Sisällys

1.Asenna Django

Django asennetaan pip-paketinhallinnalla.

ctrl + ö avaa visual studio codella terminaalin. Vaihtoehtoisesti voit käyttää komentokehoitetta tai Powershelliä. Komento: pip install django.

HUOM. Yllä oleva komento toimii vain jos ympäristömuuttujien path -muuttujassa. Mikäli mielestäsi PATH muuttuja on kunnossa, et saa komentoa toimimaan ja komento pip –version palauttaa virheilmoituksen, käytä komentoa:

py -m pip install django

2. Django-projektin luominen

Aloita luomalla uusi Django-projekti

django-admin startproject tietokanta_projekti Ohessa tietokanta_projekti on projektikansion nimi

Yllä oleva luo kansiorakenteen:

tietokanta_projekti/

manage.py

tietokanta_projekti/

__init__.py

settings.py

urls.py

asgi.py

wsgi.py

3. Käynnistä kehityspalvelin

python manage.py runserver

Tämän käynnistää Djangon sisäisen kehityspalvelimen. Voit tarkistaa selaimessa, että palvelin toimii osoitteesta http://127.0.0.1:8000/. Näet oletuskehityssivun, joka korvautuu heti kun määrität kohdassa 9 omia sisältöjä. Näin ollen tyhjälle polulle ole viittausta ja haku näyttää virheilmoituksen.

3.1 Valinnainen toimenpide.

HUOM. On erittäin suotavaa että toteutat vaiheet 1-12 ensin ja palaat sitten tähän.

Mikäli haluat myöhemmin testata projektin rakennusta vaihe vaiheelta olemassa olevalla projektilla, täytyy testisivulle luoda viittaus.

Lisää tietokanta_projekti/urls.py rivi

path('', views.test_page, name='test_page'),

Lisää tietokanta/views.py rivit

```
from django.shortcuts import render

def test_page(request):
        return render(request, 'tietokanta/test_page.html')
```

Lisää templates/tietokanta kansioon tiedosto test_page.html

Lisää luotuun tiedostoon rivit

```
<html>
<head>
  <title>Testisivu</title>
</head>
<body>
  <h1>Tämä on testisivu</h1>
  <p>Voit käyttää tätä sivua projektin testaukseen.</p>
</body>
</html>
```

Testisivu on saatavilla osoitteessa http://127.0.0.1:8000/

4 Sovelluksen luonti

Luominen tapahtuu komennolla python manage.py startapp ’sovelluksen nimi’ joko komentokehotteella, powershellillä tai projektikansiosta. Oleellista on suorittaa komento siitä kansiosta, jossa manage.py sijaitsee. Tutoriaaliesimerkissämme tietokanta_projekti\manage.py

Tässä tutoriaalissa luodaan python manage.py startapp tietokanta

lisää settings.py tiedostoon, riville 33 kohtaan

```
INSTALLED_APPS = [
    'sovelluksen_nimi'
```

esimerkissämme sovelluksen nimi = tietokanta

5 Määrittele malli (Model)

Mallit (models) määrittelevät tietokantaan tallennettavat tiedot. Esimerkiksi luodaan Triathlon_record-malli, joka edustaa tietokannan tietuetta. Tässä yhteydessä tarvitaan ehdottomasti moduulia django.db jotta päästään djangon tietokanta sovelluksiin käsiksi

Lisää tietokanta/models.py:

```
from django.db import models


class Triathlon_Record(models.Model):
    name = models.CharField(max_length=50)  # Nimi, maksimissaan 50 merkkiä
    swim = models.DurationField()  # Uintiaika
    T1 = models.DurationField()  # Ensimmäinen vaihto
    cycle = models.DurationField()  # Pyöräilyaika
    T2 = models.DurationField()  # Toinen vaihto
    run = models.DurationField()  # Juoksuaika


    @property
    def total(self):
        """Laskee kaikkien aikojen summan"""
        return self.swim + self.T1 + self.cycle + self.T2 + self.run


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

6. Suorita tietokantojen päivitys

python manage.py makemigrations

Tämä komento valmistelee tietokantamuutokset mutta ei vielä lisää muutoksia. Jos esimerkiksi lisäät uuden mallin tai tietokentän olemassa olevaan malliin, tarvitaan tätä komentoa.

python manage.py migrate

Tämä komento soveltaa muutokset tietokantaan. Eli komento varmistaa, että kehityspalvelimella oleva tietokanta vastaa djangon malleissa olevaa rakennetta.

7. Rakennetaan tietueet

Tähän on muutamia vaihtoehtoja kuten djangon komentorivi, django shell tai hallintatyökalu. Käytämme komentorivi eli management vaihtoehtoa.

7.1 luo kansiot ”tietokanta\management\commands”

Tässä voidaan hyödyntää komentokehoitetta tai powershelliä. Komento:

mkdir tietokanta\management\commands.

Kansiot voidaan luoda myös muilla keinoilla.

7.1.1 lisää init.py

Lisää kumpaankin luotuun kansioon tyhjä tiedosto __init__.py

Huom. sovelluksen nimi ei ole välttämättä sama kuin projektikansion, kts tarvittaessa kohta 4.1

7.2 Lisää kohdassa 7.1 luotuun commands -kansioon:

Lisätään kansioon tiedosto, jonka nimestä tulee management komento. Tässä tapauksessa luomme kansion add_triathlon_record. Jokaiselle tietokantataululle ei välttämättä tarvita omaa komentoa, mutta sellainen voidaan tietenkin tarvittaessa rakentaa. Tarkoituksenmukaisin (komennot yhdessä vai erikseen) lähestymistapa riippuu projektin luonteesta.

Lisää commands\add_triathlon_record.py rivit:

