LAB- Dependency Injection using Java Configuration

In this lab you will be understanding how to configure beans using Java Configuration

In this lab, you will be working on **01javaconfig-start** project

Steps

1) Open pom.xml and observe that all the dependencies are inherited from the parent project.

2) Open the application-config.xml and observe the bean definitions. We want to eliminate xml configuration totally.

Create a new Java Class with name **ApplicationConfig** under **com.way2learnonline.config** package

To tell that this is a configuration file, annotate it with @Configuration

Configure DataSource as a bean as shown below :

```
@Bean
public DataSource dataSource(){
	BasicDataSource dataSource= new BasicDataSource();
	dataSource.setDriverClassName("com.mysql.jdbc.Driver");
	dataSource.setUrl("jdbc:mysql://localhost/springdb");
	dataSource.setUsername("root");
	dataSource.setPassword("root");
	return dataSource;
}
```

A method annotated with @Bean is equivalent to <bean/> tag in xml. method name will the bean id by default . You can configure a bean name using the following syntax :

```
@Bean(name="ds")
```

But don't do it now. let us go with defaults.

Right now we are hard coding all the database configuration values like url,username,etc. We will externalize them also in next steps.

We assume that tables are already created by now. So, don't want to execute the scripts right now. We will see how to configure to execute scripts in next steps.

3) Next define JdbcAccountRepositoryImpl ,JdbcRewardRepositoryImpl, JdbcTransactionRepositoryImpl as beans as shown below :

```
@Bean
public AccountRepository accountRepository(){
	return new JdbcAccountRepositoryImpl(dataSource());
}

@Bean
public RewardRepository rewardRepository(){
	return  new JdbcRewardRepositoryImpl(dataSource());
}
@Bean
public TransactionRepository transactionRepository(){
	return new JdbcTransactionRepositoryImpl(dataSource());
}
```

Understand that calling datasource() method is equivalent to using ref="" in bean tag in xml

4) Configure EmailServiceImpl and BankServiceImpl also as beans as shown below :

```
@Bean
public EmailService emailService(){
	return new EmailServiceImpl();
}

@Bean
public BankService bankService(){
	BankServiceImpl bankServiceImpl= new BankServiceImpl();
	bankServiceImpl.setAccountRepository(accountRepository());
	bankServiceImpl.setEmailService(emailService());
	bankServiceImpl.setRewardRepository(rewardRepository());
	bankServiceImpl.setTransactionRepository(transactionRepository());
	return bankServiceImpl;
}
```

Again observe that we are injecting the dependencies by calling corresponding methods.

5) As we are done with configuring all the bean definitions, open BankServiceTests.java and create AnnotationConfigAplicationContext as shown below :

```
context= new AnnotationConfigApplicationContext(ApplicationConfig.class);
```

7) Run the test case and make sure that it will pass.

8) Now we want all the infrastructure related beans to be defined in a separate class.
Create **InfrastructureConfig.java** under **com.way2learnonline.config of src/main/java .**
Annotate the class with @Configuration as it is a configuration file

9)Cut the dataSource bean definition from ApplicationConfig.java and paste it inside InfrastructureConfig.java

10) Now since we have 2 configuration files, Create application context using both files as shown below :

```
 new
AnnotationConfigApplicationContext(ApplicationConfig.class,InfrastructureConfig.class);
```

11) Run the test and make sure it will pass.

12) Now we want to externalize the database configuration values to a properties file.
Already u have been provided with db-prod.properties and db-test.properties under src/main/resources and src/test/resources respectively.

To specify the location of property file, use `@PropertySource("classpath:db-test.properties")` on top of `InfrastructureConfig.java`

13) All the properties will be populated inside Environment Object. You can autowire Environment into InfrastructureConfig.java as show below :

```
@Autowired private Environment env;
```

14) Now modify the DataSource Bean Definition as shown below :

```
@Bean
public DataSource dataSource(){
	BasicDataSource dataSource= new BasicDataSource();
	dataSource.setDriverClassName(env.getProperty("db.driverclassname"));
	dataSource.setUrl(env.getProperty("db.url"));
	dataSource.setUsername(env.getProperty("db.user"));
	dataSource.setPassword(env.getProperty("db.password"));
	return dataSource;
}
```

15) Run the test case and make sure that it will pass

16) Now we want to initialize database by executing scripts similar to below tag in xml :

```
<jdbc:initialize-database data-source="dataSource">
	<jdbc:script location="classpath:dbscripts.sql"/>
```

```
</jdbc:initialize-database>
```

17) First inject dbscripts.sql as resource as shown below in InfrastructureConfig.java

```
@Value("classpath:dbscripts.sql")private Resource dbscript;
```

Now Configure the beans as below :

```
@Bean
public DataSourceInitializer dataSourceInitializer(DataSource dataSource){
	DataSourceInitializer initializer= new DataSourceInitializer();
	initializer.setDataSource(dataSource);
	initializer.setDatabasePopulator(databasePopulator());
	return initializer;
}

private DatabasePopulator databasePopulator() {
    ResourceDatabasePopulator populator = new ResourceDatabasePopulator();
    populator.addScript(dbscript);
    return populator;
}
```

18) Now drop the database springdb and recreate it. Now run the test case and make sure it will pass . Go to db and check if tables are created according to script.

19) We dont want to configure Bean definitions for all our services and repositories manually.

So, go to ApplicationConfig.java and comment out all the @Bean annotatated methods.

Enable Component Scanning for com.way2learnonline package . Goto BankServiceImpl, EmailServiceImpl, JdbcAccountRepositoryImpl, JdbcRewardRepositoryImpl, JdbcTransactionRepositoryImpl and annotate them with @Component and wire their dependencies using @Autowired

Run the Testcase and make sure it will pass.

Congratulations !! You know how to configure beans using java configuration