1. Geliştiricinin proje yaparken karşılaştığı birden çok problem vardır. Bunlar karşısında kendisi bireysel olarak çözüm üreteceğine framework’ler yardımıyla bu sorunları daha basit ve çözülebilir hale getirebiliyor.

Spring MVC üzerinden örnek verecek olursak, IoC ile bağımlılıkları azaltırız yani biz değil de Container yapısı ile bu yükü üzerimizden atacağız. DI ile anotasyonları kullanarak nesne üretilmeden injection yapısı ile bu işi çözeceğiz.

JSP ile örnek verecek olursak, dinamik web sayfaları oluştururken sağladığı kolaylıkları söyleyeceğiz. Sunucu ile istemci arasındaki işlemler için Servlet sınıfını kullanacağız.

Struts ile örnek verecek olursak, MVC katmanının tabanı şeklindedir. Yani MVC’yi temel olarak almış ve MVC mimarisini kullanan bir framework’tür.

Vaadin ile örnek örnek verecek olursak, bir web framework’üdür. HTML ve Javascript’in detaylarından bizi kurtarır. Bu framework ile birçok Javascript kütüphanesini entegre edip kullanmamızı sağlar. Google Web Toolkit’in sunucu ve istemci olayını bu framework sayesinde entegre edip kullanmamızı sağlar.

1. Katmanlı mimari, projemizi oluşturmaya başladığımızda belli kurallar çerçevesinde oluşturup zorlanmaya gerek kalmadan daha basit bir şekilde oluşturmamızı ve karşılaştığımız herhangi bir sorun karşısında basit olarak çözmemize kolaylık sağlar.
2. Garbage collector, çöp toplayıcısı demektir. Yani oluşturduğumuz nesnelerden referansı verilmeyenler var ise bunların otomatik olarak silmesini sağlar ve heap bellekte yer açar.

C++ da bu işi geliştirici yapar, yani yöneticinin kontrolündedir. Geliştirici bu işlemleri malloc() ve free() gibi fonksiyonlar yardımıyla yapar. Şöyle ki yazdığı uygulama ile işi bittiğinde bu hafızayı tekrar geri bırakmalı ki uygulama çalışsın. Fakat Java da bu böyle değildir. Bu işlem otomatik olarak yapılır ve geliştiriciye bırakılmaz.

1. Spring de kullandığımız tasarım desenleri şunlardır:

* **Factory Method:** Birbirinden farklı iki sınıf var ama bu iki sınıfın işlevlerinin benzer olduğu desendir.
* **Singleton:** Yalnızca bir obje bir defa yaratılır.
* **Prototype:** Nesneyi yazdığımız kodların class’larına bağımlı hale gelmeden kopyalamayı sağlar.
* **Proxy:** Bir obje için yedek olarak yer tutan bir desendir.
* **Template Method:** Bir üst sınıfta algoritmanın kabataslağını oluşturur fakat belli adımlarını değiştirmemize olanak sağlayan desendir.
* **Observer:** Bir objeden diğer objeye olan bilgilendirmeyi sağlar yani bir müşterinin kargosunun gelip gelmediğini öğrenmek için sürekli bilgi almaya gitmesi gibidir.
* **Mediator:** Objeler arasında bağımlılığı azaltır yani bir nesne ile diğer nesne arasında doğrudan iletişim kurmasına izin vermez araya aracı koyar ve iletişimi onun üzerinden sağlar.
* **Front Controller:** Buradaki önemli olan amaç, gelen istekleri merkezde toplayıp ilgili yere göndermektir.

1. **Abstract Factory, Factory Method, Builder, Prototype, Singleton**