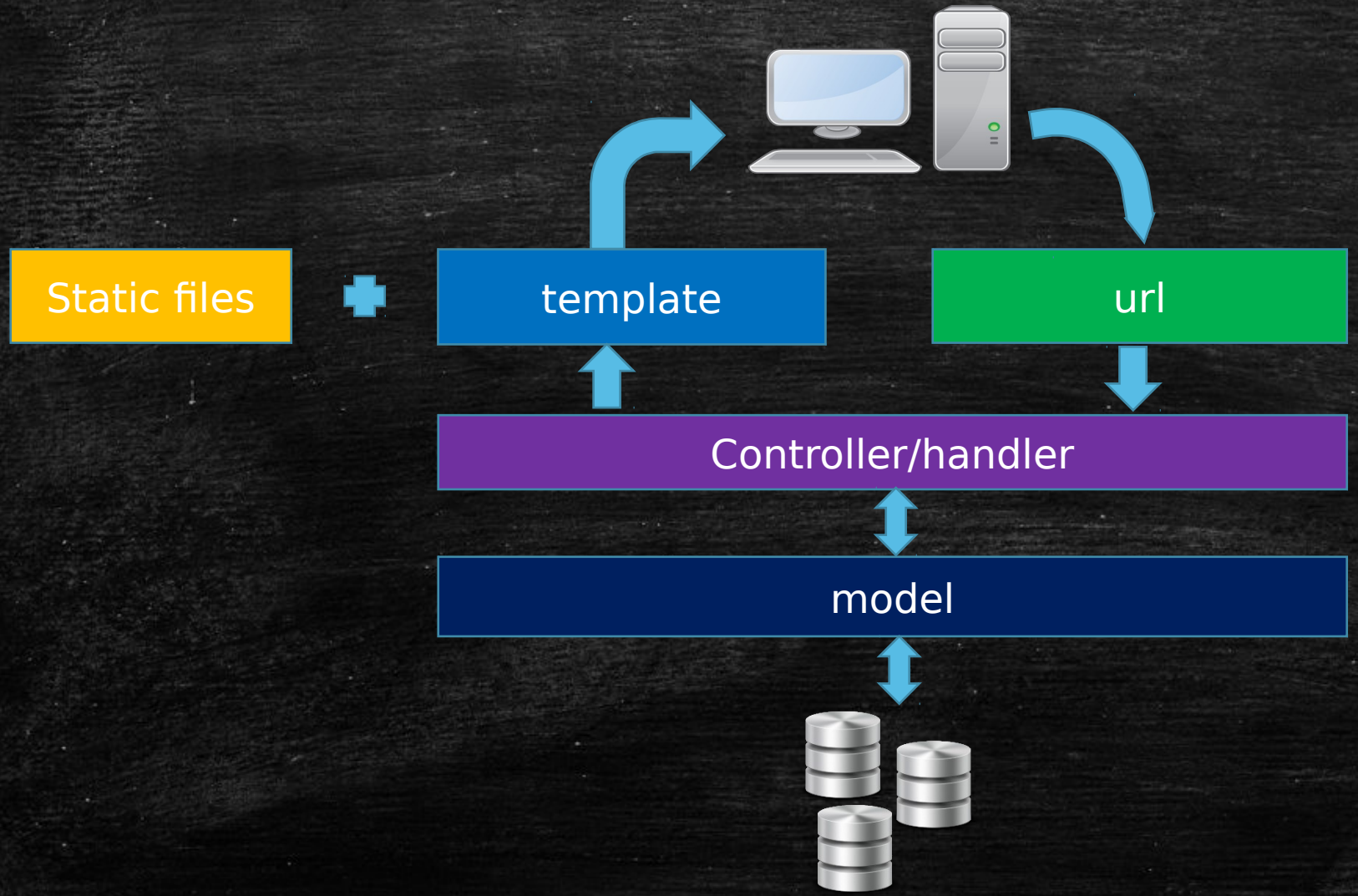
FLASK

Workshop on a micro-framework for Python

By Ankit Rana

A general web framework



Installation

- $ sudo easy_install virtualenv or Install pip if not present
- $ mkdir may-workshop
- $ cd may-workshop
- $ virtualenv zomato

  New python executable in venv/bin/python

  Installing distribute.....done.
- $. zomato/bin/activate
- $ Pip install Flask
- $ Pip install sqlalchemy

Hello World in flask

Create a file <your_project>/hello.py

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Python hello.py

Demonstration

Route

- Bind a function with a url
- app.route(url)
- Can also be used to pass variable
- Eg1:-

  ```
  @app.route('user/<username>')
  def show():
      return 'User %s' % username
  ```

