

**T.C.
ONDOKUZ MAYIS ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**



Veri Tabanı Laboratuvarı

Deney Föyü-6

T.C
ONDOKUZMAYIS ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Veri Tabanı Lab. Dersi Deney Föyü-6

Konu: ORM (Object Relational Map) Mimarisinin anlaşılması ve uygulamalı olarak kullanılması

Deney Sonucu İstenen: Aşağıda ORM mimarisi ile verilen bilgileri okuyarak bu konu ile ilgili bir uygulama geliştirmeniz beklenmektedir. Gerçekleştireceğiniz uygulama şu şekilde olacaktır.

Bir öğrenci tablosuna öğrenci ekleme işlemi yapılacaktır. Daha sonra seçilen fakülte ve bölüm bilgisine göre öğrenci listeleme işlemi yapılacaktır.

Öğrenci Tablosu : tOgrenci(ogrenciID, ad, soyad, bolumID)

Bolum Tablosu : tBolum(bolumID, bolumAd, fakulteID)

Fakulte Tablosu : tFakulte(fakulteID, fakulteAd)

Seçtiğiniz ORM Framework' ü kullanarak yine seçeceğiniz bir VTYS üzerinde bir veritabanı oluşturunuz. Seçeceğiniz bir yazılım platformunu kullanarak bir web uygulaması geliştirmeniz gerekiyor. Uygulamanızın index sayfasında tablolara veri girişi imkanı sağlayacak formlara yönlendirme yapınız. Her tablo için bir veri giriş formu tasarlayınız. Sorgulama işlemi için bir sorgulama formu tasarlayınız ve ogrenciID bilgisi girilen bir öğrenciyi veritabanından sorgulayarak [ad, soyad, bolumAd, fakulteAd] bilgilerini bir tablo içerisinde getiriniz.

Föy-4' de öğrendiğiniz web servisi yöntemini bu uygulama için de kullanmak adına öğrenci numarasına göre öğrenci bilgilerini ([ad, soyad, bolumAd, fakulteAd]) sorgulama imkanı sunan bir web servisi hazırlayınız. Bu web servisine öncelikle web uygulamanız üzerinden bağlantı sağlayarak sorgulama işlemini gerçekleştiriniz (Uygun bir form tasarlayarak).

Android bir uygulama geliştirerek yukarıda hazırladığınız web servisine bağlanarak benzer şekilde öğrenci numarası girilen kullanıcının bilgilerini android uygulama ekranına getiren bir program yazınız. <http://mobilekran.wordpress.com/2012/02/28/android-mysql-database-connection/> linkinden faydalanabilirsiniz.

Aynı uygulamayı kullandığınız VTYS' yi değiştirerek örneğin mySQL' den Oracle' a aktarma işlemini yaparak çalıştırmayı deneyiniz.

Uygulamanın index sayfası

Fakülte Ekle
Bölüm Ekle
Öğrenci Ekle
Öğrenci Sorgula
Öğrenci Sorgula (Web Servisi ile)

ORM nedir?

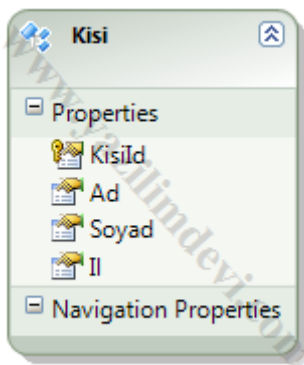
ORM, veritabanında oluşturulan her bir nesneye (tabloya) karşılık uygulama tarafında bir nesne oluşturma işidir. Bu işlem bazı Frameworklerde ara yazılımlar sayesinde (ORM Tools), bazı frameworklerde ise elle gerçekleştirilmektedir.

Bir örnek ile inceleyelim:

Veritabanındaki tablomuz : **Kisi** .

Kisi tablomuzun sütunları : **KisiId, Ad, Soyad, İl**

Veritabanındaki bu tabloya karşılık gelen sınıfımız:



ORM sayesinde veritabanına kayıt ekleme (INSERT), çekme (SELECT), düzenleme (UPDATE) işlemleri çok kolay yapılmaktadır. Bu işlemler direk ORM üzerinden gerçekleştirilmektedir. Bu yöntemin dışında katı SQL kodu yazarak da yapılabileceği gibi, ORM araçlarının kendi dilleri de entegre olabilmektedir. Buna en güzel örnek, Hibernate'teki HQL ve Microsoft'un runtime'da sorgu hatalarını en aza indirmeyi planladığı ve birçok noktada kolaylık sağlayan LINQ (Entity Framework).

ORM ile neler yapılabilir?

ORM sayesinde SQL sorgularıyla yapılan birçok işlem SQL sorgusu kullanılmadan gerçekleştirilmektedir. Örneğin veritabanından veri çekmek için SELECT sorgusu yerine oluşturmak yerine oluşturulan nesneden ID'si xxx olan eleman gelsin diyerek veritabanından o elemanı alabiliriz. Başka bir şekilde örneklersek, Ad'ı yyy olan elemanları Liste şeklinde getirmek istiyorsak "BU NESNEDEN Ad'ı yyy OLANLARI GETİR" diyerek SELECT sorgusu yazıp, sonra onu List'e atmayla uğraşmadan getirebiliyoruz.

ORM veritabanı bağımsız nasıl çalışıyor?

ORM'nin en önemli özelliği veritabanı bağımsız çalıştırabilmektir. Her ne kadar kimi ORM frameworkleri veritabanı bağımsız özelliğini bazı veritabanları ile SINIRLANDIRSA da başka bir framework ile istenen veritabanına bağlantılar gerçekleştirilebilmektedir. ORM'de tüm işlemler nesneler ve nesneler arası ilişkiler ve bu ilişkilerin UYGULAMA KATMANINDA tanımlanması sebebiyle veritabanından sadece TABLO ve tabloların ALANlarının istenen formatta olması yeterli olmaktadır. Örneğin Oracle ve SQL Server sunucularımızda birbirinin aynı veritabanları bulunmaktadır. Herhangi bir ORM aracıyla oluşturulan projenin Oracle'dan SQL Server'a geçirilmesi gerekmektedir. Dedğimiz gibi, projelerimiz veritabanından bağımsız, nesnelere bağımlı çalıştığı için veritabanının HANGİ VERİTABANI olduğu bilgisinin verilmesi yeterlidir. SQL Server'a geçiş yapılacaksa, bağlantı parametrelerinin SQL Server'a göre düzenlenmesi ile (config dosyasındaki küçük bir düzenleme olarak düşünülebilir) projemiz SQL Server'da çalışmaktadır.

ORM'nin avantajları nelerdir?

ORM'yi bu kadar anlattık, avantajlarından bahsedelim. En büyük avantajı Transaction Yönetimidir. Bir Transaction başlatıp başlattığımız transaction içerisinde birçok INSERT/UPDATE/DELETE işlemi gerçekleştirebiliriz. VERİTABANINA KAYDET demeden bu işlemler localde gerçekleşmiş olacaktır. Kayıt işlemini gerçekleştirdiğimiz anda (Transaction tamamlandığında) veritabanına etkileyecektir.

Yukarıdaki açıklamamızdan şu örnekleme de yapılabilir : Bir grup kayıt çektik, çektiğimiz kayıtları düzenledik, ekleme ve silme yaptık (localde). Birçok işlemi tamamladıktan sonra veritabanına etkilemesini sağlayabiliriz. Yani her işlemde veritabanına kaydet dememize gerek yoktur. Bu hem veritabanında yapılabilecek kayıt hatalarından, hem de sürekli veritabanı işlemlerini yapmaktan bizi kurtarmaktadır. Yani performansı artırmaktadır.

Performansı artırır dedik, performans konusunda şu soru sorulabilir : Veri locale alınıyor, işlemler yapılıyor, tekrar güncelleniyor. Bu performansı düşüren bir etken değil midir? Bir bakış açısından cevabı evet, eğer örneğin bir web sayfasının yönetim panelini yapıyorsak, sadece ürün ekleme silme düzenleme işlemleri yapılıyorsa böyle bir yapı performansı

düşürebilmektedir. Başka bir bakış açısından cevabı hayır, veritabanından kayıtları alıp üzerinde düzenlemeler yapma, düzenlemeleri veritabanına etkileme ve bu etkilemeden sonraki sonuçları görmemiz, gerektiğinde düzenlememiz gereken geniş çaplı işlemler gerekebilir. Bu işlemler için sürekli veritabanında INSERT/UPDATE/DELETE işlemleri gerçekleştirip çok sayıda Transaction oluşturmak yerine daha az Transaction ile veritabanından kayıtları alıp düzenlemeleri yapıp tek seferde düzenleme işlemini yapmak bu noktada performansı artırmaktadır diyebiliriz.

Bu açıklamalara dayanarak, kullanım sıklığı ve performans avantajları düşünülüp projede ORM Frameworkünün avantajları ve dezavantajlarını inceleyip kararı doğru vermek gerekmektedir.

