Syllabus
Calendar
Announcements
Resources
Forums
Assignments
Tests & Quizzes
Site Info
Gradebook
Roster
Polls
Evaluation System
Proctortrack
Help

Written Test 2

Return to Assessment List

Part 1 of 3 - True/False **5.0 / 5.0 Points**

Question 1 of 12	1.0	1.0 Points

IoC container is DI.

- ○ True
- ◉ False

Rationale:

IOC container is not DI because IOC(Inversion Of Control) container where IOC helps to perform Dependency Injection which is a type of IOC principle. IOC container is responsible for creation of beans and also perform all the lifecycle operations in which DI is a phase in this lifecycle. DI has several types or can be done using constructor injection , setter injection, method injection or field injection.

Question 2 of 12	1.0	1.0 Points

When having both XML and Java Code configuration for Spring context xml overrides the Code configuration.

- ◉ True
- ○ False

Rationale:

Xml overrides the code configuration in order to support the backward compatibility. Ealier spring had only xml configuration and annotation were introduced later. Backward compatibility is the application property to support the software written using old techniques. So spring context xml overrides the code configuration.

Question 3 of 12	1.0	1.0 Points

There are several ways to implement MVC.

- (●) True
- ( ) False

Rationale:

Yes there are in total 3 ways to implement the MVC pattern. At first the request reaches to the controller.
1>1st way where the Controller asks the Model to provide the data. It takes the data and supplies to the view.
2> 2nd way where the controller asks the model to provide the data to the view . then view returns the data to the client.
3> 3rd way where the controller asks the view to ask the data to the model . After returning data view later sends data back to the client.

Question 4 of 12	1.0	1.0 Points

Cyclic dependency is not possible in Spring, thanks to Spring DI.

- ( ) True
- (●) False

Rationale:

Cyclic dependency is where more than 2 classes requires dependency of each other in such manner that it forms a cycle. In such scenarios if we use constructor injection , our application will throw an exception. So it can be resolved by using various techniques. One of which is setter injection in which after the instances are created the dependencies are injected later only when needed in application code.

Question 5 of 12	1.0	1.0 Points

Spring Context manages the full lifecycle of all beans.

- ( ) True

Question 5 of 12	1.0	1.0 Points

Spring Context manages the full lifecycle of all beans.

- ○ True
- ◉ False

Rationale:

Only singleton beans lifecycle is fully managed by the spring context whereas in case of prototype beans spring context doesn't manage its destroy phase meaning the destroy methods of the beans are not called by default. This behavior is there for memory and runtime optimization as there can be any number of prototype beans. However the developer can have custom implementation of destroy behaviour of prototype beans by implementing BeanPostProcessor and FactoryAware interfaces.

Part 2 of 3 - Short Answers **4.1 / 5.0 Points**

Question 6 of 12	1.0	1.0 Points

What is the difference between Concern and Cross-cutting concern?

A software application as a whole is a collection of functions where each function solves a particular functionality known as concerns. One of the example is a function which saves an object to a database. Other example is a function which sends email .

A cross cutting concern is any particular single task which each of these concern is doing such as logging to a file or database. Hence a cross cutting concern is a same particular concern which is scattered all over the application across multiple classes and layers of application such as logging , transaction handling, authentication etc. Now such concerns now can be handled using DRY principle where we write a single method and all other methods are calling the same method from different places. But a better solution is Aspect Orienting Programming AOP where we dont need to call each time from different places instead to achieve this we write configuration with framework provided AOP options.

Question 7 of 12	1.0	1.0 Points

Question 7 of 12	1.0	1.0 Points

Instead of configuring beans in XML or Java code, you can use annotations and spring will scan for the beans. When doing so, it is said that it is better not to use @Component and use one of its sub-annotations. Why?
@Component annotation is inherited by 3 other annotations @Controller @Service and @Repository. These help to identify easliy what each of the class is doing in particular and helps to separate the layers in an application. @Controller will have the methods that handles the incoming requests. @Service has the business logic . @Repository will have methods to perform the database operations. Since they are inherting from component annotation they will be treated as beans and also they provide their particular functionalities like whenever a request comes at dispatcherServlet it will look for methods in classes with @Controller annotation. Also @Repository will automatically convert any exception in database to Spring specific unchecked exception.

Question 8 of 12	1.5	2.0 Points

State two advantages and one disadvantage for using @Pointcut when using AspectJ.
Advantages:

1>Pointcut is a colleciton of joinpoints. @Pointcut helps to define a function which can be used as values of annotations like @Before @After . Thus helps to define what the pointcut does in easily understandable language and same pointcut can be used in multiple places. If changed at one place it needn't be changed at all the places.
2> Supports pointcut language expression and helps to define a particular collection of joinpoints where we can apply the advice and thus achieve finer level of control.

Disadvantages:

1>Pointcut supports the regular expressions which requires a large time to filter out the joinpoints where our aspect should be executed.

Comment:I don't fully understand your disadvantage.

Question 9 of 12	0.6	1.0 Points

What is written in each layer of the 5 layered (View, Control, Service, Domain, and Persistence)? Give exact examples not just terminology, maybe use your MPP course project or your Job-Search application that you plan to write.
In MPP we had the library management system based on JAVAFX.
View:
In view layer we write the presentation logic. It is related to how the data is shown to the user.
We used generated xml for this.

