Formation Groovy ET Grails

IHAB ABADI

Sommaire Séance 2 & 3

- Fondation Grails
- Architecture Grails
- Architecture Grails et Structure d'un projet
- Convention Grails
- Configuration Grails (BuildConfig, Config, DataSource, BootStrap)
- Externalisation de la configuration
- Configuration et multi-environnements
- Configuration grails Mapping
- Rappel Architecture MVC
- Les contrôleurs
- Les intercepteurs de contrôleurs
- La gestion d'exception dans un contrôleurs
- Grails et Spring
- Services
- Grails Cli et scripts

Fondation Grails

Les raisons de la mise en place de Grails sont :

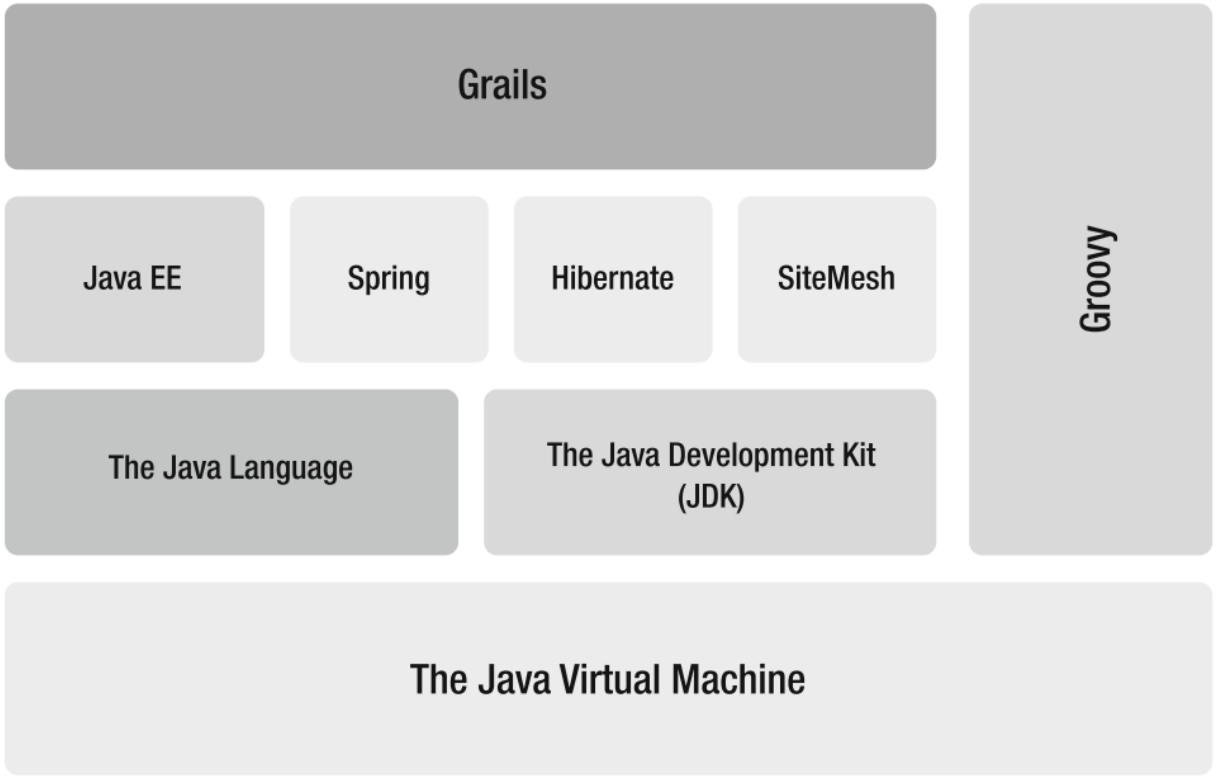
- Simplifier le développement web en JAVA
- Augmenter le niveau d'abstraction pour le développement web JAVA
- Rendre accessible au développeur web JAVA la flexibilité et simplicité existante dans d'autre plateforme depuis des années
- Conserver la flexibilité d'accès au technologie sous-jacentes et d'utiliser leur richesse et maturité

