



Spring  
Spring Boot

# Sommaire

1. Présentation de la technologie
  - Spring
  - Spring Boot
2. Objectifs du tutoriel
3. Tutoriel
  - Mise en place
  - Implémentation
  - Exécution

# Présentation de la technologie : Spring

- créée en 2003, Rod Johnson
- framework open source -> plateforme de développement
- concepts clé :
  - Injection de dépendance
  - Programmation par aspect
  - Couche abstraction pour utilisation autres frameworks Java
- Modulaire :
  - Spring Data
  - Spring Sécurité
  - Spring MVC
  - ...

# Présentation de la technologie :

## Inversion contrôle

- Patron d'architecture commun à de nombreux framework (Spring/Spring Boot)
- Inverse le flow du déroulement de l'application

# Présentation de la technologie :

## Inversion de dépendance

- Permet de faire de l'inversion de contrôle
- Le framework crée et fournit les instances de classe importante

### Sans inversion de dépendance

```
public class Store {  
    private Item item;  
  
    public Store() {  
        item = new ItemImpl1();  
    }  
}
```

### Avec inversion de dépendance

```
public class Store {  
    private Item item;  
    public Store(Item item) {  
        this.item = item;  
    }  
}
```

# Présentation de la technologie :

## Inversion de contrôle

### Bénéfices :

- Découplé l'exécution et l'implémentation des tâches.
- Plus simple de changer d'implémentation.
- Plus grande modularité du programme.
- Simplicité des tests par isolation des composants.

# Module Spring Initializr

Site permettant de faire les configuration de base du projet :

- Arborescence
- Dépendance dans fichier pom.xml

Démo : <https://start.spring.io/>

# Présentation de la technologie : Module Spring Boot

Utilité de Spring Boot :

- Création d'application autonome basé sur Spring
- Intégration de serveur Tomcat, jetty ou Undertow de façon native
- Intégration de dépendance dite "starter" pour simplifier les configurations
- Automatisation des configurations pour Spring et certaines librairies
- Management des dépendances pour éviter les problèmes de versionnages



# Objectif du tutoriel

- Création d'une application utilisant une architecture microservice
- Apprendre à utiliser Spring et Spring Boot

# Tutoriel : Outils utilisés

- Eclipse : IDE
- Maven 3.6.1
- PostgreSQL pour window
- Postman : outil pour utilisation des APIs REST



# Tutoriel : Description de l'application

- Gestion d'une collection de livre très simple
- Action de l'utilisateur
  - emprunter un livre
  - rendre un livre
  - consulter sa collection de livre

# Tutoriel : Description de l'application

## Le client :

possibilité : navigateur ou une API externe

choix : simulation navigateur via postman

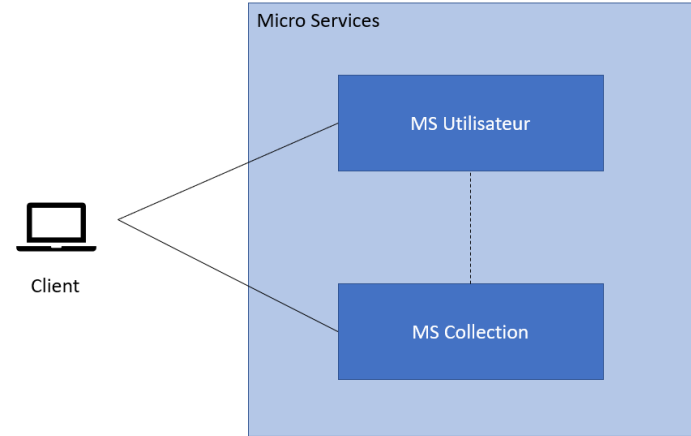
## MS Utilisateur :

Gestion des inscriptions, connexions des utilisateurs et demande comparaison entre identifiant BDD et identifiant utilisateur

