Http Status

**1xx - Informational**

* **100 Continue:**
  + The server has received the request headers and the client should proceed to send the request body.
* **101 Switching Protocols:**
  + The server is switching protocols as requested by the client, usually in the Upgrade header.

**2xx - Success**

* **200 OK:**
  + The request was successful, and the server has returned the requested data.
* **201 Created:**
  + The request has been fulfilled, and a new resource has been created as a result.
* **204 No Content:**
  + The server successfully processed the request but there is no additional content to send in the response.

**3xx - Redirection**

* **301 Moved Permanently:**
  + The requested resource has been permanently moved to a new location, and future requests should use the new URL.
* **302 Found:**
  + The requested resource temporarily resides under a different URL. The client should continue to use the original URL.
* **304 Not Modified:**
  + Used in response to a conditional GET request to indicate that the resource has not been modified since the specified date.

**4xx - Client Errors**

* **400 Bad Request:**
  + The server could not understand the request, often due to malformed syntax or invalid parameters.
* **401 Unauthorized:**
  + The request requires user authentication. The client must provide valid credentials.
* **403 Forbidden:**
  + The server understood the request, but the client does not have permission to access the requested resource.
* **404 Not Found:**
  + The server could not find the requested resource.
* **405 Method Not Allowed:**
  + The method specified in the request is not allowed for the resource.

### 5xx - Server Errors

* **500 Internal Server Error:**
  + A generic error message indicating that an unexpected condition was encountered on the server.
* **501 Not Implemented:**
  + The server does not support the functionality required to fulfill the request.
* **502 Bad Gateway:**
  + The server, while acting as a gateway or proxy, received an invalid response from an upstream server.
* **503 Service Unavailable:**
  + The server is not ready to handle the request. Common causes include maintenance or temporary overloading of the server.

***ResponseEntity***

It is a class in the Spring Framework, particularly in Spring Web, that represents the entire HTTP response. It allows you to customize the HTTP response status code, headers, and body when building a response from your Spring MVC controller methods. It provides a more flexible and powerful way to handle responses compared to the simpler **return** statement.

Here are the key aspects of **ResponseEntity--🡪**

**1.Customization of HTTP Status Code:**

* You can set a specific HTTP status code for the response using the **HttpStatus** enum. This allows you to communicate the success or failure of the operation to the client.

ResponseEntity<String> responseEntity = new ResponseEntity<>("Success", HttpStatus.OK);

**2.Setting Headers:**

* You can add custom headers to the HTTP response. This is useful for providing additional metadata or instructions to the client.

HttpHeaders headers = new HttpHeaders();

headers.add("Custom-Header", "Value");

ResponseEntity<String> responseEntity = new ResponseEntity<>("Data", headers, HttpStatus.OK);

**3.Response Body:**

* You can include a response body along with the HTTP status and headers. The response body can be of any type, such as a String, JSON object, or custom DTO (Data Transfer Object)

ResponseEntity<MyObject> responseEntity = new ResponseEntity<>(myObject, HttpStatus.OK);

**4.Handling Error Responses:**

* **ResponseEntity** is particularly useful for handling error scenarios. You can customize the HTTP status code and provide additional details in the response body.

ResponseEntity<String> responseEntity = new ResponseEntity<>("Error message", HttpStatus.BAD\_REQUEST);

**5.Convenience Methods:**

* Spring provides convenience methods for common scenarios, such as **ok()**, **notFound()**, **badRequest()**, etc., which return **ResponseEntity** instances with predefined HTTP status codes.

ResponseEntity<String> responseEntity = ResponseEntity.ok("Success");

**6.Generics Support:**

* **ResponseEntity** is a generic class, allowing you to specify the type of the response body. This makes it type-safe and helps with automatic deserialization of the response body.

ResponseEntity<MyObject> responseEntity = new ResponseEntity<>(myObject, HttpStatus.OK);

**7.Integration with Spring MVC:**

* **ResponseEntity** is commonly used in Spring MVC controller methods to build and return HTTP responses. It integrates well with the overall Spring Web infrastructure.
* Here's a simple example of using **ResponseEntity** in a Spring MVC controller method:

@RestController

@RequestMapping("/api")

public class MyController {

@GetMapping("/example")

public ResponseEntity<String> getExample() {

// Some logic to determine the response data

String responseData = "Hello, World!"

// Build and return a ResponseEntity with the response data, status, and headers

return new ResponseEntity<>(responseData, HttpStatus.OK);

}

}

Here are some common HTTP status codes provided by the **HttpStatus** enum and their typical use cases:🡪

**1.HttpStatus.OK:**

* Represents the HTTP status code 200 (OK).
* Indicates that the request has succeeded.
* Often used for successful responses to GET or POST requests

ResponseEntity<String> responseEntity = ResponseEntity.ok("Success");

2.**HttpStatus.CREATED:**

* Represents the HTTP status code 201 (Created).
* Indicates that the request has been fulfilled and has resulted in one or more new resources being created.
* Typically used when creating a new resource using a POST request

ResponseEntity<String> responseEntity = ResponseEntity.status(HttpStatus.CREATED).body("Resource created");

3. **HttpStatus.NO\_CONTENT:**

* Represents the HTTP status code 204 (No Content).
* Indicates that the server successfully processed the request but there is no additional content to send in the response.
* Often used for successful DELETE requests.

ResponseEntity<Void> responseEntity = ResponseEntity.noContent().build();

4. **HttpStatus.BAD\_REQUEST:**

* Represents the HTTP status code 400 (Bad Request).
* Indicates that the server could not understand the request.
* Often used when the client sends invalid data or the request is malformed.

ResponseEntity<String> responseEntity = ResponseEntity.badRequest().body("Invalid request");

5. **HttpStatus.NOT\_FOUND:**

* Represents the HTTP status code 404 (Not Found).
* Indicates that the requested resource could not be found on the server.
* Typically used when a client requests a resource that does not exist.

ResponseEntity<String> responseEntity = ResponseEntity.notFound().build();

6. **HttpStatus.INTERNAL\_SERVER\_ERROR:**

* Represents the HTTP status code 500 (Internal Server Error).
* Indicates that an unexpected condition was encountered on the server.
* Used when an error occurs on the server that is not specifically handled by the application.

ResponseEntity<String> responseEntity = ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body("Internal Server Error");