Name kiros Gebregewergis 6/0765 CS544 Final Exam May 2020

Note: Please answer on a separate sheet of blank paper, **except for questions 1, 2, 7, 8 and 12.**

1. [4 points] – Write "true" or "false" next to statements below regarding "**Singleton**" vs. "**Prototype**" beans in Spring?

4/4

- Prototype beans are by default eagerly loaded False ✓
- Singleton beans are by default eagerly loaded True ✓
- Prototype beans do not have a "destroy" cycle True ✓
- Singleton beans do not have a "destroy" cycle false ✓

2. [4 points] - What is the order of events when Spring Context loads? Put a number next to each stage.

4/4

- ✓3 • Spring sets all the properties defined for all beans (dependency injection)
- ✓1 • Spring starts reading XML config file (or runs Java Config)
- ✓4 • Spring calls the init() (or @PostConstruct) methods of all beans that have such a method
- ✓2 • Spring instantiates all singleton beans ("Constructor Arguments" will be set at this stage)

3. [2 points] - What does Spring code below do?

```
@Value("${systemProperties.default-language}" )
private String language;
```

2/2

- It inject value of default-language of system properties to the string language declared here.

4. [5 points] – Consider code below. Is this "**Autowired**" by name or type? Do you see any issues? Explain.

5

```
@Service("customerService")
public class CustomerService {

    @Autowired
    @Qualifier("emailServices")
    private EmailService emailService;

    public void addCustomer(){
        emailService.sendEmail();
    }
}

@Service("emailService")
public class EmailService { … }
```

97/100

5 5. [5 points] – Why would you want to use "Constructor" injection vs. field or property injection in Spring?

5 6. [5 points] - In the context of AOP, what is an "**Aspect**"?

5/5 7. [5 points] – Write a Pointcut Execution expression for an around advice that runs around all the **public** service layer methods. Assume that all the service layer classes are grouped in a package named: **edu.miu.cs.cs544.service**

@Around("execution(public * edu.miu.cs.cs544.service.*.*(..))")

8. [20 points] - Refactor code below and separate the cross-cutting concern into an "Around" advice.

18/20

```
@Service
public class AccountService {

 @Autowired
 private SessionFactory sessionFactory;

 @Autowired
 private AccountDAO accountDAO;

 public boolean deposit(long accountNumber, double amount) {
        Session session = null;
        Transaction tx = null;
        try {
               session = sessionFactory.openSession();
               tx = session.beginTransaction();
               System.out.println("in execution of method deposit");
               Account account = accountDAO.loadAccount(accountNumber);
               account.deposit(amount);
               accountDAO.saveAccount(account);
        } catch (HibernateException e) {
               tx.rollback();
        } finally {
               session.close();
        }
        return true;
 }
}
```

The idea is to separate the repetitive transaction/session start and stop into a reusable advice that can be applied to all the service layer methods of all the service layer classes. Now, fill in the blank below:
Hint: Use the Pointcut Execution expression you wrote for last question

```
@Aspect
@Component
public class TransactionAdvice {
```

@Autowired
SessionFactory sessionFactory;

@Around ("execution(* edu.miu.cs.cs544.service.*.*(..))")

```
public Object transactional(ProceedingJoinPoint call) throws Throwable {
        Session session = null;
        Transaction tx = null;
        try {
               session = sessionFactory.openSession();
               tx = session.beginTransaction();
               Object object = call.proceed();
        } catch (HibernateException e) {
               tx.rollback()
        } finally {
               session.close();
        }
        return object;
 }
```

18/20

9. [10 points] - Explain the "**Un-Repeatable Read**" issue and mention what isolation level solves it. How?

10. [20 points] - Explain the Spring MVC lifecycle using the following example. If you type the following in a browser: http://www.myserver.com/country-api/countries/24/cities, what happens?

```
@RestController
public class CityController {

 @Autowired
 private CityService cityService;

 @GetMapping("/countries/{countryId}/cities")
 public List<City> getCitiesForCountryId(@PathVariable Integer countryId) {

   return cityService.getCitiesForCountryId(countryId);

 }
}
```

11. [10 points] – What is security "**authentication**"? What is security "**authorization**"?

12. [10 points] – Explain the Session per Operation anti-pattern and mention at least two issues with it.

Appendix A: Validation Annotations

Annotation	Data Types	Description
@Null	Any	Check if it's null (affects column)
@NotNull	Any	Check that it's not null
@Valid	Any non-primitive	Go into the object and validate it
@AssertFalse	Boolean	Check that it's false
@AssertTrue	Boolean	Check that it's true
@Future	Date or Calendar	Check that it's in the future
@Past	Date or Calendar	Check that it's in the past
@Size(min=,max=)	String / Collection	Check size is >= min and <= max, column length set to max
@Pattern(regex=,flag=)	String	Check that it matches the regex
@Min(value=)	Numeric types	Check that it's not lower
@Max(value=)	Numeric types	Check that it's not higher
@DecimalMin(value=,inclusive=)	Numeric types	Check that it's not lower
@DecimalMax(value=,inclusive=)	Numeric types	Check that it's not higher
@Digits(integer=,fraction=)	Numeric types	Checks if it has less digits /fractions than given
@CreditCardNumber	String	Credit Cards
@EAN	String	Barcode
@Email	String	Email address

Pointcut Execution Language: **("execution(public * *.*.*(..))")**

ProceedingJoinPoint class (from AOP library) has the following methods:

Return type	Method
java.lang.Object	**proceed**(java.lang.Object[] args)
java.lang.Object	**proceed**()
java.lang.Object[]	**getArgs**()
java.lang.Object	**getTarget**()

@Before, @Around, @After, @AfterThrowing, @AfterReturning

kins above sewers Gloves to

3) It injects value of default-language of system properties to the string language declared here.

