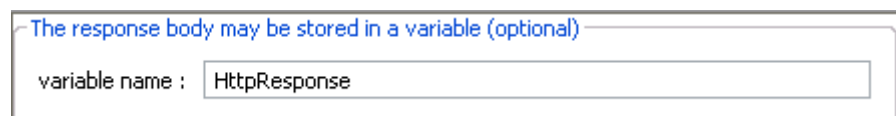


In our case we will only set :

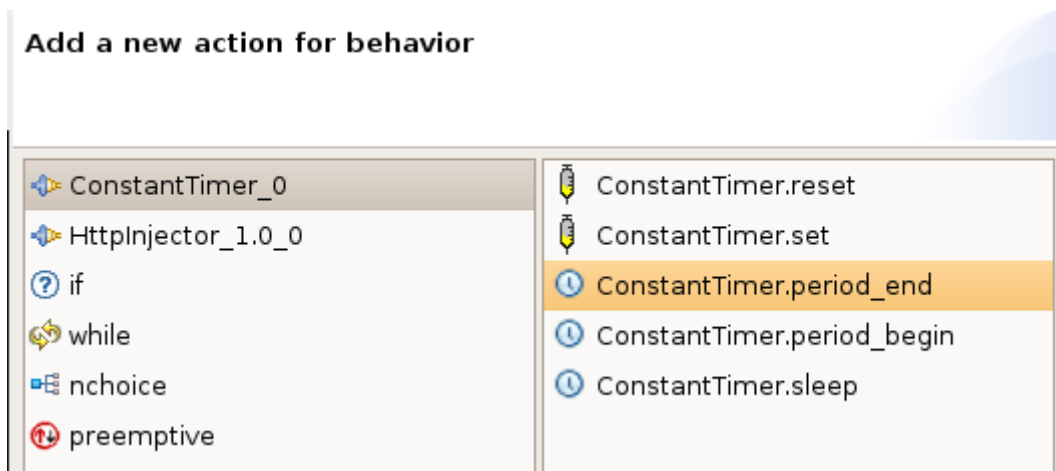
- the URI



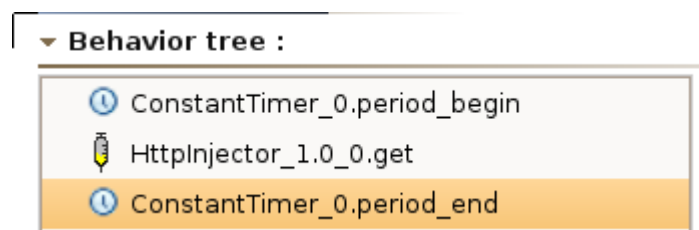
- the Response Body :



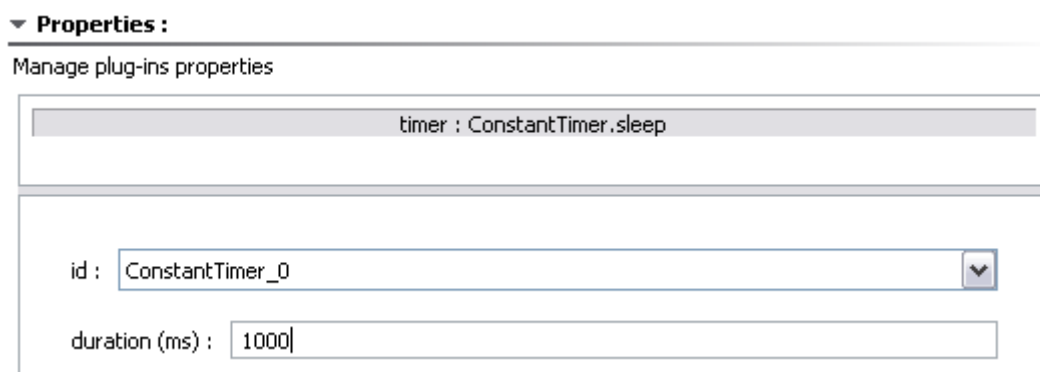
And finally we add the period_end method of the ConstantTimer plug-in :



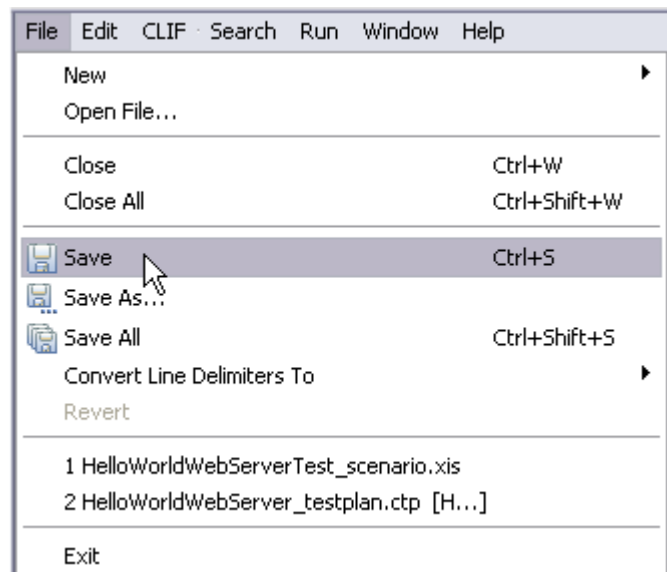
Then click on the ConstantTimer_0.period_end of The behavior tree :



And then set his duration properties to 1000 ms :



You can save your scenario file if you didn't do it before.



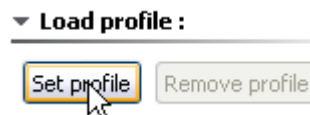
3.4. Load profile

Load profiles enable predefining how the population of each behavior will evolve, by setting the number of active instances according to time. A load profile is a sequence of lines or squares. For each load profile, a flag states if active instances shall be stopped to enforce a decrease of the population, or if the extra behaviors shall complete in a kind of a “lazy” approach.

To create a load profile you should be on the Behavior Page and click on the Load profile link



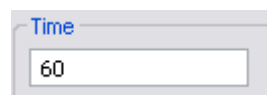
and after it on the Set profile button :



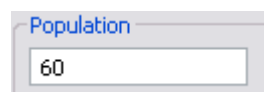
Be carefull, the time box contains the time in seconds since the start of the test and not the time in seconds of the ramp we are defining.

Now we want a ramp that increases the number of simultaneous scenarii. It begins at time 0 during 60 seconds and increases the number of simultaneous scenarii from 0 to 60.

Enter 60 in the Time text field :

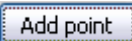


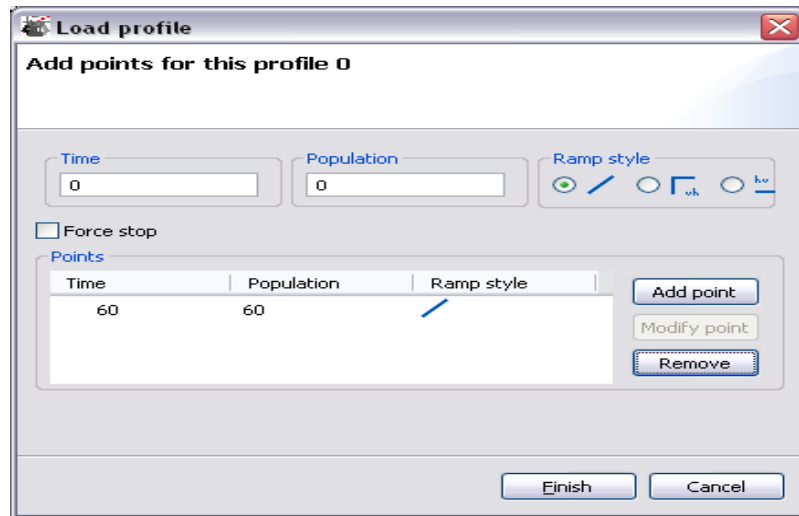
Enter 60 in the Population text field :



Choose the ramp style :




To save these settings click on the Add point button : 




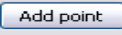
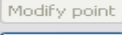
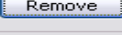
Load profile

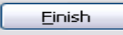

Add points for this profile 0

Time: 0 Population: 0 Ramp style: 

☐ Force stop

Time	Population	Ramp style
60	60	

After we want a stable number of simultaneous scenarii during 2 minutes (120 seconds); so the profile duration will be about 180 seconds. And at the end of the 2 minutes, the number of simultaneous scenarii will rise to 100 scenarii.