Fondation Grails

Grails utilise des concepts tels que :

- Convention over configuration CoC
- Don’t Repeat Yourself DRY
- Utilisation un langage dynamique tel que Groovy et son DSL

Architecture Grails



Architecture Grails

- Grails fournit à travers son architecture des technologies qui simplifient la vie sans sacrifier la robustesse
- Grails se base sur des technologies open source et populaires qui ont fait leur preuve dans l'écosystème JAVA
- Spring
- Hibernate
- SiteMesh
- HSQLDB

Architecture Grails structure d’un projet

- Assets : Ressources Web
- Config
- Controllers : Contrôleurs web
- Domain : Définition du modèle
- i18n : Internationalisation
- Init : contient le « Bootstrap »
- Services : Couche de service
- Taglib : Contient les « Taglibs » que vous pourriez créer
- Utils : Contient les « Classes codec » (encoders/decoders)
- Views : Groovy Server Pages⁄
- Src : Autre sources (Java / Groovy) ainsi que les tests unitaires / d’intégration

Les conventions Grails

Grails utilise des conventions pour implémenter des comportements par défauts tel que :

- Les répertoires des sources
- La nomenclature des classes (les controllers sont suffixés de controller…)
- Utilisation du domain driver design
- Utilisation du prototypage à l'aide des fonctionnalités du scaffolding
- Utilisation de plugin

La configuration en Grails 2.X

- Grails utilise CoC, mais nous laisse la possibilité de personnaliser de surcharger les conventions par de la configuration.
- Grails dans sa version 2.5 possède plusieurs fichiers de configuration.
- Les fichiers de configuration utilisent ConfigSlurper de Groovy

Configuration en Grails 2.X- Fichiers de configuration

- BuildConfig.groovy : Fichier utilisé pour la configuration des dépendances, propriétés des VMs, Plugins
- Config.groovy : Fichier utilisé pour la configuration générale du projet
- DataSource.groovy : Fichier utilisé pour la configuration des accès aux base de données
- UrlMappings.groovy : Fichier utilisé pour la configuration des schémas d'url
- BootStrap.groovy

Configuration Grails – BuildConfig.groovy

- BuildConfig.groovy est uniquement accessible uniquement à la compilation
- On peut accéder au donner de configuration du BuildConfig.groovy à l'aide la propriété grailsSettings.config dans les commandes
- Quelque propriété du BuildConfig.groovy

Configuration Grails – Config.groovy

- Config.groovy est accessible en runtime
- On peut accéder au donner de configuration du Config.groovy à l'aide l'objet grailsApplication
- Quelque propriété du Config.groovy

Configuration Grails - DataSource

- Configuration base de données (connecteur, serveur, utilisateur, mot de passe …)
- Définition du comportement au moment du lancement du projet
- Exemple de DataSource avec mysql

Configuration Grails – Bootstrap.groovy

- Bootstrap.groovy une classe qui permet de définir un comportement au lancement de notre application et la destruction de notre application

- Exemple d'utilisation de Bootstrap.groovy

Configuration grails – externalisation de la configuration

- Grails donne la possibilité d'avoir plusieurs source de configuration
- Les sources de configuration peuvent être des fichiers ou classes
- On indique le chemin des sources de configuration dans la propriété grails.config.locations
- On peut également spécifier le nom complet de classe de configuration

Configuration Grails, utilisation du multi-environements

- Grails supporte l'utilisation de plusieurs environnement de développement

- Les fichiers Config.groovy, DataSource.groovy, BootStrap.groovy peuvent définir plusieurs environnements

