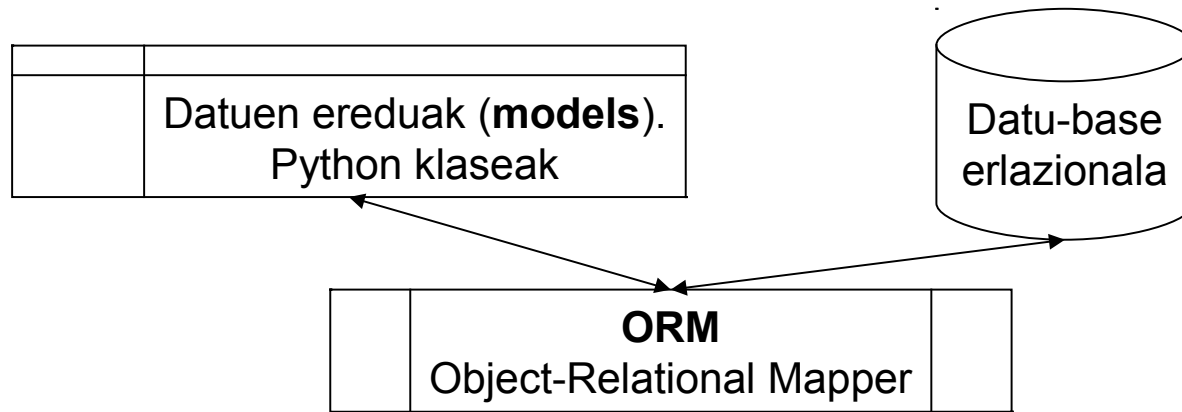


Django

Ereduggeruza (*model layer*)

Eredu-geruza (model layer)

- Eredu geruzan datuen errepresentazioa eta datuekiko interfazea lantzen da.





Model klaseak

- Eredu-geruzan *ereduak* definitzen dira.
- Eredu horiek aplikazioko datuak definitzeko oinarritzko baliabideak dira.
 - Eredu bakoitza Python klase bat da, *django.db.models.Model* gainklasetik heredatzen dena.
 - Eredu bakoitza datu-baseko taula batekin parekatu ohi da.
 - Ereduek datuen eremuak eta portaera definitzen dituzte. Ereduko atributuek datu-baseko eremuak errepresentatzen dituzte.
 - Modu honetan, Django datu-basea atzitzeko APIa eskaintzen du.



Model klaseak Adibidea

- models.py fitxategian:

```
from django.db import models

class Pertsona(models.Model):
    izena = models.CharField(max_length=30)
    abizena = models.CharField(max_length=30)
```

- Pertsona ereduak bi eremu (*field*) ditu: izena eta abizena.
- Honako datu-baseko taula sortuko luke:

```
CREATE TABLE aplikazioa_pertsona (
    "id" serial NOT NULL PRIMARY KEY,
    "izena" varchar(30) NOT NULL,
    "abizena" varchar(30) NOT NULL
);
```



Ereduak erabiltzeko

```
nireproiektua/  
manage.py  
nireproiektua/  
    __init__.py  
    settings.py  
    urls.py  
    wsgi.py
```

```
aplikazioa/  
    __init__.py  
    models.py  
    tests.py  
    views.py
```

```
$ python manage.py syncdb
```



```
INSTALLED_APPS = (  
    #...  
    'aplikazioa',  
    #...  
)
```



Eremuak (*fields*)

- Eredu baten definizioiko osagai beharrezkoak dira eremuak.
- Klase-atributuen bidez adierazten dira.

```
class Musikari(models.Model):  
    izena = models.CharField(max_length=50)  
    abizena = models.CharField(max_length=50)  
    instrumentua = models.CharField(max_length=100)  
  
class Album(models.Model):  
    musikari = models.ForeignKey(Musikari)  
    izenburua = models.CharField(max_length=100)  
    argitaratze_data = models.DateField()  
    izar_kopurua = models.IntegerField()
```



Eremuen motak

- Eremu bakoitza *Field* klaseko instantzia izango da.
- Django eremuen motak erabiltzen ditu, honakoetarako:
 - Datu-basearen *zutabearen* mota finkatzeko (INTEGER, VARCHAR...).
 - Eremu horiek formularioen bidez maneiatzeko, nolako kontrolak erabiliko diren erabakitzeko (<input type="text">, <select>...).
 - Automatikoki sortutako formularioetan, balidazio-betebehar gutxienekoak zehazteko.



