Outlines

- ORM
- MODELS
- MODEL FORM
- TEMPLATE INHERITANCE
- META CLASS

**ORM**

Object-Relational Mapper is a programming technique that helps application to interact with database such as SQLite, MySQL, PostgreSQL, Oracle, MS SQL.

- create a database schema from defined classes or models.
- generate SQL from Python code for a particular database
- helps to change the database easily
- use connectors to connect databases with a web application

benefits of using the Django ORM:

**Abstraction:**

The ORM provides a layer of abstraction between your Python code and the database. This makes it easier to write and maintain your code, as you don't need to worry about the details of the database schema.

**Portability:**

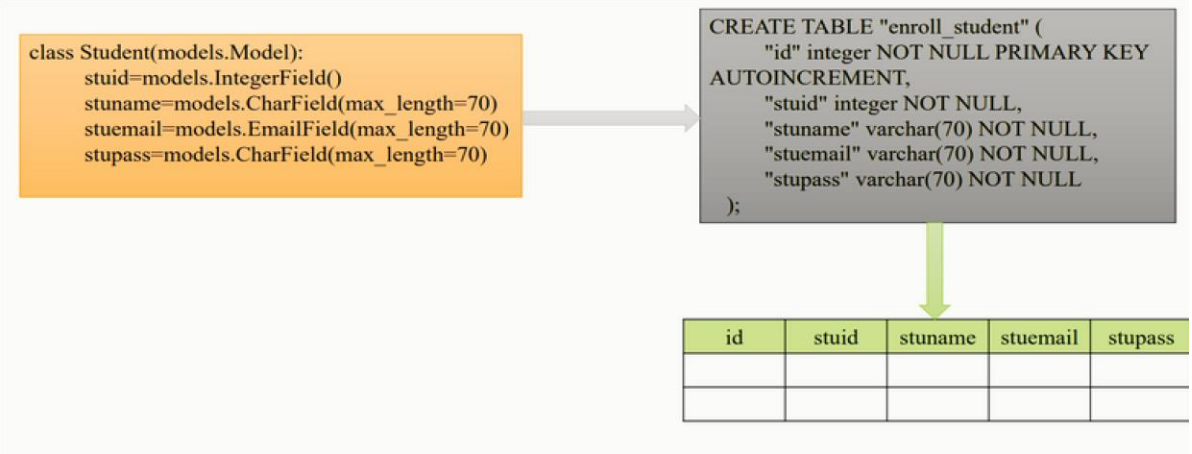
The ORM supports a variety of database backends, including MS SQL, Oracle, PostgreSQL, MySQL, and SQLite. This makes it easy to port your Django application to a different database backend.

**Performance:**

The ORM is optimized for performance. It uses a number of techniques to reduce the number of database queries that your application needs to make.

**Security:**

The ORM helps to protect your application from SQL injection attacks. It does this by escaping all user input before it is sent to the database.




**QuerySet**

A QuerySet can be defined as a list containing all those objects, we have created using the Django model.

QuerySets helps us

- read the data from the database
- filter it
- order it etc.

```python
def edit(request,id):
    book=BookStoreModel.objects.get(pk=id)
    form=BookStoreForm(instance=book)
    if request.method == 'POST':
        form=BookStoreForm(request.POST,instance=book)
        if form.is_valid():
            form.save(commit=True)
            return redirect('show')
    return render(request,'store_book.html',context={'form':form})


def delete(request,id):      You, 2 weeks ago • week 01 task done
    book=BookStoreModel.objects.get(pk=id)
    book.delete()
    return redirect('show')
```

**MODEL**

A model is the single, definitive source of information about our data.

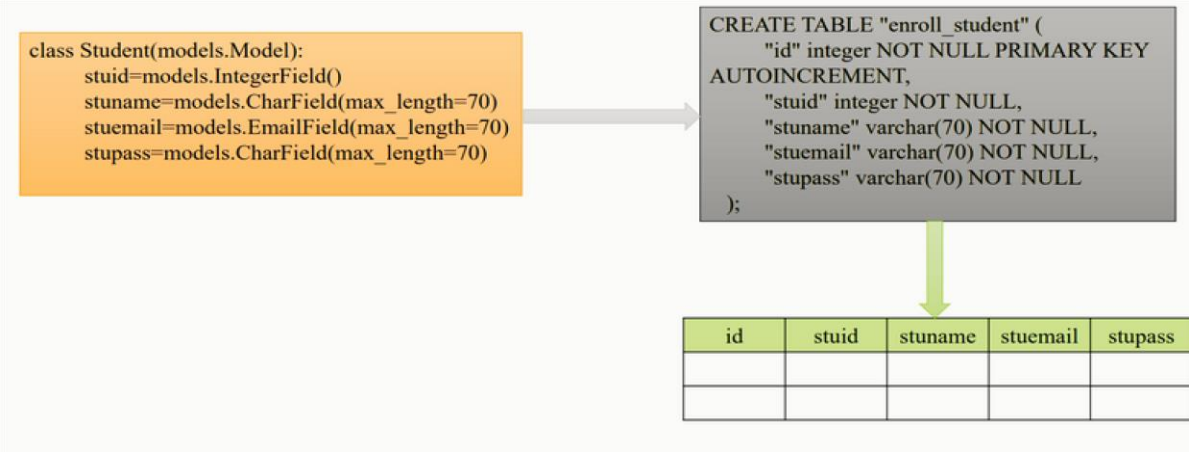
It contains the

- essential fields and behaviors of the data.
- each model maps to a single database table.