## MS Collection :

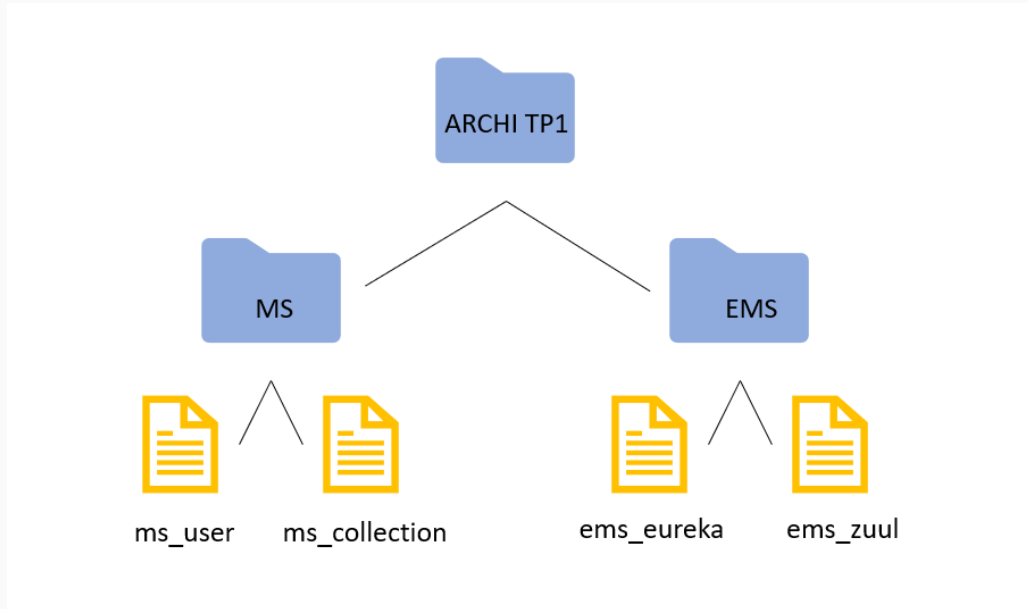
Gestion collections : l'ajout, retrait d'un de livre de la collection d'un utilisateur et consultation de sa collection par l'utilisateur

## Architecture



# Tutoriel : Création projet sous Eclipse

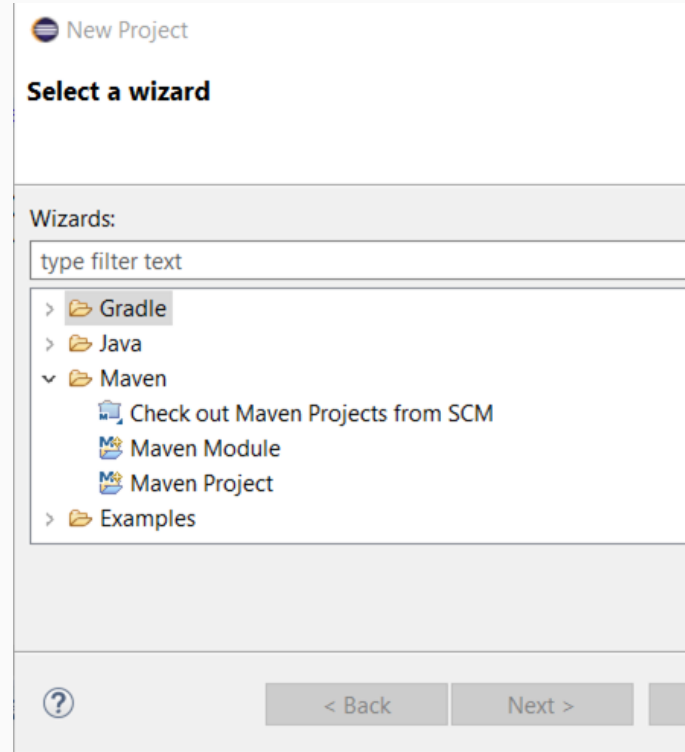
## Arborescence de l'application



# Tutoriel : Création projet sous Eclipse

Création répertoire racine :

Sélectionner Maven project  
Cocher "Create a simple project"



# Tutoriel : Création projet sous Eclipse

Artifact

Group Id:

ARCHITP1

▼

Artifact Id:

ARCHITP1

▼

Version:

0.0.1-SNAPSHOT

▼

Packaging:

pom

▼

Name:

ARCHITP1

▼

Description:

⬆ ⬇ ⬆

Parent Project

Group Id:

▼

Artifact Id:

▼

Version:

▼

Browse...