- On peut accéder au type d'environnement à l'aide de lasse Environment

- Exemple d'utilisation d'environnements dev et prod

Configuration Grails – UrlMappings.groovy

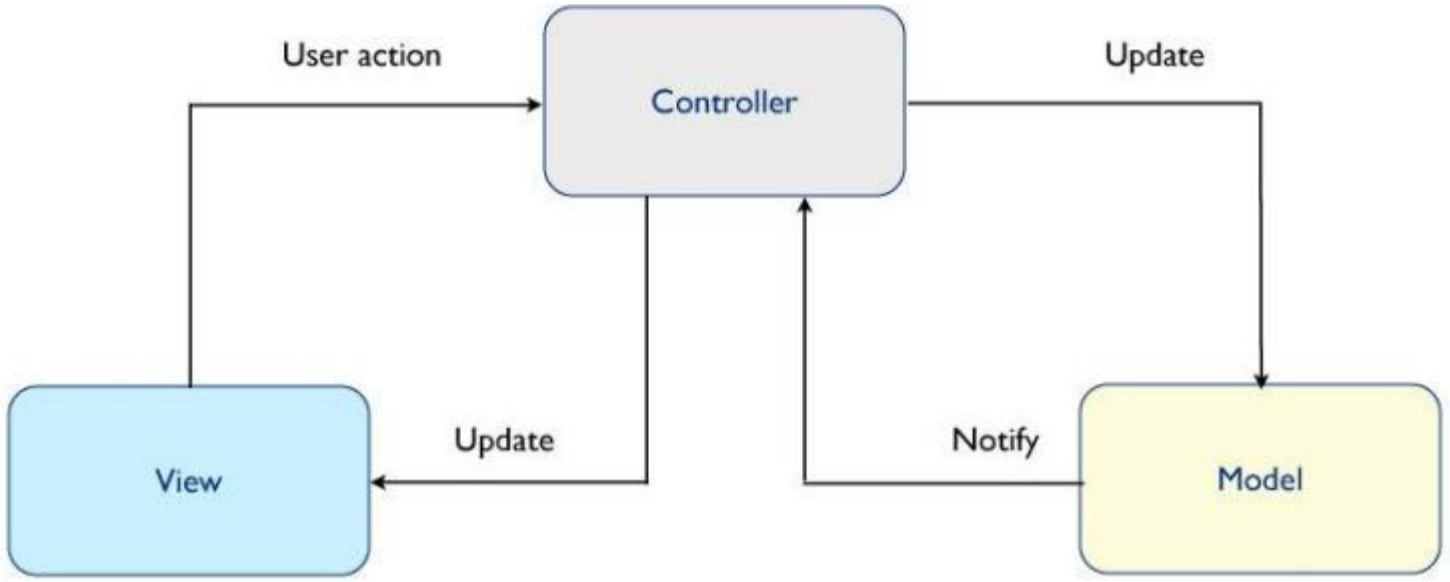
UrlMappings.groovy permet la configuration des liens vers :

- Des controllers / actions

- Des vues

- Des ressources (WebService / API REST)

Architecture Grails MVC



Controller Flow

- Un controller peut contenir plusieurs méthodes
- Une action est une méthode avec un scope bien définit
- Chaque URI Définit dans UrlMappings.groovy renvoie vers une action
- Un controller possède une action par défaut
- L’action par défaut est index
- L’action par défaut peut être surchargée

Controller - scopes

Grails supporte différentes façons de sauvegarder et faire transiter les informations entre action.

- Request :
  - Portée : uniquement la requête
  - Stockage des données uniquement pour la requête
  - Instance de HttpServletRequest

Controller - scopes

Flash Scope

- Temporaire
- Durée de vie de deux requêtes (la courante et la suivante)
- Utile pour passer des messages juste avant une redirection

Controller - scopes

Session Scope

- Porté : session
- Association d’état avec un utilisateur
- Via les cookies
- Instance de HttpSession

Controller - Scopes

ServletContext Scope

- Portée : Application
- Instance de ServletContext
- Scope très utile pour gérer des données qui ne dépendent d’aucun utilisateur (nombre de login par exemple)
- Scope utile pour charger des ressources à partir de notre application web

Controller - Scopes

Params Scope

Portée uniquement sur la requête

C’est une instance de Map contenant les paramètres de la requête

Controller - Scopes

Un contrôleur peut lui-même être définit avec plusieurs scopes possible

Prototype : un nouveau controller est créé pour chaque requête

Session : un controller est créé par session

Singleton : Une instance unique du controller pendant toute la durée de vie de notre application

