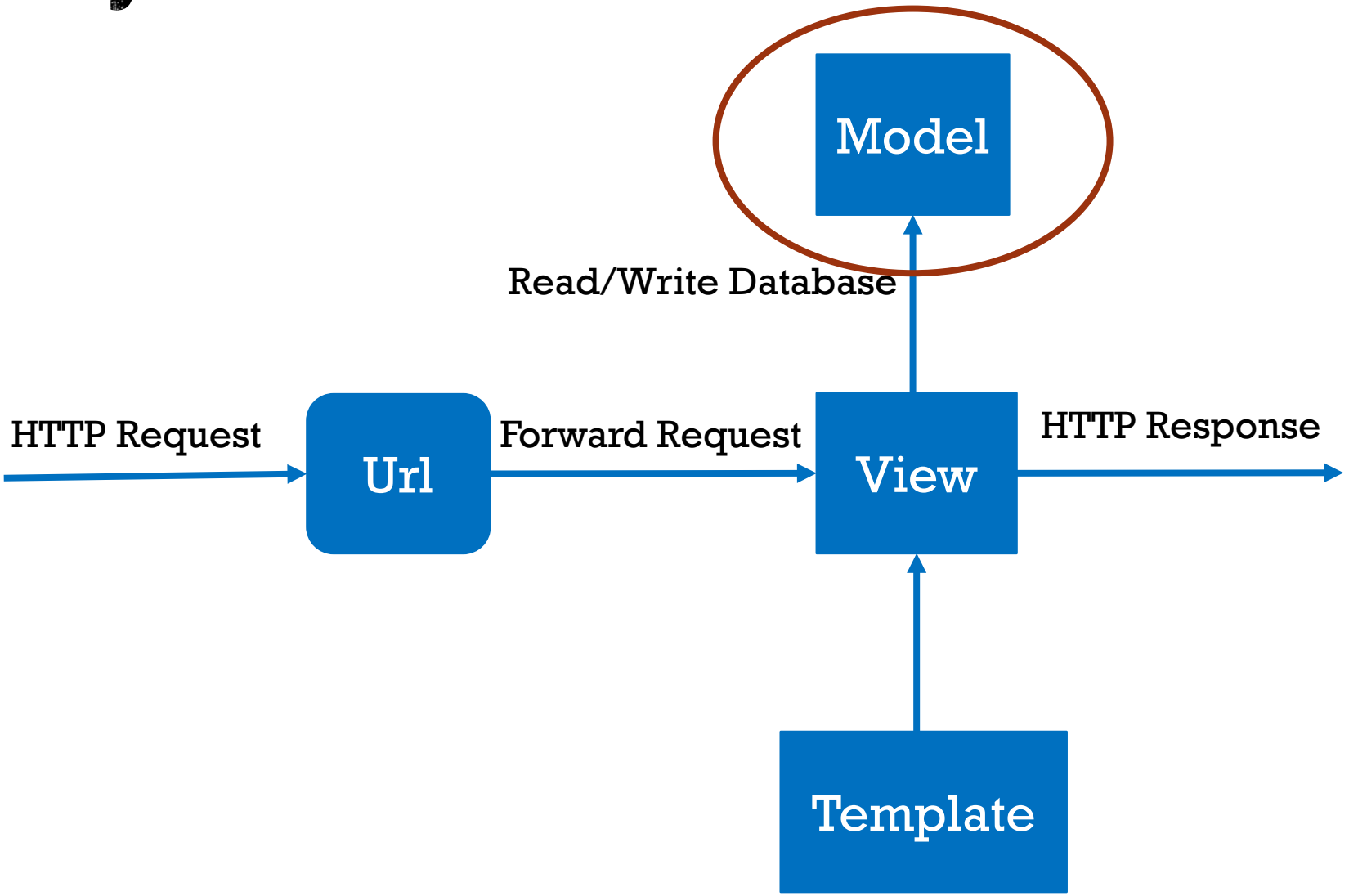
DJANGO MODEL

江琳

DATABASE SETUP

DJANGO MVT PATTERN



DATABASE SETUP

- **Open up settings.py.**
  - **By default, the configuration user SQLite. SQLite is included in Python, so you won't need to install anything else to support your database.**
  - **If you're new to databases, or you're just interested in trying Django, this is the easiest choice.**

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

WHAT IS SQLITE?

