Web Programlama I

Ders 06: EJB

Slaytlar Hakkında

- Bu slaytlar oldukça yüksek seviyede genel bir bakış açısı sunmaktadır
- Detaylar Git repository’sindeki yorum satırlarında bulunmaktadır

Proxy Class İyileştirmeleri

- EJB'leri yazarken iş mantığına odaklanırsınız
- JEE Container fonksiyonelliği otomatik olarak ekleyecektir
- Ör, Her bir metot için transaction başlatma ve commit etme
- Ayrıca daha farklı fonksiyonellikler de mevcuttur ve yalnızca @annotations yazarak aktif edilir

@Asynchronous

- @Asynchronous olarak işaretlenmiş bir metot arka planda farklı bir thread'de yürütülür ve caller'a hemen geri döner (metodun bitmesini beklemez)
- Uzun işlemlerde ve caller'ın sonucu alması gerekmediğinde kullanışlıdır (side-effect önemlidir)
- Thread oluşturmanıza (ki maliyetlidir) veya bir thread havuzunu manuel olarak yönetmenize gerek yoktur… JEE Container bu annotation'ı kullanarak sizin için otomatik olarak yapacaktır.

@Schedule

- Diyelim ki belirli bir zaman aralığında bir veya birkaç kez bir işlem yapmanız gerekiyor
  - ör, harici bir kaynağı 30 saniyede bir kontrol etmek için
  - ör., doğum günü olan kullanıcılara özel bir teklif göndermek için
- Kendi thread ve planlayıcınızı oluşturabilirsiniz… veya EJB metoduna @Schedule annotate'i ile belirtebilirsiniz

Transaction’lar

- EJB metotları bir transaction içerisinde çalıştırılır
- Ancak bir EJB bir başka EJB içerisinde çağırılırsa ne olur?
  - Yeni bir transaction mı olacaktır?
  - Transaction’lar tek bir transaction olarak birleştirilir mi?
- EJB metodu aynı EJB'de başka bir metodu çağırırsa ne olur?
- EJB metot çağrısı sırasında bir exception meydana gelirse ne olur?
  - EntityManager önbelleğinde şu ana kadar yapılan değişiklikler commit mi?
  - Veya her şey geri mi alındı?
- Bütün bu senaryolar annotation’lar ile ayarlanabilir
- …varsayılan ayarlamalar çoğu zaman en iyi seçim olacaktır ancak transaction’ın başlama ve bitişini iyi anlamak gerekir

Transaction’lar

- EJB public metotlar varsayılan olarak transaction içerisinde ele alınır
- @annotation kullanarak ayarlamalar yapılabilir
  - *REQUIRED*: varsayılan ayar, aktif bir transaction yoksa yenisini başlat, varsa mevcut olana katıl
  - *SUPPORTS*: Devam eden bir transaction varsa ona katıl
  - *REQUIRES_NEW*: her zaman yeni bir transaction başlat eğer devam eden varsa onu askıya al
  - *MANDATORY*: bir devam eden transaction’da koşmalı yoksa hata ver
  - *NOT_SUPPORTED*: devam eden transaction’ı beklemeye al
  - *NEVER*: bir transaction içerisindeyse exception fırlat

Git Repository Modülü

- *NOT: açıklamaların büyük bir çoğunluğu kod içerisinde yorum satırı olarak bulunmaktadır, burada slaytlarda bulunmamaktadır*
- **intro/jee/ejb/async**
- **intro/jee/ejb/time**
- **intro/jee/ejb/transactions**
- Ders 06 alıştırması (dokümantasyona bakınız)