

# ICAR 2008 - Atelier Fractal

[Lionel Seinturier](#)

## Objectif de l'atelier

L'objectif de cet atelier est de mettre en oeuvre les notions d'architectures logicielles à base de composants et de reconfiguration dynamique de ces architectures. Cet atelier utilise le modèle de composants [Fractal](#).

L'archive [atelier-fractal.zip](#) contient l'ensemble des librairies nécessaires au développement avec Fractal, des exemples et des fichiers permettant de réaliser cet atelier. Cette archive est basée sur [fractal-distribution 1.0](#), une distribution prête à l'emploi de Fractal utilisable avec [Ant](#). La page [fractal-distribution](#) fournit les tutoriels et la documentation afférents.

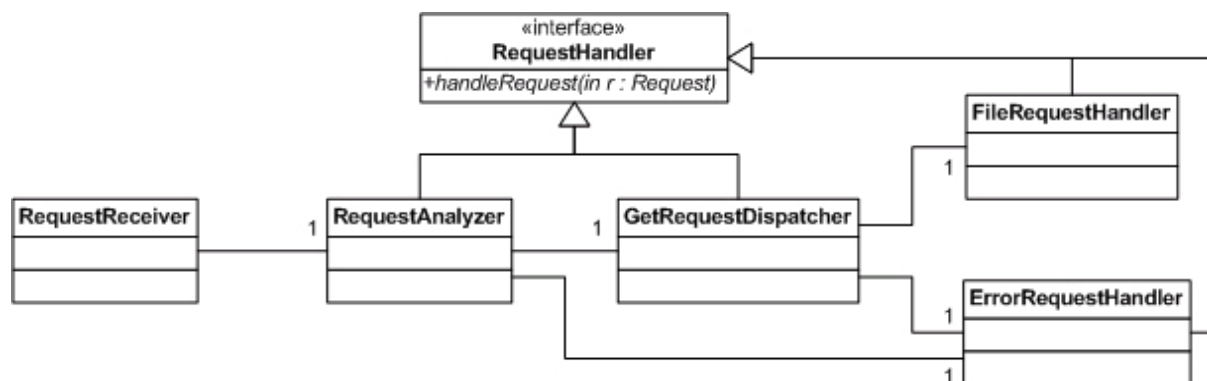
## Exercice 0 : Hello World en Fractal

Le répertoire `helloworld-julia-fractlet-ant-common/` contient un exemple Hello World utilisant [Fractal ADL](#) et [Fractlet](#).

1. Se positionner dans ce répertoire et, pour lancer l'application, lancer la commande: `ant`
2. Examiner les fichiers `.java` et `.fractal` du répertoire `src/helloworld/` pour vous familiariser avec la syntaxe Fractal ADL et Fractlet.
3. Noter, dans le fichier Ant `build.xml`, la façon dont la cible `run` lance l'application:
  - o appel de la classe `org.objectweb.fractal.adl.Launcher` avec
  - o comme paramètre, la référence du fichier `.fractal`: `-fractal helloworld.ClientServerImpl`
  - o comme propriété système, la référence de Julia: `-Dfractal.provider=org.objectweb.fractal.julia.Julia`

## Exercice 1 : Serveur Web en Fractal

Le but de cet exercice est d'illustrer l'intérêt des composants et des architectures logicielles pour la reconfiguration des applications. Soit un serveur Web dont la description est fournie ci-dessous en UML :



