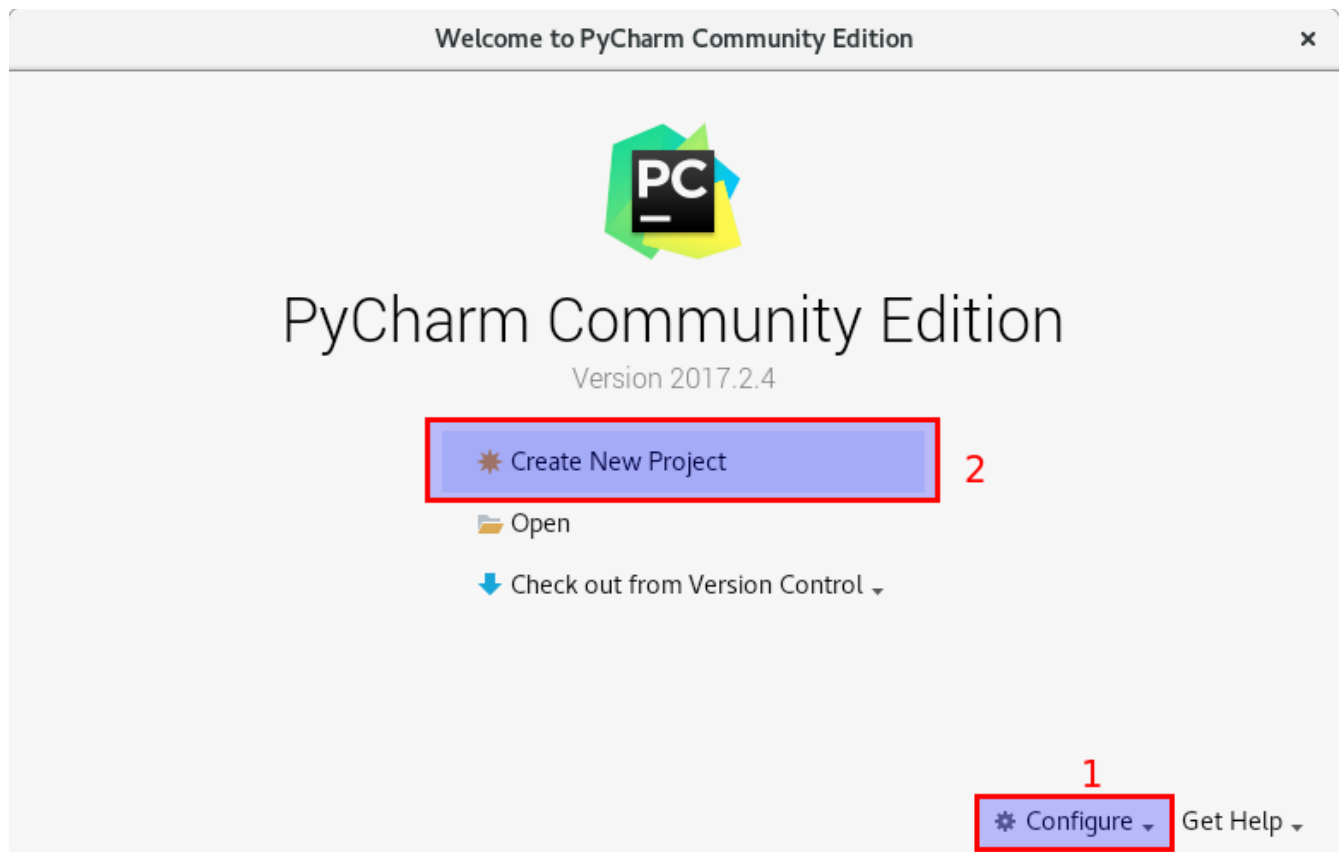


Workshop Python - Unas

Dalam modul ini akan dibahas tentang pembuatan aplikasi web berbasis Python menggunakan framework Flask, adapun IDE yang digunakan adalah PyCharm yang tersedia versi *community* dan dapat diunduh di <https://www.jetbrains.com/pycharm/download/>

1. Pengenalan PyCharm dan Flask

PyCharm merupakan salah satu IDE Python untuk memudahkan dalam mengembangkan perangkat lunak menggunakan Python.

