Modul Pelatihan

Python with Django Framework

Instruktur:

Erick Kurniawan, M.Kom, MCPD, MCT, MVP

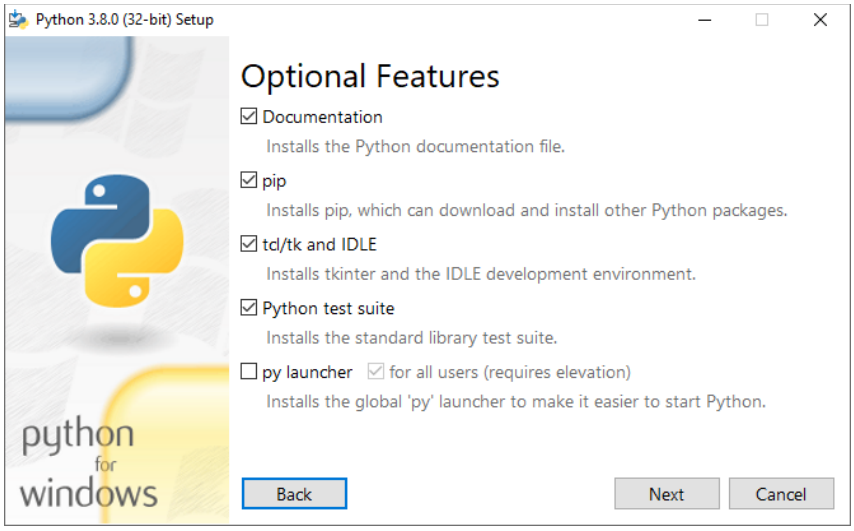
Actual Training

http://actual-training.com
info@actual-training.com
Jl.Gowongan Lor 46, Yogyakarta
087876133054 / (0274) 566584

Daftar Isl

Instalasi Python beserta dengan Pip

Untuk instalasi python 3 beserta dengan library pip, download python pada link berikut: https://www.python.org/downloads/windows/



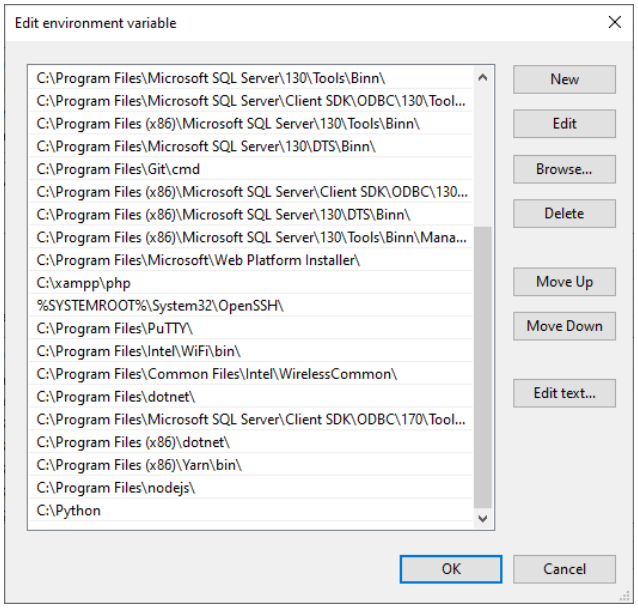
Set path agar perintah dikenali, anda dapat menambahkan PATH C:\Python pada Environment Variable.

```
C:\>set PATH=C:\Program Files\Python 3.8;%PATH%
C:\>set PYTHONPATH=%PYTHONPATH%;C:\My_python_lib
C:\>python
```

Sesuai tutorial berikut: https://www.codepolitan.com/tutorial/membangun-aplikasi-mini-hrd-django-pengenalan-django-admin

Install di folder C:\Python, kemudian cari folder Script.

Setting system environment variable



Python Editor

Install pylint untuk vs code

pip install pylint-django

Membuat Project Python baru

Untuk membuat proyek baru di Django cukup mudah. Berbeda dengan beberapa web framework yang bisa digunakan setelah mengunduh source code-nya, installer Django disimpan di sistem operasi untuk kemudian lagi di-update versinya bila diperlukan. Hanya dengan menggunakan django-admin.py di konsol kita dapat membuat sebuah proyek Django baru. Untuk tutorial ini silahkan simpan saja di folder manapun yang Anda sukai, kemudian panggil perintah berikut untuk membuat proyek baru.

Membuat Project Baru

Untuk membuat project baru buka console dan masuk ke folder C:\Phyton\Scripts

```
C:\Python\Scripts>django-admin startproject learning_site
```

Sekarang kita coba lihat ada apa sajakah di dalam sebuah proyek Django yang baru lahir.

manage.py merupakan file yang akan menjadi jembatan untuk menggunakan perintah konsol yang dimiliki Django, ada cukup banyak perintah konsol yang dimiliki Django seperti membuat modul baru, meng-generate database dari ORM ke RDBMS, dan lainnya.

Learning_site/ _ _ init _ _ .py merupakan sebuah file yang menjadi penanda bagi sebuah direktori di dalam proyek aplikasi agar dikenali Python dan dapat terlibat dalam proses import modul dan file. Cocok sekali bila kita terbiasa membuat sebuah aplikasi terbagi menjadi beberapa file atau modul.

Learning_site/settings.py merupakan file yang berisi pengaturan Django mulai dari pengaturan database, pengaturan dimana file statik seperti CSS dan Javascript berada, pengaturan aplikasi atau sistem yang terintegrasi dengan aplikasi Django, dan berbagai pengaturan lainnya.

urls.py merupakan file yang berisi pemetaan URL ke views atau modul yang ada di dalam Django, misal ketika kita ingin mengakses http://localhost:8000/about maka kita akan diarahkan ke urls.py kemudian dia akan melihat views mana yang akan memproses request tersebut, misal nama views-nya adalah homepage.views.about

wsgi.py merupakan file yang akan digunakan ketika akan melakukan proses deploy aplikasi Django di web server seperti Apache atau Nginx

manage.py adalah entry point

Untuk menjalankan aplikasi yang baru saja dibuat pada server tambahkan kode berikut:

learning_site>python manage.py runserver

anda dapat melihat pesan yang ditampilkan.