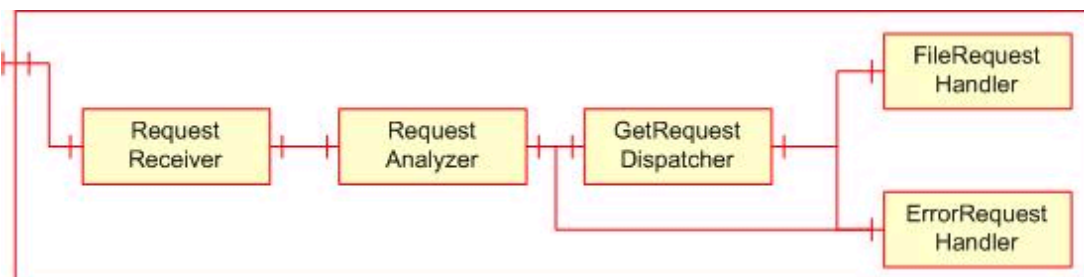
La classe `RequestReceiver` reçoit les requêtes HTTP sur le port 8080 et les transmet à la classe `RequestAnalyzer`. Celle-ci extrait les informations de la requête et la transmet à un *dispatcher*. Il y a un *dispatcher* par type de requête HTTP (GET, POST, PUT, etc.). Sur la figure, la classe `GetRequestDispatcher` correspond au *dispatcher* qui traite les requêtes HTTP de type GET. La classe `RequestAnalyzer` implémente un *pattern* Chaîne de responsabilité et parcourt séquentiellement la liste de *dispatchers* à sa disposition, transmet la requête à chacun d'eux et s'arrête dès qu'un *dispatcher* a su traiter la requête. En bout de liste, la classe `ErrorRequestHandler` génère une erreur pour signaler qu'aucun *dispatcher* n'a su traiter la requête.

La classe `GetRequestDispatcher` gère une liste de gestionnaires. Chaque gestionnaire traite une catégorie de demande. On peut ainsi avoir un gestionnaire de fichiers, un gestionnaire de *servlets*, de scripts PHP, etc. Deux gestionnaires sont représentés : un qui traite les requêtes de demande de fichiers (quel que soit le format du fichier), et un gestionnaire qui signale une erreur. La classe `GetRequestDispatcher` implémente un *pattern* Chaîne de responsabilité et parcourt séquentiellement la liste de gestionnaires à sa disposition, transmet la requête à chacun d'eux et s'arrête dès qu'un gestionnaire a su traiter la requête.

## Étape 1 : Définition des composants et de l'architecture de l'application

Le code du serveur est fourni dans le répertoire `serveur-web/src/web/`. On souhaite obtenir l'architecture suivante.

1. Annoter le afin de définir les composants et écrire le fichier Fractal ADL correspondant à l'architecture représentée ci-dessous.



1. Annoter les classes des composants
2. Écrire les classes des composants
3. Créer un fichier Ant `build.xml` (en s'inspirant de celui de l'exemple précédent)

Lancer le serveur et tester qu'il fonctionne (par exemple en chargeant l'URL <http://localhost:8080/gnu.jpg> dans un navigateur).

## Étape 2 : Ajout d'un composant de traces

On souhaite obtenir une trace des fichiers qui sont retournés par le composant `FileRequestHandler`.

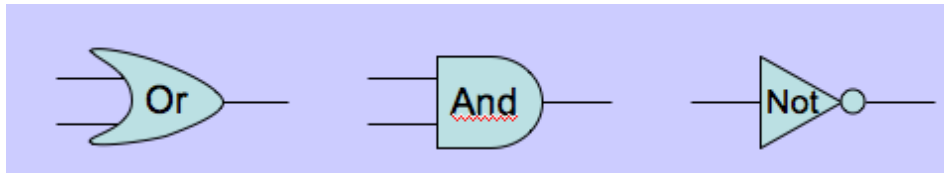
1. Écrire un composant `Log` pouvant être inséré entre les composants `GetRequestDispatcher` et `FileRequestHandler` afin d'afficher à l'écran un *log* des fichiers chargés.
2. Inclure le composant `Log` dans la description de l'architecture.
3. Utiliser la console `FractalExplorer` pour reconfigurer "à la main" l'architecture en retirant le composant `Log` située entre les composants `GetRequestDispatcher` et `FileRequestHandler`.
4. Fournir un composant `AdminRemoveLog` automatisant la procédure de retrait du composant `Log`.
5. Fournir un composant `AdminAddLog` automatisant la procédure d'ajout du composant `Log`.

