Les tests

Exercice 1 :

Créer un projet Maven Java qui permet de chercher le nombre max d'une liste d'entiers.

- Générer l'architecture du projet

```
mvn archetype:generate
```

- Modifier le code source de classe java principale **App.java** en ajouter la fonction maxListeNombres

```
public int maxListeNombres (int[] list) {
  int i,max=0;
  for(i = 0;i<list.length ; i++) {
        if(list[i] > max) {
              max = list[i];
        }
  }
  return max;
}
```

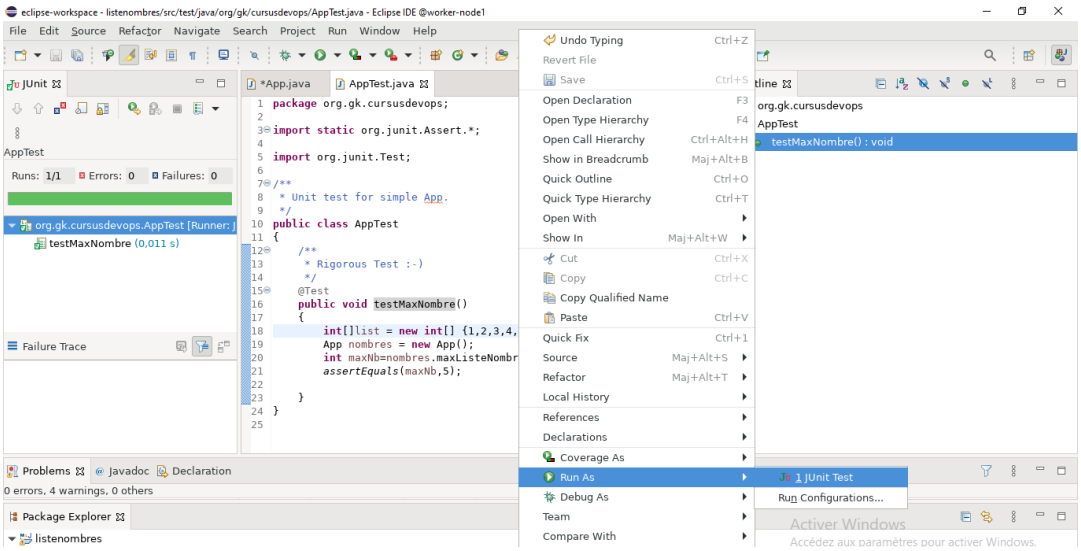
- Modifier le code source de classe Test java principale **AppTest.java** en ajouter la fonction maxListeNombres

```
@Test
public void testMaxNombre()
{
    int[]list = new int[] {1,2,3,4,5};
    App nombres = new App();
    int maxNb=nombres.maxListeNombres(list);
    assertEquals(maxNb,5);
}
```

- Tester en mode terminal

```
[root@jenkins ~]# mvn test
```

- Tester en mode graphique



Ajout des autres cas de figures de test

- 2ème liste dans la même fonction

```
        int[] list2 = new int[] {10,20,30};
         int maxNb2=nombres.maxListeNombres(list);
         assertEquals(maxNb,30);
```

- testMaxNombreNegatifs

```
  @Test
    public void testMaxNombreNegatifs() {
         int[]list = new int[] {-1,-2,-3,-4,-5};
         App nombres = new App();
         int maxNb=nombres.maxListeNombres(list);
         assertEquals(maxNb,-1);
    }
```

⇨ il faut changer la valeure **`max=Integer.MIN_VALUE`**

- Test d'une liste non trié

```
         int[]list2 = new int[] {-1,-4,-2,-5,-3};
         int maxNb2=nombres.maxListeNombres(list2);
         assertEquals(maxNb2,-1);
```

- Test d'une liste avec une seule valeur

```
    @Test
    public void testMaxNombreAvecUneValeur()     {
         int[]list = new int[] {100};
         App nombres = new App();
         int maxNb=nombres.maxListeNombres(list);
         assertEquals(maxNb,100);
    }
```

- Test avec valeurs dupliquées

```
    @Test
    public void testMaxNombreDupliques()    {
        int[]list = new int[] {1,2,1,2,4,1};
        App nombres = new App();
        int maxNb=nombres.maxListeNombres(list);
        assertEquals(maxNb,4);
    }
```