Clear

▶ Advanced

?

< Back

Next >

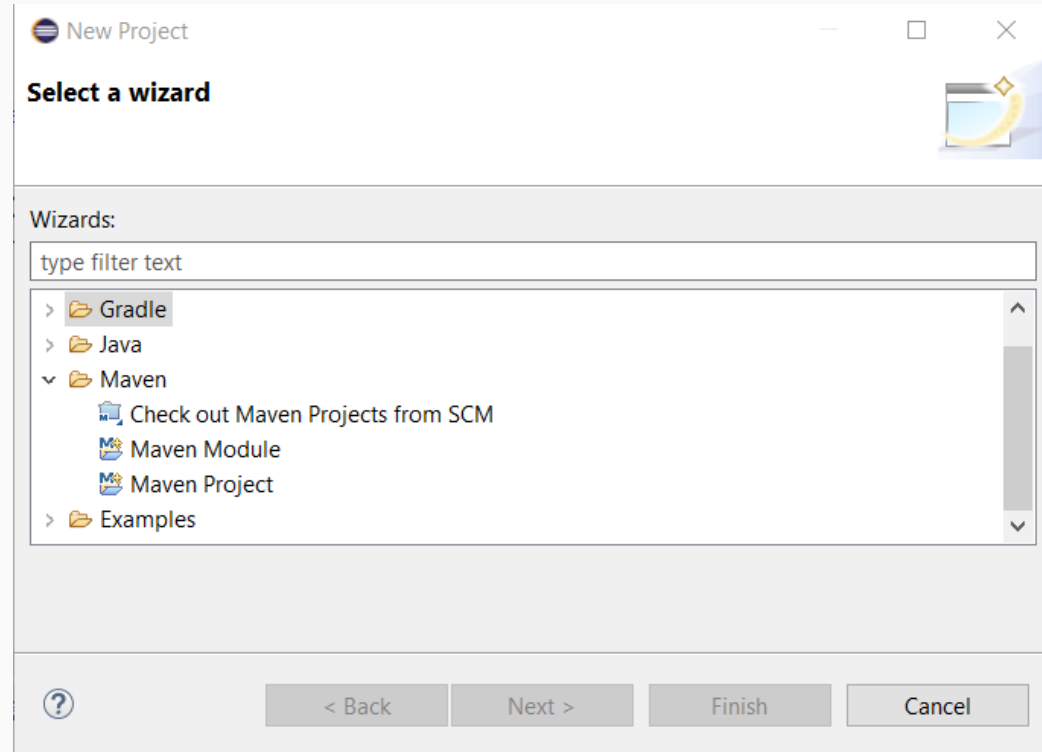
Finish

Cancel

# Tutoriel : Création projet sous Eclipse

Création des sous répertoire MS et EMS :

Sélectionner Maven module  
Cocher "Create a simple project"





# Tutoriel : Création projet sous Eclipse

## Création des microservices :

- Cliquez droit sur MS
- New > Projet...
- Sélectionnez Maven Module et cocher "Create a simple project"
- Entrez le nom du microservice (ms\_user ou ms\_collection)








## Création des edges microservices :

- Cliquez droit sur MS
- New > Projet...
- Sélectionnez Maven Module et cocher "Create a simple project"
- Entrez le nom du edge microservice (ems\_eureka ou ems\_zuul)

NB : Ribbon est directement intégré dans les microservices donc pas besoin de le créer

# Tutoriel : Création projet sous Eclipse

## Résultat

- >  ARCHITP1 [ARCHITP1 master]
- >  EMS [ARCHITP1 master]
- >  ems\_eureka [ARCHITP1 master]
- >  ems\_zuul [ARCHITP1 master]
- >  MS [ARCHITP1 master]
- >  ms\_collection [ARCHITP1 master]
- >  ms\_user [ARCHITP1 master]

