
TP n°5 : RESTful en Java

💡 Objectif

Ce TP sera enseigné en mettant l'accent sur :

- ✓ La création d'un service Web RESTful en Java à l'aide de Spring Boot
- ✓ La manipulation des méthodes GET, POST, PUT et DELETE

🔔 Rappel

- L'architecture **REST (Representational State Transfer)** définit un ensemble de principes architecturaux par lesquels vous pouvez concevoir des services Web qui se concentrent sur les ressources d'un système, y compris la façon dont les états des ressources sont adressés et transférés via HTTP par un large éventail de clients écrits dans différentes langues.
- **REST** a été largement utilisé, remplaçant les services Web basés sur SOAP et WSDL.
- Un **service Web RESTful** est un service Web qui met en œuvre l'architecture REST.
- La fonction de base d'un service Web RESTful est la même que celle de la navigation sur Internet.
- Les étapes générales de tout appel d'un **service Web RESTful** :
 - ✓ Le client envoie une requête au serveur.
 - ✓ Le serveur reçoit la demande et la traite en interne.
 - ✓ Le serveur renvoie une réponse au client. La réponse contient des informations qui indiquent au client si la demande a abouti. La réponse comprend également toute information demandée par le client.
- Les **services Web RESTful** exigent que les demandes contiennent les principaux composants suivants :
 - ✓ **Identifiant unique de la ressource (URI)** : Pour les services REST, le serveur effectue généralement l'identification des ressources en utilisant un localisateur de ressources uniformes (URL, Uniform Resource Locator). L'URL spécifie le chemin d'accès à la ressource. Elle est similaire à l'adresse du site Web.
 - ✓ **Méthode http** : Une méthode HTTP indique au serveur ce qu'il doit faire à la ressource. Voici quatre méthodes HTTP courantes :
 1. GET : Les clients utilisent GET pour accéder aux ressources qui se trouvent à l'URL spécifié sur le serveur.
 2. POST : Les clients utilisent POST pour envoyer des données au serveur.
 3. PUT : Les clients utilisent PUT pour mettre à jour les ressources existantes sur le serveur.
 4. DELETE : Les clients utilisent la requête DELETE pour prélever la ressource existante sur le serveur. Une requête DELETE peut modifier l'état du serveur.

✍ Travail demandé

Vous devez préparer un compte rendu « CR-TP-4-5-*Prénom-Nom*.pdf » qui sera envoyé à la fin de la séance par mail dont l'objet est « TP-4-5-ING-A2-*N°groupe-N°sousGroupe* ».

Le compte rendu doit contenir les points suivants pour chaque exercice :

- ✓ les classes implémentées en capture d'écran
- ✓ les résultats d'exécution affichés dans SoapUI (*Calculator avec SOAP*) et dans votre navigateur (*Premiers pas avec RESTful*) en capture d'écran.

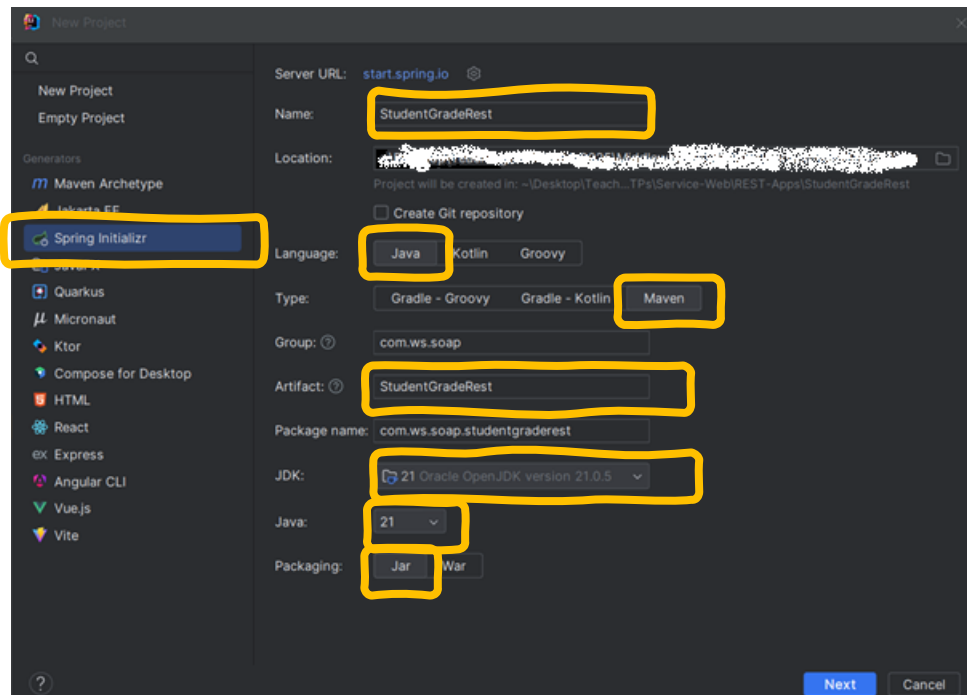
Énoncé

Exercice 1 : Premiers pas avec RESTful

NB : cet exercice est un simple tutoriel permettant de mettre en œuvre un service Web RESTful en Java.

Un exemple simple d'application qui affiche la liste des étudiants sur une page Web accessible depuis un navigateur et propose d'autres pages permettant d'ajouter, mettre à jour et supprimer (CRUD) des étudiants.

