Bases du développement d'API

Controleurs

Afin de répondre à des requêtes HTTP, on utilise des controleurs. Ce sont des classes qui vont contenir des méthodes particulières, les méthodes endpoints. Une méthodes endpoint est une méthode qui gère des requêtes HTTP pour une route et une méthode HTTP donnée. Un controleur peut possèder un préfixe de route qui sera le début de la route gérée par ses méthodes endpoints.

Pour développer une API ReST (par opposition avec une application à vues), nous allons donc utiliser l'annotations `@RestController` sur nos controleurs pour les enregistrer auprès du framework.

Commencez par créer un package `controller` qui contiendra tous nos controleurs. Créez ensuite une nouvelle classe, votre premier controleur :

```
@RestController
@RequestMapping("/api/todos")
public class TodoController {


}
```

Ce controleur est notre controleur de Todos, grâce à l'annotation `@RequestMapping` on déclare qu'il va gérer les requêtes sur les routes qui commencent par `/api/todos`.

Méthode Endpoint

Pour déclarer une méthode Endpoint, il suffit de l'annoter avec `@GetMapping()`, `@PostMapping()`, `@PutMapping()` ... en fonction de la méthode a gérer. La route gérée par la méthode est passée en paramètre de cette annotation. S'il n'est pas renseigné, alors la méthode gère la route racine du controleur pour cette méthode HTTP.

Par défaut dans Spring Boot, lorsque vous retournez un objet d'une méthode endpoint, ce dernier est sérialisé en JSON et le résultat est écrit dans la réponse de la requête.

Exemple :

```
@GetMapping
public List<Todo> getTodos(){
  return Arrays.asList(
    new Todo("3f13bb4c-6d88-4cc5-97c8-868569ac2e94","todo 1","todo 1 description"),
    new Todo("e23d5839-1299-4334-ba35-2a9c62ef17a3","todo 2","todo 2 description")
    );
}
```

Cette méthode Endpoint n'as pas de route précisée, elle va donc gérer la route racine du constructeur pour la méthode GET. Elle retourne également une liste de Todo, cette dernière va être automatique transformée en JSON, le résultat de la requête sera donc :

```
[
  {
    "id":"3f13bb4c-6d88-4cc5-97c8-868569ac2e94",
    "name":"todo 1",
    "description":"todo 1 description"
  },
  {
    "id":"e23d5839-1299-4334-ba35-2a9c62ef17a3",
    "name":"todo 2",
    "description":"todo 2 description"
  }
]
```

Paramètre d'URL

Une méthode endpoint peut récupérer un paramètre d'URL de la requête grâce à l'annotation `@RequestParameter` qui prend en paramètre le nom du paramètre :

```
@GetMapping
public List<Todo> getTodos(@RequestParam("name") String todoName){
  ...
}
```

Cette méthode récupère en paramètre `todoName` le paramètre d'URL name de la requête.

Paramètre de route

Une méthode endpoint peut récupérer un paramètre dans le chemin de la requête. Le nom du paramètre est template dans la route de méthode avec des accolades. Il est ensuite récupéré par la méthode grâce à l'annotation `@PathVariable` qui prend en paramètre le nom templaté dans la route :

```
@GetMapping("/{id}")
public Todo getTodoFromId(@PathVariable("id") String id){
...
}
```

Cette méthode ne gère plus les requêtes GET sur `/api/todos` mais sur `/api/todos/quelquechose`. La méthode va récupérer ce "quelquechose" en paramètre.

Body de la requête

Les méthodes Endpoint qui gère des requêtes dont la méthode HTTP peut contenir un Body peuvent récupérer ce Body sous la forme d'un objet Java avec l'annotation `@RequestBody` :

```
@PostMapping()
public Todo createTodo(@RequestBody Todo todo){
...
}
```

Ici, lors d'une requête POST sur `/api/todos`, la Framework va essayer de désérialiser le Body de la requête (au format JSON) dans un objet Java de type `Todo`.

Revision #9
Created 2021-01-22 09:46:09 UTC by Arsène Lapostolet
Updated 2021-01-28 15:01:04 UTC by Arsène Lapostolet