How to Handle NoUniqueBeanDefinitionException

Problem

Caused by: org.springframework.beans.factory.**NoUniqueBeanDefinitionException**: **No qualifying bean of type 'com.spring.core.di.MessageService' available: expected single matching bean but found 2: emailService,SMSService at org.springframework.beans.factory.config.DependencyDescriptor.resolveNotUnique(DependencyDescriptor.java:220)**

**Solution**

In the next lecture, we will learn constructor-based dependency injection. We will create spring beans for EmailService and SMSService classes to demonstrate the constructor-based dependency injection.

In Spring Framework, when you declare a bean without specifying a name for it, the Spring container uses a default naming convention to generate a name for that bean.

**Default Naming Convention**: The default bean name is the **unqualified class name, with the first letter changed to lowercase**.

In our case, we have the SMSService class which contains the first four capital letters, and the Spring container configures the default bean name as "SMSService" (since the name starts with an acronym).

We want the bean name as "smsService" as we inject SMSService in MessageSender:

@Autowired

public MessageSender(@Qualifier("emailService") MessageService messageService,

MessageService smsService) {

this.messageService = messageService;

this.smsService = smsService;

System.out.println("Constructor based dependency injection2");

}

Due to this bean name ambiguity, the Spring container throws NoUniqueBeanDefinitionException.

So to resolve this NoUniqueBeanDefinitionException exception, we need to configure the bean name smsService for the SMSService class explicitly.

For example:

@Component("smsService")

Complete code:

@Component("smsService")

public class SMSService implements MessageService{

@Override

public void sendMessage(String message){

System.out.println(message);

}

}

Make sure that in the next lecture, you configure the bean name for the SMSService class: @Component("smsService")

Quick Guide on Spring Bean Naming Convention

In Spring Framework, when you declare a bean without specifying a name or ID for it, the framework uses a default naming convention to generate a name for that bean.

Default Naming Convention: The default bean name is the unqualified class name, with the first letter changed to lowercase.

For example:

If your bean class is com.example.MyService, the default bean name will be myService.

If your bean class is com.example.HTTPService, the default bean name will be HTTPService (since the name starts with an acronym).

However, remember that if you provide a name or ID explicitly, that will take precedence over the default naming convention.

For example:

@Component("smsService")

public class SMSService implements MessageService{

@Override

public void sendMessage(String message){

System.out.println(message);

}

}

In this case, the bean name will be smsService instead of the default SMSService.