İlişkilerin Lazy/Eager Yüklenmesi

harezmi bilişim çözümleri

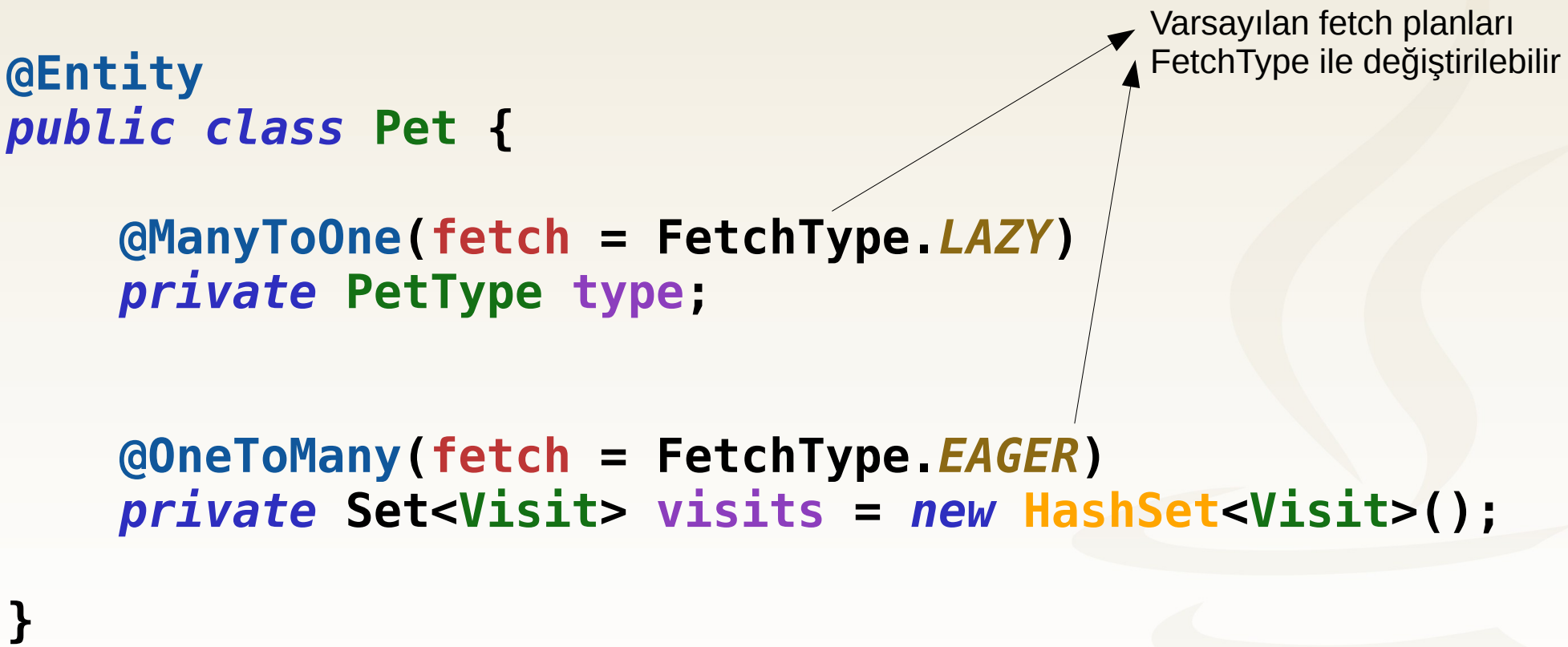
İlişkilerin Yüklenmesi

- Bir entity'nin sahip olduğu **ilişkilerin ne zaman ve nasıl yükleneceği** uygulama tarafından yönetilebilir
- İlişkilerin ne zaman yükleneceğine **fetch plan/type** adı verilir
  - Lazy, Eager
- İlişkilerin nasıl yükleneceğine ise **fetch strategy** adı verilir
  - Select, Join