4 It is autowired by name. yes I see an issue the name we used to inject in customer service class is different from actual name of the bean Email Service.

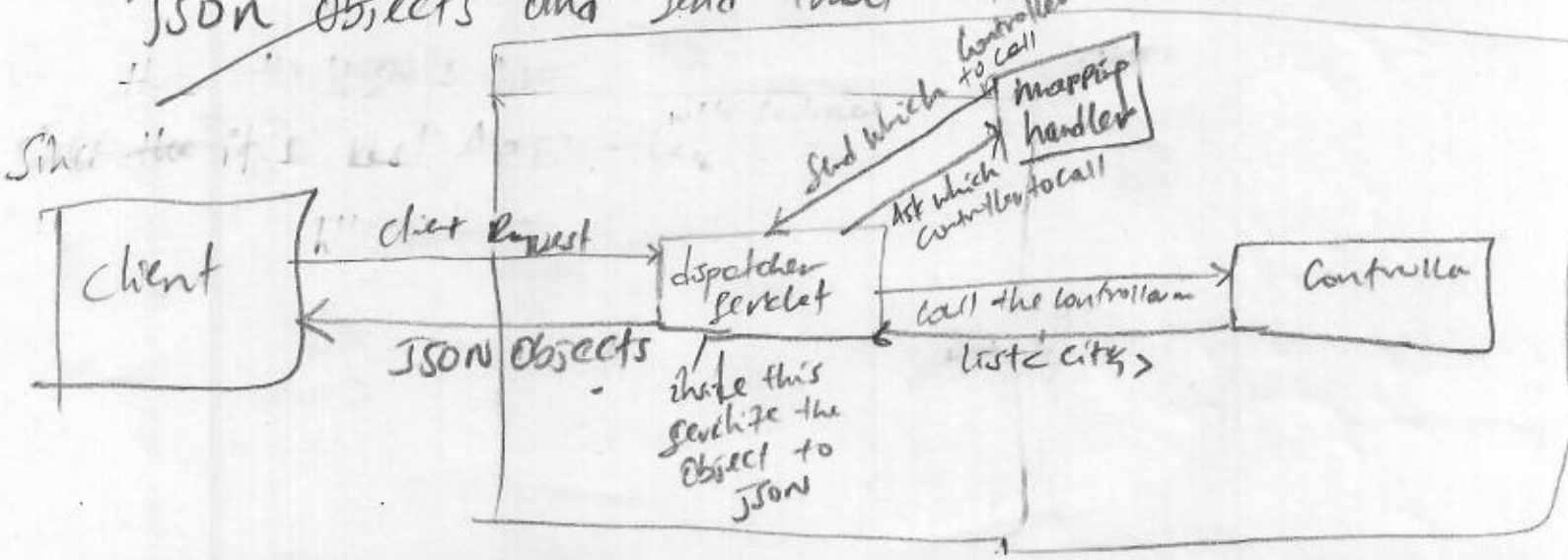
5. Well if you want to execute some intialization code using the injected attribu you may want to use Constructor injection. We shall can achieve this through the setter injection and separated init method but if the prog rammer want it to directly intialize then programmer can use Constructor injection directly
The Other you will not get invalid state exception when you use Constructor injection.

6) Aspect - What crosscutting concern implemelation to execute (Advice) at which location in the code (point cut). which is in short Application of Advice to point cut.

9) Un-Repeatable Read is when you read a data inside a transaction twice, the second read might be different because of concurrent update.
Repeatable solve the issue by applying both Read and write lock on data → Repeatable Read -1
if a trxn read a data then read lock applied to it if if any other transaction try to update the data the waits until the read lock is released or until the trxn that have the read lock finishes.

kiros Gebre generous 6/07 GS

10) The client requests a Get method by typing http://www.myserver.com/countr-api/countries/24/cities in his/her browser After that the Web Container we have only one dispatcher servelet which intercepts all the incoming request from the client. Which does deserialization the incoming request into given java object, validation and other stuffs but in our case the client is requesting get method not post so there is no deserialization. Then then the dispatch servelet calls a handler mapping to identify which controllers method handle the request based on the Request mapping. So the handler mapping map the Request to the controller based on mapping implemention it has then it returns the controller method to be called. then the dispatcher servlet call the controller method based on that then Request mapping. the controller method extract the variable it need requires from the URI and calls the service method that Return a list of city based on country id then the controller method Return list of city to the Dispatcher servlet then the dispatcher servlet serialize the Java object into JSON objects and send that to the client.



11 authentication - is identifying Who the user is

authorization - is all about does the user have previlage to access.
=> does the user have Access controll to this URL

12) session per operation Anti pattern is Session is Open and close per operation.

- session cache never used
- No way to load related entities
- loaded entities are detached automatically

you mean lazy?

No + only that since session is open and close per operation transaction is also closed and opened per operation which result non transactional (not atomic) behavior

The trxn doesn't span a unit of work rather per operation which Violates the ACID of trxn.