

# Mr. StataCookie

## Artifactory & Jenkins

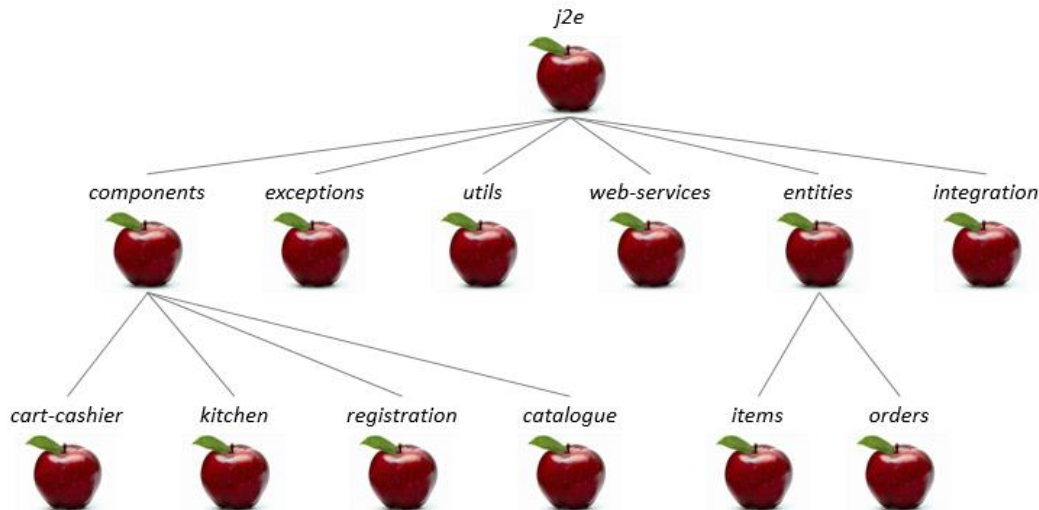
L'objectif du TD est de transformer l'architecture monolithique du projet Cookie Factory en une architecture modulaire de façon à rendre sa modification plus flexible.

### I. Architecture

Pour implémenter une telle architecture, nous décomposerons le projet initial en plusieurs modules (artéfacts) et nous utiliserons les outils Maven, Artifactory et Jenkins.

#### 1. Découpage en modules

Le découpage en modules s'effectue au niveau du serveur J2E. Voici celui que nous implémenterons.



*Découpage du serveur J2E en modules*

Ce découpage a été réalisé de façon à n'avoir aucun cycle de dépendances entre les modules. Vous trouverez en document annexe le contenu de chacun des modules.

## 2. Artifactory

Artifactory est un repository manager. Il nous permettra de stocker le build de chacun des modules et de les mettre à disposition de Jenkins et des utilisateurs ayant besoin des dépendances manquantes.

La configuration de Artifactory correspond simplement en la création d'un dépôt de type Maven pour le stockage des modules. L'outil traitera ensuite automatiquement la sauvegarde des builds de chaque module.

## 3. Jenkins

Jenkins est un outil d'intégration continue dérivé de Hudson fonctionnant avec le web container Apache TomCat. Nous l'utiliserons pour automatiser les builds à chaque push vers le dépôt contenant le projet. Il enverra les builds en succès vers le dépôt d'Artifactory.

Jenkins utilise un système de « job » qui correspond à une tâche à effectuer. Nous utiliserons principalement 3 jobs :

- un job pour compiler les modules du j2e
- un job pour les tests d'intégration
- un job pour le lancement et l'arrêt du server .Net.

Jenkins est configuré de façon à :

- pouvoir exécuter 2 jobs simultanément
- recompiler seulement les modules qui ont été modifiés et les modules qui en dépendent
- envoyer seulement les builds en succès vers Artifactory

## II. Forces & Faiblesses

Dans cette partie nous aborderons les forces et faiblesses de l'architecture utilisée, comparée à la structure monolithique de départ.

### 1. Points forts

#### Modularité

- Seules les modules modifiés et les modules qui en dépendent vont être recompilés (Maven)  
=> gain de temps (à développer)
- Pas besoin d'avoir le projet tout entier pour travailler puisque les dépendances sont téléchargées depuis Artifactory (à développer)
- Nous avons un projet fortement découpé. Plus le projet est découpé en modules, plus nous réduisons la récupération de ressources dont nous n'avons pas besoin. (à développer)

## Fiabilité

- Lorsqu'un module est push alors qu'il ne compile pas ou qu'un des tests ne passe pas, le module n'est pas push vers Artifactory grâce à Jenkins. Les erreurs des uns n'empêchent donc pas les autres de travailler puisque Artifactory disposera et desservira toujours d'un module fonctionnel.

## Gestion de la mémoire

- En raison de l'utilisation d'une machine virtuelle, nous disposons d'une capacité de mémoire limitée. C'est pourquoi nous avons configuré Jenkins de façon à ce qu'il ne garde que les dix derniers builds (dont le dernier build en succès).

## Intégration continue

- Les tests d'intégrations sont effectués régulièrement pour assurer la cohésion entre les différents serveurs.
- Jenkins est configuré de façon à ce que le serveur dotNet soit lancé avant chaque exécution des tests d'intégration, et arrêté ensuite. (dire pourquoi est-ce mieux que s'il était lancé indéfiniment)

## 2. Points faibles

### Modification non prise en compte

- Si nous modifions un module A et mettons un test d'un module B en erreur (A et B indépendant), le nouveau module A n'est pas push sur Artifactory à cause de l'erreur de B. (à développer)

### Attente active

- Jenkins scrute le projet toute les minutes pour recompiler les modules modifiés. Si nous avons beaucoup de jobs qui font ça, la bande-passante est saturée.
- S'il y a plusieurs push entre 2 scrutations et qu'un des push provoque une erreur de compilation ou de test, rien n'est envoyé sur Artifactory.

### Plus long

- Le téléchargement des dépendances rallonge le temps de build comparé au code monolithique (à développer)

### Gestion de la mémoire

- Contrairement à Jenkins, Artifactory garde tous les builds, la mémoire est n'est donc pas nettoyée. (à développer)

### Release ??

- Artifactory ne contient pas de release (cf Snapshot) (Kévin !!)