



Conception d'Applications Hétérogènes Distribuées

Master 2 Traitement de l'Information

Lionel Médini

Septembre-novembre 2015

Présentation de l'UE

- Positionnement
 - Suite de MIF17 (et un peu MIF13)
 - Couplée avec TIW5 (services Web)
- Format
 - Cours court
 - TP (parfois long)
- Évaluation
 - TPs
 - Examen

Objectifs généraux

- Être rapidement employable
 - Ne plus concevoir “from scratch”
 - Se familiariser avec des outils utilisés dans l’industrie
- S’insérer dans un SI existant
 - Appréhender la complexité des SI
 - Hétérogènes
 - Distribués

Objectifs méthodologiques

- Comprendre le fonctionnement interne des outils les plus répandus
 - Patterns sous-jacents
 - Fonctions des différentes “briques”
 - Spécificités
- Savoir prendre du recul pour
 - Factoriser
 - S’adapter

Thèmes abordés

- Paradigmes de programmation
 - Patrons de conception
 - Inversion de Contrôle
 - Contexte, Annuaire
 - Pooling...
 - Programmation par composants
 - Du POJO aux objets transactionnels distribués
 - Composants dynamiques
 - Micro-services

Thèmes abordés

- SI distribués
 - Principes de communication entre objets
 - Quelques frameworks existants
- Introduction à l'urbanisation des SI
 - La métaphore de la ville
 - Référentiels et outils existants

Plan du cours

- Retour sur les patrons de conception
 - Inversion de contrôle (conteneurs d'objets)
 - Contexte et annuaire (communication dans un conteneur)
 - Réserve d'instances (gestion du cycle de vie)
- Systèmes d'information distribués
 - Langage de spécification d'interface (IDL)
 - Appel de méthodes distantes (RPC, CORBA, RMI)

Plan du cours

- Aperçu de quelques outils
 - Conteneurs légers (Spring)
 - Conteneurs “lourds” (Java EE)
 - Composants déployables “à chaud” (OSGi)
 - Micro-services (Docker)
- Introduction à l’urbanisation des SI
 - Urbanisme
 - Urbanisation
 - Gouvernance

Rappels sur les design patterns

Plan

- Rappels sur les patrons de conception
- Inversion de contrôle
- Conteneurs
- Injection de dépendances
- Frameworks

- Composantes d'un patron
 - Nom : évocation de la solution
 - Problème à solutionner
 - Contexte d'application du patron et limites de la solution
 - Forces/contraintes de la solution par rapport au contexte
 - Solution mise en oeuvre (avec variantes éventuelles)

L'inversion de contrôle (IoC)

Plan

Rappels sur les patrons de conception

➤ Inversion de contrôle

Conteneurs

Injection de dépendances

Frameworks

- Problème
 - Réduire les dépendances (couplage) entre des objets dont l'implémentation peut varier
 - Diminuer la complexité de gestion du cycle de vie de ces objets (patterns singleton et factory)
- Principe
 - S'appuie sur le pattern d'indirection
 - Le contrôle du flot d'exécution d'une application n'est plus géré par l'application elle-même mais par une structure externe (conteneur)

L'inversion de contrôle (IoC)

Plan

Rappels sur les patrons de conception

➤ Inversion de contrôle

Conteneurs

Injection de dépendances

Frameworks

- Définition (M. Fowler)

L'IoC différencie un framework d'une bibliothèque logicielle

- Exemples

- Interface à modèle événementiel (Swing)
- Serveur Web
- Conteneurs d'objets (servlets, EJB)

- Autres noms

- Recherche / Injection de dépendances
- Injection de code

L'inversion de contrôle (IoC)

