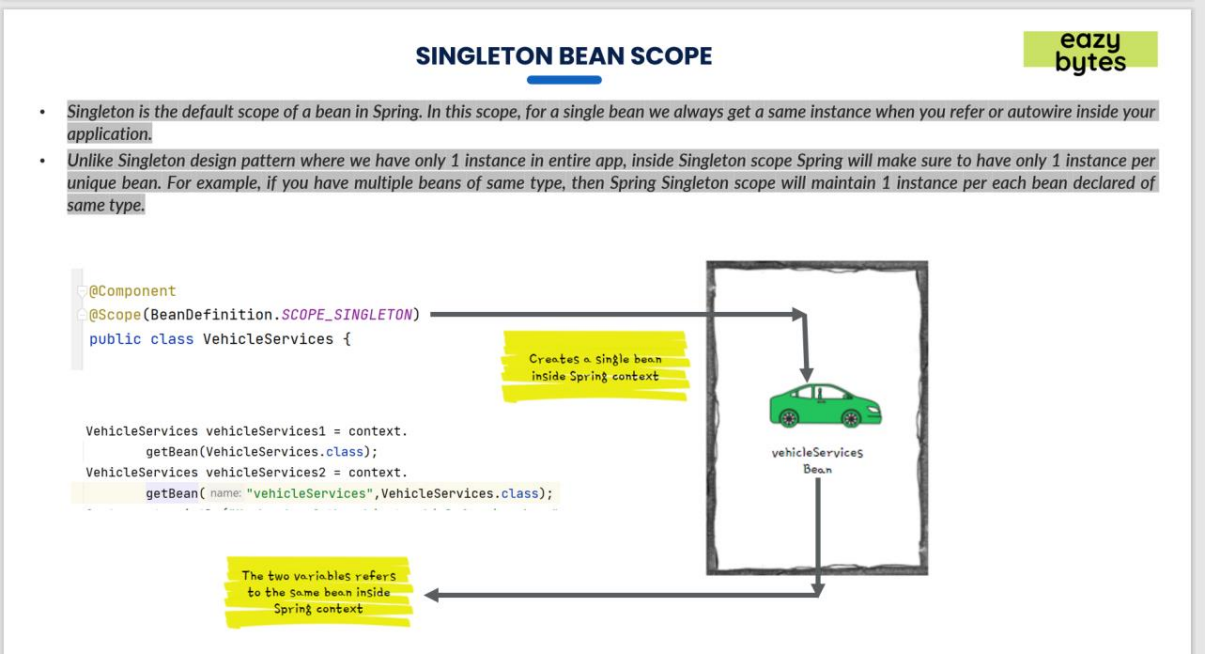
40.Introduction to Beans scope inside Spring

Bean scope is a concept inside spring which will define how a bean has to be created and maintained by spring IOC container, whether it needs to be maintained forever inside an application till the restart happens, or whether it needs to create a new bean every time a reference is asked.

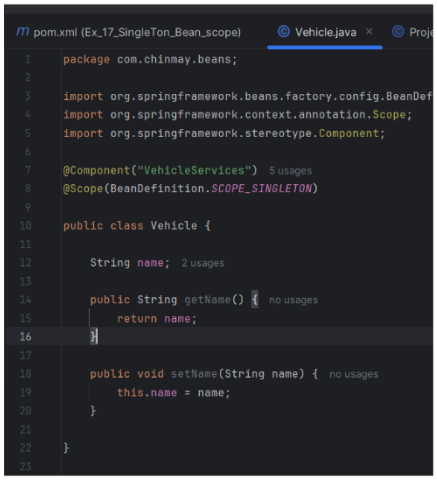
So all these are controlled with the help of bean scopes inside spring

Request, Session and Application are web application scopes.

