Controller Upgrades

Let’s add the support for few more HTTP types such as PUT/PATCH/DELETE that will do its respective operations on the Employee Entity objects of my table.

Before we start there are some mis-writtern fields from the DTO and Entity of the Employee that needs to be corrected for the fields with `is` pre-fix as jackson will not treat them as valid values.

```
@Entity
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Table(name = "employees")
public class EmployeeEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    private String email;

    private Integer age;

    private LocalDate dateOfJoining;

    @JsonProperty("isActive")
```

```
    private boolean active;
}
```

Now to simplify the DTO as well; we can add lombok and update the mappings:

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class EmployeeDTO {
    private long id;

    private String name;

    private String email;

    private Integer age;

    private LocalDate dateOfJoining;

    @JsonProperty("isActive")
    private boolean active;
}
```

Now that the plumbing work is done we move to the core topics of adding multiple new HTTP methods to handle on an entity at the controller level first.

```
@PutMapping(path = "/{employeeId}")
public EmployeeDTO updateEmployeeById(
        @RequestBody EmployeeDTO employeeDTO,
        @PathVariable Long employeeId
    ) {

    return employeeService.updateEmployeeById(employeeId, employeeDT
O)
```

```
}

@DeleteMapping(path = "/{employeeId}")
public boolean deleteEmployeeById(@PathVariable Long employeeId) {
   return employeeService.deleteEmployeeById(employeeId);
}

@PatchMapping(path = "/{employeeId}")
public EmployeeDTO updatePartialEmployeeById(
         @RequestBody Map<String, Object> employeeUpdates,
         @PathVariable Long employeeId
      ) {

   return employeeService.updatePartialEmployeeById(employeeId, employ
eeUpdates);

}
```

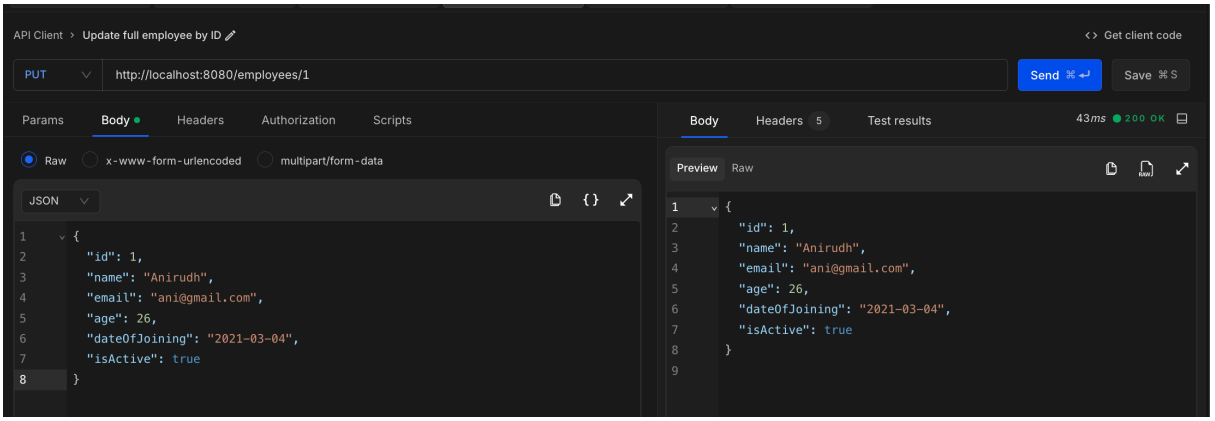
To double down on them and see what these abstractions are doing lets dive into the service layer implementations:

```
// PUT Mapping
public EmployeeDTO updateEmployeeById(Long employeeId, EmployeeDT
O employeeDTO) {
   EmployeeEntity employeeEntity = modelMapper.map(employeeDTO, Emp
loyeeEntity.class);
  employeeEntity.setId(employeeId);

