@techwithvishalraj

Java

SpringBoot Annotation

Follow

Share

@techwithvishalraj

Annotation on Main Class

@SpringBoot Application	import org.springframework .boot.autoconfigure .SpringBootApplication;	Indicate the main configuration class of a Spring Boot application. It combines (@Configuration + @EnableAutoConfiguration + @ComponentScan)
@Configuration	import org.springframework .context.annotation .Configuration;	It is a class-level annotation. The class annotated with @Configuration used by Spring Containers as a source of bean definitions.
@EnableAuto Configuration	import org.springframework .boot.autoconfigure .EnableAutoConfiguration;	It auto-configures the bean that is present in the classpath and configures it to run the methods.
@Component Scan	import org.springframework .context.annotation .ComponentScan;	It is used when we want to scan a package for beans. It is used with the annotation @Configuration. We can also specify the base packages to scan for Spring Components. @ComponentScan(basePackages = "com.java")

Annotation on Bean

@Bean	import org.springframework .context.annotation.Bean;	It is a method-level annotation. It is an alternative of XML <bean> tag. It tells the method to produce a bean to be managed by Spring Container.

@techwithvishalraj

Annotation on Controller Class

@RestController	import org.springframework.web.bind.annotation.RestController;	It is a specific type of Spring MVC controller that is used to build RESTful web services it combines (@Controller + @ResponseBody)
@Validated	import org.springframework.validation.annotation.Validated;	Validation is commonly used to check user input, validate data from external sources
@RequestMapping(value= "/uriPath")	import org.springframework.web.bind.annotation.RequestMapping;	It is used to map the web requests. It has many optional elements like consumes, header, method, name, params, path, produces, and value. We use it with the class as well as the method.

Annotation on Object

@Autowired	import org.springframework.beans.factory.annotation.Autowired;	It is used to autowire spring bean on setter methods, instance variable, and constructor. When we use @Autowired annotation, the spring container auto-wires the bean by matching data-type.

@techwithvishalraj

CRUD Operation Annotation

@PostMapping (value= "/uripath”)	import org.springframework. web.bind.annotation. GetMapping;	It maps the HTTP POST requests on the specific handler method. It is used to create a web service endpoint that creates resource. It is used instead of using: @RequestMapping(method = RequestMethod.POST)
@GetMapping (value="/uriPath”)	import org.springframework. web.bind.annotation. GetMapping;	It maps the HTTP GET requests on the specific handler method. It is used to create a web service endpoint that fetches resource. It is used instead of using: @RequestMapping(method = RequestMethod.GET)
@PutMapping (value="/uripath”)	import org.springframework. web.bind.annotation. PutMapping;	It maps the HTTP PUT requests on the specific handler method. It is used to create a web service endpoint that creates or updates resource. It is used instead of using: @RequestMapping(method = RequestMethod.PUT)
@DeleteMapping (value="/uripath”)	import org.springframework. web.bind.annotation. DeleteMapping;	It maps the HTTP DELETE requests on the specific handler method. It is used to create a web service endpoint that deletes a resource. It is used instead of using: @RequestMapping(method = RequestMethod.DELETE)

@techwithvishalraj

Annotation on REST request

@RequestBody	import org.springframework.web .bind.annotation.RequestBody;	It is used to bind HTTP request with an object in a method parameter.
@PathVariable ("email")	import org.springframework.web.bind. annotation.PathVariable;	used to extract values from the URI path and use them as method parameters
@RequestParam (”email”)	import org.springframework.web.bind .annotation.RequestParam;	Annotation which indicates that a method parameter should be bound to a webrequest parameter.
@Pattern(regexp = "[a-z]", message = "abc”) @NotNull, @Min, @Max	import javax.validation .constraints.*;	These annotations are used to validate incoming data before processing it in your application.
@Valid	import jakarta.validation.Valid;	If you have a REST endpoint then use @Valid with a DTO class as a method parameter.

Annotation on DTO

@NotNull, @Email @Size(min = 2, max = 50) @Pattern(regexp = "[a-z]", message = "abc”)	import javax.validation.constraints.*;	apply validation constraints to fields.

