**@Controller**

* @Controller is typically used for web applications that serve HTML content. It returns views (usually JSP, Thymeleaf, etc.) instead of direct response bodies.
* When you return a String from a method in a class annotated with @Controller, Spring assumes you're returning a view name, which the view resolver will try to resolve to a template.

**@RestController**

* @RestController is a specialized version of @Controller that automatically adds @ResponseBody to all methods. It is used for RESTful web services and API endpoints.
* When you return a String from a method in a class annotated with @RestController, Spring treats the String as the response body and returns it directly to the client.

**Example and Explanation:**

**Using @Controller:**

If you use @Controller and return a String, Spring will treat it as a view name by default. If there's no view resolver configured or the view is not found, you might get a 500 Internal

### \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### Uses of optional 2. ****Avoiding Null Checks****

By using Optional, you can avoid explicit null checks, which can reduce the risk of NullPointerException (NPE). Optional provides methods to handle the absence of values in a more elegant way, such as isPresent(), ifPresent(), orElse(), and orElseGet().

### 3. ****Functional Programming Style****

Optional encourages a functional programming style by providing methods that allow for chaining and more declarative code. This can make the code more concise and readable. For example:

public Optional<Article> getArticleById(int id) {

return articleRepository.findById(id);

}

// Handling the Optional value

Optional<Article> articleOptional = getArticleById(id);

articleOptional.ifPresent(article -> {

// Process the article if present

});  
  
  
Util classes contain all helper class code.

Target and retention annotations are used to make custom annotations.  
responsebody is used to convert the java object to json object while sending data from server to client because frontend can only understand json.  
handling exceptions in java is different from handling exceptions in springboot