- **SQLite is a software library that provides a relational database management system. The lite in SQLite means light weight in terms of setup, database administration, and required resource.**
- **SQLite has the following noticeable features: self-contained, serverless, zero-configuration, transactional.**

DATABASE SETUP

- **When starting a real project, usually you may want to user a more scalable database like PostgreSQL and MySQL, to avoid database-switching headaches down the road.**
- **To use another database, changing some keys in the DATABASES to match your database connection settings.**

EXAMPLE -- PostgreSQL setup

- **PostgreSQL**

```
DATABASES = {
    'default': {
        'ENGINE':  'django.db.backends.postgresql_psycopg2',
        'NAME': 'lib',
        'USER': 'postgres',
        'PASSWORD': 'postgres',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

EXAMPLE -- MySQL setup

- **Mysql**

```
DATABASES = {
    'default': {
        'ENGINE':  'django.db.backends.mysql',
        'NAME': 'lib',
        'USER': 'admin',
        'PASSWORD': 'root',
        'HOST': '127.0.0.1',
        'PORT': '',
    }
}
```

ENGINE

sqlite

- **django.db.backends.sqlite3**

MySQL

- **django.db.backends.mysql**

PostgreSQL

- **django.db.backends.postgresql**

Oracle

- **django.db.backends.oracle**

NAME

- **The name of the database**
- **If using SQLite, the database will be a file on your computer. NAME should be the full absolute path, including filename.**
- **Default value:**

  os.path.join(BASE_DIR, 'db.sqlite3')

USER, PASSWORD, HOST, PORT

- **USER**
  - **The username to user when connecting to the database.**
- **PASSWORD**
  - **The password to use when connecting to the database.**
- **HOST**
  - **Which host to user when connecting to the database. An empty string means loacalhost.**
- **PORT**
  - **The port to user when connecting to the database.**

START AN APP

CREATE AN APPLICATION

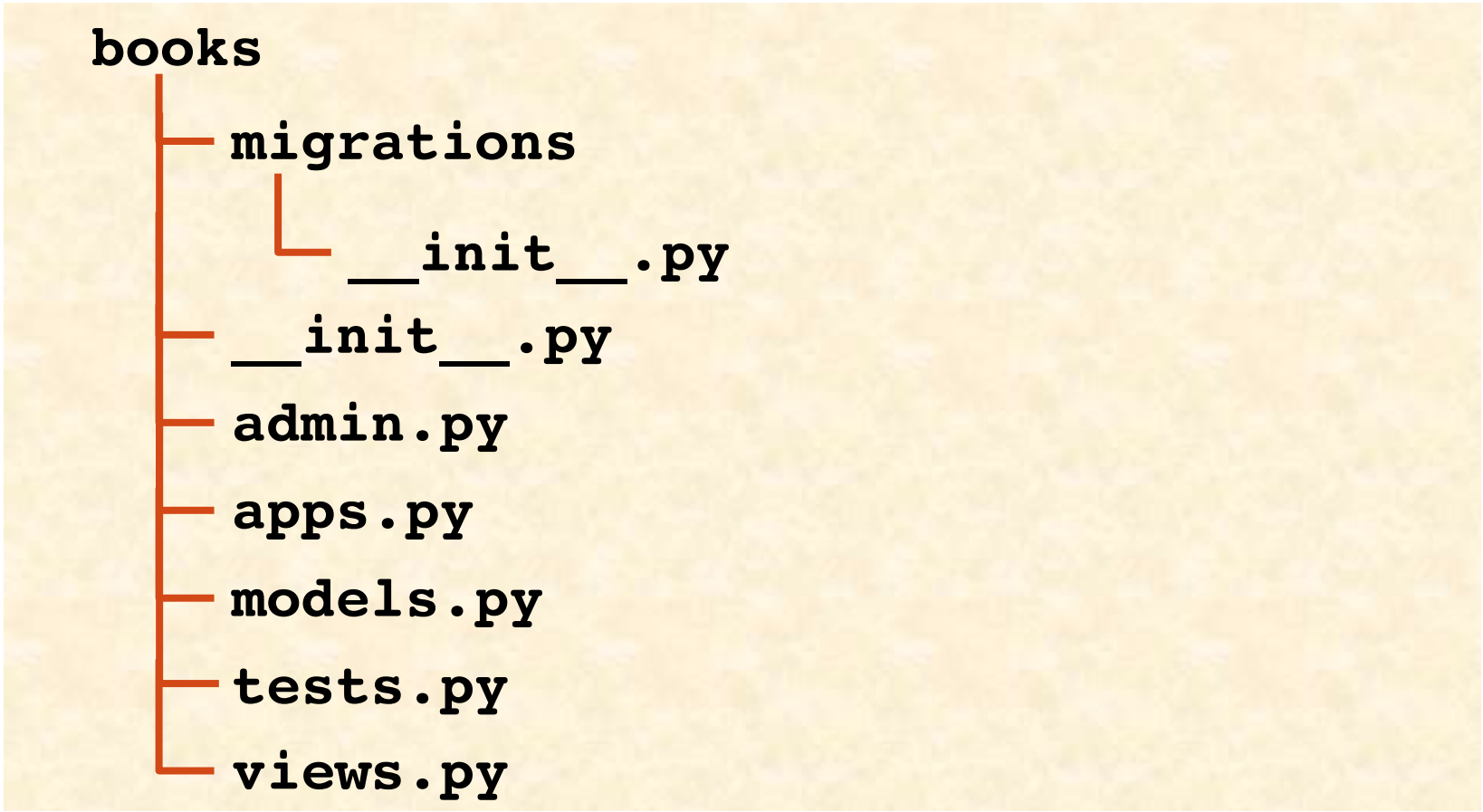
- **A project is a sum of many applications.**
- **Every application has an objective and can be reused into another project.**
- **Applications can live anywhere on your Python.**

CREATE AN APPLICATION

- **Create an app named books**