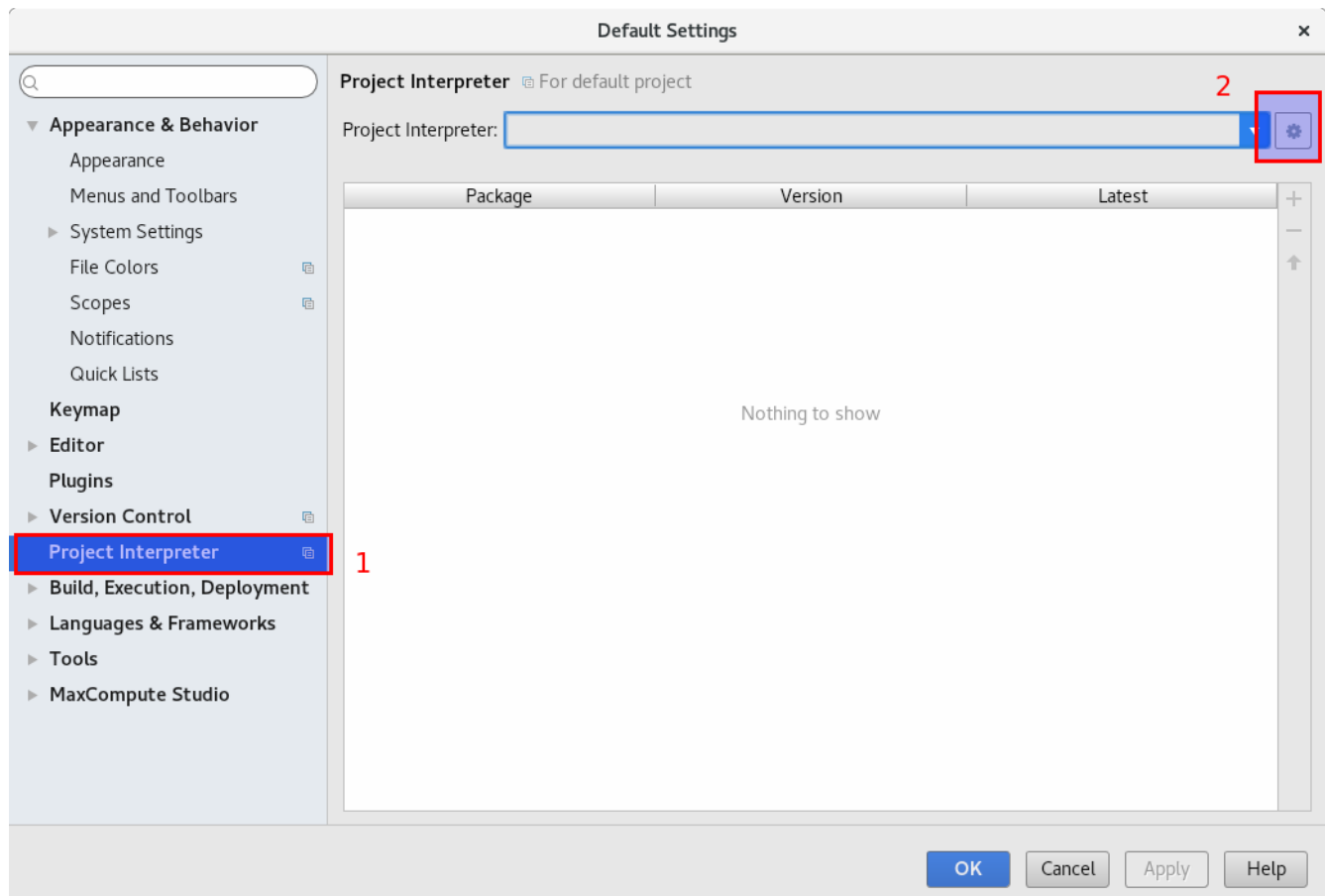


Gambar 1: Halaman Awal PyCharm

Membuat VirtualEnv

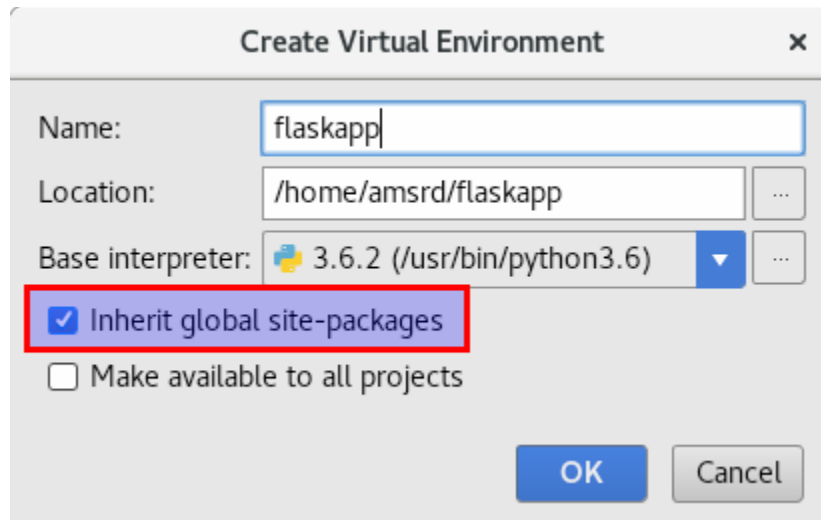
Berikutnya kita butuh memasang Flask, sebaiknya setiap project dibuatkan lingkungan khusus supaya library-library yang terpasang terdapat pada lingkungan nya masing-masing, hal ini dapat dilakukan dengan bantuan *virtualenv*. Untuk itu kita akan membuat *virtualenv* dan hal tersebut dapat dilakukan melalui PyCharm.

Untuk membuat *virtualenv* maka klik pada '*Configure*' seperti pada nomor 1 di Gambar 1, maka akan muncul jendela Setting PyCharm.



Gambar 2: Jendela Setting

1. Klik pada "Project Interpreter" seperti pada nomor 1 di Gambar 2
2. Kemudian klik ikon gear (nomor 2 di Gambar 2).
3. Kemudian pilih "Create VirtualEnv"
4. Maka akan muncul jendela "Create Virtual Environment"

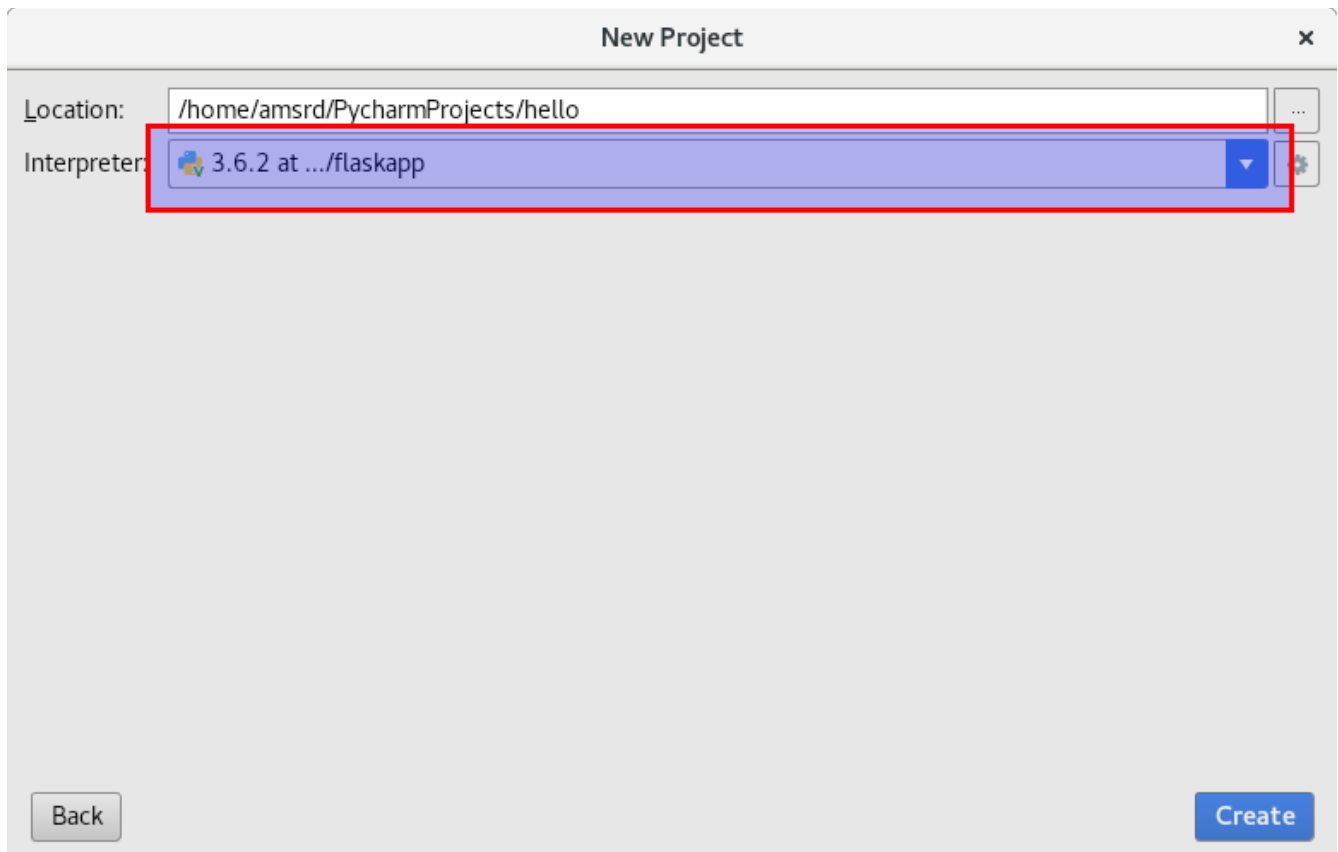


Gambar 3: Create Virtual Environment

5. Kemudian isi nama, dan lokasi
6. Aktifkan pilihan “*Inherit Global site-packages*”, hal ini maksudnya adalah library yang kita install di sistem utama Python akan dikenali di lingkungan virtual yang kita buat.
7. Kemudian klik OK. Tunggu hingga proses selesai, dan sampai di sini kita sudah selesai membuat VirtualEnv dan sudah siap untuk melanjutkan proses berikutnya.

