Référentiel Coordonnées JAVA

Architecture et solutions techniques

Thomas Duchatelle (thomas.duchatelle-ext@yrnet.com)

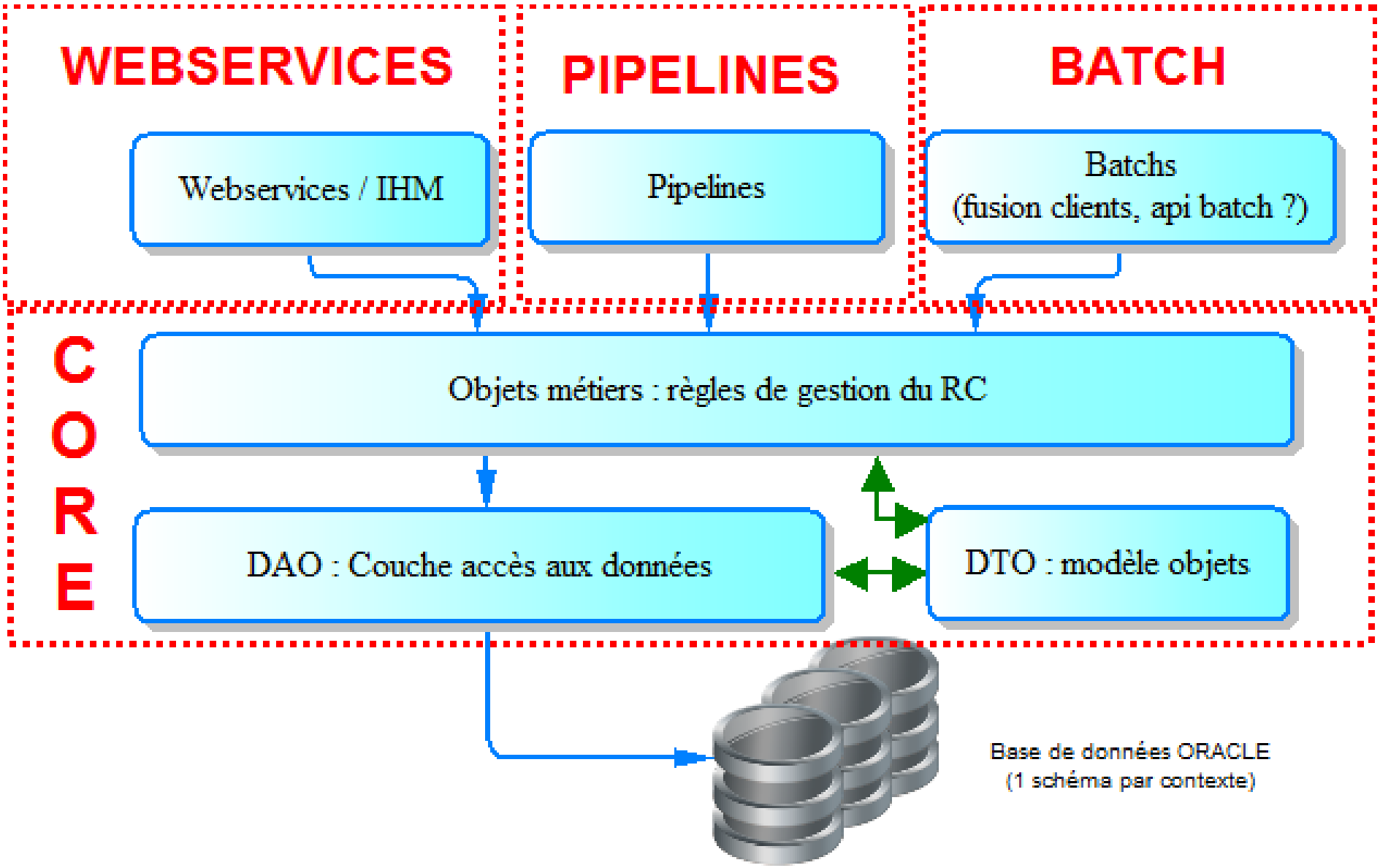
Yves Rocher

February 13, 2013

Sommaire

Référentiel Coordonnées

1 application, 4 modules



Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Pipelines Intégration de données par fichiers CSV (jar exécutable)

