**What is Bean Lifecycle?**

**What is Lifecycle?**

• Lifecycle is life time that an object goes through.

• There are many milestones called events during the lifecycle of a bean.

• What lifecycle and events mean depend on environments.

• For example in **Spring** events like creation of the bean instance, initialization of that instance with all its dependencies and destruction of the instance when its IoC contanier shuts down are all events in the lifecycle of the bean.

• Or JPA has different events thus lifecycle model for entities.

Objenin hayatı boyunca gittiği yol.

Hayat döngüsü dediğimiz yerlerde köse tasları vardır(**milestones**) buraya vardım gibisinden. İste bu **milestones’larda** eventlere denk geliyor.

Mesela Springde beanlerin instancelarının yaratılması, initialization edilmesi, all its dependencies ile birlikte bir milestone’dur. Çünkü ondan önce bean ile alakalı bir sey yapamayız. Yapmamalıyız.

Ya da bean instanceların destruction olması IOC container ile kapatılması(shutdown) farklı yöntemlerle destruction olması yine beanin hayatında önemli bir adım.

• In non-managed environment all these events are created and managed by the programer.

• But in managed environments such as containers, lifecycle is managed eand events are fired by the container.

• Containers such as Java EE’s servlet and EJB containers and Spring IoC container have their own models for the lifecycle of their obejcts such as servlets, EJBs and beans, respectively.

• So they manage the lifecycle of its objects or beans and provide facilities such as notifications for those events.

Spring’de JPA, EJB ya da farklı yapılarda managed yapılarda gördük. Yani ortamları hep birileri yönetiyor.

Eğer birileri yönetmezse biz kendimiz yönetirsek o zaman nesnenin ne zaman olusturuldugunu ne zaman destroy oldugunu ya da farklı business nesneleri varsa farklı statelerde varsa, a state b statene nasıl gecti gibi bunları managed ortamda değilsek yani container tarafından yönetilen bir ortamda değilsek bu durumda biz yönetiriz.

Mesela herhangi bir hesap acıldıgında ya da kapatıldıgında bize haber gönder gibi.

Genelde business alanında olan yönetme durumları developer tarafında karsılanır ama objenin daha cok ic yasam döngüsüne ait daha teknik durumlarda ise container yönetir bu durumda container bizim icin facilities nesneler daha önemli nesneler olusturur notifications ya da events’ler gibi.

**Events and Callback Methods**

• In managed environments when something important happens to the object whose lifecycle is managed by the container, the container creates an event and notifies those objects that would have an interest for that event.

• The method that receives notifications regarding the events are called **callback method.**

• And objects that represent events are passed to callback methods.

• They are mostly methods of a specific interface and the container calls those methods and passes them event objects.

Mesela **manage** **ortamlarda** **objenin** **lifecycle** ile ilgili önemli bir sey oldugu zaman **container** **yönettiği** **icin** **container** **burada** **eventler** olusturur.(obje olusturuldu ya da obje vs vs oldu gibisinden bilgi tarzı eventlar). Bu eventleri ilgili yerlere bu eventlerla alakalı bilgi almak isteyen diğer objelere gönderir buna da **notification** denir.

**Event olustu, X olayı oldu bununla alakalı X eventini olusturup, ilgililere göndermem lazım yani notification etmek lazım uyarmak lazım. İste o ilgililerin üzerindeki cagıralacak metotlara da callback method denir. Yani o ilgililer kaydolur, sistem burada onları event oldukca bilgilendirir. Dolayısıyla eventi tipik olarak callback methodlarına geceriz.**

\*\*\*Bu durum publisher **subscriber**, **event** **notification**, **observer** design patternlarla yönetilir.

Callback methodları bir interface üzerinde bir method olmalı ki o interface implement eden obje kendini gecmeli ki olay olursa cagır der, objeyi cagır der gibi burada da objeyi cagıramaz dolayısıyla interface üzerindeki methodu cagırabiliriz.

• And of course the objects of those interfaces should register themselves for those notifications i.e. events.

• **This mechanism is implemented mostly using design patterns such as Observer of GoF (also called Publisher-Subscriber, Event-Notification etc.)**