Exercice 2 : Test d'intégration

- Télécharger le projet : https://gitlab.com/meddeb/calculator
- Vérifier les tests unitaires du projet
- Ajouter la classe de test d'intégration CalculatorServiceIT.java avec le code suivant

```
package com.devops.calculator;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
import org.junit.Test;
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;

public class CalculatorServiceIT {

    @Test
    public void testPing() throws Exception {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpGet                httpGet                =                new
HttpGet("http://localhost:9999/calculator/api/calculator/ping");
        HttpResponse response = httpclient.execute(httpGet);
        assertEquals(200, response.getStatusLine().getStatusCode());
        assertThat(EntityUtils.toString(response.getEntity()),
containsString("Welcome to Java Maven Calculator Web App!!!"));
    }

    @Test
    public void testAdd() throws Exception {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpGet                httpGet                =                new
HttpGet("http://localhost:9999/calculator/api/calculator/add?x=8&y=26");
        HttpResponse response = httpclient.execute(httpGet);
        assertEquals(200, response.getStatusLine().getStatusCode());
        assertThat(EntityUtils.toString(response.getEntity()),
containsString("\"result\":34"));
```

```
    }

    @Test
    public void testSub() throws Exception {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpGet                httpGet                =                new
HttpGet("http://localhost:9999/calculator/api/calculator/sub?x=12&y=8");
        HttpResponse response = httpclient.execute(httpGet);
        assertEquals(200, response.getStatusLine().getStatusCode());
        assertThat(EntityUtils.toString(response.getEntity()),
containsString("\"result\":4"));
    }

    @Test
    public void testMul() throws Exception {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpGet                httpGet                =                new
HttpGet("http://localhost:9999/calculator/api/calculator/mul?x=11&y=8");
        HttpResponse response = httpclient.execute(httpGet);
        assertEquals(200, response.getStatusLine().getStatusCode());
        assertThat(EntityUtils.toString(response.getEntity()),
containsString("\"result\":88"));
    }

    @Test
    public void testDiv() throws Exception {
        CloseableHttpClient httpclient = HttpClients.createDefault();
        HttpGet                httpGet                =                new
HttpGet("http://localhost:9999/calculator/api/calculator/div?x=12&y=12");
        HttpResponse response = httpclient.execute(httpGet);
        assertEquals(200, response.getStatusLine().getStatusCode());
        assertThat(EntityUtils.toString(response.getEntity()),
containsString("\"result\":1"));
    }
}
```

- Ajouter la configuration suivante dans le fichier pom.xml pour le test d'intégration

```
            <plugin>
                <groupId>org.eclipse.jetty</groupId>
                <artifactId>jetty-maven-plugin</artifactId>
                <version>9.4.11.v20180605</version>
                <configuration>
                    <webApp>
                        <contextPath>/calculator</contextPath>
                    </webApp>
                    <webAppSourceDirectory>target/${project.artifactId}-
${project.version}</webAppSourceDirectory>
                    <stopPort>8079</stopPort>
                    <stopKey>stop-jetty-for-it</stopKey>
                    <stopWait>10</stopWait>
                    <httpConnector>
```

```
                    <port>9999</port>
                    <idleTimeout>60000</idleTimeout>
                </httpConnector>
            </configuration>
            <executions>
                <execution>
                    <id>start-jetty</id>
                    <phase>pre-integration-test</phase>
                    <goals>
                        <goal>start</goal>
                    </goals>
                    <configuration>
                        <scanIntervalSeconds>0</scanIntervalSeconds>
                        <daemon>true</daemon>
                    </configuration>
                </execution>
                <execution>
                    <id>stop-jetty</id>
                    <phase>post-integration-test</phase>
                    <goals>
                        <goal>stop</goal>
                    </goals>
                </execution>
            </executions>
        </plugin>


        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <version>2.22.0</version>
            <executions>
                <execution>
                    <id>run-integration-test</id>
                    <phase>integration-test</phase>
                    <goals>
                        <goal>test</goal>
                    </goals>
                    <configuration>
                        <includes>
                            <include>**/*IT.java</include>
                        </includes>
                    </configuration>
                </execution>
            </executions>
        </plugin>
```

Exercice 2 : Test de performance

Configuration de l'outil jmeter sous Centos

- Téléchargez le binaire du Jmeter

```
[root@jenkins ~]# wget https://downloads.apache.org/jmeter/binaries/apache-jmeter-5.4.3.tgz
```