```
$ cd <project folder>
$ python manage.py startapp books
```

CREATE AN APPLICATION



REGISTERING THE POSTS APP

- **We have to register it with the project so that it will be included.**

1. **Open settings.py**
2. **Find the definition for the INSTALLED_APPS list**
3. **Add the posts app at the end of the list.**

REGISTERING THE POSTS APP

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'books',
)
```

REGISTERING THE APP

apps	explain
django.contrib.admin	Admin site
django.contrib.auth	An authentication system
django.contrib.contentty pes	A framework for content types
django.contrib.sessions	A session framework
django.contrib.messages	A messaging framework
django.contrib.staticfiles	A framework for managing static files

MODEL

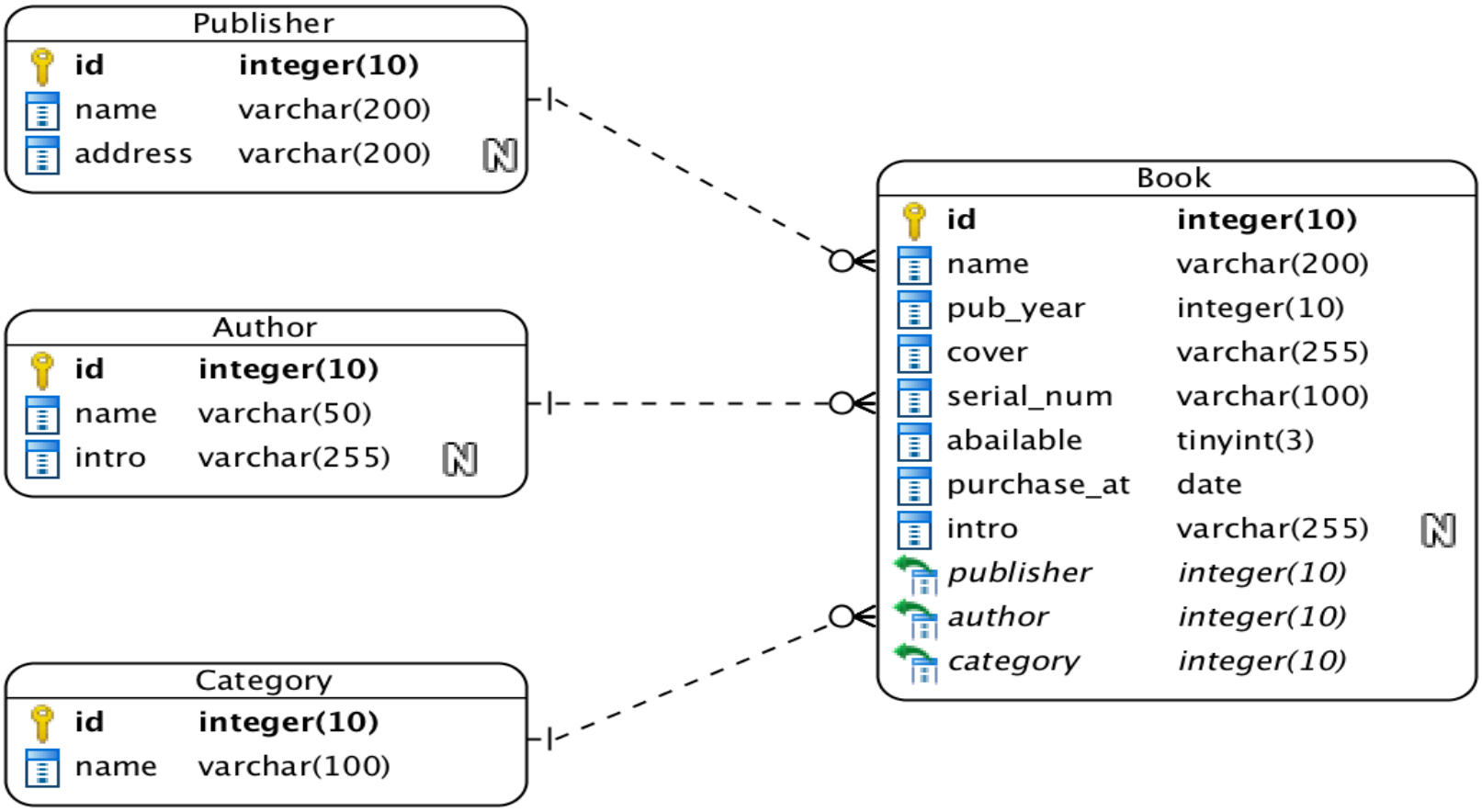
PHILOSOPHY

- **A model is the single, definitive source of truth about your data.**
- **It contains the essential fields and behaviors of the data you are storing.**

PHILOSOPHY

- **The goal is to define your data model in one place and automatically derive things from it.**
- **A model is a class that represents table or collection in our DB**

ERD


Publisher
id integer(10)
name varchar(200)
address varchar(200) N
Author
id integer(10)
name varchar(50)
intro varchar(255) N
Category
id integer(10)
name varchar(100)
Book
id integer(10)
name varchar(200)
pub_year integer(10)
cover varchar(255)
serial_num varchar(100)
abailable tinyint(3)
purchase_at date
intro varchar(255) N
publisher integer(10)
author integer(10)
category integer(10)

MODEL -- Publisher

```
from django.db import models

class Publisher(models.Model):
    name = models.CharField(max_length=200)
    address = models.CharField(max_length=200, null=True,
blank=True)
```

- **Each model is represented by a class that subclasses *django.db.models.Model***
- **Each model has a number of class variables, each of which represents a *database fields* in the model.**

Field

- **Each field is represented by an instance of a Field class:**
  - ***CharField*** **for character fields**
  - ***DataTimeField*** **for datatimes**
  - ***IntegerField*** **for integer**
  - ***TextField*** **for a large text field**
  - ***BooleanField*** **for a true or false field**
  - ***ImageField*** **for upload a image**
