Exercise (Spring Container)

Q1:

Output:

hey from message1

Explanation:

When the Spring Boot application runs, it will print "hey from message1" to the console and register the string "1" in the container, because there is only one @Bean method in the application.

Q2:

Output:

hey from message1

hey from message2

Explanation:

When the Spring Boot application runs, the first method (doesn't depend on anything else) runs and registers the return value in the container, and the second method runs afterward using that registered value from the first method.

Q3:

Output: 1/ hey from message1

hey from message3

hey from message2

1/ When the Spring Boot application runs, the first method (getMessage1()) runs (it doesn't depend on anything else) and registers the return value "1" in the container, and the third method (getMessage3()) runs afterward, registering the return value "3". Then, the second method (getMessage2()) runs using the registered value "3" from the third method.

2/ hey from message3

hey from message2

hey from message1

2/ When the Spring Boot application runs, the third method (getMessage3()) runs first (it doesn't depend on anything else) and registers the return value "3" in the container. Then, the second method (getMessage2()) runs, using the registered value "3" from the third method. Finally, the first method (getMessage1()) runs afterward and registers the return value "1" in the container.

3/ hey from message3

hey from message1

hey from message2

3/ When the Spring Boot application runs, the third method (getMessage3()) runs first (it doesn't depend on anything else) and registers the return value "3" in the container. Then, the first method (getMessage1())(it doesn't depend on anything else also) runs afterward and registers the return value "1" in the container. Finally, the second method (getMessage2()) runs using the registered value "3" from the third method.

Q4:

Output:

1/ hey from message1

hey from Main Controller

hey from message3

hey from message2

Scenario 1:

- The first method (getMessage1()) runs (it doesn't depend on anything else) and -- registers the return value "1" in the container. This prints hey from message1.

- The **MainController** constructor is executed next, and since @Qualifier("1") is used, Spring injects the value "1" into the data variable. This prints hey from Main Controller.

- The third method (getMessage3()) runs afterward and registers the value "3" in the container, printing hey from message3.

- Finally, the second method (getMessage2()) runs, using the registered value "3" from getMessage3() and printing hey from message2.

2/ hey from message3

hey from message2

hey from message1

hey from Main Controller

Scenario 2:

The third method (getMessage3()) runs first (it doesn't depend on anything else) and registers the value "3" in the container, printing hey from message3.

The second method (getMessage2()) runs next, using the "3" bean from getMessage3() and printing hey from message2.

The first method (getMessage1()) runs afterward, printing hey from message1 and registering "1" in the container.

Finally, the **MainController** constructor is executed, and the value "1" is injected into the data variable, printing hey from Main Controller.

3/hey from message1

hey from message3

hey from message2

hey from Main Controller

Scenario 3:

The first method (getMessage1()) runs first, printing hey from message1 and registering "1" in the container.

The third method (getMessage3()) runs next and registers "3" in the container, printing hey from message3.

The second method (getMessage2()) runs afterward, using the "3" bean from getMessage3() and printing hey from message2.

Finally, the **MainController** constructor is executed, and the value "1" is injected into the data variable, printing hey from Main Controller.

4/hey from message3

hey from message1

hey from message2

hey from Main Controller

Scenario 4:

The third method (getMessage3()) runs first, printing hey from message3 and registering "3" in the container.

The first method (getMessage1()) runs next, printing hey from message1 and registering "1" in the container.

The second method (getMessage2()) runs afterward, using the "3" bean from getMessage3() and printing hey from message2.

Finally, the MainController constructor is executed, and the value "1" is injected into the data variable, printing hey from Main Controller.

5/hey from message3

hey from message1

hey from Main Controller

hey from message2

Scenario 5:

The third method (getMessage3()) runs first, printing hey from message3 and registering "3" in the container.

The first method (getMessage1()) runs next, printing hey from message1 and registering "1" in the container.

The **MainController** constructor is executed next, and the value "1" is injected into the data variable, printing hey from Main Controller.

Finally, the second method (getMessage2()) runs, using the "3" bean from getMessage3() and printing hey from message2

Q5:

Output: 1/ hey from message3

hey from message2

hey from Main Controller

hey from message1

Explanation:

getMessage3() runs first (it doesn’t depend on anything else), prints hey from message3, and registers "3" in the container.

getMessage2() runs next, since it depends on @Qualifier("3") (the value from getMessage3()). It prints hey from message2 and registers "3" in the container.

The **MainController** constructor runs next because getMessage2() is completed, and it injects the value from "2" into the controller. This prints hey from Main Controller.

Finally, getMessage1(MainController mainController) runs, now that the MainController is available. It prints hey from message1 and registers "1" in the container.

Name: Sara Mohammed