- Extraction des fichiers dans l'emplacement approprié

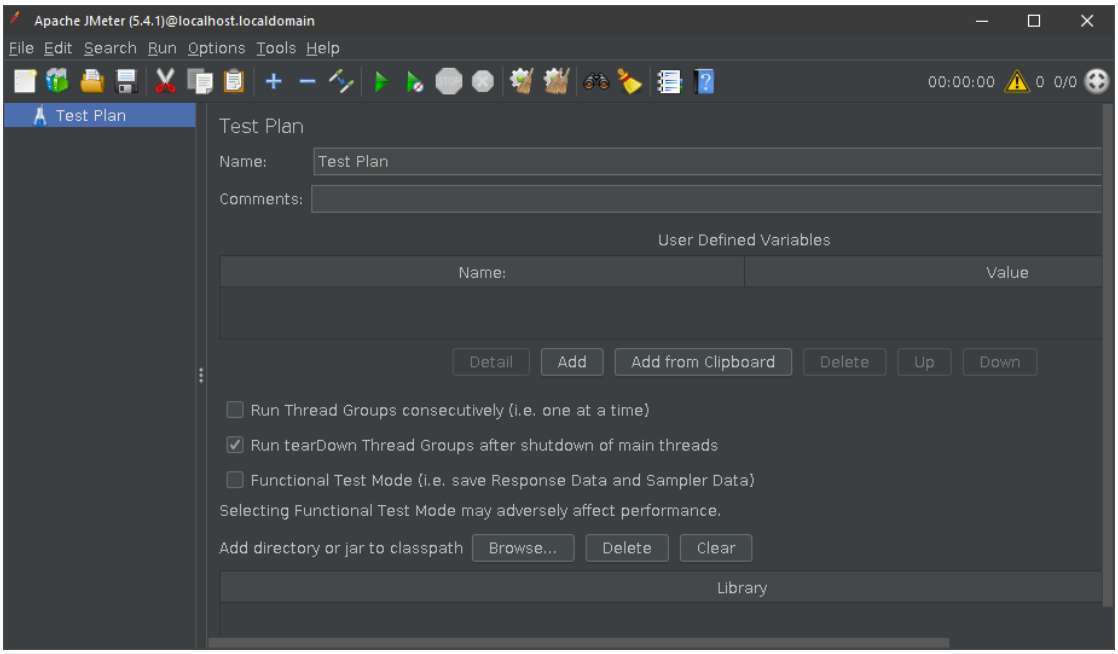
```
[root@jenkins ~]# tar xvfz apache-jmeter-5.4.3.tgz
[root@jenkins ~]# mv apache-jmeter-5.4.3 /opt/jmeter
```

Premier plan de test JMeter utilisant une interface graphique

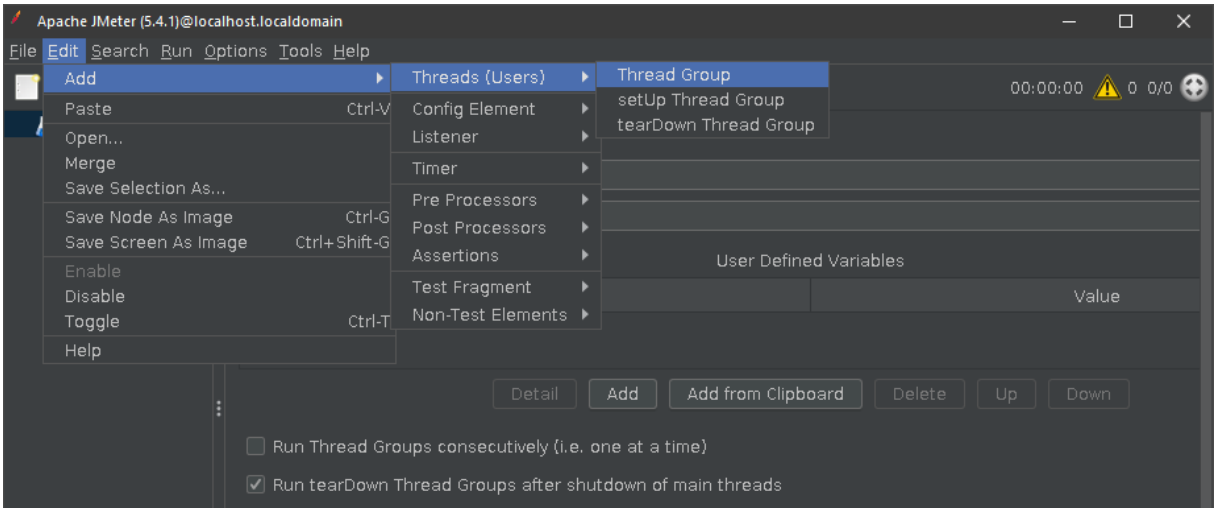
JMeter utilise des plans de test pour organiser chaque test. Une fois configuré, Jenkins appellera tous les plans de test définis sur un pipeline, puis affichera les résultats dans les rapports de construction. Cela signifie que vous devez définir tous les plans de test sur JMeter dans un premier temps, puis configurer Jenkins pour interpréter le résultat.

