**Inversion of Control (IoC)**

**What is Framework ?**

* Framework ile library farkı, library son ürünken, framework son ürün değildir, semi-completeddir.
* Framework ciddi soyutlamalar sağlar.
* Frameworkler applicationlar tarafından kullanılabilmek icin her sekilde extend etmek gerekir.
* Framework son ürün olmadığı icin, yarı completed oldugu icin bir tamamlanmaları lazım ki kullanılabilsinler, tamamlanmak icin de onları extend etmek lazım.
* Extend dediğimiz yerler **hook points** dediğimiz yerler oradaki abstractları extend etmek ya da interfaceleri implement etmek vasıtasıyla onu genisletiriz, ürün haline getiririz.
* Ürün haline gelen framework değil bizim uygulamamız oluyor, framework’ün sağladığı islevleri kullanarak.
* Bundan dolayı frameworkleri kullanabilmek icin library’den daha fazla bilgiye ihtiyacımız var.

**Frameworklerin** farklı amacları olabilir.

* **Application development frameworkleri** 
  + Buna örnek olarak spring diyebiliriz.
* **Security frameworkleri**
  + Spring security buna örnek olabilir.
* **Business domain frameworkleri**
  + Acord buna örnek olabilir.

**What is IoC ?**

Bir applicationun sahip oldugu akıs kontrolünü elinden bırakması karsı tarafa vermesi temelde framework tarafından uygulanan bir mekanizmadır.

Yani frameworkü extend ettiğimiz zaman belli interfaceleri implement edip belli metotlar override ediyoruz vs. nihayetinde bütün bu extend edilen noktaların cagrılması isi kodu biz yazmamıza ragmen bizde değil frameworkte.

Frameworkü ayağa kaldırdığımız zaman framework o kontrolü kendisi ele alır ve kendi nesneleri ve bizim hook pointlerle sağladığımız nesneler üzerindeki metot çağrılarını ne zaman yapacağını, objeleri nasıl olusturacağını, hangi nesneler arasında DI oldugunu, onları nasıl yöneteceğine, hangi eventleri fırlatacağına vs. bütün bunlara karar veren frameworkün kendisidir. **İste buna IoC denir. Bu bir prensiptir.**

Dolayısıyla yukarıdaki islemde objeleri o olusturuyor, exceptionları o fırlatıyor vs gibi islemler yapıyor.

**Dolayısıyla framework bu akıs kontrolünü ele aldıgı zaman IoC olur.**

**Neyi ne zaman yapacağına karar verir. Bu anlamda applicationumuz frameworkün librarysi olur. Biz applicationun eklemlenen onun librarysi gibi oluruz.**

**Tomcat as A Servlet Container**

Java EE’nin icinde birden fazla frameworkler vardır.

Servlet tarafı baslı basına bir frameworktür.

Servletler tek basına hic bir ise yaramıyor, ama http isteklerini karsılayıp, get, post’lu metotlara cevap verecek sekilde bütün objeleri olusturan, bir istek geldiği zaman gerekli objeleri yaratıp(request, response objelerini yaratan) benim declare ettiğim kodunu yazdığım servlet metodunu map ettigim, ve developer olarak benim yazdıgım servletin objesini olusturan onla alakalı, **ServletContext**  gibisinden bütün objeleri ayağa kaldıran doXXX() lu metodu çağırıp request response gecip, requestten bütün bilgileri almama izin veren responsen üzerine koymama izin veren sey aslında **servlet containerdir**.

**NOT : main metot yok , lifecycle metod felsefesini ortaya çıkarıyor.\*\*lifecycle felsefe arastır.**

**HISTORY OF IoC**

Stefano Mazzocchi used this concept in Avalon of Apache (<https://avalon.apache.org/closed.html>)

and he says that IoC has been first coined by Michael Mattson in his thesis on "Object Oriented Frameworks: a survey on methodological issues" published in 1996

**Michael Mattson says in page 96 of his thesis:**

**An object-oriented framework is “a (generative) architecture designed for maximum reuse, represented as a collective set of abstract and concrete classes; encapsulated potential behaviour for subclassed specializations.”**

Yani azami miktarda tekrar kullanımı ortaya cıkarmak icin tasarlanmıs üreticidir. Abstract ve concrete classlarıda vardır, bunlarla olusturulmustur, temsil edilmistir. Onu extend eden alt sınıflar tarafından daha özel davranıslara izin verecek sekilde potansiyel bir davranısı vardır. \***Comment : O davranısı yerine getirmek demek zaten IoC sinyalini veriyor.**

**The major difference between an object-oriented framework and a class library is that the framework calls the application code. Normally the application code calls the class library. This inversion of control is sometimes named the Hollywood principle, “Do not call us, we call You”.**

Burada yukarıda belirttiğimiz gibi library ile framework farkı framework bizim kodumuzu call eder, ama library kullansaydık biz library codunu call ederiz.

But Johnson and Foote uses the term in their paper titled Designing

Reusable Classes in JOOP of June 1988 <http://www.laputan.org/drc/drc.html>

Comment : ganf of four’daki **Ralph Johnson**

**One important characteristic of a framework is that the methods defined by the user to tailor the framework will often be called from within the framework itself, rather than from the user's application code. The framework often plays the role of the main program in coordinating and sequencing application activity. This inversion of control gives frameworks the power to serve as extensible skeletons. The methods supplied by the user tailor the generic algorithms defined in the framework for a particular application.**

Yukarıdakine benzer bir acıklama yapmıslardır. Framework burada ciddi bir koordine sağlar, yani uygulamayı götüren frameworktür. Genisletilebilen iskelet olarak hizmet vermesini sağlar.

**Keyword : generic algorithms**

**Later M. Fowler wrote on the topic** (<https://martinfowler.com/bliki/> InversionOfControl.html):

**Inversion of Control is a key part of what makes a framework different to a library. A library is essentially a set of functions that you can call, these days usually organized into classes. Each call does some work and returns control to the client. A framework embodies some abstract design, with more behavior built in. In order to use it you need to insert your behavior into various places in the framework either by subclassing or by plugging in your own classes. The framework's code then calls your code at these points.**

**Benefits of Using Frameworks**

Framework kullanarak neler sağlanıyor dersek:

Daha hızlı development sağlıyor.

Business tarafına odaklanmamızı sağlıyor.

Daha reliable code sahip.

Etc.,

**DI and IoC**

DI ile Ioc arasındaki fark :

DI ile Ioc ile basarılır.

Bu anlamda IoC , DI’n arka taraftaki prensibi diyebiliriz.

DI ise IoC elde ettiğimiz durumlardan birisi.

DI olmasa IoC ile farklı yapıları implemente edebiliriz.