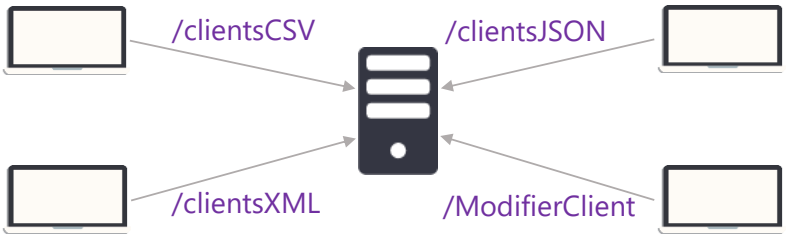
Le framework Spring

Module 6 – Spring REST

Objectifs

- Introduction l'architecture REST
- Mise en place de @RestController
- Utilisation de Spring Data JPA et @RestController
- Création d'une API avec Spring Data REST



La problématique dans un grand nombre d'applications web est la multiplication des URLs pour manipuler une ressource

Par exemple, pour interroger une base de clients, il est possible d'avoir les URL suivantes :

/clientsCSV pour obtenir la liste des clients au format CSV

/clientsXML pour obtenir la liste des clients au format XML

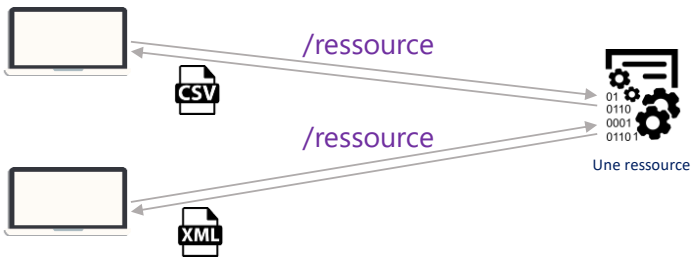
/clientsJSON pour obtenir la liste des clients au format JSON

/ModifierClient pour modifier un client

Il est possible d’en ajouter encore beaucoup en lien avec les fonctionnalités attendues.

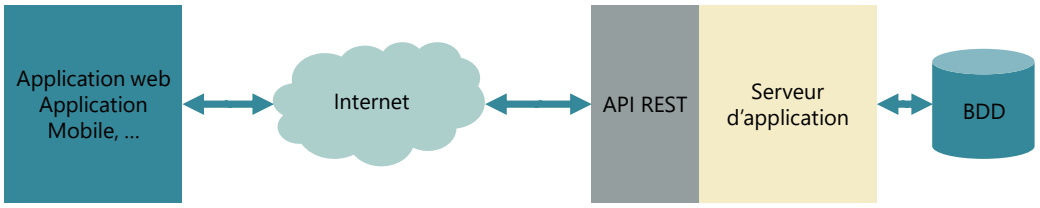
Il devient alors difficile de s’y retrouver, difficile de retrouver l’URL à utiliser pour une certaine fonctionnalité…

Il est nécessaire de simplifier cette représentation.
L’idée est d’avoir une URL unique pour manipuler une ressource que ce soit en lecture ou en écriture,



REpresentational State Transfer

Une ressource doit être vue comme des données brutes et non par sa représentation dans un format spécifique.
Ainsi, pour travailler avec une ressource (en lecture, en écriture) une seule URL suffit.
Chaque client reçoit la donnée dans le format désiré.
Une URL devient un point d'accès unique à une ressource indépendamment de sa représentation.
C'est le principe du style d'architecture REST.
L'en-tête de requête HTTP Accept a toute son importance pour déterminer la représentation (l'état) attendue par l'émetteur de la requête.
REST veut dire Representational State Transfer.



Une API est un logiciel qui offre des services à d'autres logiciels.

Plusieurs applications de type et langages différentes veulent interroger un même serveur.

Exemple : une application mobile, une application Web en Angular, ...

Pour permettre de manipuler le même serveur, il faut donner l'accès à la ressource.

La tendance actuelle consiste à exposer une API REST qui permet de lire, créer, modifier et supprimer des

ressources, en s'appuyant sur le protocole HTTP et le format JSON.

API REST est un service
Application Web va interagir par le protocole HTTP
À un serveur d'application associé à une base de données
Le serveur d'application va exposer une API REST

Une API REST permet donc de disposer d'un ensemble de méthodes pour manipuler des ressources, quel que soit la technologie utilisée côté client et côté serveur

Cette approche se standardise et se répand : la majorité des services sont aujourd'hui exposés de cette manière,

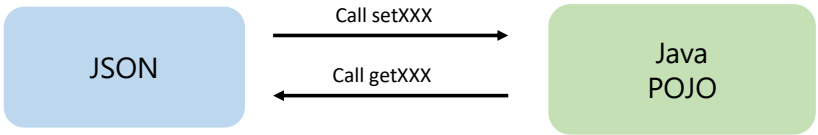
même les administrations commencent à publier leurs données à travers des API REST

Une API REST est stateless.
→ L'appel (l'exécution d'une méthode) ne doit pas dépendre de l'appel d'une méthode précédente