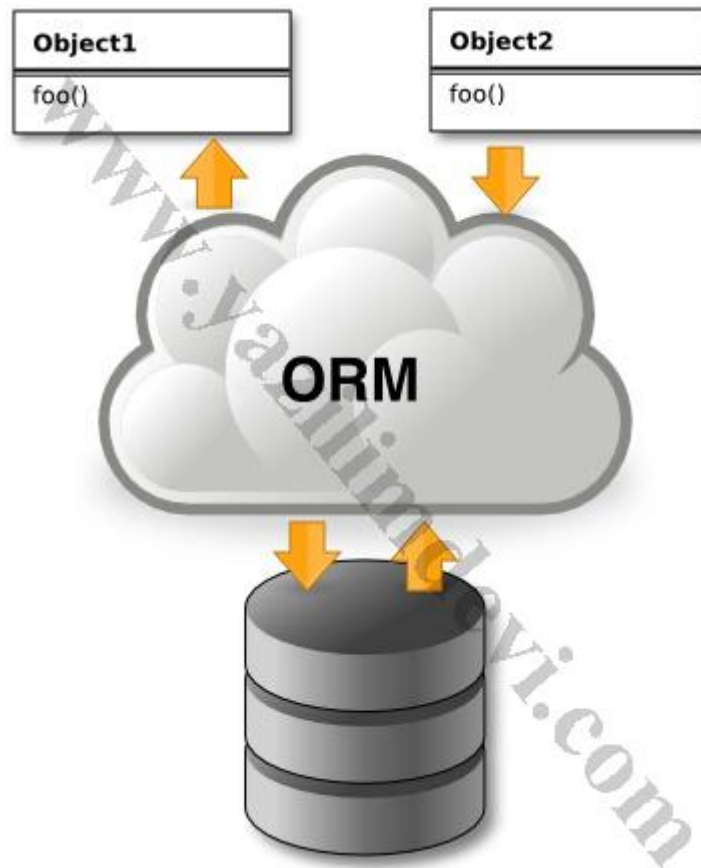
ORM ile Veritabanı Modelleme Zorunlu mu?

Bir başka avantajı da, veritabanı modellemeyi uygulama tarafında yapabilmeye izin vermesidir. Tablolar arası bağlantıları veritabanı katmanında yapabileceğimiz gibi uygulama katmanında da gerçekleştirebiliriz. Bunun avantajı ise, veritabanının başka bir sunucuya (SQL Server'dan Oracle'a gibi) aktarımında tablolararası ilişkileri yeni veritabanında da oluşturmak için zaman kaybetmeyi önlemektedir.

ORM'de Mapping Nedir?

Mapping, ORM'de veritabanı ile nesnelerimiz arasındaki bağı kuran yapımızdır. Hangi nesnenin (sınıfın) hangi tabloyla bağlanacağını, bağlanan tablolarda hangi property'nin (özellik ya da değişkenin) tablonun hangi alanıyla bağlanacağını, tablonun özelliklerini (ID'sinin ne olduğu, ID'sinin autoincrement olup olmadığı vb.) bilgilerin tanımlandığı yapımızdır. Kimi ORM frameworkleri bu işlemi yazılımcının yapmasını istese de (Hibernate, Nhibernate) kimi frameworkler bu işlemleri kendisi gerçekleştirmekte, istediği taktirde yazılımcının mapping'de düzenleme yapmasına izin vermektedir (Entity Framework). Aşağıdaki şekilde, Orm'nin yapısı daha net anlaşılmaktadır.

Object1 ve Object 2 isminde 2 sınıfımız (nesnemiz) bulunuyor.Bu sınıfları veritabanına bağlayan ise ORM 'dir.



ORM Araçları

1. Java için ORM frameworkleri:

- Hibernate
- JPA
- OpenJPA
- Toplink
- EclipseLink
- Apache Cayenne
- MyBatis

2. .Net için ORM frameworkleri

- Entity Framework
- Nhibernate
- .Net Persistence
- BBADataObjects-
- DataObjects.NET
- DotNorm
- FastObjects.NET

- Norm
- OJB.NET

3. PHP için ORM frameworkleri:

- Propel
- Doctrine
- PHP-Activerecord
- PdoMap
- RedBean

Java'da Nesne İlişkisel Eşleme (Object Relational Mapping) ve JPA Uygulaması

Adından da anlaşılacağı gibi nesne ilişkisel eşleme yani ORM, veri tabanı ile ilgili işlemleri kod tarafında nesneler (sınıflar) ile yönetebilmemizi sağlayan programlama tekniğidir. ORM ya da O/RM ya da O/R Mapping, Object Relational Mapping'in kısaltmasıdır. Bu teknik sayesinde programcı veri tabanı ile ilgili işlemler yaparken sorgu çekmekle yani SQL yazmakla uğraşmaz. Onun yerine sınıflar üzerinden işlemlerini gerçekleştirir.

Diğer dillerde olduğu gibi Java dilinde de birçok ORM kütüphanesi vardır. En çok kullanılanlardan bazıları şunlardır.

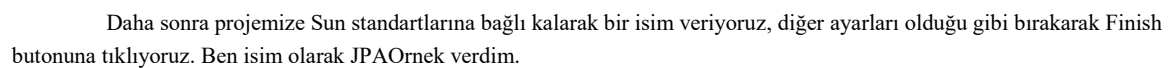
- JPA
- Hibernate
- OpenJPA
- Toplink
- EclipseLink
- Apache Cayenne

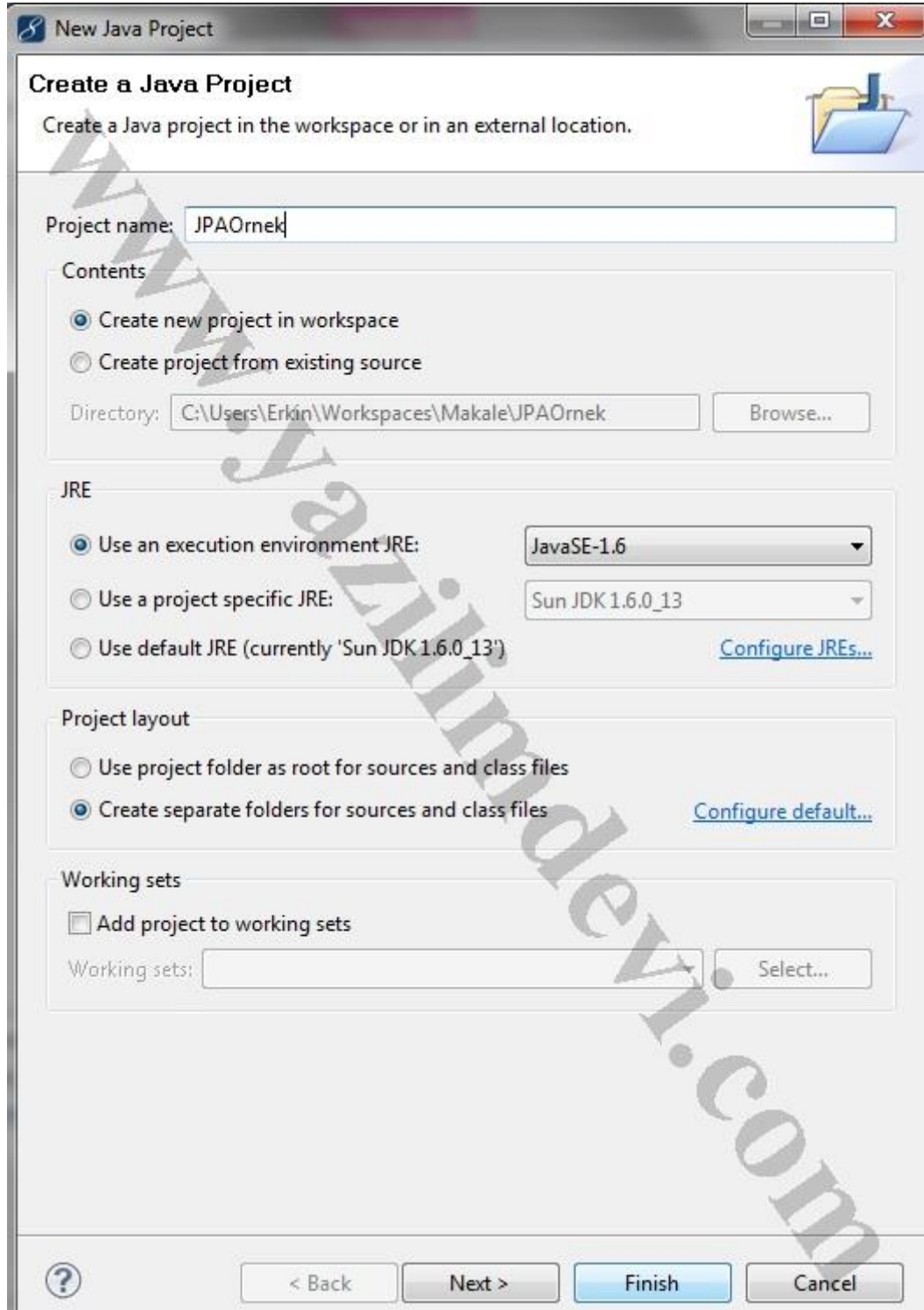
Veri tabanlarında yer alan tablo, kolon, tablolar arası ilişki (association) gibi kavramların yazılım tarafında birebir karşılıkları yoktur. Bu kavramların karşılıkları olarak örneğin tablo için sınıf, kolon için değişken gibi bileşenler kullanılır. Fakat sadece bu kadarı büyük projeler için yeterli olmayacaktır. Tablolar arası ilişkiler arttıkça ve karmaşılaşmaya başladıkça yazılımcının veri tabanından yapacağı sorgular da işin içinden çıkılmaz hale gelecektir. JDBC ile veri tabanına erişmek çok hızlı olmasına rağmen bahsettiğim sebeplerden ötürü büyük çaptaki projelerde ORM araçları sıklıkla kullanılır.