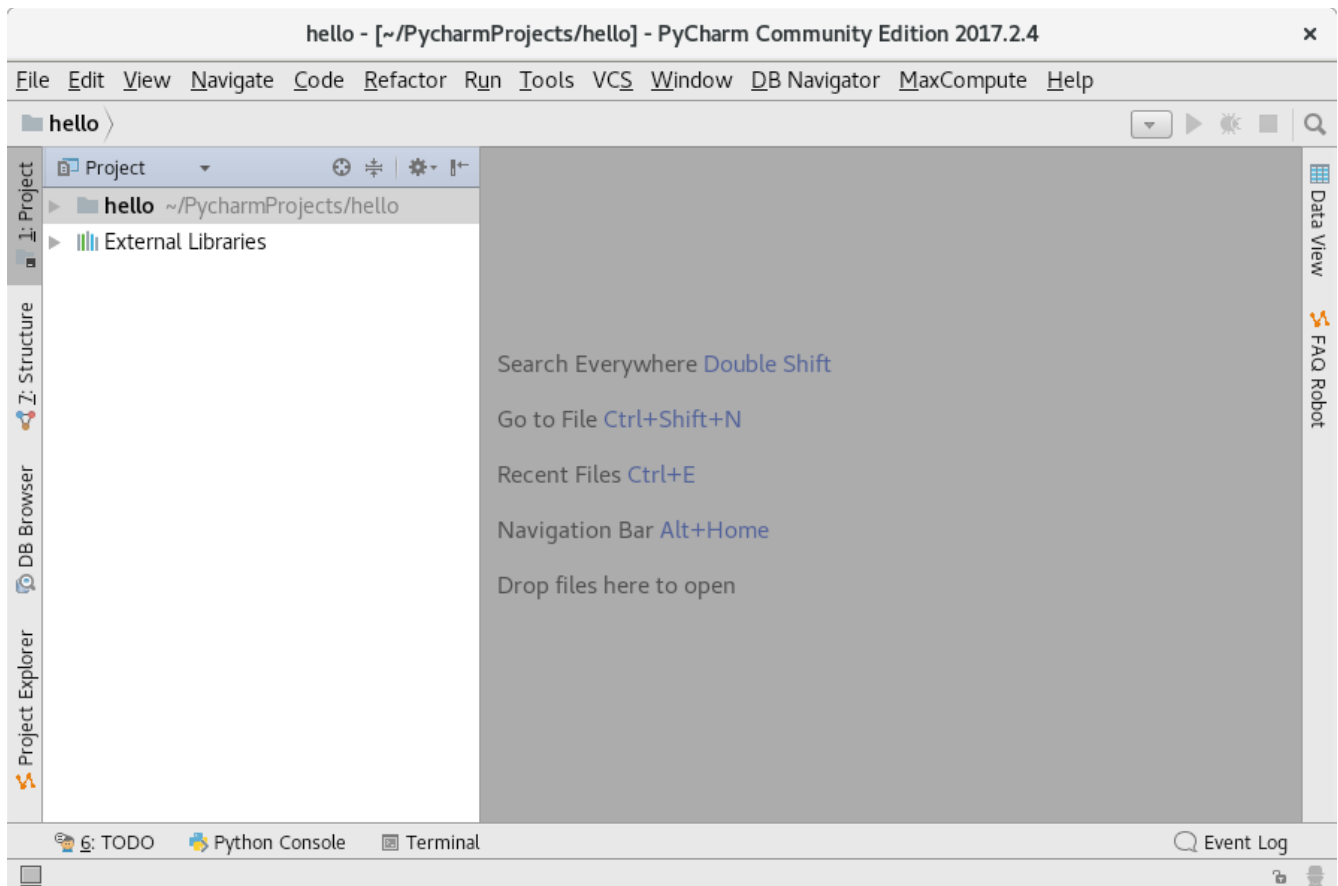
Membuat Project

Berikutnya kita akan membuat Project, untuk itu klik “*Create New Project*” (lihat pada Gambar 1, yang diberi nomor 2), maka akan muncul jendela “*New Project*”



Gambar 4: Jendela New Project

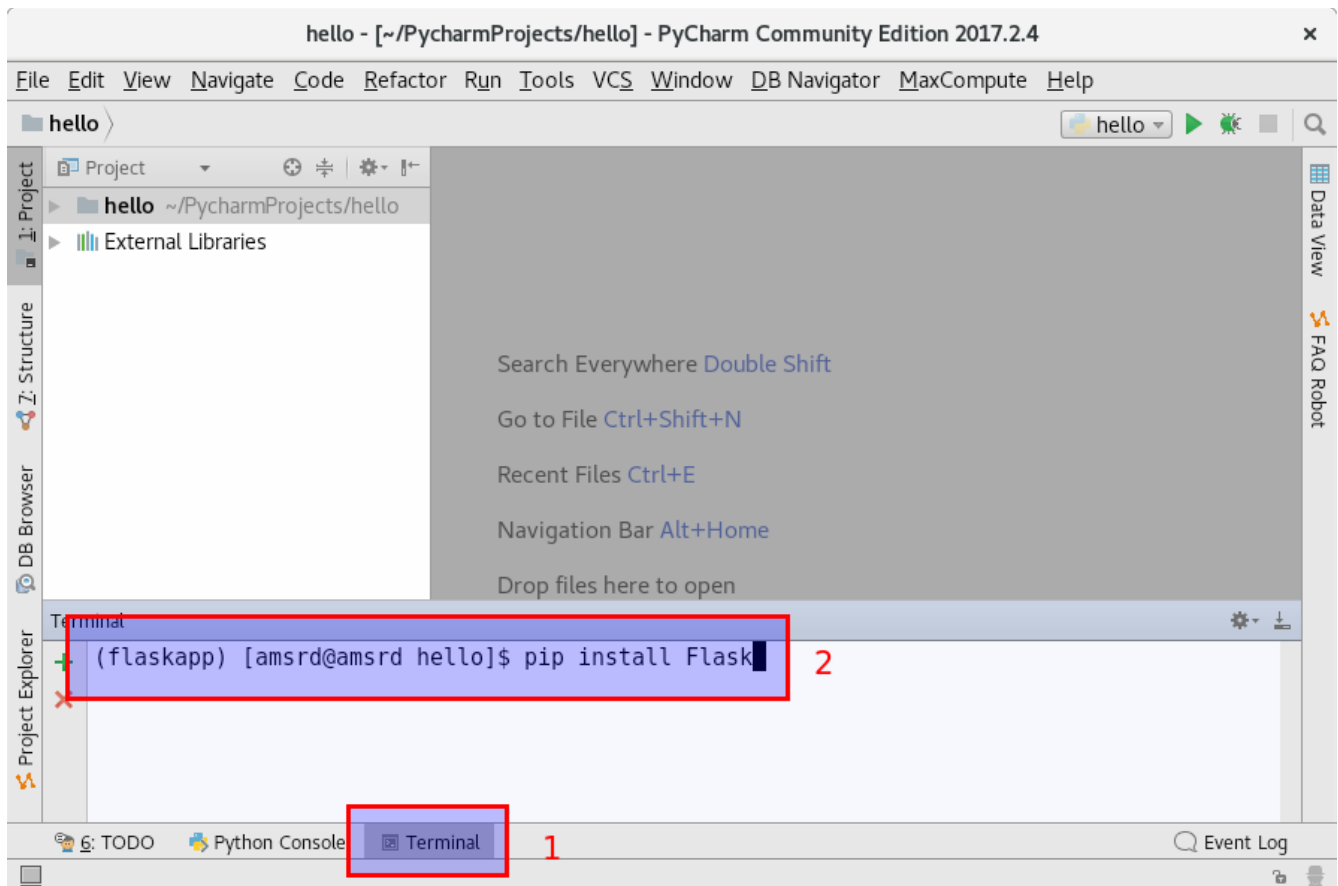
Isi pada bagian “Location” alamat folder yang akan digunakan untuk menyimpan file-file project. Kemudian pada pilihan Interpreter pilih *virtualenv* yang sebelumnya sudah kita buat, dan kemudian klik *create*. Tunggu beberapa saat sampai selesai dan kemudian akan muncul jendela utama PyCharm.