Plan

Rappels sur les patrons de conception

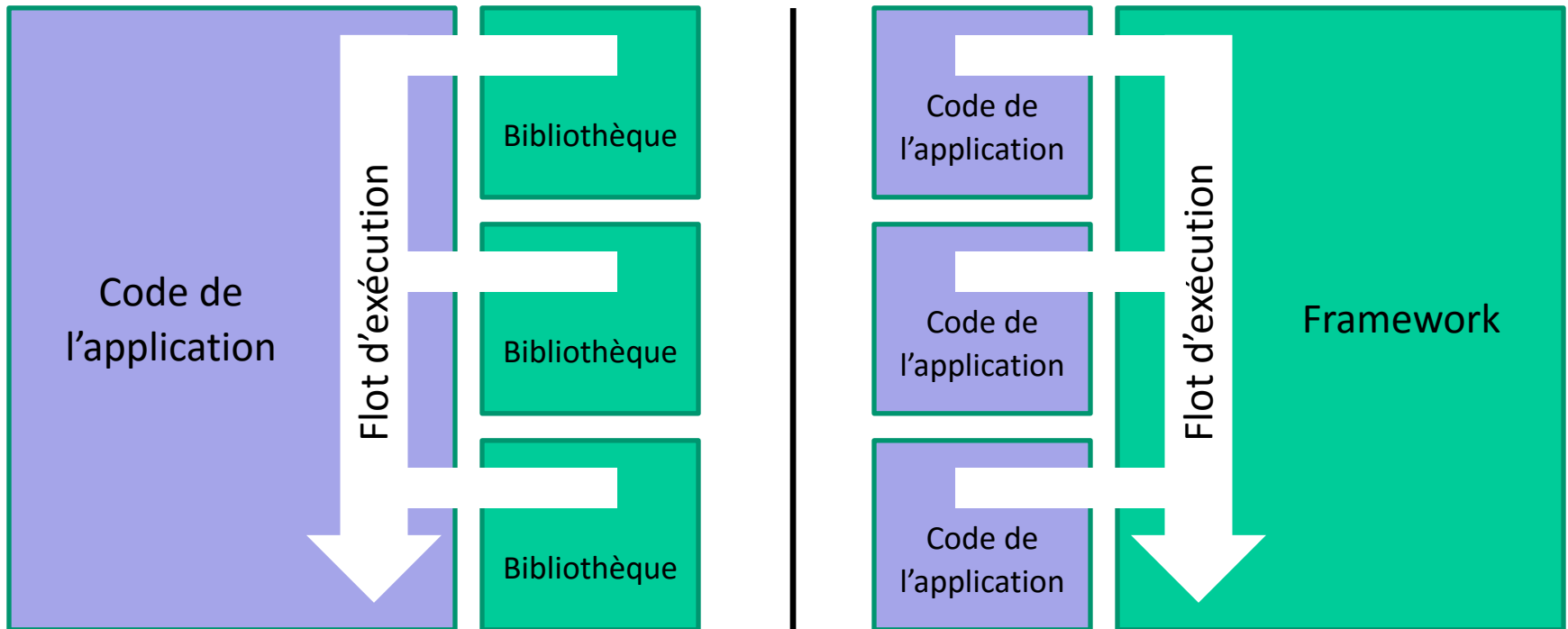
➤ Inversion de contrôle

Conteneurs

Injection de dépendances

Frameworks

- Différence bibliothèque / framework



- Remarque : dans la littérature, on trouve l'appellation « framework » pour beaucoup de choses qui n'en sont pas

L'inversion de contrôle (IoC)

Plan

Rappels sur les patrons de conception

➤ Inversion de contrôle

Conteneurs

Injection de dépendances

Frameworks

- Fonctionnement
 - Un framework initialise un (ou plusieurs) **conteneur(s)**
 - Un conteneur communique avec des services extérieurs et avec les objets de l'application
 - Les objets programmés sont les **composants** de l'application
 - Le conteneur **instancie, isole, utilise et détruit** ces composants

Conteneur : définition

Plan

Rappels sur les patrons de conception

Inversion de contrôle

> Conteneurs

Injection de dépendances

Frameworks

- Le conteneur contient le “main” de l’application
- Il a pour but de
 - Permettre le bon fonctionnement des composants de l’application
 - Gérer leur cycle de vie
 - Injecter les dépendances
 - Isoler ces composants (pattern façade)
 - Vis-à-vis de l’extérieur : il communique avec les services extérieurs et filtre les entrées / sorties de l’application
 - Entre eux : il peut permettre de séparer les couches de l’application (ex : MVC)