ORM kullanan bir programcının SQL ya da JDBC bilmesine gerek kalmaz. Kodunu geliştirirken daha kısa zamanda daha az kod yazarak veri tabanına bağlanabilir. Fakat bunun yanında SQL'ler çalışma anında dinamik olarak üretildiği için yavaş çalışır.

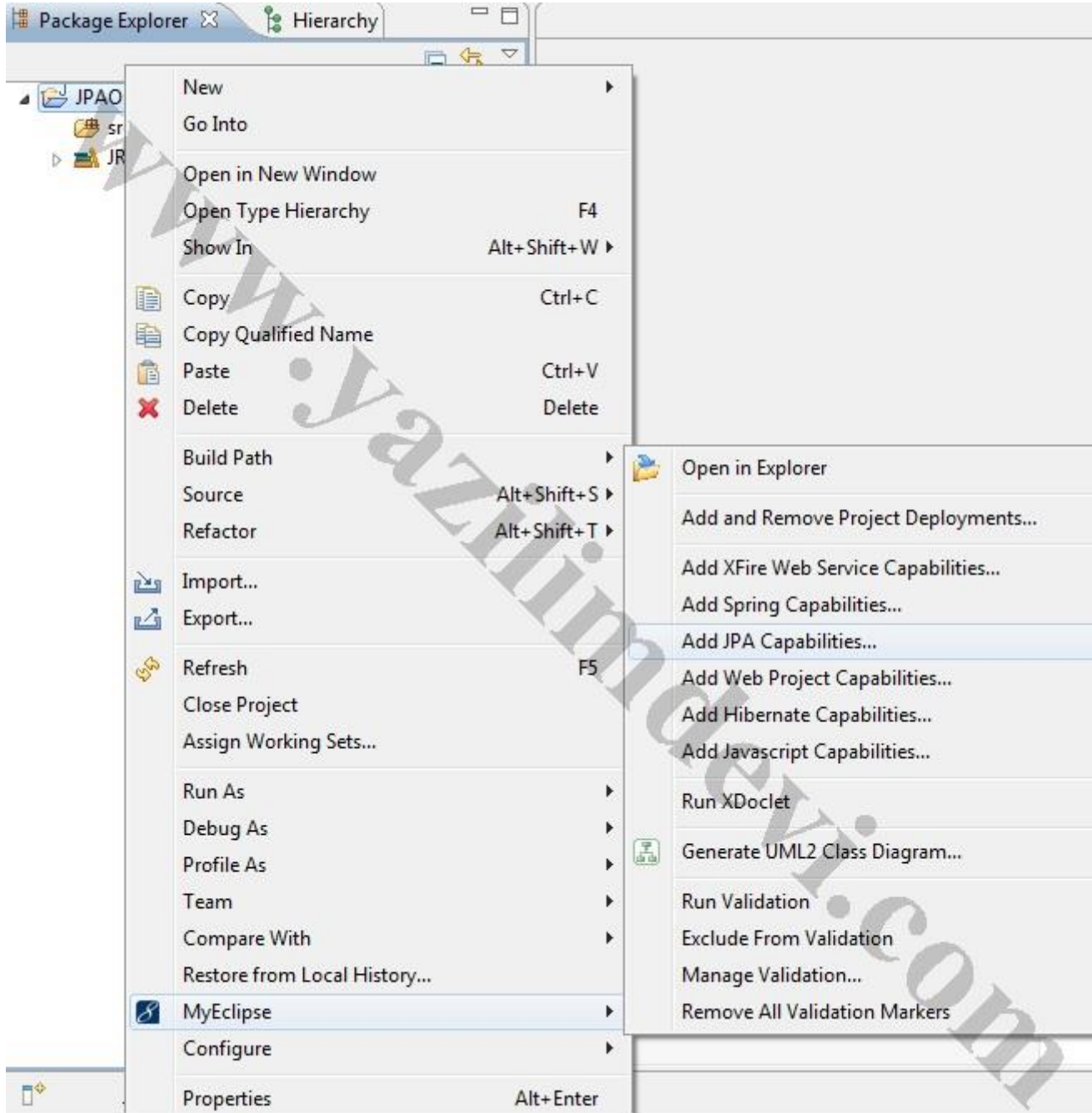
Şimdi JPA kullanarak veri tabanına basit bir insert işlemi gerçekleştiren uygulama yazalım. Bu uygulamayı MyEclipse ortamında Derby veri tabanı kullanarak yazacağız. Siz

Önce Package Explorer penceresinde New -> Java Project diyerek bir proje yaratıyoruz.

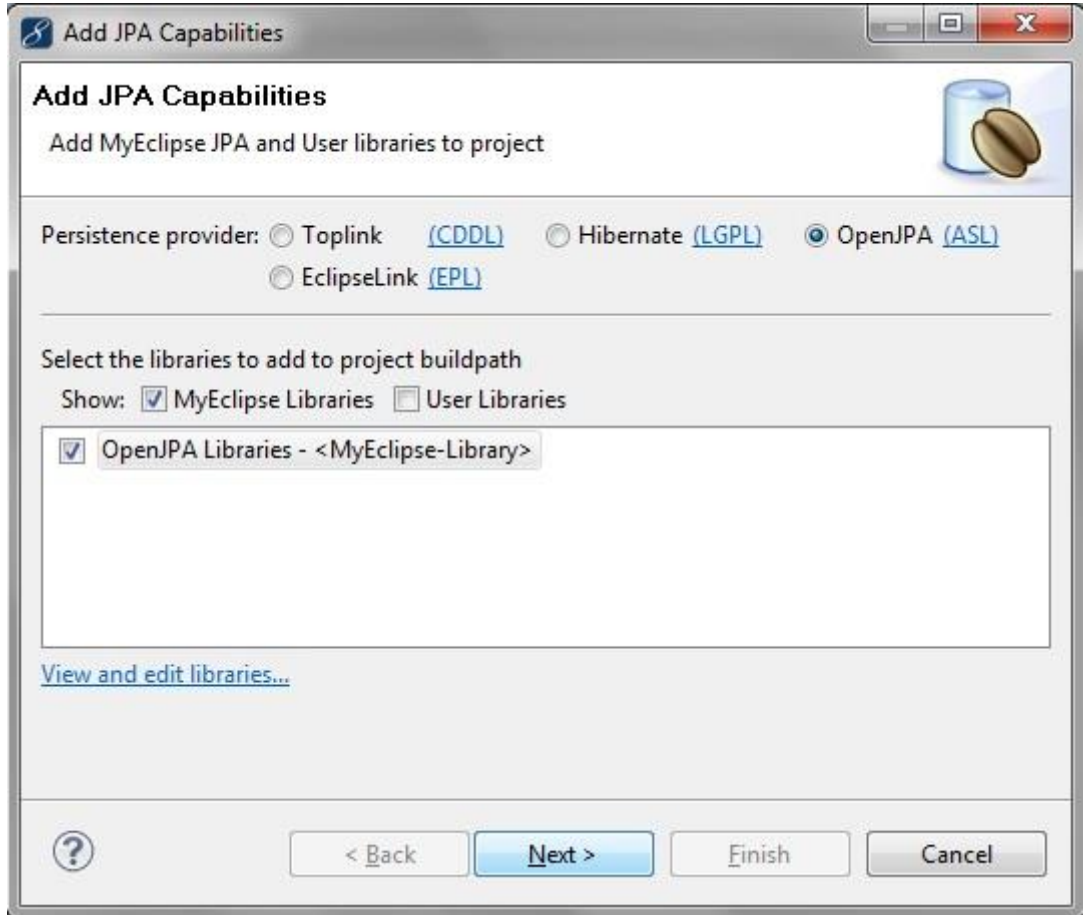




Projemiz sınıflarımızın bulunacağı src isimindeki bir dizin ve Java'nın standart kütüphaneleri ile yaratıldı. Şimdi projemizde JPA kullanacağımızı belirteceğiz. Bunun için proje üzerinde sağ klik, **MyEclipse** menüsünden **Add JPA Capabilities...** seçeneğine tıklıyoruz.



