

Table of Contents

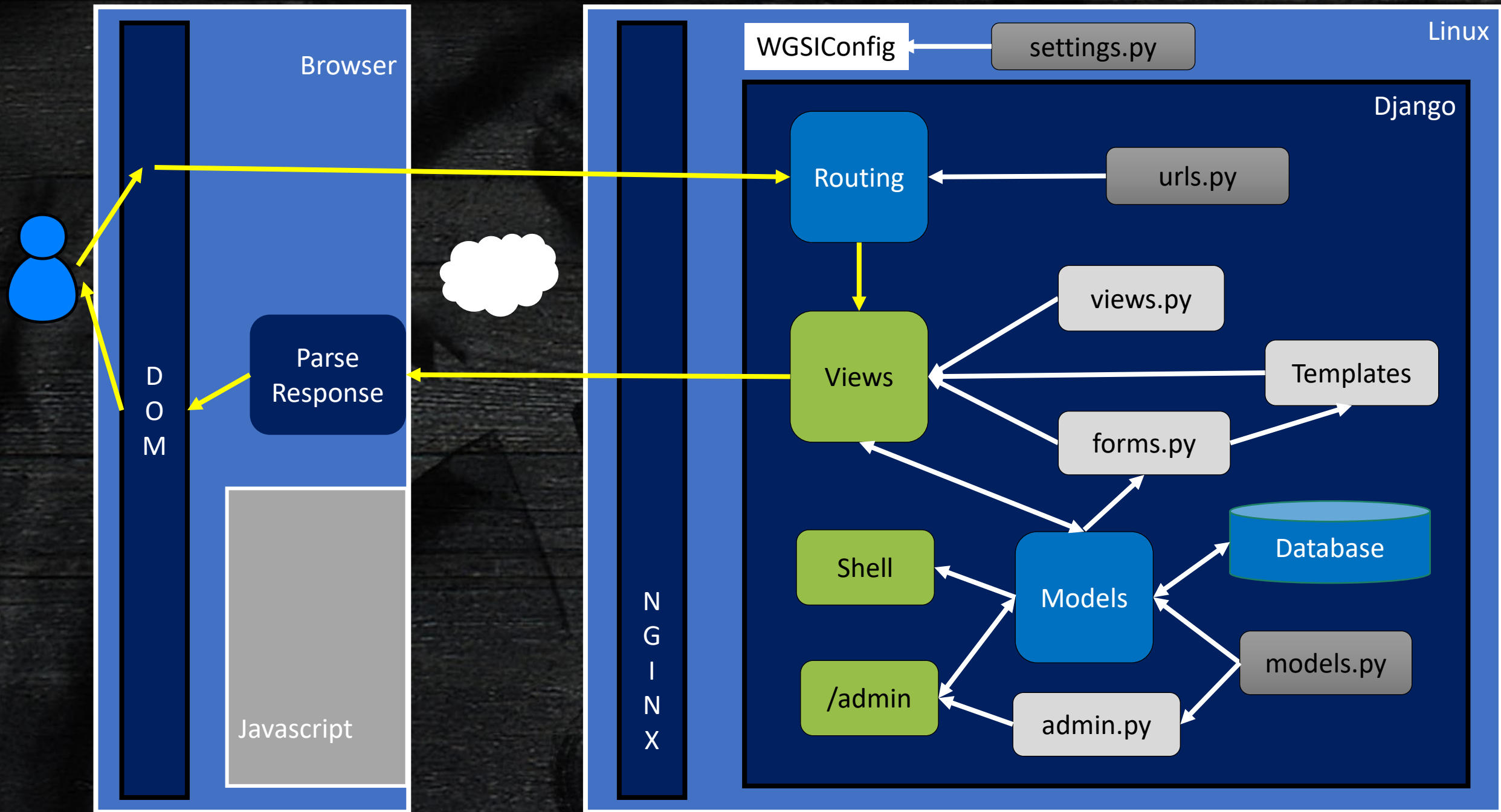
This slide deck consists of slides used in 2 lecture videos in Week 1. Below is a list of shortcut hyperlinks for you to jump into specific sections.