**MODEL CLASS**

Model class is a class which will represent a table in database.

- Each model is a Python class that subclasses django.db.models.Model
- Each attribute represents a database field.
- Django gives automatically-generated database-access API  Django provides sqlite database by default.
- We can use other database like MS SQL, MySQL, Oracle SQL etc.




**MIGRATIONS**

Migrations are way of propagating changes to make models (adding a field, deleting a model, etc.) into your database schema.

**makemigrations :** is used convert model class into sql statements. create a file which will contain sql statements. This file is located in Application's migrations folder.

python manage.py makemigrations

**migrate :** is used to execute sql statements generated by makemigrations.

python manage.py migrate

**showmigrations :** This lists a project's migrations.

**BUILD IN FIELDS**

**null:** contain either True or False. If True, Django will store empty values as NULL in the database. Default is False.

**blank:** contain either True or False. If True, the field is allowed to be blank.

Note: null is purely database-related, whereas blank is validation-related.

**default:** default value for the field.

**verbose_name :** A human-readable name for the field. If the verbose name isn't given, Django will automatically create it using the field's attribute name, converting underscores to spaces.

**db_column :** The name of the database column to use for this field. If this isn't given, Django will use the field's name.

**primary_key: If** True, that field will be the primary key for the model.

**unique :** If True, this field must be unique throughout the table. This is enforced at the database level and by model validation.

Some More fields :

- IntegerField
- AutoField
- FloatField
- TextField
- CharField
- BooleanField
- EmailField
- URLField

**MODEL FORM**

In Django, a ModelForm is a form that is automatically generated from an existing model.

- A ModelForm includes all the fields defined in the model, along with any additional validation and processing logic that you define in the form class.

```python
from django.db import models

class Article(models.Model):
    title = models.CharField(max_length=200)
    author = models.CharField(max_length=100)
    content = models.TextField()
```

```python
from django import forms
from .models import Article

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        fields = ['title', 'author', 'content']
```

**META CLASS**

A Meta class is a class that defines how other classes should behave. Provides additional information about the model form. Here are some common options that can be defined in the Meta class:

**model:** The model that the form is based on.

**fields:** A list of fields to include in the form. If this option is not specified, all fields in the model will be included in the form.

**exclude:** A list of fields to exclude from the form.

**widgets:** A dictionary of field names and their associated widgets.

**labels:** A dictionary of field names and their associated labels.

**help_texts:** A dictionary of field names and their associated help text.

**error_messages:** A dictionary of field names and their associated error messages.

```python
from django import forms
from .models import Article

class ArticleForm(forms.ModelForm):
    class Meta:
        model = Article
        fields = ['title', 'author', 'content']
        labels = {
            'title': 'Title of the article',
            'author': 'Author name',
            'content': 'Content of the article'
        }
        help_texts = {
            'title': 'Enter a descriptive title for the article',
            'author': 'Enter the name of the author',
            'content': 'Enter the content of the article'
        }
```

| Model Field | Form Field |
|---|---|
| AutoField | Not Represented in the Form |
| BigAutoField | Not Represented in the Form |
| BigIntegerField | IntegerField with min_value set to -9223372036854775808 and max_value set to 9223372036854775807. |
| BinaryField | CharField, if editable is set to True on the model field, otherwise not represented in the form. |
| BooleanField | BooleanField, or NullBooleanField if null=True. |
| CharField | CharField with max_length set to the model field's max_length and empty_value set to None if null=True. |
| DateField | DateField |
| DateTimeField | DateTimeField |
| DecimalField | DecimalField |
| DurationField | DurationField |

| Model Field | Form Field |
|---|---|
| EmailField | EmailField |
| FileField | FileField |
| FilePathField | FilePathField |
| FloatField | FloatField |
| ForeignKey | ModelChoiceField |
| ImageField | ImageField |
| IntegerField | IntegerField |
| IPAddressField | IPAddressField |
| GenericIPAddressField | GenericIPAddressField |
| ManyToManyField | ModelMultipleChoiceField |
| NullBooleanField | NullBooleanField |
| PositiveIntegerField | IntegerField |

| Model Field | Form Field |
|---|---|
| PositiveSmallIntegerField | IntegerField |
| SlugField | SlugField |
| SmallAutoField | Not represented in the form |
| SmallIntegerField | IntegerField |
| TextField | CharField with widget=forms.Textarea |
| TimeField | TimeField |
| URLField | URLField |
| UUIDField | UUIDField |