- **https://docs.djangoproject.com/en/1.11/ref/models/fields/**

MODEL -- Author and Category

```
class Author(models.Model):
    name = models.CharField(max_length=50)
    intro = models.TextField(null=True, blank=True)

class Category(models.Model):
    name = models.CharField(max_length=100)
```

MODEL -- Book

```
class Book(models.Model):
    name = models.CharField(max_length=200)
    pub_year = models.IntegerField(default=0)
    cover = models.ImageField(upload_to='cover/%Y/%m/%d/')
    serial_num = models.CharField(max_length=100)
    available = models.BooleanField(default=True)
    author = models.ForeignKey(Author)
    publisher = models.ForeignKey(Publisher)
    category = models.ForeignKey(Category)
    intro = models.TextField(null=True, blank=True)
    purchase_at = models.DateField(auto_now_add=True)
```

RELATIONSHIP

- One-to-One relationship
  - models.OneToOneField
- Many-to-Many relationship
  - models.ManyToManyField
- One-to-Many relationship
  - models.ForeignKey()

ACTIVATING MODELS

- **Runing makemigrations:**

```
$ python manage.py makemigrations books
```

- **Telling Django that you have make some changes to your models and that you would like the changes to be stored as a migrations**

ACTIVATING MODELS

```
$ python manage.py makemigrations books
Migrations for 'books':
books/migrations/0001_initial.py
    - Create model Author
    - Create model Book
    - Create model Category
    - Create model Publisher
    - Add field category to book
    - Add field publisher to book
```

MIGRATIONS

- **Migrations are how Django stores changes to your models – they're just files.**
- **You can read the migration file at:**

  books/migrations/0001_initial.py

SQLMIGRATE

- **The *sqlmigrate* command takes migration names and returns their SQL:**

```
$ python manage.py sqlmigrate books 0001
```

MIGRATE

- **Running migrate to create this model tables in your database:**
  - ***python manage.py migrate***
- **The migrate command takes all the migrations that haven't been applied and runs them against your database**

CHANGE YOUR MODELS

1. Run **python manage.py makemigrations** to create migrations for those change.
2. Run **python manage.py migreate** to apply those changes to the database.

Questions?