MikroServis Mimarisinde Koordinasyon ve Haberleşme Modelleri

Mikroservisler arasındaki koordinasyon ve haberleşme modellerini belirleyen en önemli şey **birden fazla servise yayılan süreçler**in nasıl ele alınacağı konusudur

Koordinasyon Modelleri

- Birden fazla servise yayılan süreçleri yönetirken mikroservisler arasındaki **koordinasyonu** sağlamak için **iki temel yaklaşım** söz konusudur
  - Orkestrasyon
  - Kareografi

Orkestrasyon vs Kareografi

- Orkestrasyon daha **merkeziyetçi** bir yapıdır
- Servisleri **yöneten ve koordine eden** merkezi bir yapı vardır
- Bir orkestradaki müzisyenleri yöneten **bir şef**e benzetilebilir

Orkestrasyon vs Kareografi

- Kareografide ise sistemi oluşturan servislerin **görev ve sorumlulukları** belirlenmiştir
- Her bir servisin bu görev ve sorumlulukları **nasıl yerine getirecekleri** kendilerine kalmıştır

Mikroservisler Arası Haberleşme Modelleri

- Mikroservisler arasında **haberleşme modelleri iki boyut**ta ele alınabilir
  - Birinci boyut: 1-1 veya 1-M haberleşme
  - İkinci boyut: senkron veya asenkron haberleşme
- Mikroservislerin birbirleri ile ne şekilde iletişim kuracakları bu **iki boyutun farklı kombinasyonları**na göre belirlenir

Mikroservisler Arası Haberleşme Modelleri

	1-1	1-M
Senkron	Request/Senkron response	-
Asenkron	Notification	Publish/Subscribe
	Request/Asenkron response	Publish/Asenkron Response'lar

Request/Response

- İstemci **sunucuya bir istek** gönderir
- Bu istek sunucu tarafından ele alınır ve **bir cevap** üretilir
- Bu cevap dönene kadar **istemci beklemeye devam** eder
- Request/response yöntemini hayata geçirmek için de temelde **iki yöntem** vardır
  - **RPC** tabanlı
  - **REST** tabanlı

Request/Response

- SOAP, RMI, Thrift gibi haberleşme yöntemleri **RPC tabanlı**dır
- RESTful yaklaşım ise Web'in çalışma prensiplerinden esinlenmiş **mimarisel bir yaklaşım**dır

REST Tabanlı Yaklaşım

- REST tabanlı yöntemde en önemli şey giden/gelen **Resource**'tur
- Bir Resource'un sunucu tarafında gösterimi veya saklanması ile **dış dünyaya nasıl sunulduğu** tamamen birbirinden farklı olabilir
- REST yaklaşımının implement edildiği **en yaygın protokol HTTP**'dir
- HTTP'nin bazı özellikleri RESTful yaklaşımı hayata geçirmeyi **kolaylaştırmakta**dır

Notification

- Bu haberleşme modelinde **sadece request** söz konusudur
- İstemci request'i yaptıktan sonra sunucudan herhangi bir **cevabın dönmesini beklemez**

Request/ Asenkron Response

- İstemci, sunucuya bir request gönderir, ancak **cevap** sunucudan **asenkron** biçimde dönülür
- İstemci cevap için beklememesi gerektiğini, **cevabın** dönmesinin **belirli bir süre alabileceğ**ini bilir

Publish/Subscribe

- Servislerden birisi gerçekleşen bir işlem veya durumla ilgili bir **olay bilgisi (event) yayımlar** ve kendisi kaldığı yerden işine devam eder
- Bu işlem veya durumla ilgilenen diğer bileşenler/servisler bu **olay bilgisinden haberdar** olduklarında harekete geçerek gerekli işlemleri yürütürler
- Servislerin birbirlerini request/response modeldeki gibi **beklemesi söz konusu değil**dir

Event Tabanlı Haberleşme

- Asenkron **event tabanlı haberleşme**yi hayata geçirmek için de farklı yöntemler mevcuttur
  - **AMQP**/RabbitMQ
  - **ESB**
  - **ATOM**

Publish/ Asenkron Response

- İstemci bir **istek yayımlar** ve belirli bir süre ilgili servislerden **cevapların dönmesi**ni bekler
- Bu bekleme anında **istemcinin bloklanması söz konusu değil**dir

Haberleşme Modelleride Dikkat Edilecek Noktalar

- HTTP protokolü “low latency” noktasında çok da **başarılı değil**dir
- Özellikle yük arttıkça HTTP’den kaynaklı **gecikmeler artmakta**dır
- Bu noktada **hafif sıklet** bir takım **message broker’lar** daha performanslıdır
- Dolayısı ile **servisler arası**ndaki haberleşmelerde **asenkron** yol tercih edilir
- **Servis içi** haberleşmelerde ise **senkron** yöntem daha uygundur

Haberleşme Modelleride Dikkat Edilecek Noktalar

- Asenkron haberleşmede **her bir servisi ayrı bir state-machine** olarak kabul etmek çok faydalıdır
- Örneğin, belirli bir **entity'nin yaşam döngüsü** ile ilgili bütün **event'ler** ilgili mikroservisin kontrolünde olmalıdır

Haberleşme Modelleride Dikkat Edilecek Noktalar

- Servislerin kendi **içindeki işlemler**de mümkün olduğunca **DRY** (don't repeat yourself) prensibine uyulmalıdır
- Ancak **servisler arası**ndaki haberleşme ve işlemlerde DRY prensibinden **taviz** verilebilir
- Aksi durum servisler arasındaki **coupling** problemini artıracaktır
- Bundan kaçmak için kod **duplikasyon**una göz yumulabilir

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com