



Thank you. Your test is submitted successfully.

You have cleared this assessment.

Obtained Percentage

Obtained Marks

75 %

15 / 20

Best Attempt Score: 75 % on 15-05-2024

[Review Your Attempt](#)

Result: CLEARED

Overall Percentage: 75% (15/20)

Spring Rest Self Assessment

Section Score: 15/20

 Correct  Incorrect  Unattempted

What should be inserted in Line 1, Line 2 to differentiate given controller methods using Request Parameter versioning

```
//Line1  
public PlanDTO fetchPlanById(@PathVariable("planId") int planId)  
//Line 2  
public String fetchPlanById2(@PathVariable("planId") int planId)
```

- ☐ Line1 - @GetMapping(value = "{planId}/VER=1")
- ☐ Line2 - @GetMapping(value = "{planId}/VER=2")
- ☐ Line1 - @GetMapping(value = "{planId}?ver=1")
- ☐ Line2 - @GetMapping(value = "{planId}?ver=2")
- ☐ Line1 - @GetMapping(value = "{planId}", params = "VER=1")
- ☒ Line2 - @GetMapping(value = "{planId}", params = "VER=2")
- ☐ Line1 - @GetMapping(value = "{planId}") @param(ver=1)
- ☐ Line2 - @GetMapping(value = "{planId}") @param(ver=2)

Q2 of 20

Which of the below configurations are necessary to enable basic authentication with default username and password? (Choose any 2 options)

- ☐ Add Security starter dependency in POM.XML.
- ☐ Set security related properties in application.properties.
- ☒ Extend WebSecurityConfigurerAdapter class and override required methods.
- ☒ Add web starter dependency in POM.XML

Q3 of 20

Which class in Spring security is used to hash passwords?

- ☒ PasswordEncoder
- ☐ PasswordHasher
- ☐ HashEncoder
- ☐ SecureEncoder

Have a look at the following controller. (Assume all necessary imports are done.)

```
@Controller
public class EmployeeController {
    @Autowired
    EmployeeService empService;

    @PostMapping(value = "/update/{phoneNum}")
    public String updateEmployeeAddress(@PathVariable("phoneNum") long phoneNumber, EmployeeDTO employeeDTO) {
        String resp = empService.updateEmployee(phoneNumber, employeeDTO);
        return resp;
    }
}
```

On fixing which of the below issue the above controller works properly.

- ☐ Should use @PutMapping annotation instead of @PostMapping over updateEmployeeAddress method and @RestController, @RequestMapping annotations are missing over EmployeeController class.
- ☒ @ResponseBody annotation is missing over EmployeeController class and @RequestBody annotation should precede employeeDTO object
- ☐ @ResponseBody, @RequestMapping annotations are missing over EmployeeController class.
- ☐ Should use @PutMapping annotation instead of @PostMapping over updateEmployeeAddress method and @RequestBody annotation should precede employeeDTO object.

Q5 of 20

Which of the below are ways of realizing and implementing SOA?

- ☒ WebServices
- ☐ Lisp
- ☐ Perl
- ☒ Jini

Have a look at the following controller method. (Assume that the REST resources corresponding to employee and address are available in the same application.)

```
@PostMapping()  
public String createEmployee(@RequestBody EmployeeDTO empdto) {  
    RestTemplate restTemplate = new RestTemplate();  
    String url = "http://localhost:8080/address/{phoneno}";  
    Long Phoneno=9897969543L;  
    //Line 1  
    //code to insert employee in table  
}
```

What should be inserted at Line 1 to invoke another controller method with URL `http://localhost:8080/address/{phoneno}` to get the latest address details which are used in adding employee data to the table.

- ☐ Address response = restTemplate.exchange(url, HttpMethod.Get, phoneno);
- ☒ ResponseEntity<Address> response = restTemplate.exchange(url, HttpMethod.Get, null, Address.class, phoneno);
- ☐ Address response = restTemplate.getForEntity(url, phoneno);
- ☐ ResponseEntity<Address> response = restTemplate.getObject(url, Address.class, phoneno);

Observe the following ExceptionHandler class.

```
@RestControllerAdvice
public class ExceptionControllerAdvice {
    @ExceptionHandler(Exception.class)
    public String exceptionHandler(Exception ex) {
        return "exceptionHandler handled the exception";
    }
    @ExceptionHandler(NullPointerException.class)
    public String exceptionHandler1(ClassCastException ex) {
        return "exceptionHandler1 handled the exception";
    }
}
```

What will be the output when the below 2 exceptions are thrown from controller class.

