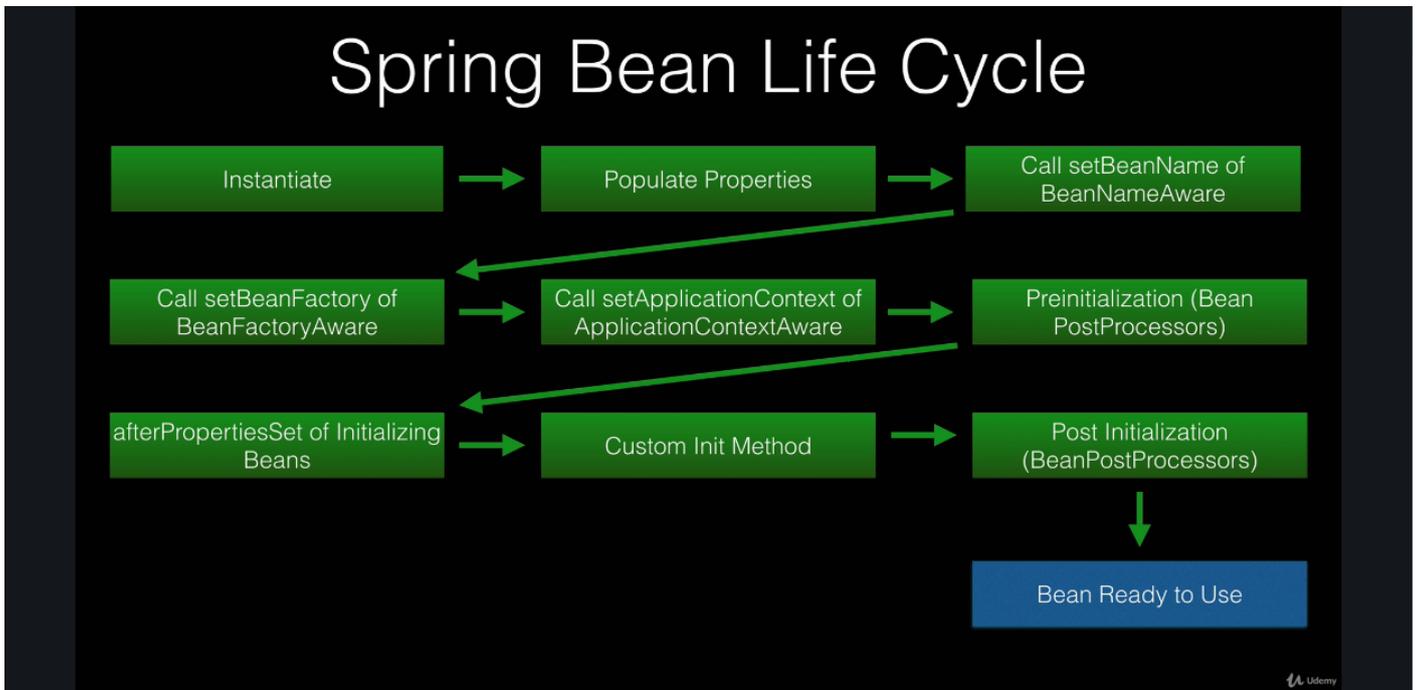


# Spring Bean Life Cycle



- When the class is created we can see there are 'Aware' interfaces that get initialized
  - BeanNameAware
  - BeanFactoryAware
  - ApplicationContextAware

## Shutdown

**Container Shutdown -> Disposable Bean's destroy() -> Call custom destroy method -> Terminated**

## Callback Interfaces

- Spring has two interfaces you can implement for call back events
- *InitializingBean.afterPropertiesSet()*
  - called after properties are set
- *DisposableBean.destroy()*
  - called during bean destruction in shutdown

# Life Cycle Annotations

- Spring has two annotations you can use to hook into the bean life cycle
- *@PostConstruct*
  - called after the bean has been constructed but before its returned to the requesting object
- *@PreDestroy*
  - called just before the bean is destroyed by the container

## Bean Post Processors

- Gives you a means to tap into the Spring context life cycle and interact with beans as they are processed
- Implement interface *BeanPostProcessor*
- *postProcessBeforeInitialization*
  - called before bean initialization method
- *postProcessAfterInitialization*
  - called after bean initialization

Note: The guru admits in all his years he has never used these

## 'Aware' Interfaces

- Spring has over 14 Aware Interfaces
- These are used to access the Spring Framework infrastructure
- Rarely used by Devs, mostly used within the framework itself