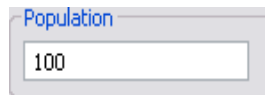
Now we want a ramp that increase the number of simultaneous scenarii. It begins at time 0 during 60 seconds and increase the number of simultaneous scenarii from 0 to 60.

Enter 60 in the Time text field :



Time: 180

Enter 60 in the Population text field :




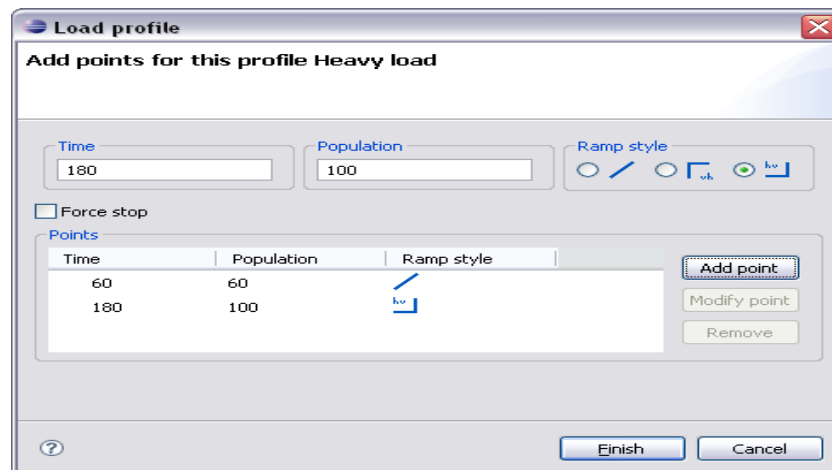
Population: 100

Choose the ramp style :




Ramp style: 

To save these settings click on the Add point button : 






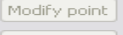
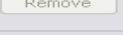
Load profile

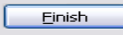

Add points for this profile Heavy load

Time: 180 Population: 100 Ramp style: 

☐ Force stop

Time	Population	Ramp style
60	60	
180	100	

And finally we want to decrease the number of simultaneous scenarii from 100 to 0 during 60 seconds.

Time

Enter 60 in the Time text field :

Enter 60 in the Population text field :

Population

Choose the ramp style :

Ramp style

☒ ☐ ☐ ☐ ☐

To save these settings click on the Add point button :

Add point

Load profile

Add points for this profile Heavy load

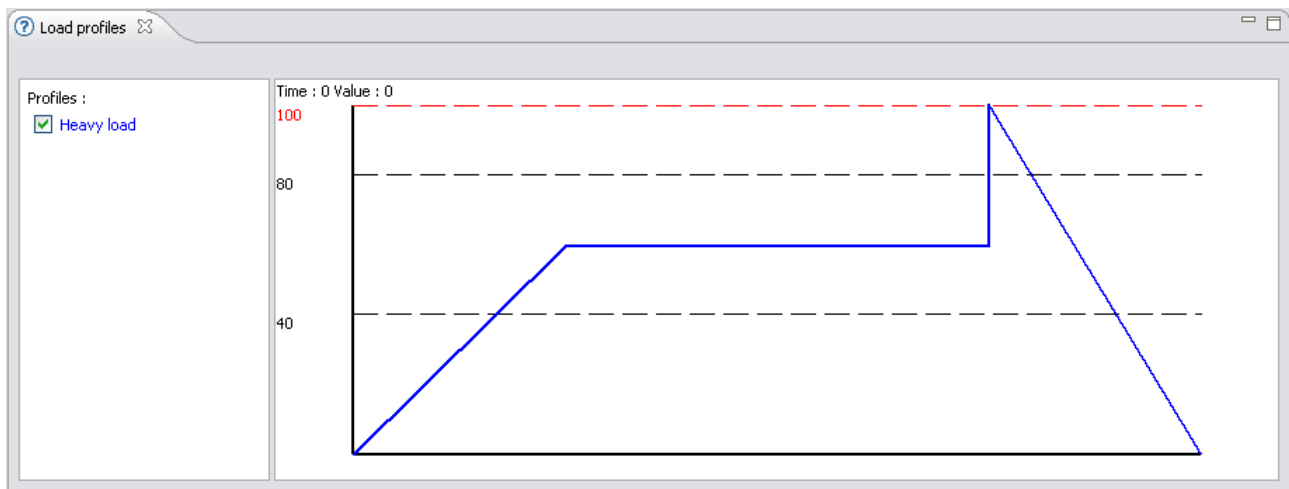
Time: Population: Ramp style: ☒ ☐ ☐ ☐ ☐

☐ Force stop

Points

Time	Population	Ramp style
60	60	<input checked="" type="radio"/>
180	100	<input type="radio"/>
240	0	<input type="radio"/>

Click on the Finish button :



You can see the load profile form watching the “Load profiles” view :

Don't forget to save your work (or Ctrl + s)

3.5. Launch load test

3.5.1. Start the Registry

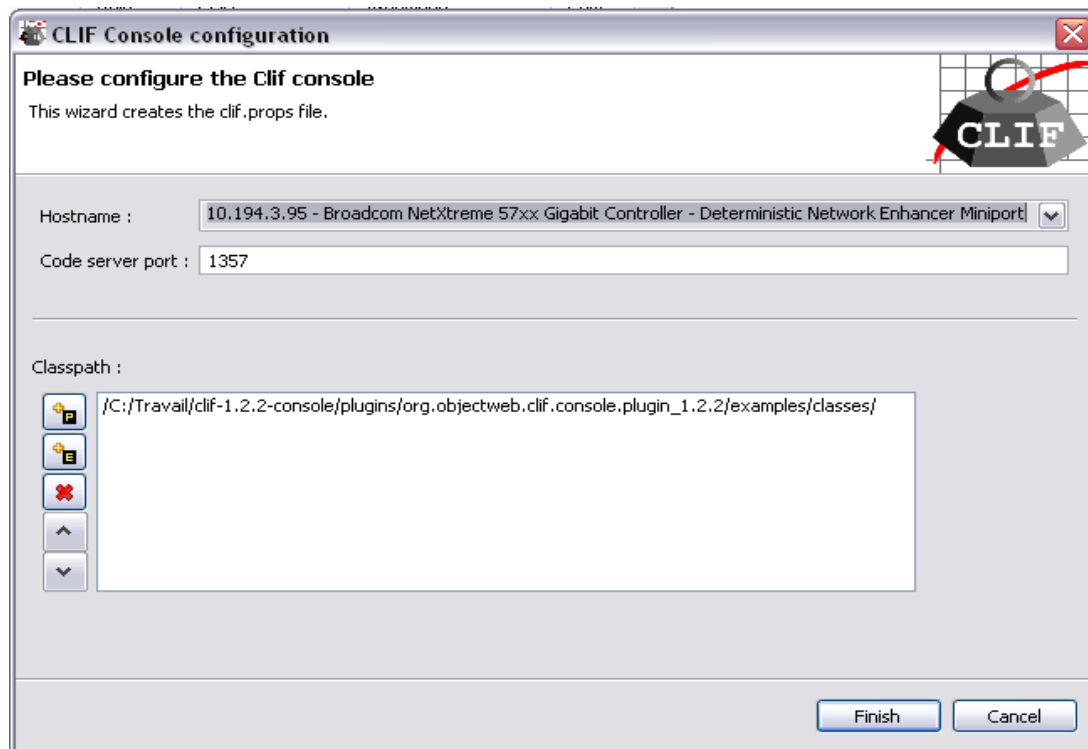
Click on the start registry button :  in the task button bar

Then you can set the :




- the host name which identify the server where the registry will be launch,
- the code server port
- and the classpath which is the path to access to the ctp, xis, csv (etc) files.