IllegalArgumentException

NullPointerException

Choose the correct option.

- ☐ IllegalArgumentException - exceptionHandler handled the exception
- ☐ NullPointerException - exceptionHandler handled the exception
- ☐ IllegalArgumentException -- application fails
- ☐ NullPointerException - exceptionHandler1 handled the exception
- ☐ IllegalArgumentException - application fails
- ☒ NullPointerException - application fails
- ☐ IllegalArgumentException - exceptionHandler handled the exception
- ☐ NullPointerException - application fails

Observe the following controller. (Assume necessary imports are done)

```
@RestController
@RequestMapping("/customer")
public class CustomerController {

    // Line 1
    public String insertCustDetails (...) {
        //code to insert customer details in table.
    }
}
```

What should be inserted at line1 and in parameter section of insertCustDetails method for the below URL to invoke insertCustDetails method successfully.

<http://localhost:8080/customers/ravi/infosys/custdet?id=6785&phno=8889997865>

- ☐

```
@PostMapping("/{custName}/{organization}/custdet")
public String insertCustDetails (@PathVariable() String custName, @PathVariable() String organization, @RequestParam("id") int custId, @RequestParam("phno")
```
- ☐

```
@PostMapping("/{name}/{org}/custdet")
public String insertCustDetails (@PathVariable("name") String custName, @PathVariable("org") String organization, @RequestParam() int Id, @RequestParam() l
```
- ☐

```
@PostMapping("/{customerName}/{organization}/custdet/")
public String insertCustDetails (@PathVariable("Name") String custName, @PathVariable("Org") String organization, @RequestParam() long Phno, @RequestParam
```
- ☒

```
@PostMapping("/{custName}/{organization}/custdet")
public String insertCustDetails(@PathVariable String custName, @PathVariable String organization, @RequestParam int custId, @RequestParam long phoneNo)
```

Q9 of 20

Which of the below properties should be configured to set port number and path for the application? (Choose any 2 options)

- ☒ server.port
- ☐ server.servlet.context-path
- ☐ server.address
- ☒ server.servlet.context-path

Given:

```
public class CustomerDTO {  
    String name;  
    Long phoneNumber;  
    String email;  
    Integer age;  
}
```

Anmol wants to perform validations on the incoming customer object from the request body of a URL. He wants the name of the customer to be not null, should have some phone number, his mail Id should be in valid format and his age should be between 25 to 60. Which of the below options is specifying correct annotation for the attributes of CustomerDTO class to support validations.

name - @NotNull

phoneNumber - @NotBlank

☐ email - @Pattern(regex = "[a-zA-Z]*@[a-z].com))

age - @Size(25, 60)

name - @NotNull

phoneNumber - @NotBlank

☒ email - @Email

age - @Size(min=25, max=60)

name - @NotBlank

phoneNumber - @NotEmpty

☐ email - @Email

age - @Limit(min=25, max=60)

name - @NotNull

phoneNumber - @NotEmpty

☐ email - @Pattern(regex = "[a-zA-Z]*@[a-z].com")

age - @Limit(25,60)

Q11 of 20

What is the correct way to annotate the controller class to make it accept GET, POST requests coming from other origins like "http://localhost:7000/" "http://localhost:8687/" ?

- ☒ `@CrossOrigin(origins = {"http://localhost:7000", "http://localhost:8687"}, methods= {RequestMethod.GET, RequestMethod.POST})`
- ☐ `@CrossOrigins(allowedOrigins = {"http://localhost:7000", "http://localhost:8687"}, allowedMethods= {RequestMethod.GET, RequestMethod.POST})`
- ☐ `@EnableCrossOrigins(origins = {"http://localhost:7000", "http://localhost:8687"}, allowedMethods= {RequestMethod.GET, RequestMethod.POST})`
- ☐ `@EnableCrossOrigin(allowedOrigins = {"http://localhost:7000", "http://localhost:8687"}, Methods= {RequestMethod.GET, RequestMethod.POST})`

Q12 of 20

Which of the below are not valid annotations to use in documenting Rest API's using swagger? (Choose any 2)

