Créer des API RESTful avec Spring Web

9. Construire des API RESTful avec Spring Web

L'un des usages les plus courants de Spring Boot est la création d'**API RESTful**. Une API REST permet à différents systèmes (comme un dashboard d'administration Angular ou une application mobile Flutter) de communiquer avec votre serveur backend (Spring Boot) via le protocole HTTP, en échangeant généralement des données au format **JSON**.

Le module `spring-boot-starter-web` simplifie énormément ce processus. Il configure automatiquement un serveur web (Tomcat) et inclut une bibliothèque appelée **Jackson**, qui s'occupe de transformer (sérialiser/désérialiser) vos objets Java en JSON, et inversement, de manière totalement transparente.

Les annotations essentielles pour les API REST :

- `@RestController`: Combine `@Controller` et `@ResponseBody`. Elle indique que toutes les méthodes de cette classe renverront directement des données (JSON) au client, et non une vue HTML.
- `@RequestMapping`: Permet de définir le préfixe de l'URL (la route de base) pour tout le contrôleur.
- `@GetMapping`, `@PostMapping`, `@PutMapping`, `@DeleteMapping`: Définissent les routes spécifiques en fonction de la méthode HTTP utilisée.
- `@RequestBody`: Indique que les données envoyées par le client dans le corps de la requête (souvent en JSON) doivent être converties en un objet Java.
- `@PathVariable`: Permet d'extraire une valeur dynamique directement depuis l'URL (ex: `/api/suspects/{id}`).

10. Cas pratique : Création d'une API de gestion de suspects

Pour illustrer cela, imaginons que nous développons l'API pour un dossier spécifique intitulé "*The Vanishing Diplomat*". Nous voulons pouvoir consulter la liste des suspects (GET) et ajouter un nouveau suspect au dossier (POST).

Étape 1 : Créer le Modèle de données (POJO)

Commençons par définir ce qu'est un "Suspect". Créez une simple classe Java qui représentera la structure de nos données.

apcpedagogie.com

```
package com.studiogames.mysterybackend.model;

public class Suspect {
    private String id;
    private String nom;
    private String description;
    private int niveauSuspicion; // de 1 à 100

    // Constructeur vide (nécessaire pour la conversion JSON par Jackson)
    public Suspect() {}

    public Suspect(String id, String nom, String description, int niveauSuspicion) {
        this.id = id;
        this.nom = nom;
        this.description = description;
        this.niveauSuspicion = niveauSuspicion;
    }

    // Getters et Setters (Indispensables pour que Spring puisse lire/écrire le JSON)
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getNom() { return nom; }
    public void setNom(String nom) { this.nom = nom; }

    public String getDescription() { return description; }
    public void setDescription(String description) { this.description = description; }

    public int getNiveauSuspicion() { return niveauSuspicion; }
    public void setNiveauSuspicion(int niveauSuspicion) { this.niveauSuspicion =
niveauSuspicion; }
}
```

Étape 2 : Créer le Contrôleur REST (Endpoints GET et POST)

Maintenant, créons le contrôleur qui va exposer nos routes web et manipuler les suspects.

```
package com.studiogames.mysterybackend.controller;

import com.studiogames.mysterybackend.model.Suspect;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("/api/vanishing-diplomat/suspects") // Route de base du contrôleur
public class SuspectController {

    // Une liste temporaire en mémoire pour simuler une base de données
    private List<Suspect> suspectsDatabase = new ArrayList<>();

    public SuspectController() {
        // Initialisation avec des données de test pour notre étape 1
        suspectsDatabase.add(new Suspect("S01", "Victor Thorne", "Vu près du bureau
```

```
diplomatique avec une valise sombre.", 75));
        suspectsDatabase.add(new Suspect("S02", "Elena Rostova", "Refuse de fournir un alibi
pour la nuit de la disparition.", 85));
    }

    /**
     * 1. Requête GET : Récupérer des données
     * Endpoint : GET http://localhost:8080/api/vanishing-diplomat/suspects
     */
    @GetMapping
    public List<Suspect> getAllSuspects() {
        // Spring va automatiquement transformer cette liste Java en un tableau JSON
        return suspectsDatabase;
    }

    /**
     * 2. Requête GET avec PathVariable : Récupérer un suspect spécifique
     * Endpoint : GET http://localhost:8080/api/vanishing-diplomat/suspects/S01
     */
    @GetMapping("/{id}")
    public Suspect getSuspectById(@PathVariable String id) {
        return suspectsDatabase.stream()
                .filter(s -> s.getId().equals(id))
                .findFirst()
                .orElse(null); // Dans un vrai projet, on retournerait une erreur 404 (Not
Found)
    }

    /**
     * 3. Requête POST : Envoyer de nouvelles données au serveur
     * Endpoint : POST http://localhost:8080/api/vanishing-diplomat/suspects
     */
    @PostMapping
    public Suspect addSuspect(@RequestBody Suspect nouveauSuspect) {
        // @RequestBody indique à Spring de prendre le JSON envoyé par le client
        // et de le transformer en objet 'Suspect'
        suspectsDatabase.add(nouveauSuspect);

        // On retourne l'objet créé en guise de confirmation (souvent accompagné d'un code
HTTP 201 Created)
        return nouveauSuspect;
    }
}
```

Comment tester cela ?

Pour tester la requête **GET**, il suffit d'ouvrir l'URL `http://localhost:8080/api/vanishing-diplomat/suspects` dans votre navigateur. Vous obtiendrez un beau JSON contenant Victor et Elena.

Pour la requête **POST** (l'ajout d'un suspect), un navigateur web ne suffit pas. Vous devez utiliser un outil comme **Postman**, **Insomnia**, ou l'extension **Thunder Client** dans VS Code pour construire une requête POST et inclure un objet JSON dans l'onglet *Body*.

Exemple de ce qui se passe sous le capot (La magie de Jackson)

Lorsque votre client (votre interface) fait un `GET` sur la liste des suspects, le framework Spring intercepte la `List<Suspect>` renvoyée par Java et la convertit automatiquement dans ce format JSON universel :

```
[
  {
    "id": "S01",
    "nom": "Victor Thorne",
    "description": "Vu près du bureau diplomatique avec une valise sombre.",
    "niveauSuspicion": 75
  },
  {
    "id": "S02",
    "nom": "Elena Rostova",
    "description": "Refuse de fournir un alibi pour la nuit de la disparition.",
    "niveauSuspicion": 85
  }
]
```

*Rédigé et structuré par : **Med Khalil Kribi***

Source : https://apcpedagogie.com/creer-api-restful-spring-web · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com