–CLIF user manual and programmer's guide

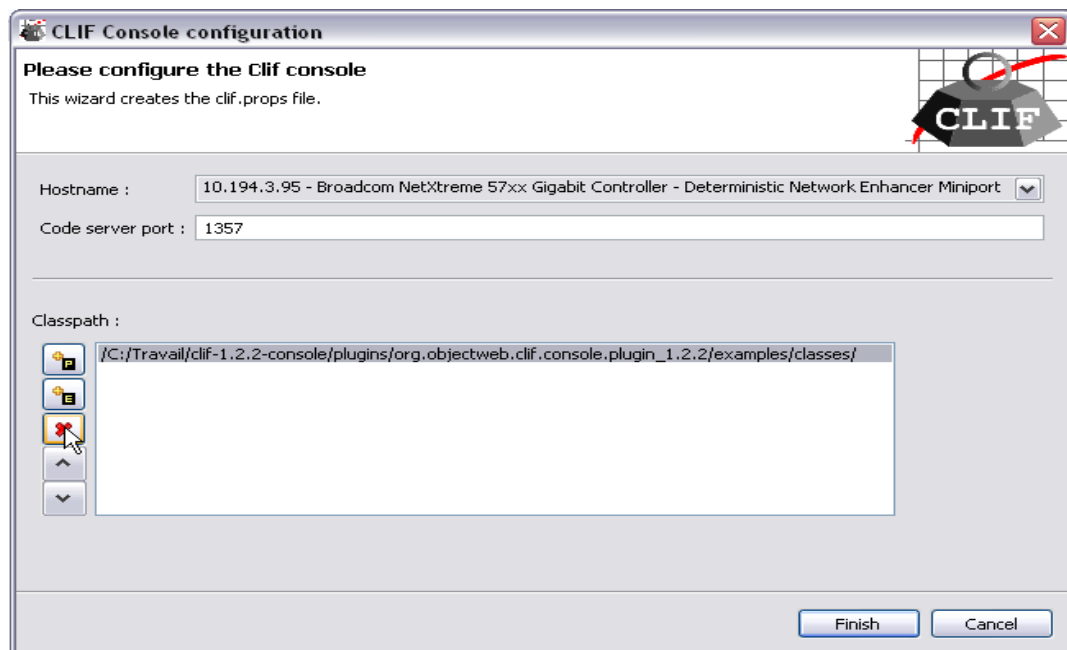


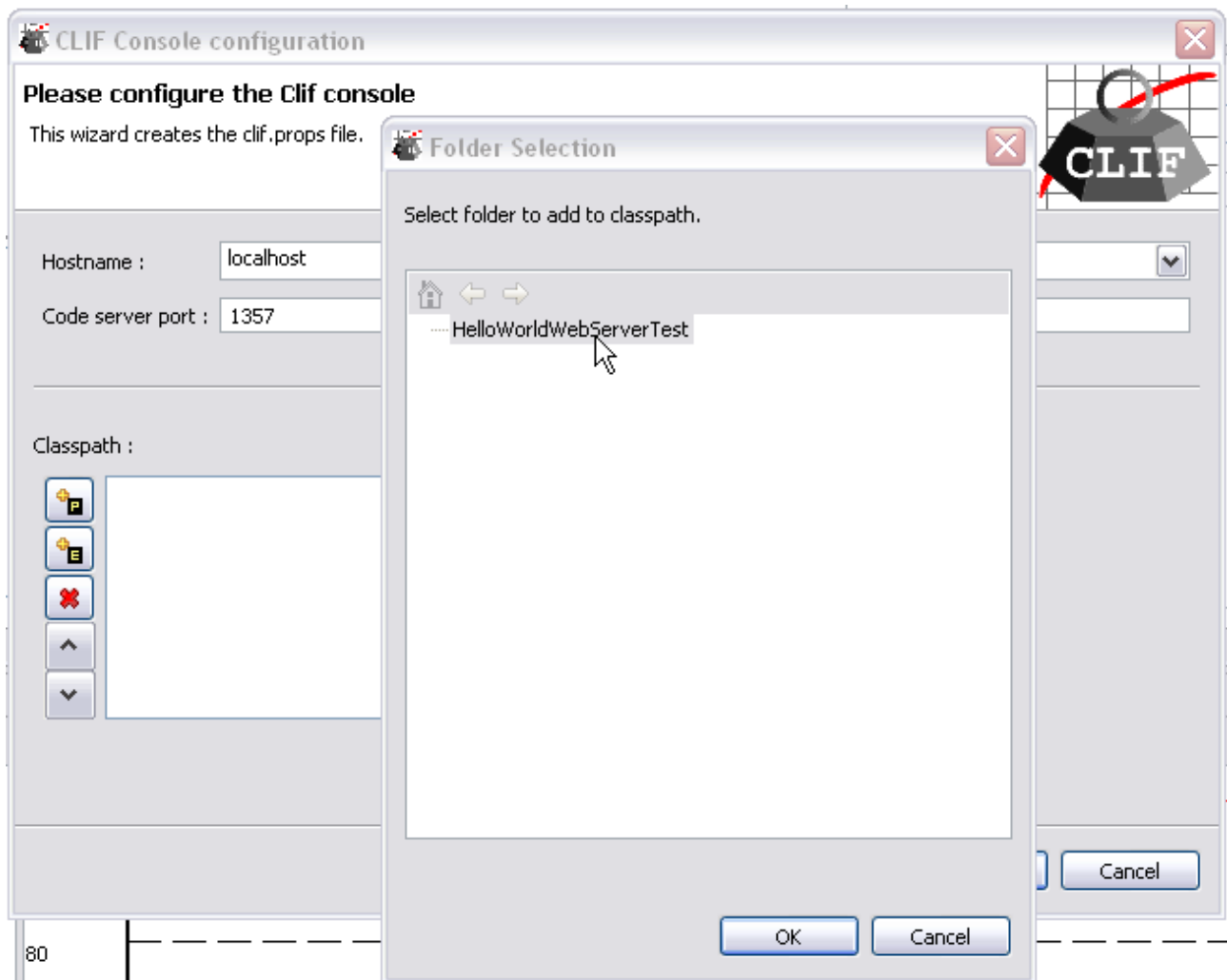
You can add or delete paths using these buttons :

-  to add projects path
-  to add externals path
-  to delete the selected path

In our case we will delete the default path and add our project path.

Select the default classpath and click on the delete button.

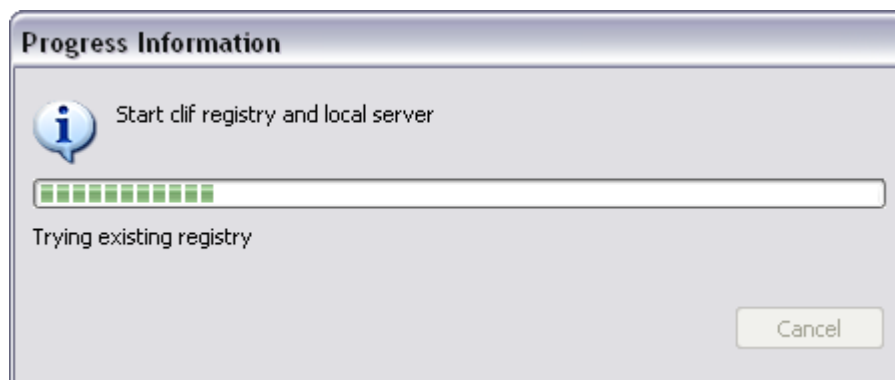




Then click on the “add project path” button. Select the HelloWorldWebServerTest project and click on the OK button.

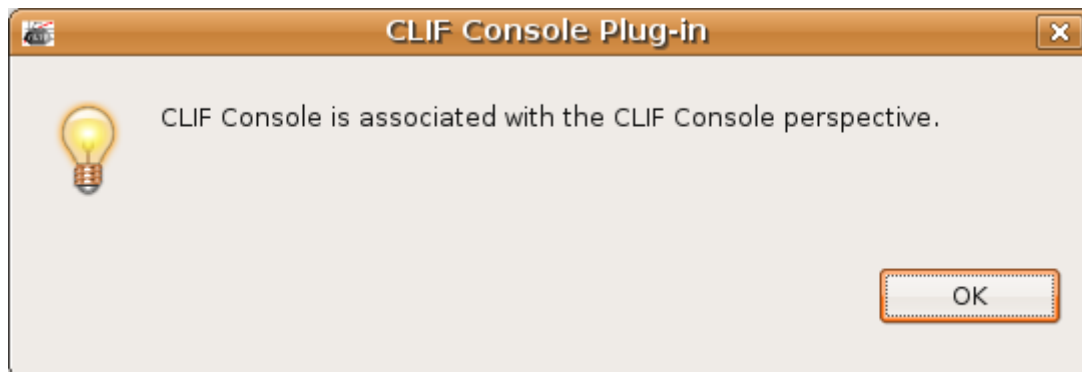
And finally click on finish to start the registry.

This pop up will appear during the launching of the registry.



–CLIF user manual and programmer's guide

When the registry will be launched, a pop-up will appear. Click on the OK button.