Eremuen motak (II)

- Django hainbat eremu mota eskaintzen ditu, besteak beste:
 - **BooleanField** (true/false)
 - **CharField** (string). `CharField.max_length` argumentua eskatzen du.
 - **DateField** (data, *datetime.date* klaseko instantzia)
 - **DateTimeField** (data eta denbora, *datetime.datetime* klaseko instantzia)
 - **EmailField** (CharField email balidazioarekin)
 - **FileField** (klase honek baditu metodo batzuk fitxategiak kudeatzeko)
 - **ImageField** (FileField klasetik heredatzen ditu atributuak eta metodoak, baina kargatutako fitxategia irudia dela ere balidatzen du)
 - **IntegerField** (zenbaki osoa)
 - **TextField** (testu handia. Formularioko kontrola: textarea)
 - **TimeField** (denbora, *datetime.time* klaseko instantzia)
 - **URLField** (CharField URL balidazioarekin)
- <http://teknikariak.informatika.ehu.es/Django/ref/models/fields.html>



Eremuen motak (III)

- Eremu erlazionalak: erlazioak adierazten dituzten eremuak.
 - **ForeignKey** (hainbat-bat edo *many-to-one* erlazioa)
 - **ManyToManyField** (hainbat-hainbat edo *many-to-many* erlazioa)
 - **OneToOneField** (bat-bat edo *one-to-one* erlazioa)
- Eremu hauek argumentu bat eskatzen dute: zein klaserekin dauden erlazionatuta.
- *Many-to-many* erlazioetan **ManyToManyField** eremua eredu bakarrean jartzen da, ez du axola zeinetan.
 - Formulario bidez aldatuko den ereduan jartzen da.



Eremuen motak (IV)

Erlazioen adibideak

- *Hainbat-bat*

```
class Musikari(models.Model):
    izena = models.CharField(max_length=50)
    abizena = models.CharField(max_length=50)
    instrumentua = models.CharField(max_length=100)

class Album(models.Model):
    musikari = models.ForeignKey(Musikari)
    izenburua = models.CharField(max_length=100)
    argitaratze_data = models.DateField()
    izar_kopurua = models.IntegerField()
```

- *Hainbat-hainbat*

```
class Osagai(models.Model):
    # ...

class Pizza(models.Model):
    # ...
    osagaiak = models.ManyToManyField(Osagai)
```



Bestelakoak

- Lehen mailako gako automatikoa

- Django ere du bakoitzari eremu bat gehitzen dio, automatikoki inkrementatzen den lehen mailako gakoak:

```
id = models.AutoField(primary_key=True)
```

- Meta-aukerak

```
class Musikari(models.Model):  
    izena = models.CharField(max_length=50)  
    abizena = models.CharField(max_length=50)  
    instrumentua = models.CharField(max_length=100)  
  
    class Meta:  
        ordering = ["izena"]  
        verbose_name_plural = "musikariak"
```



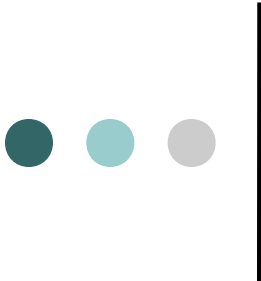
Ereduen metodoak

- **Eredu-metodoek** (*model method*), erregistro edo instantzia mailako funtzionalitatea ematen dute.
 - Horrela lortzen da negozioaren logika (ereduaren manipulazioa) ereduan bertan, gune bakarrean, ondo antolatuta edukitzea.
 - Programatzailearen esku geratzen da datu-eredu horiek manipulatzeko metodoak sortzea eta garatzea.
- **Manager** metodoek taula mailako funtzionalitate estandarra eskaintzen dute.
- Badira automatikoki sortzen diren eredu-metodoak.
 - Adibidez:
 - `__unicode__()`
 - Metodo honek ereduaren unicode errepresentazioa itzultzen du. Eredu-instantzia bat string gisa erakutsi behar denean, Django metodo hau erabiltzen du. Kotsola interaktiboan, adibidez, eredu-instantzia bat erakusten denean, metodo hau erabiltzen da
 - `get_absolute_url()`
 - Metodo honek objektu baten URLa kalkulatu du. Djangoaren arkitekturan garrantzitsua da zenbaitetan objektuak URLekin identifikatzea.



Kontsultak

- Behin datu-ereduak sortuta, Djangok API bat eskaintzen du objektuak sortu, bilatu, aldatu eta ezabatzeko.
- Garrantzitsua da honakoa gogoan izatea:
 - Djangoren eredu-geruzan
 - eredu-klase bat = datu-baseko taula bat
 - eredu-klasearen instantzia bat = datu-baseko taularen erregistro partikular bat



Kontsultak

Objektuak sortu

- Objektuak sortzeko
 - Inportatu ereduaren klasea
 - Instantziatu klasea dagozkion argumentuekin
 - Salbatu sortu den instantzia
- Suposatuz *models.py* fitxategia *nireproiektua/aplikazioa/models.py* direktorioan dagoela:

```
>>> from aplikazioa.models import Musikari
>>> b = Musikari(izena='Luper', abizena='Ordorika', instrumentua='gitarra')
>>> b.save()
```

