Annotations with import packages

	Class type	Annotations	Imports
1.	main	**@SpringBootApplication** //above the main class	import org.springframework.boot.autoconfigure.SpringBootApplication;
2.	REST API	**@RestController** //above the REST API class	import org.springframework.web.bind.annotation.RestController;
		@Validated //above the REST API class	import org.springframework.validation.annotation.Validated;
		@RequestMapping(value= "/uriPath") //above the REST API class	import org.springframework.web.bind.annotation.RequestMapping;
		@Autowired //to be used when 'inject' keyword is given	import org.springframework.beans.factory.annotation.Autowired;

		@GetMapping(value="/uriPath")
		//for Get Request
		
		@PostMapping(value= "/uripath")"
		//for Create Request
		
		@PutMapping(value="/uripath")
		//for Update Request
		
		@DeleteMapping(value="/uripath")
		//for Delete Request
		@RequestBody
		//while creating or updating for providing JSON body
			import org.springframework.web.bind.annotation.GetMapping;
			import org.springframework.web.bind.annotation.PostMapping;
			import org.springframework.web.bind.annotation.PutMapping;
			import org.springframework.web.bind.annotation.DeleteMapping;
			
			import org.springframework.web.bind.annotation.RequestBody;

3.	DTO	**@NotNull** **@Min** **@Max**	import javax.validation.constraints.NotNull; import javax.validation.constraints.Min; import javax.validation.constraints.Max;

4.	Entity	`@Entity` //above Entity classes	`import javax.persistence.Entity;`
		`@Id` //above primary key of that particular Entity class	`import javax.persistence.Id;`
		`@GeneratedValue(strategy=GenerationType.IDENTITY)` //usedfor auto generation	`import javax.persistence.GenerationType;`
		`@Column``(name="xxx_xx")` //to be used when asked to map table	`import javax.persistence.Column;`
		`@JoinColumn``(name="xxx_xx")` //to be used above foreign/reference key	`import javax.persistence.JoinColumn;`

		`@ManyToOne(cascade= CascadeType.ALL)`
		
		`@OneToOne``(cascade= CascadeType.ALL)`
		//any one will be used according to the qstn given
			`import javax.persistence.ManyToOne;`
			
			`import javax.persistence.OneToOne;`

5.	Repository	`extends CrudRepository< x , y>`
		
		`//where x is Entity class name & y is the Wrapper Class datatype of pk of x (y can be Integer)`
			`import org.springframework.data.repository.CrudRepository;`

		`List<EntityName>`	`import java.util.List;`
6.	Service	**`@Service`**`(value="xyzXyz")` //specific stereotype annotation to be used above ServiceImpl class **//@Controller @Repository** for Controller and Repository respectively class if asked in qp	`import org.springframework.stereotype.Service;`
		`@Transactional` //to be used whenever 'manages transactions' is mentioned	`import org.springframework.transaction.annotation.Transactional;`
		`@Autowired` //to be used when 'inject' keyword is given	`import org.springframework.beans.factory.annotation.Autowired;`
		`Optional< EntityName > optional` //to be used when findById() is invoked	`import java.util.Optional;`
		`@RestControllerAdvice` //to be used above class	`import org.springframework.web.bind.annotation.RestControllerAdvice;`

7.	ExceptionHandler	**@ExceptionHandler**(Exception.**class**) //super class of all exceptions	**import** org.springframework.web.bind.annotation.ExceptionHandler;

		@ExceptionHandler(InfyInternException.**class**)
		//projectName specific exceptions
		//in place of InfyInternException the Exception name given in QP will be used	

		@ExceptionHandler({MethodArgumentNotValidException.**class** , ConstraintViolationException.**class**}) //to be used when method has if else condition inside it and handles two exceptions	**import** org.springframework.web.bind.MethodArgumentNotValidException; **import** javax.validation.ConstraintViolationException;
8.	*For logging*	LOGGER declaration: both should be used for logging details **private static final** Log ***LOGGER*** = LogFactory.*getLog*(ExceptionControllerAdvice.**class**); ***LOGGER***.error(exception.getMessage(), exception);	**import** org.apache.commons.logging.LogFactory; **import** org.apache.commons.logging.Log;
		HTTP_STATUS **HttpStatus** //eg., HttpStatus.OK, HttpStatus.CREATED	**import** org.springframework.http.HttpStatus;
9.	LoggingAspect	**@Aspect** //above logging Aspect class	**import** org.aspectj.lang.annotation.Aspect;

		@Component
		//generic stereotype annotation
		//above logging Aspect class
			import org.springframework.stereotype.Component;

		@AfterThrowing //above method(s) of Aspect class with pointcut declaration	**import** org.aspectj.lang.annotation.AfterThrowing;
		@SpringBootTest //above Test class	**import** org.springframework.boot.test.context.SpringBootTest;

10.	Test	**@Mock** //above object/class ref. declaration which will be mocked	`import org.mockito.Mock;`
		@InjectMocks //above object/class ref. declaration which will mock	`import org.mockito.InjectMocks;`
		@Test //above Test methods	`import org.junit.jupiter.api.Test;`
		Assertions *assertThrows/ assertEquals*	`import org.junit.jupiter.api.Assertions;`