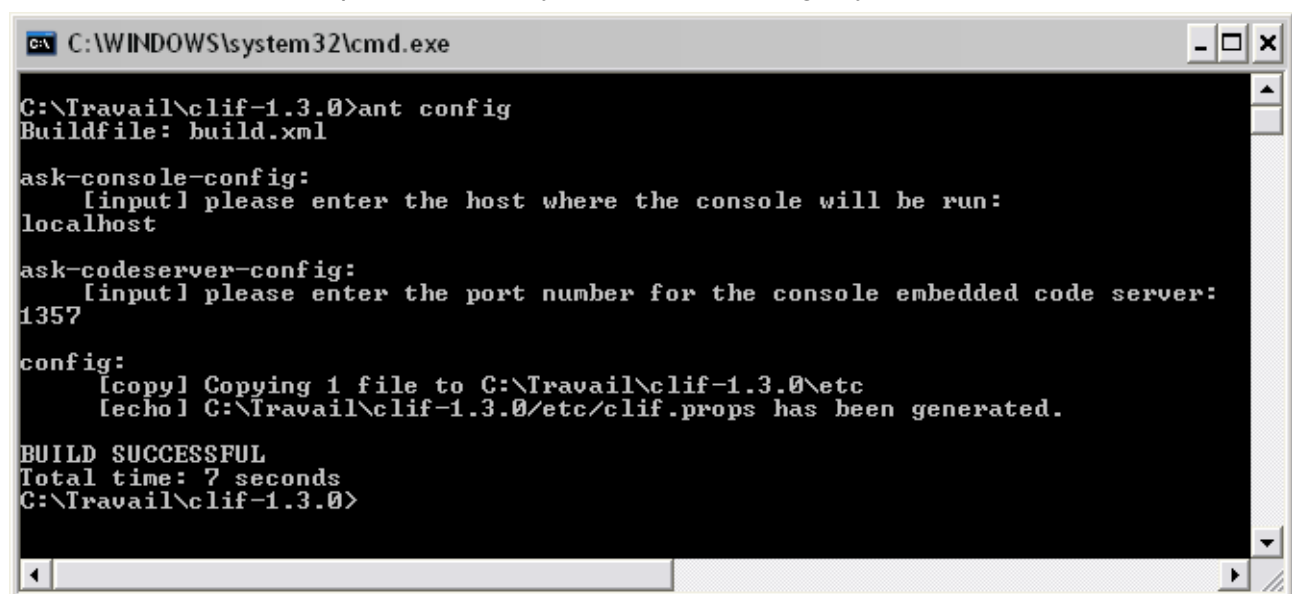
3.5.2. Run CLIF server

To be able to run CLIF server you have to get the `clif-server-<clifversion>.zip` archive and unzip it on your computer.

It will create a repertory name `clif-<clifversion>`.

Go into it using a command window and tape :

`ant config` : it will ask you to enter your host name or ip address and the port number that you setted in the code server port field when you launched the registry.

A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window shows the execution of the command `ant config` in the directory `C:\Travail\clif-1.3.0`. The output includes prompts for host name (answered with `localhost`) and port number (answered with `1357`), followed by a successful build message and the generation of `clif.props`.

```
C:\WINDOWS\system32\cmd.exe
C:\Travail\clif-1.3.0>ant config
Buildfile: build.xml

ask-console-config:
[input] please enter the host where the console will be run:
localhost

ask-codeserver-config:
[input] please enter the port number for the console embedded code server:
1357

config:
[copy] Copying 1 file to C:\Travail\clif-1.3.0\etc
[echo] C:\Travail\clif-1.3.0\etc\clif.props has been generated.

BUILD SUCCESSFUL
Total time: 7 seconds
C:\Travail\clif-1.3.0>
```

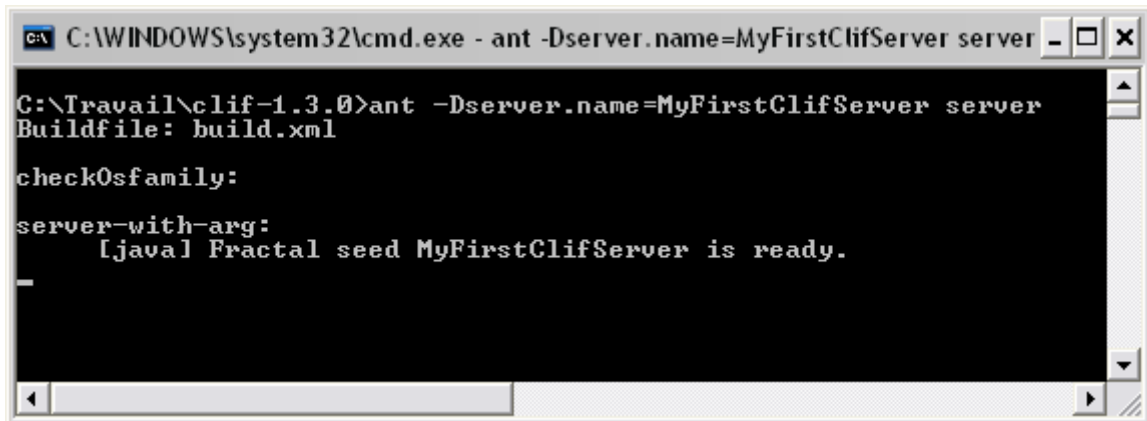
Now if your registry is running you can launch your Clif server otherwise you have to launch your registry (refer to chapter 3.5.1).

To launch your CLIF server use this command line :

`ant server`

or

ant -Dserver.name=MyFirstClifServer server if you to identify your server.



```

C:\WINDOWS\system32\cmd.exe - ant -Dserver.name=MyFirstClifServer server

C:\Travail\clif-1.3.0>ant -Dserver.name=MyFirstClifServer server
Buildfile: build.xml

checkOsfamily:

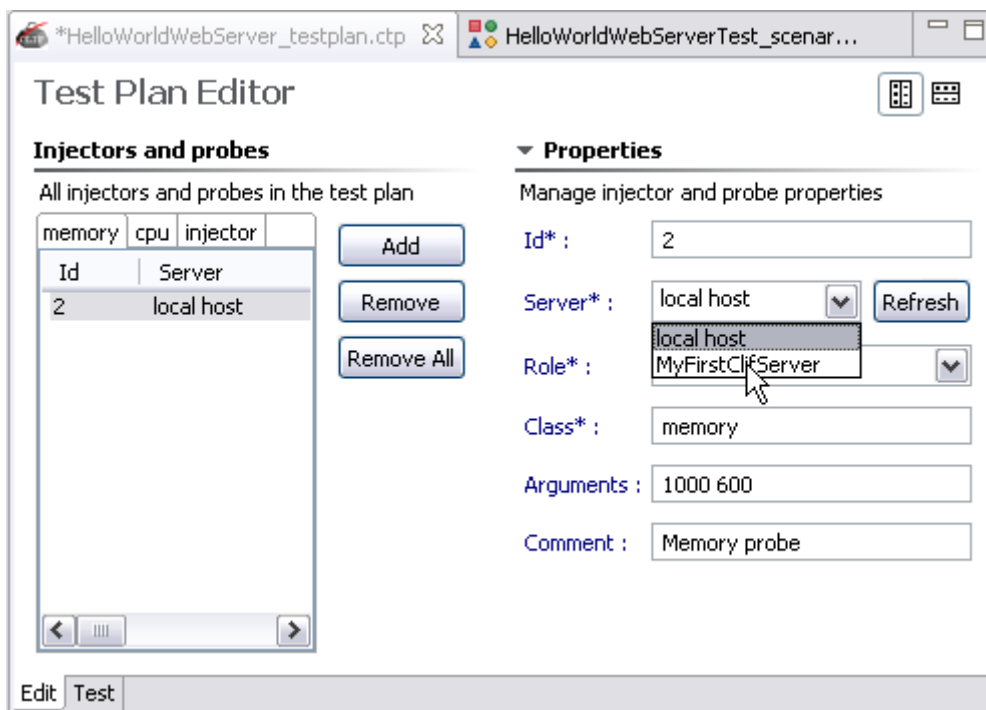
server-with-arg:
    [java] Fractal seed MyFirstClifServer is ready.
  
```

Now you have to modify your test plan to change the default server name (local host) by your server name give by the line “[java] Fractal seed **MyFirstClifServer** is ready”.

Go back to the CLIF console.