# Tutoriel : Implémentation

## Création des packages des microservices :

- `fr.tse.myapp` : package racine, contient `App.java` le point d'entrée
- `fr.tse.myapp.api` : contient les classe de routing
- `fr.tse.myapp.domain` : contient classes objets et classes dont instances seront transmise sur le réseau
- `fr.tse.myapp.repository` : contient classes pour la communication avec la BDD
- `fr.tse.myapp.service` : contient les classes contenant le code métier

# Tutoriel : Implémentation

## Création classe principale :

- Cliquez droit sur package fr.tse.myapp
- New > Class
- Puis configuration
- Mettre le code
- Ajouter dépendances associé dans pom.xml de la racine du projet ARCHITP1

The screenshot shows the 'New Class' dialog box in an IDE. The configuration is as follows:

- Source folder:** ms\_collection/src/main/java (with a 'Browse...' button)
- Package:** fr.tse.myapp (with a 'Browse...' button)
- Enclosing type:** (empty, with a 'Browse...' button)
- Name:** App
- Modifiers:** ☒ public, ☐ package, ☐ private, ☐ protected, ☐ abstract, ☐ final, ☐ static
- Superclass:** java.lang.Object (with a 'Browse...' button)
- Interfaces:** (empty list, with 'Add...' and 'Remove' buttons)
- Which method stubs would you like to create?**
  - ☒ public static void main(String[] args)
  - ☐ Constructors from superclass
  - ☒ Inherited abstract methods
- Do you want to add comments?** (Configure templates and default value [here](#))
  - ☐ Generate comments

# Tutoriel : Implémentation

## Création fichier Application.properties

- Sur src/main/resources Cliquez droit
- New > File
- Entrez application.properties comme nom
- Dans le fichier ajouter le nom du microservice et le port du server

```
spring.application.name=ms_collection  
server.port = 8083
```

# Tutoriel : Implémentation

## Création classe Collection :

- Créer une nouvelle classe java dans le package domain
- Au-dessus de la déclaration de classe ajouter les annotations `@Entity` et `@Table` pour que Spring reconnaisse que cette classe sera utilisé pour la base de donnée
- Pour chaque attributs qui représente un colonne de la table ajouter l'annotation `@Column` au-dessus.
- Si en plus l'attribut est la clé de la table il faut ajouter `@Id` et nous avons décidé d'ajouter en plus `@GeneratedValue` pour incrémenté automatiquement la clé.
- Ajouter la méthode `toString` en la surchargeant pour permettre les échanges.
- Ajouter les dépendances nécessaire dans le `pom.xml` du microservice

# Tutoriel : Implémentation

Création classe CollectionRepository :

- Créer une nouvelle classe java dans le package repository
- Ajouter le code
- Ajouter les dépendances nécessaire dans le pom.xml du micro service si elle ne sont utile que pour un microservice sinon dans le pom.xml global

# CollectionRepository

Gère les interactions vers la base de données

- @Repository défini la classe, elle sera donc trouvée par @ComponentScan
- Extend JpaRepository ce qui donne accès à des méthodes tel que save() deleteBy..()
- @Query permet d'écrire des queries personnalisées



# Base de données

## PostgreSQL

Configuration dans le fichier  
application.properties

```
## PostgreSQL
spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql: l'adresse de votre base
spring.datasource.username= votre nom d'utilisateur
spring.datasource.password= votre mot de passe

#drop n create table again, good for testing, comment this in production
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL9Dialect
spring.jpa.show-sql=false
spring.jpa.hibernate.ddl-auto=create
```

# Tutoriel : Implémentation

Création classe CollectionService :

- Créer une nouvelle classe java dans le package service
- Ajouter le code

# CollectionService

Gère le code métier

Fait le lien entre le routing et le repository

S'occupe de la transformation et le tri des données

- @Service
- @Autowired pour l'inversion de dépendance: avoir une instance de la classe CollectionRepository
- RestTemplate pour communiquer avec les autres microservices
- DTO pour l'échange d'information entre service

# Tutoriel : Implémentation

Création classe CollectionApi :