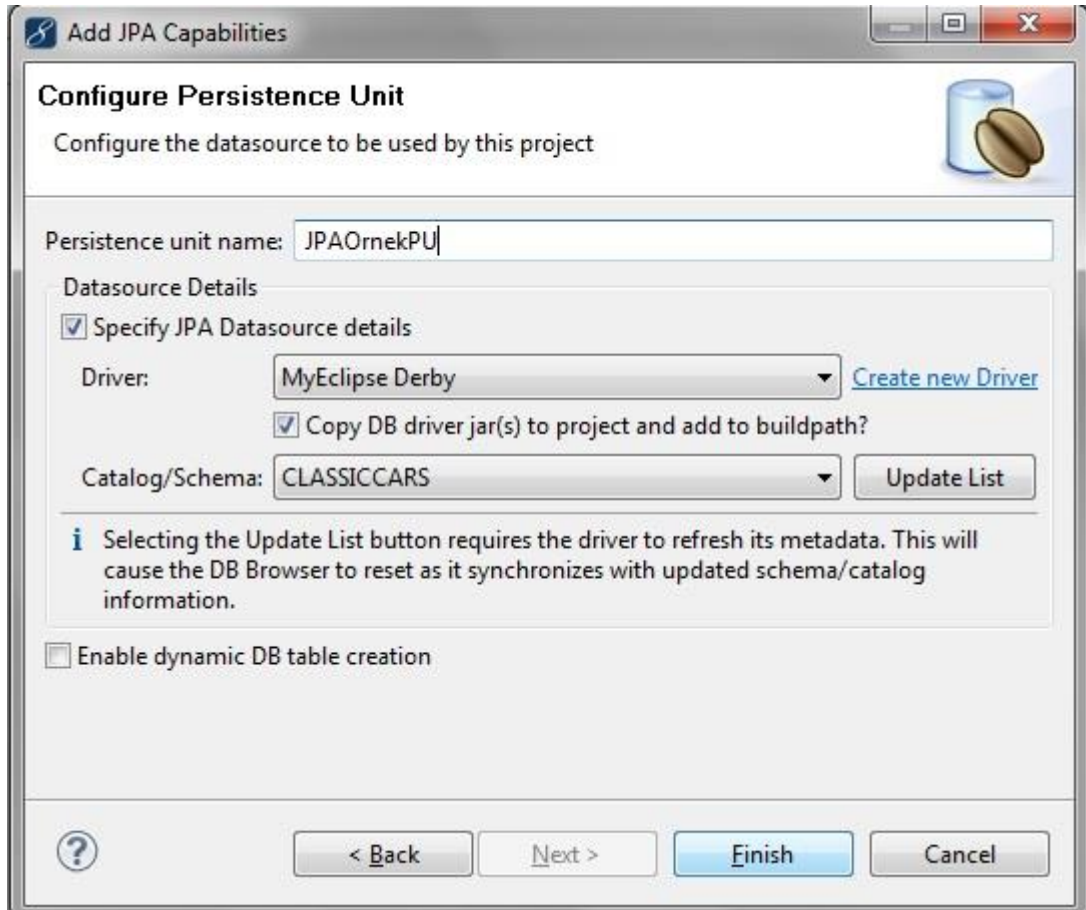
Açılan sayfada bize JPA'nın implementasyonları olan teknolojilerden hangisini kullanmak istediğimizi soruyor. Biz bu uygulama için **OpenJPA** seçelim ve Next diyelim. Bütün teknolojilerin kullanımı birbirine çok benzer. Siz isterseniz **Hibernate** ya da **Toplink** teknolojilerinden birini de seçebilirsiniz.



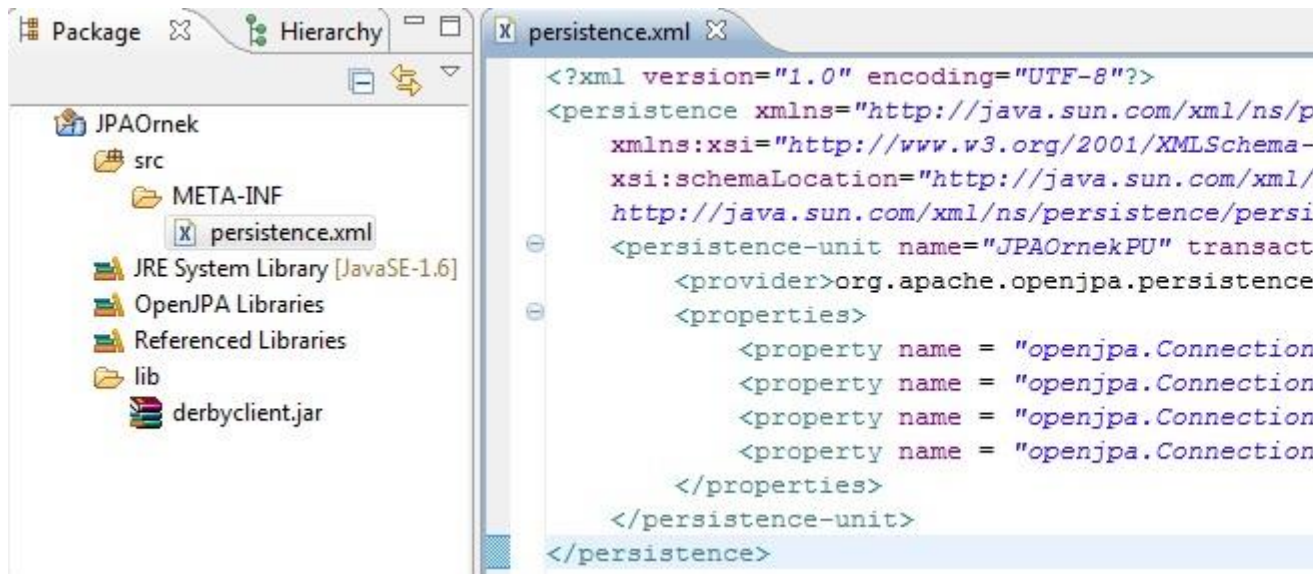
Açılan sayfada bağlanılacak olan veri tabanı hakkında driver bilgisi ile schema bilgisini gireceğiz. Ben MyEclipse içerisinde hazır gelen Derby'yi kullanacağım için onun driver'ını seçiyorum. SQL Server, Oracle ya da başka bir veri tabanına bağlanmak istiyorsanız yan taraftaki **Create new driver** linkinden yeni bir driver tanımlayabilirsiniz.

Bu sayfada ayrıca **Persistence unit name** bilgisini girmemiz gerekiyor. Standart olarak proje isminin sonuna PU eklenmiş ama istersek değiştirebiliriz. Bu isim daha sonra **Entity Manager** yaratırken kullanılacağı için önemlidir.

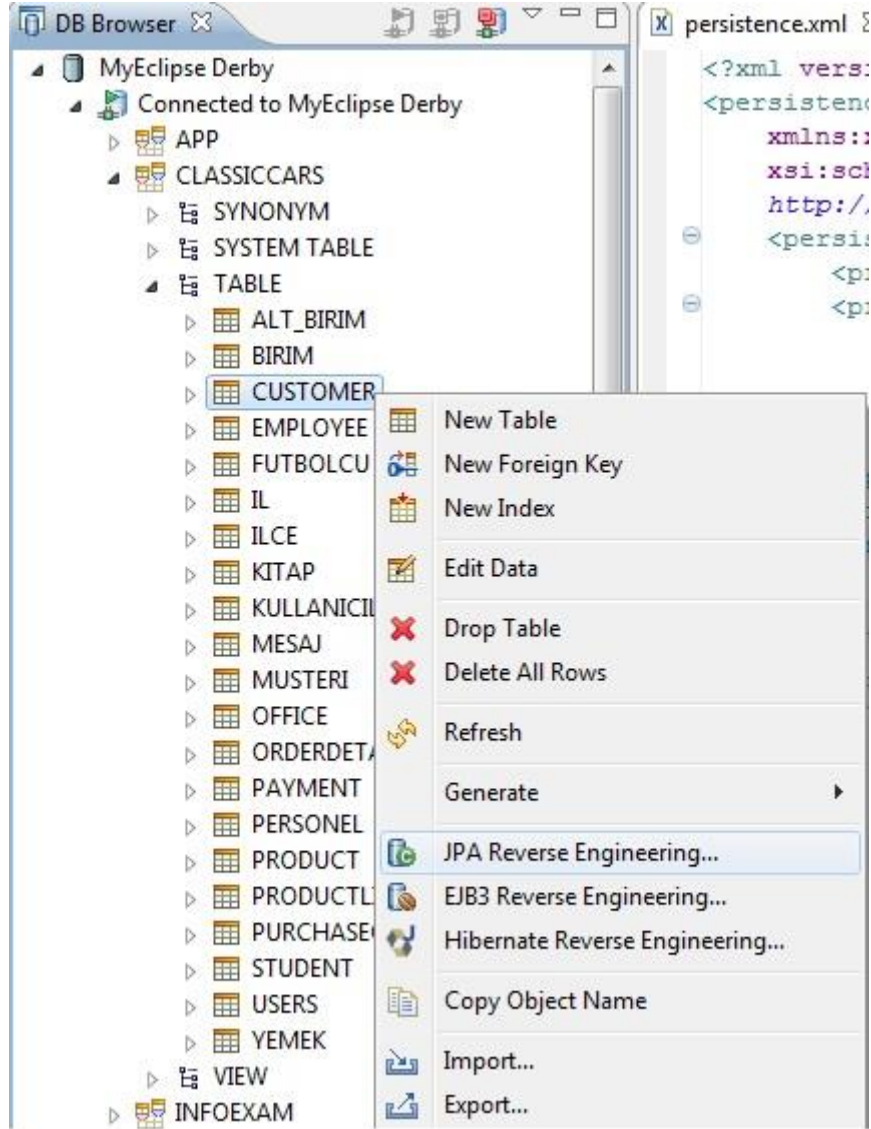
Driver bilgisinin hemen altında **Copy DB driver jar(s)** ile başlayan bir seçenek var. Bunu işaretlediğimiz takdirde projemizin veri tabanına bağlanabilmek için ihtiyaç duyduğu **client driver** otomatik olarak projeye eklenecektir.



Finish butonun tıkladığımızda **OpenJPA** kütüphaneleri, **clientDriver.jar** projeye eklenmiş ve **META-INF** adlı özel dizin içerisinde **persistence.xml** adında bir dosya oluşturulmuştur.



