**Usage of @Lookup Annotation in Spring – 2022**

A method annotated with @Lookup tells Spring to **return an instance of the method's return type** when we invoke it. @Lookup annotation in Spring is helpful when we like to implement a factory method which returns a new bean on every call without we implementing the method. the container will generate runtime subclasses of the method's containing class via CGLIB, which is why such lookup methods can only work on beans that the container instantiates through regular constructors (i.e.**lookup methods cannot get replaced on beans returned from factory methods** where we can't dynamically provide a subclass for them).

Practical Usage

**Use Case-1**:

@Component

**@Scope("prototype") 🡸 If you remove this line, you will get same instance**

public class EmailNotification {

}

@Service

public class SampleServiceImpl {

// @Autowired 🡸 If you use @Autowired, you will always get the same instance

// private EmailNotification notification;

@Lookup

public EmailNotification getNotification() {

return null; 🡸 Mark it, it is returning null

}

}

Note: **Simply autowiring an object will not work**

**How to use it**

@Component

public class Startup {

@Autowired

private SampleServiceImpl service;

@EventListener(ApplicationReadyEvent.class)

public void startup() {

System.out.println("Notification: "+service.getNotification());

System.out.println("Notification: "+service.getNotification());

System.out.println("Notification: "+service.getNotification());

}

}

OUTPUT

Notification: [com.ddlab.rnd.model.EmailNotification@18371d89](mailto:com.ddlab.rnd.model.EmailNotification@18371d89) 🡸 Different instance

Notification: [com.ddlab.rnd.model.EmailNotification@4f3faa70](mailto:com.ddlab.rnd.model.EmailNotification@4f3faa70) 🡸 Different instance

Notification: [com.ddlab.rnd.model.EmailNotification@4832f03b](mailto:com.ddlab.rnd.model.EmailNotification@4832f03b) 🡸 Different instance

**Use Case-2**

In case of an object, we need child object to be created as new instance every time.

@Data

@Component

public class Employee {

private String name;

@Lookup

public Address getAdrs() {

return null;

}

}

@Component  
@Scope("prototype")  
public class Address {  
 private String cityName;  
}

@Component  
public class Startup {  
  
 @Autowired  
 private Employee emp;  
  
 @EventListener(ApplicationReadyEvent.class)  
 public void startup() {  
 System.out.println("Employee: "+emp);  
 System.out.println("Address: "+emp.getAdrs());  
 System.out.println("Address: "+emp.getAdrs());  
 }  
}

**OUTPUT**

Employee: Employee(name=null)

Address: [com.ddlab.rnd.model.Address@5b275811](mailto:com.ddlab.rnd.model.Address@5b275811) 🡸 Different Instance

Address: [com.ddlab.rnd.model.Address@2f0ed952](mailto:com.ddlab.rnd.model.Address@2f0ed952) 🡸 Different instance