

Guide de l'utilisateur

LOG8430:TP4

**Mohammed Benbachir
Clément Duffau
Guillaume Rivest**

24 Mars 2014

Contenu

Projets de la solution de travail.....	2
Exécution.....	2
Ajouté des commandes au système.....	2

Projets de la solution de travail

Notre solution de travail Eclipse ne contient qu'un seul projet. Ce projet contient le traitement du serveur ainsi que la définition des types de données. C'est également ce projet qui gère la partie Fractal.

Exécution

1. Démarrer l'application Fractal dans Eclipse.
2. Compiler et exécuter avec Ant dans Eclipse.
3. Ouvrir un navigateur web.
4. Entrer l'url du « localhost » avec le bon port et y ajouter le chemin vers le dossier désirer puis finalement, y ajouter la commande à appliquer.
5. Le résultat de la commande demandé s'affichera dans le navigateur.

Ajouter des commandes au système

Afin d'ajouter des commandes au système il faut, au préalable, les avoir compilées en des « .class ». Une fois ceci fait, il faut ajouter ces fichiers dans le dossier « build/commanche ». Ces commandes pourront ensuite être chargées dynamiquement par l'application. Noté qu'il n'y a aucune nécessité de redémarrer l'application pour que ce mécanisme fonctionne.

Exécuter une commande

Afin d'exécuter une commande, l'utilisateur doit connaître le nom de la classe de la commande. Pour l'exécution un pattern a été défini :

URL:NomCommande[,NomCommandeBis]

Ce qui donne par exemple :

<http://localhost:9090/src:CommandeTaille> pour exécuter la commande

CommandeTaille

<http://localhost:9090/src:CommandeTaille,CommandeNombreMots> pour exécuter la commande CommandeTaille et CommandeNombreMots