Aware Interface	Description	Aware Interface	Description
<i>ApplicationContextAware</i>	Interface to be implemented by any object that wishes to be notified of the <i>ApplicationContext</i> that it runs in.	<i>LoadTimeWeaverAware</i>	Set the <i>LoadTimeWeaver</i> of this object containing <i>ApplicationContext</i> .
<i>ApplicationEventPublisherAware</i>	Set the <i>ApplicationEventPublisher</i> that this object runs in.	<i>MessageSourceAware</i>	Set the <i>MessageSource</i> that this object runs in.
<i>BeanClassLoaderAware</i>	Callback that supplies the bean class loader to a bean instance.	<i>NotificationPublisherAware</i>	Set the <i>NotificationPublisher</i> instance for current managed resource instance.
<i>BeanFactoryAware</i>	Callback that supplies the owning factory to a bean instance.	<i>PortletConfigAware</i>	Set the <i>PortletConfig</i> that this object runs in.
<i>BeanNameAware</i>	Set the name of the bean in the bean factory that created this bean.	<i>PortletContextAware</i>	Set the <i>PortletContext</i> that this object runs in.
<i>BootstrapContextAware</i>	Set the <i>BootstrapContext</i> that this object runs in.	<i>ResourceLoaderAware</i>	Set the <i>ResourceLoader</i> that this object runs in.
		<i>ServletConfigAware</i>	Set the <i>ServletConfig</i> that this object runs in.
		<i>ServletContextAware</i>	Set the <i>ServletContext</i> that this object runs in.

The more useful ones:

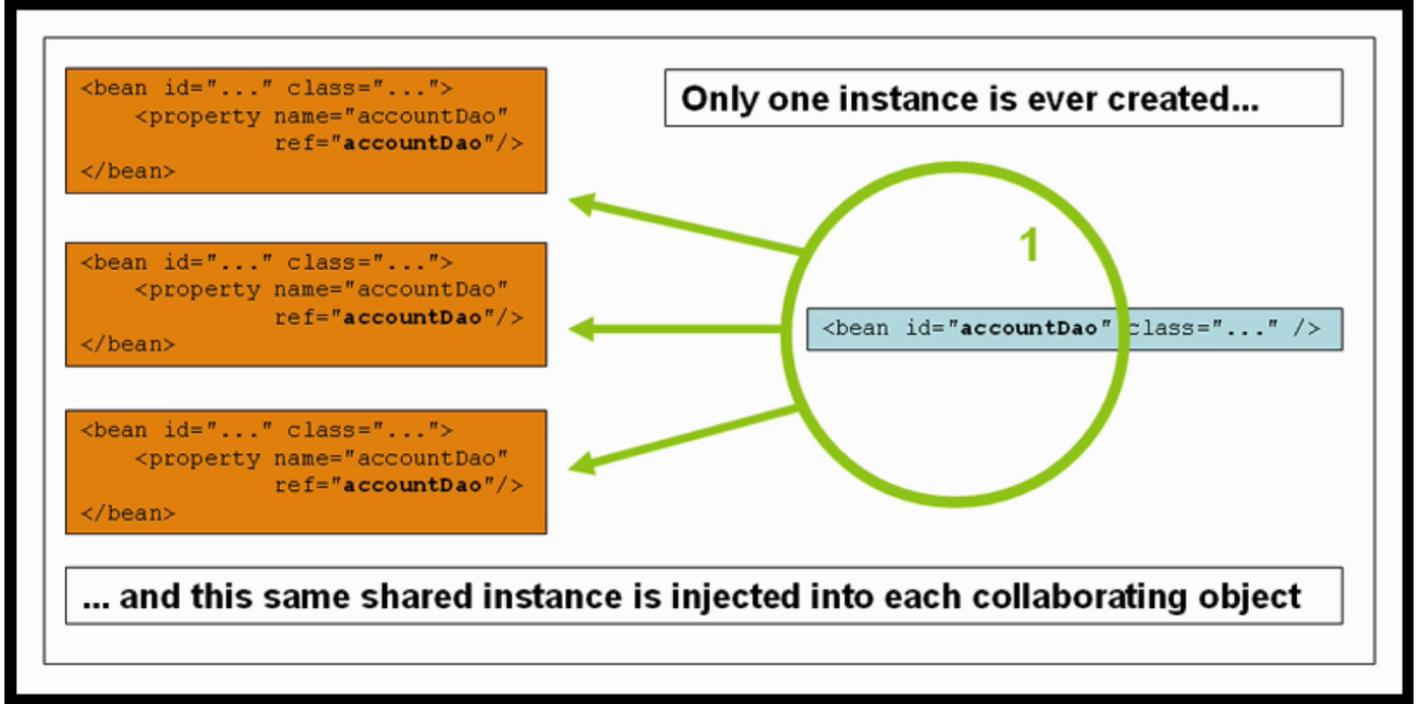
- ApplicationEventPublisherAware - used for creating custom events inside spring and set up event listeners
- BeanFactoryAware - If you need to handle a bean within a process

