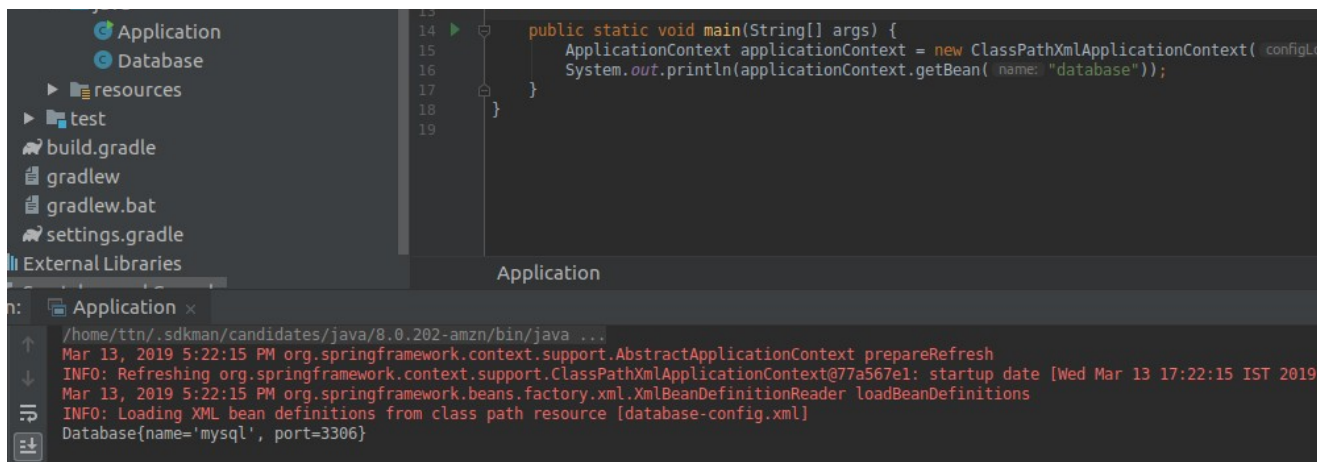


Exercise

1. Create a class Database with 2 instance variables port and name. Configure Database class bean in spring container through Spring XML. Initialize instance variables using setter method.

```
<!-- Using Setter method to Initialize Database instance -->
<bean class="Database" id="database">
    <property name="name" value="mysql"/>
    <property name="port" value="3306"/>
</bean>
```

2. Get the bean of the class from spring container and print the values of the instance variable.



```
public static void main(String[] args) {
    ApplicationContext applicationContext = new ClassPathXmlApplicationContext("config.xml");
    System.out.println(applicationContext.getBean("database"));
}
```

Application

```
/home/ttn/.sdkman/candidates/java/8.0.202-amzn/bin/java ...
Mar 13, 2019 5:22:15 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@77a567e1: startup date [Wed Mar 13 17:22:15 IST 2019]
Mar 13, 2019 5:22:15 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [database-config.xml]
Database(name='mysql', port=3306)
```

3. Create a class Restaurant. Create an interface HotDrink with an abstract method prepareHotDrink. Create two classes Tea and ExpressTea which implements HotDrink Class. Create an instance variable of type HotDrink in Restaurant class. Configure Tea and ExpressTea classes beans in Spring XML. create a bean with the name teaRestaurant of type Restaurant which inject Tea object as dependency using setter method.

```
<bean class="Tea" id="tea" autowire-candidate="false">
    <property name="name" value="Chai"/>
</bean>

<bean class="ExpressTea" id="expressTea" autowire-candidate="false">
    <property name="name" value="ExpressChai"/>
</bean>

<bean class="component.Restaurant" id="teaRestaurant" autowire-candidate="false" scope="prototype">
    <property name="hotDrink" ref="tea"/>
</bean>
```

4. Get both the beans and invoke prepareHotDrink method via hotDrink instance variables. Try inner bean with expressTeaRestaurant.

```
HotDrink hotDrink = (HotDrink) applicationContext.getBean("tea");
System.out.println(hotDrink.prepareHotDrink());
HotDrink hotDrink1 = (HotDrink) applicationContext.getBean("expressTea");
System.out.println(hotDrink1.prepareHotDrink());
Restaurant restaurant = (Restaurant) applicationContext.getBean("teaRestaurant");
System.out.println(restaurant);
Restaurant restaurant1 = (Restaurant) applicationContext.getBean("expressTeaRestaurant");
System.out.println(restaurant1);
```

