

Question - 1

Spring Data Repository Type

Consider the following code.

Admin.java

```
@Entity
@SequenceGenerator(name = "default_gen", sequenceName = "SEQ_ADMIN")
public class Admin {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "default_gen")
    @Column(name = "id")
    private String name;
    private String password;
    private String locale;
    private String email;
    //getter & setters
}
```

Admin Repository.java

```
@Repository
public interface AdminRepository extends X <Admin, Long> {

    Page<Admin> findAllByLocaleOrderByName(String locale, Pageable pageable);
}
```

To run the *findAllByLocaleOrderByName* method with pagination ability, which of the following options can be used in place of *X* in the *AdminRepository* interface?

- ☐ CrudRepository
- ☐ JpaRepository
- ☐ PagingRepository
- ☐ PagingAndSortingRepository

Question - 2

Controller Correct Definition Problem

Given the following controller code, which of the statements are true?

```
@RestController
public class ScoreController {
    @RequestMapping("/myScore/{id}")
    public void myScore(@PathVariable("id") int id, @RequestParam Optional<String> name) {
```

```
}  
}
```

Note: Assume that application is running on *http://localhost:8080*

- ☐ The method is fine and *http://localhost:8080/myScore/id?name=Adam* maps to the method *myScore*.
- ☐ The method is fine, and *http://localhost:8080/myScore/1?name=Adam* maps to the method *myScore*.
- ☐ The method is fine, and *http://localhost:8080/myScore/23?surname=Adamson* maps to the method *myScore*.
- ☐ The mapping of the request param is invalid because *Optional* can't be used for mapping.
- ☐ The path param definition is invalid.

Question - 3

Spring Data Repositories

Given the *users* table that contains the following columns,

```
id: integer  
name: varchar(50)  
age: integer
```

and the following spring data JPA code,

```
@Entity  
@Table(name = "users")  
public class User {  
    @Id  
    private int id;  
    private String name;  
    private String surname;  
    private int age;  
    //getters and setters  
}
```

```
public interface UserRepository extends JpaRepository<User, Integer> {  
    //<CODE 1>  
    @Query("update User u set u.age = ?2 where u.id = ?1")  
    void setAge(int id, int age);  
}
```

```
@Service  
@Transactional(readOnly = true)  
public class UserService {  
  
    @Autowired  
    private UserRepository userRepository;  
    //<CODE 2>  
    public void setUserAge(int id, int age) {  
        userRepository.setAge(id, age);  
    }  
}
```

which of the statements are true?

- ☐ `setUserAge` method runs fine and sets user matching `id` age to the specified `age` value.

- ☐ `setUserAge` throws `InvalidDataAccessApiUsageException` when executed
- ☐ `setUserAge` method runs fine and sets the user matching `id` age to the specified `age` value if `@Transactional(readOnly = false)` is inserted instead of <CODE 2>
- ☐ `setUserAge` method runs fine and sets the user matching `id` age to the specified `age` value if `@Modifying` is inserted instead of <CODE 1> and `@Transactional(readOnly = false)` is inserted instead of <CODE 2>
- ☐ `setUserAge` method runs fine and sets the user matching `id` age to the specified `age` value if `@Transactional(readOnly = false)` is inserted instead of <CODE 1>

Question - 4

Spring Data Transactional

Given the following code, what happens when *StudentRepositoryTest* runs?

Lesson.java

```
@Entity
public class Lesson {
    @Id
    private Long id;
    private String name;
    private Integer courseLength;
    private Integer courseLevel;
    @JsonBackReference
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "student_id")
    private Student student;
    private boolean inProgress;
    //getter & setters
}
```

Student.java

```
@Entity
public class Student {
    @Id
    private Long id;
    private String name;
    private String surname;
    @JsonManagedReference
    @OneToMany(mappedBy = "student", fetch = FetchType.EAGER, cascade = {CascadeType.PERSIST,
CascadeType.REFRESH})
    private List<Lesson> lessons;
    //getter & setters
}
```

StudentRepository.java

```
@Repository
public interface StudentRepository extends JpaRepository<Student, Long>{
    //Y
    Optional<Student> findById(Long id);
}
```

LessonRepository.java

```
@Repository
public interface LessonRepository extends JpaRepository<Lesson, Long> {
```

```
}
```