- (page 2) [Week 1: Django Data Models](#)
- (page 15) [Week 1: Django Migrations](#)

Charles Severance
www.dj4e.com

Simple Django Models

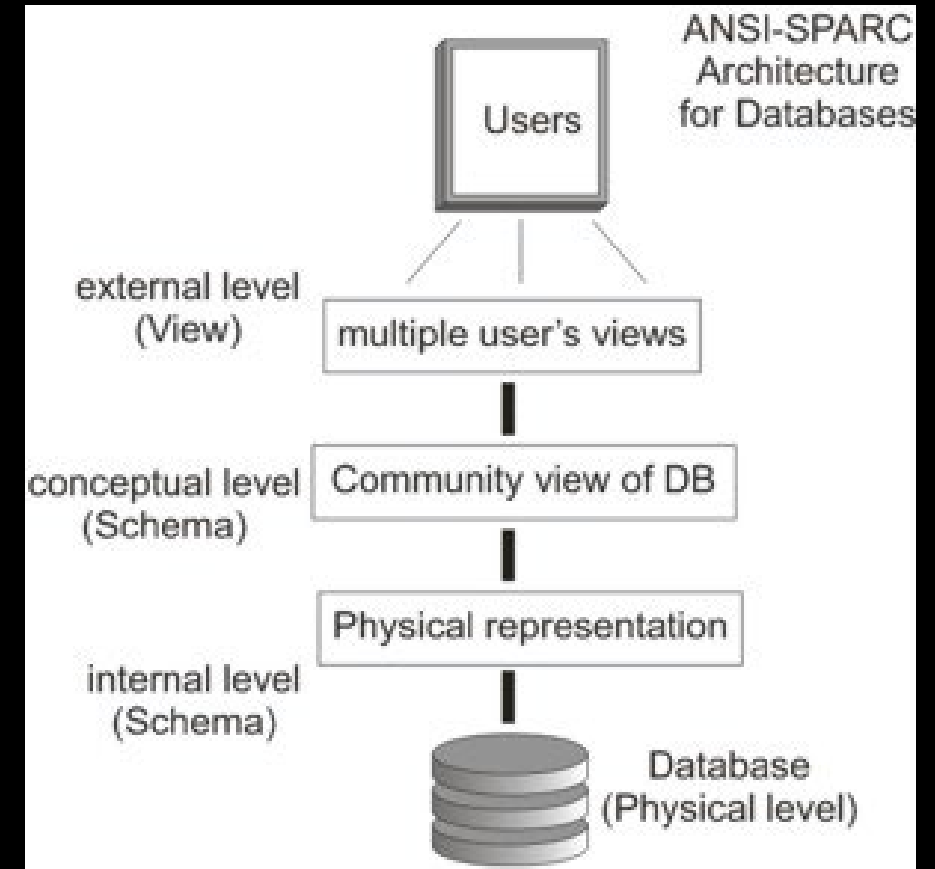




SQL

Structured Query Language is the language we use to issue commands to the database

- Create/Insert data
- Read/Select some data
- Update data
- Delete data



<http://en.wikipedia.org/wiki/SQL>

https://en.wikipedia.org/wiki/ANSI-SPARC_Architecture

Start Simple - A Single Table

```
$ sqlite3 zip.sqlite3
SQLite version 3.11.0 2016-02-15 17:29:24
Enter ".help" for usage hints.
sqlite> .tables
sqlite> CREATE TABLE Users(
...>     id INTEGER NOT NULL
...>         PRIMARY KEY AUTOINCREMENT,
...>     name VARCHAR(128),
...>     email VARCHAR(128)
...> ) ;
sqlite> .tables
Users
sqlite> .schema Users
CREATE TABLE Users(
    id INTEGER NOT NULL
        PRIMARY KEY AUTOINCREMENT,
    name VARCHAR(128),
    email VARCHAR(128)
);
sqlite>
```

```
CREATE TABLE Users(
    id integer NOT NULL
        PRIMARY KEY
        AUTOINCREMENT,
    name VARCHAR(128),
    email VARCHAR(128)
);
```