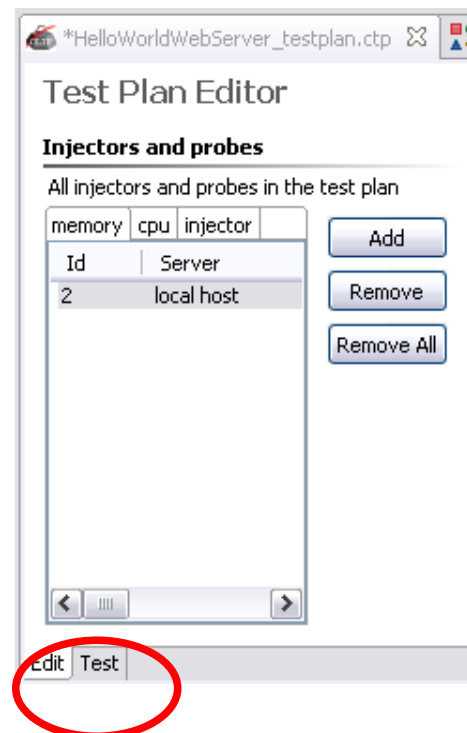
For on each probe or injector that you created you have to modify the default Server value.

Select the name that you gave to your server when you launch your clif server :



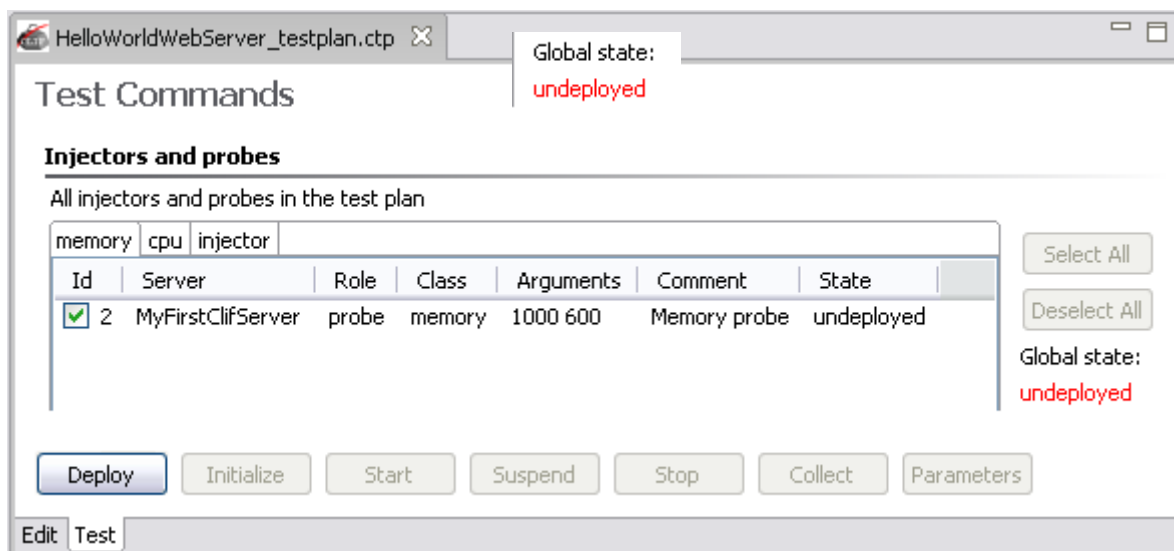
3.5.3. Launch load test

When you launched your registry (cf chapter 3.5.1) a new “Test” tabulation appeared at the bottom of the Test Plan Editor view.



Click on this new tabulation.

The Test Commands view appears.



Now you can see :

- the global state of your test plan
- the state of all your probe and injector.

memory cpu injector						
Id	Server	Role	Class	Arguments	Comment	State
<input checked="" type="checkbox"/> 2	MyFirstClifServer	probe	memory	1000 600	Memory probe	undeployed

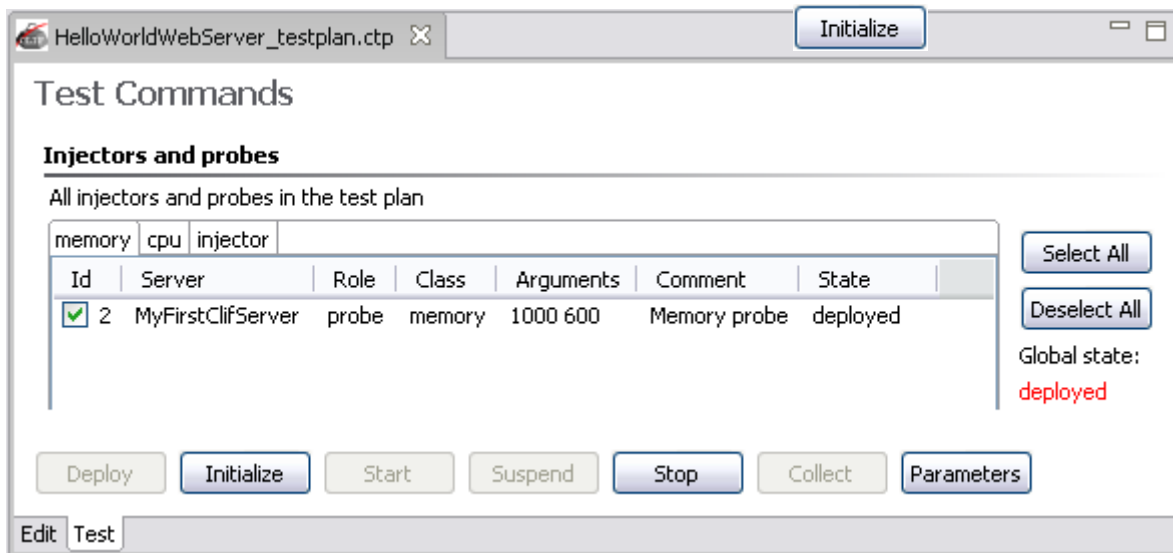
-

Now you deploy your testplan by clicking on the Deploy  button :

The global state and the probes and injector state changed from undeployed to deployed.

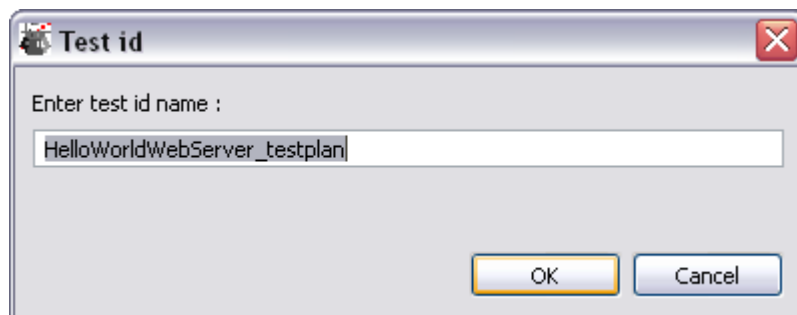
And now you can access to others actions :

- Initialize
- Stop
- Parameters



Then initialize your testplan by clicking on the Initialize button :

You have to set Test id name (by default it is the name off your test plan) and click on the OK button :

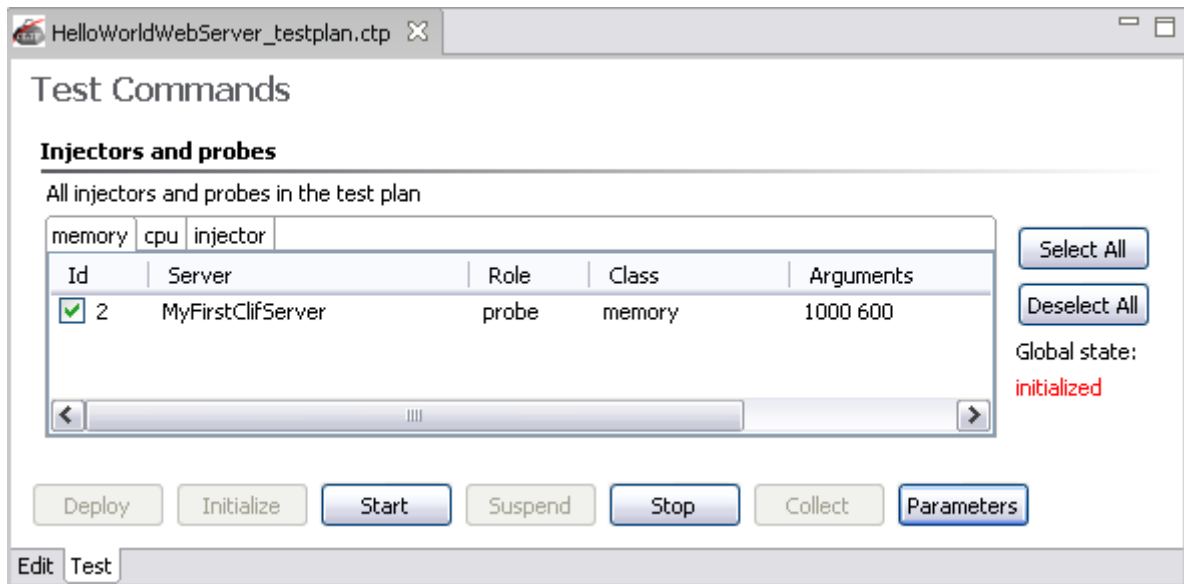


The global state and the probes and injector state changed from deployed to initialized.

