Online Event Management - Put and Delete

Description
Objective:

To work with @RestController, http methods - PUT and DELETE

Concept Explanation:

1. **HTTP PUT** method is like web's handy "overwrite" button. It's perfect for times when you need to give something a complete makeover. Often used for completely replacing an existing resource with the data provided in the request body.

2. The **HTTP PATCH** method is akin to a "quick fix" tool on the web. It's perfect for making partial updates to existing records without needing to send the entire object.

3. The **HTTP DELETE** method acts as the web's "erase" button. It's used to remove data from the server, ensuring that unwanted or obsolete information can be cleaned up from your application.

Concept Implementation:

1. In the REST Controller, named **EventController** for Event Management, we should map the endpoint **/OEMS/updateEventType/3/Wedding** to the annotation corresponding to the or **HTTP PATCH** request, as it is designed for updating the event type of an existing event.

2. Additionally, we should map the endpoint **/OEMS/deleteEvent/4** to the annotation corresponding to the **HTTP DELETE** request, which is used for removing an event from the system based on the given event ID.

Online Event Management

Online Event Management System desires to use a web service for adding new events to their schedule ,retrieving details of existing events, modifying the event types and delete the event. Help them automate this process by developing a REST Service using Maven to add and retrieve event details.

As an enhancement to the previous requirements, EventController, which is the RestController, should now include services to modify and delete event details.

Request URL --> /OEMS/updateEventType/3/Wedding: This service should invoke the **updateEventType** method of the EventService class and update the event type for the given event ID.

Request URL --> /OEMS/deleteEvent/4: This service should delete the event by invoking **deleteEvent** method of the EventService class for the given event ID from the list.

Note:

- The EventService class is provided with four sample events added to a list as part of the skeleton. Do not alter the same.
- Partial Maven solution for the same is provided. Do not alter the className/packageName/methodName.
- You can add new methods/attributes/classes if required.
- In the EventService class, include the required business logic.
- The data returned from the Controller should be JSON.
- Inject the EventService inside the EventController and invoke the appropriate methods.

EventManagement/pom.xml

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 3         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
 4         <modelVersion>4.0.0</modelVersion>
 5         <parent>
 6                 <groupId>org.springframework.boot</groupId>
 7                 <artifactId>spring-boot-starter-parent</artifactId>
 8                 <version>3.0.6</version>
 9                 <relativePath/> <!-- lookup parent from repository -->
10         </parent>
11         <groupId>com.example.demo</groupId>
12         <artifactId>EventManagement</artifactId>
13         <version>0.0.1-SNAPSHOT</version>
14         <name>EventManagement</name>
15         <description>EventManagement</description>
16         <properties>
17                 <java.version>17</java.version>
18         </properties>
19         <dependencies>
20
21                 <dependency>
22                         <groupId>org.springframework.boot</groupId>
23                         <artifactId>spring-boot-starter-web</artifactId>
24                 </dependency>
25                 <dependency>
26                         <groupId>org.springframework.boot</groupId>
27                         <artifactId>spring-boot-starter-test</artifactId>
28                         <scope>test</scope>
```

```
                </dependency>

                <dependency>
                        <groupId>org.springframework.restdocs</groupId>
                        <artifactId>spring-restdocs-mockmvc</artifactId>
                        <scope>test</scope>
                </dependency>

                <dependency>
                        <groupId>junit</groupId>
                        <artifactId>junit</artifactId>
                        <version>4.12</version>
                        <scope>test</scope>
                </dependency>

        </dependencies>


        <build>
                <plugins>

                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>


</project>
```

EventManagement/src/main/java/com/controller/EventController.java

```
package com.controller;

import java.util.List;

import com.model.Event;
import com.service.EventService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/OEMS")
public class EventController {

        @Autowired
        private EventService eventService;

      @PutMapping("/updateEventType/{eventId}/{eventType}")
   public boolean updateEventType(@PathVariable Integer eventId,@PathVariable String eventType) {
      return eventService.updateEventType(eventId, eventType);
   }

   @DeleteMapping("/deleteEvent/{eventId}")
   public boolean deleteEvent(@PathVariable Integer eventId) {
      return eventService.deleteEvent(eventId);
   }

}
```

EventManagement/src/main/java/com/example/demo/EventManagementSystemApplication.java

```java
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan("com.controller com.service")
public class EventManagementSystemApplication {

	public static void main(String[] args) {

			SpringApplication.run(EventManagementSystemApplication.class, args);

	}
}
```

EventManagement/src/main/java/com/model/Event.java

```java
package com.model;

public class Event {

    private int eventId;
    private String eventName;
    private String eventType;
    private String eventDate;


	public Event(int eventId, String eventName, String eventType, String eventDate) {
				super();
				this.eventId = eventId;
				this.eventName = eventName;
				this.eventType = eventType;
				this.eventDate = eventDate;
			}


	public Event()
	{

	}


	public int getEventId() {
			return eventId;
	}


	public void setEventId(int eventId) {
			this.eventId = eventId;
	}


	public String getEventName() {
			return eventName;
	}


	public void setEventName(String eventName) {
			this.eventName = eventName;
```

```
	}


	public String getEventType() {
		return eventType;
	}


	public void setEventType(String eventType) {
		this.eventType = eventType;
	}


	public String getEventDate() {
		return eventDate;
	}


	public void setEventDate(String eventDate) {
		this.eventDate = eventDate;
	}


}
```

EventManagement/src/main/java/com/service/EventService.java

```
package com.service;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Component;

import com.model.Event;
import java.util.stream.Collectors;

@Component
public class EventService {

	ArrayList<Event> eventList = new ArrayList<Event>();



	public ArrayList<Event> getEventList() {
		return eventList;
	}

	public void setEventList(ArrayList<Event> eventList) {
		this.eventList = eventList;
	}

	public EventService(){
		Event eventObj1 = new Event(1, "Tech Conference", "Conference", "2024-03-15");
	Event eventObj2 = new Event(2, "Music Festival", "Festival", "2024-06-20");
		Event eventObj3 = new Event(3, "Art Exhibition", "Exhibition", "2024-09-10");
	Event eventObj4 = new Event(4, "Literature Workshop", "Workshop", "2024-11-05");
	eventList.add(eventObj1);
	eventList.add(eventObj2);
	eventList.add(eventObj3);
	eventList.add(eventObj4);

	}


```

```
38          public boolean updateEventType(int eventId, String eventType) {
39
40              for(Event event: eventList){
41                  if(event.getEventId()==eventId){
42                      event.setEventType(eventType);
43                      return true;
44                  }
45              }
46                      return false;
47              }
48
49              public boolean deleteEvent(int eventId) {
50                  List<Event> updatedList=eventList.stream()
51                                  .filter(event->event.getEventId()!=eventId)
52                                  .collect(Collectors.toList());
53                  if(updatedList.size()!=eventList.size()){
54                      eventList=new ArrayList<>(updatedList);
55                      return true;
56                  }
57                   return false;
58              }
59  }
```

EventManagement/src/main/resources/application.properties

```
1 server.port=5090
2
```

Grade

Reviewed on Tuesday, 22 April 2025, 8:43 AM by Automatic grade

Grade 100 / 100