Exemple :
Si vous devez accéder à votre panier.
Vous devez être connecté
Le jeton d'authentification (token) sera transmis entre chaque demande avec l'API REST
→Une requête doit donc contenir toutes les informations nécessaires à son traitement
→ Pas de notion de session

Spring REST
Mapping JSON - Java

- Spring utilise la librairie Jackson pour le mapping



- Starters :

```
dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-web'
  developmentOnly 'org.springframework.boot:spring-boot-devtools'
}
```

build.gradle

Au travers des API REST, il faut transmettre les informations des objets Java.
Et récupérer des données venant du réseau et les utiliser en Java
Il faut donc une solution de mapping entre les objets JavaScript et ceux de Java et inversement.
Spring utilise la librairie Jackson pour gérer le mapping entre les objets JSON et Java.

Spring boot, avec le starter web intègre par défaut Spring Rest

Pour convertir, utilisation de la classe
com.fasterxml.jackson.databind.ObjectMapper

- Convertion JSON vers Java ➔ appel des setter de la classe POJO + la méthode readValue(JSON, …class)
- Convertion du Java vers JSON ➔ appel des getter de la classe POJO + la méthode writeValue

- C'est un Bean Spring
  - Défini avec l'annotation @RestController
  - Elle hérite de @Controller
- Il gère les requêtes et les réponses de l'architecture REST
- Spring gère automatiquement le mapping JSON – Java POJO
- L'application crée avec @RestController est une application Web Service REST

C'est un Bean Spring
Défini avec l'annotation @RestController
Elle hérite de @Controller

Il gère les requêtes et les réponses de l'architecture REST

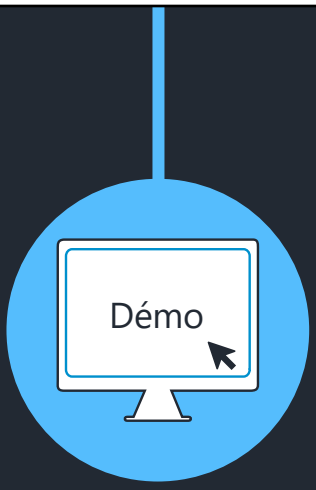
Spring Rest gère automatiquement le mapping JSON – Java POJO

Il est aussi possible d'ajouter un contrôleur pour gérer les exceptions Java et les rediriger vers un status 404 par exemple
Ce contrôleur est injecté par l'AOP, il réagit au type d'exception précisé :

```
@ControllerAdvice
class NotFoundAdvice {

@ResponseBody
@ExceptionHandler(RuntimeException.class)
@ResponseStatus(HttpStatus.NOT_FOUND)
String notFoundHandler(RuntimeException
ex) {
return "notFoundHandler - " +
ex.getMessage();
}
}
```

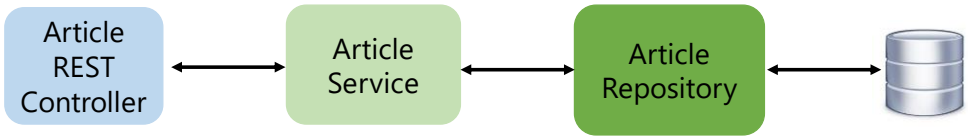
L’application crée avec @RestController est une application Web Service REST.
Mais ce n’est pas encore une API car nous gérons le choix des méthodes du protocole http exposées.
Et nous devons conserver des informations en session pour gérer la sécurité par exemple.



RestController

- Voyons un premier exemple d'utilisation de Spring Rest.

- Association Spring Rest et Spring Data JPA



- Starters :

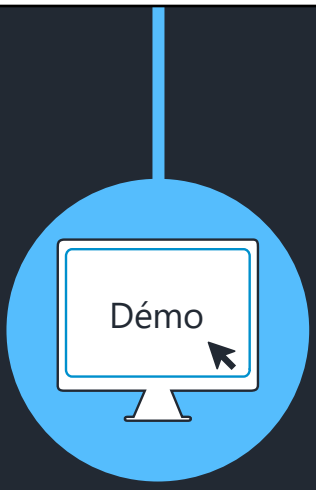
```
dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-web'
  developmentOnly 'org.springframework.boot:spring-boot-devtools'
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  runtimeOnly 'mysql:mysql-connector-java'
}
```

build.gradle

Il est évidement possible d'associer la couche DAL de Spring Data JPA avec Spring Rest.
Cela permettra de transmettre les données provenant d'une base.
Il faut ajouter :

- le starter de Spring Data JPA
- Le driver de la base de données
- Configurer l'accès à la ressource dans application.properties
- Déclarer les entités
- Et les classes de Repository

