Guide du développeur

LOG8430:TP4

**Mohammed Benbachir**

**Clément Duffau**

**Guillaume Rivest**

**24 Mars 2014**

Contenu

[Présentation du guide du développeur 3](#_Toc383298605)

[Utilisation de bibliothèques spécifique 3](#_Toc383298606)

[Explication de l’architecture et Choix de codage 3](#_Toc383298607)

[Questions de l’énoncé 3](#_Toc383298608)

[R2) 3](#_Toc383298609)

[R3) 4](#_Toc383298610)

[R4) 4](#_Toc383298611)

# Présentation du guide du développeur

Ce guide est l’attention des développeurs. On y explique les choix architecturaux, on y présente les librairies et on y répond aux questions posées dans l’énoncé de travail.

# Utilisation de bibliothèques spécifique

Dans le cadre de ce travail, nous avons utilisé la bibliothèque Java « Fractal ». Cette bibliothèque nous a permis d’implémenter une architecture avec serveur qui nous permet d’appliquer des commandes pré-chargées ou chargées dynamiquement sur des fichiers/dossiers via ce serveur.

# Explication de l’architecture et Choix de codage

L’application développée en ce moment agît en serveur en recevant différente commande par l’url et en les redirigeant et en les exécutants. Nous nous sommes donc collés à l’architecture « client-serveur » qui était, peut-on dire, proposé par la base de code. Nous avons également préservé l’utilisation du patron commande pour le design des commandes à appliquer aux dossiers/fichiers passé par l’url.

En termes d’utilisation de Fractal, nous avons choisi un modèle de composant avec liaisons et avec interface. Ceci nous a permis d’intégrer la gestion des commandes directement dans le flux de l’application, et non de le faire d’une manière externe.

Également, dans le projet Eclipse, nous avons la sous paquetage « model » qui contient nos types de donnés pour les commandes dynamiques. Le paquetage principal, quant à lui, contient le traitement du serveur. Il s’agit donc d’une architecture de type MVC, mais sans la partie vue, qui est implémentée en retour de requête, donc afficher dans le navigateur web qui à appeler le serveur.

# Questions de l’énoncé

## R2)

« FileRequestHandler », « RequestDispatcher » et « RequestAnalyzer » sont les composants qui ont été modifiés. Le premier est celui qui gère la commande spécifique implémentée ici, et les deux suivants sont ceux qui gère, analyse et redirige la requête vers le bon « handler », donc notre cas, « FileRequestHandler ».

## R3)

« CommandeRequestHandler » est le composant qui s’occupe de charger les commandes. On vérifie d’abord s’il y a des commandes déjà présentes que l’on pourrait charger lors du démarrage de l’application. Ensuite, par ce composant, on gère l’ajout de nouvelle commande lors de l’exécution continue.

## R4)

Le composant décrit précédemment est intégrer à l’architecture par la classe « Server » qui est le point de départ de l’application et par le « handler » de requête principale qui gère à au niveau les requêtes.