5. Create Class Complex as follows: class complex { List list;

Set set;

Map map;

} Initialize all the instance variables of the complex class using Spring XML. Give bean name as complexBean. Get the bean and display the properties.

```
Mar 13, 2019 5:29:33 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@77a567e1: startup date [Wed
Mar 13, 2019 5:29:33 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [complex-config.xml]
Mar 13, 2019 5:29:33 PM Application main
INFO: Complex{list=[1, 2, 3], set=[a, b, c], map={10=a, 11=b, 12=c}}
```

6. Autowire hotDrink in Restaurant with tea bean byName, byType and constructor.

```
<!--Autowire by constructor-->
<bean class="component.Restaurant" id="teaRestaurantByConstructor" autowire="constructor"/>

<bean class="Tea" id="hotDrink1">
  <property name="name" value="Chai By Constructor autowiring"/>
</bean>
```

```
<!--Autowire by Name-->
<bean class="component.Restaurant" id="expressTeaRestaurantByName" autowire="byName"/>

<bean class="ExpressTea" id="hotDrink">
  <property name="name" value="Express Chai By Name Autowiring"/>
</bean>
```

```
<!--Autowire by type-->
<bean class="component.Restaurant" id="teaRestaurantByType" autowire="byType"/>
<bean class="Tea">
  <property name="name" value="Chai By Type Autowiring"/>
</bean>
```

```
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@77a567e1: start
Mar 13, 2019 7:42:31 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefi
INFO: Loading XML bean definitions from class path resource [restaurant-config.xml]
Mar 13, 2019 7:42:31 PM org.springframework.context.support.AbstractApplicationContext prepareRefre
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@5a8e6209: start
Restaurant{hotDrink=Chai By Constructor autowiring Prepared}
Restaurant{hotDrink=Express Chai By Name Autowiring Prepared}
Mar 13, 2019 7:42:31 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefi
INFO: Loading XML bean definitions from class path resource [restaurant-autowire-type-config.xml]
Restaurant{hotDrink=Chai By Type Autowiring Prepared}
```

7. Set the scope of the teaRestaurant bean as proptotype and check the scope type after accessing the bean.

```
<!--Setting scope prototype-->
<bean class="component.Restaurant" id="teaRestaurant" autowire-candidate="false" scope="prototype">
  <property name="hotDrink" ref="tea"/>
</bean>
```

```
//Exercise 7
Restaurant teaRestaurant = (Restaurant) applicationContext.getBean( name: "teaRestaurant");
System.out.println("Checking prototype scope as references equality gives : " + (restaurant == teaRestaurant));
```

8. Use @Required annotation for hotDrink setter method in Restaurant class.

```
<bean class="Tea" name="hotDrink2">
  <property name="name" value="Chai by Setter"/>
</bean>

<!--HotDrink setter method in Restaurant class is Required-->
<bean class="component.Restaurant" id="teaRestaurantBySetterAnnotation">
  <!--<property name="hotDrink" ref="hotDrink2"/>-->
</bean>
```

Required properties missing: 'hotDrink' [more...](#) (Ctrl+F1)

```
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@eec5a4a: startup da
Mar 13, 2019 7:55:23 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitio
INFO: Loading XML bean definitions from class path resource [restaurant-annotation-autowiring.xml]
Using Setter Autowire : Restaurant{hotDrink=Chai by Setter Prepared}
```

9. Use @Autowired annotation to wire Tea with Restaurant class Using setter method, field and constructor.

```
<bean class="Tea" name="hotDrink2">
  <property name="name" value="Chai by Setter"/>
</bean>

<!--HotDrink setter method in Restaurant class is Required-->
<!--@Autowired On Setter-->
<bean class="component.Restaurant" id="teaRestaurantBySetterAnnotation">
</bean>
```

```
<bean class="Tea" name="hotDrink">
  <property name="name" value="Chai by Field"/>
</bean>

<!--@Autowired by Field Name-->
<bean class="component.Restaurant" id="teaRestaurantByFieldAnnotation"/>
```

10. Use @Component, @Controller, @Repository etc annotation to configure Tea and Restaurant in Spring Container.

```
@Component("restaurantAnnotation")
public class Restaurant {
  // ...
}
```

```
<!--For using annotation's like @Component and using them as beans-->
<context:component-scan base-package="component"/>
```