Spring Boot Main annotations

--------------------------------------------------

1. @SpringBootApplication //=@EnableAutoConfiguration + @ComponentScan + @Configuration

2. @ComponentScan –scan for bean in parent and its child package of springbootapplication class

3. @EanbleAutoConfiguration – enables Spring Boot to auto-configure the application context. Therefore, it automatically creates and registers beans based on both the included jar files in the classpath and the beans defined by us.

4. @Configuration –with the help of @Bean , it create bean and pass on where bean is required.

e.g if we want to use RestTemplate – we need to define bean for it in some config class

Stereotype annotation

----------------------------------------

1. @Component

2. @Service

3. @RestController / @Controller

4. @Repository

Spring Core related Annotations:

----------------------------------------------------

@Configuration

@Bean

@Autowired

@Qualifier --

AnnotationServiceImplV1/ AnnotationServiceImplV2 implements AnnotationService

And we autowire AnnotationService and hence spring find two bean :-

Field annotationService in com.springbootannotations.controller.AnotationsController required a single bean, but 2 were found:

- annotationServiceImplV1: defined in file [D:\My Works\Eclipse-workspace\springboot-annotations\springboot-annotations\target\classes\com\springbootannotations\service\AnnotationServiceImplV1.class]

- annotationServiceImplV2: defined in file [D:\My Works\Eclipse-workspace\springboot-annotations\springboot-annotations\target\classes\com\springbootannotations\service\AnnotationServiceImplV2.class]

Action:

Consider marking one of the beans as @Primary, updating the consumer to accept multiple beans, or using @Qualifier to identify the bean that should be consumed

@Lazy 🡪> to avoid creation of bean until we autowire it or use it .

@Value – to fetch value from application.properties @Value(${key})

@PropertySource

@ConfigurationProperties

@Profile

@Scope -> default is singleton

Singleton – even we hit url multiple means calls controller (bean), bean will be created once.

Prototype – every times we will hit url meaning controller class will be called, bean will be created.

REST API related Annotations:

------------------------------------------------

@RestController

@RequestMapping

@GetMapping

@PostMapping

@PutMapping

@DeleteMapping

@RequestBody

@PathVariable

@RequestParam

@ControllerAdvice & @ExceptionHandler

Once the controller method throws exception, the request went to @RestControllerAdvice/ControllerAdvice class, if they found the matched exception, it will show customized message tto user.

Ex..

@RequestMapping(value = "/test1", method = RequestMethod.***GET***) throws AnnotationNotFoundException

**public** String method1() {

System.***out***

.println("from :" + from + ", " + "to :" + to + ", " + "port : " + port + "," + "message : " + message);

**return** "pass1";

}

|  |
| --- |
| public class AnnotationNotFoundException extends Exception{ |
|  |  |
|  | public AnnotationNotFoundException(String message) { |
|  | super(message); |
|  | } |
|  | } |

**import** java.util.UUID;

@RestControllerAdvice

**public** **class** AnnotationExceptionHandler {

@ExceptionHandler(AnnotationnsNotFoundException.**class**)

**public** ResponseEntity<AppError> handleException(AnnotationnsNotFoundException exception) {

AppError error = **new** AppError(UUID.*randomUUID*().toString(), exception.getMessage(),

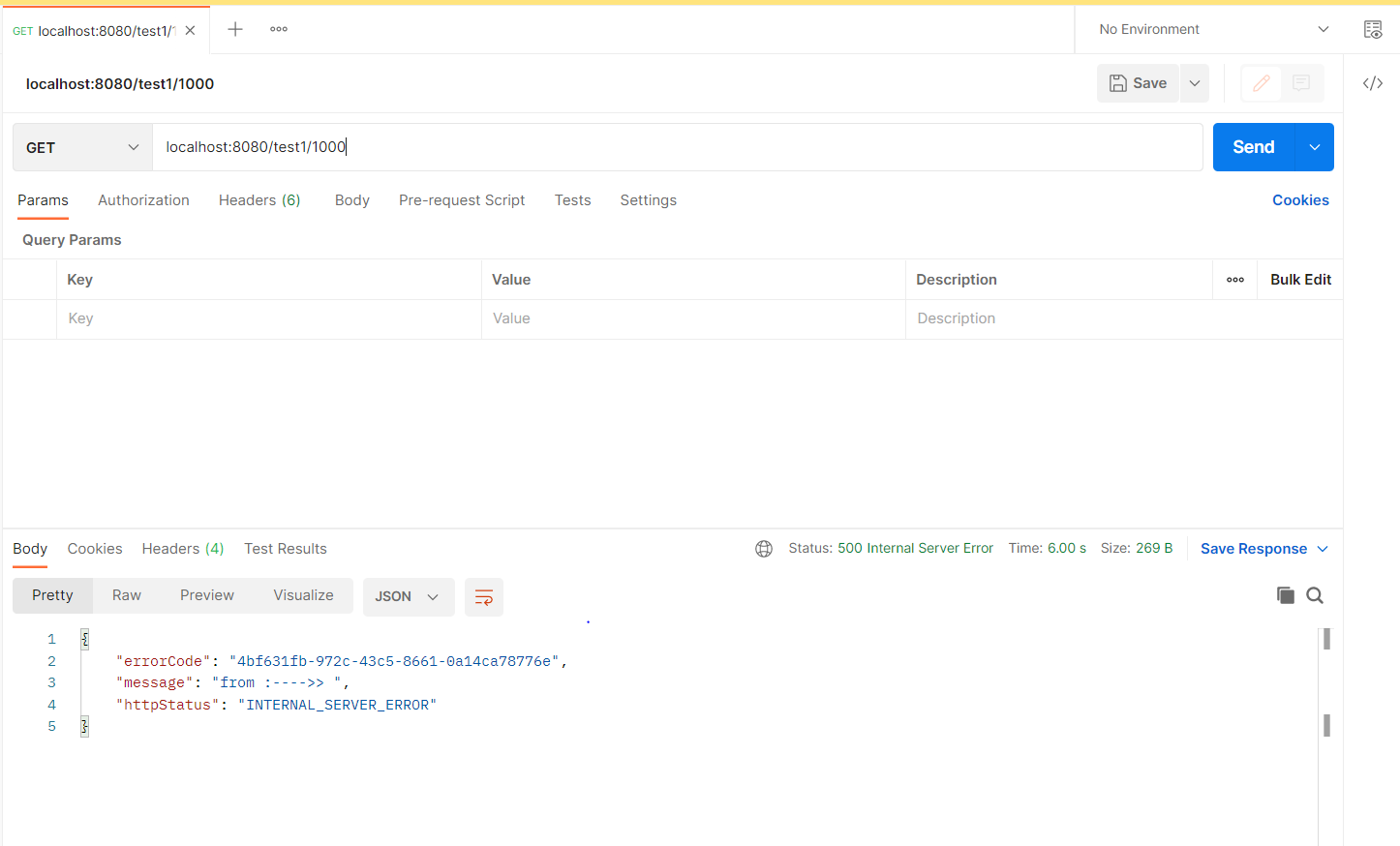
HttpStatus.***INTERNAL\_SERVER\_ERROR***);

**return** **new** ResponseEntity<>(error, HttpStatus.***INTERNAL\_SERVER\_ERROR***);

}

}

Customized o/p if exception throw using @ControllerAdvice & @ExceptionHandler



Spring Data JPA related annotations:

------------------------------------------------------------

@Entity

@Table

@Column

@Transactional

Entity class relationships

@OnetoOne

@OnetoMany

@ManytoOne

@ManytoMany