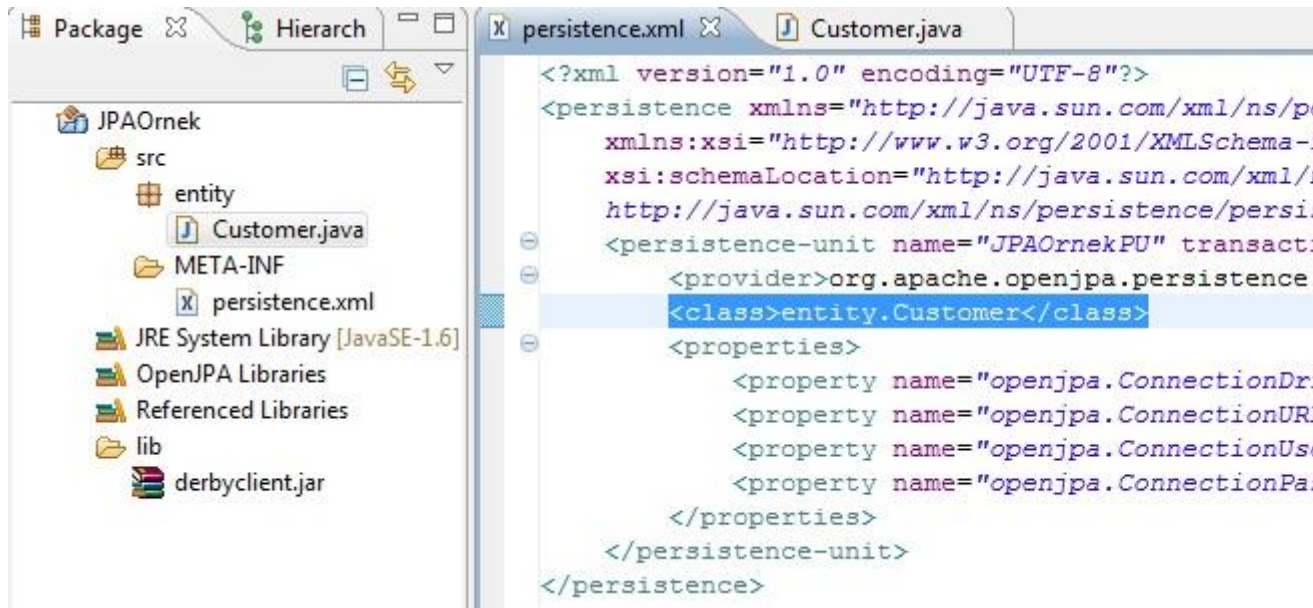
Bu dosya içerisinde projenin bağlandığı veri tabanı hakkında **driver**, **connection-url** gibi bilgiler yer almaktadır. Daha sonra veri tabanına karşılık gelen **entity** sınıflarımızı da yine bu dosya içerisine tanımlayacağız. Entity sınıflarımızı elle yazabildiğimiz gibi otomatik olarak yaratmak da mümkündür. Bunun için **MyEclipse Database Explorer** perspektifine geçerek, entity sınıfını yaratacağımız tablonun üzerinde sağ klik **JPA Reverse Engineering...** seçeneğine tıklıyoruz.



Açılan pencerede entity sınıfımızı nereye yaratmak istediğimizi belirtiyoruz. Ben daha önceden yaratmış olduğum **entity** paketi altına yaratılsın diyorum. Yaratılırken de persistence.xml dosyası güncellensin seçeneğini işaretliyorum. Bu sayede **persistence.xml** dosyasına entity sınıfımız otomatik olarak tanımlanmış oluyor.



Finish dediğimizde entity paketi altında seçtiğimiz tablo ile aynı isimde bir sınıf yaratılarak bu sınıfın referansı persistence.xml dosyasına yazılmış oluyor.



Örnek olarak kullandığımız Customer tablosunun tüm kolonları için Customer sınıfında birer değişken yaratıldı.

```

1. package entity;
2.
3. import javax.persistence.Column;
4. import javax.persistence.Entity;
5. import javax.persistence.Id;
6. import javax.persistence.Table;
7.
8. /**
9.  * Customer entity. @author MyEclipse Persistence Tools
10. */
11. @Entity
12. @Table(name = "CUSTOMER", schema = "CLASSICCARS")
13. public class Customer implements java.io.Serializable {
14.
15.     // Fields
16.
17.     private static final long serialVersionUID = -4279273036023102468L;
18.     private Integer customernumber;
19.     private String customername;
20.     private String contactlastname;
21.     private String contactfirstname;
22.     private String phone;
23.     private String addressline1;
24.     private String addressline2;
25.     private String city;
26.     private String state;
27.     private String postalcode;
28.     private String country;
29.     private Integer salesrepemployeenumber;
30.     private Double creditlimit;
31.
32.     // Constructors
33.
34.     /** default constructor */
35.     public Customer() {

```

```

36.     }
37.
38.     /** minimal constructor */
39.     public Customer(Integer customernumber) {
40.         this.customernumber = customernumber;
41.     }
42.
43.     /** full constructor */
44.     public Customer(Integer customernumber, String customername,
45.         String contactlastname, String contactfirstname, String phone,
46.         String addressline1, String addressline2, String city,
47.         String state, String postalcode, String country,
48.         Integer salesrepemployeenumber, Double creditlimit) {
49.         this.customernumber = customernumber;
50.         this.customername = customername;
51.         this.contactlastname = contactlastname;
52.         this.contactfirstname = contactfirstname;
53.         this.phone = phone;
54.         this.addressline1 = addressline1;
55.         this.addressline2 = addressline2;
56.         this.city = city;
57.         this.state = state;
58.         this.postalcode = postalcode;
59.         this.country = country;
60.         this.salesrepemployeenumber = salesrepemployeenumber;
61.         this.creditlimit = creditlimit;
62.     }
63.
64.     // Property accessors
65.     @Id
66.     @Column(name = "CUSTOMERNUMBER", unique = true, nullable = false)
67.     public Integer getCustomernumber() {
68.         return this.customernumber;
69.     }
70.
71.     public void setCustomernumber(Integer customernumber) {
72.         this.customernumber = customernumber;
73.     }
74.
75.     @Column(name = "CUSTOMERNAME", length = 50)
76.     public String getCustomername() {
77.         return this.customername;
78.     }
79.
80.     public void setCustomername(String customername) {
81.         this.customername = customername;
82.     }
83.
84.     @Column(name = "CONTACTLASTNAME", length = 50)
85.     public String getContactlastname() {
86.         return this.contactlastname;
87.     }
88.
89.     public void setContactlastname(String contactlastname) {
90.         this.contactlastname = contactlastname;
91.     }
92.
93.     @Column(name = "CONTACTFIRSTNAME", length = 50)
94.     public String getContactfirstname() {
95.         return this.contactfirstname;
96.     }
97.
98.     public void setContactfirstname(String contactfirstname) {
99.         this.contactfirstname = contactfirstname;
100.    }
101.

```



```

102. @Column(name = "PHONE", length = 50)
103. public String getPhone() {
104.     return this.phone;
105. }
106.
107. public void setPhone(String phone) {
108.     this.phone = phone;
109. }
110.
111. @Column(name = "ADDRESSLINE1", length = 50)
112. public String getAddressline1() {
113.     return this.addressline1;
114. }
115.
116. public void setAddressline1(String addressline1) {
117.     this.addressline1 = addressline1;
118. }
119.
120. @Column(name = "ADDRESSLINE2", length = 50)
121. public String getAddressline2() {
122.     return this.addressline2;
123. }
124.
125. public void setAddressline2(String addressline2) {
126.     this.addressline2 = addressline2;
127. }
128.
129. @Column(name = "CITY", length = 50)
130. public String getCity() {
131.     return this.city;
132. }
133.
134. public void setCity(String city) {
135.     this.city = city;
136. }
137.
138. @Column(name = "STATE", length = 50)
139. public String getState() {
140.     return this.state;
141. }
142.
143. public void setState(String state) {
144.     this.state = state;
145. }
146.
147. @Column(name = "POSTALCODE", length = 15)
148. public String getPostalcode() {
149.     return this.postalcode;
150. }
151.
152. public void setPostalcode(String postalcode) {
153.     this.postalcode = postalcode;
154. }
155.
156. @Column(name = "COUNTRY", length = 50)
157. public String getCountry() {
158.     return this.country;
159. }
160.
161. public void setCountry(String country) {
162.     this.country = country;
163. }
164.
165. @Column(name = "SALESREPEMPLYEENUMBER")
166. public Integer getSalesrepemployeenumber() {
167.     return this.salesrepemployeenumber;

```

```

168. }
169.
170. public void setSalesrepemployeenumber(Integer salesrepemployeenumber) {
171.     this.salesrepemployeenumber = salesrepemployeenumber;
172. }
173.
174. @Column(name = "CREDITLIMIT", precision = 52, scale = 0)
175. public Double getCreditlimit() {
176.     return this.creditlimit;
177. }
178.
179. public void setCreditlimit(Double creditlimit) {
180.     this.creditlimit = creditlimit;
181. }
182. }

```

Son olarak da veri tabanına insert işlemini yapacak olan bir test sınıfı yazalım.

```

1. package test;
2.
3. import javax.persistence.EntityManager;
4. import javax.persistence.EntityManagerFactory;
5. import javax.persistence.EntityTransaction;
6. import javax.persistence.Persistence;
7.
8. import entity.Customer;
9.
10. public class Test {
11.     public static void main(String[] args) {
12.         EntityManagerFactory entityManagerFactory = Persistence.createEntityManagerFactory("JPAOrnekPU");
13.         EntityManager em = entityManagerFactory.createEntityManager();
14.         EntityTransaction userTransaction = em.getTransaction();
15.
16.         userTransaction.begin();
17.         Customer customer = new Customer();
18.         customer.setAddressline1("Yenimahalle");
19.         customer.setAddressline2("Çayyolu");
20.         customer.setCity("Ankara");
21.         customer.setContactfirstname("Tuba");
22.         customer.setContactlastname("Pehlivan");
23.         customer.setCountry("Türkiye");
24.         customer.setCreditlimit(9700.00);
25.         customer.setCustomername("Erkin");
26.         customer.setCustomernumber(111111);
27.         customer.setPhone("312899423");
28.         customer.setPostalcode("06810");
29.         customer.setSalesrepemployeenumber(12);
30.         customer.setState("İç Anadolu");
31.         em.persist(customer);
32.         userTransaction.commit();
33.         em.close();
34.         entityManagerFactory.close();
35.     }
36. }

```

EntityManagerFactory yaratırken kullandığımız `createEntityManagerFactory()` metoduna parametre olarak `persistence.xml` dosyasında tanımladığımız `persistence unit name` değerini veriyoruz. Daha sonra factory üzerinden EntityManager instance'ı oluşturarak transaction alıyoruz. Transaction'ı başlatıp entity sınıfımızın değişkenlerini set ediyoruz. Farkettiğiniz gibi

Customer tablosu ile hiç uğraşmadık. Sadece entity sınıfımızın değişkenlerini set ettik. Arka tarafta JPA, [JDBC](#) alt yapısını kullanarak veri tabanına bağlandı ve yeni kayıt ekledi.