StudentService.java

```
@Service
public class StudentService{
    private StudentRepository studentRepository;
    public StudentService(StudentRepository studentRepository, LessonRepository lessonRepository) {
        this.studentRepository = studentRepository;
        this.lessonRepository = lessonRepository;
    }

    public void save(Student student) {
        studentRepository.save(student);
    }

    //other crud metods

    //X
    public void deleteStudentWithLessons(Long id) throws SQLException {
        Student student =
            studentRepository.findById(id).orElseThrow(() -> new RuntimeException("Student not found"));
        List<Lesson> lessons = student.getLessons();
        lessons.stream().forEach(lesson -> lesson.setStudent(null));
        studentRepository.delete(student);

        boolean inProgress =lessons.stream()
            .map(Lesson::isInProgress)
            .filter(progress -> progress == true)
            .findFirst()
            .orElseGet(() -> false);
        if (inProgress) {
            throw new SQLException("Cannot delete all Lessons. There are lessons in progress state");
        }
        lessonRepository.deleteAll(lessons);
    }
}
```

StudentRepositoryTest.java

```
@SpringBootTest
class StudentRepositoryTest {
    @Autowired
    StudentRepository studentRepository;
    @Autowired
    StudentService studentService;
    @Autowired
    LessonRepository lessonRepository;
    Student student;
    @BeforeEach
    public void init() {
        student = Student.builder().name("Jorn").surname("Deynhoven").build();
        studentRepository.save(student);
        lessonRepository.saveAll(List.of(
            Lesson.builder().name("Music").courseLength(10).student(student).courseLevel(123).build(),
            Lesson.builder().name("Art").courseLength(20).courseLevel(123).student(student).build(),
            Lesson.builder().name("Trance").courseLength(30).courseLevel(123).student(student).inProgress(true).build()
        ));
    }
    @Test
    void deleteStudentWithLessonsTest() throws SQLException {
        studentService.deleteStudentWithLessons(student.getId());
    }
}
```

```
}  
}  
}
```

- ☐ It will try to delete the student then the DataIntegrityViolationException will be thrown.
- ☐ It will delete the student successfully then an SQLException will be thrown with the message "Cannot delete all Lessons. There are lessons in progress state".
- ☐ If @Transactional annotation is put in place of X (StudentServer.java), after saving the student and lessons, it will update lessons and remove the student. Then it will throw an SQLException and all database changes will be rolled back.
- ☐ It will throw a RuntimeException with the message "Student not found".
- ☐ If @Transactional annotation is put in place of X (StudentServer.java), after saving the student and lessons, it will update lessons and remove the student. Then it will throw an SQLException.

Question - 5

Spring Transactional

```
@Transactional(propagation= Propagation.REQUIRED)  
public void saveProduct(Product product) {  
    productService.save(product);  
    try{  
        saveProductToLogService(product);  
    }catch (Exception ex){  
        log(ExceptionUtils.getStackTrace(ex));  
    }  
}  
@Transactional(propagation=Propagation.REQUIRES_NEW)  
public void saveProductToLogService(Product account) {  
    productLogService.save(product);  
}
```

Which of the following options describes the given code functionality with @Transactional annotation?

- ☐ saveProductToLogService will use the saveProduct's transaction to save the product.
- ☐ saveProductToLogService will use a new transaction for productLogService.save method.
- ☐ As soon as saveProduct method execution is done, saveProductToLogService.save method execution will be reflected to saveProductToLogService database.
- ☐ None of the above

Question - 6

Spring MVC RestController

```
@RestController  
@RequestMapping("/api/product/categories")  
public class ProductCategoryController{  
    @Autowired  
    private CategoryService categoryService;  
  
    @GetMapping("/{id}/{locale}")
```

```
public ResponseEntity<CategoryDto> getByIdAndLocale(@PathVariable("id") Long id,
                                                    @PathVariable("locale") String locale {
    CategoryDto dto = categoryService.getByIdAndLocale(id, locale);
    return new ResponseEntity<>(dto, HttpStatus.OK);
}
}
```

Which of the following options are true about the given controller method?

- ☐ getByIdAndLocale endpoint can return the response in JSON format
- ☐ getByIdAndLocale endpoint does not accept GET request
- ☐ getByIdAndLocale endpoint does not accept POST request
- ☐ getByIdAndLocale endpoint is not accesible without providing id and locale variables

Question - 7

Spring Data Repository

To use Spring data repositories within another jar module and to use them in a Spring Boot application, what annotation should be used in Spring Boot configuration/ SpringBootApplication classes?