Conteneur : instanciation

Plan

Rappels sur les patrons de conception

Inversion de contrôle

➤ Conteneurs

Injection de dépendances

Frameworks

- Instanciation par le framework avant les objets de l'application
- 2 techniques (non incompatibles)
 - Inversion de contrôle par configuration (statique)
 - Fichiers de configuration (XML)
 - Annotations (Java)
 - Inversion de contrôle dynamique
 - Le conteneur est un objet d'une application (framework)
 - Il peut être paramétré programmatiquement

Conteneur : initialisation

Plan

Rappels sur les patrons de conception

Inversion de contrôle

> Conteneurs

Injection de dépendances

Frameworks

- Le conteneur recherche dans sa configuration les composants à instancier
- Il peut résoudre un **référentiel de dépendances** et rechercher les dépendances à injecter dans chaque composant
- Il peut isoler les composants à l'aide d'un **contexte d'application**
- Il appelle les méthodes des composants en fonction des besoins de l'application

Conteneurs : fonctionnement

Plan

Rappels sur les patrons de conception

Inversion de contrôle

> Conteneurs

Injection de dépendances

Frameworks

- Gestion des composants par le conteneur
 - Appel de méthodes spécifiques des objets par le conteneur pour contrôler leurs changements d'états
 - Méthodes de gestion du cycle de vie
 - Méthodes de service
 - Méthodes formalisées par des interfaces ou par le paramétrage du conteneur
 - Exemples : doGet(), doPost(), EJBCreate(), EJBActivate()...

Conteneurs : fonctionnement

Plan

Rappels sur les patrons de conception

Inversion de contrôle

> Conteneurs

Injection de dépendances

Frameworks

- Gestion explicite du cycle de vie des objets
 - Création (factory)
 - Instanciation / destruction
 - Création + configuration (builder)
 - Dépendances entre composants
 - Cas particulier : gestion des singletons
 - Nombre d'instances créées géré par le conteneur
 - Déclaration d'une classe comme singleton dans les paramètres de configuration du conteneur
 - Permet de s'abstraire d'un type d'implémentation particulier (classe abstraite, constructeur privé)