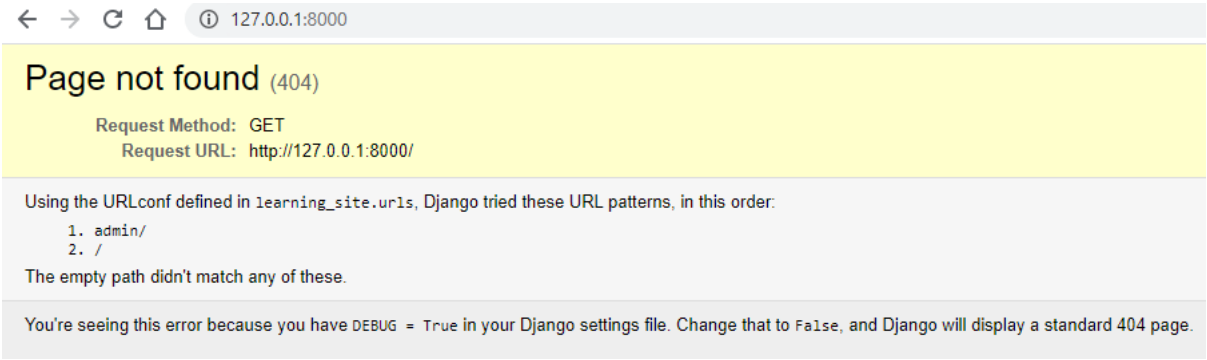
```
You have 17 unapplied migration(s). Your project may not work properly until you apply the migrations
for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
December 26, 2019 - 20:03:38
Django version 3.0, using settings 'learning_site.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CTRL-BREAK.
```

Pesan ini keluar karena anda belum menjalankan perintah migrasi.

Jalankan perintah >python manage.py migrate

Maka akan ditambahkan file sqlite3.db kedalam folder yang kita buat.

Kemudian jalankan runserver

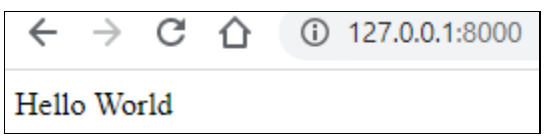


Pada root folder learning_site tambahkan views.py, kemudian tambahkan kode berikut ini:

```python
from django.http import HttpResponse
def hello_world(request):
    return HttpResponse('Hello World')
```

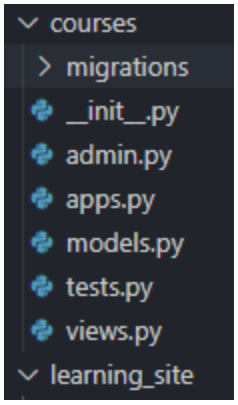
kemudian perlu untuk set routing pada file url.py

```python
from django.contrib import admin
from django.urls import path
from . import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('',views.hello_world),
]
```



Membuat Django Apps

learning_site>py manage.py startapp courses



Untuk menambahkan apps yang barusan dibuat kedalam aplikasi kita, tambahkan pengaturan di settings.py

```
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'Asia/Jakarta'
```

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'courses'
]
```

Model Administration

Pada courses\models.py tambahkan kode berikut:

```
from django.db import models


# Create your models here.
class Course(models.Model):
    created_at=models.DateTimeField(auto_now_add=True)
    title = models.CharField(max_length=255)
    description = models.TextField()
```

Jalankan perintah migrasi berikut:

learning_site>py manage.py makemigrations courses

```
Migrations for 'courses':
  courses\migrations\0001_initial.py
    - Create model Course
```

Pada folder migrations dapat dilihat detail table dan field yang akan ditambahkan. Kemudian jalankan perintah migrations.

learning_site>py manage.py migrate courses

```
Operations to perform:
  Apply all migrations: courses
Running migrations:
  Applying courses.0001_initial... OK
```

```
C:\Python\Scripts\learning_site>py manage.py shell
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from courses.models import Course
>>> c = Course()
>>> c.title = "Python Basics"
>>> c.description = "Learn the basics of Python"
>>> c.save()
>>> Course.objects.all()
<QuerySet [<Course: Course object (1)>]>
```

Buat record kedua

```
>>> from courses.models import Course
>>> Course(title="Python Collection",description="Learn about list,dict, and tuple").save()
>>> Course.object.all()
```

Buat record ketiga

```
>>> Course.objects.create(title="OOP with Python",description="Learn about Python classes")
<Course: Course object (3)>
```

Tambahkan method berikut pada model

```
class Course(models.Model):
    created_at=models.DateTimeField(auto_now_add=True)
    title = models.CharField(max_length=255)
    description = models.TextField()


    def __str__(self):
        return self.title
```

kemudian masuk kembali kedalam shell, dan jalankan perintah berikut

```
>>> from courses.models import Course
>>> Course.objects.all()
<QuerySet [<Course: Python Basics>, <Course: Python Collection>, <Course: OOP with Python>]>
```

Menampilkan Daftar Courses

Pada app course tambahkan kode berikut pada views.py untuk menampilkan daftar courses.

```
from django.http import HttpResponse
from django.shortcuts import render
from .models import Course

def course_list(request):
    courses = Course.objects.all()
    output = ', '.join([str(course) for course in courses])
    return HttpResponse(output)
```

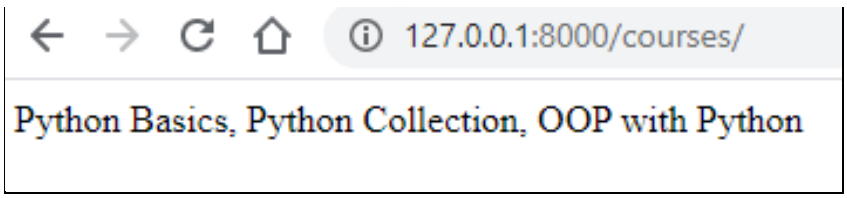
kemudian agar dapat diakses dari route, tambahkan file urls.py pada app Courses

```
from django.urls import path
from . import views

urlpatterns = [
    path("",views.course_list)
]
```

Pada root folder tambahkan juga kode berikut pada file urls.py

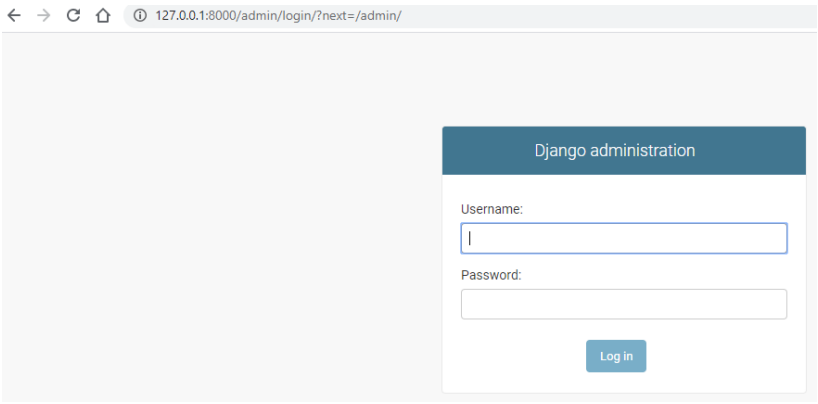
```
from django.contrib import admin
from django.urls import path,include
from . import views

