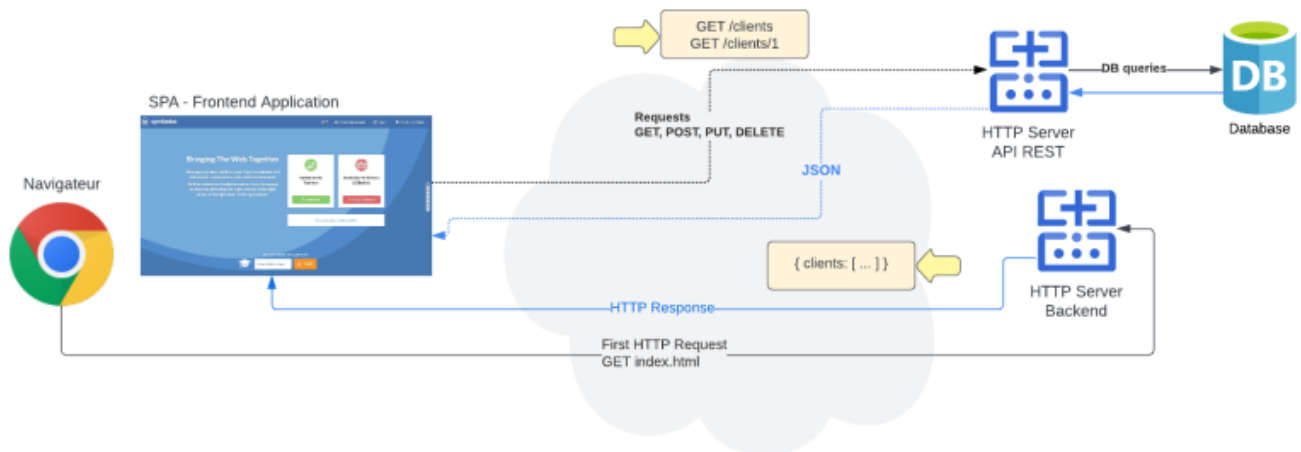


# TD n°4

## SpringBoot - VueJS

Créer la branche **td4** à partir de la branche **td3** de votre projet (reprenez éventuellement la correction).

### Client/serveur REST



### Rest controller

Lire [APIs web](#)

### Configuration

Ajouter la dépendance Rest Data à **pom.xml** :

```

<!--
https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-data-rest -->
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-rest</artifactId>
  <version>3.0.4</version>
</dependency>

```

Configurer l'api REST pour qu'elle expose les identifiants de certaines entités en créant une nouvelle classe de configuration :

```

@Configuration
class RestConfig:RepositoryRestConfigurer {
  override fun configureRepositoryRestConfiguration(config:

```

```
RepositoryRestConfiguration?, cors: CorsRegistry?) {
    config?.exposeIdsFor(Master::class.java, Dog::class.java)
}
}
```

## Création des ressources

Créer une nouvelle interface **RestMasterResource** héritant de `JpaRepository`, annotée avec `@RestController` :

```
@RepositoryRestResource(collectionResourceRel = "masters", path = "masters")
interface RestMasterResource: JpaRepository<Master, Int> {
}
```

Même chose pour les Dogs :

```
@RepositoryRestResource(collectionResourceRel = "dogs", path = "dogs")
interface RestDogResource: JpaRepository<Dog, Int> {
}
```

## Mise à jour des entités

- L'annotation `@JsonBackReference` sur le champ **master** de la classe **Dog** évite la sérialisation JSON infinie des objets liés entre eux.
- L'annotation `@RestResource` relie le champ **master** à la ressource correspondante, non exportée (non accessible directement par l'URL).

```
@Entity
open class Dog() {
    constructor(name:String):this(){
        this.name=name
    }
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @JsonSerialize
    open var id = 0

    @Column(length = 30, nullable = false)
    open var name: String=""

    @ManyToOne(optional = true)
    @RestResource(exported = false,rel = "master", path = "master")
    @JsonBackReference
    open var master: Master?=null

    @ManyToMany
    open val toys= mutableSetOf<Toy>()
}
```

## Tests de l'api

Tester les requêtes suivantes avec [Reqbin](#)

| Méthode | URL     | Data                                 | Résultat                          |
|---------|---------|--------------------------------------|-----------------------------------|
| Get     | /dogs   |                                      | Liste des chiens                  |
| POST    | /dogs   | {"name": "Rex"}                      | Ajout d'un chien                  |
| PUT     | /dogs/1 | {"name": "Rex", "master": {"id": 1}} | Modification complète d'un chien  |
| PATCH   | /dogs/1 | {"name": "Rexa"}                     | Modification partielle d'un chien |
| DELETE  | /dogs/1 |                                      | Suppression d'un chien            |

| Méthode | URL        | Data                                     | Résultat                           |
|---------|------------|--|------------------------------------|
| Get     | /masters   |  | Liste des maîtres                  |
| POST    | /masters   | {"firstname": "John", "lastname": "DOE"} | Ajout d'un maître                  |
| PUT     | /masters/1 | {"firstname": "John", "lastname": "DOE"} | Modification complète d'un maître  |
| PATCH   | /masters/1 | {"firstname": "Johnny"}                  | Modification partielle d'un maître |
| DELETE  | /masters/1 |  | Suppression d'un maître            |