Exemples de bonne utilisation de l'application

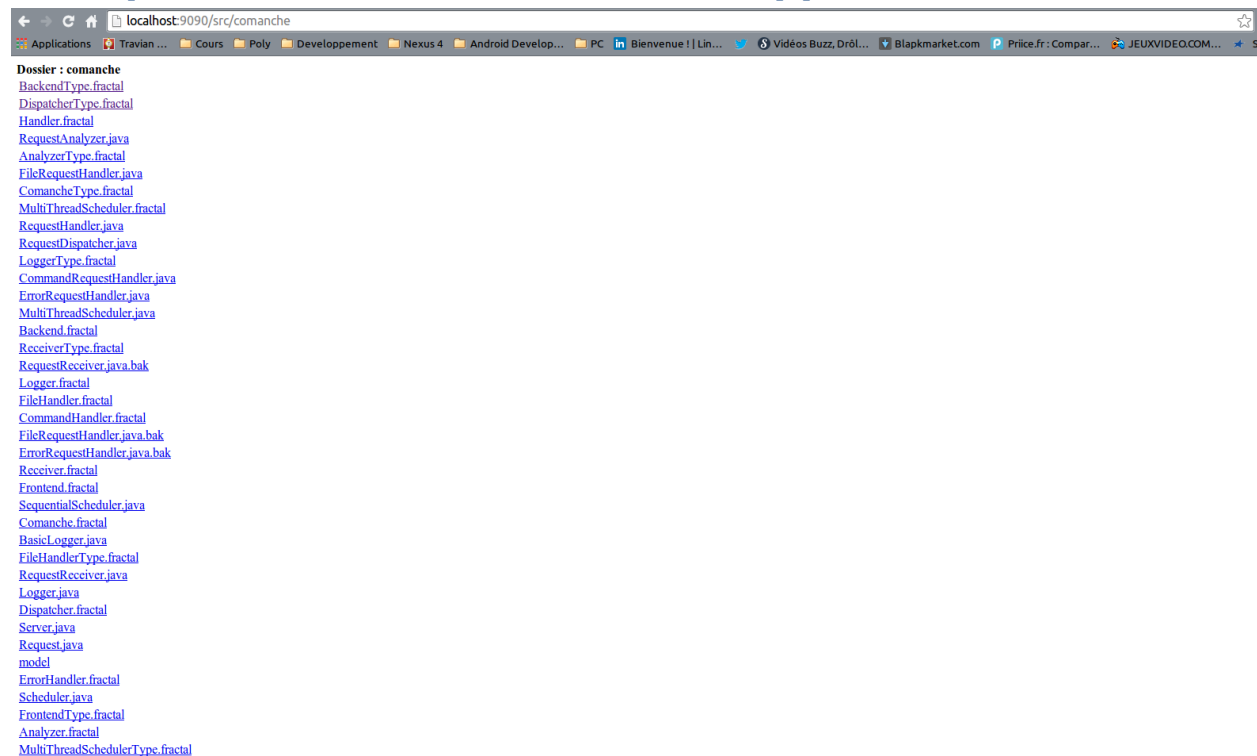


Figure 1 : Exemple sur un dossier

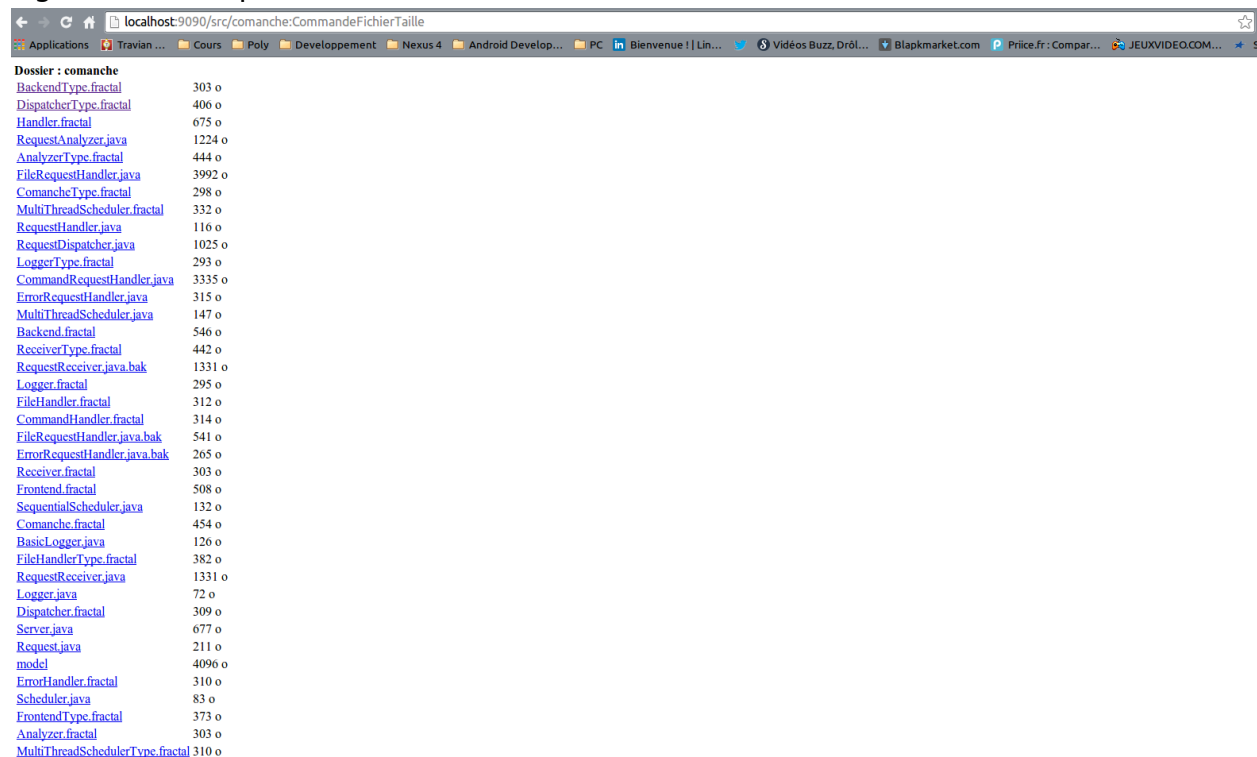
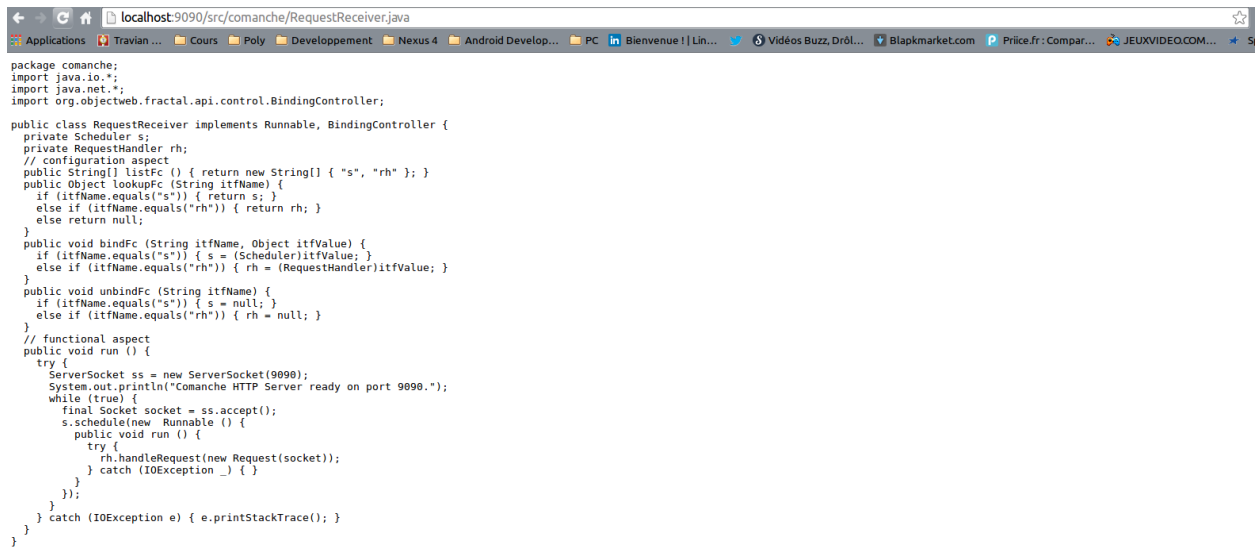