- Eg2:-

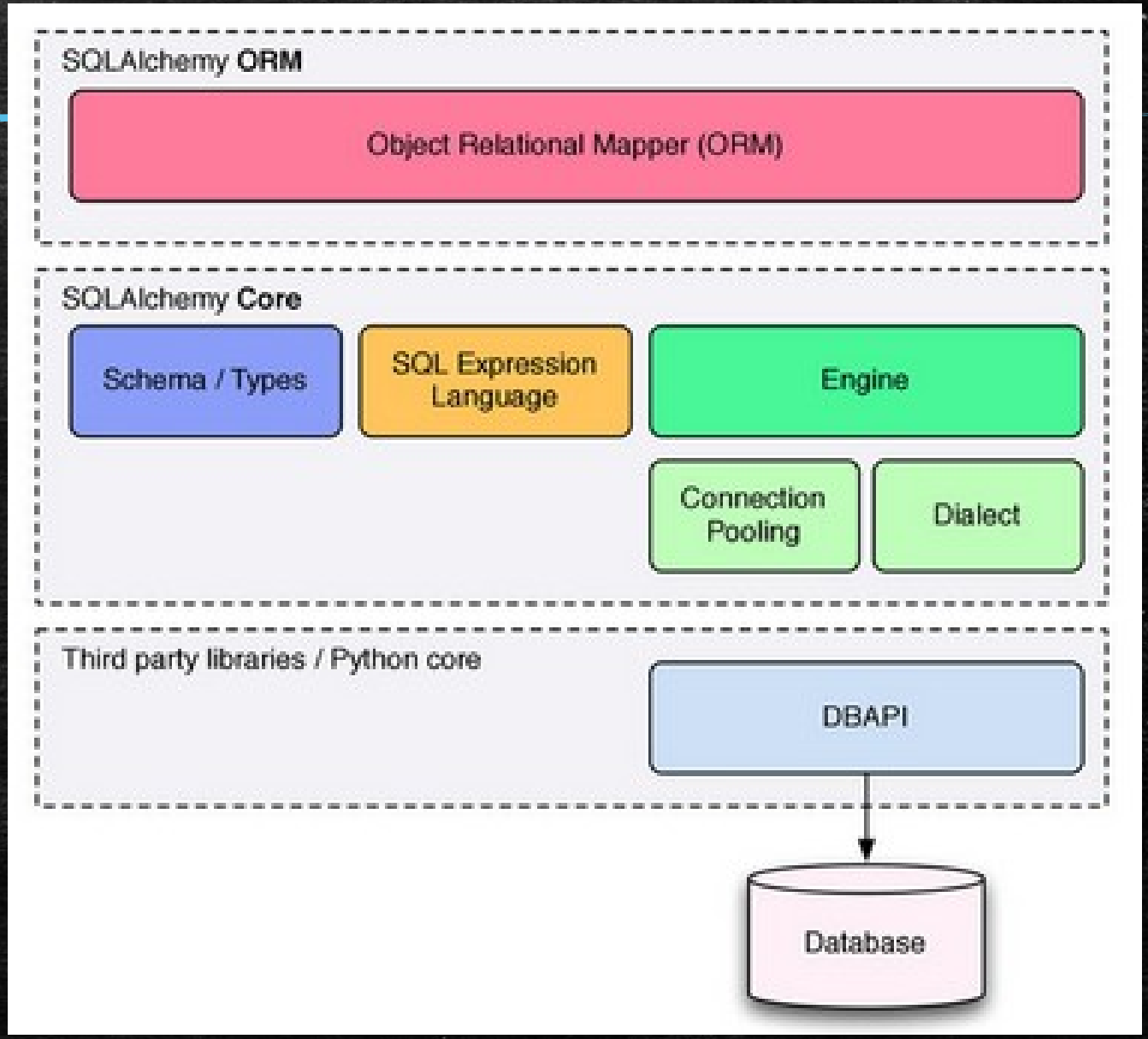
  ```
  @app.route('user/<int:user_id>')
  ```