SQL Summary

```
INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')
```

```
DELETE FROM Users WHERE email='ted@umich.edu'
```

```
UPDATE Users SET name="Charles" WHERE email='csev@umich.edu'
```

```
SELECT * FROM Users
```

```
SELECT * FROM Users WHERE email='csev@umich.edu'
```

```
SELECT * FROM Users ORDER BY email
```

Object Relational Mapping (ORM)

- Allows us to map tables to objects and columns
- We use those objects to store and retrieve data from the database
- Improved portability across database dialects (SQLite, MySQL, Postgres, Oracle)



Defining a table

SQL:

```
CREATE TABLE Users(  
    name VARCHAR(128),  
    email VARCHAR(128)  
);
```

models.py:

```
from django.db import models
```

```
class User(models.Model):
```

```
    name = models.CharField(max_length=128)
```

```
    email = models.CharField(max_length=128)
```


Creating the Table from the Model

`models.py:`

```
from django.db import models
```

```
class User(models.Model):  
    name = models.CharField(max_length=128)  
    email = models.CharField(max_length=128)
```

```
$ cd ~/dj4e-samples  
$ python3 manage.py makemigrations  
Migrations for 'users':  
users/migrations/0001_initial.py  
    - Create model User  
  
$ python3 manage.py migrate  
Running migrations:  
Applying contenttypes.0001_initial... OK  
...  
Applying sessions.0001_initial... OK  
Applying users.0001_initial... OK
```

Checking...

```
$ cd ~/dj4e-samples
$ sqlite3 db.sqlite3
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
sqlite> .tables
auth_group                                django_admin_log
[ ..snip ..]
auth_user                                django_session
auth_user_groups                         users_user
auth_user_user_permissions
sqlite> .schema users_user
CREATE TABLE IF NOT EXISTS "users_user" (
    "id" integer NOT NULL PRIMARY KEY AUTOINCREMENT,
    "name" varchar(128) NOT NULL,
    "email" varchar(128) NOT NULL
);
sqlite> .quit
```

Inserting a Record

```
$ cd ~/dj4e-samples
$ python3 manage.py shell
>>> from users.models import User
>>> u = User(name='Kristen', email='kf@umich.edu')
>>> u.save()
>>> print(u.id)
1
>>> print(u.email)
kf@umich.edu
>>>
```

INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')

Checking...

```
>>> from django.db import connection
>>> print(connection.queries)
[
{'sql': 'BEGIN', 'time': '0.000'},
{'sql': 'INSERT INTO "users_user" ("name", "email")
      VALUES (\'Kristen\', \'kf@umich.edu\')',
 'time': '0.002'}
]
>>>
```

INSERT INTO Users (name, email) VALUES ('Kristin', 'kf@umich.edu')

CRUD in the ORM

```
u = User(name='Sally', email='a2@umich.edu')  
u.save()
```

```
User.objects.values()  
User.objects.filter(email='csev@umich.edu').values()
```

```
User.objects.filter(email='ted@umich.edu').delete()  
User.objects.values()
```

```
User.objects.filter(email='csev@umich.edu').update(name='Charles')  
User.objects.values()
```

```
User.objects.values().order_by('email')  
User.objects.values().order_by('-name')
```

Model Field Types

- AutoField
- BigAutoField
- BigIntegerField
- BinaryField
- BooleanField
- CharField
- DateField
- DateTimeField
- DecimalField
- DurationField
- EmailField
- FileField
- FilePathField
- FloatField
- ImageField
- IntegerField
- GenericIPAddressField
- NullBooleanField
- PositiveIntegerField
- PositiveSmallIntegerField
- SlugField
- SmallIntegerField
- TextField
- TimeField
- URLField
- **ForeignKey**
- **ManyToManyField**
- **OneToOneField**