  EmployeeEntity savedEmployeeEntity = employeeRepository.save(employ
eeEntity);

  return modelMapper.map(savedEmployeeEntity, EmployeeDTO.class);
}

// DELETE Mapping
public boolean deleteEmployeeById(Long employeeId) {
   boolean existsById = isExistsByEmployeeId(employeeId);
```

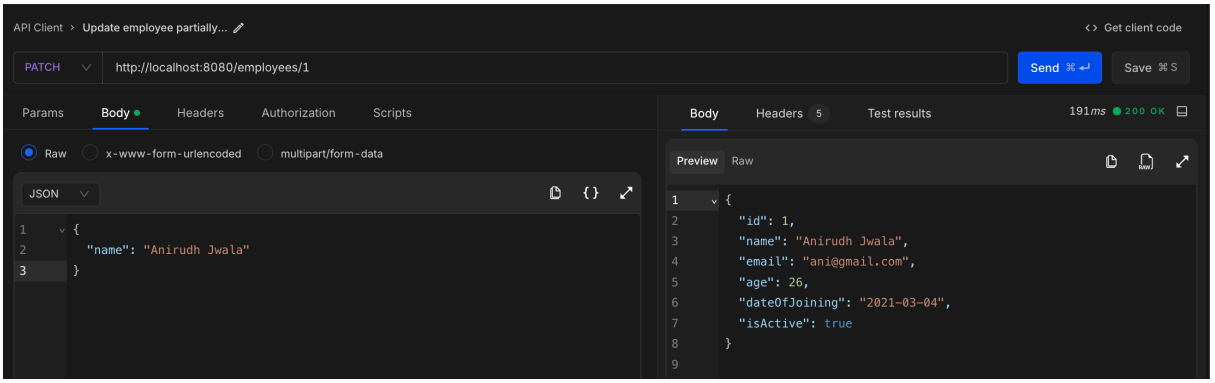
```
  if (!existsById) {
     return false;
  }

  employeeRepository.deleteById(employeeId);
  return true;
}

// PATCH Mapping
public EmployeeDTO updatePartialEmployeeById(
        Long employeeId,
        Map<String, Object> employeeUpdates
    ) {
    boolean existsById = isExistsByEmployeeId(employeeId);

  if (!existsById) {
        return null;
    }

  EmployeeEntity employeeEntity = employeeRepository.findById(employeeI
d).get();
  employeeUpdates.forEach((field, value) -> {
     Field fieldToUpdate = ReflectionUtils.findField(EmployeeEntity.class, fiel
d);

    if (fieldToUpdate == null) {
        return;
        }

    fieldToUpdate.setAccessible(true);
    ReflectionUtils.setField(fieldToUpdate, employeeEntity, value);
    });

    employeeRepository.save(employeeEntity);

