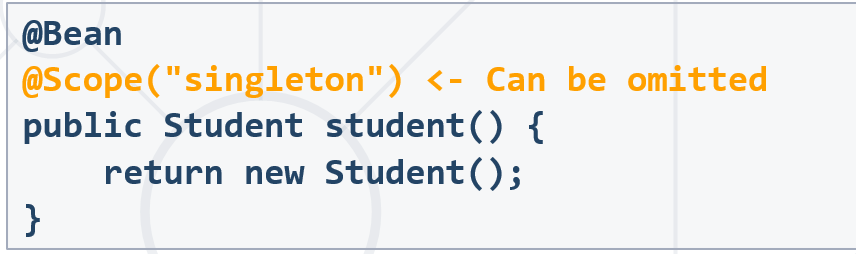
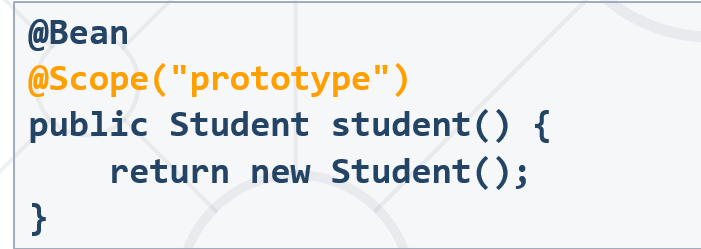
1. **Beans scopes**

* **Singleton –** Returns the same object



* **Prototype –** Will return a different instance every time it is requested from the container



* **RequestScope –** A different bean instance being returned for each request



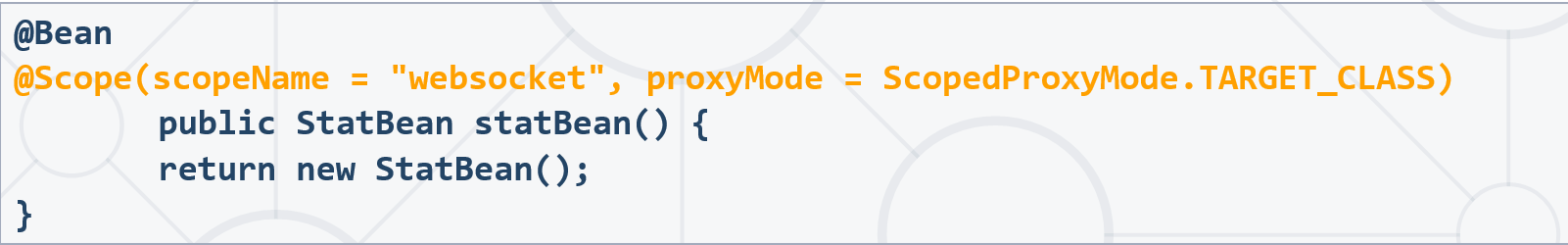
* **SessionScope –** Will creates the bean instance for the lifecycle of a session



* **ApplicationScope –** Will creates the bean instance for the lifecycle of a ServletContext

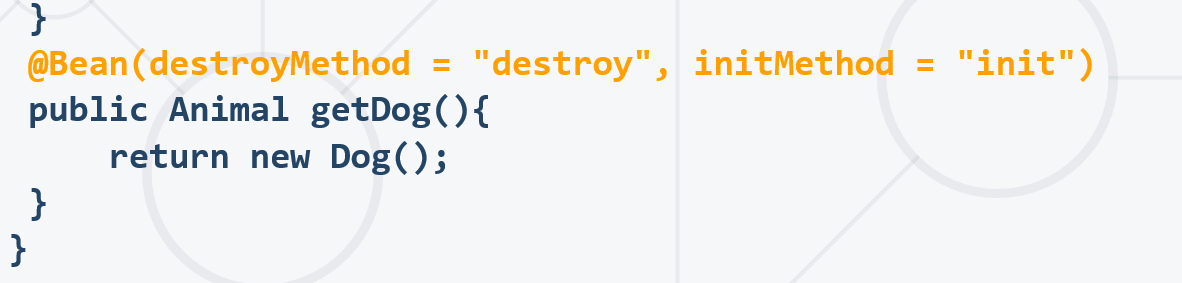


* **WebSocket Scope –** returns the same instance of the bean whenever that bean is accessed during the entire WebSocket session



1. **Bean Lifecycle**

* Ще извика метод “destroy” и метод „init” при инициализация и затваряне на приложението



* **@PostConstruct –** изпълнява се след инициализация



* **@PreDestroy –** изпълнява се преди да се затвори App-a
* **BeanNameAware interface –** имплементира се и позволява сетване на пропърти „name = “” “ в @Bean анотацията
* **InitializingBean interface –** ще стартира afterPropertiesSet() след като се сетнат всички настройки на Bean-**a**
* **DisposableBean interface –** ще стартира destroy() след като Spring отсвободи Bean-a

@Transactional – тази анотация позволява да се изпълнят всички .save methods задъжлително. Дори да са LAZY пак ще ги запише. Може да се анотира клас или метод.