- 1- Créez un projet Spring Boot basé sur Maven « *StudentGradeRest* » tout en ajoutant les dépendances « Spring Web Services » et « Spring Boot DevTools ».



Sous le package « *com.ws.soap.studentgraderest* » :

- 2- Créez une classe nommée « *Student.java* » qui représente les données d'une entité « étudiant » avec ces attributs : « id », « name » et « grade » et fournit les fonctions getter et setter.
- 3- Créez et complétez la classe « *StudentRepository.java* » qui ajoute l'attribut suivant :

```
private static Map<Integer, Student> studentRepo = new HashMap<>();
```

Cette classe et son attribut « *studentRepo* » vont permettre de gérer une base de données d'étudiants "en mémoire".

 Notez bien que l'identifiant de l'étudiant de la classe « *Student* » servira de clé à la map data.

Dans cette classe, il y a :

- i. Un constructeur qui initialise le répertoire avec des données des étudiants,
- ii. Une méthode « *viewAllStudent* » qui renvoie la liste des étudiants du répertoire,
- iii. Une méthode « *addNewStudent* » qui ajoute un nouvel étudiant dans le répertoire,
- iv. Une méthode « *updateStudent* » qui mis à jour la note « *grade* » d'un étudiant dans le répertoire,
- v. Une méthode « *deleteStudent* » qui supprime un étudiant dans le répertoire.

```
public class StudentRepository
{
    9 usages
    private static Map<Integer, Student> studentRepo = new HashMap<>();
    1 usage
    public StudentRepository()
    {
        studentRepo.put(1, new Student( id: 1, name: "FirstStudent", grade: 10));
        studentRepo.put(2, new Student( id: 2, name: "SecondStudent", grade: 11));
        studentRepo.put(3, new Student( id: 3, name: "ThirdStudent", grade: 12));
    }

    1 usage
    public Map<Integer, Student> viewAllStudent()
    {
        return studentRepo;
    }

    1 usage
    public Map<Integer, Student> addNewStudent(Student newStudent) {
        //Complétez le code
        //...
        return studentRepo;
    }

    1 usage
    public Map<Integer, Student> updateStudent(String name, double grade) {

        Student newStudent=null;
        int key = 0;

        //Complétez le code
        //...

        studentRepo.put(key,newStudent);
        return studentRepo;
    }

    1 usage
    public Map<Integer, Student> deleteStudent(String name)
    {
        int key = 0;

        //Complétez le code
        //...

        studentRepo.remove(key);
        return studentRepo;
    }
}
```

4- Créez une classe « *StudentController.java* ».

Dans cette classe, intégrez les méthodes CRUD qui permettent de manipuler les données des étudiants : une pour ajouter, une pour modifier, une pour consulter le répertoire et enfin une pour supprimer un étudiant.

Ces méthodes auront comme signature respective :

- i. **Create:** *public Map addStudent(@RequestParam int id, @RequestParam String name, @RequestParam double grade)*
- ii. **Read:** *public Map getAllStudents()*
- iii. **Update:** *public Map updateStudentGrade(@RequestParam String name, @RequestParam double grade)*
- iv. **Delete:** *public Map deleteStudentGrade(@RequestParam String name)*

```
@RestController
@RequestMapping("/ws/grade")
public class StudentController
{
    // usages
    private static StudentRepository Students;

    @GetMapping(path="/index")
    public String studentRepositoryInitializer()
    {
        Students= new StudentRepository();
        return String.format("Hey there! Congratulations on creating your first Spring Boot Rest Service to initialize your students' repository...");
    }

    // GET endpoint to get all student
    @RequestMapping("/allStudent")
    @GetMapping
    public Map getAllStudents()
    {
        return Students.viewAllStudent();
    }

    // POST endpoint to add a new student
    @RequestMapping("/addStudent")
    @PostMapping
    public Map addStudent(@RequestParam int id, @RequestParam String name, @RequestParam double grade)
    {
        Student newStd=new Student(id, name, grade);
        return Students.addNewStudent(newStd);
    }

    // PUT endpoint to update a student's grade
    @RequestMapping(value="/updateStudent")
    @PutMapping
    public Map updateStudentGrade(@RequestParam String name, @RequestParam double grade)
    {
        return Students.updateStudent(name, grade);
    }

    // DELETE endpoint to delete a student
    @RequestMapping(value="/deleteStudent")
    @DeleteMapping
    public Map deleteStudentGrade(@RequestParam String name)
    {
        return Students.deleteStudent(name);
    }
}
```

5- Démarrez l'application :

Méthode 1 :



Effectuez la compilation de maven à l'aide de « mvn clean install » et démarrez l'application à l'aide de la commande « java -jar .\target\studentGrade-0.0.1-SNAPSHOT.jar ».

Ou

Méthode 2 :

Exécutez tout simplement la classe « *StudentGradeRestApplication* » en tant qu'application Java.

6- Testez avec votre navigateur les différents services Web :

- i. Initialisez le répertoire des étudiants,
 <http://localhost:8080/ws/grade/index>
- ii. Affichez la liste des étudiants du répertoire,
- iii. Ajoutez cet étudiant : « 4 », « *FourthStudent* » et « 14 »,
 <http://localhost:8080/ws/grade/addStudent?id=4&name=FourthStudent&grade=14>
- iv. Mettez à jour la note de l'étudiant « *FourthStudent* »,
- v. Supprimez l'étudiant « *SecondStudent* »,
- vi. Réaffichez la liste des étudiants du répertoire.