**NOTE SUR SPRING MVC :**

Le principe : si on arrive à envoyer une requête http en Java, le Controller peut générer du json et on est capable de récupérer le json dans n’importe quel langage et de le traiter.

Web service Rest

Juste pour bien redef le but d'un controller, sa mission, c'est de catch des requetes Http (le mapping que tu fais actuellement, genre /produit/list ) et de savoir aller chercher / envoyer des infos à tes services, puis te renvoyer un resultat. La en Spring MVC, ton controller doit renvoyer une jsp qui affiche les datas, ou un redirect pour aller vers un autre Controller.

Maintenant, avec des ControllersREST, leur but c'est toujours de catch une requete Http, d'aller communiquer avec tes services, mais maintenant il te retourne et ne recoit que du JSON, et toi coté front tu vas recevoir ce JSON, le traiter et afficher tes données

Curl : envoie une requete http et attend un retour . Commande de base pour un web service. Derrière on met l’url <http://localhost:8080/eshopRest/hello>

Postman permet de faire la meme chose. Curl marche qu’en ligne de commande. Donc ce n’est pas très sexy.

Le but d’un navigateur est d’afficher une page html. Ici j’ai de la data, donc je n’ai pas besoin d’un navigateur. En plus un nav classique ne sait gérer que le get. Alors qu’on doit utiliser d’autres. Pour avoir au post on ne eut pas le faire avec la méthode d’avant sauf en passant par un formulaire.

Body :

* Raw : texte brut
* Json
* …

Intérêt : possibilité de donner u objet en json avec par exemple un nom un prenom et une catégorie qui serait elle-même un objet json avec un nom et une valeur. Et en réalité le web service est capable de recup ce cod json et instancier un objet qui correspond à cette structure. Pour faire ça je vais rajouter ma méthode dans eclipse

On va déclarer dans la méthode un fournisseur en entrant en utilisant RequestBody (on envoie en Json les infos que l’on souhaite cr&éer, spring se charge de le recup et de le transformer en objet java). Comme on recup l’id du fournisseur en obet on envoie le retour de cet methode. A partir du moment où le webservice renvoie un objet, on doit préciser une vue, sinon on se mange une exception.

Dès qu’on a une méthode qui renvoie quelque chose, il faut préciser une vue pour definir qu’elle méthode json est envoyé pour bien convertir.

*@PostMapping*("")

*@JsonView*(JsonViews.Fournisseur.class)

*@ResponseStatus*(code=*HttpStatus*.***CREATED***)

public Fournisseur create(*@Valid* *@RequestBody* Fournisseur fournisseur, BindingResult br){

if(br.hasErrors()) {

throw new ResponseStatusException(*HttpStatus*.***BAD\_REQUEST***);

}

return fournisseurSrv.create(fournisseur);

}

@RequestBody  on peut au moment de la creation d’un objet demander à spring de valider cet objet. Qui dit validation dit recup de ce resultat par un bindingResult. Comme le fournisseur doit forcément avoir un nom et un contact, quand le webservice recup les param de la requet, il est capable de verif qu’il a bien un nom et un contact, sinon il génère une erreur dans le bindingResult.

Le but n’est pas de fait de l’échange d’info : ce sont des traitements que l’on demande. Donc l’url crée ou pas un fournisseur.

* Mise à jour d’un fournisseur

Se fait par la méthode Put. On va demander en put d’envoyer l’id de la ressource que l’on souhaite modifier. Si on est pas capable de fournir l’id, cela veut dire qu’elle n’existe pas et donc passer par un POST. Le reste du code ne change pas. On recup un fournisseur par la requete on peut meme valider le fournisseur avec BondingResult. Si on met un paramètre dans la requete il faut le recup par l’intermédiaire de PathVarible. Une fois recup toutes les infos. On a un service qui permet de mettre à jour le fournisseur.

*@PutMapping*("/{id")

public Fournisseur update(*@Valid* *@RequestBody* Fournisseur fournisseur, BindingResult br, *@PathVariable* Long id) {

fournisseur.setId(id);

return fournisseurSrv.update(fournisseur);

}

* Supprimer un fournisseur

Principe de la requête : même chose que pour le Pu ou la recherche d’un fournisseur. On fait une requete avec en entrée id qui est recup par PathVaraioble. On a ensuite juste a apeler le service grâce à la méthode qui supprime le fournisseur.

*@DeleteMapping*("/{id}")

*@ResponseStatus*(code=*HttpStatus*.***NO\_CONTENT***)

public void delete(*@PathVariable* Long id) {

fournisseurSrv.delete(id);

}

Commande :

Id d’un produit et une quantité. Je dois créer une nouvelle classe ElementPanier avec ces informations. Pour ensuite dans la COntrollerRest de commande je peux mettre mon ElementPanier dans le postMapping pour recupere les informations et les transformer en un objet.

Pour rentrer plusieurs elements de ElmeentPanier dans Json il faut le transformer en tableau 🡪 permet d’envoyer un « liste » qui copntient plusieurs objets. Dans mon postMapping, je dois alors préciser que c’est une lliste<ElmentPanier><