Microsoft Entity Framework Kullanılarak ORM uygulama örneği

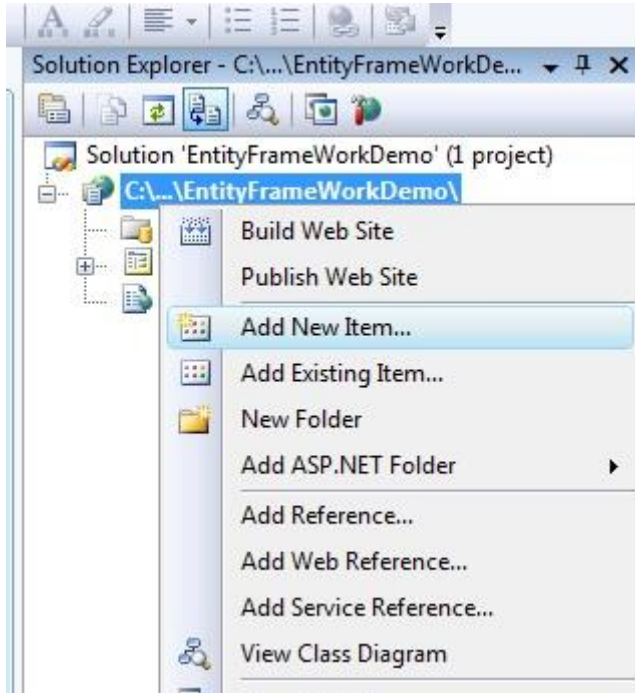
Bu videoyu izleyiniz:

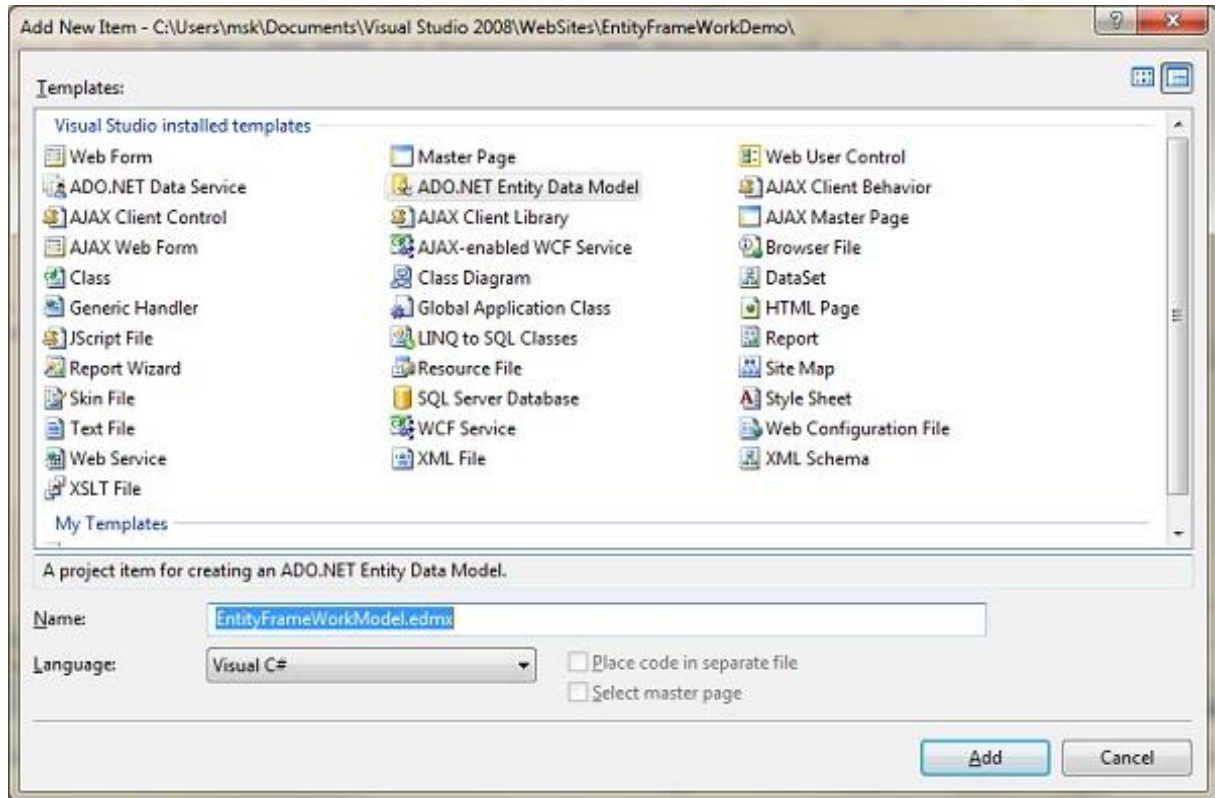
<http://www.yazilimdevi.com/yazilimdevi/Video.aspx?id=858&VideoUrl=./Videos/EntityFramework/EntityFrameworkGiris1.swf&yid=29>

Aşağıdaki makaleyi okuyunuz.

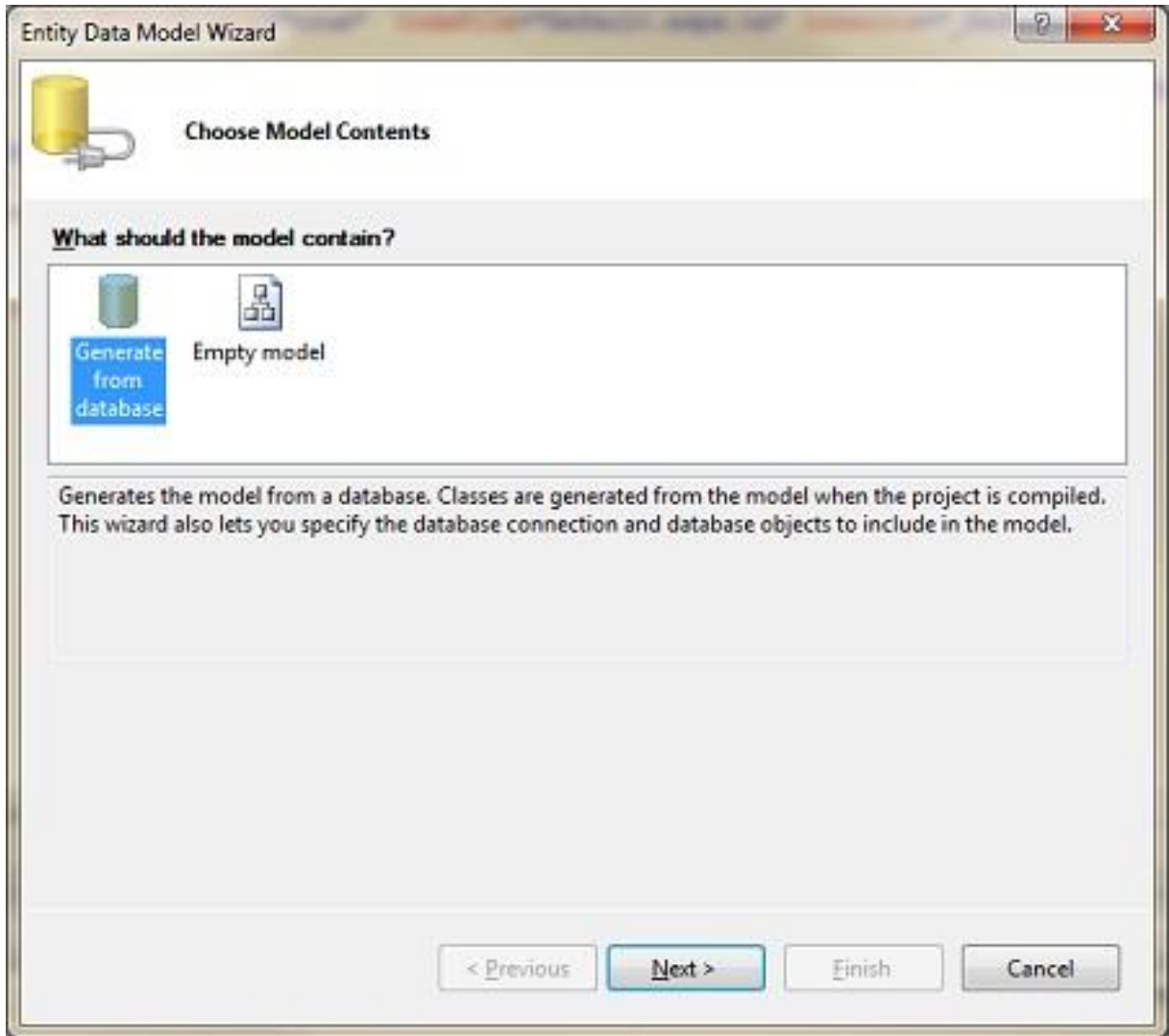
Bu makalemizde Microsoftun ORM araçlarından Entity Framework yapısını kullanarak bir data modeli oluşturacağız ve bu model ile veritabanından nasıl select işlemi yapacağımızı öğreneceğiz.