And now you can access to others actions :

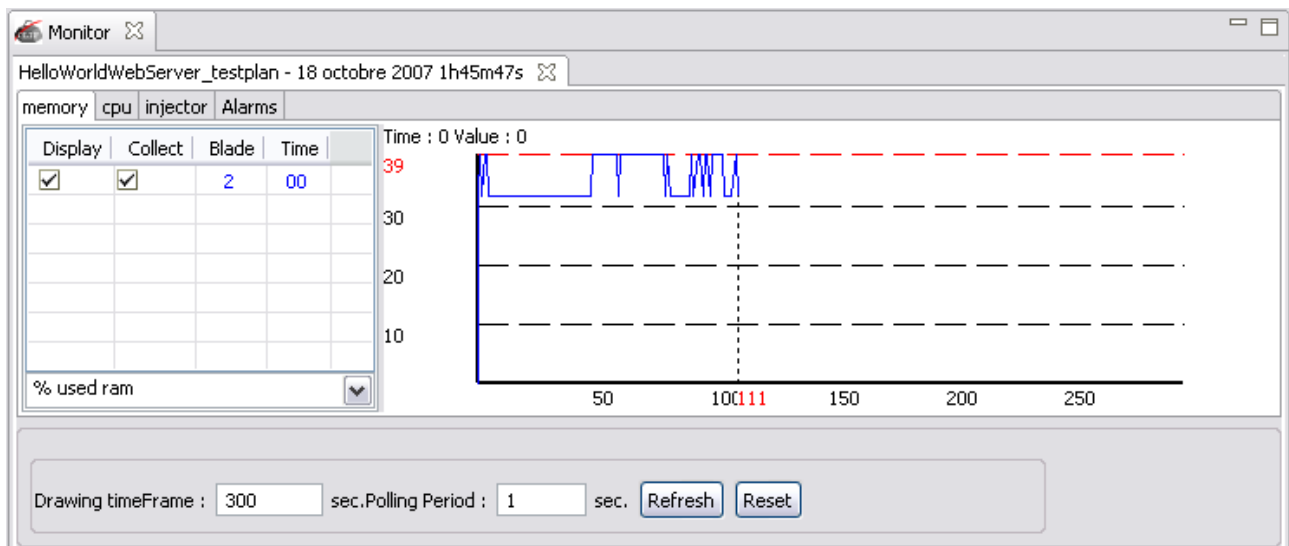
–CLIF user manual and programmer's guide

- Start
- Stop
- Parameters



Once the initialization done a new view “Monitor” appear down to the Test Commands view.

On this view you can see the monitoring of all your deployed probes and injectors.



Now you just have to start the load test by clicking on the **Start** button and look at the monitor view to inspect the behavior of your probes and injectors.

4. Appendix A – Hello world web server

Here is the Java code of the Hello world web server. To create this web server copy this code and paste it into a Java file named WebServer.java.

```
import java.net.*;
import java.io.*;
import java.util.*;

import sun.net.www.protocol.http.HttpURLConnection;

//    ** Main Class of the server, manage global information **
public class WebServer {
    static int port;
    static int nbConnection = 0;

    public static void main(String args[]) {
        ServerSocket server;
        //port definition
        try {
            port=Integer.parseInt(args[0]);
        } catch (Exception e) {
            port=1235; //default value
        }
        //installation
        try {
            server=new ServerSocket(port);
            System.out.println("Web Server running...");
            while (true) {
                //creation of a new connection
                Socket s=server.accept();
                nbConnection++;
                new Connection(s, nbConnection);
            }
        } catch (IOException e) {
            System.out.println("Error when trying to create a socket object :
"+e.getMessage());
            System.exit(1);
        }
    }
}

class Connection implements Runnable {
    Socket client; //link with the client
    BufferedReader fromClient; //reception of requests
    PrintWriter toClient; //send of responses

    public Connection(Socket client, int nbConnection) {
        this.client=client;
        try {

            // creation of data flows from/to the client
```

–CLIF user manual and programmer's guide

```
fromClient=new BufferedReader(  
new InputStreamReader(client.getInputStream()));  
toClient=new PrintWriter(  
new OutputStreamWriter(client.getOutputStream()),true);  
  
// Creation of the http response  
StringBuilder httpResponse = new StringBuilder();  
httpResponse.append("HTTP/1.0 200 OK\n");  
httpResponse.append("Date : "+Calendar.getInstance().getTime()+"\n");  
httpResponse.append("Content-Type : text/HTML\n");  
httpResponse.append("\n");  
httpResponse.append("Hello World");  
  
System.out.println("A client is connecting to the server web Hello  
World");  
  
toClient.println(httpResponse.toString());  
//System.out.println("Voici l'URL envoy   par le client :  
"+depuisClient.readLine());  
} catch (IOException e) {  
try {  
client.close();  
} catch (IOException ee) {}  
}  
//mise en route du processus par appel de la m  thode run  
new Thread(this).start();  
}  
  
public void run() {  
stop();  
}  
  
public void stop() {  
try {  
client.close();  
} catch (IOException e) {  
System.out.println("Exception when try to close socket : "+e);  
}  
}  
}
```

Now you have to compile this Java file. In the directory where you create this file enter the following command line :

```
javac WebServer.java
```

This will create to .class files : Connection.class and WebServer.class

Now to run the Hello World web Server you just have to enter this command line :

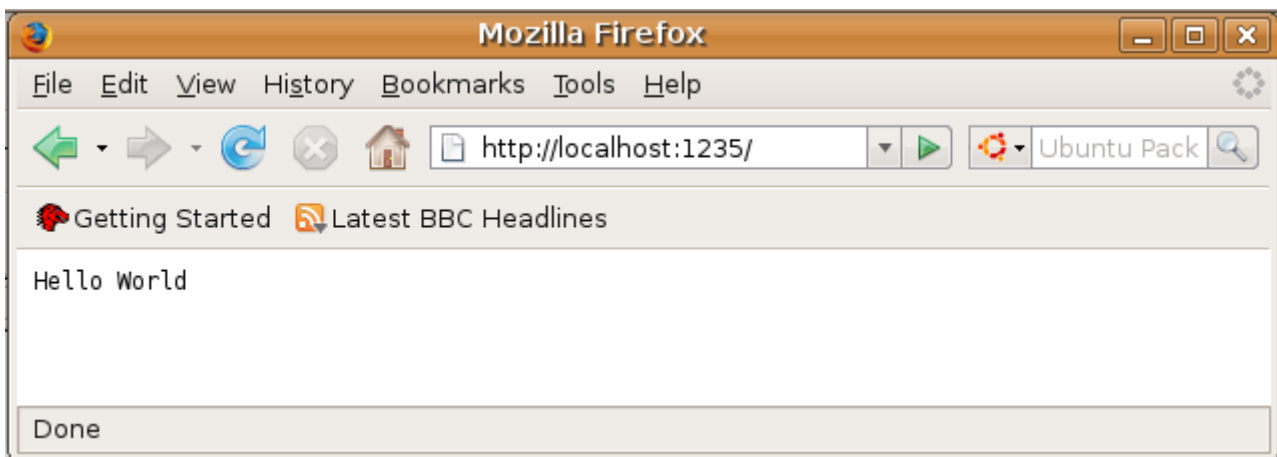
java WebServer

The following line appears : Web Server running...