- Démarrez l'outil Jmeter

```
[root@jenkins ~]# /opt/jmeter/bin/jmeter
```

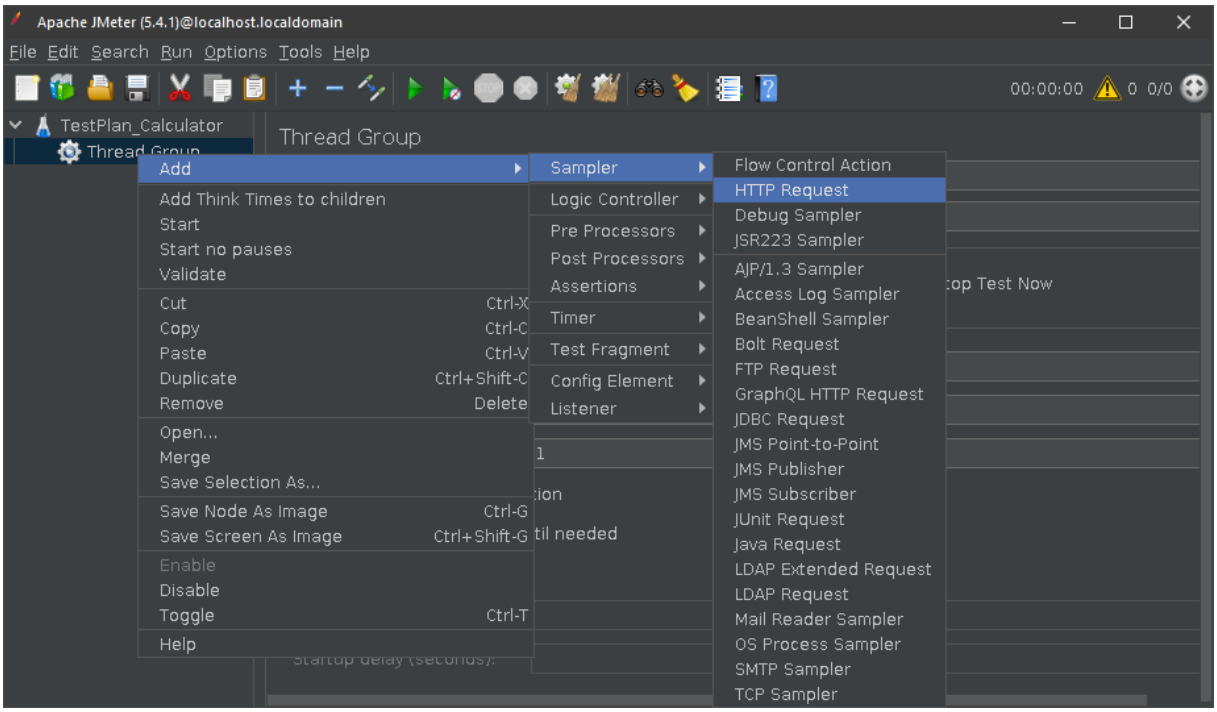


- Ajoutez le « thread Group » pour construire le plan du test

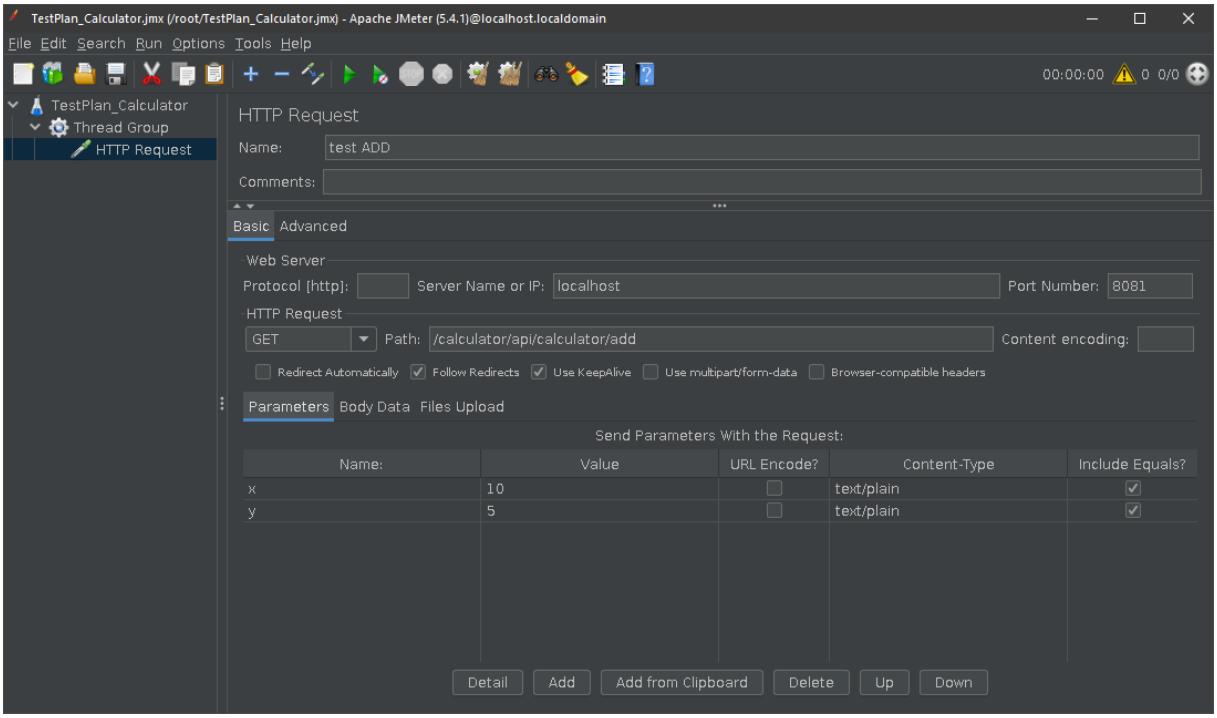