Gambar 5: Jendela Utama PyCharm

Install Flask

Selanjutnya kita akan memasang Flask ke dalam *virtualenv* yang sudah kita buat, untuk itu langkahnya adalah sebagai berikut:



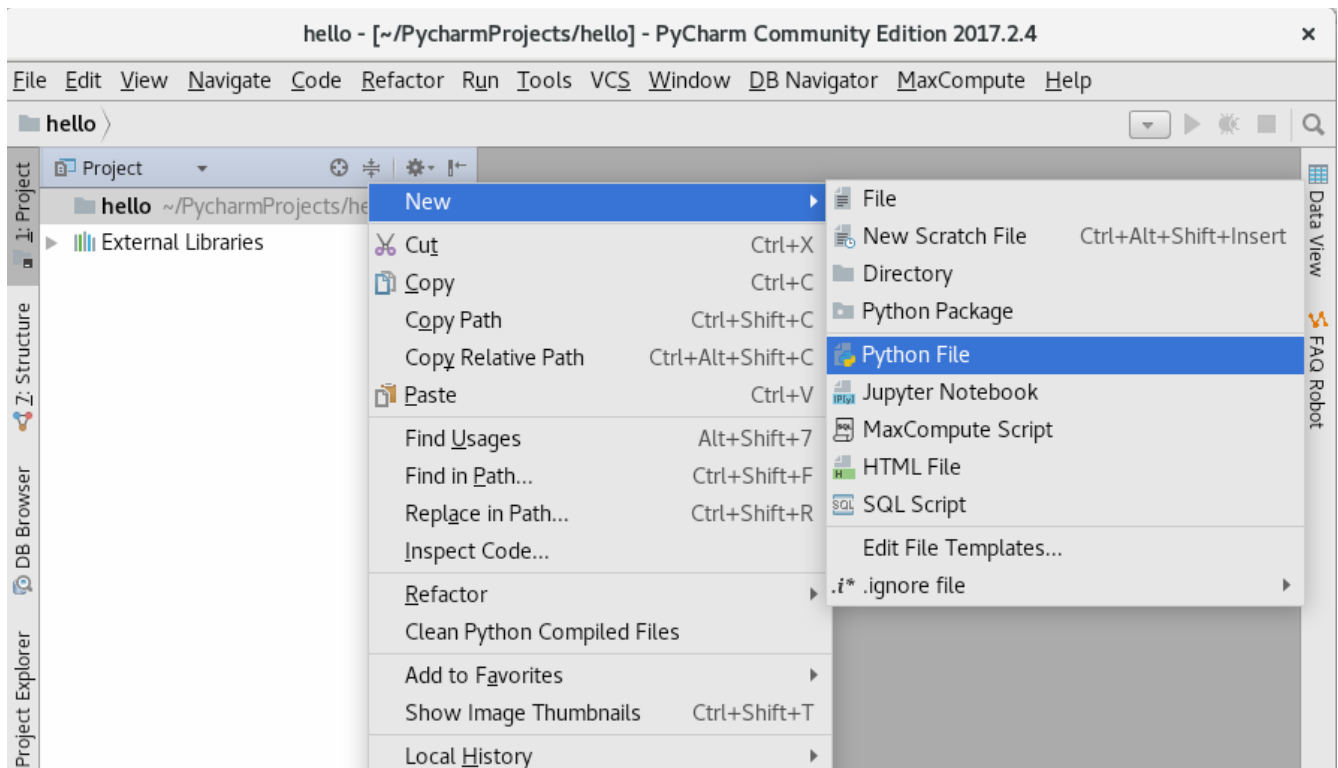
Gambar 6: Install Flask

1. Klik pada bagian Terminal (seperti pada nomor 1 di Gambar 6)
2. Kemudian ketik: pip install Flask (seperti pada nomor 2 di Gambar 6)
3. Kemudian tekan enter, dan tunggu sampai prosesnya selesai.

Hello World Flask

Seperti pada umumnya ketika pertama kali berkenalan dengan bahasa pemrograman yaitu membuat program Hello World yang bertujuan untuk mengenalkan struktur dasar dari bahasa tersebut. Di sini kita akan membuat program Hello World Flask, untuk itu langkahnya adalah sebagai berikut:

1. Buat file **hello.py**, caranya adalah klik kanan pada panel sebelah kiri di folder hello, seperti pada gambar di bawah ini



Gambar 7: Create New File

2. Lalu pilih New >> Python File
3. Kemudian isi nama file, dalam hal ini **hello.py**
4. Kemudian klik OK, maka akan muncul halaman *hello.py*
5. Berikutnya ketik isi dari hello.py seperti pada gambar di bawah

 A screenshot of the 'hello.py' file in PyCharm. The code is as follows:

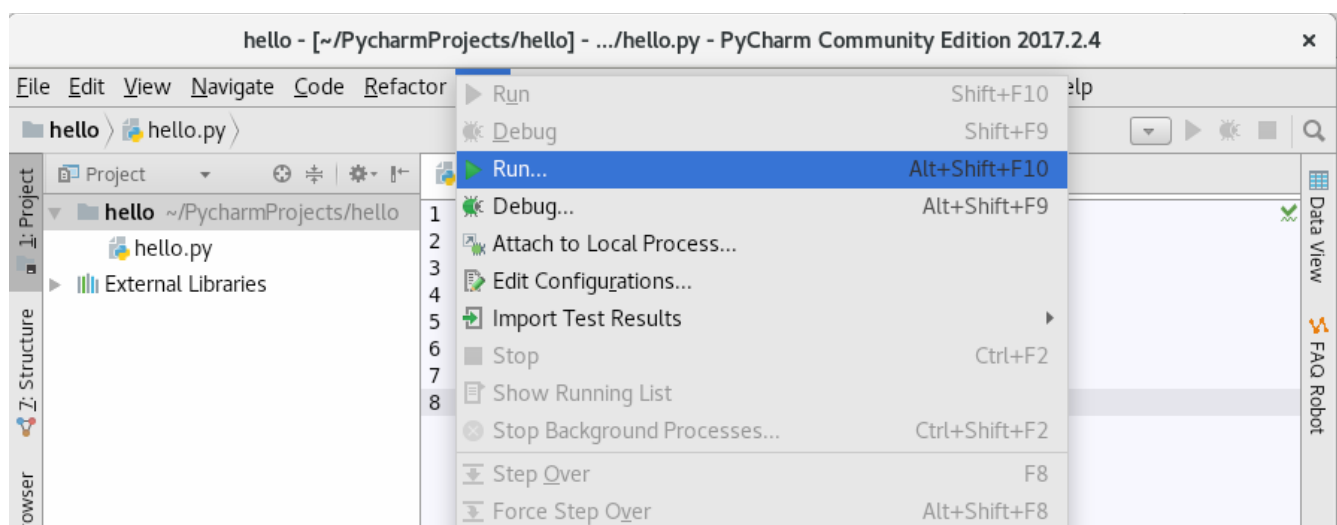

```

1  from flask import Flask
2  app = Flask(__name__)
3
4
5  @app.route('/')
6  def hello():
7      return "Hello Unas"
8
9  app.run()
10
  
```

Gambar 8: Hello World Flask

Pada baris pertama kita mengimpor kelas Flask, instance dari kelas ini merupakan aplikasi WSGI, lalu pada baris kedua kita membuat instance dari kelas Flask, argumen pertama adalah nama dari aplikasi. Baris ke lima kita menggunakan dekorator route() untuk memberitahu URL yang akan memicu eksekusi fungsi hello() kepada Flask, dalam hal ini adalah URL “/”. Baris ke 6 dan 7 adalah fungsi yang akan dijalankan ketika ada request ke URL “/” atau root URL, fungsi ini hanya akan menampilkan text “Hello Unas” di browser, dan baris terakhir yaitu baris ke 9 kita menjalankan aplikasi.