- Eg3:-

  ```
  @app.route('number/<float:num>')
  ```

Database Connectivity

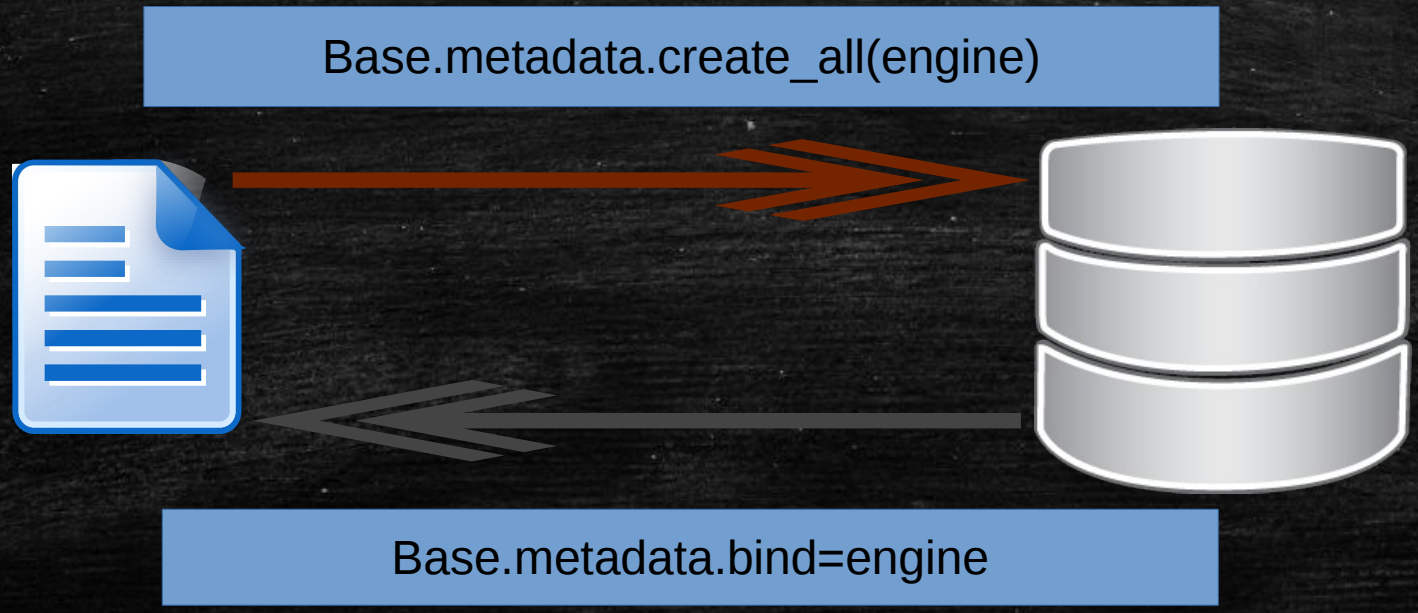
- Sqlalchemy
- ORM tool
- database abstraction



Demonstration

Declarative_base

- From sqlalchemy.ext.declarative import declarative_base
- Base=declarative_base()
- Represents that class is mapped to some table



Engine

- Starting point for any sqlalchemy application

```
create_engine('mysql+mysqlconnector://root:passwd@localhost:port/zomato')
```

Attribute and Types

- restra=relationship(tablename)
- ForeignKey('restra.id')
- nullable=False
- primary_key=True

Some Basic Types

- `Integer()` - basic integer type, generates INT
- `String()` - ASCII strings, generates `VARCHAR`
- `Unicode()` - Unicode strings - generates `VARCHAR`, `NVARCHAR` depending on database
- Boolean() - generates `BOOLEAN`, `INT`, `TINYINT`
- `DateTime()` - generates `DATETIME` or `TIMESTAMP`, returns Python `datetime()` objects
- `Float()` - floating point values
- `Numeric()` - precision numerics using Python `Decimal()`

CRUD OPERATION

- Execute database operation through session
- Session represent database connection
- Session represent cached table data

```
From sqlalchemy.orm import sessionmaker
DBSession = sessionmaker(bind=engine)
```

CRUD OPERATION

- Add data
  - t=table(attr=value,..)
  - Session.add(t)
  - Session.commit()
- Search
  - list=session.query(table).all()
  - var=session.query(table).first()
  - list=session.query(table).filter_by(query)
- Update
  - Search data
  - Data.attr=change
  - Session.add(data)
  - Session.commit()
- Delete
  - Search data
  - Session.delete(data)
  - Session.commit

Demonstration

Rendering Template

- Writing whole code in the return value of view is tedious!
- Create a folder ‘template’ next to your module and create templates in that folder.
- To associate a template with a handler ,use

```
render_template(name,var=value,..)
```

Jinja2 templates

- Conditional

```
{%  if  cond  %}
...
{%  else  %}
...
{%  endif  %}
```

- Condition

  index == 3

Jinja2 templates

- {% for item in items %}

  ....

  {% endfor %}

- To access variable/context_dict send by handler

  {{ }}

- Comments

  {# #}

Demonstration

Static Files

- All the css, js files and images are kept here
- Path anywhere can be given up with
  - {{ url_for('static',<path>) }}

Demonstration

Thank you

- You can learn more from the documentation at
- http://flask.pocoo.org/docs/0.10/