NB : adaptez les paramètres Number of threads (users) et Loop Count selon la taille du test que vous désirez.

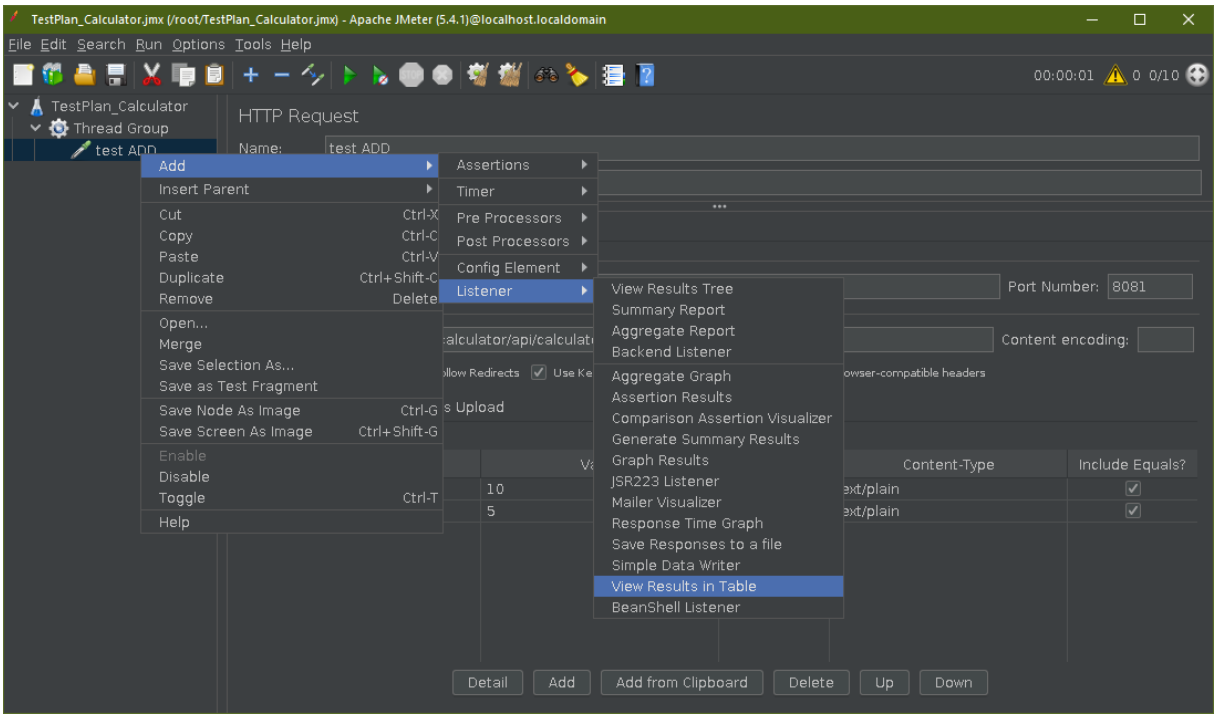
- Ajoutez les tests de type "http Request" pour tester les différentes pages du projet