- Créer une nouvelle classe java dans le package service
- Ajouter le code

# CollectionApi

Gère le code métier

Fait le lien entre le routing et le repository

S'occupe de la transformation et le tri des données

- `@RestController` qui permet d'enregistrer cette classe en tant que Rest Controller
- `@Autowired` pour l'inversion de dépendance
- `@RequestMapping` qui permet de définir la route vers notre application

# Edge microservice

## Eureka !

Registre pour les  
microservices et leurs  
instances

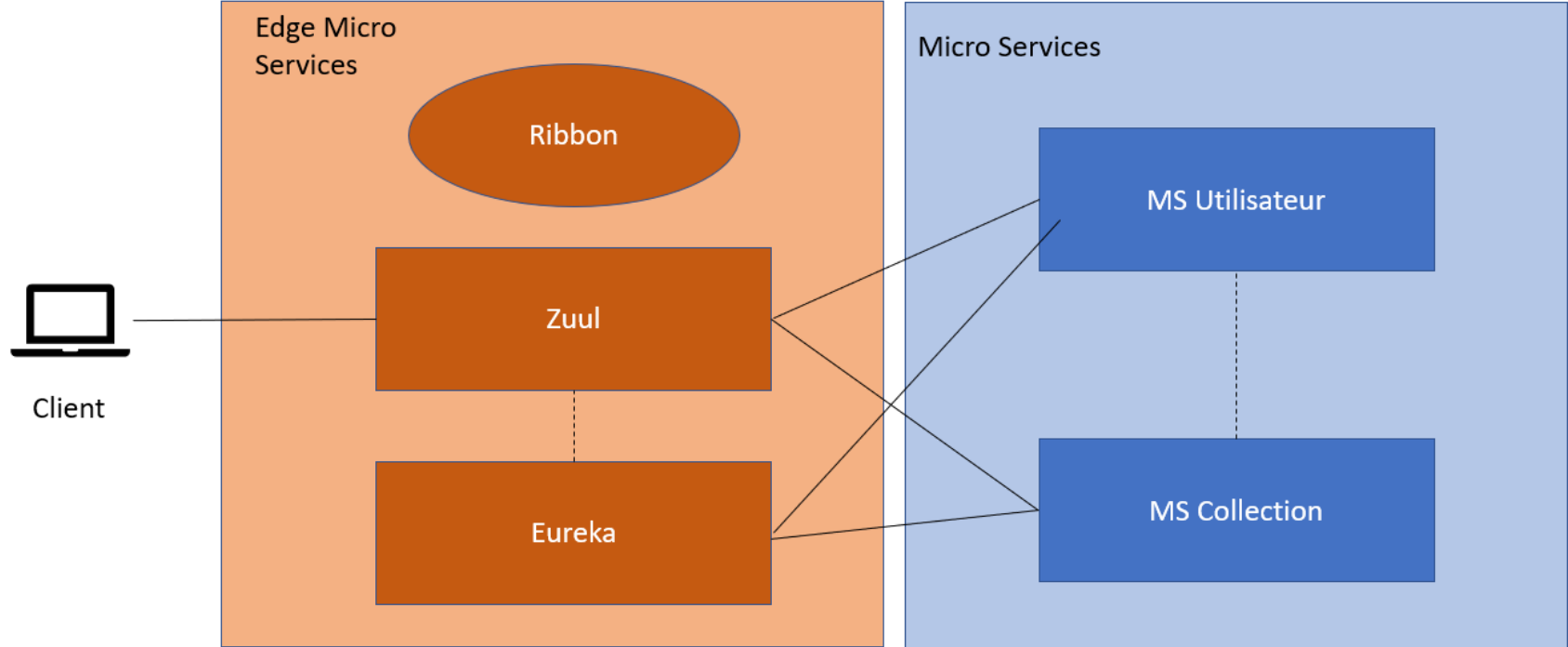
## Ribbon

Load Balancer côté client

## Zuul

Point d'entrée unique de  
l'application  
Peut filtrer les requêtes  
arrivantes

# Architecture améliorée



# Conclusion

- Spring / Spring Boot sont d'excellent outils pour construire rapidement des applications modulable et prête à la production
- Parfait pour les micro services de par la création d'application autonome et modulable
- Un grand choix d'outil et de module tel que Spring Cloud, Spring Data etc ...