urlpatterns = [
    path('courses/',include('courses.urls')),
    path('admin/', admin.site.urls),
    path('',views.hello_world),
]
```

Kemudian jalankan halaman berikut:



Django Admin

Django menyediakan fitur admin yang dapat digunakan untuk kebutuhan pembuatan user.

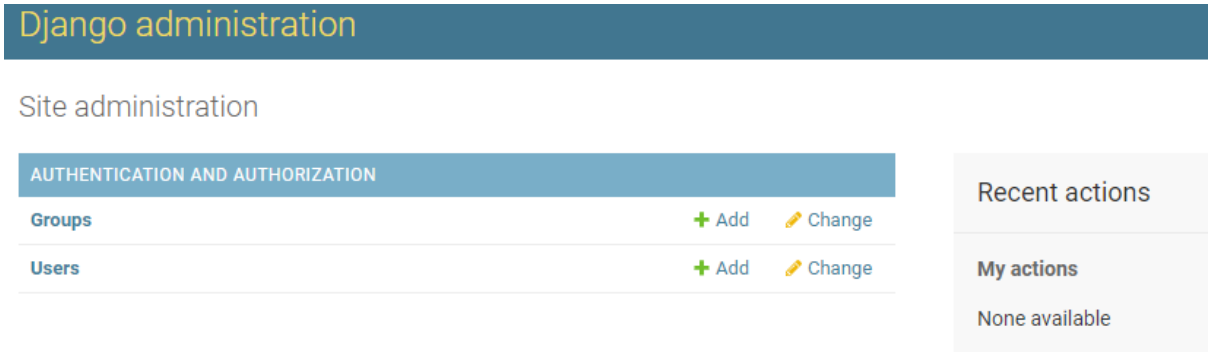


Membuat Akun Superuser

Untuk membuat user tambahkan perintah berikut untuk membuat superuser.

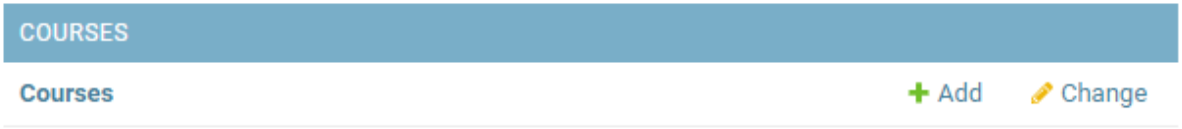
```
C:\Python\Scripts\learning_site>py manage.py createsuperuser
Username (leave blank to use 'erick'): erick
Email address: erick@actual-training.com
Password:
Password (again):
Superuser created successfully.
```

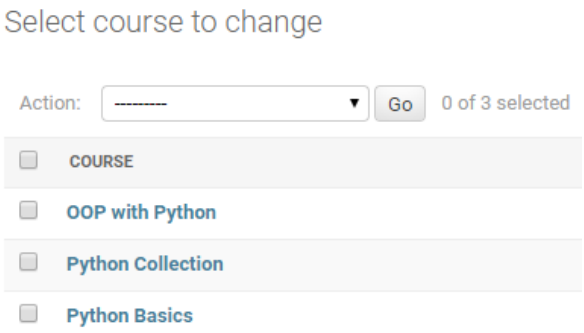
Setelah itu anda dapat melakukan login kedalam menu admin



Untuk menambahkan Courses yang sudah kita buat, pada Courses app folder tambahkan kode berikut pada file admin.py

```
from django.contrib import admin
from .models import Course


admin.site.register(Course)
```



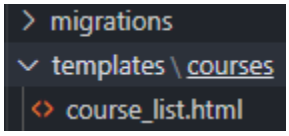


Anda juga dapat menambahkan record baru kedalam Courses.

Django Template

Template dapat digunakan sebagai master page template untuk semua halaman.

Pada Courses apps folder tambahkan folder baru dengan nama templates\courses, kemudian tambahkan file course_list.html



Tambahkan kode template berikut:

```
{% for course in courses %}
<h2>{{ course.title }}</h2>
{{ course.description }}
{% endfor %}
```

Kemudian ubah kode pada views.py

```
def course_list(request):
    courses = Course.objects.all()
    return render(request,'courses/course_list.html',{'courses':courses})
```

kemudian coba tampilkan view courses

Tambahkan folder template kedalam root folder (folder learning_site yang paling atas). Kemudian tambahkan file home.html.

```
<h2>Welcome to Django</h2>
```

Kemudian pada folder learning_site, buka file settings.py kemudian tambahkan pengaturan template default.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            ------
        },
    },
]
```

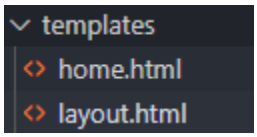
Pada views.py tambahkan kode berikut

```
from django.shortcuts import render

def hello_world(request):
    return render(request,'home.html')
```

kemudian jalankan aplikasi:

Template Inheritance Python

Tambahkan file layout.py kedalam folder templates



Kemudian tambahkan kode berikut untuk layout

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>{% block title %}{% endblock title %}</title>
</head>
<body>
    {% block content %}{% endblock content %}
</body>
</html>
```

Kemudian pada home.py tambahkan kode berikut:

```
{% extends "layout.html" %}

{% block title %}Django Course{% endblock %}


{% block content %}
<h2>Welcome to Django</h2>
{% endblock %}
```

Pada courses tambahkan juga pada file course_list.html

```
{% extends 'layout.html' %}
```

```
{% block title %} Available Courses {% endblock %}

{% block content %}
    {% for course in courses %}
        <h2>{{ course.title }}</h2>
        {{ course.description }}
    {% endfor %}
{% endblock content %}
```

Menambahkan Static Asset

Untuk membuat static assets, tambahkan folder assets pada root folder learning_site, kemudian buat folder dengan nama css pada folder assets. Pada folder css, tambahkan file dengan nama layout.css.

```
html {
    box-sizing: border-box; }

  *, *::after, *::before {
    box-sizing: inherit; }

  .site-container {
    max-width: 68em;
    margin-left: auto;
    margin-right: auto; }
    .site-container::after {
      clear: both;
      content: "";
      display: table; }
    .site-container nav {
      width: 100%;
      display: table;
      width: 100%;
      table-layout: fixed;
      border-bottom-left-radius: 10px;
      border-bottom-right-radius: 10px;
      border-top-left-radius: 10px;
      border-top-right-radius: 10px;
      border-bottom-left-radius: 10px;
      border-top-left-radius: 10px;
```

```
    border-bottom-right-radius: 10px;
    border-top-right-radius: 10px;
    background-color: #642039;
    margin-bottom: 1em;
    overflow: hidden; }
    .site-container nav a {
      display: table-cell;
      width: 8.33333%;
      padding: 10px;
      border-bottom: 2px solid #642039;
      color: #e6557e;
      letter-spacing: 0.075em;
      line-height: 60px;
      text-align: center;
      text-decoration: none; }
      .site-container nav a:hover {
        background: #6d3753;
        border-bottom: 2px solid #642039;
        color: white; }

