@Validated : @Validated annotation is a class level that we can use to tell spring to validate parameters that are passed into a method of the annotated class

1. **@Operation Annotation:**
   * This annotation is part of the Springdoc OpenAPI library and is used to describe the operation or endpoint. It provides metadata about the API operation.
2. **@ApiResponses Annotation:**
   * This annotation is also part of the Springdoc OpenAPI library and is used to specify possible responses for the API operation. In this case, it defines two possible responses, one for a successful creation (HTTP status code 201) and another for a successful update (HTTP status code 200).
3. **@Content**

The annotation may be used to define the content/media type of a parameter, request or response, by definining it as field [Parameter.content()](https://docs.swagger.io/swagger-core/v2.0.0-RC3/apidocs/io/swagger/v3/oas/annotations/Parameter.html#content--), [RequestBody.content()](https://docs.swagger.io/swagger-core/v2.0.0-RC3/apidocs/io/swagger/v3/oas/annotations/parameters/RequestBody.html#content--) or [ApiResponse.content()](https://docs.swagger.io/swagger-core/v2.0.0-RC3/apidocs/io/swagger/v3/oas/annotations/responses/ApiResponse.html#content--).

1. **@Schema**

The *@Schema* annotation allows us to control Swagger-specific definitions such as description, name, type, example values, and allowed values for the model properties.

Also, it offers additional filtering properties in case we want to hide the property in certain scenarios.

Let’s add a few model properties to the *User’s* *firstName* field:

@Schema(

description = "first name of the user",

name = "firstName",

type = "string",

example = "Vatsal")

String firstName;

1. **@Valid**

@Valid annotation ensures the validation of the whole object. Importantly, it performs the validation of the whole object graph.

1. **@RequestHeader**

You can use the RequestHeader annotation to bind a request header to a method argument in a controller. It binds header values to a method parameters.

1. **@NotNull**

It means that the field must be not null.

1. **@NotBlank**

It checks that the annotated character sequence/String is not null and the trimmed length is greater than zero.

1. **Final HttpServletRequest :**

let's break down the parameters **final HttpServletRequest httpRequest** and **@NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER)** in your code snippet:

1. **final HttpServletRequest httpRequest**:
   * This parameter is of type **HttpServletRequest**, which is a class in Java's Servlet API. It represents the HTTP request that was made to the server.
   * By including this parameter in the method signature, you are providing access to various details about the incoming HTTP request, such as headers, query parameters, request method, and more.
   * It allows you to inspect and interact with the incoming request, which can be useful for tasks like accessing request headers, extracting query parameters, or examining the request body.

**@NotNull @RequestHeader(BOSConstants.CORRELATION\_ID\_HEADER)**:

1. **@RequestHeader** is an annotation in Spring Framework used to extract values from HTTP request headers.
2. **BOSConstants.CORRELATION\_ID\_HEADER** likely refers to a constant string that represents the name of an HTTP header, such as "Correlation-Id."
3. **@NotNull** is an additional annotation used to indicate that the header value must not be null. In other words, it enforces that a value for this header must be present in the HTTP request, and it cannot be missing.
4. The combined effect of these annotations is that Spring will extract the value of the "Correlation-Id" header from the incoming HTTP request and assign it to the **correlationId** parameter. If the header is missing or has a null value, it will likely result in a validation error or exception (depending on your application's error handling configuration).