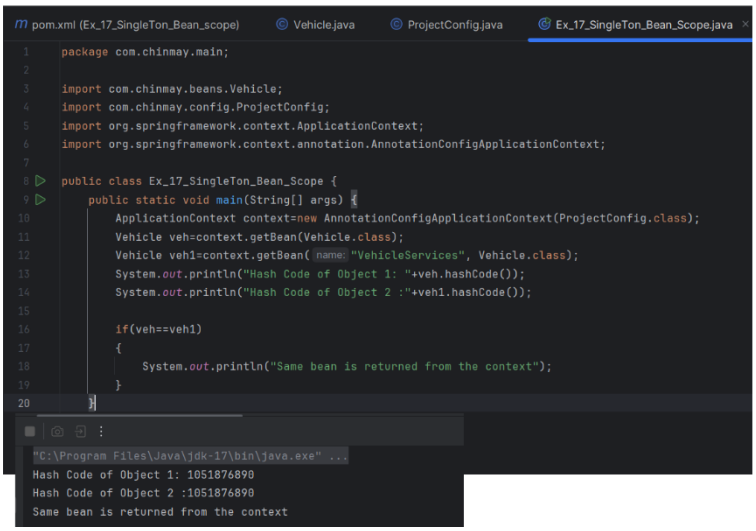
41.Deep dive on Singleton Bean Scope:



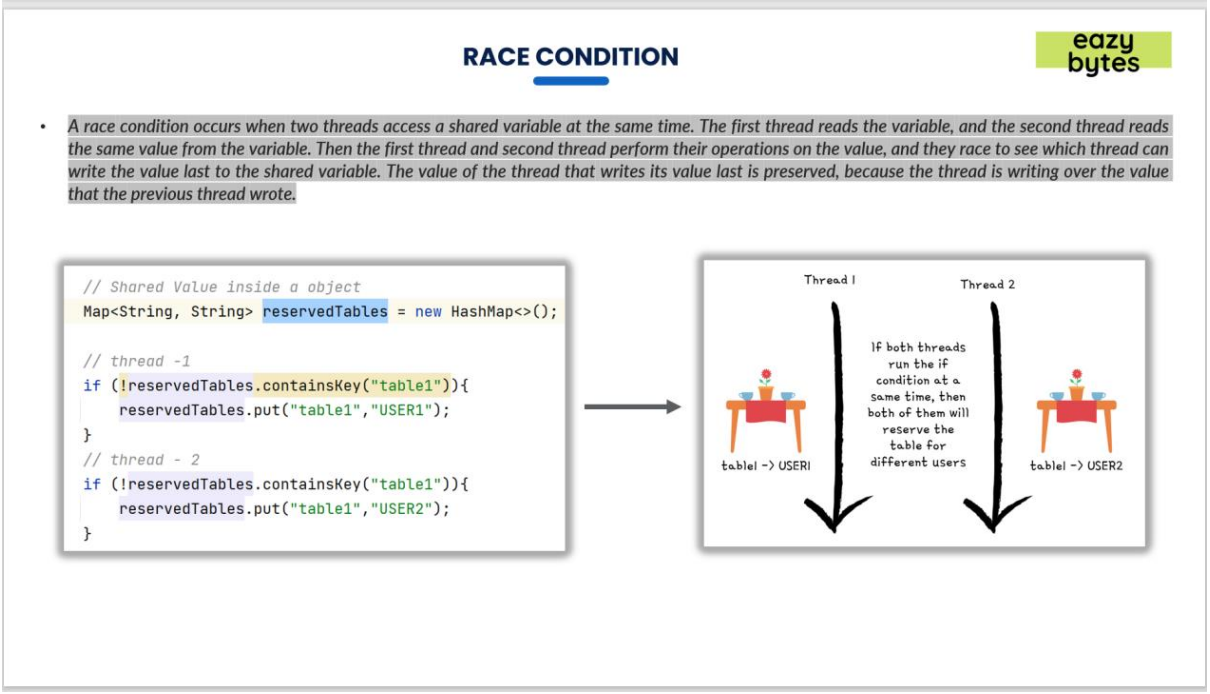
By default if we don't specify any scope Single ton will be considered as default scope.



We could see here as the Scope is Single ton Same instance of the bean has been returned both times when we have fetched the Bean from the context.