.cards {
  display: -webkit-box;
  display: -moz-box;
  display: box;
  display: -webkit-flex;
  display: -moz-flex;
  display: -ms-flexbox;
  display: flex;
  -webkit-box-lines: multiple;
  -moz-box-lines: multiple;
  box-lines: multiple;
  -webkit-flex-wrap: wrap;
  -moz-flex-wrap: wrap;
  -ms-flex-wrap: wrap;
  flex-wrap: wrap;
  -webkit-box-pack: justify;
  -moz-box-pack: justify;
  box-pack: justify;
```

```
  -webkit-justify-content: space-between;
  -moz-justify-content: space-between;
  -ms-justify-content: space-between;
  -o-justify-content: space-between;
  justify-content: space-between;
  -ms-flex-pack: justify; }

.card {
  -webkit-flex-basis: 15em;
  -moz-flex-basis: 15em;
  flex-basis: 15em;
  -ms-flex-preferred-size: 15em;
  -webkit-flex-grow: 1;
  -moz-flex-grow: 1;
  flex-grow: 1;
  -ms-flex-positive: 1;
  -webkit-transition: all 0.2s ease-in-out;
  -moz-transition: all 0.2s ease-in-out;
  transition: all 0.2s ease-in-out;
  background-color: #f6f6f6;
  border-radius: 3px;
  border: 1px solid gainsboro;
  box-shadow: 0 2px 4px #e6e6e6;
  margin: 0 1em 1.5em 1em;
  position: relative; }
  .card header {
    -webkit-transition: all 0.2s ease-in-out;
    -moz-transition: all 0.2s ease-in-out;
    transition: all 0.2s ease-in-out;
    background-color: #f6f6f6;
    border-bottom: 1px solid gainsboro;
    border-radius: 3px 3px 0 0;
    font-weight: bold;
    line-height: 1.5em;
    padding: 0.5em 0.75em; }
    .card header a {
      color: #642039;
      text-decoration: none; }
```

```
        .card header a:hover {
          text-decoration: underline; }
    .card .card-copy {
      font-size: 0.9em;
      line-height: 1.5em;
      padding: 0.75em 0.75em; }
      .card .card-copy p {
        margin: 0 0 0.75em; }

  article h2 a,
  article h3 a {
    color: #642039;
    text-decoration: none; }
    article h2 a:hover,
    article h3 a:hover {
      text-decoration: underline; }
  article .steps {
    border-top: 2px solid #e6557e; }
    article .steps h3 + p {
      margin-left: 1em; }
```

Pada file settings.py tambahkan pengaturan untuk direktori assests

```
STATIC_URL = '/static/'
STATICFILES_DIRS = ( #static assets directory for css, images, etc
    os.path.join(BASE_DIR, 'assets'),
)
```

Kemudian pada urls.py tambahkan kode staticfiles berikut:

```
from django.contrib import admin
from django.urls import path,include
from django.contrib.staticfiles.urls import staticfiles_urlpatterns

from . import views

urlpatterns = [
    path('courses/',include('courses.urls')),
    path('admin/', admin.site.urls),
```

```
    path('',views.hello_world),
]

urlpatterns += staticfiles_urlpatterns()
```

Pada folder template file layout.py tambahkan kode berikut ini:

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <title>{% block title %}{% endblock title %}</title>
    <link rel="stylesheet" href="{% static 'css/layout.css' %}">
</head>
<body>
    {% block content %}{% endblock content %}
</body>
</html>
```

Kemudian jalankan aplikasi dan cek bahwa css sudah ditambahkan.

Menambahkan Steps pada Courses

Pada courses tambahkan pada models.py yaitu Step model.

```
class Step(models.Model):
    title = models.CharField(max_length=255)
    description = models.TextField()
    order = models.IntegerField(default=0)
    course = models.ForeignKey(Course,on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```

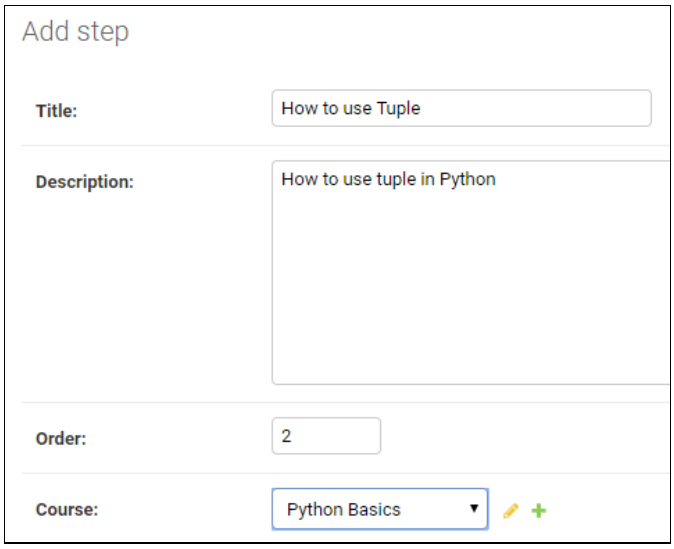
kemudian jalankan perintah migrasi

```
C:\Python\Scripts\learning_site>py manage.py makemigrations courses
Migrations for 'courses':
  courses\migrations\0002_step.py
    - Create model Step
```

```
C:\Python\Scripts\learning_site>py manage.py migrate courses
Operations to perform:
  Apply all migrations: courses
Running migrations:
  Applying courses.0002_step... OK
```

Kemudian pada admin.py tambahkan kode berikut ini:

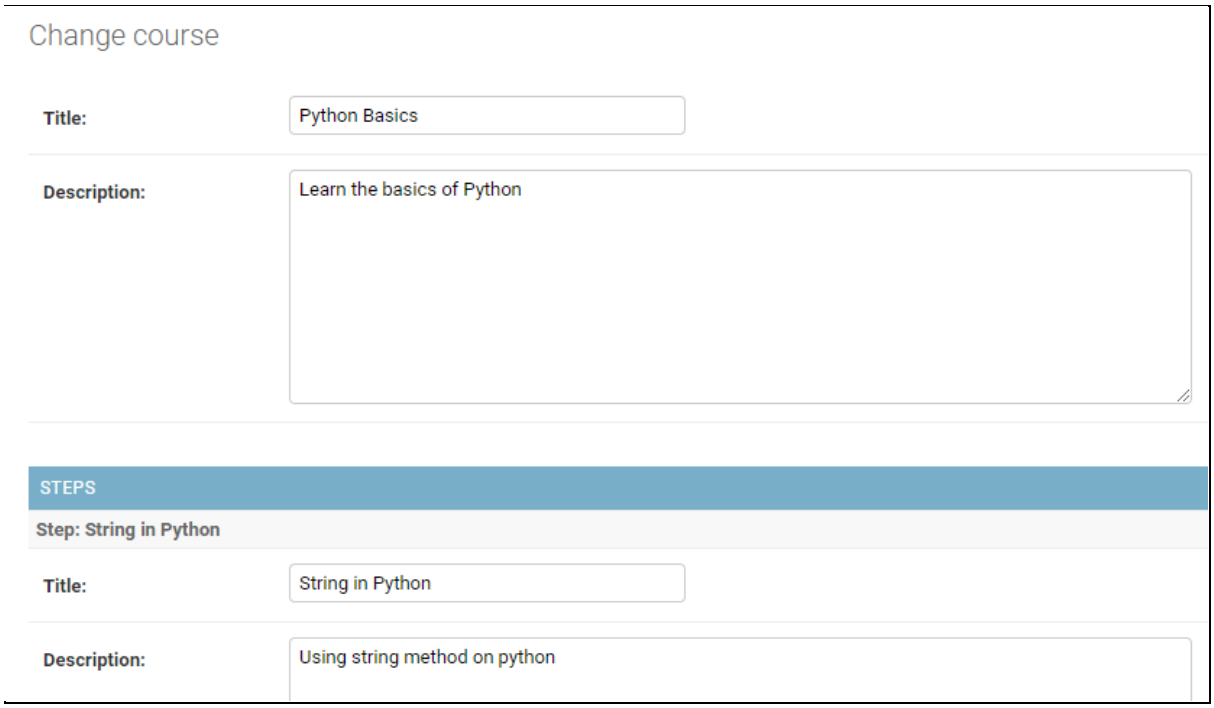
```
from django.contrib import admin
from .models import Course,Step

