SPRING MVC VALIDATION

Java cuenta con un mecanismo que permite indicar reglas o restricciones que deben tener los valores que se le pueden dar a los atributos de un objeto, el JSR 349 (Bean Validation). Este, como todos los JSR, es una especificación que indica cómo deben comportarse las validaciones pero no proporciona una implementación de la misma. Es por eso que debemos elegir una implementación como Hibernate Validator.

El soporte de Bean Validation de Spring Boot viene con el starter de validación, que podemos incluir en nuestro proyecto:

implementation('org.springframework.boot:spring-boot-starter-validation')

Básicamente, Bean Validation funciona definiendo restricciones a los campos de una clase anotándolos con ciertas anotaciones. Las comunes son:

* @NotNull: para decir que un campo no debe ser nulo.
* @NotEmpty: para decir que un campo de lista no debe estar vacío.
* @NotBlank: para decir que un campo de cadena no debe ser la cadena vacía (es decir, debe tener al menos un carácter).
* @Min and @Max: para decir que un campo numérico sólo es válido cuando su valor está por encima o por debajo de un determinado valor.
* @Pattern: para decir que un campo de cadena sólo es válido cuando coincide con una determinada expresión regular.
* @Email: para decir que un campo de cadena debe ser una dirección de correo electrónico válida.

La Spring MVC Validation se utiliza para restringir la entrada proporcionada por el usuario. Para validar la entrada del usuario, la versión Spring 4 o superior admite y utiliza la API de validación Bean. Puede validar tanto las aplicaciones del lado del servidor como las del lado del cliente.

La API de Bean Validation es una especificación de Java que se utiliza para aplicar restricciones en el modelo de objetos a través de anotaciones. Aquí podemos validar una longitud, un número, una expresión regular, etc. Aparte de eso, también podemos proporcionar validaciones personalizadas.

Como la API de Bean Validation es sólo una especificación, requiere una implementación. Por lo tanto, para ello, utiliza Hibernate Validator. El validador de Hibernate es una implementación totalmente compatible con JSR-303/309 que permite expresar y validar las restricciones de la aplicación.

Spring boot aprovecha esta funcionalidad y proporciona la anotación “@ExceptionHandler” que permite indicar que un método especial debe ser invocado al momento de ocurrir un error. Con esto podemos indicar qué tratamiento debe darle la aplicación a este tipo de excepciones. También proporciona una excepción especial, “ResponseStatusException”, con la cual podemos regresar un estatus particular y un mensaje específico dentro de nuestro código.

La forma más rápida de resolver el problema con el estado es lanzar la ResponseStatusException proporcionada por Spring que acepta un HttpStatus. Podemos configurar un @ControllerAdvice que puede lanzar métodos para manejar ResponseStatusException. Este método tiene que ser anotado por @ExceptionHandler que especifica qué tipo de excepciones puede manejar un método.

De forma similar, si queremos darle el mismo tratamiento a una excepción sin importar en qué parte de la aplicación ocurra podemos usar la anotación “@ControllerAdvice”.

@ControllerAdvice es una especialización de la anotación @Component que permite manejar excepciones en toda la aplicación en un componente de manejo global. Puede ser visto como un interceptor de excepciones lanzadas por métodos anotados con @RequestMapping y similares.

Declara métodos @ExceptionHandler, @InitBinder, o @ModelAttribute para ser compartidos a través de múltiples clases @Controller.

ResponseEntityExceptionHandler es una clase base conveniente para las clases @ControllerAdvice que desean proporcionar un manejo de excepciones centralizado en todos los métodos @RequestMapping a través de métodos @ExceptionHandler. Proporciona un método para manejar las excepciones internas de Spring MVC. Devuelve una ResponseEntity en contraste con DefaultHandlerExceptionResolver que devuelve un ModelAndView.

Con estos mecanismos podemos ofrecer un correcto manejo de los errores dentro de las APIs o servicios web que estemos creando.