

Nesnelerarası Baęlantılar

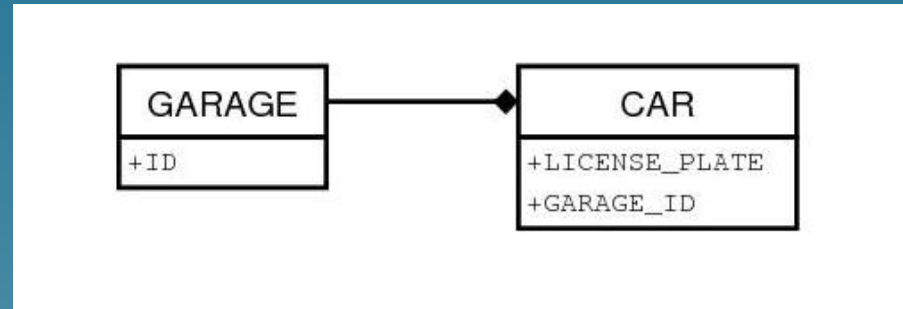
Burak Bayramlı

Bu belge, \LaTeX ile üretilmiştir

Çetrefil Bağlantılar

JBossWS, Hibernate'in idare ettiği daha girift nesne bağlantılarını da XML'e çevirip otomatik olarak geriye döndürebilir.

Bire çok



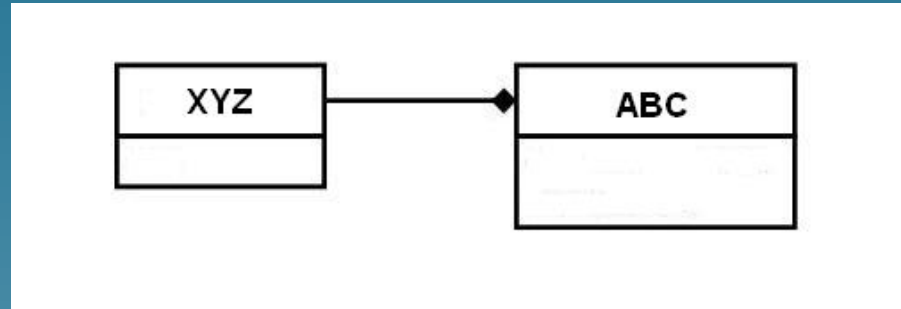
Bire çok - Veri Tabanında

- Bire çok ilişkisi veri tabanında “foreign key” kavramı kullanılarak gerçekleştirilir.
- A’dan bir B’den çok var ise, B’nin asal anahtarını A’ya yabancı anahtar olarak koyarız.
- Herhangi bir A satırı ile ilişkili olan B’leri bulmak için önce A’yı bulup, oradan A’nın yabancı anahtarına uyan tüm B’leri filtreleriz (SELECT .. WHERE kullanarak)

Bire çok - Hibernate

- @OneToMany annotation'ı bire çok ilişki için kullanılır
- Bir tarafından bir Java listesi alacağız - bu liste List, Set, ya da Map olabilir.

Bire çok - Hibernate



Bire çok - Bir

```
@Entity
public class Xyz {

    @Id ...

    @OneToMany
    @JoinColumn(name="xyz_id")
    List<Abc> abcList;

    public List getAbcList() { .. }
```

...
}

Bire çok - çok

```
@Entity
public class Abc {

    @Id ...
    private String id1;

}
```

Bu Kadar!

- JBossWS bu ilişkiyi olduğu gibi alıp çağıran tarafına döndürebilir.
- Abc nesnesini yükleriz, ve onu döndürürken ilişkisi olan Xyz nesnelere de gelirler.

Daha Zor Baęlantılar

Daha Zor Bağlantılar

- Şimdiye kadar nesnelerimizin XML'e nasıl dönüştüğü ile hiç ilgilenmedik.
- Bunun sebebi, bu dönüşümün “default” ayarlar ile çoğu zaman yapılabilir olmasıdır.
- Bu çevrimi arka planda yapan paket JAXB adlı XML kütüphanesidir.

Daha Zor Bağlantılar

JBossWS'in bize hissetirmeden kullandığı JAXB ile olan iletişime özel şartlar sözkonusu olduğunda karışmamız gerekiyor.

Bire çok - Bir

```
@Entity
public class Xyz {

    @Id ...

    @OneToMany
    @JoinColumn(name="xyz_id")
    List<Abc> abcList;

    public List getAbcList() { .. }
```

...
}

Bire çok - çok

```
@Entity
public class Abc {

    @Id ...

    @ManyToOne
    Xyz xyz;

    public Xyz getXyz() { ... }
    ...
}
```

}

XML'e Çevirilebilir mi?