admin.site.register(Course)
admin.site.register(Step)
```



Untuk menambahkan inline insert Step pada Course, tambahkan kode berikut pada admin.py

```
from django.contrib import admin
from .models import Course,Step

class StepInline(admin.StackedInline):
    model = Step

class CourseAdmin(admin.ModelAdmin):
    inlines = [StepInline,]
```

```
admin.site.register(Course,CourseAdmin)
admin.site.register(Step)
```



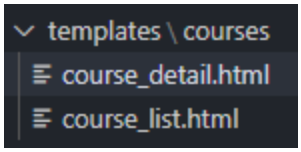
Menambahkan Detail View

Pada Course app file viees.py tambahkan kode berikut untuk menampilkan detail dari courses

```
def course_list(request):
    courses = Course.objects.all()
    return render(request,'courses/course_list.html',{'courses':courses})

def course_detail(request,pk):
    course = Course.objects.get(pk=pk)
    return render(request,'courses/course_detail.html',{'course':course})
```

pada urls.py tambahkan

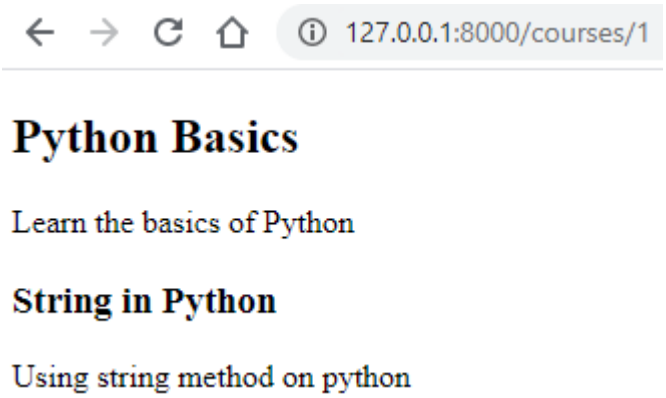
```
urlpatterns = [
    path("",views.course_list),
    path("<int:pk>",views.course_detail)
]
```




```
{% extends 'layout.html' %}

{% block title %} {% endblock title %}

{% block content %}
<article>
    <h2>{{ course.title }}</h2>
    {{ course.description }}
    <section>
        {% for step in course.step_set.all %}
            <h3>{{ step.title }}</h3>
            {{ step.description }}
        {% endfor %}
    </section>
</article>
{% endblock content %}
```

Untuk pengurutan berdasarkan step pada models.py dapat ditambahkan class meta:

```
class Step(models.Model):
---------

    class Meta:
        ordering = ['order',]

    def __str__(self):
        return self.title
```



Menampilkan Step Detail

Tambahkan kode berikut pada file views.py di Courses

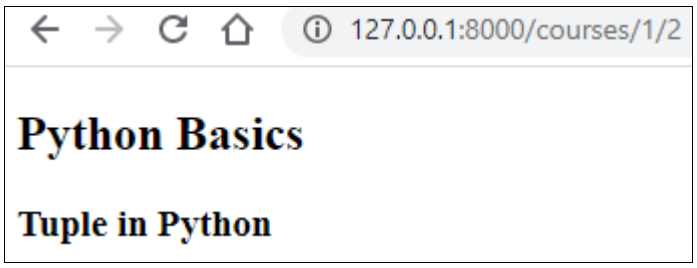
```
def step_detail(request,course_pk,step_pk):
    step = get_object_or_404(Step,course_id=course_pk,pk=step_pk)
    return render(request,'courses/step_detail.html',{'step':step})
```

kemudian tambahkan route pada urls.py

```
urlpatterns = [
    path("",views.course_list),
    path("<int:course_pk>/<int:step_pk>",views.step_detail),
    path("<int:pk>",views.course_detail),
]
```

Kemudian buat file step_detail.html pada folder template.

```
{% extends 'layout.html' %}

{% block title %} {{step.title}} - {{step.course.title }} {% endblock %}

{% block content %}
  <article class="">
    <h2>{{step.course.title}}</h2>
        <h3>{{step.title}}</h3>
        {{ step.content|linebreaks }}
  </article>
{% endblock %}
```



Tambahkan field content kedalam model step

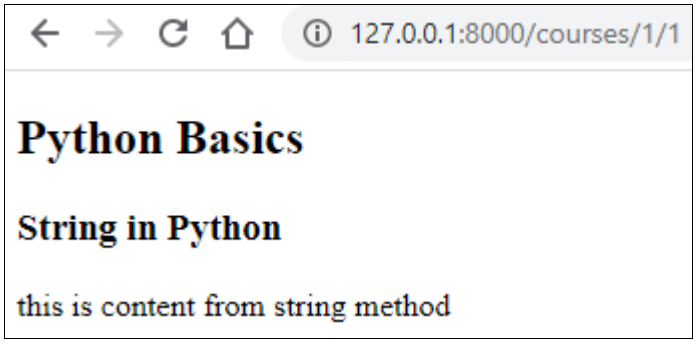
```
class Step(models.Model):
    -----
    content = models.TextField(blank=True,default='')
-------


    def __str__(self):
        return self.title
```

kemudian jalankan perintah migration

```
C:\Python\Scripts\learning_site>py manage.py makemigrations courses
Migrations for 'courses':
  courses\migrations\0003_auto_20191229_2259.py
    - Change Meta options on step
    - Add field content to step
```