To check if your Hello Web server is running correctly, you can open your internet browser and tape into the address bar the following url :

<http://localhost:1235>

You will see the next page :



And the console where you launch your Web server will display this :

```
Web Server launched...  
A client is connecting to the server web Hello World
```

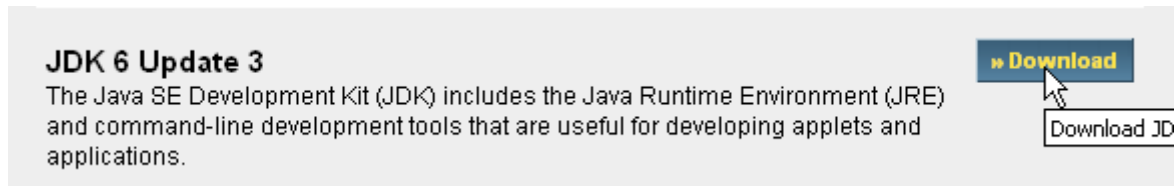
5. Appendix B : Install mandatory requirements

5.1. Download requirements

5.1.1. Sun J2SDK™ 1.5 (1.5 version or greater is mandatory)

Click on the following link : <http://java.sun.com/javase/downloads/?intcmp=1281>






Click on the download button of the current version of the JDK :



Accept the licence agreement :



Select the platform where CLIF will be used :

Windows Platform - Java(TM) SE Development Kit 6 Update 3			
<input checked="" type="checkbox"/> 			
Download the full version as a single file .			
	 Windows Offline Installation, Multi-language	jdk-6u3-windows-i586-p.exe	65.64 MB
	 Windows Online Installation, Multi-language	jdk-6u3-windows-i586-p-iftw.exe	373.39 KB

or

Linux Platform - Java(TM) SE Development Kit 6 Update 3			
<input checked="" type="checkbox"/>			
<input type="checkbox"/>		Linux RPM in self-extracting file	jdk-6u3-linux-i586-rpm.bin 61.64 MB
<input type="checkbox"/>		Linux self-extracting file	jdk-6u3-linux-i586.bin 65.40 MB

and save the downloading file on your computer.

Once the download done, you can launch the installation on the environment you want to install it.

5.1.2. Apache ant utility version 1.5.4 or greater

Click on the following link : <http://ant.apache.org/bindownload.cgi>

Click on the current release of apache ant utility depending of the platform where CLIF will be used (For windows select the .zip file and for linux the .tar.gz file) :

Current Release of Ant

Currently, Apache Ant 1.7.0 is the best available version, see the [release notes](#).

Note

Ant 1.7.0 has been released on 19-Dec-2006 and may not be available on all mirrors for a few days.

Tar files may require gnu tar to extract

Tar files in the distribution contain long file names, and may require gnu tar to do the extraction.

- .zip archive: [apache-ant-1.7.0-bin.zip](#) [PGP] [SHA1] [MD5]
- .tar.gz archive: [apache-ant-1.7.0-bin.tar.gz](#) [PGP] [SHA1] [MD5]
- .tar.bz2 archive: [apache-ant-1.7.0-bin.tar.bz2](#) [PGP] [SHA1] [MD5]

Then save the downloading file on your computer.

Once the download done, you can unzip the archive file where you want on your platform.

5.2. Environement variables setting

5.2.1. Windows OS

Now you have to set the following environment variable :

- JAVA_HOME=C:\Program Files\Java\jdk1.6.0_03
- ANT_HOME=C:\Program Files\apache-ant-1.7.0

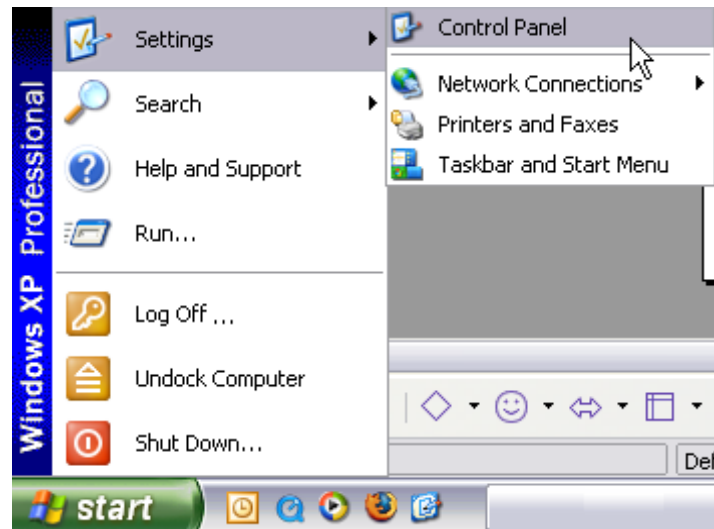
And to modify :

- PATH=<value already in path>;%JAVA_HOME%\bin;%ANT_HOME%\bin

Go to :

Start > Settings > Control Panel

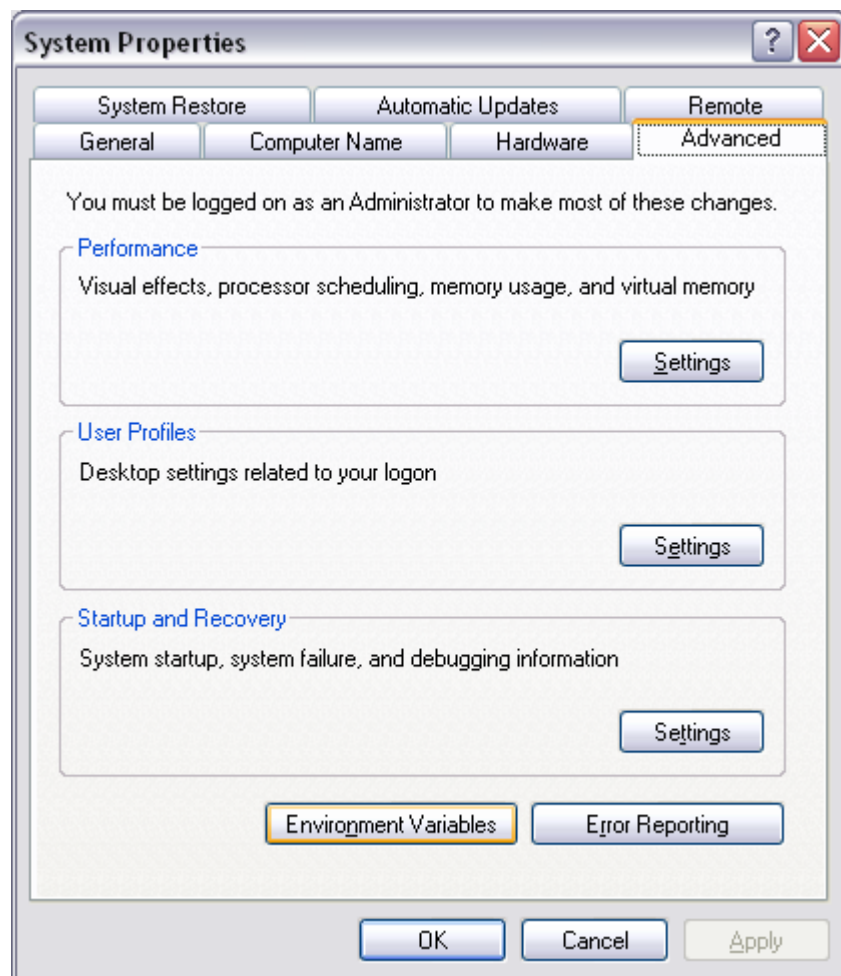
–CLIF user manual and programmer's guide



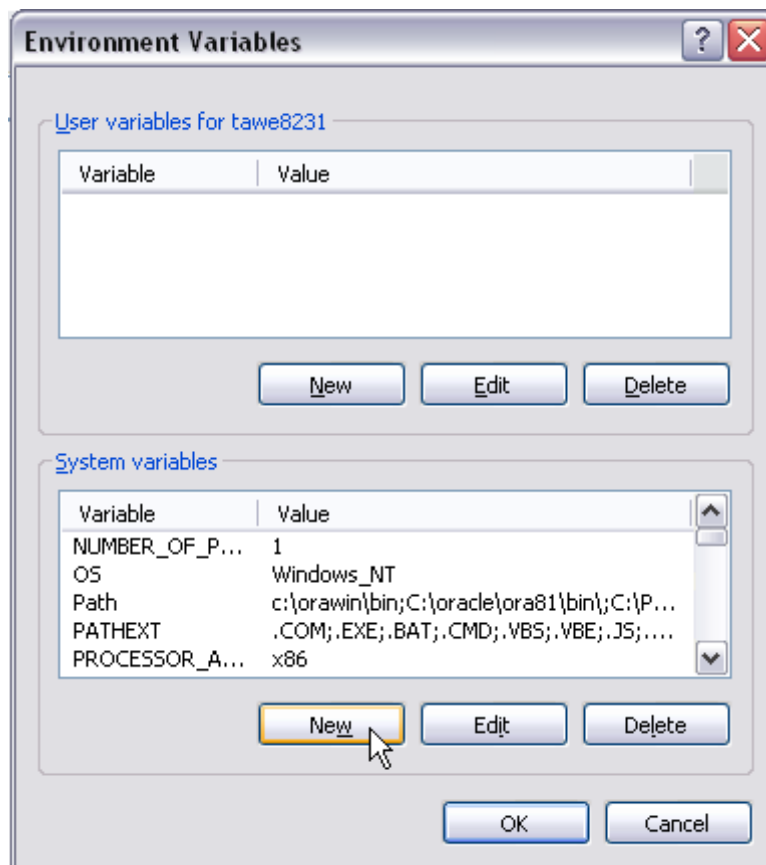
Double click on the system icon :