- Burada karşımıza problem çıkacak. XML hiyerarşik bir yapıdır ve bu türlü ileri-geri (tekrar ileri) ilişki çeşidi JAXB tarafından default haliyle kabul edilmeyecektir. “Bitmez döngü (cycle) var” gibi bir hata ortaya çıkacaktır.
- Bu durumu düzeltmek için JAXB annotation'larını kullanarak XML eşlemesinde ilişkinin bir tarafını geçici

(transient) olarak tanımlamak yani ağırlığı öteki tarafa vermek gerekiyor.

XML Eşlemesi

```
@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement
@Entity
public class Xyz {

    @XmlElement(nillable = true)
    @Id ...

    @OneToMany
    @JoinColumn(name="xyz_id")
```

```
@XmlElement(nillable = true)
List<Abc> abcList;

public List getAbcList() { .. }
...
}
```

XML Eşlemesi

```
@XmlAccessorType(XmlAccessType.FIELD)
@Entity
public class Abc {

    @Id ...
    @XmlElement(nillable = true)

    @ManyToOne
    @XmlTransient
    Xyz xyz;
```

```
public Xyz getXyz() { ... }  
...  
}
```

Lazy Loading Hatası

- Bazen bir uygulama servis tarafından tek bir obje döndürüp, *client tarafında* o objeden yan objelere atlamak isteyebilir.
- Aynı JVM içinde bu istenirse, problem olmazdı, çünkü Hibernate Session hala açık
- Fakat client / server ortamında metod çağrısı bitince, session kapanır. Client yan objeleri alamaz

- Alınmaya uğraşılırsa, Lazy Loading ile ilgili bir Exception görülecektir.

Lazy Loading Hatası

Çözüm: Servis metodu bitmeden yan objelerin yüklenmesini zorlamak

- Servis tarafında liste gezme metodu
- Query API kullanıp, tek obje alırken bile JOIN yapılmasını zorlamak.

Lazy Loading Hatası

Liste gezme metodu

- Kullanımı basit: `obj.getXyzList()` ile alınan nesne, teker teker ziyaret edilir, ve her objenin bir get metodu çağırılır (böylece Hibernate o her obje için bir SELECT yapmaya mecbur kalacaktır).
- Performansı kötü: Bu kullanım N+1 problemi denen N tane yan nesne için N tane SELECT uygulanması

demektir (+1 de ana objenin kendisi). Fazla SELECT iyi değildir.

Lazy Loading Hatası

JOIN zorlama metodu:

- Yükleme biraz daha uzadı: `find` yerine artık Query API kullanıp `select x from Xyz x where ..` gibi bir ifade kullanıyoruz.
- Hızlı: Sadece bir `SELECT` ile işi hallediyoruz.

```
select x from Xyz x
join fetch x.abcList
```

Görevler

- Car class'ına ek olarak Garage class'ı kodlayın. Bu class'ın garageId ve description adında iki ögesi olsun. Garage'ın bir Car listesi olsun. Ters yöndeki ilişkiye başta gerek yok.
- CarService üzerindeki getCar çağrısını değiştirmeden işletin. Sonuç ne?
- Şimdi Car'dan Garage'a @ManyToOne bağlantısını

ekleyin.

- Şimdi gerekli JAXB annotation'larını ekleyin. Ana obje Garage, geçici XML ögesi Car objesinde olsun.
- CarService'e getGarage adlı bir metot ekleyin. Bu metot verilen garageId'yi kullanıp Garage'ı yükler ve geri döndürür.

Import

```
import javax.persistence.Entity;
import javax.persistence.Column;
import javax.persistence.Id;
import javax.persistence.ManyToOne;
import javax.persistence.JoinColumn;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlRootElement;
```

```
import javax.xml.bind.annotation.XmlTransient;
```

Import

```
import java.util.List;
import javax.persistence.Entity;
import javax.persistence.Column;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlRootElement;
```

Import

```
import javax.ejb.Stateless;  
import javax.ejb.TransactionAttribute;  
import javax.ejb.TransactionAttributeType;  
import javax.persistence.PersistenceContext;  
import javax.persistence.EntityManager;  
import javax.jws.WebService;  
import javax.jws.WebMethod;  
import javax.jws.soap.SOAPBinding;  
import javax.jws.soap.SOAPBinding.Style;
```