```
C:\Python\Scripts\learning_site>py manage.py migrate courses
Operations to perform:
  Apply all migrations: courses
Running migrations:
  Applying courses.0003_auto_20191229_2259... OK
```

Tambahkan content field pada form step.



Tambahkan name pada urls.py untuk memberi nama route yang akan kita tuju

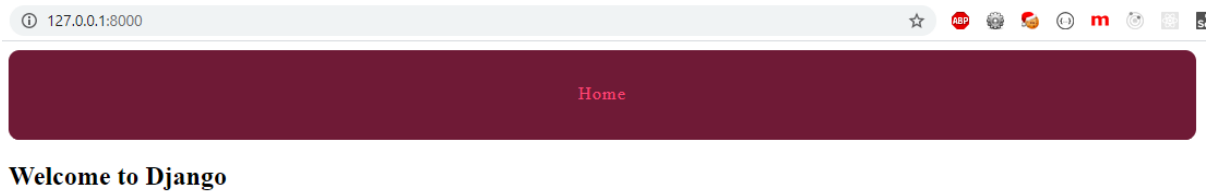
```
urlpatterns = [
    path('',views.hello_world,name='home'),
```

```
]
```

Pada layout.html tambahkan kode berikut untuk navigasi halaman

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <title>{% block title %}{% endblock title %}</title>
    <link rel="stylesheet" href="{% static 'css/layout.css' %}">
</head>
<body>
     <div class="site-container">
      <nav>
        <a href="{% url home' %}">Home</a>
      </nav>
      {% block content %} {% endblock %}
    </div>
</body>
</html>
```

Jika dijalankan tampilannya adalah sebagai berikut:



Welcome to Django

Namespace dan name dapat digunakan untuk membuat navigasi. Dengan menggunakan namespace maka pembuatan navigasi akan lebih mudah.

```
urlpatterns = [
    path('courses/',include('courses.urls',namespace='courses')),
    -------------------------
    path('',views.hello_world,name='home'),
]
```

Kemudian pada courses\urls.py tambahkan kode berikut untuk menambahkan name.

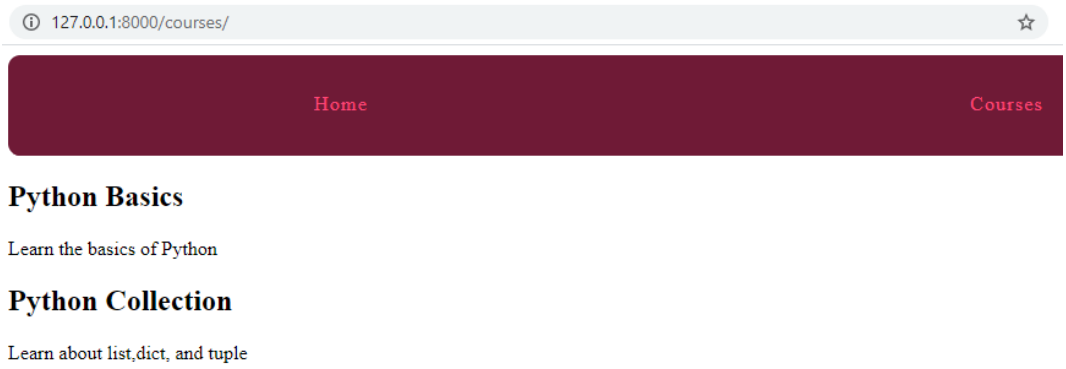
```
app_name = 'courses'
urlpatterns = [
```

```
    path("",views.course_list,name='list'),
    path("<int:course_pk>/<int:step_pk>",views.step_detail,name='step'),
    path("<int:pk>",views.course_detail,name='detail'),
]
```

Jangan lupa untuk menambahkan app_name agar namespace dapat dikenali

Pada layout tambahkan navigasi baru ke courses.

```
      <nav>
        <a href="{% url 'home' %}">Home</a>
        <a href="{% url 'courses:list' %}">Courses</a>
      </nav>
```



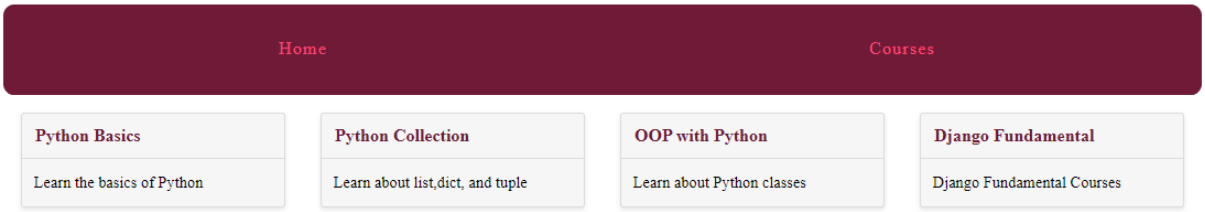
Untuk membuat tampilan menjadi lebih mudah dibaca tambahkan folder static\css, kemudian tambahkan file courses.css

```
.card header a {
    color: #000080;
}
```

Kemudian pada course_list.html tambahkan kode

```
{% extends 'layout.html' %}
{% load static %}

{% block static %}<link rel="stylesheet" href="{% static 'courses/css/cour
ses.css' %}"> {% endblock %}
{% block title %} Available Courses {% endblock %}


{% block content %}
<div class="cards">
```

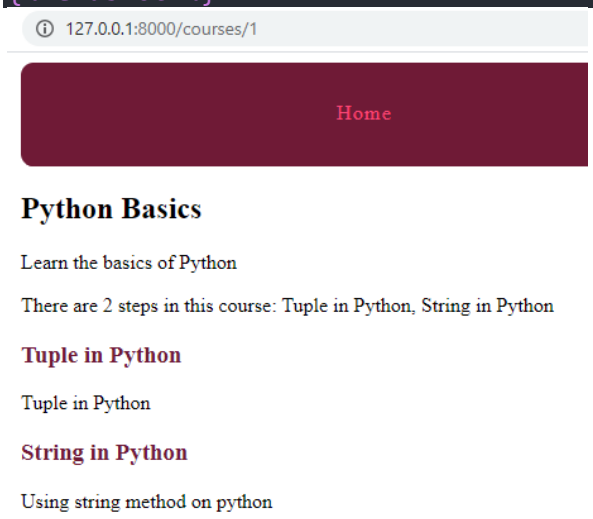
```
  {% for course in courses %}
  <div class="card">
    <header><a href="{% url 'courses:detail' pk=course.pk %}">{{ course.ti
tle}}</a></header>
      <div class="card-copy">
        {{ course.description }}
      </div>
  </div>
  {% endfor %}
</div>
{% endblock %}
```



Kemudian pada course detail tambahkan juga kode berikut ini:

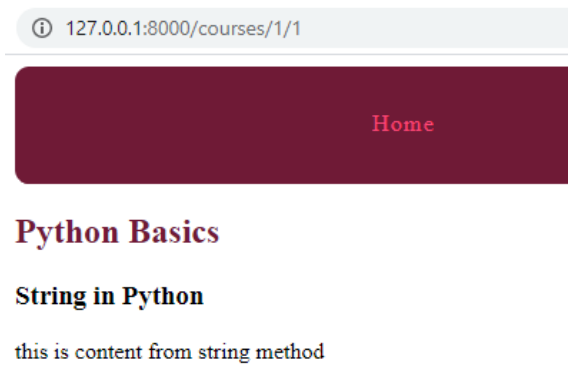
```
{% extends 'layout.html' %}

{% block title %}{{course.title}} {% endblock %}

{% block content%}
<article class="">
  <h2>{{ course.title }}</h2>
  {{ course.description}}

  <p> There are {{ course.step_set.count }} step{{ course.step_set.count|p
luralize }} in this course: {{ course.step_set.all|join:", "}}</p>
  <section>
    {% for step in course.step_set.all %}
      <h3><a href="{% url 'courses:step' course_pk=step.course.pk step_pk=
step.pk %}">{{step.title}}</a></h3>
        {{step.description}}
    {% endfor %}

  </section>
```

```
</article>
{% endblock%}
```



Kemudian pada step tambahkan kode berikut:

```
{% extends 'layout.html' %}

{% block title %} {{step.title}} - {{step.course.title }} {% endblock %}

{% block content %}
  <article class="">
    <h2><a href="{% url 'courses:detail' pk=step.course.pk %}">{{step.cour
se.title}}</a></h2>
        <h3>{{step.title}}</h3>
        {{ step.content|linebreaks }}

  </article>

{% endblock %}
```



RESTful API with Python and Django

Pada Hands On Labs berikut akan dibahas bagaimana cara membuat RESTful API dengan menggunakan bahasa Python dan Django Framework.

Instalasi Awal

Buat project baru untuk django

Install djangorestframework

```
C:\Python\Scripts>pip install djangorestframework
Collecting djangorestframework
  Downloading https://files.pythonhosted.org/packages/be/5b/9bbde4395a
558ca919ffaa2e0068/djangorestframework-3.11.0-py3-none-any.whl (911kB)
     |                                | 614kB 163kB/s eta 0:00:02
```

Buat django project baru dengan nama invoic

```
C:\Python\Scripts>django-admin startproject invoiceproject
```

Kemudian tambahkan app project dengan nama invoices

```
C:\Python\Scripts\invoiceproject>py manage.py startapp invoices
```

Kemudian buka invoices pada visual studio code

Pada setting.py tambahkan kode berikut

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'invoices.apps.InvoicesConfig'
]
```

Pada invoices\model tambahkan model dengan nama Invoice

```
# Create your models here.
class Invoice(models.Model):
    created = models.DateTimeField(auto_now_add=True)
    name = models.CharField(max_length=100)
    description = models.TextField(blank=True,default='')
    total = models.DecimalField(max_digits=7,decimal_places=2)
    paid = models.DecimalField(max_digits=7,decimal_places=2)
```

kemudian jalankan migrasi

```
C:\Python\Scripts\invoiceproject>py manage.py makemigrations invoices
```

Lanjutkan dengan perintah

```
C:\Python\Scripts\invoiceproject>manage.py migrate
```

Kemudian buka shell untuk menambahkan invoice baru

```
C:\Python\Scripts\invoiceproject>py manage.py shell
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from invoices.models import Invoice
>>> Invoice.objects.all()
<QuerySet []>
>>> invoice = Invoice(name='My First Invoice',description='Deskripsi Invoice',total=100.00,paid=0.00)
>>> invoice.name
'My First Invoice'
>>> invoice.total
100.0
>>>
```

Setelah perintah save ditambahkan maka record invoice akan ditambahkan

```
>>> Invoice.objects.all()
<QuerySet []>
>>> invoice.save()
>>> Invoice.objects.all()
<QuerySet [<Invoice: Invoice object (1)>]>
```

Menambahkan Serializer

Pada folder invoices tambahkan file dengan nama serializers.py

```
from rest_framework import serializers
from invoices.models import Invoice

class InvoiceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Invoice
        fields = ('name','description','total','paid')
```

Untuk mencoba menjalankan serializer buka shell kemudian jalankan perintah berikut

```
>>> from invoices.models import Invoice
>>> from invoices.serializers import InvoiceSerializer
>>> invoice = Invoice.objects.get()
>>> invoice.name
'My First Invoice'
>>> serializer = InvoiceSerializer(invoice)
>>> serializer.data
{'name': 'My First Invoice', 'description': 'Deskripsi Invoice', 'total': '100.00', 'paid': '0.00'}
```

Menambahkan Routing Url

Pada urls.py di folder invoiceproject tambahkan kode berikut ini

```
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('',include('invoices.urls')),
    path('admin/', admin.site.urls),
]
```

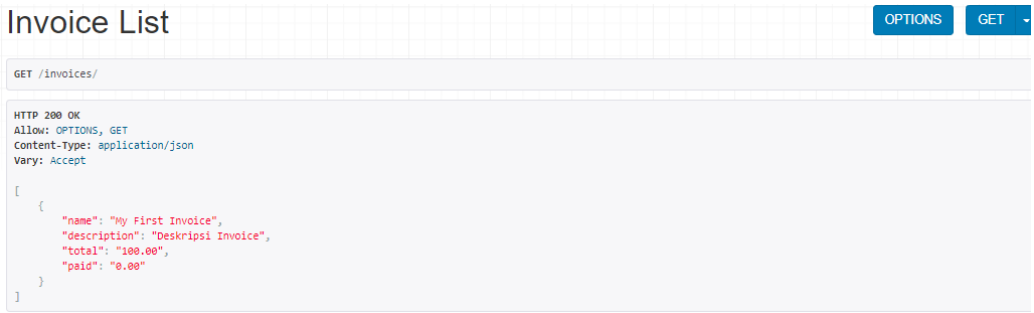
Kemudian tambahkan juga pengaturan urls.py pada folder invoices

```
from django.urls import path
from invoices import views

urlpatterns = [
    path('invoices/',views.invoice_list),
]
```

Pada file views.py di folder invoices tambahkan kode berikut

```
from rest_framework.decorators import api_view
from rest_framework.response import Response
from invoices.models import Invoice
from invoices.serializers import InvoiceSerializer