<https://docs.djangoproject.com/en/2.1/ref/models/fields/#field-types>

Models, Migrations, and Database Tables

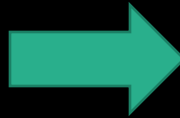
Migrations: From Model to Database

- The **makemigrations** command reads all the **models.py** files in all the applications, and creates / evolves the migration files
- Guided by the applications listed in **settings.py**
- Migrations are portable across databases
- The **migrate** command reads all the **migrations** folders in the application folders and creates / evolves the tables in the database (i.e. db.sqlite3)

makemigrations

```
dj4e-samples$ ls */models.py
```

```
autos/models.py      many/models.py
bookone/models.py    menu/models.py
crispy/models.py      myarts/models.py
favs/models.py        pics/models.py
favsql/models.py      rest/models.py
form/models.py         route/models.py
forums/models.py      session/models.py
getpost/models.py     tmp1/models.py
gview/models.py        tracks/models.py
hello/models.py        users/models.py
home/models.py         views/models.py
dj4e-samples$
```

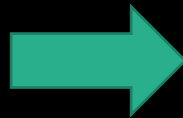


```
dj4e-samples$ ls */migrations/0*.py
```

```
autos/migrations/0001_initial.py
bookmany/migrations/0001_initial.py
bookone/migrations/0001_initial.py
favs/migrations/0001_initial.py
favsql/migrations/0001_initial.py
forums/migrations/0001_initial.py
gview/migrations/0001_initial.py
many/migrations/0001_initial.py
myarts/migrations/0001_initial.py
pics/migrations/0001_initial.py
rest/migrations/0001_initial.py
tracks/migrations/0001_initial.py
users/migrations/0001_initial.py
dj4e-samples$
```

migrate

```
dj4e-samples$ ls */migrations/0*.py
autos/migrations/0001_initial.py
bookmany/migrations/0001_initial.py
bookone/migrations/0001_initial.py
favs/migrations/0001_initial.py
favsql/migrations/0001_initial.py
forums/migrations/0001_initial.py
gview/migrations/0001_initial.py
many/migrations/0001_initial.py
myarts/migrations/0001_initial.py
pics/migrations/0001_initial.py
rest/migrations/0001_initial.py
tracks/migrations/0001_initial.py
users/migrations/0001_initial.py
dj4e-samples$
```



```
dj4e-samples$ sqlite3 db.sqlite3
SQLite version 3.24.0 2018-06-04 14:10:15
Enter ".help" for usage hints.
sqlite> .tables
auth_group          gview_car
auth_group_permissions gview_cat
auth_permission     gview_dog
auth_user           gview_horse
auth_user_groups    many_course
auth_user_user_permissions many_membership
autos_auto          many_person
autos_make          myarts_article
bookone_book        pics_pic
bookone_instance    rest_breed
bookone_lang        rest_cat
django_admin_log    social_auth_association
django_content_type social_auth_code
django_migrations   social_auth_nonce
django_session       social_auth_partial
favs_fav            social_auth_usersocialauth
favs_thing          tracks_album
favsql_fav          tracks_artist
favsql_thing        tracks_genre
forums_comment      tracks_track
forums_forum        users_user
sqlite> .quit
dj4e-samples$
```

Re-running makemigrate

```
dj4e-samples$ rm bookone/migrations/0001_initial.py
MacBook-Pro-92:dj4e-samples csev$ python3 manage.py makemigrations
Migrations for 'bookone':
  bookone/migrations/0001_initial.py
    - Create model Book
    - Create model Instance
    - Create model Lang
    - Add field lang to book
dj4e-samples$
```

Re-running migrate from scratch

```
dj4e-samples$ rm db.sqlite3
dj4e-samples$ python3 manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, autos, bookone, contenttypes, ...
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
[ ...snip ... ]
  Applying social_django.0008_partial_timestamp... OK
  Applying tracks.0001_initial... OK
  Applying users.0001_initial... OK
dj4e-samples$
```

Summary

- The Django Models feature implements an Object Relational Mapper
- Benefits
 - We can write only Python code (i.e. no explicit SQL)
 - We gain database portability
 - Migrations both create and evolve our database schema
 - A sweet administrator interface
 - Automatic form generation and validation (later)

Acknowledgements / Contributions

These slides are Copyright 2019- Charles R. Severance (www.dr-chuck.com) as part of www.dj4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here