- Aldaketak:

```
>>> b.izena = 'Ruper'
>>> b.save()
```



Kontsultak

Objektuak bilatu eta berreskuratu

- *QuerySet*: datu-baseko objektuen multzoa.
- Beraz, objektuak berreskuratzen direnean, *QuerySet* bat osatzen da.
 - *QuerySet*-ak SQLko SELECT kontsulten emaitzatzat har daitezke.
- *QuerySet*-ei buruzkoak osatzeko, ikus:
 - <http://teknikariak.informatika.ehu.es/Django/ref/models/queriesets.html>
- *QuerySet*-ak *Manager* baten bidez berreskuratzen dira. Besterik ezeko manager-a **objects** da.
 - *Manager*-ak klase-metodoak dira.
- Klase bateko **objektu guztiak** berreskuratzeke: `all()` metodoa

```
>>> musikari_guztiak = Musikari.objects.all()
```



Kontsultak

Objektuak bilatu eta berreskuratu (II)

- Taula bateko objektu guztiak ez, baizik eta **objektu espezifiko batzuk** berreskuratu nahi direnean, bi metodo daude:
 - `filter(**kwargs)`
 - Murritzapenak (*lookup parameter*) betetzen dituzten erregistroak bueltatzen dira.
 - `exclude(**kwargs)`
 - Murritzapenak (*lookup parameter*) betetzen **ez** dituzten erregistroak bueltatzen dira.
 - *Lookup* parametroek formatu hau hartzen dute:
 - `eremua__lookuptype=balioa`
 - *Lookup* parametro guztien zerrenda, hemen:

<http://teknikariak.informatika.ehu.es/Django/ref/models/querysets.html#field-lookups>

```
>>> q = Musikari.objects.filter(izena__startswith="R")
```

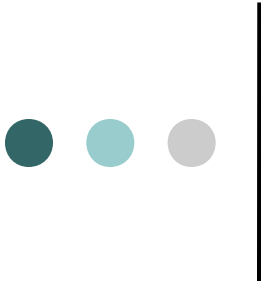



Kontsultak

Objektuak bilatu eta berreskuratu (III)

- Objektu multzo bat ez, baizik eta **objektu bakarra** berreskuratu nahi denean. `get()` erabili ohi da. Honako abibidean, *pk* parametroak *primary key* esan nahi du:

```
>>> erregistro_bakarra = Musikari.objects.get(pk=1)
```



Kontsultak

Objektuak ezabatu

- Datu-ereduko objektuak (instantziak) ezabatzeko `delete()` metodoa erabiltzen da.
- Adibideak

```
>>> erregistro_bakarra.delete()
```

```
>>> Musikari.objects.filter(izena__startswith="R").delete()
```

```
>>> b = Musikari.objects.get(pk=1)
# Musikari instantzia bat
# eta hura erreferentziazten duten album objektu guztiak ezabatzen ditu.
>>> b.delete()
```



Kontsultak

Erlazionatutako objektuen atzipena

- Eredu batean erlazioa definitzen denean (*ForeignKey*, *OneToOneField*, edo *ManyToManyField*), eredu horren instantziek API bat dute erlazionatutako objektuak atzitzeko.
- hainbat-bat erlazioa
 - Adibidez, `a` albumaren musikaria atzi daiteke, horrela: `a.musikari`
 - APIak eskaintzen du aukera erlazioaren beste aldetik ere objektuak atzitzeko. Adibidean, `Musikaria` eredutik `Album` eredura.
 - Adibidez, `m` musikaritik bere `Album` guztiak atzi daitezke `album_set` atributuaren bidez: `m.album_set.all()`
 - Adibide honetan `album_set` atributua *manager* bat da (*related manager*), `QuerySet`ak itzultzen ditu, eta lehen ikusi dugu nola erabil daitezkeen.



Kontsultak

Erlazionatutako objektuen atzipena (II)

- hainbat-hainbat erlazioa
 - Bi aldeetatik queryset-ak atzitzen dira.
 - Alde batetik `_set` atzizkia erabiltzen da, bestetik ez.
 - Adibidez, Pieza eta Osagai ereduaren arteko erlazioan oinarrituta:

```
p = Pizza.objects.get(id=3)
p.osagaiak.all() # p pizzaren osagai guztiak itzultzen ditu.
p.osagaiak.count()
p.osagaiak.filter(...)

o = Osagai.objects.get(id=5)
o.pizza_set.all() # o osagaia dagoen pizza guztiak itzultzen ditu.
```



Eredui buruzko ezagutza sakontzeko

- ◉ <http://teknikariak.informatika.ehu.es/Django/topics/db/models.html>