## Exercice 2 : Composants électroniques

Cet énoncé est repris avec l'aimable autorisation de Sébastien Jean.

### Étape 1 : Composant OR, AND et NOT.

On considère trois composants Fractal représentant des portes logiques OR, AND et NOT. Chaque fil à gauche du composant représente une entrée pouvant contenir un bit et le fil à droite du composant fournit la valeur en sortie.

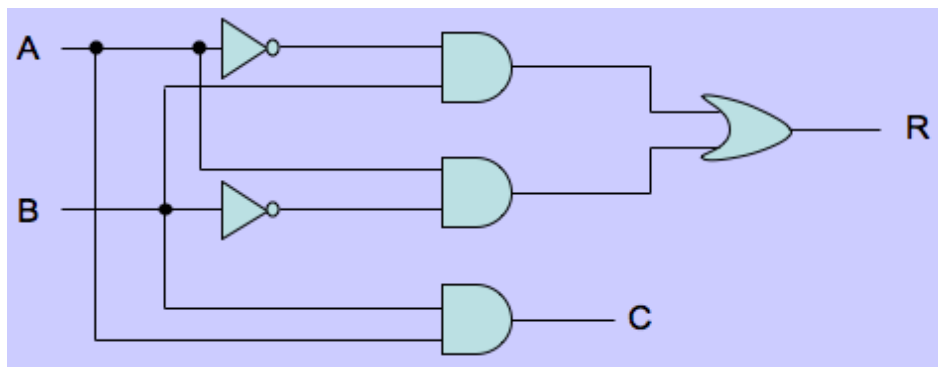


Écrire avec Fractal et Fractal-ADL trois composants représentant respectivement une porte OR, AND et NOT. Fournir pour chacun de ces trois composants un programme de test qui permet d'en vérifier le bon fonctionnement. Vous pourrez être amené à définir un composant Display permettant d'afficher le résultat fourni en sortie par les composants.

### Étape 2 : Composant *Half-Adder*

Définir le composant composite *Half-Adder* contenant les sous-composants connectés selon le schéma ci-dessous. Le composant *Half-Adder* a deux entrées (A et B) et deux sorties R (*result*) et C (*carry*). Les points dans le schéma représentent des multiplexeurs : la valeur fournie en entrée est répliquée sur l'ensemble des fils de sortie.

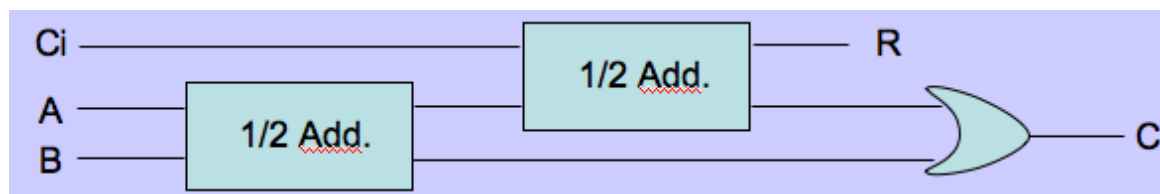
Écrire avec Fractal et Fractal ADL les composants multiplexeurs et *Half-Adder*. Fournir des programmes de test permettant de vérifier le bon fonctionnement de chacun de ces deux composants.



### Étape 3 : Composant *Adder*

Soit le composant *Adder* représenté ci-dessous et résultant de l'assemblage de deux composants *Half-Adder* et d'un composant OR. Le composant *Adder* possède trois fils d'entrée (Ci, A et B) pouvant contenir chacun 1 bit et deux fils de sortie (R et C).

Écrire le composant *Adder*. Fournir un programme permettant d'en vérifier le bon fonctionnement.



---

[Lionel Seinturier](#). 21/08/08.