6. Lalu jalankan, dengan cara klik menu “Run”



Gambar 9: Run

7. Tunggu beberapa saat, dan perhatikan pada bagian bawah PyCharm akan muncul keterangan aplikasi Flask berjalan pada alamat <http://127.0.0.1:5000/>
8. Buka alamat tersebut di browser
9. Maka akan muncul “Hello Unas” di browser
10. Perhatikan pada bagian kanan atas PyCharm, terdapat tombol untuk menghentikan dan menjalankan.



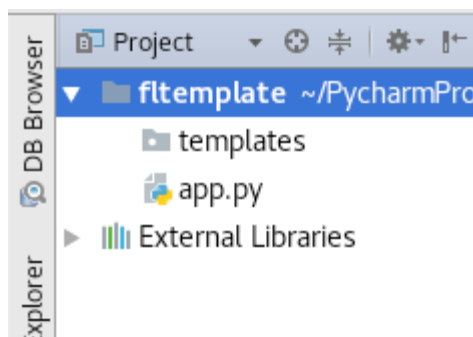
Gambar 10: Tombol Untuk Menjalankan dan Menghentikan

Sampai di sini kita sudah berkenalan dengan PyCharm dan juga berkenalan dengan framework Flask, hal ini sangat penting sekali untuk proses selanjutnya, dengan pengetahuan dasar ini dapat memperlancar proses belajar untuk materi-materi berikutnya.

2. Flask dan Template

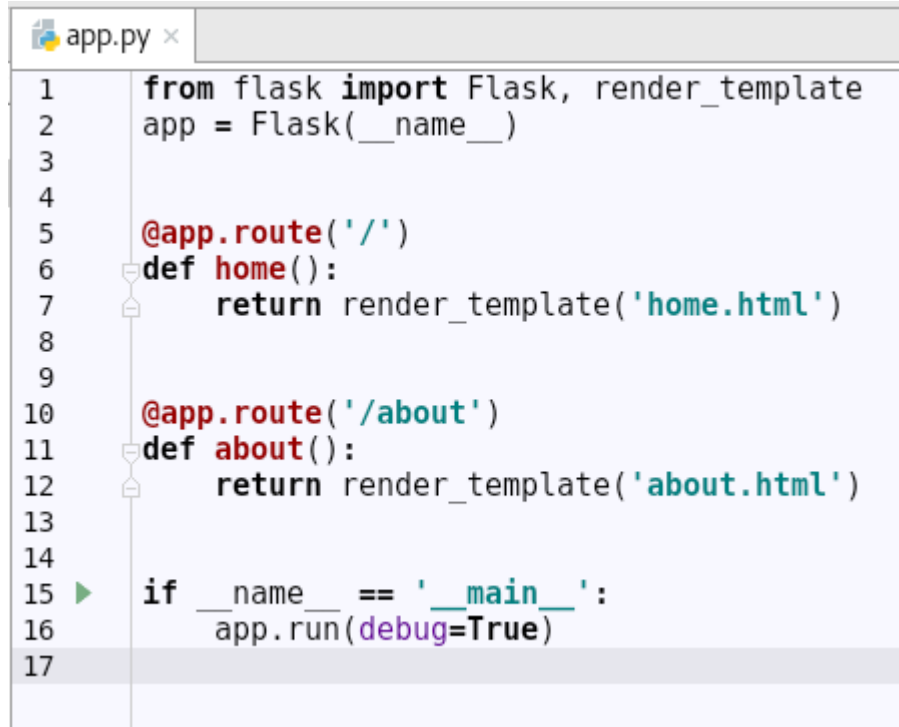
Sampai di sini kita sudah bisa menggunakan PyCharm dan juga sudah paham dengan struktur dasar aplikasi Flask, selanjutnya kita akan berkenalan dengan penggunaan template di Flask menggunakan mesin template Jinja2.

1. Langkah pertama adalah membuat project baru, langkahnya adalah seperti yang sudah dilakukan pada pembahasan sebelumnya.
2. Kemudian buat sebuah file Python dengan nama '**app.py**', langkahnya seperti yang sudah dilakukan pada pembahasan sebelumnya.
3. Selanjutnya buat sebuah direktori baru dengan nama '**templates**', sehingga hasil akhirnya adalah seperti pada gambar berikut



Gambar 11: Struktur Project 2

4. Kemudian buka file **app.py** dan ubah isinya hingga seperti pada gambar berikut

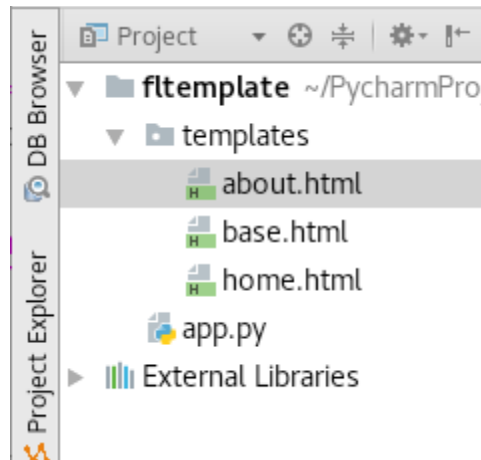


```
1  from flask import Flask, render_template
2  app = Flask(__name__)
3
4
5  @app.route('/')
6  def home():
7      return render_template('home.html')
8
9
10 @app.route('/about')
11 def about():
12     return render_template('about.html')
13
14
15 if __name__ == '__main__':
16     app.run(debug=True)
17
```

Gambar 12: File app.py

Pada aplikasi ini kita menggunakan *method* **render_template** untuk me-render template, kemudian kita melewati sebuah argumen yaitu nama dari file template yang akan kita render, contohnya pada fungsi **home()** kita merender template **home.html**, Flask akan mencari file tersebut di dalam folder **templates**.