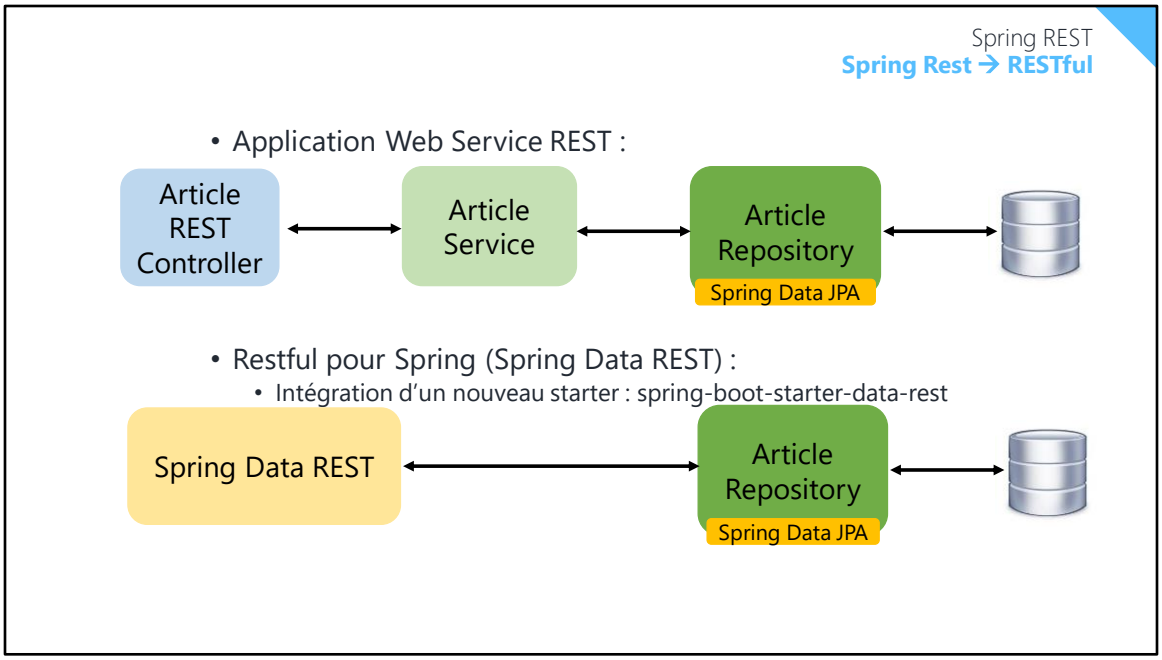
- Ainsi, vous avez une application avec l'architecture Rest complète



Spring Data JPA et Spring REST

- Voyons un premier exemple d'utilisation de Spring Rest.

TP – Cave à Vin
Web Service REST



Jusqu'à présent, nous avons créé un service Web qui gère les opérations de base impliquant les données des articles.
Mais cela ne suffit pas pour rendre les choses "RESTful".

Notre application est plus une solution RPC (Remote Procedure Call).
Il n'y a aucun moyen de savoir comment interagir avec ce service.
Si vous publiez ceci aujourd'hui, vous devrez également rédiger un document ou héberger un portail de développeur quelque part avec tous les détails.

Spring nous propose donc de créer une API Rest en mappant directement les URLs sur le Repository.
Avec cette solution, nous aurons accès sans coder de contrôleur à toutes les méthodes de Repository au travers d'une URL correspondant par défaut à «/articles »

- Attention à la sécurité.

- Voici quelques exemples des méthodes Http exposées

Méthodes Http	Action Spring DATA
Get /articles	Retournes tous les articles
Get /articles/{articleId}	Retourne l'article correspondant à id
Post /articles + données en JSON	Créer un nouvel article
Put /articles/{articleId} + données en JSON	Mise à jour de l'article
Delete /articles/{articleId}	Suppression de l'article
…	

- Spring Data REST permet
  - D'exploiter la pagination, le tri, …
- Dans application.properties, Spring Data REST permet
  - De référencer une url pour exposer les ressources des Repository
  - de spécifier le nombre d'éléments par page

Spring Data REST permet

D'exploiter la pagination, le tri, …

Pagination :

→ Spring Data REST par défaut retourne 20 éléments par page.

→ http://localhost:8080/departments?page=0

Tri :

→ Il est possible de trier selon les différents attributs de la classe.

Dans application.properties, Spring Data REST permet

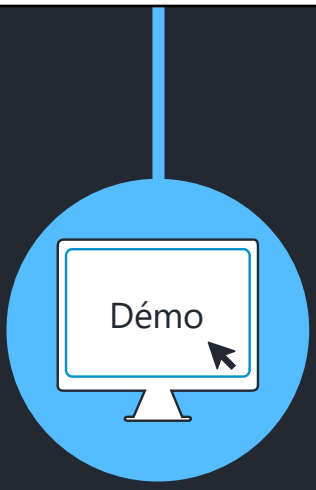
De référencer une url pour exposer les ressources des Repository

exemple : `spring.data.rest.base-path=/api-rest`

de spécifier le nombre d'éléments par page

exemples :

```
spring.data.rest.default-page-size=10
spring.data.rest.max-page-size=15
```



Spring Data REST = API REST

- Création d'une API REST

TP – API REST