Exemple : conteneur Web

Plan

Rappels sur les patrons de conception

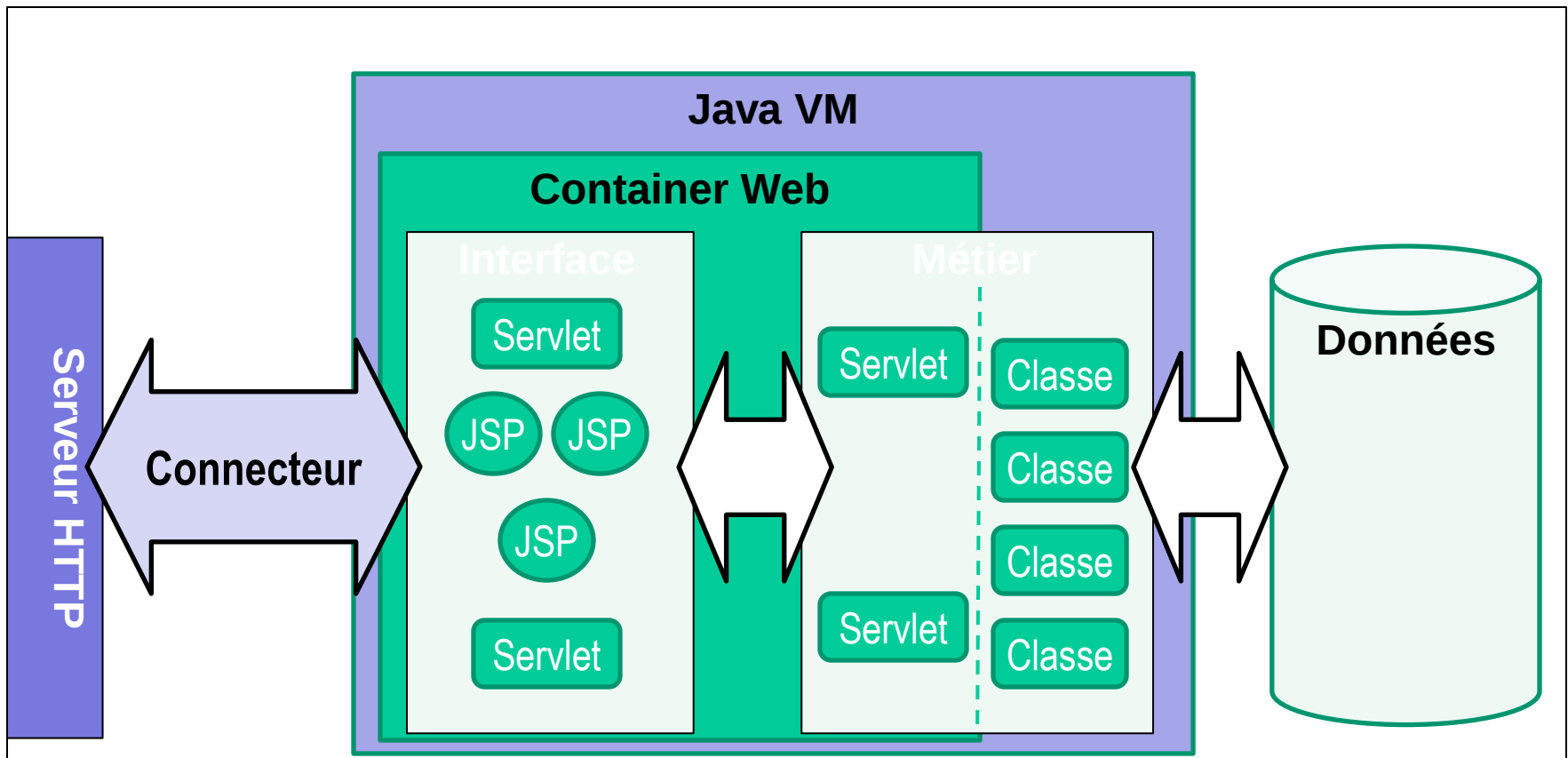
Inversion de contrôle

➤ Conteneurs

Injection de dépendances

Frameworks

- Programmation côté serveur en Java



Conteneurs : autres exemples

Plan

Rappels sur les patrons de conception

Inversion de contrôle

➤ Conteneurs

Injection de dépendances

Frameworks

- Conteneurs légers (Spring, Pico, NanoContainer)
 - Ne contiennent que les services nécessaires
- Conteneurs lourds (conteneurs d'EJB)
 - Sont eux-mêmes des patterns singletons
 - Possèdent un large éventail de fonctionnalités
- Conteneurs dynamiques (OSGi)
 - Permettent de modifier les dépendances entre les composants pendant le déroulement de l'application

Injection de dépendances

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

➤ Injection de dépendances

Frameworks

- Objectif
 - Rendre l'inversion de contrôle transparente pour les objets
- Fonctionnement
 - Un référentiel de dépendances décrit explicitement les liens des objets entre eux et avec le conteneur
 - Le conteneur résout ce référentiel
 - Le conteneur instancie les composants
 - Éventuellement dans un ordre compatible avec le référentiel
 - Le conteneur utilise une ou plusieurs méthodes d'injection pour injecter les dépendances entre les composants

Injection de dépendances

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

➤ Injection de dépendances

Frameworks

- Méthodes d'injection de dépendances

- Injection par constructeur

- Passage d'une référence aux objets connus dans le constructeur

- Exemple

```
public ObjetA {  
    ObjetB objb;  
    public ObjetA (ObjetB o) { (...) objb = o; (...) } (...) }  
}
```

- Avantage : conforme aux bonnes pratiques de la POO

- Inconvénients

- Paramètres du constructeur non nommés mais ordonnés
 - Devient fouillis quand il y a de nombreux paramètres
 - Pas d'héritage de cette configuration entre les objets

Injection de dépendances

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

> Injection de dépendances

Frameworks

- Méthodes d'injection de dépendances
 - Injection par modificateurs
 - Utilise les modificateurs (setters) des attributs des objets
 - Exemple

```
public ObjetA {  
    ObjetB objb;  
    public setObjb (ObjetB o) { objb = o; } (...) }
```
 - Avantages
 - Apparition explicite des noms des dépendances
 - Les modificateurs peuvent effectuer des opérations complexes
 - Inconvénient : non conforme à la POO (l'initialisation « sort » du constructeur)