5. Langkah selanjutnya adalah buat tiga buah file html di bawah folder templates dengan nama: base.html, home.html, dan about.html, hingga hasil akhirnya adalah seperti pada gambar berikut:



Gambar 13: File Template

6. Berikutnya kita akan membuat base template yang berisi kerangka dasar sebuah berkas html, base template ini akan digunakan oleh template-template yang lain.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <title>{% block title %}{% endblock %}</title>
6  </head>
7  <body>
8      {% block content %}
9      {% endblock %}
10 </body>
11 </html>
12

```

Gambar 14: Base Template

Flask menggunakan Jinja2 (<http://jinja.pocoo.org>) sebagai template engine, salah satu fitur yang sangat berguna adalah *inheritance*, kita dapat membuat sebuah base template yang berisi komponen-komponen umum yang akan digunakan oleh beberapa template, kemudian base template tersebut dapat digunakan oleh template turunannya dengan mengisi *block-block* yang sudah didefinisikan. Pada file base.html di baris ke 5 kita membuat sebuah *block* dengan nama **title**, begitu juga dengan baris ke 8 dan 9, perintah *block* memberitahu template engine bahwa sub template dapat menempa isi dari *block* tersebut.

7. Langkah berikutnya kita membuat file **home.html** dan mengisinya menjadi seperti gambar di bawah ini.

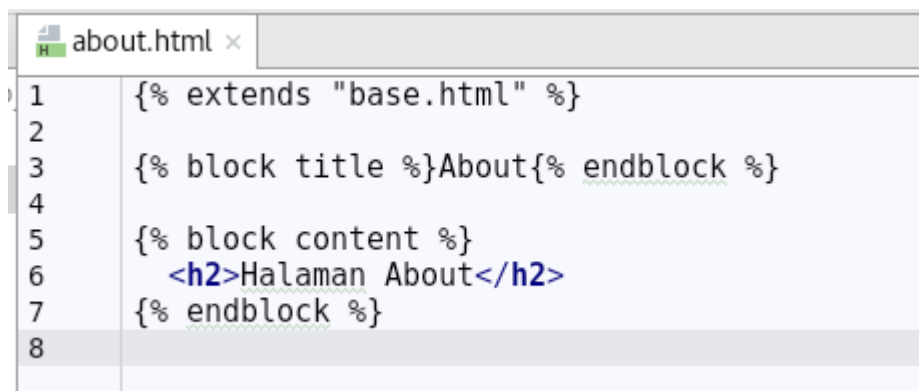


```
home.html x
1 {% extends "base.html" %}
2
3 {% block title %}Home{% endblock %}
4
5 {% block content %}
6     <h2>Halaman Home</h2>
7 {% endblock %}
8
```

Gambar 15: File home.html

Pada baris pertama kita menggunakan `{% extends "base.html" %}` untuk memberitahu template engine bahwa template **home.html** merupakan perluasan (extends) dari template lain. Baris ke tiga kita mengisi block title dengan teks "Home". Kemudian baris 5,6,7 kita mengisi block konten.

8. Selanjutnya kita akan membuat template about, buka file about.html dan isi file tersebut sehingga seperti gambar di bawah.



```
about.html x
1 {% extends "base.html" %}
2
3 {% block title %}About{% endblock %}
4
5 {% block content %}
6     <h2>Halaman About</h2>
7 {% endblock %}
8
```

Gambar 16: File about.html

9. Sampai di sini kita sudah selesai membuat aplikasi contoh penggunaan template di Flask, jalankan aplikasi lalu test di browser untuk halaman home dan about.

2. Flask dan Database

Pada bab ini kita akan membahas tentang bagaimana mengelola data dengan database menggunakan Flask untuk membuat aplikasi berbasis web, kita akan menggunakan sebuah contoh kasus yaitu mengelola data anggota yang akan disimpan di database, aplikasi diharapkan dapat melakukan beberapa hal berikut ini:

1. Menampilkan daftar anggota
2. Menambah data anggota
3. Memperbarui data anggota
4. Menghapus data anggota

Aplikasi seperti ini sering disebut sebagai CRUD (create, read, update, delete), aplikasi ini akan kita gunakan sebagai sarana belajar mengenal proses pembuatan aplikasi berbasis web untuk mengelola data di database menggunakan Flask.

Untuk kebutuhan di atas, maka kita dapat memanfaatkan ekstensi Flask yang tersedia dan cukup lengkap, daftar extension Flask dapat dilihat pada link <http://flask.pocoo.org/extensions/> . Adapun aplikasi ini akan menggunakan extension-extension berikut ini:

- Flask-SQLAlchemy – digunakan untuk menambahkan dukungan SQLAlchemy pada aplikasi Flask, SQLAlchemy sendiri merupakan pustaka Python untuk menghadirkan fitur Object Relational Mapping (ORM) dalam mengakses database.
- Flask-WTF – digunakan untuk bekerja dengan HTML form

Untuk database yang digunakan yaitu MySQL, untuk memudahkan instalasi MySQL maka dapat menggunakan aplikasi XAMPP. Untuk mengakses database MySQL menggunakan Python maka kita membutuhkan library **mysqlclient**, untuk itu kita juga butuh memasang library ini ke dalam virtualenv yang sudah kita buat sebelumnya.

Persiapan

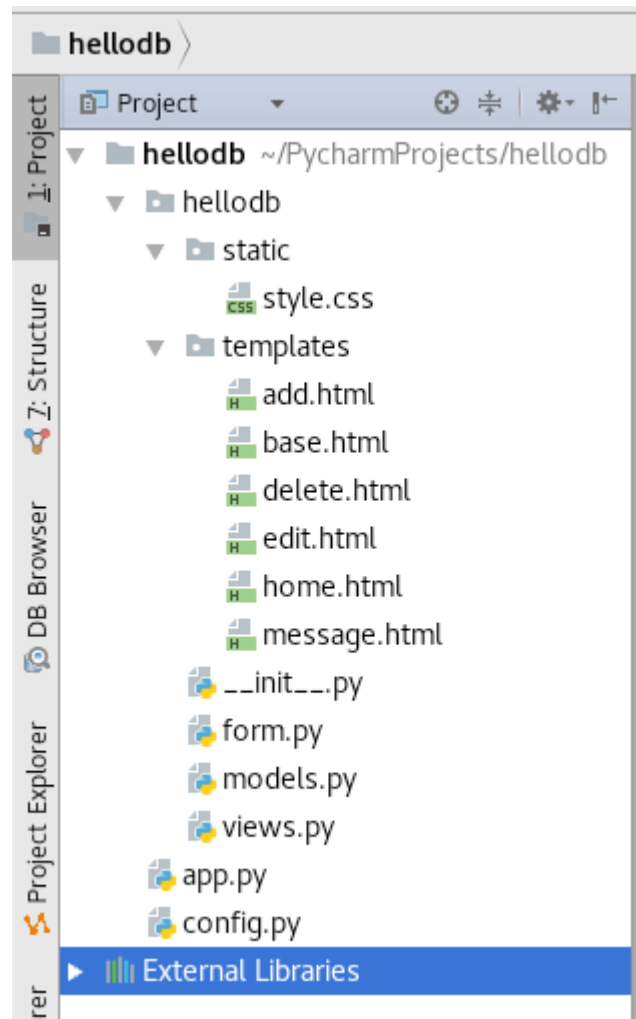
Kita akan memasang library-library python yang dibutuhkan menggunakan **pip**, pada bab sebelumnya sudah dibahas. Ketik perintah berikut di terminal yang terdapat pada PyCharm:

pip install Flask-SQLAlchemy Flask-WTF mysqlclient

Tunggu hingga proses selesai.

Struktur Direktori

Flask tidak ada aturan spesifik mengenai struktur direktori, kita bebas untuk menyusun struktur sesuai dengan preferensi masing-masing, untuk project ini kita akan menggunakan struktur direktori seperti pada gambar di bawah.



Gambar 17: Struktur Direktori Project

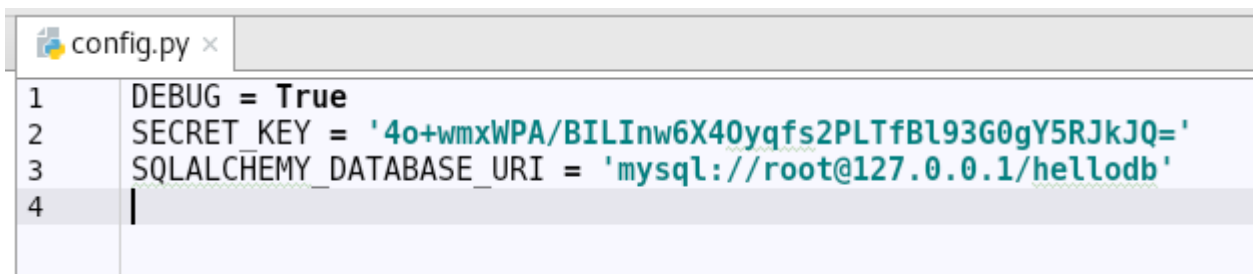
1. Direktori **hellodb** yang paling atas adalah direktori root project, yang akan menampung seluruh file yang digunakan di dalam project ini.
2. Di bawah direktori **hellodb** terdapat file dan folder: **hellodb**, **app.py**, dan **config.py**. Kenapa ada folder **hellodb** lagi?, direktori **hellodb** yang ke dua digunakan untuk menampung file-file aplikasi yang di bangun, jadi ada direktori **hellodb** untuk keseluruhan project, dan ada folder **hellodb** untuk aplikasi. File **app.py** digunakan untuk menjalankan aplikasi, dan **config.py** untuk menyimpan pengaturan project.

3. Di dalam direktori **hellodb** yang ke dua, terdapat direktori dan file:

- Folder **static** : digunakan untuk menyimpan file-file seperti css, javascript, images.
- Folder **templates** : digunakan untuk menyimpan file-file template html
- File `__init__.py` : File `__init__.py` digunakan untuk menandai agar sebuah direktori dianggap sebuah package Python.
- File `form.py` : File ini berisi kode program untuk form
- File `models.py` : Mendefinisikan model
- File `views.py` : Berisi kode program untuk menangani request

Kita akan membuat satu per satu file dan direktori di atas.

1. File `config.py`

A screenshot of a code editor window titled 'config.py'. The editor shows four lines of Python code. Line 1: 'DEBUG = True'. Line 2: 'SECRET_KEY = '4o+wmxWPA/BILInw6X40yqfs2PLTfBl93G0gY5RJkJQ=''. Line 3: 'SQLALCHEMY_DATABASE_URI = 'mysql://root@127.0.0.1/hellodb''. Line 4: A cursor is positioned at the end of the line.