- ☒ `@ApiMethod(value = " ")`
- ☐ `@ApiResponses`
- ☐ `@ApiResponse`
- ☒ `@ApiModels(description=" ")`

Q13 of 20

Consider the following controller:

```
@RestController
@Valid
@RequestMapping("accounts")
public class AccountController {
    @Autowired
    private AccountService acctService;

    @GetMapping("/account")
    public List<AccountDTO> getAccountDetails(@Max(value=10000000, message="the value is out of range") @NotEmpty(message="account number should no
        return acctService.fetchAccountDetails(acctNo, ifscCode);
    }
}
```

What is wrong with above controller?

- ☐ @Valid is a parameter level annotation. It should be placed before acctno Request parameter.
- ☒ @Validated annotation should present over controller rather than @Valid annotation.
- ☐ There is no wrong with this controller.
- ☐ Should not apply two validations on acctno Request parameter.

What is the output when the below controller method is invoked from the postman?. (Assume necessary imports are done.)

```
@RestController
@RequestMapping("/student")
public class StudentController {
    @Autowired
    StudentService stdservice;

    @GetMapping(value = "/{rollno}/fetch")
    public StudentDTO getStudentDetails(@PathVariable() long Rollno) {
        StudentDTO stdTO = stdservice.fetchStudent(Rollno);
        return stdTO;
    }
}
```

URL used to invoke the controller method.

<http://localhost:8080/college/student/8870/fetch>

- ☐ Exception is thrown as the path variable annotation is not given any input parameters.
- ☐ The controller method executes successfully and returns a studentDTO object.
- ☒ Exception is thrown as the path variable in URI and in "getStudentDetails" method parameter are not same.
- ☐ The controller method executes successfully and returns null.

Q15 of 20

Jhon wants to call a controller method in application A which inserts a record in Book table from another controller method in application B. Which of the below RestTemplate methods help jhon in fulfilling his requirement?

- ☐ getForObject(....)
- ☒ postForObject(....)
- ☐ getForMethod(....)
- ☐ postForMethod(....)

Q16 of 20

Which of the following statements are true regarding Matrix variables? (Choose any 2 options)

- ☒ We cannot use Matrix Variables in URI without required configuration.
- ☐ Matrix variables should always be to the end of URI
- ☒ We separate matrix variables by Semi colon ";"
- ☐ Matrix Variable should always be of type Map as it has to hold key, value pairs.

Q17 of 20

What is the by default URL for the Spring Security login page?

- ☒ /login
- ☐ /signin
- ☐ /auth
- ☐ /index

Observe the following controller. (Assume necessary imports are done)

```
@RestController
@RequestMapping("/books")
public class BookController {
    @GetMapping("/det/{bookPrice}")
    public List<BookDTO> getBooks(@RequestParam() String bookName, @PathVariable() int bookPrice) {
        //code to search for books based on book name and cost.
    }
}
```

Which of the below is the correct URI to invoke the getBooks() controller method?

- ☐ <http://localhost:8080/bookstore/books/det/400/bookName=java>
- ☐ <http://localhost:8080/bookstore/books/det/bookPrice=400?bookName=java>
- ☒ <http://localhost:8080/bookstore/books/det/400?bookName=java>
- ☐ <http://localhost:8080/bookstore/books/det?bookName=java/400>

What is true regarding the following Spring REST controller?

```
@RestController
@RequestMapping("/customerst")
public class StudentController {

    @PostMapping
    public Integer createStudent(@RequestBody Student student) {
        //code for creating a student goes here
    }
}
```

input JSON data for createStudent method from client:

```
{
  name : "Ramya",
  marks : "80"
}
```

- ☐ The controller throws exception as the consumes attribute is missing in createStudent method
- ☒ The input Student's JSON data is not in correct format
- ☐ @PostMapping annotation should hold some URI path
- ☐ Cannot return Integer value from a controller method

Q20 of 20

Identify the correct order of request processing in Spring Rest.

- ☒ Client, DispatcherServlet, HandlerMapper, Controller, ViewResolver, Client
- ☐ Client, DispatcherServlet, HandlerMapper, Controller, Client
- ☐ Client, HandlerMapper, DispatcherServlet, Client
- ☐ Client, HandlerMapper, ViewResolver, DispatcherServlet, Client