# Spring Bean Scopes

- Singleton (default) - Only one instance of the bean is created in the IoC container
- Prototype - A new instance is created each time the bean is requested
- Request - Single instance per http request.\*
- Session - Single instance per http session.\*
- Global-session - A single instance per global session. Typically only used in a Portlet context.\*
- Application - bean is scoped to the lifecycle of a ServletContext.\*
- WebSocket - Scopes a single bean definition to the lifecycle of a WebSocket.\*
- Custom scope - extensible, define your own "Scope" interface. See docs for details

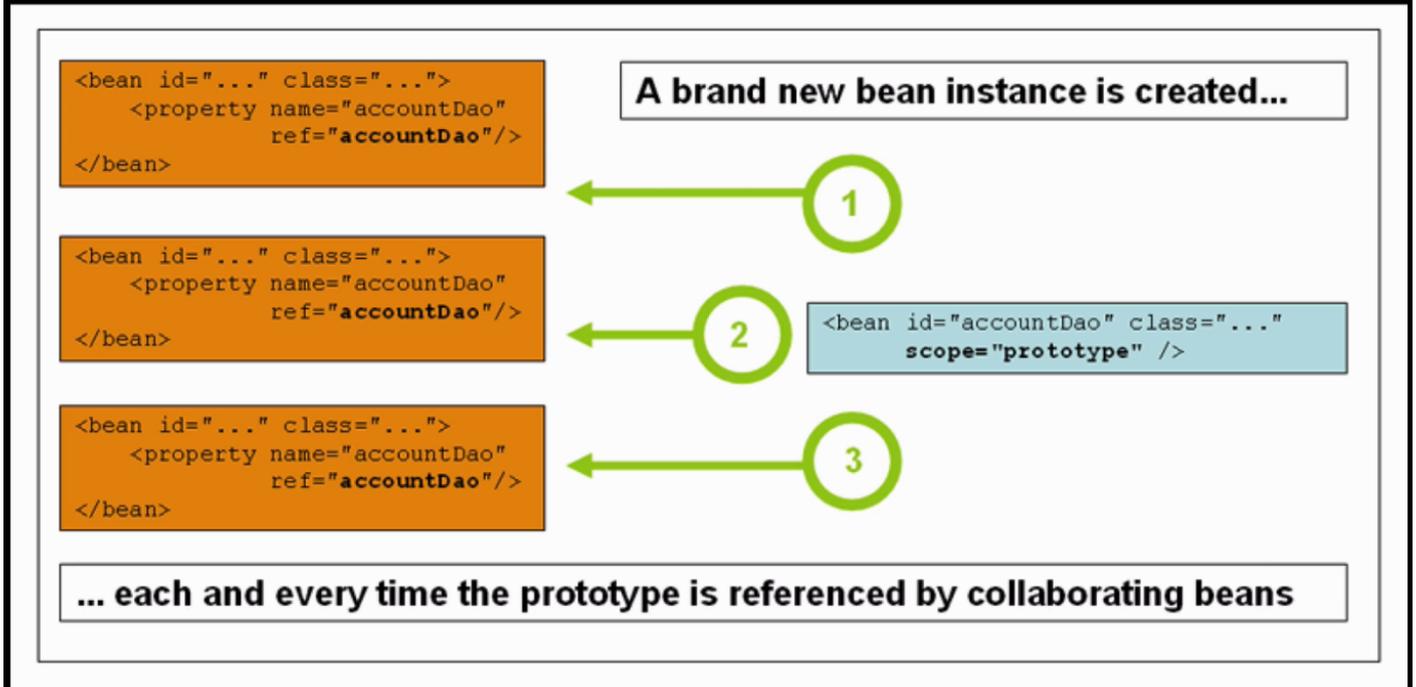
\*Only valid in the contex of a web-aware Spring ApplicationContext

## Singleton Scope



No declaration is needed for singleton scope.

# Prototype Scope



In Java configuration use the @Scope annotation. We see the xml configuration above.

Revision #1

Created 17 April 2022 00:37:02 by Elkip

Updated 17 April 2022 00:39:13 by Elkip