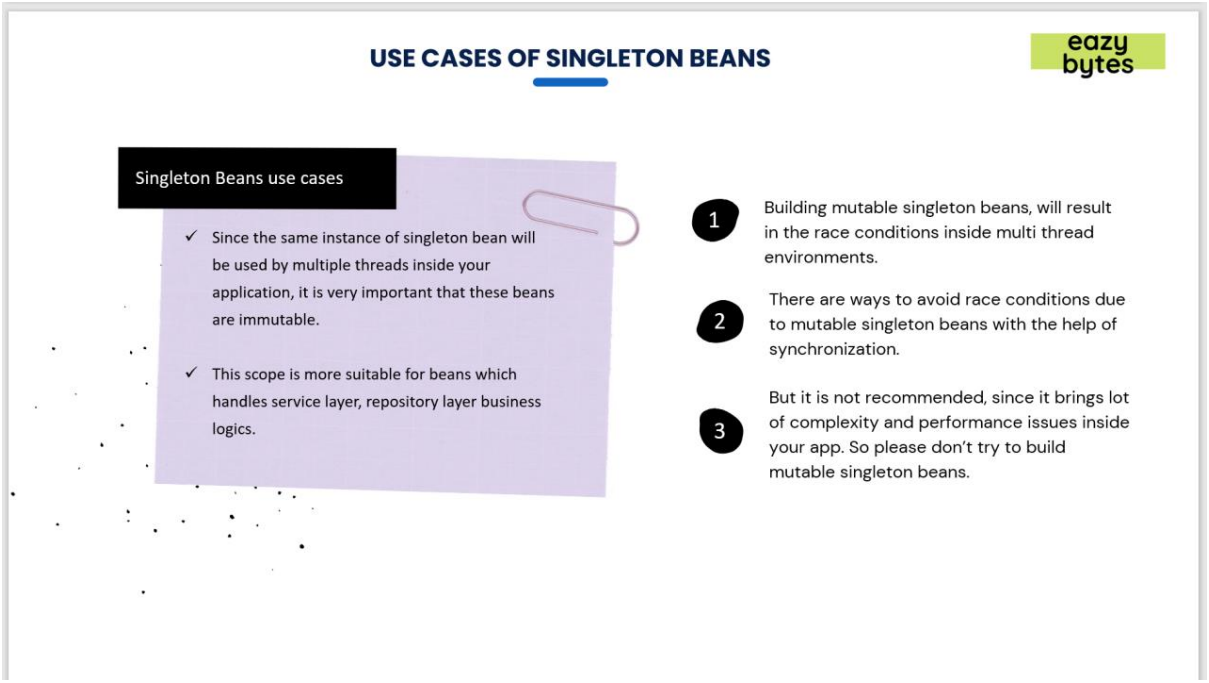


42.Race Condition:



Since the same instance of the bean will be returned every time when we try to fetch the bean .There might be chance that two different beans will try to update the bean details and same time and overwriting of the data takes place.

43.Use cases of Singleton bean scope



44.Eager and Lazy instantiation of Singleton scope.

We can control when the Singleton beans will be created by making the bean as Lazy instantiation. So that bean will not be created until there is need for that.

Most of the times we will go with the Eager instantiation.

But we change the bean instantiation to Lazy instantiation when bean is used very remotely or very rarely.