```
1 DEBUG = True
2 SECRET_KEY = '4o+wmxWPA/BILInw6X40yqfs2PLTfBl93G0gY5RJkJQ='
3 SQLALCHEMY_DATABASE_URI = 'mysql://root@127.0.0.1/hellodb'
4 |
```

Gambar 18: File `config.py`

File `config` berisi konfigurasi aplikasi, `DEBUG` dan `SECRET_KEY` merupakan konfigurasi yang digunakan di dalam framework Flask, sementara untuk `SQLALCHEMY_DATABASE_URI` digunakan oleh Flask-SQLAlchemy untuk konfigurasi koneksi database. Daftar konfigurasi internal Flask dapat dilihat pada tautan berikut: <http://flask.pocoo.org/docs/0.12/config/>

2. File `app.py`



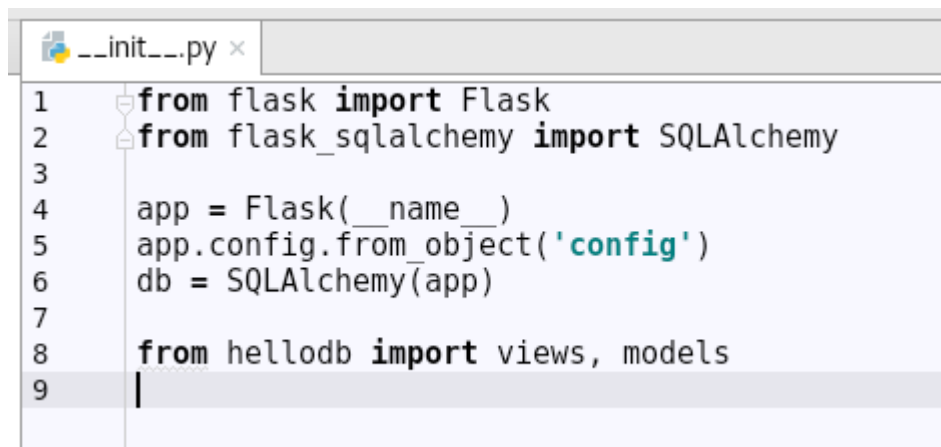
```

1  from hellodb import app
2
3
4  if __name__ == '__main__':
5      app.run()
6

```

Gambar 19: File app.py

3. File __init__.py



```

1  from flask import Flask
2  from flask_sqlalchemy import SQLAlchemy
3
4  app = Flask(__name__)
5  app.config.from_object('config')
6  db = SQLAlchemy(app)
7
8  from hellodb import views, models
9

```

Gambar 20: File __init__.py

File __init__.py digunakan untuk menandai agar sebuah direktori dianggap sebuah package Python, file ini bisa berupa file kosong, atau juga bisa berisi perintah untuk inisiasi sebuah package. Dalam hal ini file __init__.py yang kita buat berisi inisiasi dari aplikasi hellodb. Pada baris ke 5 kita memberitahu flask untuk membaca konfigurasi dari berkas config.py dengan perintah **app.config.from_object('config')**.

Lalu pada baris terakhir kita mengimpor **views** dan **models**, praktik ini sebenarnya tidak sesuai dengan PEP8 (<https://www.python.org/dev/peps/pep-0008/>) namun penjelasan dari Flask dapat dibaca pada tautan berikut: <http://flask.pocoo.org/docs/0.12/patterns/packages/>

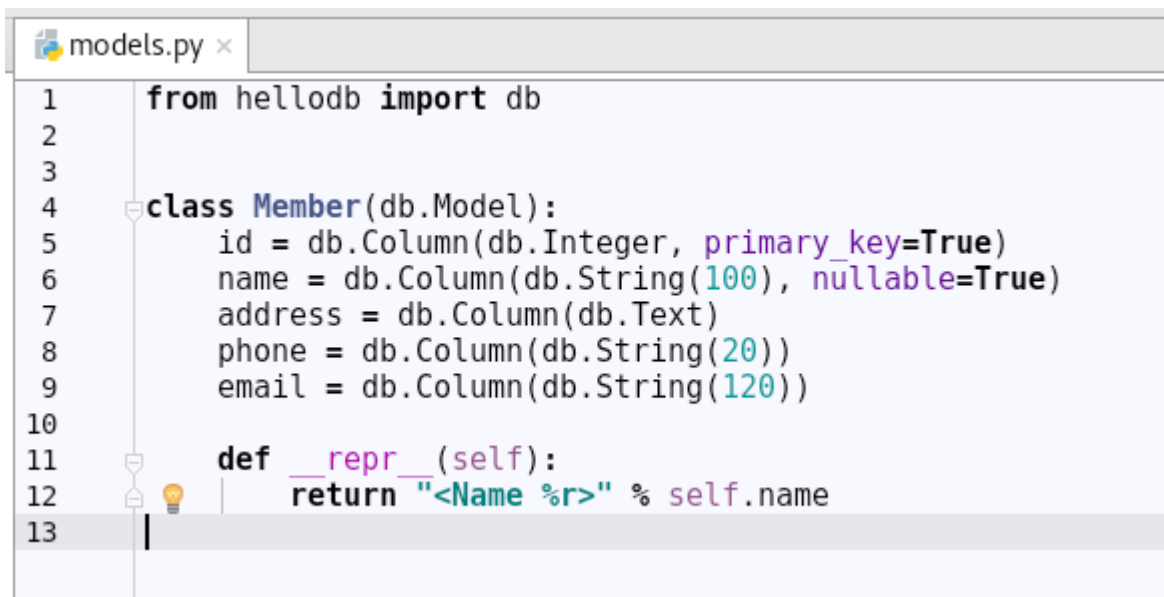
4. File form.py


