MikroServisler ve Loglama & İz Sürme

Mikroservisler ve Loglama & İz Sürme

JAVA Eğitimleri

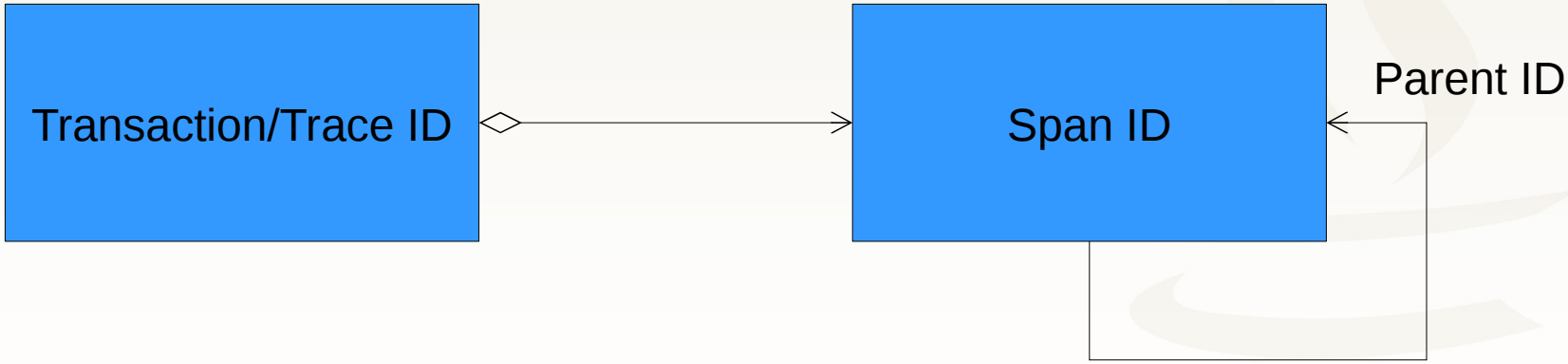
- Mikroservis mimarisinde **isteklerin karşılanması** genellikle **birden fazla servis** üstünden geçerek sağlanır
- Her bir servis kendi içerisinde **log çıktısı** üretebilir
- İsteğin hangi servisler tarafından karşılandığı, bu **servislerin ürettiği log çıktıları**nın ne olduğu kolayca tespit edilebilmelidir

Logların Birleştirilmesi & İz Sürme

- Farklı servislerin ürettiği log çıktılarının **tek bir noktada birleştirilmesi** ve merkezi bir lokasyondan yönetilmesi önemlidir
- Bu sistemin **büyük miktarda log çıktısı**nı ele alabilecek kabiliyette olması gerekir
- Bu noktada farklı servislerden gelen logların hangi istekle alakalı olduğunu tespit etmek için **benzersiz bir log/trace id**'ye ihtiyaç duyulur

Benzersiz Log/Trace Id

- Sistem genelindeki servis çağrılarının izini sürmeyi sağlayacak trace ID bilgisi **correlation-id** olarak adlandırılmaktadır



Logların Birleştirilmesi & İz Sürme

- İstemci tarafı correlation id hakkında bilgilendirilmelidir
- Log mesajları içerisinde arama yapılabilmesi önemlidir
- **Log düzeyleri** sistem genelinde **dinamik** olarak değiştirilebilmelidir
- Log mesajlarının merkezi sisteme toplanırken **kaybolmaması** da önemlidir

Logların Birleştirilmesi & İz Sürme

- **Hata mesajları ve exception'lar**ın da ortak bir noktada tekrarlanmadan toplanması gerekir
- Ayrıca hata mesajları ve exception'lar hakkında **geliştiricilerin anlık olarak haberdar edilmesi** de önemlidir

Health Check API

- Servislerin isteklere cevap vermeye **hazır durumda olup olmadıkları**nın tespit edilmesi önemlidir
- Herhangi bir servisin hangi bileşenlerinde sorun olduğunun **anlık olarak tespit edilmesi** ve geliştiricilerin bundan haberdar edilmesi gerekir
- Servislerin bu amaçla bir "**health check**" endpoint'e sahip olması gerekir

Health Check API

- Genelde health check API **service registry** tarafından invoke edilir
- Health check API, servisin çalıştığı yerdeki **disk kapasitesi, CPU, hafıza durumu** gibi değerlere de erişmeyi sağlayabilir
- Servisin **diğer servislere** (email, broker vb) olan **bağlantı durumları**nı kontrol edebilir
- Uygulamaya özel **iş mantığının çalışırlığı**nı kontrol edebilir

Audit Logging

- Servisler üzerinde gerçekleştirilen **operasyonların hangi kullanıcılar tarafından, ne zaman ve nereden gerçekleştirildiğ**inin takip edilmesine audit logging adı verilir
- Mikroservis mimarisinde audit kayıtlarının **merkezi bir lokasyon**da toplanması önemlidir
- Audit kayıtlarının **birbirleri ile ilişkisi** de sağlanmalıdır

Servis Metrikleri

- Servislerin istekleri karşılama performansları ve sıklıkları ile ilgili pek çok **metrik** üretilir
- Bu metriklerin merkezi bir lokasyonda **toplanması ve yönetilmesi** gerekir
- Ayrıca bu metriklerle ilgili kullanıcıların ve geliştiricilerin de **anlık olarak haberdar edilmesi** önemlidir

Servis Metrikleri

- Servis metriklerinin merkezi bir metrik servisi tarafından toplanması için **iki farklı yaklaşım** söz konusudur
  - Push
  - Pull

Log & Tracing Örnekleri

- **Zipkin+Kafka** ile sistem genelinde tracing yapılabilir
- **Grafana**, web tabanlı bir dashboard ve log visualization aracıdır
- **Prometheus** ile de zaman duyarlı verinin saklanması ve sorgulanması sağlanabilir

İletişim

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