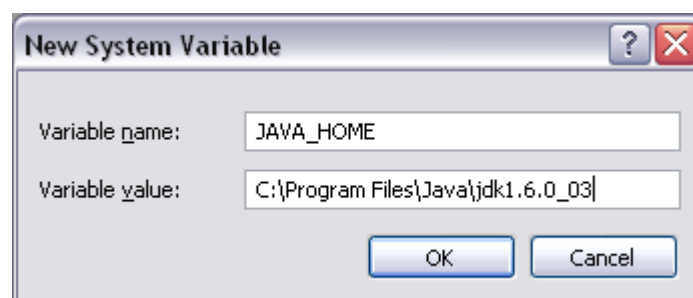
Choose advanced tab and click on the environment Variables button :



Now click on the New button of the System variables parameters group :

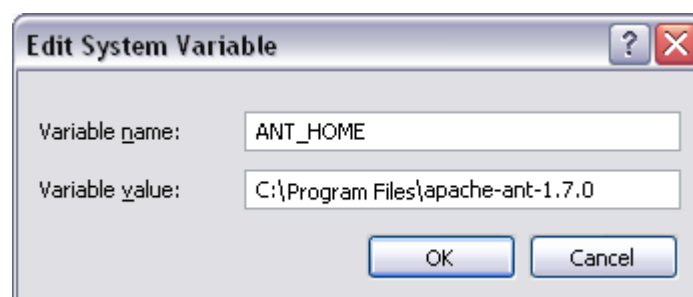


Then enter the variable name and its value :

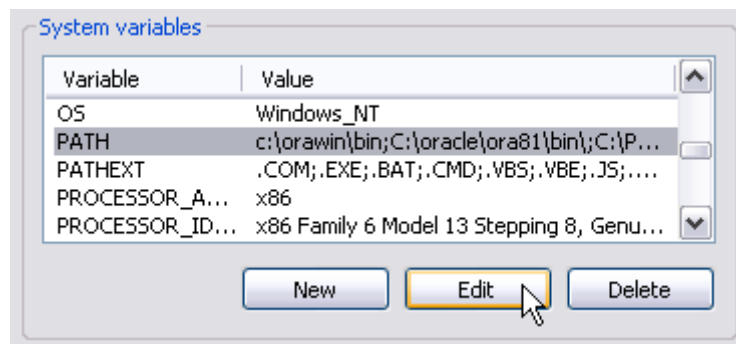


Do the same thing for ANT_HOME variable :

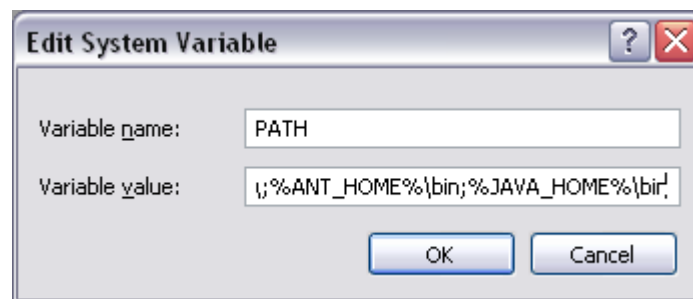
Now modify the PATH variable. Select the PATH variable and click on the Edit button :



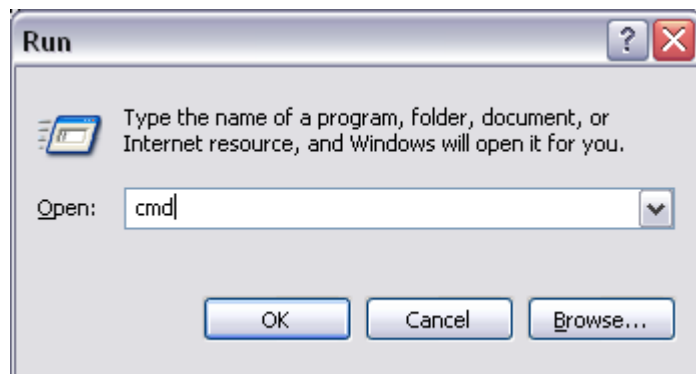
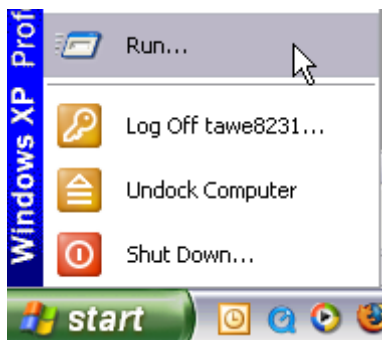
–CLIF user manual and programmer's guide

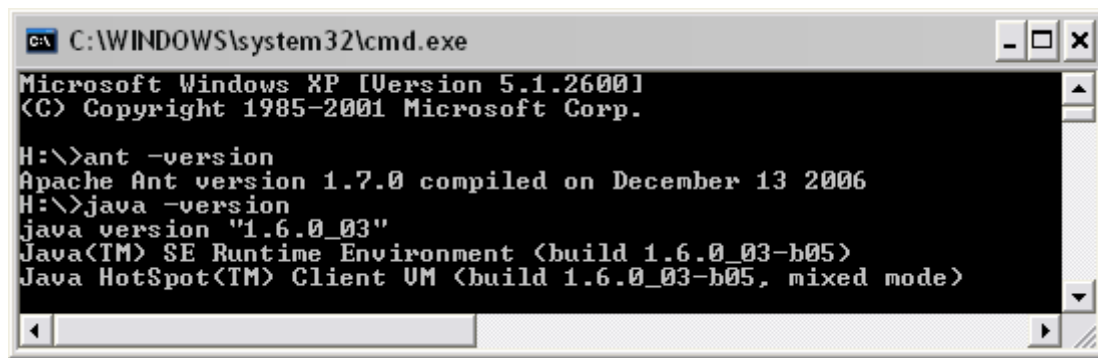


Now add the reference to the ANT_HOME\bin and JAVA_HOME\bin repertory at the end of the PATH line :



Now you just have to check if the good java version and ant version are used by your system.





```

C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

H:\>ant -version
Apache Ant version 1.7.0 compiled on December 13 2006
H:\>java -version
java version "1.6.0_03"
Java(TM) SE Runtime Environment (build 1.6.0_03-b05)
Java HotSpot(TM) Client VM (build 1.6.0_03-b05, mixed mode)

```

5.2.2. Linux OS

Now you have to set the following environment variable :

- JAVA_HOME=/usr/lib/jdk1.6.0_03
- ANT_HOME=/usr/lib/apache-ant-1.7.0

And to modify :

- PATH=\$PATH:\$HOME/bin:\$ANT_HOME/bin:\$JAVA_HOME/bin

Go to the root directory of your user account.

Modify the .bash_profile file with the following command line : vi .bash_profile

If this file doesn't exist you can create it.

Put in it the following line :

```

# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
    . ~/.bashrc
fi

# User specific environment and startup programs

ANT_HOME=/usr/lib/apache-ant-1.7.0

JAVA_HOME=/usr/lib/jdk1.6.0_03

PATH=$PATH:$HOME/bin:$ANT_HOME/bin:$JAVA_HOME/bin

export ANT_HOME
export JAVA_HOME
export PATH

```

```
unset USERNAME
```

Now you have to logg off from your platform and then logg on to reload the .bash_profile file.

Now you just have to check if the good java version and ant version are used by your system.

```
[clif@ ~]$ ant -version
Apache Ant version 1.7.0 compiled on December 13 2006
[clif@ ~]$ java -version
java version "1.6.0_02"
Java(TM) SE Runtime Environment (build 1.6.0_02-b05)
Java HotSpot(TM) Server VM (build 1.6.0_02-b05, mixed mode)
[clif@ ~]$
```

5.2.3. Mac OS X

[TODO]