Controller – renvoyer un objet

Un controller peut retourner un objet de plusieurs façons

En renvoyant un objet de type Map

Les instances définient dans Map seront envoyées à la vue

La vue porte le même nom que l'action

La vue se trouve dans un dossier qui porte le nom de notre controller

Controller – renvoyer un objet

Renvoyer un objet en utilisant respond

Essaie de retourner la réponse la plus approprée au Content type demandé dans le header Accept

Possibilité de limiter les formats

Controller – renvoyer un objet

Renvoyer un objet en utilisant la méthode render

Retourner une instance avec la possibilité de désignée une vue spécifique

Controller – renvoyer un objet

Renvoyer un objet en utilisant la méthode withFormat

withFormat permet de sépcifier le ou les formats à renvoyer en fonction du mimes types déclarés dans le Accept du header

Mimes types sont spécifiés dans config.groovy

Controller - Redirect

Une action d’un controller peut demander une redirection vers

Une autre action du même controller

Une autre action d’un autre controller

Une URI spécifique

Envoyer des paramètres

Controller – XML/JSON

Une action peut définir plusieurs renders avec des contentType différent

Exemple un contentType xml et un contentType json

Démo

Controller - Intercepeteurs

La possibilité de définir un comportement à exécuter avant l'appel des actions en utilisant beforeInterceptor

La possibilité de définit un comportement à exécuter après l'appel des actions en utilisant