Injection de dépendances

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

➤ Injection de dépendances

Frameworks

- Méthodes d'injection de dépendances
 - Injection par interface
 - Chaque objet implémente autant d'interfaces que de dépendances
 - Chaque interface définit une méthode publique d'injection
 - Exemple

```
public class ObjetA implements InjectObjetB, ... {
    ObjetB objb;
    public void injectObjetB(ObjetB o) { objb = o; } (... ) }
```
 - Mêmes avantages et inconvénients que les setters
 - Inconvénient supplémentaire
 - forme du code imposé assez lourde
 - lisibilité délicate

Injection de dépendances

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

➤ Injection de dépendances

Frameworks

- Méthodes d'injection de dépendances
 - Par proxy
 - Un proxy intercepte l'appel au constructeur d'un objet
 - Il réalise à la fois la création et l'injection de dépendances
 - Par constructeur
 - Par modificateurs
 - Avantages
 - Avantages de la méthode d'injection employée
 - Découplage code métier / injection de dépendances grâce au proxy
 - Inconvénients
 - Inconvénients de la méthode d'injection employée
 - Possible baisse de performances due à l'introduction du proxy

Injection de dépendances

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

➤ Injection de dépendances

Frameworks

- Pourquoi / comment réaliser de l'injection de dépendances sur des valeurs et non sur des objets ?

Les frameworks

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

Injection de dépendances

➤ Frameworks

- Définition
 - Outil qui contrôle le flot de déroulement de l'application
 - Exemples : serveur Web, frameworks Web MVC, serveur d'applications...
- Composants
 - Conteneur(s) d'objets
 - Contexte(s) applicatifs
 - MVC
 - Mécanismes de configuration des applications
 - Services annexes (logs, sécurité, transactions...)

Les frameworks

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

Injection de dépendances

> Frameworks

- Intérêt pour le programmeur
 - Évitent de reprogrammer les fonctionnalités récurrentes
 - De nombreux services déjà disponibles
 - Respectent les règles de bonnes pratiques
 - Compréhensibilité et réutilisabilité des modules de l'application
- Contraintes : pour bien utiliser un framework, il faut
 - En comprendre la “philosophie” (finalité, limites)
 - En respecter les règles (API)
- Rappel
 - Ne pas confondre avec une bibliothèque (composants annexes appelés par le programme)

Les frameworks

MVC

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

Injection de dépendances

➤ Frameworks

- Exemples de frameworks
 - Symfony
 - Struts, Spring, Java EE
 - .Net
 - Zope
 - ...
- Choix d'une solution
 - dépend des caractéristiques de l'application
 - dépend des autres responsabilités du contrôleur

Références

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

➤ Injection de dépendances

Frameworks

• Ouvrages

- E. Gamma, R. Helm, R. Johnson, J. Vlissides (1994), Design patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley, 395 p.
 - Traduction française : Design patterns. Catalogue des modèles de conception réutilisables, Vuibert 1999
- Martin Fowler (2002) Patterns of Enterprise Application Architecture, Addison Wesley
- E. Freeman, K. Sierra , B. Bates , M-C. Baland (2005), Tête la première – Design Patterns, O'Reilly Eds
- Robert-C Martin (2009), Coder Proprement, Pearson Education

Références

Plan

Rappels sur les patrons de conception

Inversion de contrôle

Conteneurs

Injection de dépendances

Frameworks

- Design patterns

<http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html>

<http://hillside.net/patterns/onlinepatterncatalog.htm>

<http://www.martinfowler.com/>

<http://liris.cnrs.fr/~lmedini/enseignement/MIF17/CM-patterns.pdf>

<http://www.dotnetguru.org/articles/articlets/contextPattern/Contexte.htm>

- Conteneurs

<http://www.dotnetguru.org/articles/dossiers/ioc/ioc.htm>

<http://www.picocontainer.org/>

<http://www.nanocontainer.org/>