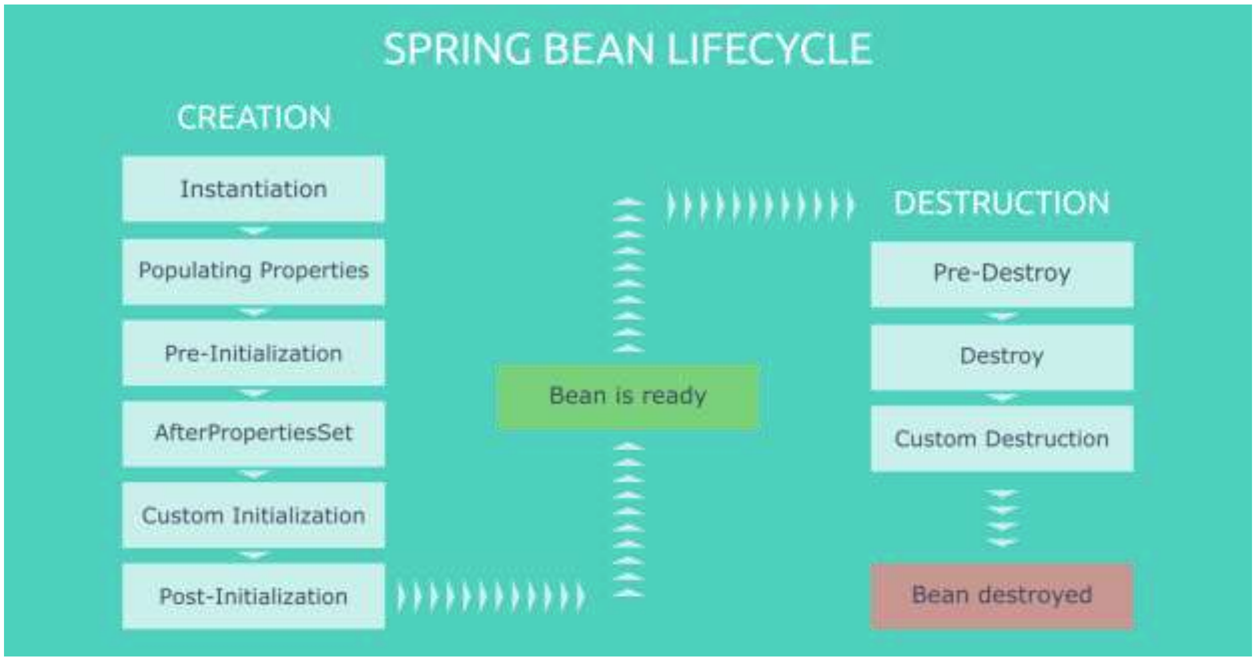
# The Spring Bean Lifecycle

When we look into the lifecycle of Spring beans, we can see numerous phases starting from the object instantiation up to their destruction.

To keep it simple, **we group them into creation and destruction phases:**



## Bean Creation Phases

- **Instantiation:** This is where everything starts for a bean. Spring instantiates bean objects just like we would manually create a Java object instance.

- **Populating Properties:** After instantiating objects, Spring scans the beans that implement Aware interfaces and starts setting relevant properties.

- **Pre-Initialization:** Spring's BeanPostProcessors get into action in this phase. The postProcessBeforeInitialization() methods do their job. Also, @PostConstruct annotated methods run right after them.

- **AfterPropertiesSet:** Spring executes the afterPropertiesSet() methods of the beans which implement InitializingBean.

- **Custom Initialization:** Spring triggers the initialization methods that we defined in the initMethod attribute of our @Beanannotations.

- **Post-Initialization:** Spring's BeanPostProcessors are in action for the second time. This phase triggers the postProcessAfterInitialization() methods.

**Bean Destruction Phases**

- **Pre-Destroy:** Spring triggers@PreDestroy annotated methods in this phase.

- **Destroy:** Spring executes the destroy() methods of DisposableBean implementations.

- **Custom Destruction:** We can define custom destruction hooks with the destroyMethod attribute in the @Bean annotation and Spring runs them in the last phase.

## Singleton and Prototype Bean Scopes in Java Spring

**Bean Scopes** refers to the lifecycle of Bean that means when the object of Bean will be instantiated, how long does that object live, and how many objects will be created for that bean throughout. Basically, it controls the instance creation of the bean and it is managed by the spring container.

## Bean Scopes in Spring

The spring framework provides five scopes for a bean. We can use three of them only in the context of web-aware **Spring ApplicationContext** and the rest of the two is available for both **IoC container and Spring-MVC**

**container**. The following are the different scopes provided for a bean:

1. **Singleton:** Only one instance will be created for a single bean definition per Spring IoC container and the same object will be shared for each request made for that bean.

2. **Prototype:** A new instance will be created for a single bean definition every time a request is made for that bean.

3. **Request:** A new instance will be created for a single bean definition every time an HTTP request is made for that bean. But Only valid in the context of a web-aware Spring ApplicationContext.

4. **Session:** Scopes a single bean definition to the lifecycle of an HTTP Session. But Only valid in the context of a web-aware Spring ApplicationContext.

5. **Global-Session:** Scopes a single bean definition to the lifecycle of a global HTTP Session. It is also only valid in the context of a web-aware Spring ApplicationContext.

### Singleton Scope:

If the scope is a singleton, then only one instance of that bean will be instantiated per Spring IoC container and the same instance will be shared for each request. That is when the scope of a bean is declared singleton, then whenever a new request is made for that bean, spring IOC container first checks whether an instance of that bean is already created or not. If it is already created, then the IOC container returns the same instance otherwise it creates a new instance of that bean only at the first request. By default, the scope of a bean is a singleton.

## Prototype Scope:

If the scope is declared **prototype**, then spring IOC container will create a new instance of that bean every time a request is made for that specific bean. A request can be made to the bean instance either programmatically using **getBean()** method or by XML for Dependency Injection of secondary type. Generally, we use the prototype scope for all beans that are stateful, while the singleton scope is used for the stateless beans.

## Difference betweeen Singleton and Prototype

| **Singleton** | **Prototype** |
|---|---|
| Only one instance is created for a single bean definition per Spring IoC container | A new instance is created for a single bean definition every time a request is made for that bean. |
| Same object is shared for each request made for that bean. i.e. The same object is returned each time it is injected. | For each new request a new instance is created. i.e. A new object is created each time it is injected. |
| By default scope of a bean is singleton. So we don't need to declare a been as singleton explicitly. | By default scope is not prototype so you have to decalre the scope of a been as prototype explicitly. |
| Singleton scope should be used for stateless beans. | While prototype scope is used for all beans that are stateful |