Q1 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})

Q2 of 20

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.

Q3 of 20

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

Q4 of 20

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)

Q5 of 20

Nina is trying to add version to her controller methods using accept header versioning, which of the below is a valid accept header versioning format that nina can use?

- @GetMapping(value = "/{studentid}", produces = "application/vnd.student.v1.json")
- @GetMapping(value = "/{studentid }", produces = "application/vnd.student.app-v1-json")
- @GetMapping(value = "/{studentid }", produces = "application/vnd.v1+json")
- @GetMapping(value = "/{studentid }", produces = "application/json+vnd.student.app.v1")

Q6 of 20

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

- [x] server.port
- [ ] server.servlet.context-path
- [x] server.address
- [ ] server.servlet.context-parameters

Q7 of 20

Have a look at the following controller.

```
@RestController
public class EmployeeController {
    @GetMapping("/infy")    // Line 1
    public List<EmployeeDTO> getEmployees() {
        //List of employees get returned
    }
    @RequestMapping(value = "/infy", consumes="application/json") // Line 2
    public List<Employee> createEmployee(EmployeeDTO emp) {
        //employee get created
    }
}
```

What is wrong with above controller class (Choose any 2 options)

- [x] @RequestBody annotation is missing before EmployeeDTO object.
- [ ] @GetMapping annotation does not has produces attribute, specifying the type of produced data.
- [x] HTTP request method is not specified in @RequestMapping annotation at Line 2.
- [ ] @RequestMapping should not be used at method level.

Q8 of 20

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

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

Q9 of 20

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

- /login
- /signin
- /auth
- /index

Q10 of 20

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

- [x] WebServices
- [ ] Lisp
- [ ] Perl
- [x] Jini

Q11 of 20

Have a look at the following controller.

```
@RestController
@RequestMapping("/bikes")
public class BikeController   {
    //Line 1
    public List<BikeDTO> getBikeDetails() {
        //List of bikes get returned
    }
    // Line 2
    public String createBike(BikeDTO bike) {
        //bike got created
    }
}
```

Which of the below options has statements that can be inserted at Line 1, Line 2 for proper working of the controller?

Line 1 - @RequestMapping(value = “/detail”)

Line 2 - @PutMapping(“/create”)

Line 1 - @GetMapping(“/detail”)

Line 2 - @RequestMapping(value = “/create”, method = “Post”)

Line 1 - @GetMapping(“/detail”)

Line 2 - @PostMapping()

Line 1 - @RequestMapping(value = “/detail”)

Line 2 - @PostMapping(“/create”)

Q12 of 20

Which class in Spring security is used to hash passwords?

- PasswordEncoder
- PasswordHasher
- HashEncoder
- SecureEncoder

Q13 of 20

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

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

Q15 of 20

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

Q16 of 20

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(regexp = "[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(regexp = "[a-zA-Z]*@[a-z].com")

age - @Limit(25,60)

Q17 of 20

Which of the below statements are true regarding Swagger (Choose any 2 options.)

- [ ] "springfox-swagger-ui" is the only dependency required in POM.XML to enable swagger.
- [x] Springfox is the Swagger implementation for SpringBoot.
- [x] OpenAPI Specification is an API description format for REST APIs.
- [ ] Swagger uses DocketApi Bean to search for Rest Controllers.

Q18 of 20

Which of the below are correct ways to handle validation failures in DTO's? (Choose any 2 options)

- [x] Take Errors object as second parameter to the method that is validating incoming DTO's and handle validation failures.
- [ ] Handle "MethodArgumentNotValidException" in a centralized way using @ExceptionHandler annotation.
- [x] Handle "ConstraintViolationException" in a centralized way using @ExceptionHandler annotation.
- [ ] Take ValidationError object as second parameter to the method that is validating incoming DTO's and handle validation failures.

Q19 of 20

What is the correct way of configuring to enable CORS for the entire application?

- Add @EnableCrossOrigin annotation over the class with main() method.
- Implement WebMvcConfigurer Interface and override addCorsMappings method to perform configuration in class with main() method
- Add @CrossOrigins annotation over the class with main() method.
- Implement WebRestConfigurer Interface and override addCorsMappings method to perform configuration in class with main() method

Q20 of 20

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") long phoneNo)
```

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

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

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