afterInterceptor

Peut être utilisé pour des besoins de logging ou débug

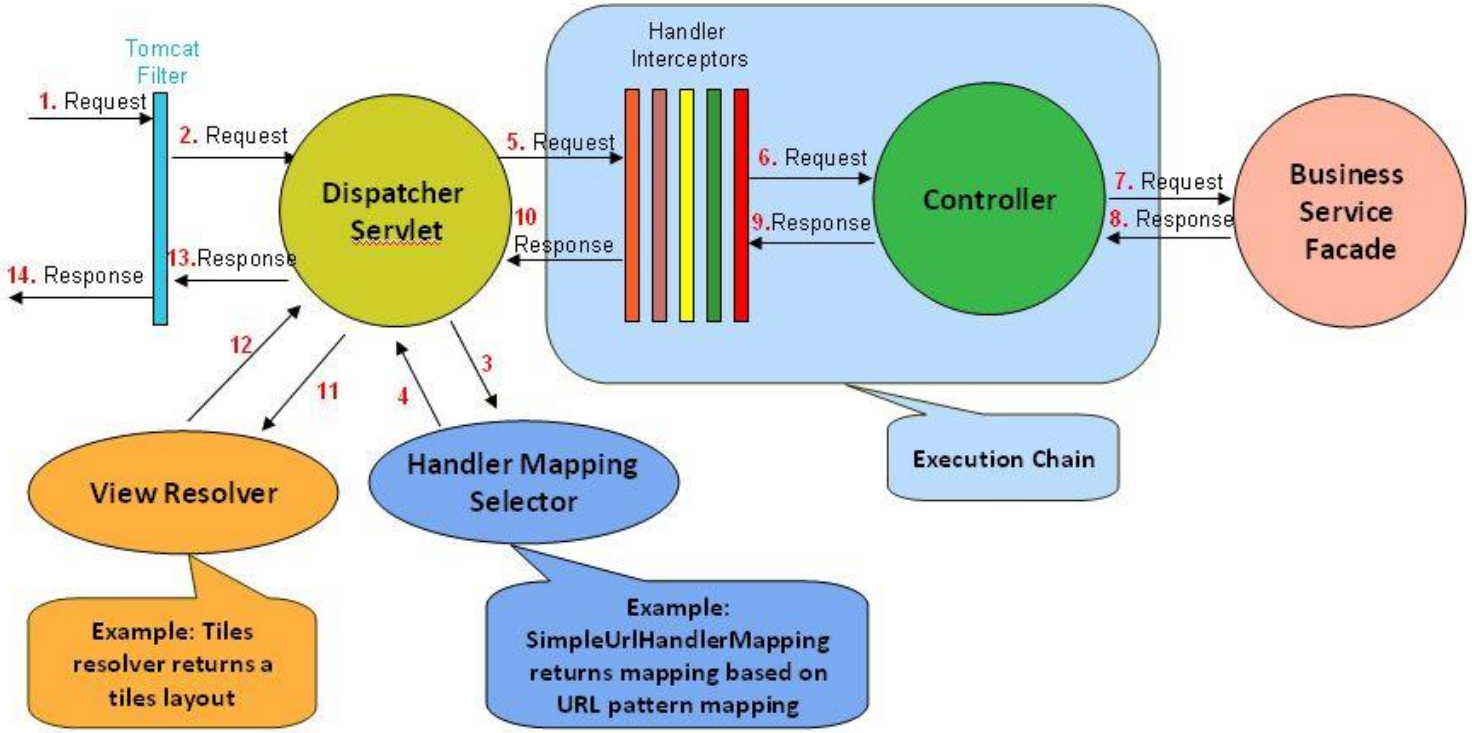
Grails et Spring

Grails est un top framework de spring

Grails utilise les mécanisme de Spring pour les services, l'injection de dépendance et inversion de contrôle

Pour des applications avec une logique métier non complexe le lien entre grails et spring peut rester invisible

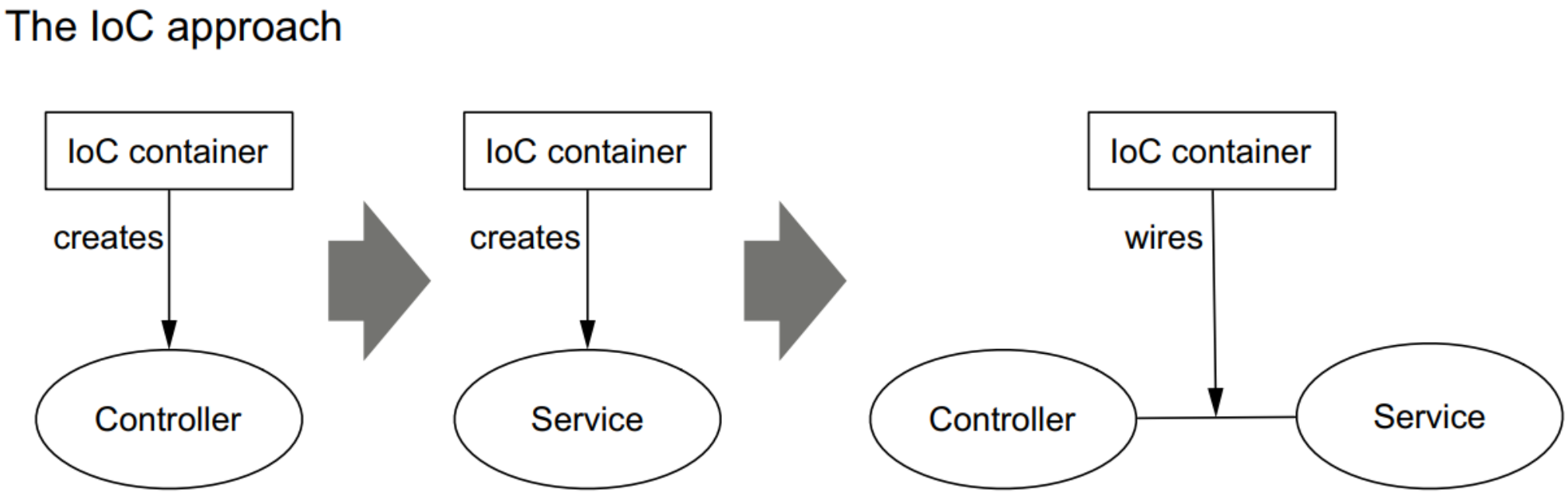
Grails et Spring



Grails et Spring

Spring est un DI Framework

Spring utilise IoC



Grails et Spring

L’utilisation de l’approche IoC facilite

- Les tests
- Rendre les implémentations de services interchangeable
- L’utilisation du paradigme AOP

Grails et Spring

Une application Grails crée un hôte de Spring Beans

La totalité des artefacts grails sont concidérés comme SpringBeans (controllers, tag librairies, services,….)

