أكاديمية طويق
Tuwaiq Academy

Exercise (Spring Container)

What is the output ?

Q1

```
1 usage
@SpringBootApplication
public class SpringPollApplication {

    public static void main(String[] args) { SpringApplication.run(SpringPollApplication.class, args); }


    @Bean
    public String getMessage1(){
        System.out.println("hey from message1");
        return "1";
    }
}
```

There is only one probability: hey from message1

Why this happened ?? because Bean will add this method to the context and the context will add this method to the spring Container

And then will print the message1 .

Q2

```java
1 usage
@SpringBootApplication
public class SpringPollApplication {

    public static void main(String[] args) { SpringApplication.run(SpringPollApplication.class, args); }


    @Bean
    @Qualifier("1")
    public String getMessage1(){
        System.out.println("hey from message1");
        return "1";
    }
    @Bean
    public String getMessage2(@Qualifier("1") String data ){
        System.out.println("hey from message2");
        return data ;
    }
```

The flow of this program will be as follows:
first the spring see and execute getMessage1 then print the Sout in the console (hey from message 1) after that the getMessage2 will call the Qualifier (“1”) annotation and then inject it to the method getMessage2 parameter and print the Sout method in the getMessage2 (hey from message2)
It can’t be more than 1 probability because getMessage2 depend on getMessage1 by using Qualifier (“1”).

Q3

```java
@Bean
@Qualifier("1")
public String getMessage1(){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

3 probabilities:

First: execute getMeesage1 then getMessage3 then getMessage2

second: execute getMeesage3 then getMessage2 then getMessage1

third: execute getMeesage3 then getMessage1 then getMessage2 or vice versa 3 and 1

Q4

```java
@Bean
@Qualifier("1")
public String getMessage1(){
    System.out.println("hey from message1");
    return "1";
}

@Bean
@Qualifier("2")
public String getMessage2(@Qualifier("3") String data ){
    System.out.println("hey from message2");
    return data;
}

@Bean
@Qualifier("3")
public String getMessage3(){
    System.out.println("hey from message3");
    return "3" ;
}
```

```java
@Component
public class MainController {

    1 usage
    String data;

    public MainController(@Qualifier("1") String data){
        this.data=data;
        System.out.println("hey from Main controller");
    }

}
```

5 probabilities:

First: execute getMeesage1 then getMessage3 then getMessage2 then mainController.

second: execute getMeesage3 then getMessage1 then getMessage2 then mainController.

third: execute getMeesage1 then getMessage3 then mainController then getMessage2.

Four: execute getMeesage1 then mainController getMessage3 then getMessage2.

Five: second: execute getMeesage3 then getMessage2 then getMessage1 then mainController.

The idea here is, it should execute the getMessage3 or MainController before the Other Methods.

Q5

```java

    @Bean
    @Qualifier("1")
    public String getMessage1(MainController mainController){
        System.out.println("hey from message1");
        return "1";
    }

    @Bean
    @Qualifier("2")
    public String getMessage2(@Qualifier("3") String data ){
        System.out.println("hey from message2");
        return data;
    }

    @Bean
    @Qualifier("3")
    public String getMessage3(){
        System.out.println("hey from message3");
        return "3" ;
    }
```

```java
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

1 usage
@Component
public class MainController {

    1 usage
    String data;

    public MainController(@Qualifier("2") String data){
        this.data=data;
        System.out.println("hey from Main controller");
    }

}
```

1 probability:

execute getMeesage3 then getMessage2 then mainController after that getMessage1 > the idea here is about dependency .