Fetch Planı

- Herhangi bir ilişkinin, ana nesne yüklendiğinde mi (**EAGER**) veya ilişkiye ilk erişim anında mı (**LAZY**) yükleneceğine **fetch planı** denir
- Default olarak Hibernate **bütün 1:M ve N:M ilişkileri (entity veya bileşen) LAZY** olarak yönetir
- Lazy bir **collection'ın initialize edilmesi** için iterate(), size(), contains(), isEmpty(), add(), remove(), clear() gibi metotları kullanmak gerekir
- M:1 ve 1:1 ilişkiler ise **default durumda EAGER** yüklenir

Fetch Planı


Varsayılan fetch planları
FetchType ile değiştirilebilir
@Entity
public class Pet {
@ManyToOne(fetch = FetchType.LAZY)
private PetType type;
@OneToMany(fetch = FetchType.EAGER)
private Set<Visit> visits = new HashSet<Visit>();
}

1:1 Embeddable Bileşen ve Lazy

- 1:1 **embeddable bileşenler** ise doğrudan hemen yüklenir
- Bunları **LAZY** yapmak **mümkün değil**dir

Lazy İlişkiler ve Persistent Collection

JAVA Eğitimleri

- Hibernate 1:M ve M:N lazy ilişkileri yönetmek için entity nesneyi yüklerken lazy collection değişkenlerine değer olarak kendi **Persistent collection** sınıflarından uygun bir instance'ı atar
  - Set için PersistentSet
  - List için PersistentList
  - Bag için PersistentBag
  - Map için PersistentMap
- Entity'nin **lazy collection ilişkisine erişildiği vakit** ilgili persistent collection, elemanlarını DB'den **ayrı bir SELECT** ile yükler

Hibernate Extra Lazy Kabiliyeti

```
@OneToMany
@LazyCollection(LazyCollectionOption.EXTRA)
private Set<Pet> pets = new HashSet<Pet>();
```

- Hibernate'e özel "**extra lazy**" collection eşleme yöntemi kullanılırsa **size()**, **contains()**, **isEmpty()** gibi metotlara erişim de **initialization'ı tetiklemez**
- Sadece **ilgili bilgi** için DB'ye ayrı bir sorgu atılır
- Collection tipi Map veya List ise **containsKey()** ve **get()** metodları için de doğrudan DB'ye erişilir

Lazy İlişkiler ve Proxy

- Lazy olarak tanımlanmış M:1 ve 1:1 ilişkiler için ise entity yüklenirken M:1 veya 1:1 ilişkisi mevcut ise **hedef entity yerine geçecek bir proxy nesne** yaratılarak bu set edilir
- Entity'nin lazy M:1 veya 1:1 ilişkisine erişildiği vakit **proxy nesne ayrı bir SELECT ile** kendisini initialize edecektir

LazyInitializationException Hatası ve Önleme Yolları

JAVA Eğitimleri

- Lazy ilişkilerin veya proxy'lerin initialize edilebilmeleri için **source entity'nin yüklendiği Session'ın açık olması** gerekir
- Aksi takdirde **LazyInitializationException** hatası ortaya çıkacaktır
- Lazy hatası ile karşılaşmamak için **değişik çözüm yolları** mevcuttur

LazyInitializationException Hatası ve Önleme Yolları

JAVA Eğitimleri

- Proxy'nin veya persistent collection'ın herhangi **bir property'sine Session açık iken erişmek**
- Hibernate.**initialize**(Object **proxy**)
  - Proxy nesne veya persistent collection **Session kapanmadan önce** initialize edilmelidir!

LazyInitializationException Hatası ve Önleme Yolları

JAVA Eğitimleri

- Entity'nin Session'a **reattach** yapılması
  - Proxy nesne veya persistent collection'ı içeren entity lock, update, merge gibi metotlarla yeni bir Session ile ilişkilendirilebilir
- Spring ile çalışırken **OpenSessionInViewFilter**'ın kullanılması
- Lazy ilişkinin "**fetch join**" ile eager initialize edilmesi
- Hibernate 4 ile gelen **hibernate.enable_lazy_load_no_trans** property değerinin "**true**" yapılması

İletişim

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