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Pipelines Intégration de données par fichiers CSV (jar exécutable)

- Démarrage/Arrêt d'un *démon*

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Pipelines Intégration de données par fichiers CSV (jar exécutable)

- Démarrage/Arrêt d'un *démon*
- Détection de l'arrivée de fichiers

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Pipelines Intégration de données par fichiers CSV (jar exécutable)

- Démarrage/Arrêt d'un *démon*
- Détection de l'arrivée de fichiers
- EAI et lecture de fichier

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Pipelines Intégration de données par fichiers CSV (jar exécutable)

- Démarrage/Arrêt d'un *démon*
- Détection de l'arrivée de fichiers
- EAI et lecture de fichier
- Parallélisation des traitements

Modules du référentiel coordonnées

Core Coeur de l'application (jar) :

- Modèle objets : représentation *client* / *coordonnées*
- couche d'accès au données : structure de la base, gestion des schémas
- règles de gestion : calcul des consentements, règles de mises à jour, dé-duplication

Webservices Application Java EE (war) exposant des webservices

- correspondance *modèle objet interne* $\leftrightarrow$ *contrat WSDL*
- Interface WEB : suivi des pipelines

Pipelines Intégration de données par fichiers CSV (jar exécutable)

- Démarrage/Arrêt d'un *démon*
- Détection de l'arrivée de fichiers
- EAI et lecture de fichier
- Parallélisation des traitements

Batch Exports, traitements de types : fusion, statistiques, ...

Organisation des modules

En utilisant *Maven*

Maven

Apache Maven est un logiciel de gestion de projet. Basé sur le concept de *Project Object Model* (POM), il gère le processus de compilation, rapports, ...

Organisation des modules
En utilisant *Maven*

Maven

Apache Maven est un logiciel de gestion de projet. Basé sur le concept de *Project Object Model* (POM), il gère le processus de compilation, rapports, ...

Le Référentiel Coordonnées utilise *Maven* pour :

- la gestion des dépendances
- le packaging : compilation, tests automatique, archives zip et ear
- le déploiement des pipelines et batch (FTP)
- configuration du poste de travail : *Eclipse*
- tests en local des Webservices et IHM

Projets et sous-projets *Maven*

Le référentiel coordonnées est répartie sur 6 projets *Maven* :

- **addressrepository** : *version, dépendances, modules (pom)*

Projets et sous-projets *Maven*

Le référentiel coordonnées est répartie sur 6 projets *Maven* :

- **addressrepository** : *version, dépendances, modules (pom)*
  - **address-core** : *coeur de l'application* (jar)
    package : `net.yvesrocher.services.address.core`

Projets et sous-projets *Maven*

Le référentiel coordonnées est répartie sur 6 projets *Maven* :

- **addressrepository** : *version, dépendances, modules (pom)*
  - **address-core** : *coeur de l'application* (jar)
    package : `net.yvesrocher.services.address.core`
  - **address-pipelines** : *démon (jar exécutable)*
    package : `net.yvesrocher.services.address.pipelines`

Projets et sous-projets *Maven*

Le référentiel coordonnées est répartie sur 6 projets *Maven* :

- **addressrepository** : *version, dépendances, modules (pom)*
  - **address-core** : *coeur de l'application* (jar)
    package : `net.yvesrocher.services.address.core`
  - **address-pipelines** : *démon (jar exécutable)*
    package : `net.yvesrocher.services.address.pipelines`
  - **address-webservices** : *War du webservices*
    package : `net.yvesrocher.services.address.webservices`

Projets et sous-projets *Maven*

Le référentiel coordonnées est répartie sur 6 projets *Maven* :