Les artefacts grails utilise le mécanisme du autowring de Spring pour injecter les dépendances

L’injection se fait en se basant sur la convention de nommage

Comme Spring, l’injection peut se faire en fonction du type

Grails et Spring

Certains artefacts de grails n’utilise pas la convention de nommage standard

Le but recherché est d’interdire l’injection de ces artefacts

Exemple Controller, Domaine classe

Grails et Spring

Quelque SpringBeans à utiliser dans Grails

- grailsApplication
- sessionFactory
- groovyPagesTemplateEngine
- messageSource

Chaque plugin peut ajouter ses propres SpringBeans

Grails et Spring

Le mécanisme d'injection de Beans offre la possibilité de changer implémentations des beans en mode runtime

On peut définir nos Beans directement dans le fichier de configuration resources.groovy

On peut implémenter une logique métier au moment de la définition des beans

Les beans définit dans resources.groovy sont chargé après le chargement des beans du core de grails et des plugins.

Démo

Services

Les services sont le cœur métier de notre application

Les services contiennent toute la logique

Les services sont en charge de la persistance

Les services sont souvent transactionnels

On peut utiliser les annotations pour déclarer un service ou une méthode transactionnel ou non

Les services transactionnels donnent la possibilité d'exécuter un rollback sur RuntimeException

Services - scope

Singleton par défaut

Prototype : une nouvelle instance à chaque injection

Request : une nouvelle instance par requête

Flash : une nouvelle instance pour la requête courant et la suivante

Flow : une nouvelle instance par web flow (concept Spring : séquence d’étapes)

Conversation : une nouvelle instance par conversation

Session : une nouvelle instance par session HTTP

Services - Injection

Aspect de grails hérite de Spring

Injection par convention

Utilisation d’une représentation de propriété basé sur le nom du service

Services - Injection

Injection de services est également possible dans

Les classes de domaine

Les Taglibs

Grails DataBinding

- Grails donne la possibilité de binder automatiquement les paramètres envoyés d'une requête

- Le binding permet de créer un objet et lier automatiquement les clés de la map params avec les propriétés de l'objet.

- Le binding fonctionne avec tout type d'objet.

- Le binding peut être réaliser à l'aide d'objet de type commande

Grails application de comportement commun entre les contrôleurs

- Factoriser des comportements entre plusieurs contrôleurs à l'aide des Traits
- Factoriser des comportements entre plusieurs contrôleurs à l'aide de classe mère
- Factoriser des comportements entre plusieurs contrôleurs à l'aide des filters

Grails Command line Interface

Grails possède une fonctionnalité qui permet d'exécuter des scripts à l'aide de commande

Grails utilise des conventions pour lancer des scripts

On peut créer nos propres scripts à l'aide de Gant et de groovy

Liste des cli fournit par Grails

Exemple de script personnalisé

Grails Command line Interface

Grails offre la possibilités de déclencher un script à un évènement spécifique

Grails offre la possibilité de déclencher des évènements à l'aide d'un script

Liste des évènements disponibles

Etude de cas(Application e-commerce)

Le but est de réaliser une application qui propose à un utilisateur invité une section de produits en fonction d'une catégorie.

Chaque invité peut construire son panier.

La validation d'un panier en commande se fait après authentification ou enregistrement d'un utilisateur.

Chaque utilisateur peut commenter un produit qu'il a déjà acheté.

Un utilisateur admin peut ajouter, modifier, supprimer les catégories de produits, les produits, consulter les commandes

Etude de cas(Application e-commerce)/Sprint 1

-Réalisation de l’architecture contrôleurs, action et services nécessaires.