Exercise 4: Implementing CRUD Operations

Understanding the Requirements

We'll create RESTful endpoints to perform CRUD operations on `Employee` and `Department` entities using Spring MVC and the repositories created in the previous exercise.

Creating Controllers

Java

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import java.util.List;

@RestController
@RequestMapping("/employees")
public class EmployeeController {

    @Autowired
    private EmployeeRepository employeeRepository;

    @GetMapping
    public List<Employee>   getAllEmployees() {
        return employeeRepository.findAll();
    }

    @GetMapping("/{id}")
    public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id)
{
        return   employeeRepository.findById(id)
                .map(ResponseEntity::ok)
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

    @PostMapping
    public Employee createEmployee(@RequestBody Employee employee) {
        return   employeeRepository.save(employee);
    }

    @PutMapping("/{id}")
    public   ResponseEntity<Employee> updateEmployee(@PathVariable Long
id, @RequestBody Employee updatedEmployee) {
        return employeeRepository.findById(id)
                .map(employee -> {
                    employee.setName(updatedEmployee.getName());
                    employee.setEmail(updatedEmployee.getEmail());

employee.setDepartment(updatedEmployee.getDepartment());
                    return
ResponseEntity.ok(employeeRepository.save(employee));
                })
                .orElseGet(() -> ResponseEntity.notFound().build());
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteEmployee(@PathVariable Long id)   {
        employeeRepository.deleteById(id);
        return ResponseEntity.noContent().build();
    }
}
```

Similarly, you can create a `DepartmentController` with analogous methods for department CRUD operations.

Explanation of the Code

- **Controller Class:** The `EmployeeController` class is annotated with `@RestController` to indicate that it handles REST requests.
- **Repository Injection:** The `EmployeeRepository` is injected using `@Autowired`.
- **Endpoints:**
  - `@GetMapping`: Maps HTTP GET requests to retrieve all employees or a specific employee by ID.
  - `@PostMapping`: Maps HTTP POST requests to create a new employee.

    - `@PutMapping`: Maps HTTP PUT requests to update an existing employee.
    - `@DeleteMapping`: Maps HTTP DELETE requests to delete an employee.
- **ResponseEntity:** Used to return appropriate HTTP status codes along with the response body.
- **Error Handling:** The `ResponseEntity` is used to handle cases where the employee is not found (404 Not Found).

Additional Considerations

- **Exception Handling:** Consider adding global exception handlers to handle potential exceptions like `EntityNotFoundException`.
- **Input Validation:** Implement input validation to ensure data integrity.
- **Security:** For production environments, implement security measures like authentication and authorization.
- **Response Formatting:** Consider using JSON or XML for response formatting based on requirements.
- **Testing:** Write unit and integration tests to ensure the correctness of the code.