- **addressrepository** : *version, dépendances, modules (pom)*
  - **address-core** : *coeur de l'application* (jar)
    package : `net.yvesrocher.services.address.core`
  - **address-pipelines** : *démon (jar exécutable)*
    package : `net.yvesrocher.services.address.pipelines`
  - **address-webservices** : *War du webservices*
    package : `net.yvesrocher.services.address.webservices`
  - **address-ear** : *Package le war en un EAR*

Projets et sous-projets *Maven*

Le référentiel coordonnées est répartie sur 6 projets *Maven* :

- **addressrepository** : *version, dépendances, modules (pom)*
  - **address-core** : *coeur de l'application* (jar)
    package : `net.yvesrocher.services.address.core`
  - **address-pipelines** : *démon (jar exécutable)*
    package : `net.yvesrocher.services.address.pipelines`
  - **address-webservices** : *War du webservices*
    package : `net.yvesrocher.services.address.webservices`
  - **address-ear** : *Package le war en un EAR*
  - **address-batch** : *Batchs utilisant le coeur V3 (jar exécutable)*
    package : `net.yvesrocher.services.address.batch`

Outils et frameworks utilisés

Spring Inversion de contrôle, gestion des contextes, transactions

BDD

Outils et frameworks utilisés

Spring	Inversion de contrôle, gestion des contextes, transactions BDD
Hibernate	*Mapping Relationnel Objet*

Outils et frameworks utilisés

Spring	Inversion de contrôle, gestion des contextes, transactions BDD
Hibernate	*Mapping Relationnel Objet*
Hibernate Validator	Validation des données

Outils et frameworks utilisés

Spring	Inversion de contrôle, gestion des contextes, transactions BDD
Hibernate	*Mapping Relationnel Objet*
Hibernate Validator	Validation des données
SLF4J	API de log, utilise *LOG4J* en backend

Outils et frameworks utilisés

Spring	Inversion de contrôle, gestion des contextes, transactions BDD
Hibernate	*Mapping Relationnel Objet*
Hibernate Validator	Validation des données
SLF4J	API de log, utilise *LOG4J* en backend
Commons Apache	Utilitaires sur les chaines de caractères, la détection des fichiers, ...

Outils et frameworks utilisés

Spring	Inversion de contrôle, gestion des contextes, transactions BDD
Hibernate	*Mapping Relationnel Objet*
Hibernate Validator	Validation des données
SLF4J	API de log, utilise *LOG4J* en backend
Commons Apache	Utilitaires sur les chaines de caractères, la détection des fichiers, ...
JUNIT	Tests unitaires (utilisé avec *Mockito* et *FestAssert*)

Compilation et déploiement

Compilation

Maven exécute les tests unitaires et crée les binaires s'il n'y a pas d'erreur.
Les binaires sont présents dans le répertoire `target` de chaque module.

Compilation et déploiement

Compilation

Maven exécute les tests unitaires et crée les binaires s'il n'y a pas d'erreur.
Les binaires sont présents dans le répertoire `target` de chaque module.

Commande Maven de compilation et déploiement sur FTP :

```
1  mvn clean install -DdeployPipelines
```

Sommaire

Inversion de Contrôle

Avec *Spring* !

Spring

Spring fournit l'inversion de contrôle : cycle de vie des *beans*, injection de dépendances et gestion de la configuration.

Inversion de Contrôle

Avec *Spring* !

Spring

Spring fournit l'inversion de contrôle : cycle de vie des *beans*, injection de dépendances et gestion de la configuration.

Les fichiers de configuration se trouvent dans `src/main/resources` :

- `spring` : fichiers de configuration *Spring* :

Inversion de Contrôle

Avec *Spring* !

Spring

Spring fournit l'inversion de contrôle : cycle de vie des *beans*, injection de dépendances et gestion de la configuration.

Les fichiers de configuration se trouvent dans `src/main/resources` :

- `spring` : fichiers de configuration *Spring* :
  - `addressrepository-core.xml` : inclue les fichiers nécessaires

Inversion de Contrôle

Avec *Spring* !