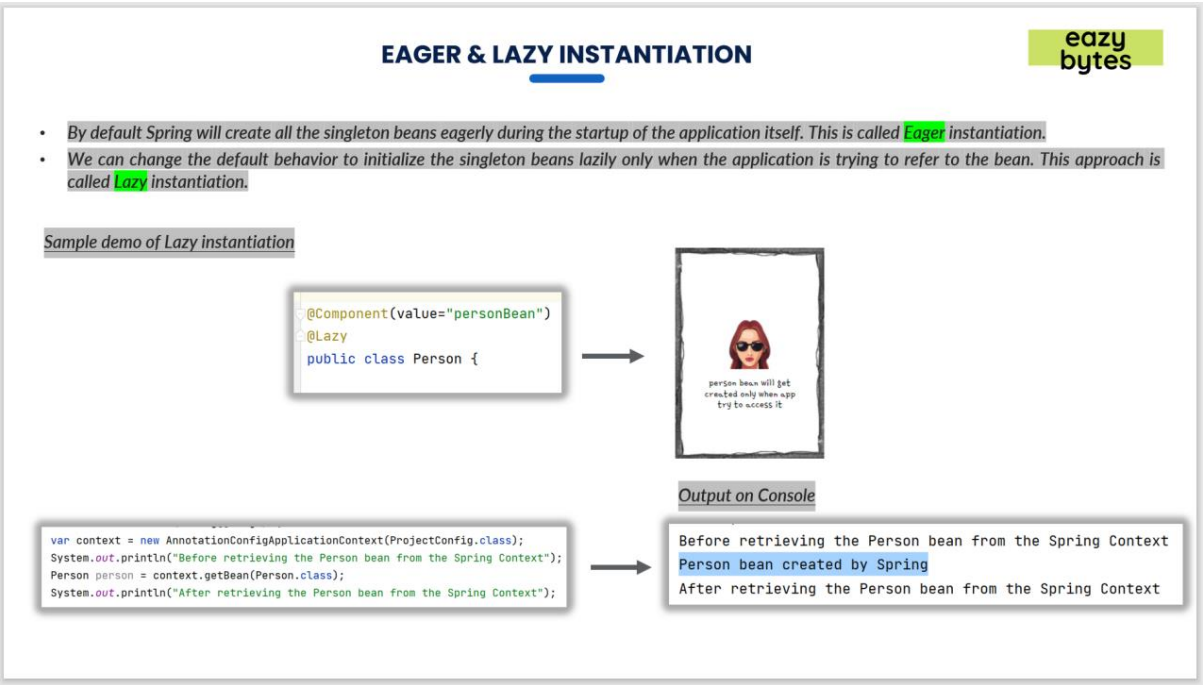
Ex: Deletion of Account

There might be chance these beans might not be used so creating them during the startup will not have much use and increase the number of the beans in the context.

Issue with Lazy instantiation:

In case of Lazy instantiation an exception will be raised during the run time if there is an issue while creating the bean.

Whereas if go with Eager instantiation the server with not startup if there is an issue while creating the bean.

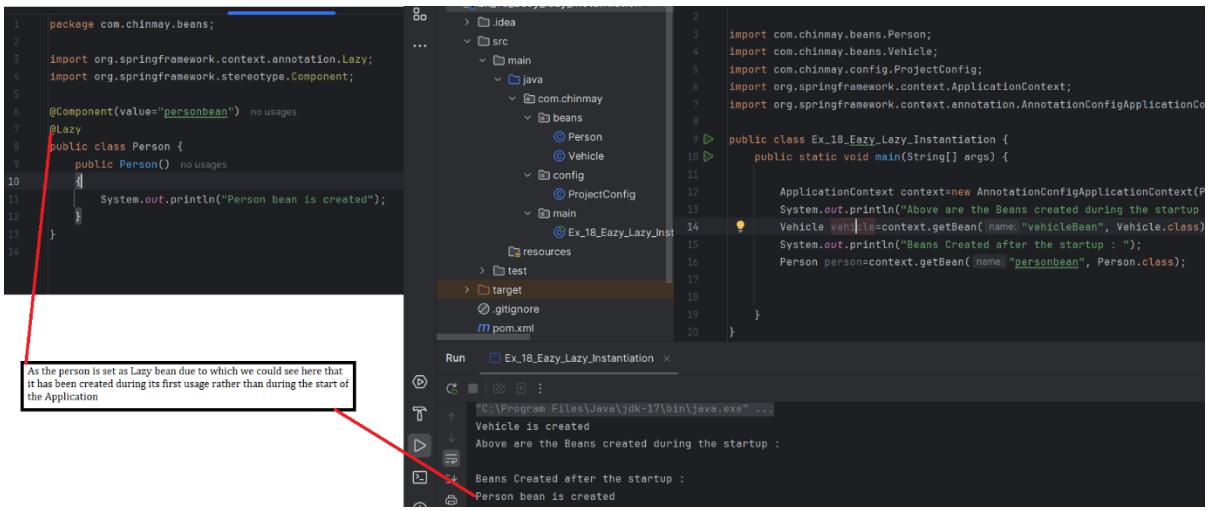


Bean will get created when it is referred for the first time rather than during the startup of Application.