@api_view(['GET'])
def invoice_list(request):
    if request.method == 'GET':
        invoices = Invoice.objects.all()
        serializer = InvoiceSerializer(invoices,many=True)
        return Response(serializer.data)
```

hasilnya dapat dilihat pada gambar dibawah ini



Untuk menampilkan single record tambahkan field id serializer.py

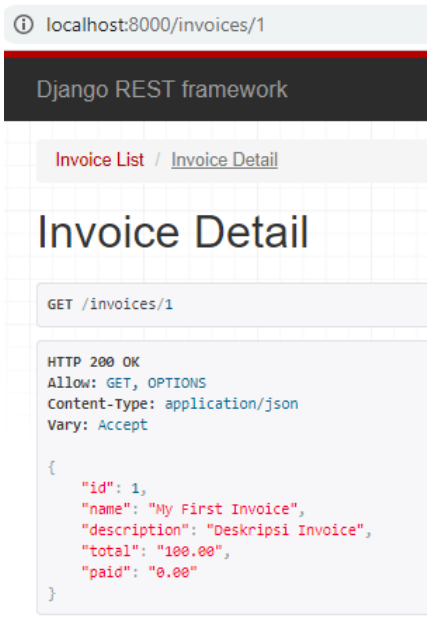
```
from rest_framework import serializers
from invoices.models import Invoice

class InvoiceSerializer(serializers.ModelSerializer):
    class Meta:
        model = Invoice
        fields = ('id','name','description','total','paid')
```

kemudian tambahkan kode untuk mengambil satu data saja pada views.py

```
from rest_framework import status
-----

@api_view(['GET'])
def invoice_detail(request,pk):
    try:
        invoice = Invoice.objects.get(pk=pk)
    except Invoice.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)
    serializer = InvoiceSerializer(invoice);
    return Response(serializer.data)
```

kemudian pada urls.py tambahkan juga kode berikut

```
urlpatterns = [
    path('invoices/',views.invoice_list),
    path('invoices/<int:pk>',views.invoice_detail),
]
```

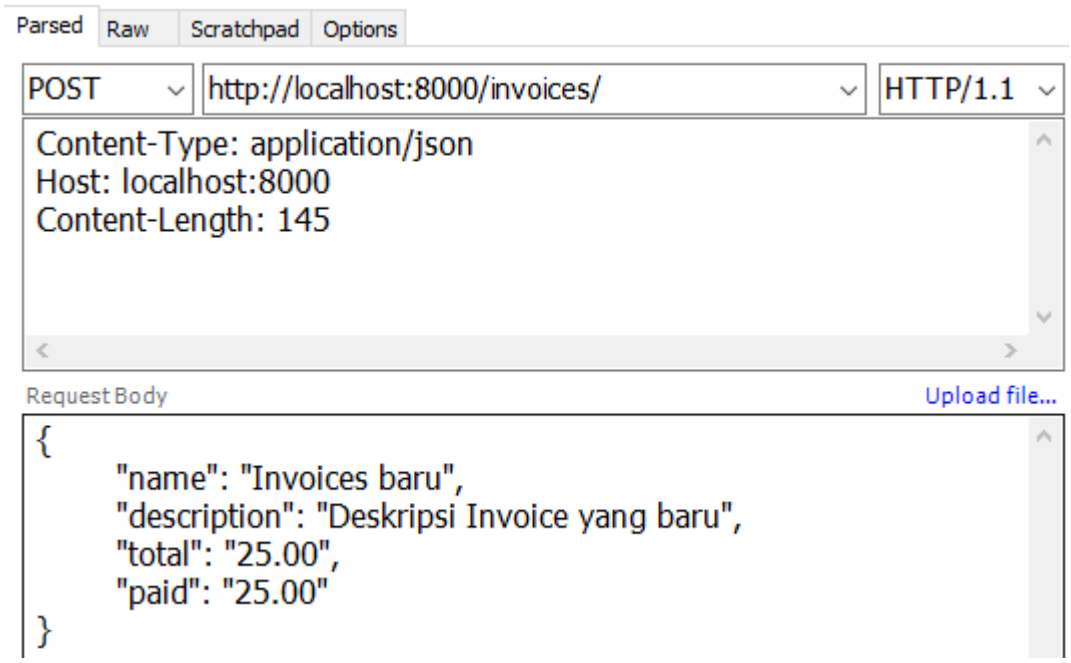


Menambahkan Data Invoice

Untuk menambahkan data dengan method POST tambahkan kode berikut:

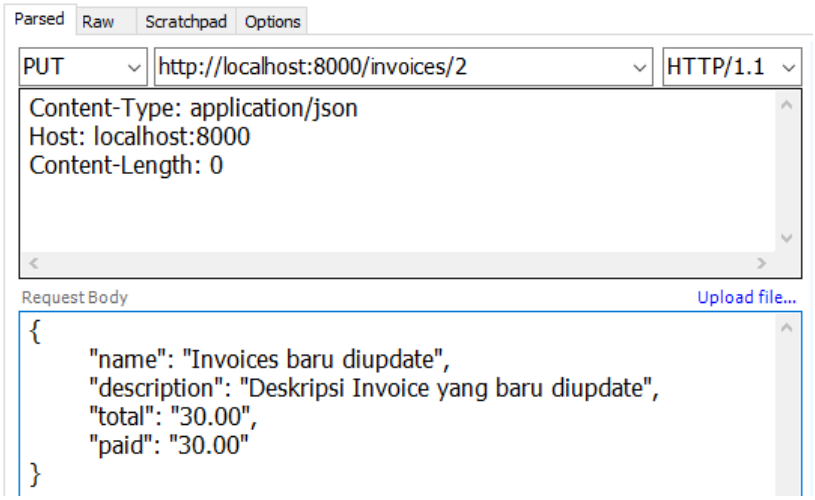
```
@api_view(['GET','POST'])
def invoice_list(request):
    if request.method == 'GET':
        invoices = Invoice.objects.all()
        serializer = InvoiceSerializer(invoices,many=True)
        return Response(serializer.data)
    else request.method=='POST':
        serializer = InvoiceSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data,status=status.HTTP_201_CREATED
)
        return Response(serializer.errors,status=status.HTTP_400_BAD_REQUE
ST)
```

kemudian buka fiddler/postman dan coba untuk menambahkan data sebagai berikut:



Update Data Invoice

```
@api_view(['GET','PUT'])
def invoice_detail(request,pk):
    try:
        invoice = Invoice.objects.get(pk=pk)
    except Invoice.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method=='GET':
        serializer = InvoiceSerializer(invoice)
        return Response(serializer.data)

    if request.method == 'PUT':
        serializer = InvoiceSerializer(invoice,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)

    return Response(serializer.errors,status=status.HTTP_400_BAD_REQUEST)
```



Delate Data Invoices

Untuk delete data invoice tambahkan kode berikut ini

```
@api_view(['GET','PUT','DELETE'])
def invoice_detail(request,pk):
    try:
        invoice = Invoice.objects.get(pk=pk)
    except Invoice.DoesNotExist:
        return Response(status=status.HTTP_404_NOT_FOUND)

    if request.method=='GET':
        serializer = InvoiceSerializer(invoice)
        return Response(serializer.data)

    elif request.method == 'PUT':
        serializer = InvoiceSerializer(invoice,data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors,status=status.HTTP_400_BAD_REQUE
ST)

    elif request.method == 'DELETE':
        invoice.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

Pengaturan Django dengan Database SQL Server

Gunakan django versi 2.2 LTS

pip install django==2.2

kemudian install package berikut:

https://github.com/ESSolutions/django-mssql-backend

jalankan perintah pip install django-mssql-backend