Project 1: Questions

Using the information provided in the assignment to answer the following questions. Please be thorough in your answer. Helpful links are below.

<https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/beans/factory/UnsatisfiedDependencyException.html>

<https://stackoverflow.com/questions/39173982/what-is-a-nosuchbeandefinitionexception-and-how-do-i-fix-it>

<https://stackoverflow.com/questions/978759/what-is-lazy-initialization-and-why-is-it-useful>

https://softwareengineering.stackexchange.com/questions/32581/how-do-you-explain-separation-of-concerns-to-others

<http://olivergierke.de/2013/11/why-field-injection-is-evil/>

<https://www.logicbig.com/tutorials/spring-framework/spring-core/inject-bean-by-name.html>

1. Explain why we received the two exceptions: UnstatisfiedDependencyException with secondary exception ***NoSuchBeanDefinitionException*** in Section 3: Step 8.
2. Explain why we received the two exceptions: UnstatisfiedDependencyException with secondary exception ***NoUniqueBeanDefinitionException*** in Section 3: Step 11.
3. Explain why we did **not** receivea ***NoUniqueBeanDefinitionException*** in Section 3: Step 13.
4. Explain why we did **not** receive a ***NoUniqueBeanDefinitionException*** in Section 3: Step 15.
5. Given the following Spring JavaConfig code below answer the following questions:
   1. What are the Bean names of each Bean? (Note: Bean names do not have to be unique)
   2. Will I receive an Exception if I replace the code below with the current HouseDisburseConfig.java in the **JavaConfig project** and run the application?
   3. Explain why Spring does **not** try to inject the “myString” method into the “houseDisburseDAO” method even though the “myString” Bean name matches the name of the DataSource parameter variable in the “houseDisburseDAO” Bean.

@Configuration

**public** **class** HouseDisburseConfig {

@Bean

**public** DataSource dataSource(){

**return** **new** EmbeddedDatabaseBuilder()

.addScript("classpath:2013Q4\_HOUSE\_DISBURSE.sql").build();

}

@Bean

**public** DataSource ds(){

**return** **new** EmbeddedDatabaseBuilder()

.addScript("classpath:potentialLeads.sql").build();

}

@Bean("ds")

**public** String myString() {

**return** "hello";

}

@Bean

**public** JdbcHouseDisburseDAO houseDisburseDAO(DataSource ds) {

JdbcHouseDisburseDAO hdDAO = **new** JdbcHouseDisburseDAO();

hdDAO.setDataSource(ds);

**return** hdDAO;

}

}

1. In Section IV, step 8 we set the component scanning to @ComponentScan(“disburse”). How would you change this code for best practices so that it only scans **ONLY** the package(s) that have the needed Spring annotations?
2. In Section IV, step 11 why did we receive an error, but did not receive an error on step 13?
3. What does the @Lazy annotation do? Why did we not receive an error in Section IV, Step 18, but did receive one in Section IV, Step 20? (Answer the question as it relates to @Lazy not as it relates to Exceptions received)
4. In the assignment we accessed a Bean by using the following code:

JdbcHouseDisburseDAO hdDAO = (JdbcHouseDisburseDAO)ctx.getBean("jdbcHouseDisburseDAO");

What are two other (and better) ways we could’ve written this code and accessed this Bean? (**and thus avoided the casting**)

1. A class is defined as a Bean below using AnnotationConfig. What is the default Bean name?

@Component

**public** **class** AccountRepositoryImpl {

**private** DataSource dataSource;

@Autowired

**public** **void** setDataSource(DataSource dataSource) {

**this**.dataSource = dataSource;

}

}

1. Below is an answer from the link shown regarding how to explain what “Separation of Concerns” is to others. Read this and answer the following question:
   1. Based on this description below - how does Spring endorse the Separation of Concerns (SoC) design principle? Give one example how Spring accomplishes this, and please be thorough with your answer.

<https://softwareengineering.stackexchange.com/questions/32581/how-do-you-explain-separation-of-concerns-to-others>

Imagine you have a program which has been released. A customer comes along and offers to pay you for an enhancement to one of its features. In order to get the money, you will need to change your program to add the new feature. Some of the things that will influence what your profit margin is are:

1. how much code you have to change
2. how easy it is to make the changes
3. how likely you are to break existing features that are being used by other customers
4. how much you can reuse you existing model/architecture

Separation of concerns helps you to get more positive answers to these questions.

1. if all of the code for a particular behavior of the application is separated out, then you will only have to change code directly associated with your new feature. Which should be less code to change.
2. if the behaviors you are interested in are neatly separated from the rest of the application it is more likely you will be able to swap in a new implementation without having to fully understand or manipulate the rest of the program. It should also be easier to find out which code you need to change.
3. Code that you do not have to change is less likely to break than code that you do change. So splitting up the concerns helps you to avoid breakage in unrelated features by preventing you from having to change code that they could call. If your features are mixed up together you might change the behavior of one by accident while trying to change another one.
4. If your architecture is agnostic to technical or business logic detail then changes to implementation are less likely to require new architectural features. For example, if your main domain logic is database agnostic then supporting a new database should be as easy as swapping in a new implementation of the persistence layer.
5. We’ve written code that looks like the below in our assignment. Why does Spring not recommend using field-level injection?

@Repository

**public** **class** JdbcHouseDisburseDAO {

@Autowired

**private** DataSource dataSource;

1. Why did we not receive an error in Section IV. Step 35? But received an error on Step 37?
2. In Section IV, Step 38 we used the @Qualifier annotation to help resolve ambiguous Beans and avoid the ***NoUniqueBeanDefinitionException***. Examine the code on the next page. Suggest **two** ways of changing the code to resolve the two ambiguous Beans on the next page such that the “loansDS” Bean is injected into the setDataSource method in the JdbcBankDAO class using the **@Autowired** annotation.
3. A class is annotated with @Component (see next page). Assuming the Configuration class is correct, if you change the annotation to @Repository, will you receive an error? If you change it to @Service, will you receive an error? Why or why not?

@Component

**public** **class** JdbcBankDAO {

**private** DataSource dataSource;

@Autowired

**public** **void** setDataSource(DataSource dataSource) {

**this**.dataSource = dataSource;

}

}

1. You are working on a Spring application that has been configured using AnnotationConfig. You’ve went through all the packages and counted over 100 class files. The @ComponentScan is pointed at the root package. Is there any way for you to know what classes in every package are annotated with @Component, @Service, or @Repository?
2. In Section V, steps 7 thru 9, what was the most obvious advantage of using a JDBC namespace in the HouseDisburseConfig.xml document?
3. Given the code below how could you change the .xsd schema documents such that it uses the most recent version of the Spring technologies?

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:jdbc=*"http://www.springframework.org/schema/jdbc"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-4.1.xsd*

*http://www.springframework.org/schema/jdbc*

*http://www.springframework.org/schema/jdbc/spring-jdbc-3.8.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.9.xsd"*>

</beans>

1. The following code comes from an XMLConfig file. Assuming the correct namespaces are called answer the following questions.
2. Is the <jdbc:embedded-database> code representing a Bean?
3. What is the name of the setter method that “acmeDS” is being injected into?

<jdbc:embedded-database id=*"acmeDS"*>

<jdbc:script location=*"classpath:acme-corporation.sql"*/>

</jdbc:embedded-database>

<bean id=*"jdbcAcmeCorpDAO"* class=*"disburse.dao.JdbcAcmeCorpDAO "*>

<property name=*"myDataSource"* ref=*"acmeDS"*> </property>

</bean>