## SPA VueJS

### Intégration

Ajouter la dépendance Springboot-vuejs dans **pom.xml** :

```
<dependency>
  <groupId>io.github.jeemv.springboot.vuejs</groupId>
  <artifactId>springboot-vuejs</artifactId>
  <version>1.1.7</version>
</dependency>
```

Ajouter une classe de configuration pour permettre l'injection des instances de **VueJS** :

```
@Configuration
@ComponentScan("io.github.jeemv.springboot.vuejs")
class AppConfig {
}
```

Ajouter la configuration suivante dans **application.properties** :

```
##vuejs
springboot.vuejs.delimiters={!,!}
springboot.vuejs.axios=true
springboot.vuejs.el=#app
springboot.vuejs.vuetify=false
springboot.vuejs.vue-version=3.*
springboot.vuejs.http-data=response.data._embedded
```

## Injection dans un contrôleur

Créer un contrôleur **SPAController** :

```
@Controller
class SPAController {

    @Autowired
    lateinit var vue: VueJS

    @ModelAttribute("vue")
    fun vue(): VueJS = vue
}
```

## Chargement et affichage de données

### Chargement des masters côté client

Les données seront chargées lorsque l'objet **vuejs** est mounted, par une requête Ajax de type GET, alimentant le tableau **this.masters** :

```
@Controller
@RequestMapping("/spa")
class SPAController {

    @Autowired
    lateinit var vue: VueJS

    @ModelAttribute("vue")
    fun vue(): VueJS = vue

    @GetMapping(path = ["/", "", "/index"])
    fun index(): String {
        vue.addDataRaw("masters", "[]")
        vue.onMounted(
            Http.get("/masters",
                Http.responseArrayToArray("this.masters", "masters"),
                "console.log('Erreur sur chargement des données!');"
            )
        )
        return "/spa/index"
    }
}
```

Template /spa/index.html :

```
<h1>Maîtres <span class="ui label">{!masters.length!}</span></h1>
<ul>
```

```
<li v-for="master in masters">
  {!master.firstname!} {!master.lastname!}
</li>
</ul>

{{> /main/footer }}
{{{vue}}}
```

L'appel de **{{{vue}}}** permet d'insérer le code vueJS généré dans la page.

### Chargement des masters côté serveur

#### Alternative

1. Injecter le repository des masters
2. Passer à l'instance de **vue** la liste des masters

```
@Controller
@RequestMapping("/spa")
class SPAController {

    @Autowired
    lateinit var masterRepository: MasterRepository

    @Autowired
    lateinit var vue: VueJS

    @ModelAttribute("vue")
    fun vue(): VueJS = vue

    @GetMapping(path = ["/", "", "/index"])
    fun index(): String {
        vue.addData("masters", masterRepository.findAll())
        return "/spa/index"
    }
}
```

### Ajout d'une méthode

Ajout d'une méthode de suppression d'un master, sur l'instance de **Vue**, dans l'action **index** de **SPAController** :

```
vue.addMethod("remove",
    Http.delete("/masters/'+master.id",
        JsArray.remove("this.masters", "master")+
        JsArray.addAll("this.dogs", "master.dogs")+
        "console.log(`Maître \${master.firstname} supprimé!`);",
        "console.log('Erreur sur suppression de master!');"
    ),
```

```
"master")
```

Template /spa/index.html avec appel de la méthode **remove** :

```
<h1>Maîtres <span class="ui label">{!masters.length!}</span></h1>
<ul>
  <li v-for="master in masters">
    {!master.firstname!} {!master.lastname!} <i class="remove icon"
@click="remove(master)"></i>
  </li>
</ul>

{{> /main/footer }}
{{{vue}}}
```

### Création de composant

Composant pour gérer l'affichage d'un message, ayant les propriétés suivantes :

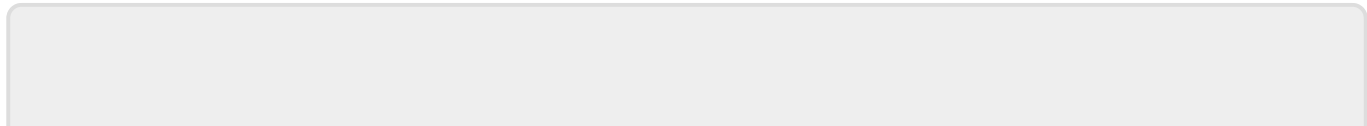
| Propriété | Type    | Rôle                                     |
|-----------|---------|--|
| visible   | boolean | Détermine si l'élément est visible       |
| error     | boolean | Détermine le type de message, et d'icône |
| title     | string  | Titre du message                         |
| content   | string  | Texte du message                         |

### Template HTML (basé sur le ui message de semantic) :

```
<div v-show="visible" :class="[error ? 'ui icon error message' : 'ui icon success
message']">
  <i :class="[error ? 'exclamation circle icon' : 'check circle icon']"></i>
  <div class="content">
    <div class="header">
      {{title}}
    </div>
    <p>{{content}}</p>
  </div>
</div>
```

### Création du composant global sur l'instance de Vue :

```
vue.addGlobalComponent("AppMessage").setProps("title","content","error","visible").
setDefaultTemplateFile()
```



From:

<http://slamwiki2.kobject.net/> - **Broken SlamWiki 2.0**

Permanent link:

<http://slamwiki2.kobject.net/web/framework/spring/td4?rev=1678150888>

Last update: **2023/03/07 02:01**