Spring

Spring fournit l'inversion de contrôle : cycle de vie des *beans*, injection de dépendances et gestion de la configuration.

Les fichiers de configuration se trouvent dans `src/main/resources` :

- `spring` : fichiers de configuration *Spring* :
  - `addressrepository-core.xml` : inclue les fichiers nécessaires
  - `addressrepository-businessservice.xml` : paramètre le coeur pour être intégré à un autre Webservice

Inversion de Contrôle

Avec *Spring* !

Spring

Spring fournit l'inversion de contrôle : cycle de vie des *beans*, injection de dépendances et gestion de la configuration.

Les fichiers de configuration se trouvent dans `src/main/resources` :

- `spring` : fichiers de configuration *Spring* :
  - `addressrepository-core.xml` : inclue les fichiers nécessaires
  - `addressrepository-businessservice.xml` : paramètre le coeur pour être intégré à un autre Webservice
- `config` : fichiers de paramétrage (*properties*)

Gestion des contextes du RC

Porté *Spring* : `context`

Scope `context`

En plus des scopes classiques `singleton` et `prototype`, le scope `context` défini un singleton d'un contexte.

Gestion des contextes du RC

Porté *Spring* : `context`

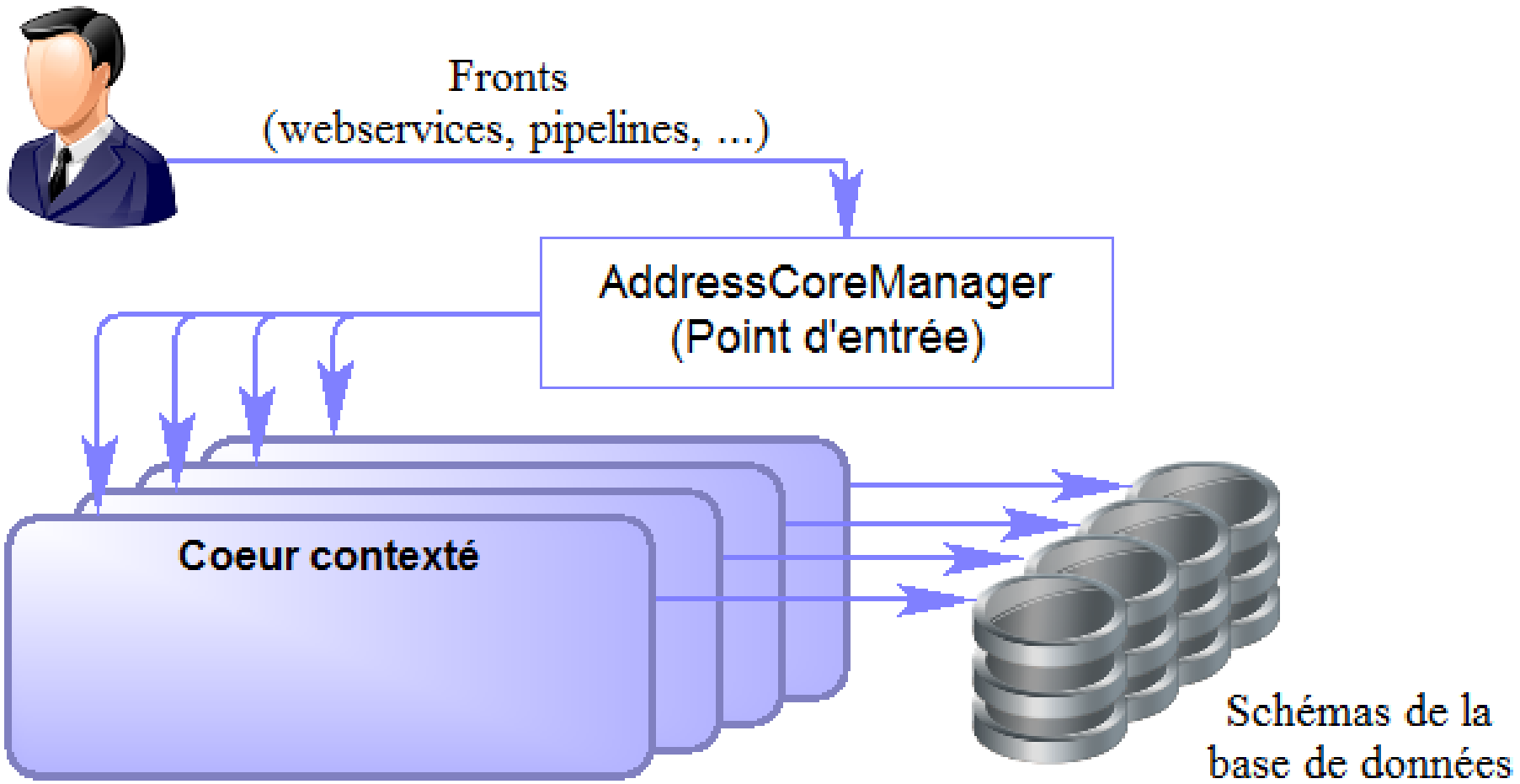
Scope `context`

En plus des scopes classiques `singleton` et `prototype`, le scope `context` défini un singleton d'un contexte.

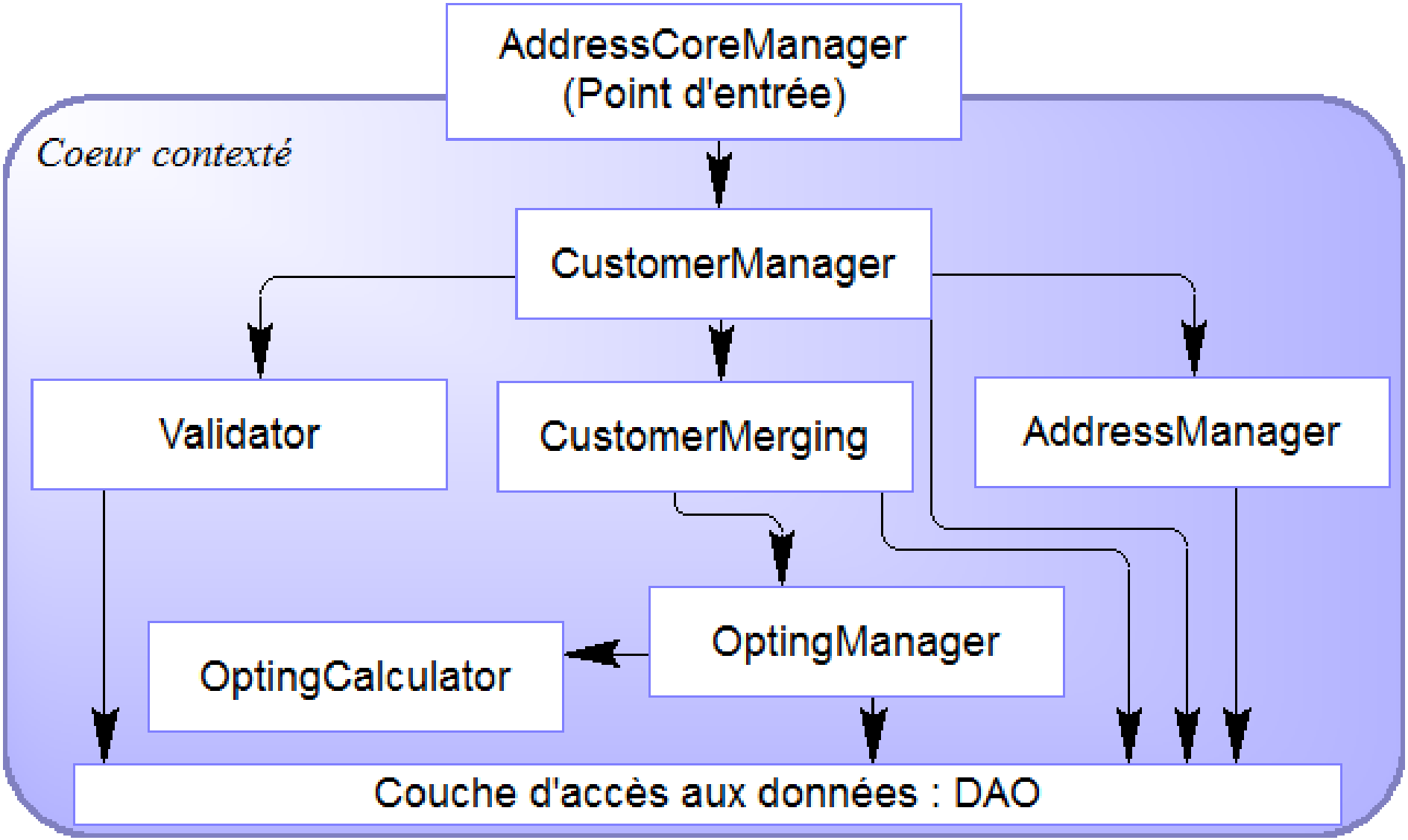
- Seule une instance de bean contexté est créée pour chaque contexte.