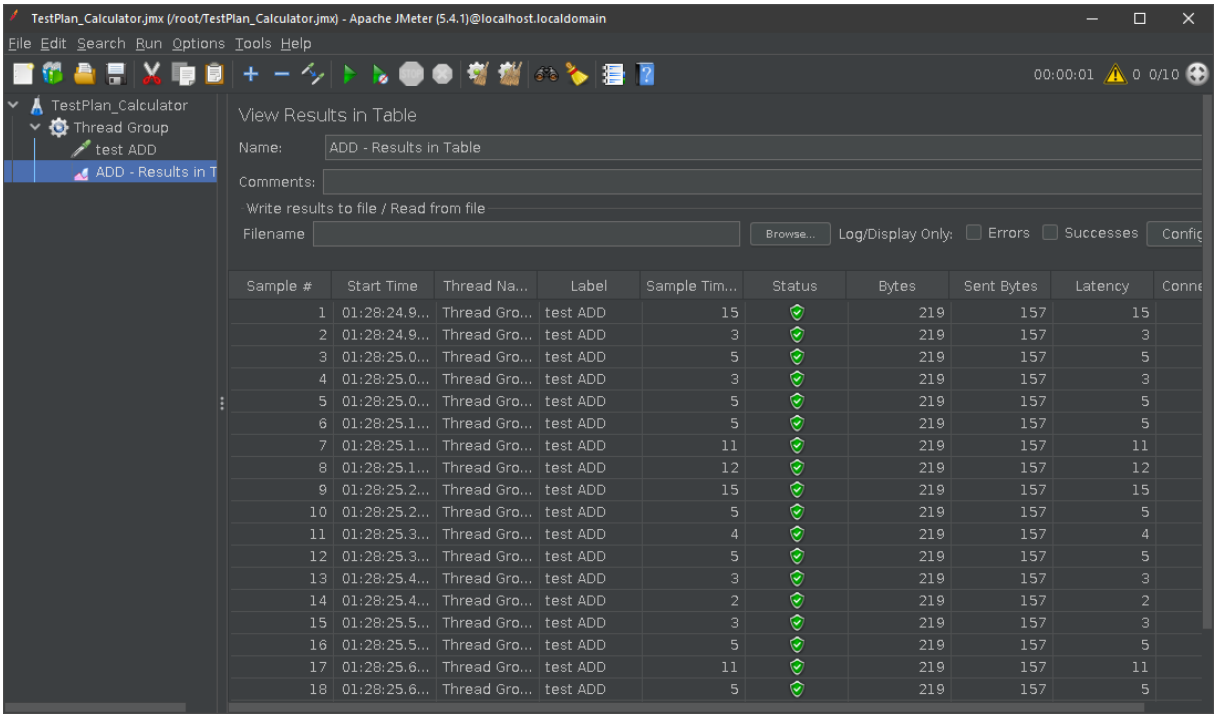
- Paramétrez le test selon les informations appropriées de chaque page :
  - Server Name : localhost (URL/IP du serveur Tomcat)
  - Port Name : 8081 (adaptez le port selon la configuration de votre serveur
  - Path : /calculator/api/calculator/add (le chemin après l'URL du projet)
  - Ajoutez deux paramètres avec des valeurs à intégrer dans le test (x et y)



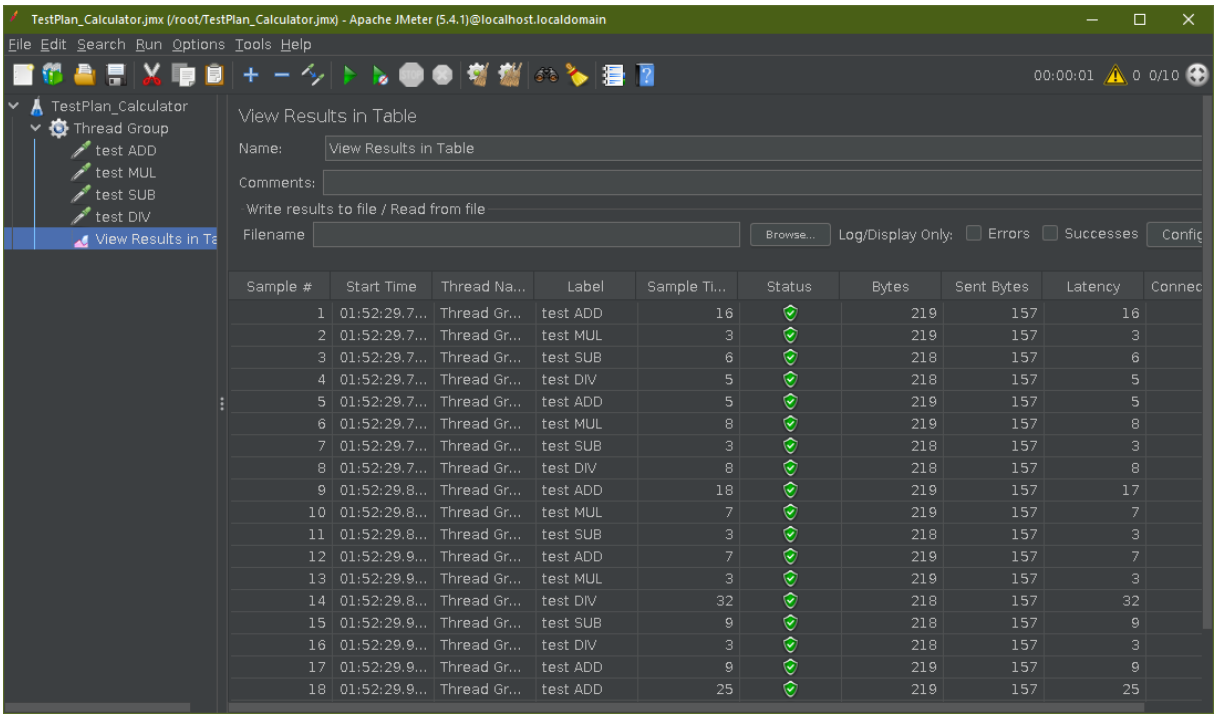
- Ajoutez le Listner pour capturer le résultat de chaque test



