ORMS

* ORM stands for object relational mapper.
* It maps my model objects into records in the database.
* It creates all the necessary queries for us so we don’t have to write them from scratch.
* It does for us the creation, manipulations and adds relations between objects.
* The advantage is that it reduces the amount of code writing.
* It reduces complexity.
* Save time.

DJANGO MANAGER AND QUERY SET

DJANGO MANAGER

* Every Django model have a manage.
* To access that manager type: ModelName.objects
* The objects attribute is a model or let’s say it’s an interface that help me interact with database
* It gives me access to bunch of methods:
  + Objects.all() return the list of records for that model
  + Objects.get() return a single recod
  + Objects.filter() to filter the result.

QUERY SET

* Most of the manager methods return a query\_set.
* The query\_set does not return the records list, but it only encapsulates the set.
* The Django will execute the query inside the query\_set when I evaluate it.
  + These scenarios are:
    - Convert it to list:
      * List(query\_set)
    - Looping over the query\_set.
    - Access specific record in list or slice some of it:
      * query\_set[0]
      * query[0:15]
* The reason Django does not return for me the records and only return an encapsulated query\_set because I might want to run complex query.
  + For example I want to query\_set.filter().order\_by().sort()
  + Because I might have millions of results, and I want to filter it.
  + I don’t have to run a query on millions of records, I want it filterd before that.

Retrieving objects:

* To get one item from database I use method .get()
  + Objects.get(id=1).
  + Instead of id I can use pk=1, the benefit of that is I don’t have to remember what is the primary key name.
  + This method return exception if item with id does not exist. ObjectDoesNotExit.
  + The next method solves it.
* Objects.filter(pk=1): it get me a set based on provided value to filter, then I add method.first() to get first item.
* .first() gives me None of item with provided filter does not exit and it will not throw exception.
* If I want to check if it exist I can use .exists() and it return a Boolean.
* If I want the total number of users I can use .all().count()

Filtering data

* In Django if I want to query something I need to pass it to filter () method.
* So if I want to get users with id greater than 5 I can’t do it like this.filter(id > 5)
  + The reason is id > 5 will produce a boolean value.
* So, in Django if I want to do filter I need to pass keyword=value to filter function filter(id=5)
* In order to execute my filter, I need field lookup:
  + Field lookup can be used by giving the keyword followed by double underscore then value.
  + example do number checking for id >, <=, <
    - greater than: filter(id\_\_gt =5)
    - greater than or equal: filter(id\_\_gte =5)
    - less than: filter(id\_\_lt =5)
    - less than or equal : filter(id\_\_lte =5)
    - between value\_1 and value\_2: filter(id\_\_range =(5, 10))
  + checking the string that contains specific keywork:
    - filter( title\_\_contains=’L’ ), this will give me all titles with L capital letter.
    - To make it give me insensitive case I add I before contains: filter( title\_\_icontains=’L’ )
  + If I want to get string starts with specific string:
    - filter( title\_\_startswith= ’character up to infinity’ )
  + If I want to get string ends with specific string:
    - filter( title\_\_endswith= ’character up to infinity’ )
  + if I want to work with date:
    - get items in specific year: filter( last\_update\_\_year = 2021 )
    - get items in specific month: filter( last\_update\_\_month = val )
    - get items in specific day: filter( last\_update\_\_day = val)
  + get item with field contains null value
    - filter( title\_\_isnull = True)
* if I want to apply complex query there are 3 ways: example of complex query( inventory <10 and price> 20 )
  + one way to add filter and pass multiple filtering options separated with comma, example:
    - filter(inventory\_\_lt=10, unit\_price\_\_gt=20)
  + second way to add option inside filter and then apply another filter.
* Third one is to use Q object by passing the query inside filter method then add logical or, and
  + To implement this first I need to import it from Django.db.models import Q
  + Then use Q (pass my option):
    - Filter(Q(inventory\_\_lt=10))
  + And operatior &:
    - Filter(Q(inventory\_\_lt=10) & Q(unit\_price\_\_gt=20) )
    - Get me items inventory less than 10 and unti price greater than 20
  + Or operator |:
    - Q(inventory\_\_lt=10) | Q(unit\_price\_\_gt=20)
  + Not operator ~: Q(inventory\_\_lt=10) & ~Q(inventory\_\_lt=10)

Limiting results:

* Sometimes I don’t want to retrieve list of records up to 1000 and send it back to user, I just want limited number e.g. I want to show for user 10 records only, or products in one page.
* In Django I can do this using slicing [ : 10].
* When the object manager retrieve a query I can add it slicing method and Django will evaluate it to list.
* Then I can send the response.
* So [ : 10 ] means from 0 index to 9 index but not 10.
* And [ 10 : 20] is from 10 to 19 but not 20

Sorting data:

* To sort result I need to use method called sort\_by(‘ sort\_field’ ).
  + Example: Product.objects.sort\_by(‘title’)
* I can combine fields for sorting: .sort\_by(‘field1’, ‘field2’)
  + Product.objects.sort\_by( ‘unit\_price’, ‘title’)
  + This example will first all retrieved items by unit price, then when 2 items or more have the same price will be sorted by title.
* Reverse sorting: To reverse soring I add minus sign before the field name: Product.objects.sort\_by(‘-title’)
  + Will give me result in reverse.
* Reverse by reverse method: Product.objects.sort\_by(‘title’).reverse()
* To get the first item reversed I have 3 ways:
* By index: Product.objects.sort\_by(‘title’)[0]
* By index last time: Product.objects.sort\_by(‘-title’).
* Product.objects.sort\_by(‘title’).first
  + Sort result then get me first item.
* Product.objects.sort\_by(‘title’).last
  + Sort array in reverse then get the first item.