Question 9 of 12	0.6	1.0 Points

What is written in each layer of the 5 layered (View, Control, Service, Domain, and Persistence)? Give exact examples not just terminology, maybe use your MPP course project or your Job-Search application that you plan to write.
In MPP we had the library management system based on JAVAFX.
View:
In view layer we write the presentation logic. It is related to how the data is shown to the user.
We used generated xml for this.

Control:
Control is done by writing the eventhandler methods. From control layer it calls the service layer.

Service:
Service layer consists of function for doing paticular piece of task. It helps to map from domain objects to db entities and vice versa.

Domain:
Domain layer refers to a business domain Object which is composed of entities which is persisted in database. It refers to the DTO object (Data Transfer Object) between the serveral domains in Service Oriented Architecture(SOA).

Persistance:
Persitance layer has the classes and methods that directly interacts with the DB.

Comment:not clear what your controler layer is doing.
mapping domain to db is the persistence layer's job.

Part 3 of 3 - Architecture **5.0 / 5.0 Points**

Question 10 of 12	2.0	2.0 Points

The library is adding magazines to the library system. You need to add magazines with the ability to perform CRUD operations on magazines. Which of the following approaches will you apply.
1- Create a Magazine entity, perform the CRUD operations in the entity. Have the transaction created in each method.
2- Create a Magazine entity, perform the CRUD operations in a service class. Have the transaction passed to each method as a parameter and use it in each method.
3- Creat an aspect for transactions, when the advice creates the transaction, and the pointcut is the CURD operation methods.

Question 10 of 12	2.0	2.0 Points

The library is adding magazines to the library system. You need to add magazines with the ability to perform CRUD operations on magazines. Which of the following approaches will you apply.
1- Create a Magazine entity, perform the CRUD operations in the entity. Have the transaction created in each method.
2- Create a Magazine entity, perform the CRUD operations in a service class. Have the transaction passed to each method as a parameter and use it in each method.
3- Creat an aspect for transactions, when the advice creates the transaction, and the pointcut is the CURD operation methods.
In Spring a transaction helps to achieve either full success or full failure by performing rollbacks.

Good : Option 3 where i will create the transation using aspects so that the advice is applied in each of the pointcut where each pointcut will be a crud opration method. Since Transaction is a crosscutting concern we can write its code separately as an aspect.

Ugly: Option 2 is bad compared to advice because we need to write transaction create at beginning of each method in a controller from where the request to the application enters. Thus it results in scattering of the same concern all over the code.

Bad: Option 1 because having each method will have a repetitive code to create a transcation . And since each operation requires only one transction, creating a transaction in each operation doesn't help to achieve transaction advantages.

Question 11 of 12	2.0	2.0 Points

We need to add function performance profiling to our application (this is where you need to time all your function calls and make sure they do not exceed a certain amount). For the sake of DRY, we already have a class (CodeProfiler) that can be used to determine execution time it just needs to be used before and after each method that needs profiling.
1- Add CodeProfiler to the beginning and end of each method that needs profiling.
2- Add advice that uses the CodeProfiler and creates a PointCut(s) to cover the methods that need profiling.
3- Add a Profiling Service and use that to invoke the domain methods that need profiling, or even use it to invoke the service methods.
Good: Option2 where we use an advice as it helps to write this specfic cross cutting concern
and supports expression language to filter which methods are to be called. There is no code repetition as it is required in other 2 cases.

Ugly: Option3 we can use a profile service in middle before invoking the domain methods but
then all the classes needs the CodeProfile service to be injected and the invoke method code is repeated.

Bad: Option1 because we need to call the codeProfiler at the beginning and end of each of the function and the code is repeated everywhere .

3- Add a Profiling Service and use that to invoke the domain methods that need profiling, or even use it to invoke the service methods.
Good: Option2 where we use an advice as it helps to write this specfic cross cutting concern
and supports expression language to filter which methods are to be called. There is no code repetition as it is required in other 2 cases.

Ugly: Option3 we can use a profile service in middle before invoking the domain methods but
then all the classes needs the CodeProfile service to be injected and the invoke method code is repeated.

Bad: Option1 because we need to call the codeProfiler at the beginning and end of each of the function and the code is repeated everywhere .

Question 12 of 12	1.0	1.0 Points

Your company wishes to add a community version of their product which will be a free version with limited functionality. There is also a possibility to add an education version in the future. The product is a Spring application. The functionality that will not exist in the community version has already been identified and refactored out into specific Classes and Services.
1- Since the code has already been refactored we simply add if statements in the Controller and/or Service layer to determine which functionality is allowed and invoke it accordingly.
2- We create a new configuration for the new community edition that will not create the Services and Classes that have been refactored. Then this becomes the community application version of the product. Copy the project so that you have two copies one for each product type (each uses a different configuration). We can use the same repository for the two projects and have each project use a different Main to select the proper configuration.
3- We create a new configuration for the new community edition that will not create the Services and Classes that have been refactored. We keep this configuration in the original project and use profiling to select the appropriate product.
Good: Option 3
Using profiling helps to achieve the same goal in a easier way.

Bad: Option 2
This also helps to achieve our goal but .

Ugly: Option 1 . Because a lot of code needs to be written. and it is a bad way because of tight coupling.