- Testez le plan de test et vérifier le bon fonctionnement



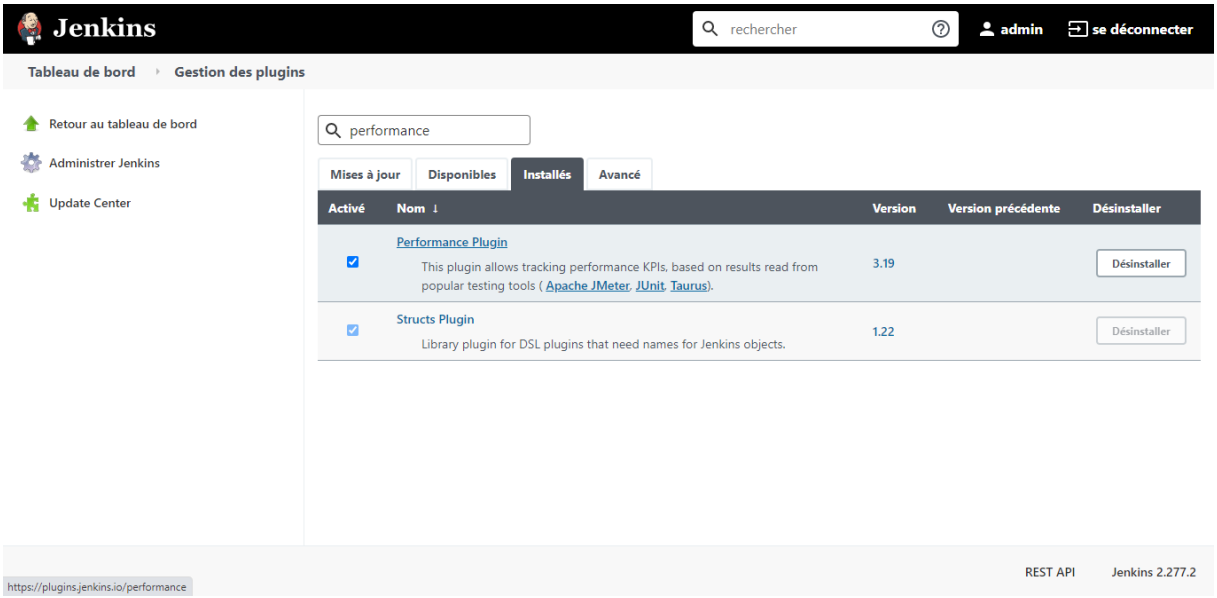
- Ajoutez les tests de performances des autres fonctionnalités (Sub, Mul et Div)


TestPlan_Calculator.jmx (/root/TestPlan_Calculator.jmx) - Apache JMeter (5.4.1)@localhost.localdomain
File Edit Search Run Options Tools Help
00:00:01 0 0/10
TestPlan_Calculator
Thread Group
test ADD
test MUL
test SUB
test DIV
View Results in Ta
View Results in Table
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename
Browse...
Log/Display Only: Errors Successes
Config
Sample # Start Time Thread Na... Label Sample Ti... Status Bytes Sent Bytes Latency Connec
1 01:52:29.7... Thread Gr... test ADD 16 219 157 16
2 01:52:29.7... Thread Gr... test MUL 3 219 157 3
3 01:52:29.7... Thread Gr... test SUB 6 218 157 6
4 01:52:29.7... Thread Gr... test DIV 5 218 157 5
5 01:52:29.7... Thread Gr... test ADD 5 219 157 5
6 01:52:29.7... Thread Gr... test MUL 8 219 157 8
7 01:52:29.7... Thread Gr... test SUB 3 218 157 3
8 01:52:29.7... Thread Gr... test DIV 8 218 157 8
9 01:52:29.8... Thread Gr... test ADD 18 219 157 17
10 01:52:29.8... Thread Gr... test MUL 7 219 157 7
11 01:52:29.8... Thread Gr... test SUB 3 218 157 3
12 01:52:29.9... Thread Gr... test ADD 7 219 157 7
13 01:52:29.9... Thread Gr... test MUL 3 219 157 3
14 01:52:29.8... Thread Gr... test DIV 32 218 157 32
15 01:52:29.9... Thread Gr... test SUB 9 218 157 9
16 01:52:29.9... Thread Gr... test DIV 3 218 157 3
17 01:52:29.9... Thread Gr... test ADD 9 219 157 9
18 01:52:29.9... Thread Gr... test ADD 25 219 157 25

