the Model Manager

Interface to the database table

What is a Manager?

The manager is an interface of the model that provides access to the database table.

Each model has at least one manager.

The default manager is called objects.

Company.objects

<django.db.models.manager.Manager object at 0x7f8f09cb71c

The Methods of the Manager

A manager has diverse methods to deliver results. For example:

get(): get one Company object
all(): get all Company objects
order_by(): get all Company objects sorted
values(): get all objects as dictionaries

Some manager methods return a queryset, some don't. You can find them here:

https://docs.djangoproject.com/en/4.2/ref/models/querysets/#methods-that-return-new-querysets

https://docs.djangoproject.com/en/4.2/ref/models/querysets/#methods-that-do-not-return-querysets

get()

```
o = Company.objects.get(pk=2)
type(o)
<Company: Suberduber AG>
```

get always returns exactly one object! If an object is not found, a Modelname.DoesNotExist exception is thrown.

```
try:
   company = Company.objects.get(pk=42)
except Company.DoesNotExist:
    raise Http404("Diese Firma existiert nicht")
```

all()

The all() method of the manager returns all results of a model.

```
qs = Company.objects.all()
type(qs)
<Queryset>
```

The result of the all() method is a queryset. Filters can be applied to a queryset (see uerySet). All querysets created with all() always refer to the original data set.

all() with Limit

```
qs = Company.objects.all()[0:2]
```

with the slice operator we can limit the query to entries. It is important to understand here that the result is not sliced, but the SQL query already contains the LIMIT statement (performance).

```
>> print(qs.query)
SELECT "company_company"."id", "company_company"."name",
"company_company"."description", "company_company"."number_of_employees",
"company_company"."company_type", "company_company"."sub_title" FROM
"company_company" LIMIT 2
```

Intermezzo: queryset

A QuerySet represents a collection of objects from the database. To restrict the result of a queryset, filters can be applied to the set. The return value of a filter is again a queryset.

A queryset is lazy, it is not executed on the database until it is actually evaluated.

When is a queryset evaluated?

https://docs.djangoproject.com/en/3.1/ref/models/querysets/#when-querysets-are-evaluated

Queryset: Example Filter()

```
qs = Company.objects.all()
qs = qs.filter(name__startswith="A“)
qs = qs.filter(description__icontains="AG")
```

```
qs
<QuerySet [<Company: Superduper AG>]>
```

https://docs.djangoproject.com/en/3.1/ref/models/querysets/

order_by()

order_by() creates an ordered query set and corresponds to the ORDER BY clause in SQL. If order_by() is executed directly on the manager, we get the whole dataset. If executed on a queryset, only the queryset is ordered.

```
qs = Company.objects.order_by('name')
```

Sort descending:

```
qs = Company.objects.order_by('-name')
```

Order_by on Queryset:

```
qs = Company.objects.all().filter(name__startswith='A').order_by('name', 'date')
```