Annotation on Entity/Model

@GeneratedValue (strategy = GenerationType .IDENTITY)	import javax.persistence .GenerationType;	Configures how the primary key is generated. Common strategies are GenerationType.IDENTITY, GenerationType.SEQUENCE, and GenerationType.AUTO.
@Entity	import javax.persistence.Entity;	Marks a class as a JPA entity, representing a table in the database.
@Table	import javax.persistence.Table;	Specifies the name of the database table to which the entity is mapped. You can also configure other table properties like indexes and unique constraints using this annotation.
@Id	import javax.persistence.Id;	Specifies the primary key of the entity.
@Column(name= "abc")	import javax.persistence.Column;	Specifies the mapping of a field to a database column, allowing you to configure properties like column name, length, and nullable constraints.
@OneToMany @OneToOne	import javax.persistence.*;	Define relationships between entities.
@JoinColumn (name="abc")	import javax.persistence .JoinColumn;	to be used foreign/reference key
@Transient	import javax.persistence.Transien;	Marks a field as not persistent, meaning it won't be mapped to a database column.

@techwithvishalraj

Stereotype Annotation

@Component	import org.springframework .stereotype.Component;	This annotation is a generic stereotype for any Spring-managed component. It tells Spring to scan the classpath for classes annotated with @Component and create beans from them.
@Controller	import org.springframework .stereotype.Controller;	This annotation is used to mark a class as a Spring MVC controller. It is typically used in web applications to handle HTTP requests.
@Service(value= "xyzXyz")	import org.springframework .stereotype.Service;	It is used at class level. It tells the Spring that class contains the business logic.
@Repository	import org.springframework .stereotype.Repository;	It is a class-level annotation. The repository is a DAOs (Data Access Object) that access the database directly. The repository does all the operations related to the database.

Transactional Annotation

@Transactional	import org.springframework.transacti on.annotation.Transactional;	If an exception is thrown within the method, the transaction can be rolled back, ensuring data consistency. You can annotate a specific method or class.

@techwithvishalraj

Exception Annotation

@RestControllerAdvice	import org.springframework .web.bind.annotation .RestControllerAdvice;	This annotation is used to define a global exception handler that can be applied to multiple controllers. You can create a class annotated with @RestControllerAdvice and define methods within it annotated with @ExceptionHandler to handle specific exceptions.
@ExceptionHandler (Exception.class) @ExceptionHandler(Inf yInternException.class)	import org.springframework.we b.bind.annotation .ExceptionHandler;	You can use the @ExceptionHandler annotation on methods within your controller classes to handle specific exceptions for that controller. This allows you to define exception handling logic on a per-controller basis.
@ExceptionHandler({ MethodArgumentNot ValidException.class, ConstraintViolation Exception.class})	import org.springframework.we b.bind.MethodArgument NotValidException; import javax.validation.Constrai ntViolationException;	MethodArgumentNotValidException exception is typically thrown when validating the request parameters, request body, or method arguments of a controller endpoint using annotation-based validation (e.g., @Valid and validation annotations like @NotNull, @Size, etc.). ConstraintViolationException exception is more closely associated with Bean Validation and is thrown when there are validation constraints violated on JavaBeans. These constraints are typically defined using annotations like @NotNull, @Size, @Email, etc.
@ResponseStatus(Http Status.NOT_FOUND)	import org.springframework .web.bind.annotation .ResponseStatus;	This annotation can be used to specify the HTTP status code to return when a specific exception occurs. You can use it in combination with @ExceptionHandler.

@techwithvishalraj

Aspects Annotation

@Aspect	import org.aspectj.lang .annotation.Aspect;	Aspects in Spring Boot are commonly used for tasks like logging, security, transaction management, and performance monitoring. They allow you to keep these cross-cutting concerns separate from your core business logic, making your codebase more modular and maintainable.
@AfterThrowing @Before, @After, @Around	import org.aspectj.lang .annotation.*;	You define pointcut expressions using annotations like @Before, @After, @Around, etc. These expressions determine where in your codebase the advice methods should be applied.

Test Annotation

@SpringBootTest @SpringBootTest(webEnvironment = WebEnvironment. MOCK)	import org.springframework.boot .test.context .SpringBootTest;	It loads the entire Spring application context, making it suitable for integration tests. You can use it with the webEnvironment attribute to specify the type of web environment(WebEnvironment.MOCK for a mock servlet environment).
@Mock	import org.mockito.Mock;	It's often used to replace real dependencies with mock objects to isolate the class under test.
@InjectMocks	import org.mockito.InjectMocks;	it is about the automatic injection of those mock objects into the class or object you are testing.
@Test	import org.junit.jupiter.api.Test;	This is a standard JUnit annotation used to mark a method as a test method.

Thank you!

EXPLORE MORE

@techwithvishalraj