Figure 2 : Exemple sur un dossier avec commande



```
package comanche;
import java.io.*;
import java.net.*;
import org.objectweb.fractal.api.control.BindingController;

public class RequestReceiver implements Runnable, BindingController {
    private Scheduler s;
    private RequestHandler rh;
    // configuration aspect
    public String[] listFc () { return new String[] { "s", "rh" }; }
    public Object lookupFc (String itfName) {
        if (itfName.equals("s")) { return s; }
        else if (itfName.equals("rh")) { return rh; }
        else return null;
    }
    public void bindFc (String itfName, Object itfValue) {
        if (itfName.equals("s")) { s = (Scheduler)itfValue; }
        else if (itfName.equals("rh")) { rh = (RequestHandler)itfValue; }
    }
    public void unbindFc (String itfName) {
        if (itfName.equals("s")) { s = null; }
        else if (itfName.equals("rh")) { rh = null; }
    }
    // functional aspect
    public void run () {
        try {
            ServerSocket ss = new ServerSocket(9090);
            System.out.println("Comanche HTTP Server ready on port 9090.");
            while (true) {
                final Socket socket = ss.accept();
                s.schedule(new Runnable () {
                    public void run () {
                        try {
                            rh.handleRequest(new Request(socket));
                        } catch (IOException _) { }
                    }
                });
            }
        } catch (IOException e) { e.printStackTrace(); }
    }
}
```

Figure 3 : Exemple sur un fichier