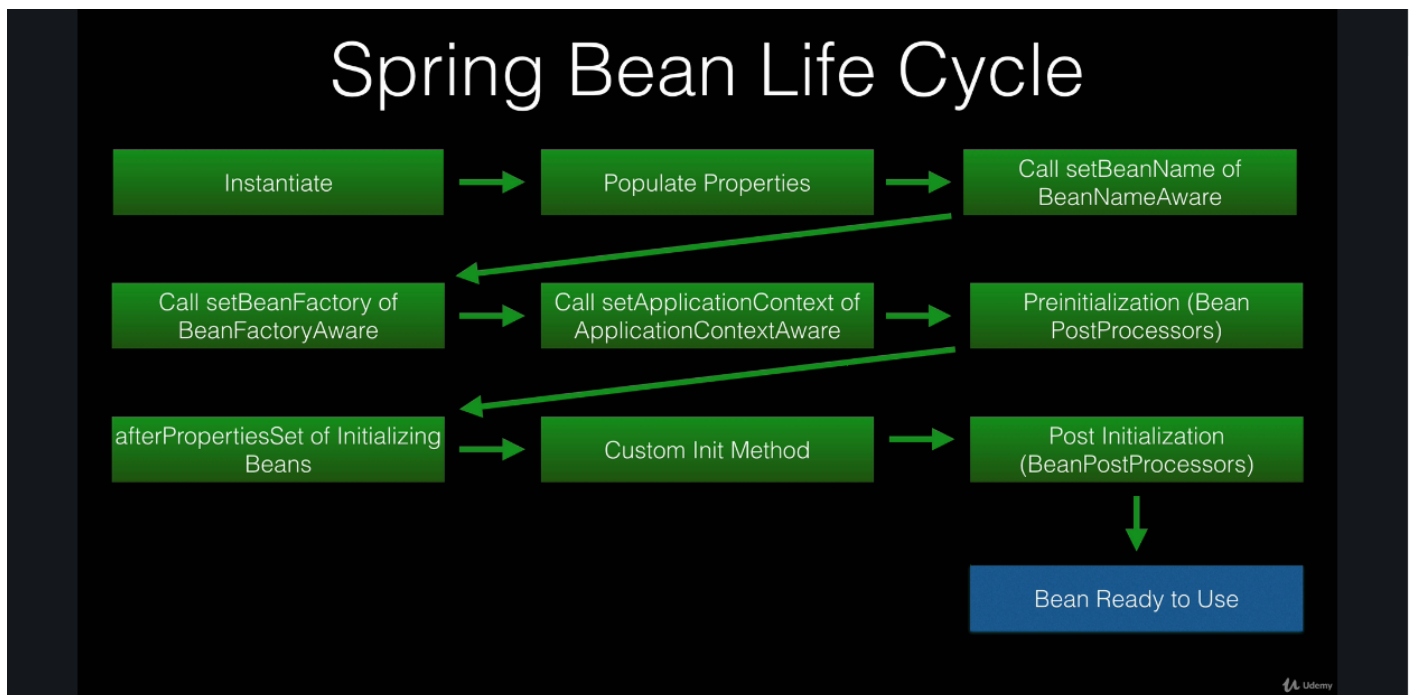


Spring Bean Life Cycle



- 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 accses the Spring Framework infrastructure
- Rarely used by Devs, mostly used within the framework itself

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

The more useful ones:

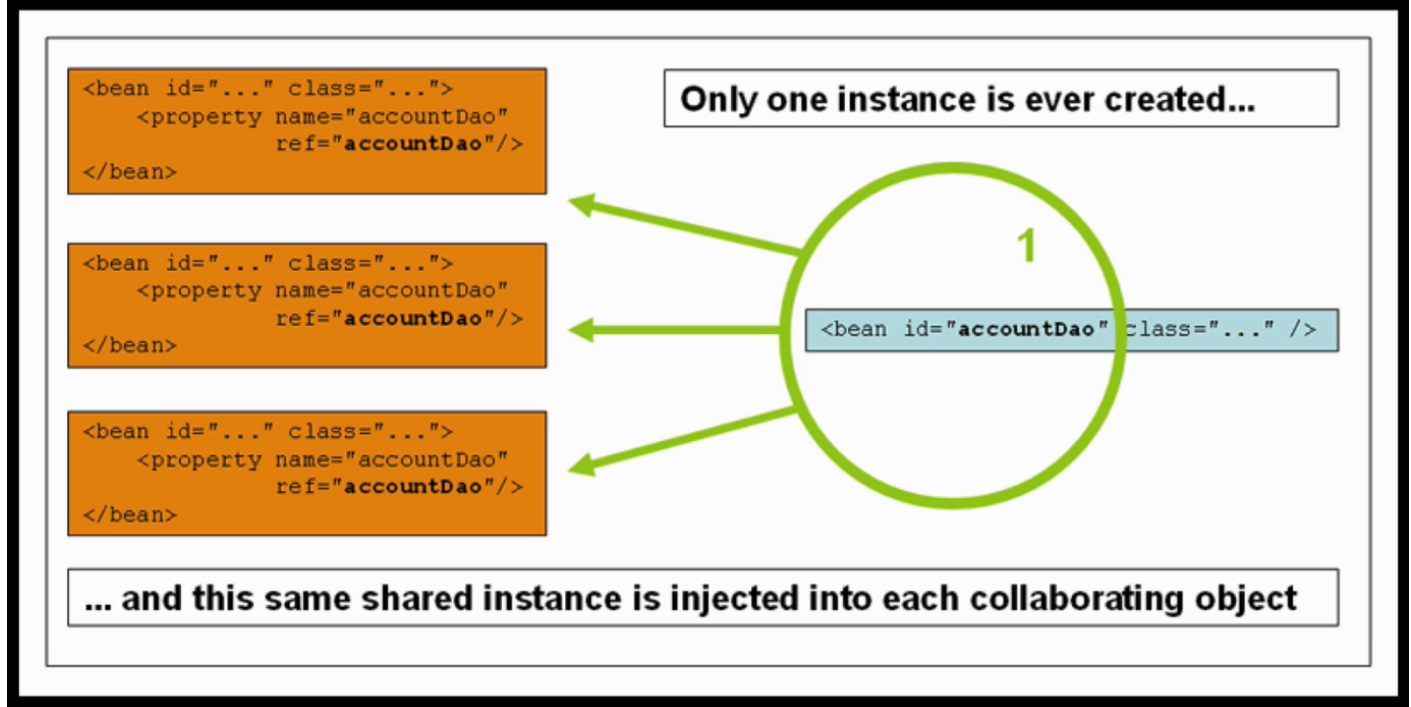
- ApplicationEventPublisherAware - used for creating custom events inside spring and set up event listeners
- BeanFactoryAware - If you need to handle a bean within a process

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

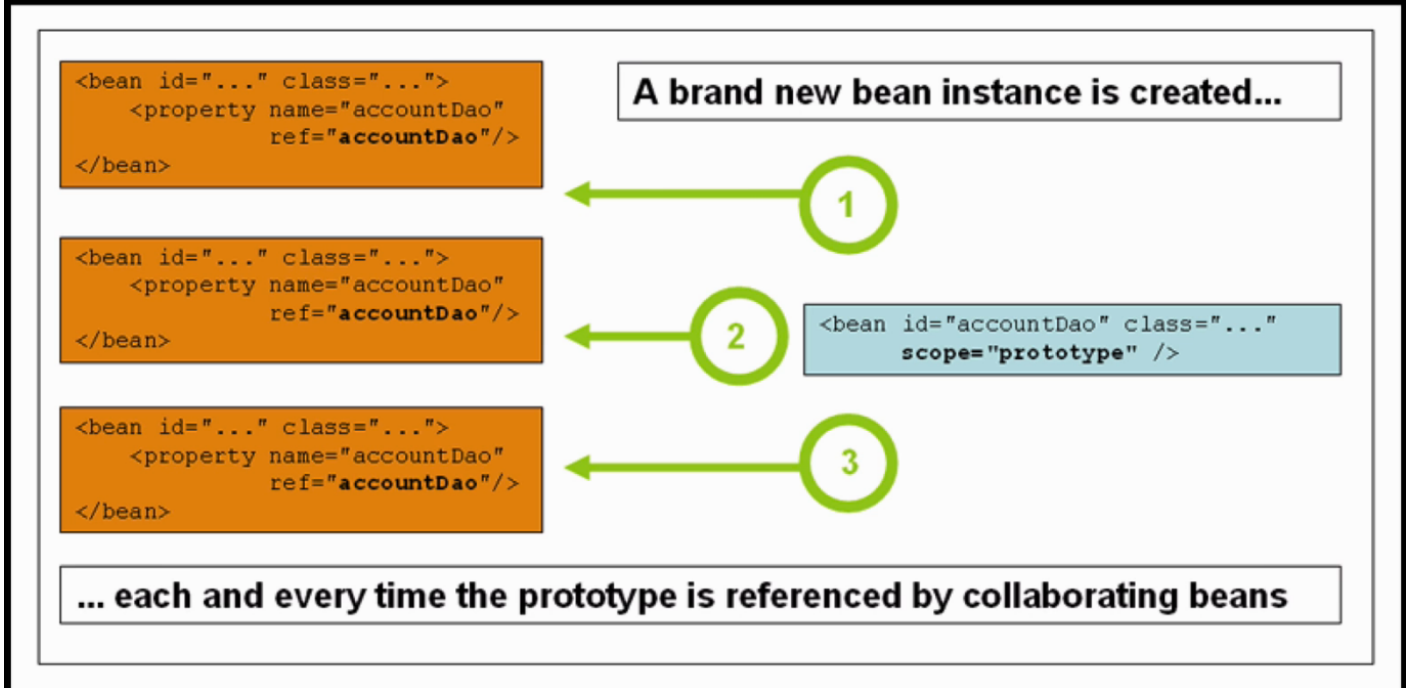
*Only valid in the context of a web-aware Spring ApplicationContext

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