    return modelMapper.map(employeeEntity, EmployeeDTO.class);
}
```

```
// re-usable utility method
public boolean isExistsByEmployeeId(Long employeeId) {
    return employeeRepository.existsById(employeeId);
}
```

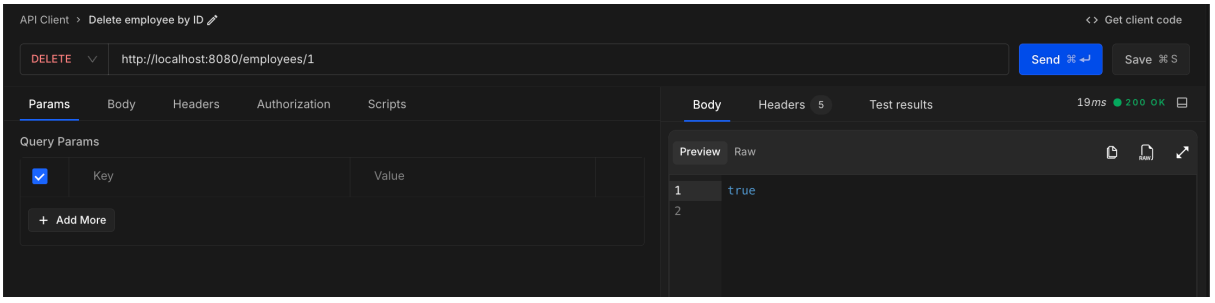
PUT Mapping Demo



PATCH Mapping Demo



DELETE Mapping Demo



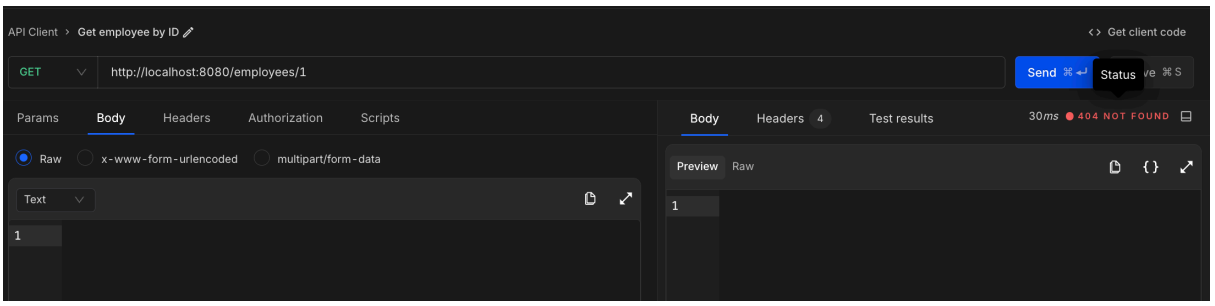
Sending back Custom Status Codes

We make use of something called as `ResponseEntity` that allows you to send HTTP status codes of 200 OK, 404 Not Found, 403 Forbidden, 201 Created, etc.

Let’s looks at them as we do in our existing controller layer:

Get a single employee record by ID

```
@GetMapping(path = "/{employeeId}")
public ResponseEntity<EmployeeDTO> getEmployeeById(@PathVariable Lo
ng employeeId) {
   Optional<EmployeeDTO> employeeDTO = employeeService.getEmploye
eById(employeeId);

   return employeeDTO
               .map(employeeDTO1 -> ResponseEntity.ok(employeeDTO1)) //
200
         .orElse(
            // 404
            ResponseEntity.notFound().build()
         );
}
```



Get all employee records

```
@GetMapping
public ResponseEntity<List<EmployeeDTO>> getAllEmployees() {
    // 200
    return ResponseEntity.ok(employeeService.getAllEmployees());
}
```

Create a new employee record

```
@PostMapping
public ResponseEntity<EmployeeDTO> createNewEmployee(@RequestBod
y EmployeeDTO inputEmployee) {
    EmployeeDTO savedEmployee = employeeService.createNewEmployee(i
nputEmployee);

    // 201
  return new ResponseEntity<>(savedEmployee, HttpStatus.CREATED);
}
```

Update an employee record

```
@PutMapping(path = "/{employeeId}")
public ResponseEntity<EmployeeDTO> updateEmployeeById(
        @RequestBody EmployeeDTO employeeDTO,
        @PathVariable Long employeeId
    ) {

    // 200
    return ResponseEntity.ok(
            employeeService.updateEmployeeById(employeeId, employeeDTO)
    );

}
```

Delete an employee record

```
@DeleteMapping(path = "/{employeeId}")
public ResponseEntity<Boolean> deleteEmployeeById(@PathVariable Long
employeeId) {
   boolean gotDeleted = employeeService.deleteEmployeeById(employeeI
d);

   if (gotDeleted) {
      // 200
     return ResponseEntity.ok(true);
   }

   // 404
  return ResponseEntity.notFound().build();
}
```

Update partial fields of an employee record

```
@PatchMapping(path = "/{employeeId}")
public ResponseEntity<EmployeeDTO> updatePartialEmployeeById(
       @RequestBody Map<String, Object> employeeUpdates,
       @PathVariable Long employeeId
   ) {
   EmployeeDTO employeeDTO = employeeService.updatePartialEmployee
ById(employeeId, employeeUpdates);

   if  (employeeDTO == null) {
      // 404
      return ResponseEntity.notFound().build();
   }

   // 200
  return ResponseEntity.ok(employeeDTO);
}
```