Entity Framework hakkında teorik bilgileri internette bolca bulabilirsiniz bu yüzden bu konulara değinmeyi düşünmüyorum. İlk olarak visual studio 2008 veya 2010 kullanarak yeni bir websitesi oluşturalım. Oluşturduğumuz web sitesine aşağıdaki gibi bir “entity framework data model” ekleyelim.





Yukarıda gördüğünüz gibi ADO.NET Entity Data Model'i seiyoruz ve bir isim veriyoruz. Daha sonra add deyip diğer adıma geçiyoruz.



Biz bu örneğimizde modelimizi var olan bir veritabanı üzerinden oluşturacağız. Tam aksine önce modeli oluşturup EntityFramework'ün veritabanını kendisinin oluşturmasını sağlayabiliriz. Next dedikten sonra gelen ekranda “new connection” deyip aşağıdaki ekranda gördüğümüz gibi var olan bir veritabanı için connection oluşturuyoruz.

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server name:
MSK-PC Refresh

Log on to the server

☒ Use Windows Authentication
☐ Use SQL Server Authentication

User name:
Password:
☐ Save my password

Connect to a database

☒ Select or enter a database name:
AdventureWorksDW2008 ▼


☐ Attach a database file:
 Browse...
Logical name:

Advanced...

Test Connection OK Cancel

Örneğimizde AdventureWorks veritabanını kullanıyoruz.

Entity Data Model Wizard

 Choose Your Data Connection

Which data connection should your application use to connect to the database?

msk-pc.AdventureWorksDW2008.dbo New Connection...

This connection string appears to contain sensitive data (for example, a password) that is required to connect to the database. Storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

☐ No, exclude sensitive data from the connection string. I will set it in my application code.

☐ Yes, include the sensitive data in the connection string.

Entity connection string:

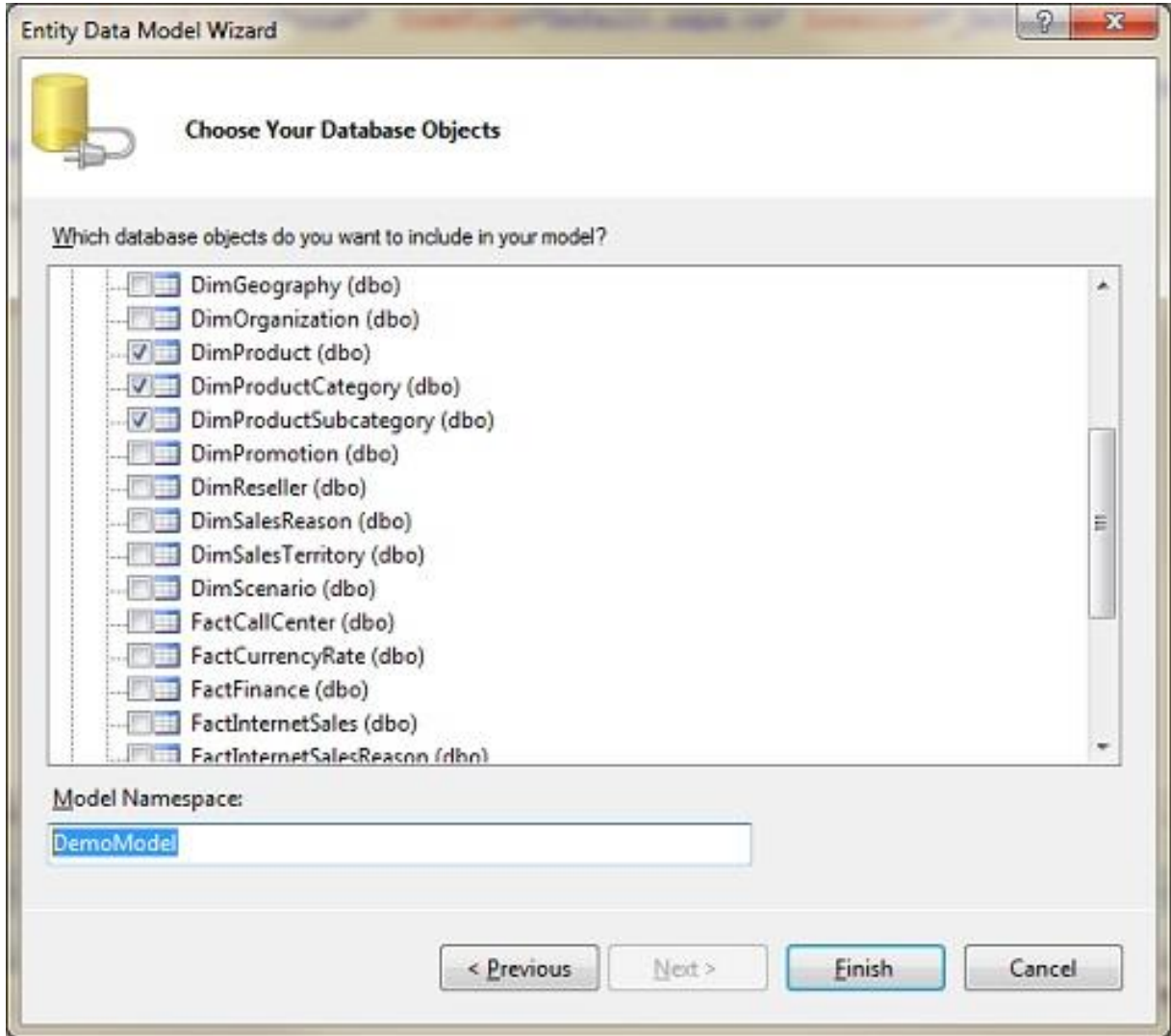
metadata=res://";provider=System.Data.SqlClient;provider connection string="Data Source=MSK-PC;Initial Catalog=AdventureWorksDW2008;Integrated Security=True"

☒ Save entity connection settings in Web.Config as:

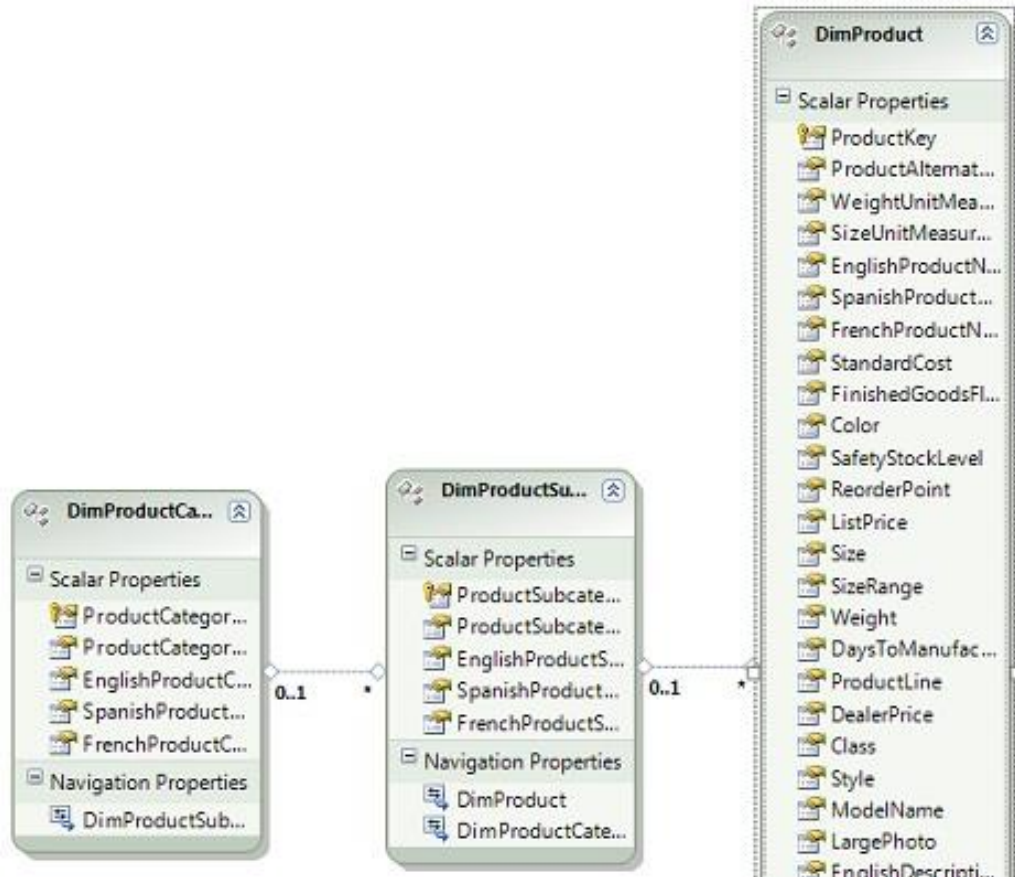
DemoEntities

< Previous Next > Finish Cancel

Bağlantı ayarlarımız yaptığımız zaman yukarıdaki ekranda bu bağlantı bilgisi için web.config dosyasında kullanılacak ismi belirleyip next diyoruz.