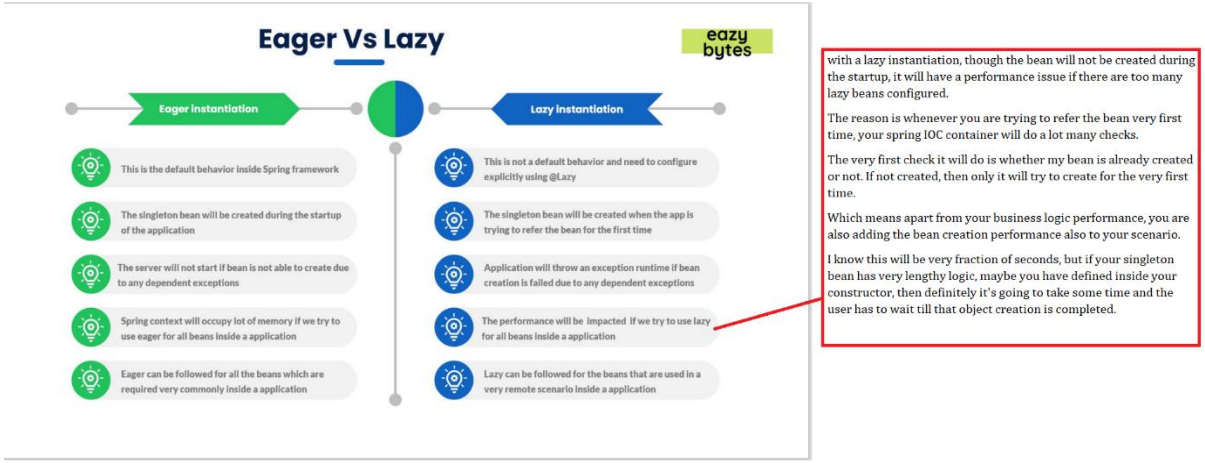
45.Demo of Eager and lazy instantiation of Singleton bean

Ex:18

Ex_18_Eazy_Lazy_Instantiation



46.Eager Initialization vs Lazy Initilization

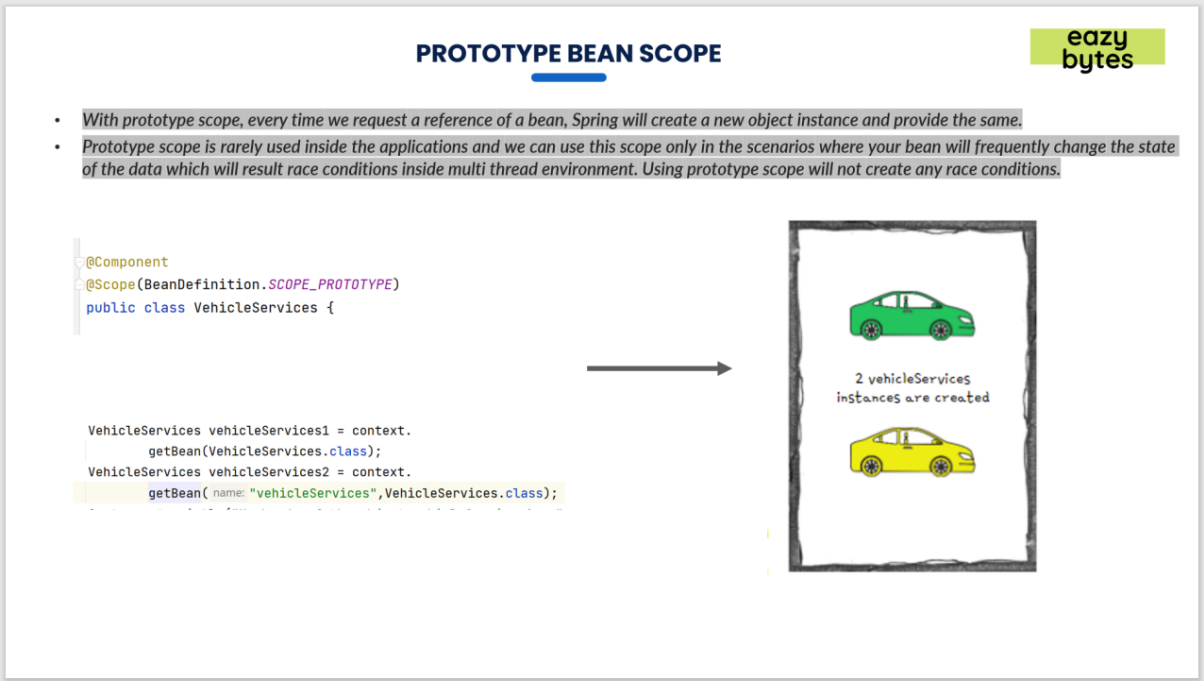


47: Deep dive of prototype Bean scope:

If there is an dependency of Prototype Scope bean inside the Singleton Scope bean how many we call it will return the same instance of bean even though it is of Proto type scope as we are accessing it from Singleton Scope.

```java
@Component("vehicleBean")
public class Vehicle {

    private String name="Honda";
    private final VehicleService vehicleService;

    @Autowired
    public Vehicle(VehicleService vehicleService){
        this.vehicleService = vehicleService;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public VehicleService getVehicleServices() {
        return vehicleService;
    }
```

Here the Vehicle is of Single ton scope where as the vehicle is dependent on VehicleServiceBean which is of prototype Scope .
Even though it is of Prototype scope as it is used in Single ton scope bean every time we request for VehicleServiceBean same instance of the copy will be returned.



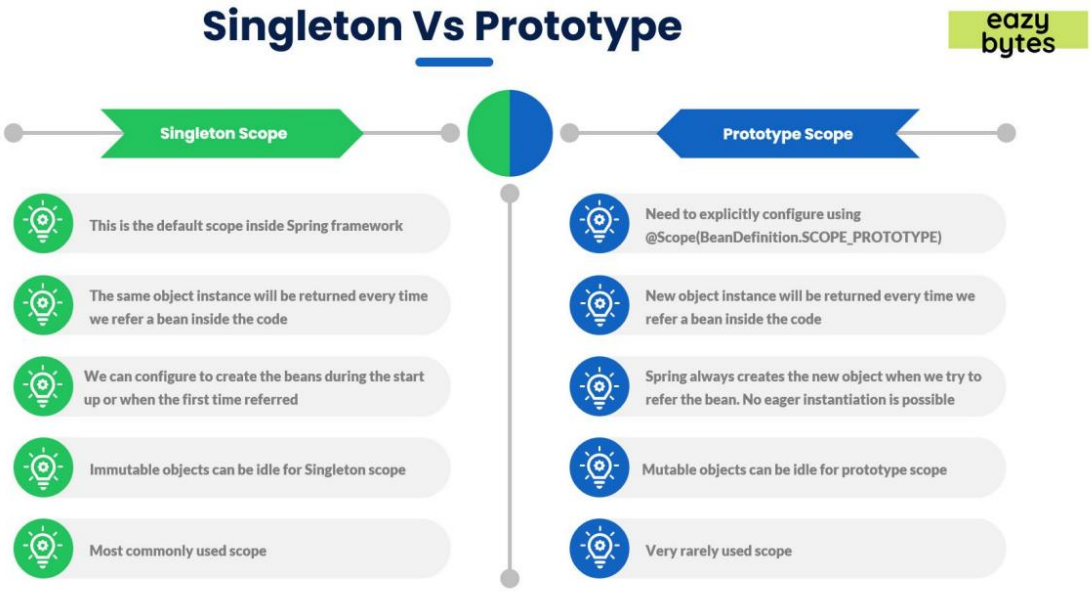

```java
package com.chinmay.beans;

import org.springframework.beans.factory.config.BeanDefinition;
import org.springframework.context.annotation.Scope;
import org.springframework.stereotype.Component;

@Component("vehicleBean")  5 usages
@Scope(BeanDefinition.SCOPE_PROTOTYPE)
public class Vehicle {

    public Vehicle() { System.out.println("Vehicle is created"); }
}
```

```
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
Person bean is created
Above are the Beans created during the startup :

Vehicle is created
Vehicle is created
Beans created are of prototype Scope

Process finished with exit code 0
```

pom.xml (Ex_19_Proto_Type_Scope) | Ex_19_Proto_Type_Scope.java | Vehicle.java | ProjectConfig.java

```java
package com.chinmay.main;

import ...

public class Ex_19_Proto_Type_Scope {
    public static void main(String[] args) {

        ApplicationContext context=new AnnotationConfigApplicationContext(ProjectConfig.class);
        System.out.println("Above are the Beans created during the startup : \n");
        Vehicle vehicle=context.getBean( name: "vehicleBean", Vehicle.class);
        Vehicle vehicle1=context.getBean( name: "vehicleBean", Vehicle.class);

        if(vehicle1==vehicle)
        {
            System.out.println("Beans created are of singleton Scope");
        }
        else {
            System.out.println("Beans created are of prototype Scope");
        }
    }
```

49.Singleton vs Prototype Scope



In Prototype Scope we don’t have lazy or Eager initialization.