ResponseEntity with HTTP Responses in Spring Boot

ResponseEntity :

ResponseEntity<T> is a Spring Boot class used to **represent the whole HTTP response**, including:

- **Body** – the actual data being returned (like a JSON object)
- **Status code** – like 200 OK, 201 CREATED, 404 NOT FOUND
- **Headers** – optional metadata sent along with the response

It gives **full control** over HTTP responses in REST APIs.

Use of ResponseEntity :

- To **return appropriate status codes** (not just data)
- To send **error or success messages clearly**
- To provide **custom headers**, if needed
- To handle various client-server interaction results

Common HTTP Status Codes Used in This Controller

Code	Meaning	Used When...
`200 OK`	Request successful	Fetch, Update, Get by ID
`201 CREATED`	New record created	POST method
`204 NO CONTENT`	Deleted successfully	DELETE method
`400 BAD REQUEST`	Invalid input or exception	try-catch in POST
`404 NOT FOUND`	Record not found	GET/PUT with invalid ID

Breakdown of Each Method with ResponseEntity

1. Get All Batch Entries

```
@GetMapping
public ResponseEntity<?> getAll()
{
   List<BatchEntry> alldata = batchEntryService.getAll();
   if (alldata != null && !alldata.isEmpty())
      {
      return new ResponseEntity<>(alldata, HttpStatus.OK); //200 OK
   }
else
{
      return new ResponseEntity<>(HttpStatus.NOT_FOUND);   // 404 NOT
FOUND
   }
}
```

Explanation:

- Returns list of batch entries with 200 status.
- If list is empty/null, returns 404.

2. Create a New Entry

```
@PostMapping
public ResponseEntity<BatchEntry> createEntry(@RequestBody BatchEntry
myentry)
{
   try
      {
      batchEntryService.saveEntry(myentry);
      return new ResponseEntity<>(myentry, HttpStatus.CREATED); //201
CREATED
   }
      catch (Exception e)
      {
      return new ResponseEntity<>(HttpStatus.BAD_REQUEST);      //400 BAD
REQUEST
   }
}
```

Explanation:

- On success: returns created object + 201.
- On failure: returns 400 (bad input or exception).

3. Get Entry by ID

```
@GetMapping("/id/{myid}")
public ResponseEntity<BatchEntry> getBatchEntryById(@PathVariable ObjectId
myid)
{
   Optional<BatchEntry> batchEntry = batchEntryService.findById(myid);
   if (batchEntry.isPresent())
{
      return new ResponseEntity<>(batchEntry.get(), HttpStatus.OK); // 200 OK
   }
else
{
      return new ResponseEntity<>(HttpStatus.NOT_FOUND);          // 404 NOT
FOUND
   }
}
```

Explanation:

- If entry found: return it with 200.
- If not: return 404.

4. Delete Entry by ID

```
@DeleteMapping("/id/{myid}")
public ResponseEntity<?> deleteEntryById(@PathVariable ObjectId myid) {
   batchEntryService.deleteById(myid);
   return new ResponseEntity<>(HttpStatus.NO_CONTENT); // ✅ 204 NO
CONTENT
}
```

Explanation:

- Deletes the entry and returns 204 (no body, just success).

5. Update Entry by ID

```
@PutMapping("/id/{myid}")
public ResponseEntity<?> updateEntryById(@PathVariable ObjectId myid,
@RequestBody BatchEntry myentry)
{
   BatchEntry old = batchEntryService.findById(myid).orElse(null);
   if (old != null)
{
      old.setName(myentry.getName());
      old.setFees(myentry.getFees());
      batchEntryService.saveEntry(old);
      return new ResponseEntity<>(old, HttpStatus.OK);          //200 OK
   }
else
{
      return new ResponseEntity<>(HttpStatus.NOT_FOUND);        //404 NOT
FOUND
   }
}
```

Explanation:

- If entry exists, updates it and returns 200.
- If not, returns 404.

Endpoint	HTTP Method	ResponseEntity Use	Status Codes
/batches	GET	Return list or 404	200, 404
/batches	POST	Create entry or show error	201, 400
/batches/id/{id}	GET	Get entry or return not found	200, 404
/batches/id/{id}	DELETE	Delete entry, no response body	204
/batches/id/{id}	PUT	Update entry or return not found	200, 404