Intégration de Jmeter avec Jenkins

- Installation du plugin « Performance »

`Administrer Jenkins > Gestion de plugins > Disponibles > performance`



- Créez un Job pour lancer le test de performance et interpréter le résultat

General

- **Description** : Test de performance du projet calculator

Gestion de code source : `Git`

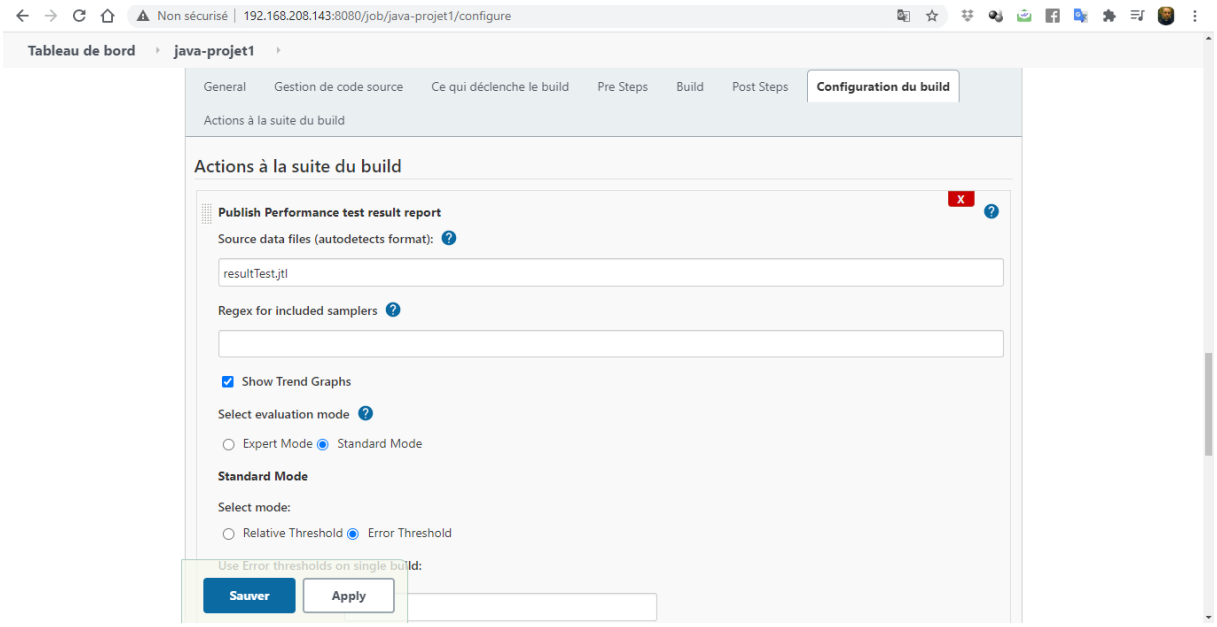
- **Repository URL**: https://gitlab.com/meddeb/calculator.
- **Credentials**: si votre projet est privé, créer votre crédential. Sinon laissez vide.
- **Branch Specifier:**
  - **Pour Github *:*** `main`
  - **Pour Gitlab *:*** `master`

Build : Script shell

```
/opt/jmeter/bin/jmeter -j jmeter.save.saveservice.output_format=xml -n -t src/test/TestPlan_Calculator.jmx -l resultTest.jtl
```

NB : il faut intégrer le plan de test dans le code source de votre projet, sinon modifier le chemin pour pointer vers le fichier source des tests

Actions à la suite du build : `Publish Performance test result report`


Non sécurisé | 192.168.208.143:8080/job/java-projet1/configure
Tableau de bord
java-projet1
General
Gestion de code source
Ce qui déclenche le build
Pre Steps
Build
Post Steps
Configuration du build
Actions à la suite du build
Actions à la suite du build
Publish Performance test result report
Source data files (autodetects format):
resultTest.jtl
Regex for included samplers
Show Trend Graphs
Select evaluation mode
Expert Mode
Standard Mode
Standard Mode
Select mode:
Relative Threshold
Error Threshold
Sauver
Apply