JPA ve İkincil Ön Bellek (Caching)

JPA ve İkincil Önbellek Desteği

- JPA 2.0 ile gelen yeniliklerden birisi de **ikincil önbellek desteği**dir
- Persistence provider tarafından yönetilen verinin DB yerine uygulamaya daha yakın bir yerde yönetilmesidir
- **Performans amaçlı** bir çözümdür
- JPA spesifikasyonunda ikincil önbellek kabiliyeti **opsiyoneldir**, JPA provider ikincil önbellek kabiliyetine sahip olmayabilir

JPA ve İkincil Önbellek Desteği

- Böyle bir durumda JPA 2 önbellek ayarları sessizce **göz ardı** edilir
- Cache konfigürasyonu **persistence unit düzeyinde** yapılır
- Hibernate'de de konfigürasyon **SessionFactory düzeyinde** yapılmaktadır
- Genel olarak Hibernate ile ikincil önbellek kabiliyetine **benzer** bir konfigürasyon ve kullanıma sahiptir
- Ancak **Hibernate'deki kadar kapsamlı bir çözüm sunmamaktadır**

JPA İkincil Önbellek Konfigürasyonu

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
    <persistence-unit name="testjpa">
        <shared-cache-mode>ALL</shared-cache-mode>
        <properties>
            <property name="hibernate.ejb.cfgfile" value="/hibernate.cfg.xml" />
        </properties>
    </persistence-unit>
</persistence>
```

Alabileceği değerler:
NONE
ALL
ENABLE_SELECTIVE
DISABLE_SELECTIVE
UNSPECIFIED

JPA İkincil Önbellek Modları

- **ENABLE_SELECTIVE**: **@Cacheable(true)** anotasyonuna sahip **entity sınıfları** için önbellekleme aktif olur
- **DISABLE_SELECTIVE**: **@Cacheable(false)** ile işaretlenmeyen bütün **entity sınıfları** otomatik olarak önbelleklemeye tabi tutulur

```
@Entity
@javax.persistence.Cacheable(true)
public class Pet {
}
```

JPA İkincil Önbellek Modları

- **NONE**: @Cacheable annotasyonuna bakılmaksızın ikincil önbellek **devre dışı** bırakılır
- **ALL**: @Cacheable annotasyonuna bakmaksızın **bütün entity'ler** için devreye alınır
- **UNSPECIFIED**: Persistence **provider'ın default davranışı**na bakılır

Programatik JPA İkincil Önbellek Konfigürasyonu

- **EntityManagerFactory** yaratılırken programatik olarak da ikincil önbellek konfigürasyonu yapılabilir

```
Properties jpaProperties = new Properties();

jpaProperties.put("javax.persistence.sharedCache.mode","ENABLE_SELECTIVE");

EntityManagerFactory emf = Persistence
            .createEntityManagerFactory("testjpa", jpaProperties);
```

JPA İkincil Önbellek Erişim Kontrolü

JAVA Eğitimleri

- İkincil önbelleğin kullanımı sonucunda DB tablolarındaki veri ile önbellekteki **verinin farklı zamanlarda değişmesi** söz konusu olur
- Bu durumda "**stale read**" problemi ortaya çıkar
- Stale read problemini çözmek için **cache retrieval ve store modları**ndan yararlanılır
- **EntityManager** düzeyinde set edilebilen **retrieveMode** ve **storeMode** property'leri ile ikincil önbellek erişimi kontrol düzenlenir
  - javax.persistence.cache.retrieveMode
  - javax.persistence.cache.storeMode

Cache Retrieve Modları

- **CacheRetrieveMode.USE**:
  - Entity nesnenin mevcut ise önbellekten okunmasını söyler
  - Eğer cache'de değil ise DB'den okuma yapılır
  - EntityManager.refresh() de göz ardı edilir, çünkü bu işlemde her zaman DB'ye gidilmelidir
- **CacheRetrieveMode.BYPASS**:
  - Önbelleğin göz ardı edilip verinin doğrudan DB'den okunmasını sağlar

Cache Store Modları

- **CacheStoreMode.USE:**
  - DB'den okuma yapıldığında veya TX commit olduğunda verinin ön belleğe yazılmasını veya önbellekte güncellenmesini söyler
  - Halihazırda önbellekte bulunan bir veri ise herhangi bir şey yapmaz
- **CacheStoreMode.BYPASS**:
  - Verinin ön belleğe yazılmamasını sağlar, Önbelleğe hiç dokunulmaz

Cache Store Modları

- **CacheStoreMode.REFRESH**:
  - DB'den okuma olduğunda veya TX commit anında verinin ön belleğe yazılmasını söyler
  - Ancak USE'dan farklı olarak veritabanından okuma yapıldığında önbellekte daha evvel mevcut olan verinin güncellenmesini de sağlar

JPA İkincil Önbellek Erişim Kontrolüne Örnekler

```
EntityManager entityManager = emf.createEntityManager();
entityManager.setProperty("javax.persistence.cache.storeMode", "BYPASS");
```

```
Map<String, Object> properties = new HashMap<String, Object>();
props.put("javax.persistence.cache.retrieveMode", "BYPASS");
Owner owner = entityManager.find(Owner.class, 1L, properties);
```

```
CriteriaBuilder cb = emf.getCriteriaBuilder();
CriteriaQuery<Owner> criteriaQuery = cb.createQuery(Owner.class);

TypedQuery<Owner> query = entityManager.createQuery(criteriaQuery);
query.setHint("javax.persistence.cache.storeMode", "REFRESH");
```

Programatik İkincil Önbellek Erişimi

- **javax.persistence.Cache** arayüzü üzerinden de ikincil **önbellek kullanımı programatik olarak** da gerçekleştirilebilir

```
Cache cache = entityManagerFactory.getCache();

boolean contains = cache.contains(Owner.class, 1L);

cache.evict(Owner.class, 1L);

cache.evict(Owner.class);

cache.evictAll();
```

JPA İkincil Önbellek ve Eşzamanlı Erişim Stratejisi

- JPA'da Hibernate'de olduğu gibi **entity veya collection ilişkisi düzeyinde eşzamanlı erişim stratejisi belirtmek mümkün değil**dir

JPA ve Query Cache

- JPA 2'de **sorguların önbelleklenmesi** ile ilgili ise herhangi net bir kabiliyet ortaya konmamıştır
- Sorgularla ilgili olarak yine **ORM çözümüne spesifik özelliklerin kullanılması** söz konusudur

```
Query query = entityManager.createQuery("select o from Owner o");
query.setHint("org.hibernate.cacheable", true);
...
```

İletişim

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