Gelen yukarıdaki ekranda “Tables” sekmesini seçip kullanacağımız tabloları seçiyoruz. Biz örneğimizde Product , productCategory ve productSubCategory tablolarını kullanıyoruz. Bu adımda Finish dediğimiz zaman modelimiz kullanılmaya hazır hale gelecek. Modelimizin görüntüsü aşağıdaki gibi olacaktır.



Şimdi gelelim bu modeli kullanarak veritabanından nasıl veri çekeceğimize.

Default.aspx sayfasına bir iki adet dropdownlist ve bir adette gridview ekleyelim. Sayfamızın html kısmı aşağıdaki gibi olacak.

```
<table>
<tr>
<td>
<asp:DropDownList ID="KategoriDropDownList" AutoPostBack="true"runat="server" OnSelectedIndexChanged
="KategoriDropDownList_SelectedIndexChanged">
<asp:DropDownList>
</td>
<td>
<asp:DropDownList ID="AltKategoriDropDownList" runat="server">
<asp:DropDownList>
</td>
<td>
<asp:Button ID="UrunleriGetirButton" runat="server" Text="Ürünleri
Getir" OnClick="UrunleriGetirButton_Click" />
</td>
</tr>
<tr>
<td colspan="3">
```

```

        <asp:GridView ID="UrunlerGridView" runat="server">
        </asp:GridView>
    </td>
</tr>
</table>

```

Şimdi ilk olarak sayfa yüklenirken kategori dropdownlistini dolduralım. Kategori seçilince alt kategorileri dolduralım.

İlk olarak kategorileri doldurmak için aşağıdaki kodu yazın.

```

protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        KategorileriGetir();
    }
}

private void KategorileriGetir()
{
    DemoEntities entity = new DemoEntities();
    List<DimProductCategory> kategoriList = entity.DimProductCategory.ToList();
    KategoriDropDownList.DataSource = kategoriList;
    KategoriDropDownList.DataValueField = "ProductCategoryKey";
    KategoriDropDownList.DataTextField = "EnglishProductCategoryName";
    KategoriDropDownList.DataBind();
}

```

İkinci olarak kategori seçildiği zaman aşağıdaki kodu yazarak alt kategorileri dolduralım.

```

protected void KategoriDropDownList_SelectedIndexChanged(object sender, EventArgs e)
{
    int KategoriKey = Int32.Parse(KategoriDropDownList.SelectedValue);
    DemoEntities entity = new DemoEntities();

    var query = from c in entity.DimProductSubcategory
                where c.DimProductCategory.ProductCategoryKey == KategoriKey
                select c;

    AltKategoriDropDownList.DataSource = query.ToList(); ;
    AltKategoriDropDownList.DataValueField = "ProductSubcategoryKey";
    AltKategoriDropDownList.DataTextField = "EnglishProductSubcategoryName";
    AltKategoriDropDownList.DataBind();
}

```

Yukarıdaki adımları tamamladığımız zaman kategori ve alt kategori alanları ile olan işlemimiz bitmiş oluyor. Şimdi ise ürünleri getir butonuna basınca seçilen kategorideki ürünleri alıp gridview içine atalım.

```

protected void UrunleriGetirButton_Click(object sender, EventArgs e)
{

```

```

int AltKategoriKey = Int32.Parse(AltKategoriDropDownList.SelectedValue);
DemoEntities entity = new DemoEntities();

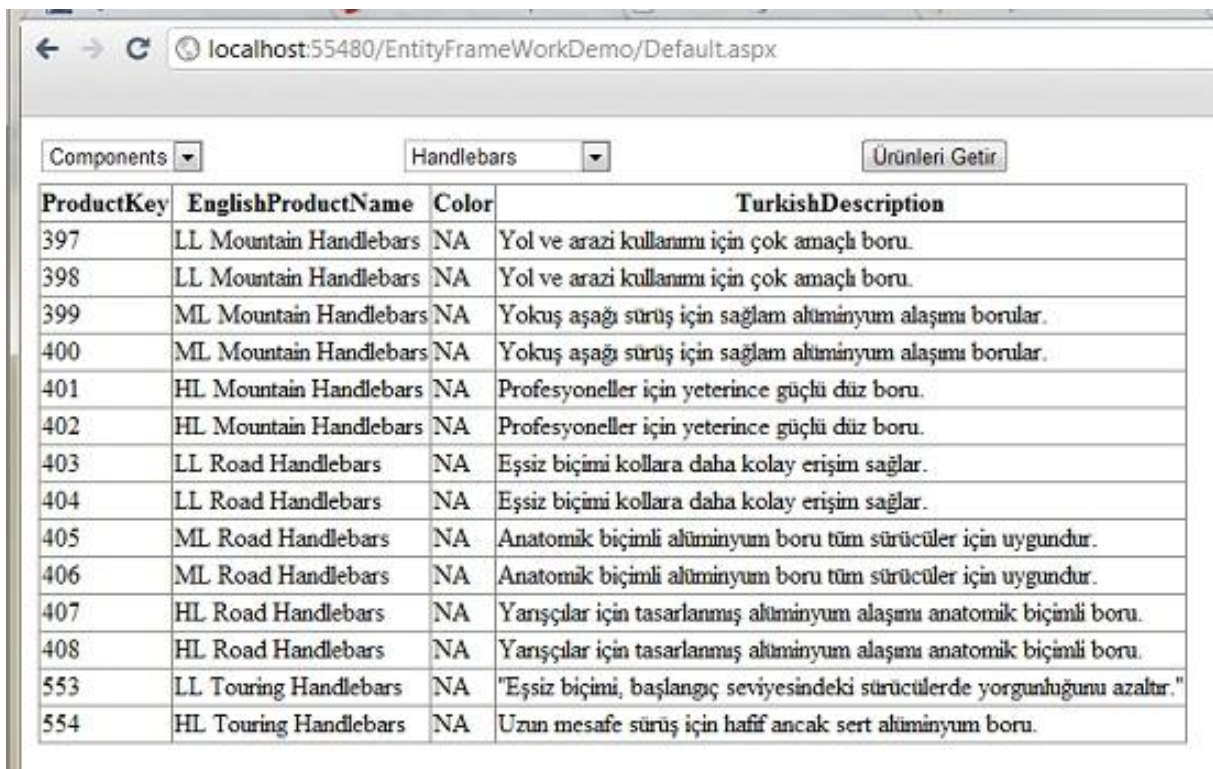
var query = from p in entity.DimProduct
            where p.DimProductSubcategory.ProductSubcategoryKey == AltKategoriKey
            select new { p.ProductKey, p.EnglishProductName, p.Color, p.TurkishDescription };

UrunlerGridView.DataSource = query.ToList();
UrunlerGridView.DataBind();
}

```

Yukarıdaki kodu inceleyecek olursanız product tablosunda sadece istediğimiz kolonları getirdik, bunun içinde new deyip yeni bir tanımlama yaptık.

Sonuç olarak işlemlerin sonucunda aşağıdaki gibi bir ekran çıktımız oldu.



ProductKey	EnglishProductName	Color	TurkishDescription
397	LL Mountain Handlebars	NA	Yol ve arazi kullanımı için çok amaçlı boru.
398	LL Mountain Handlebars	NA	Yol ve arazi kullanımı için çok amaçlı boru.
399	ML Mountain Handlebars	NA	Yokuş aşağı sürüş için sağlam alüminyum alaşımı borular.
400	ML Mountain Handlebars	NA	Yokuş aşağı sürüş için sağlam alüminyum alaşımı borular.
401	HL Mountain Handlebars	NA	Profesyoneller için yeterince güçlü düz boru.
402	HL Mountain Handlebars	NA	Profesyoneller için yeterince güçlü düz boru.
403	LL Road Handlebars	NA	Eşsiz biçimi kollara daha kolay erişim sağlar.
404	LL Road Handlebars	NA	Eşsiz biçimi kollara daha kolay erişim sağlar.
405	ML Road Handlebars	NA	Anatomik biçimli alüminyum boru tüm sürücüler için uygundur.
406	ML Road Handlebars	NA	Anatomik biçimli alüminyum boru tüm sürücüler için uygundur.
407	HL Road Handlebars	NA	Yarışçılar için tasarlanmış alüminyum alaşımı anatomik biçimli boru.
408	HL Road Handlebars	NA	Yarışçılar için tasarlanmış alüminyum alaşımı anatomik biçimli boru.
553	LL Touring Handlebars	NA	"Eşsiz biçimi, başlangıç seviyesindeki sürücülerde yorgunluğunu azaltır."
554	HL Touring Handlebars	NA	Uzun mesafe sürüş için hafif ancak sert alüminyum boru.

Propel ORM Framework ile ilgili olarak

<http://propelorm.org/> adresini ziyaret ediniz. Gerekli örnek ve dökümanlardan faydalanabilirsiniz.

Diğer bir PHP ORM olan Doctrine ORM ile ilgili olarak <http://marco-pivetta.com/doctrine-orm-zf2-tutorial/#/4> adresini ziyaret ediniz.