

# FullStack Application

## Spring Boot CRUD with MongoDB

### Packages Used

1. **entity** – contains the **data model** (BatchEntry.java)
2. **repository** – contains the **MongoDB repository interface** (BatchEntryRepository.java)
3. **service** – contains the **business logic** (BatchEntryService.java)
4. **controller** – exposes the **REST API endpoints** (BatchEntryController.java)

### 1. Entity – BatchEntry.java

**Location:** entity/BatchEntry.java

#### **Purpose:**

Defines the structure of the **MongoDB document** that is stored in the collection.

#### **Code Explanation:**

```
package entity;

import org.springframework.data.annotation.Id;
import org.springframework.data.mongodb.core.mapping.Document;

@Document(collection = "batch_entry")
public class BatchEntry
{
    @Id
    private String id;
    private String batchName;
    private String duration;
    private double fees;

    // Constructors
    public BatchEntry() {}

    public BatchEntry(String batchName, String duration, double fees)
    {
        this.batchName = batchName;
        this.duration = duration;
        this.fees = fees;
    }
}
```

```
}  
  
// Getters and Setters  
// (Used for encapsulation and JSON mapping)  
}
```

### Key Annotations:

- @Document: Maps this class to a MongoDB collection.
- @Id: Marks id as the unique identifier (primary key).

## 2. Repository – BatchEntryRepository.java

Location: repository/BatchEntryRepository.java

### Purpose:

Handles **direct database operations** using Spring Data MongoDB.

### Code Explanation:

```
package repository;  
  
import entity.BatchEntry;  
import org.springframework.data.mongodb.repository.MongoRepository;  
  
public interface BatchEntryRepository extends  
MongoRepository<BatchEntry, String>  
{  
    // No need to write any code, CRUD methods are available by  
    default.  
}
```

### Key Interface:

- MongoRepository<T, ID>: Offers all CRUD operations (save, findAll, findById, deleteById, etc.)

## 3. Service – BatchEntryService.java

Location: service/BatchEntryService.java

### Purpose:

Implements **business logic** and interacts with the repository.

## Code Explanation:

```
package service;

import entity.BatchEntry;
import repository.BatchEntryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
public class BatchEntryService
{
    @Autowired
    private BatchEntryRepository repo;

    public BatchEntry save(BatchEntry batch)
    {
        return repo.save(batch);
    }

    public List<BatchEntry> getAll()
    {
        return repo.findAll();
    }

    public Optional<BatchEntry> getById(String id)
    {
        return repo.findById(id);
    }

    public BatchEntry update(String id, BatchEntry updated)
    {
        updated.setId(id);
        return repo.save(updated); // save() updates if ID exists
    }

    public void delete(String id)
    {
        repo.deleteById(id);
    }
}
```



## 4. Controller – BatchEntryController.java

**Location:** controller/BatchEntryController.java

### **Purpose:**

Handles **HTTP requests** and maps them to the service layer.

### **Code Explanation:**

```
package controller;

import entity.BatchEntry;
import service.BatchEntryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/batches")
public class BatchEntryController
{
    @Autowired
    private BatchEntryService service;

    @PostMapping
    public BatchEntry add(@RequestBody BatchEntry batch)
    {
        return service.save(batch);
    }

    @GetMapping
    public List<BatchEntry> getAll()
    {
        return service.getAll();
    }

    @GetMapping("/{id}")
    public Optional<BatchEntry> getById(@PathVariable String id)
    {
        return service.getById(id);
    }

    @PutMapping("/{id}")
```

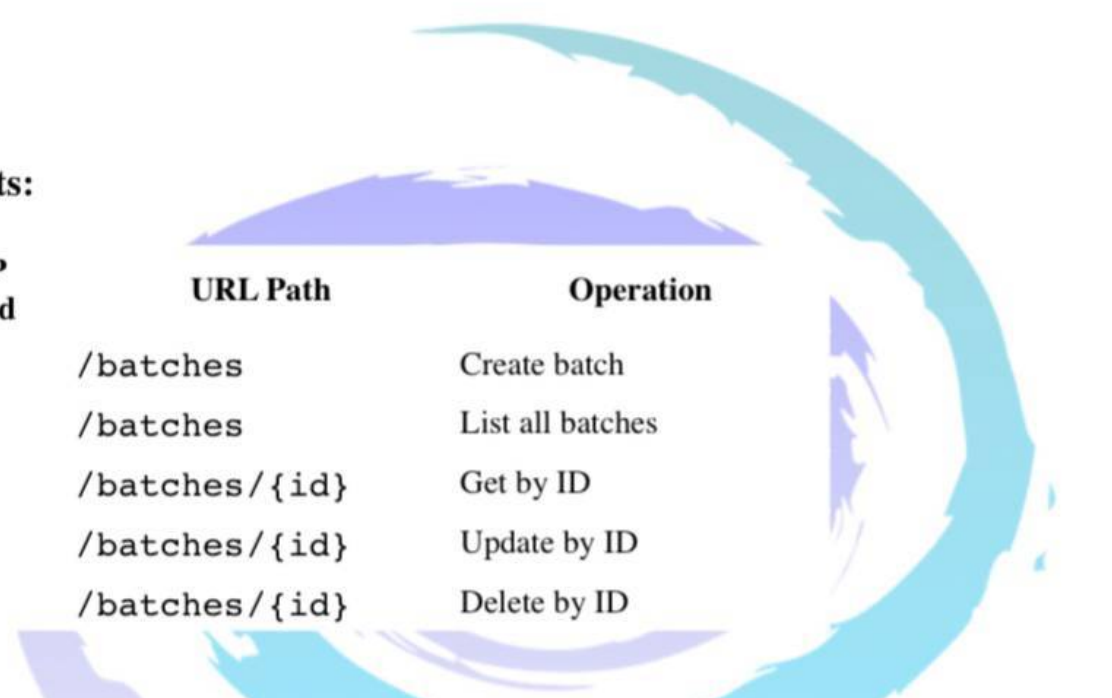
```

    public BatchEntry update(@PathVariable String id, @RequestBody
BatchEntry batch)
    {
        return service.update(id, batch);
    }

    @DeleteMapping("/{id}")
    public String delete(@PathVariable String id)
    {
        service.delete(id);
        return "Deleted batch with ID: " + id;
    }
}

```

### Endpoints:



HTTP Method	URL Path	Operation
POST	/batches	Create batch
GET	/batches	List all batches
GET	/batches/{id}	Get by ID
PUT	/batches/{id}	Update by ID
DELETE	/batches/{id}	Delete by ID

### Required Config in application.properties:

```

spring.data.mongodb.host=localhost
spring.data.mongodb.port=27017
spring.data.mongodb.database=MarvellousFullStack

```