```
from django.core.management.base import BaseCommand
from datetime import timedelta
from tietokanta.models  import Triathlon_Record

class Command(BaseCommand):
    help = 'Add multiple triathlon records'

    def handle(self):
        record1 = Triathlon_Record(
            name="Triathleetti Teemu",
            swim=timedelta(hours=0, minutes=55, seconds=15),
            T1=timedelta(minutes=3, seconds=2),
            cycle=timedelta(hours=5, minutes=45, seconds=30),
            T2=timedelta(minutes=2, seconds=15),
            run=timedelta(hours=4, minutes=20, seconds=45)
        )
        record1.save()
        self.stdout.write(self.style.SUCCESS(f'Record 1 added: Total time: {record1.total}'))

        # Tietue 2
        record2 = Triathlon_Record(
            name="Triathlon Tiina",
            swim=timedelta(hours=1, minutes=30, seconds=45),
            T1=timedelta(minutes=2, seconds=50),
            cycle=timedelta(hours=6, minutes=50, seconds=0),
            T2=timedelta(minutes=8, seconds=30),
            run=timedelta(hours=5, minutes=25, seconds=15)
        )
        record2.save()
        self.stdout.write(self.style.SUCCESS(f'Record 2 added: Total time: {record2.total}'))
```

7.3 Valmis management komento

Nyt meillä on valmis komento, joka voidaan ajaa muodossa python manage.py add_triathlon_record kansiosta jossa manage.py sijaitsee

8. Määritä sovelluksen admin-paneeli

Lisää uusi malli admin-paneeliin, jotta sitä voidaan hallinnoida selaimen kautta. Admin tiedosto sijaitsee: tietokanta/admin.py

Esimerkissämme mallin_nimi = Triathlon_Record, jonka loimme kohdassa 5.

from django.contrib import admin

from .models import **mallin_nimi**

admin.site.register(**mallin_nimi**)

8.1 Kirjaudu admin-paneeliin luomalla superuser:

python manage.py createsuperuser

Komento pyytää luomaan superuser käyttäjätunnuksen, määrittämään sähköpostin sekä valitsemaan salasanan. Superuser on käyttäjä, jolla on rajoittamattomat oikeudet ohjelmistoon. **Tietoturvasyistä näitä tulisi jakaa harkiten.** Superuser vastaa pääkäyttäjää.

Mikäli unohdat salasanan, kts yst otsikko huomioita.

Käynnistä palvelin ja mene osoitteeseen http://127.0.0.1:8000/admin/ kirjautuaksesi sisään. Mikäli saat ilmoituksen, ettei sivustoon saada yhteyttä, voit koittaa palvelimen uudelleen käynnistämistä python manage.py runserver komennolla projektikansiossa. Mikäli homma niin sanotusti pelaa, pitäisi näkymän olla seuraavanlainen:

8.2 Admineiden ja käyttäjien luominen

Superuser voi luoda admin -käyttäjät admin-paneelista (http://127.0.0.1:8000/admin/).

1. Kirjaudu sisään admin-paneeliin
2. Valitse "Users" -valikosta ja klikkaa "Add user" (lisää käyttäjä).
3. Luo uusi käyttäjä ja täytä tarvittavat tiedot (käyttäjätunnus, salasana jne.).
4. Tallenna käyttäjä.
5. Kun käyttäjä on luotu, voit antaa hänelle admin-oikeudet valitsemalla hänelle:
    - Staff status: Käyttäjä, jolla on staff-oikeudet, voi kirjautua admin-paneeliin.

8.2.1 Admin-oikeuksien rajaaminen:

Djangossa voit määrittää käyttöoikeuksia yksittäisille malleille ja toiminnoille. Näitä oikeuksia ovat muun muassa:

-Add: Käyttäjä voi lisätä uusia tietokantaobjekteja.

-Change: Käyttäjä voi muokata olemassa olevia tietokantaobjekteja.

-Delete: Käyttäjä voi poistaa tietokantaobjekteja.

-View: Käyttäjä voi tarkastella tietokantaobjekteja, mutta ei muokata niitä.

Oikeuksia voi määrittää seuraavasti:

1. Mene admin-paneeliin ja valitse users
2. Valitse käyttäjä, jolle haluat määrittää oikeudet.
3. Vieritä alas kohtaan User permissions ja valitse harkiten mitä oikeuksia haluat antaa käyttäjälle. Oikeudet voidaan määrittää mallikohtaisesti.
4. Tallenna muutokset.

8.2.2 Käyttöoikeuksien hallinta ryhmien avulla:

Voit luoda ryhmiä, joilla on tietyt käyttöoikeudet, ja lisätä käyttäjiä näihin ryhmiin. Näin voidaan luoda esimerkiksi admins ryhmä.

1. Mene admin-paneeliin ja valitse Groups.
2. Luo uusi ryhmä ja määritä sille oikeudet (kuten lisääminen, muokkaaminen jne.).
3. Lisää käyttäjiä tähän ryhmään, jolloin he perivät ryhmän käyttöoikeudet.

9. Luo näkymä (View)

Näkymä käsittelee käyttäjän pyynnöt ja palauttaa vastauksen, esimerkiksi verkkosivun. Luo yksinkertainen näkymä, joka näyttää kaikki tietokannan tietueet. Esimerkissämme **mallin_nimi** =Triathlon_Records. Huomaa, että funktiossa record_list(request) viitataan kohdassa 10 luotavaan html malliin. Näiden tulee täsmätä.

Lisää tietokanta/views.py rivit