- ☐ @EnableJpaRepositories(basePackages = {repositoryPackageOfJarModule})
- ☐ @EnableRepositories(basePackages = {repositoryPackageOfJarModule})
- ☐ @EnableEntities(basePackages = {repositoryPackageOfJarModule})
- ☐ @EnableHibernateRepositories(basePackages = {repositoryPackageOfJarModule})

Question - 8

Spring IOC Container

Which of the following options act as a Spring IOC Container?

- ☐ ApplicationContext
- ☐ DispatcherServlet
- ☐ SpringRunner
- ☐ BeanFactory
- ☐ None of the above

Question - 9

Spring Boot Localization

Which of the following options can be used for internationalization in Spring Boot?

- ☐ MessageResource
- ☐ ParameterResource

- ☐ ResourceBundle
- ☐ None of the above

Question - 10
Dispatcher Servlet

What is the purpose of Dispatcher Servlet in Spring Boot ?

- ☐ It is used for transaction management.
- ☐ It is used for database connection pool management.
- ☐ It is used to handle all HTTP requests and responses.
- ☐ It is used heavily for dependency injection.

Question - 11
Endpoint Security

In Spring Boot Security, the option to enable secure endpoints with @PreAuthorize annotation is:

- ☐ Configuration class annotated with @EnableGlobalMethodSecurity
- ☐ Configuration class annotated with @EnableGlobalMethodSecurity(securedEnabled = true)
- ☐ Configuration class annotated with @EnableGlobalMethodSecurity(prePostEnabled = true)
- ☐ None above

Question - 12
Error Handling

What scenario does this 'custom error handling' block handle?

```
@ControllerAdvice
public class CustomResourceNotFoundExceptionHandler {

    @ExceptionHandler(value = { NoHandlerFoundException.class })
    public ResponseEntity<Object> noHandlerFoundException(Exception ex) {

        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Resource Not Found");
    }
}
```

- ☐ This block of code will be executed when we try to hit an API and it does not exist in the project scope.
- ☐ This block of code will be executed when we try to hit an API and its execution encounters an error.
- ☐ This block of code will be executed when we try to hit an API and it does not have enough data to execute.

☐ This block of code will be executed when we try to hit an API which we are not authorized to access.

Question - 13

Predict the Output

Consider the following Snippet and predict the output.

The following API is triggered: <http://localhost:8080/add/employee?eId=123&fName=Ravi&lastName=Shekhar>

```
@Autowired
private EService eService;

@RequestMapping(value = "/add/employee", method = RequestMethod.GET)
public com.model.Employee add(@RequestParam("eId") String eId, @RequestParam("fName") String fName,
@RequestParam("lastName") String lastName)
{
    return eService.createEmployee(eId, fName, lastName);
}

@Aspect
@Component
public class EServiceAspect
{
    @After(value = "execution(* com.random.service.EService.*(..)) && args(eId, fName, lastName)")
    public void afterAdvice(JoinPoint joinPoint, String eId, String fName, String lastName) {
        System.out.println("After method:" + joinPoint.getSignature());
        System.out.println("Creating Employee with first name - " + fName+ ", second name - " + lastName+ " and
id - " + eId);
    }
}
```

- ☐ After method : com.test.model.Employee addEmployee(String,String,String)
Creating Employee with first name - Ravi , second name - Shekhar and id - 123
- ☐ After method : com.test.model.Employee addEmployee(string,string,string)
Creating Employee with first name - Ravi , second name - Shekhar and id - 123
- ☐ After method : com.test.model.Employee addEmployee(empId,firstName,secondName)
Creating Employee with first name - Ravi , second name - Shekhar and id - 123
- ☐ After method : com.test.model.Employee addEmployee(String,String,String)
Creating Employee with first name - Shekhar, second name - Ravi and id - 123

Question - 14

API

What is the console output when the following API is executed?

application.properties


```
logging.level.root=DEBUG
```

```
@GetMapping("/demo")
public String demo() {
    LOG.debug("ERROR log");
    LOG.trace("DEBUG log");
    LOG.info("TRACE log");
    LOG.error("INFO log");

    someRandomClass.longOperation();

    return "Log demo";
}
```

☐ ERROR log
DEBUG log
TRACE log
INFO log

☐ ERROR log
TRACE log
INFO log

☐ ERROR log
DEBUG log
INFO log

☐ DEBUG log
TRACE log
INFO log