```
form.py x
1  from flask_wtf import FlaskForm
2  from wtforms import StringField, TextAreaField
3  from wtforms.validators import DataRequired, Length
4
5
6  class MemberForm(FlaskForm):
7      name = StringField('Nama', validators=[DataRequired()],
8                          render_kw={"placeholder": "Nama"})
9      address = TextAreaField('Alamat',
10                              render_kw={"placeholder": "Alamat"})
11     phone = StringField('No. Telp', validators=[Length(max=20)],
12                          render_kw={"placeholder": "No. Telp"})
13     email = StringField('Email', validators=[Length(max=120)],
14                          render_kw={"placeholder": "Email"})
15
```

Gambar 21: File form.py

File form.py berisi definisi form yang akan kita gunakan, dalam aplikasi ini kita menggunakan extension Flask-WTF untuk memudahkan dalam bekerja dengan form.

5. File models.py



```
1 from hellobdb import db
2
3
4 class Member(db.Model):
5     id = db.Column(db.Integer, primary_key=True)
6     name = db.Column(db.String(100), nullable=True)
7     address = db.Column(db.Text)
8     phone = db.Column(db.String(20))
9     email = db.Column(db.String(120))
10
11     def __repr__(self):
12         return "<Name %r>" % self.name
13
```

Gambar 22: File models.py

Pada file models.py, kita membuat sebuah model Member, sebuah model secara sederhana dapat diartikan sebagai skema atau struktur tabel di dalam database. Pada baris ke empat, db.Model merupakan base Model yang terdapat pada SQLAlchemy, ketika kita mendeklarasikan sebuah model kelas yang kita buat harus menurunkan dari basis model ini.

Lalu untuk mendefinisikan sebuah field kita gunakan Column, argumen pertama merupakan type data, daftar type data yang tersedia dapat dibaca melalui tautan berikut: <http://flask-sqlalchemy.pocoo.org/2.3/models/>

Pada baris ke 11, kita membuat sebuah fungsi __repr__. Fungsi __repr__ dimaksudkan untuk mewakili object ketika sebuah object tersebut dicetak misalnya dengan perintah print().

6. File view.py

```
views.py x
1  from flask import render_template, request, flash, redirect, url_for
2  from hellodb import app, db
3  from hellodb.models import Member
4  from hellodb.form import MemberForm
5
6
7  @app.route('/')
8  def home():
9      members = Member.query.limit(10).all()
10     return render_template('home.html', members=members)
11
12
13  @app.route('/add', methods=['GET', 'POST'])
14  def add_member():
15      form = MemberForm(request.form)
16      if form.validate_on_submit():
17          member = Member()
18          member.name = form.name.data,
19          member.address = form.address.data,
20          member.phone = form.phone.data,
21          member.email = form.email.data
22
23          db.session.add(member)
24          db.session.commit()
25          flash('Data berhasil disimpan.', 'success')
26          return redirect(url_for("home"))
27
28     return render_template("add.html", form=form)
29
```

```

30
31 @app.route('/edit/<id>', methods=['GET', 'POST'])
32 def edit_member(id):
33     member = Member.query.filter_by(id=id).first_or_404()
34     form = MemberForm(request.form, obj=member)
35     if form.validate_on_submit():
36         form.populate_obj(member)
37         db.session.commit()
38         flash('Data berhasil diperbarui.', 'success')
39         return redirect(url_for("home"))
40     return render_template("edit.html", form=form, member=member)
41
42
43 @app.route('/delete/<id>', methods=['GET', 'POST'])
44 def delete_member(id):
45     member = Member.query.filter_by(id=id).first_or_404()
46     if request.method == 'POST':
47         db.session.delete(member)
48         db.session.commit()
49         flash('Data berhasil dihapus.', 'success')
50         return redirect(url_for('home'))
51
52     return render_template("delete.html", member=member)
53

```

Gambar 23: File views.py

Pada file views.py kita membuat empat buah fungsi, yaitu:

- home : untuk menampilkan halaman awal dan menampilkan daftar member
- add_member : untuk menampilkan form tambah anggota dan menangani request POST dan menambahkan data anggota ke database
- edit_member : untuk menampilkan form edit anggota dan menangani request POST untuk memperbarui data anggota
- delete_member : untuk menampilkan halaman konfirmasi dan menangani request POST untuk menghapus data anggota.

7. File add.html

```
add.html x
1  {% extends "base.html" %}
2  {% block title %}Add Member{% endblock %}
3
4  {% block content %}
5
6      <h3 class="page-header">Add Member</h3>
7
8      {% for field in form.errors %}
9      {% for error in form.errors[field] %}
10         <div class="alert alert-danger">
11             <strong>Error!</strong> {{error}}
12         </div>
13     {% endfor %}
14 {% endfor %}
15
16     <form action="{{ url_for('add_member') }}" method="post">
17         {{ form.csrf_token }}
18         <div class="form-group">
19             {{ form.name.label }}
20             {{ form.name(class="form-control") }}
21         </div>
22         <div class="form-group">
23             {{ form.address.label }}
24             {{ form.address(class="form-control") }}
25         </div>
26         <div class="form-group">
27             {{ form.phone.label }}
28             {{ form.phone(class="form-control") }}
29         </div>
30         <div class="form-group">
31             {{ form.email.label }}
32             {{ form.email(class="form-control") }}
33         </div>
34         <button type="submit" class="btn btn-default">Submit</button>
35     </form>
36
37 {% endblock %}
38
```

Gambar 24: File add.html

Selanjutnya untuk file-file template dan css dapat dilihat di github: <https://github.com/somat/python-unas/tree/master/projects>

Hasil akhir dari aplikasi adalah sebagai berikut:

Hello DB	Home	About	Contact
----------	------	-------	---------

Daftar Member		Add Member
Nama	Alamat	Aksi
Bambang Pamungkas	Jl. Setiabudi, No. 50, Jakarta Pusat.	Edit Delete
Egi Sujana	Jalan Merpati 1, No. 50, Sumatera.	Edit Delete
Subagyo	Jalan Waru, No. 50, Palembang.	Edit Delete

Referensi:

1. Dokumentasi Flask <http://flask.pocoo.org>
2. Dokumentasi PyCharm <https://www.jetbrains.com/pycharm/>
3. Jinja2 <http://jinja.pocoo.org>
4. VirtualEnv <https://virtualenv.pypa.io/en/stable/>
5. Flask-SQLAlchemy <http://flask-sqlalchemy.pocoo.org>
6. Flask-WTF <https://flask-wtf.readthedocs.io/en/stable/>