Distribution sur les contextes

Une instance du coeur est créée pour chaque contexte



Cartographie du coeur



Briques applicatives du RC

- **CustomerManager** Point d'entrée du coeur
  - exécute la validation
  - Recherche les doublons
  - Distribue : nouveau client, mise à jour, prospect
- **Validator** Valide les données d'un client, de ses comptes et coordonnées.
- **AddressManager** Traitement des coordonnées prospectes.
- **CustomerMerging** Confronte les données présentes en base à la mise à jour
- **OptingManager** Règle générale sur les consentements (propagations, flag)
- **OptingCalculator** Calcul sur les consentements

Couche d'accès aux données

Object Relational Mapping par *Hibernate*

Hibernate

Le framework *Hibernate* est utilisé pour gérer la relation entre le modèle objet et la base de données.

Couche d'accès aux données

Object Relational Mapping par *Hibernate*

Hibernate

Le framework *Hibernate* est utilisé pour gérer la relation entre le modèle objet et la base de données.

Session Factory

La `SessionFactory` d'Hibernate est configuré par Spring dans le fichier `context-persistence.xml`.
Elle a pour scope le **context** : une session factory par schéma.

Configuration des sources de données

Les sources de données sont fournie à *Spring* par le `DatasourceProvider`.
Il se repose sur 2 implémentations de *IDatasourceFactory* :

- `FileDatasourceFactoryImpl` : fichier properties présent sur UNIX (pipelines, batch)
- `JndiDatasourceFactoryImpl` : datasources présentes dans un dictionnaire JNDI

Pattern : *Open Session In View*

Open Session In View

Le pattern *Open Session In View* consiste à ouvrir la session le plus tôt possible, et la fermée le plus tard possible. Ainsi, les relations lazy ne sont chargées qu'au dernier moment et seulement si nécessaire.

Pattern : *Open Session In View*

Open Session In View

Le pattern *Open Session In View* consiste à ouvrir la session le plus tôt possible, et la fermée le plus tard possible. Ainsi, les relations lazy ne sont chargées qu'au dernier moment et seulement si nécessaire.

Ouverture des sessions

L'annotation `@OpenSession` permet d'ouvrir la session pour la méthode annotée, ou toutes les méthodes de la classes annotée.

Pattern : *Open Session In View*

Open Session In View

Le pattern *Open Session In View* consiste à ouvrir la session le plus tôt possible, et la fermée le plus tard possible. Ainsi, les relations lazy ne sont chargées qu'au dernier moment et seulement si nécessaire.

Ouverture des sessions

L'annotation `@OpenSession` permet d'ouvrir la session pour la méthode annotée, ou toutes les méthodes de la classes annotée.

Transactions

En revanche, les transactions sont positionnées pour garantir la cohérence de la base de données. Elle sont configurée par l'annotation `@Transactional`

Sommaire

Sommaire