Spring-beanScope-demo

This demo explain the concept of **Singleton.**

Main Function:

```
//retrieve bean from spring container
Manager manager = context.getBean("myTeam", Manager.class);

Manager myManager = context.getBean("myTeam", Manager.class);

// check if they are the same

boolean result = (manager == myManager);

// print out the result
System.out.println("\nPoint to the same object: "+ result);

System.out.println("\nMemory location for theCoach: "+ manager);

System.out.println("\nMemory location for alphaCoach: "+ myManager);
```

Console output:

Point to the same object: true

Memory location for manager: com.branch.springwork.MarkettingTeam@4e7dc304

Memory location for myManager: com.branch.springwork.MarkettingTeam@4e7dc304

Explanation:

In main function we have defined two Manager object but bot pointing to one memory location. That means there is only one instance has been created for both the manager object. This concept is called Singleton.

Lets examine the beanScope-applicationContext.xml

```
 <bean id="myTeam"
      class="com.branch.springwork.MarkettingTeam">
```

This line is same as

```
 <bean id="myTeam"
      class="com.branch.springwork.MarkettingTeam"
scope="singleton">

scope="singleton" is default.
```

However, when you change **scope to prototype** then lets see the output:

```
<bean id="myTeam"
      class="com.branch.springwork.MarkettingTeam"
scope="prototype">
```

Console output:

Point to the same object: **false**

Memory location for theCoach: com.branch.springwork.MarkettingTeam@**6895a785**

Memory location for alphaCoach: com.branch.springwork.MarkettingTeam@**184f6be2**

Console output shows that two objects are at different memory location.

Conclusion:

If Spring Bean scope is singleton => Create a single shared instance of the bean. Default scope.

If Spring Bean scope is prototype => Creates a new bean instance for each container request.