Dojo.objects.create(Name="CodingDojo Silicon Valley", City="Mountain View", State="CA")

Dojo.objects.create(Name="CodingDojo Seatlle", City="Seatlle", State="WA")

Dojo.objects.create(Name="CodingDojo New York", City="New York", State="NY")

Dojo.objects.create(Name="CodingDojo Texas", City="Austin", State="TX")

Dojo.objects.create(Name="CodingDojo Washington DC", City="Mclean", State="VA")

Dojo.objects.create(Name="CodingDojo Bismarck", City="North Dakota", State="UT")

*## loc1 = Dojo.objects.get(id=1) this adds it to the first location so on and so forth*

*# ninja = Ninja.objects.create(fname="Johnny", lname="Schmuck", dojo=loc1) this adds it to the first location*

Ninja = Dojo.objects.create(fname="Johnny", lname="Schmuck", Dojo= )

Ninja.objects.create(fname="Chad", lname="Quinn")

Ninja.objects.create(fname="Lindsay", lname="Martin")

Ninja.objects.create(fname="Lauren", lname="Fox")

Ninja.objects.create(fname="Heather", lname="Rodriguez")

Ninja.objects.create(fname="Edward", lname="Madera")

Ninja.objects.create(fname="Erik", lname="Rodriguez")

Ninja.objects.create(fname="Laura", lname="Smith")

Ninja.objects.create(fname="Allison", lname="White")

loc = Dojo.objects.get(id=1)

loc = Dojo.objects.get(id=2)

loc = Dojo.objects.get(id=3)

loc = Dojo.objects.get(id=4)

loc = Dojo.objects.get(id=5)

loc = Dojo.objects.get(id=6)

Ninja = Dojo.objects.create(fname="Johnny", lname="Schmuck", Dojo=[id=1])

Blog.objects.create(name="Star Wars Blog", desc="Everything about Star Wars")

1. Creating a new record
   1. *Blog.objects.create({{field1}}="{{value}}", {{field2}}="{{value}}", etc)* # creates a new record in the Blog table
      * Blog.objects.create(name="Star Wars Blog", desc="Everything about Star Wars") # creates a new blog record
      * Blog.objects.create(name="CodingDojo Blog") # creates a new blog record with the empty desc field
   2. Alternative way of creating a record
      * b = Blog(name="Disney Blog", desc="Disney stuff")
      * b.name = "Disney Blog!"
      * b.desc = "Disney stuff!!!"
      * b.save()
2. Basic Retrieval
   1. Blog.objects.first() - retrieves the first record in the Blog table
   2. Blog.objects.last() - retrieves the last record in the Blog table
   3. Blog.objects.all() - retrieves all records in the Blog table
   4. Blog.objects.count() - shows how many records are in the Blog table
3. Updating the record - once you obtain an object that has the record you want to modify, use save() to update the record.  For example
   1. b = Blog.objects.first() # gets the first record in the blogs table
   2. b.name = "CodingDojo Blog"  # set name to be "CodingDojo Blog"
   3. b.save() # updates the blog record
4. Deleting the record - use delete().  For example
   1. b = Blog.objects.get(id=1)
   2. b.delete() # deletes that particular record
5. Other methods to retrieve records
   1. Blog.objects.get(id=1) - retrieves where id is 1; get() retrieves one and only one record. It will return an error if it finds fewer than or more than one match.
   2. Blog.objects.filter(name="mike") - retrieves records where name is "mike"; returns multiple records
   3. Blog.objects.exclude(name="mike") - opposite of filter; returns multiple records
   4. Blog.objects.order\_by("created\_at") - orders by created\_date field
   5. Blog.objects.order\_by("-created\_at") - reverses the order
   6. Blog.objects.raw("SELECT \* FROM {{app\_name}}\_{{class/table name}}") - performs a raw SQL query
   7. Blog.objects.first().comments.all() - grabs all comments from the first Blog
   8. Blog.objects.get(id=15).comments.first() - grabs the first comment from Blog id = 15
   9. Comment.objects.get(id=15).blog.name - returns the name of the blog where Comment id = 15 belongs to
6. Setting Relationship
   1. Comment.objects.create(blog=Blog.objects.get(id=1), comment="test") - create a new comment where the comment's blog points to Blog.objects.get(id=1).

**Conditions**

You can do a more complicated search than just if a given field is equal to a given value. Instead of just passing in the field name as a keyword argument to .get, .filter, or .exclude, you'd pass the field name\_\_lookup type (that's a double underscore, also known as a "dunder" for people on-the-go).

For example

* Admin.objects.filter(first\_name\_\_startswith="S") - filters objects with first\_name that starts with "S"
* Admin.objects.exclude(first\_name\_\_contains="E") - excludes objects where first\_name contains "E"
* Admin.objects.filter(age\_\_gt=80)  - filters objects with age greater than 80

**Combining queries**

Queries can be chained together:

admin = Admin.objects.filter(last\_name\_\_contains="o").exclude(first\_name\_\_contains="o")

admin = Admin.objects.filter(age\_\_lt=70).filter(first\_name\_\_startswith="S")

If it's the same type of query, instead of being chained you can add multiple arguments to the function:

admin = Admin.objects.filter(age\_\_lt=70, first\_name\_\_startswith="S")

These are cases where the conditions are joined with AND, as in, all users younger than 70 AND whose first name starts with "S". If you want OR, as in users who are younger than 70 OR whose first\_name starts with "S", you can use | (the set union operator):

admin = Admin.objects.filter(age\_\_lt=70)|Admin.objects.filter(first\_name\_\_startswith="S")

**References**

1. <https://tutorial.djangogirls.org/en/django_orm/>

**Helpful tip**

In regard to the default

Privacy Policy

To report a mistake, highlight the selection you believe is in error.

[PREVIOUS](http://learn.codingdojo.com/m/72/5494/35477)

[NEXT](http://learn.codingdojo.com/m/72/5494/35479)

Report a mistake