**Lifecycle Methods: refresh()**

• **refresh()** is a startup method for the context objects.

• Base class for refreshable contexts is **AbstractRefreshableApplicationContext**.

• Calling its **refresh()** method creates a new internal bean factory instance.

• It loads or refreshes the persistent representation of the configuration, which might be from Java-based configuration, an XML file, a properties file, a relational database schema, or some other format.

Refresh context objesini ayağa kaldıran startup methodudur.

Base classı **AbstractRefreshableApplicationContext buradadır.**

Her seferinde refresh metodu çağırmak yeni bir new internal bean factory instance olusturur.

Load eder ilk basta ya da var olanı yeniler, constructorda ilk defa load ediyor ama set vs durumlarda var olanı yeniliyor.

• So anytime an **ApplicationContext** object is initialized or refreshed all beans are loaded, post-processor beans are detected and activated, singletons are pre-instantiated, and the **ApplicationContext** object is ready for use.

• When a context is refreshed its callback is also called.

• All post-initialization callback methods on initialized beans are called.

AC initialize edildiği zaman ya da refresh edildiği zaman bütün beanler load edilir, bunların post-processor beanları islenir, active edilir bütün singletonlar active edilir. AC o zaman hazır.

Ne zaman context refresh edilirse o zaman onun callback metodu çağırılır.

Bir AC refresh edilirse aynı zamanda Beanler üzerindeki post-initializationdeki bütün callback methodları cagırılır.

• If the configuration is passed to the constructors of **ApplicationContext** objects they scan all of the beans in configuration(s) and initialize all eagerly-loaded singletons making them ready for service.

• Subclasses of **ApplicationContext** such as **FileSystemXmlApplicationContext** and **ClassPathXmlApplicationContext** automatically refresh themselves in their constructors when they load bean definitions.

• Only after refresh it initializes and serves lazy-loaded beans upon request

Eğer constructors configurasyon gecersek bütün eaglery-loadları bize hazır ediyor ve zaten kendi icinde refresh ediyor.



• If the context object does not receive configuration through its constructor then it needs a refresh to make all beans ready when a new configuration object is passed.

.

• If a bean defined in a new configuration is asked for before refreshing the

context the container throws **IllegalStateException** with a message

**org.springframework.context.annotation.AnnotationConfig ApplicationContext@28ba21f3 has not been refreshed yet**

context kendisi dogrudan constructor almayıp da metotla register ediliyorsa bu durumda **IllegalStateException hatası veriyor ve henüz refresh etmediğimizi vurguluyor.**

• Some subclasses of **ApplicationContext**, in fact all subclasses of

**AbstractRefreshableApplicationContext** such as

**FileSystemXmlApplicationContext** and

**ClassPathXmlApplicationContext** allow multiple refresh calls.

• Spring calls them **hot** **refresh**.

• Calling **refresh()** method creates a new internal bean factory

instance with a possible new configuration making all beans registered

with the previous bean factory destroyed.

**AC : ApplicationContext**

AC’nin bazı alt sınıfları birden fazla refreshe izin veriyor.

Spring dokumanında buna **hot refresh** deniliyor.

Dolayısıyla yani AC configurasyon edilirken yarı yolda bunları tekrardan refresh edip bastan olusturabiliriz.

• After each new configuration set using **setConfiguration**() method **refresh()** should be called in order to load it.

• **If refresh()** is not called new configuration does not have any effect.

• Remember subclasses of **AbstractRefreshableApplicationContext** such as **FileSystemXmlApplicationContext** and **ClassPathXmlApplicationContext** call **refresh()** in their constructors.



Refresh’in amacı yeni bir configuration yüklemektir çagırmazsak hiçbir etkisi olmaz.

• **All** **GenericApplicationContext** classes such as

**AnnotationConfigApplicationContext** don’t support multiple

refresh calls; they allow only one refresh call.



• When multiple **refresh()** calls are made the container throws

**IllegalStateException** with a message **GenericApplicationContext**

**does not support multiple refresh attempts: just call 'refresh' once**

**GenericApplicationContext ve** altındaki sınıfını birden fazla refresh edemeyiz.

**AnnotationConfigApplicationContext**

• **AnnotationConfigApplicationContext** is a standalone

application context that suports annotation based configuration.

• It accepts as input component classes with @**Component**,

@**Configuration**, and **JSR-330** compliant classes using

**javax.inject** annotations.

• Most of the time its constructor receives a class that has

@**ComponentScan** to scan classes with @**Component** or @**Import** to

point to classes with @**Configuration**.

**@Component, @Configuration** objeleri bulup injectionları yapan sınıf.

• **AnnotationConfigApplicationContext** also makes a refresh call in its constructor



**AnnotationConfigApplicationContext** bunu kullanıyorsak yukarıda tipik kullanımları belirtilmistir.

• **AnnotationConfigApplicationContext** also allows for registering

classes both one by one using **register(Class...)** and with classpath

scanning using scan(String…).



• **AnnotationConfigApplicationContext** loads and makes ready new

configuration upon **refresh**() call since default constructor does not call

**refresh**().

• As a sub-class of **GenericApplicationContext**,

**AnnotationConfigApplicationContext** doesn’t support multiple

refresh; it allows only one refresh call in its lifetime.

• So single refresh call should be made either in its constructor if it loads a

configuration or after loading new configuration using either **register**()

or **scan**().

• Once a refresh call is made there is no need for refresh after subsequent

**register**() or **scan**() calls since it automatically loads and makes new

configuration ready.

• There is an exception to this:

• If a class with @**Configuration** is registered using **register**() or

**scan**() it always needs a new **refresh**() call to get effective.

• If the same bean with @**Component** is registered in a new configuration

later registration overrides those in earlier classes.

• Same thing applies to @**Configuration** classes, @**Bean** methods in

later classes will override those in earlier classes.

• Using **register**() or **scan**() is useful to build the configuration

programmatically at run-time.