```
from django.shortcuts import render

from .models import mallin_nimi


def record_list(request):
    records = mallin_nimi.objects.all()
    return render(request, 'tietokanta/Triathlon_record_list.html', {'records': records})
```

10. Yhdistä luotu näkymä URL -osoitteeseen

Lisää tietokanta/urls.py rivit

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

urlpatterns = [
    path('', views.record_list, name='record_list'),
]
```

Lisää tietokanta_projekti/urls.py

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('records/', views.Triathlon_Record_list, name='record_list'),
]
```

11. Luo HTML-malli

Luo polku tietokanta/templates/triathlon_record_list.html

Huomaa, että .html tiedoston nimen tarvitsee täsmätä kohdassa 8 luotuun .html tiedostonimeen.

Lisää templates/triathlon_record_list.html rivit:

```
<html>
<head>
  <title>Tietokanta</title>
</head>
<body>
  <h1>Tietokannan tietueet</h1>
  <ul>
    {% for record in records %}
      <li>{{ record.name }} - {{ record.value }} ({{ record.created_at }})</li>
    {% endfor %}
  </ul>
</body>
</html>
```

HTML-sivu näyttää otsikon "Tietokannan tietueet" ja listaa kaikki tietokannassa olevat tietueet. Tietueiden tiedot (**name**, **value** ja **created_at**) tulevat suoraan näkymän kautta **records**-muuttujasta, jonka **for**-looppi käy läpi.

12. Tarkista sovellus

python manage.py runserver

Mene selaimessa osoitteeseen http://127.0.0.1:8000/triathlon_records/ nähdäksesi tietokannan tietueet

Kohdassa 3 on valinnainen toimenpide, jolla voit luoda testisivun näkyviin osoitteeseen http://127.0.0.1:8000/

13 Huomioita

13.1 Projektikansion nimen muuttaminen

Voit halutessasi muuttaa projektikansion nimen millä tahansa kansion nimeä muuttavalla toimenpiteellä. Tämän lisäksi täytyy huolehtia seuraavista tiedostopäivityksistä. Nimiä muuttaessa korvaustoiminto on näppärä. Visual Studio Codella ctrl + h

-manage.py rivillä: os.environ.setdefault('DJANGO_SETTINGS_MODULE', '**vanha_nimi**.settings')

-> os.environ.setdefault('DJANGO_SETTINGS_MODULE', '**uusi_nimi**.settings')

-asgi.py rivillä: os.environ.setdefault('DJANGO_SETTINGS_MODULE', '**vanha_nimi**.settings')

-> os.environ.setdefault('DJANGO_SETTINGS_MODULE', '**uusi_nimi**.settings')

-settings.py rivillä: WSGI_APPLICATION = '**vanha_nimi**.wsgi.application'

-> WSGI_APPLICATION = '**uusi_nimi**.wsgi.application'

-wsgi.py rivillä: os.environ.setdefault('DJANGO_SETTINGS_MODULE', ' **vanha_nimi**.settings')

os.environ.setdefault('DJANGO_SETTINGS_MODULE', ' **uusi_nimi**.settings')

Yllämainitut ovat kriittisiä muutoksia. Lisäksi tiedostoissa asgi, wsgi, settings ja urls.py on docstring joissa on vanha nimi. Näiden muuttaminen ei ole kriittistä ohjelman toiminnan kannalta.

13.2 Superuser salasanan vaihto

1. Avaa projektikansio (se kansio jossa manage.py sijaitsee) esimekiksi komentokehotteella tai powershellillä.
2. Suorita komento python manage.py shell
3. Suorita alla oleva, jossa super_user = haluttu käyttäjä ja uusi_salasana = uusi salasana.

```
from django.contrib.auth.models

import User

user = User.objects.get(username='super_user')

 user.set_password('uusi_salasana') user.save()
```

13.3 Superuser salasanan vaihto kun käyttäjätunnus ei ole tiedossa

1. Avaa projektikansio (se kansio jossa manage.py sijaitsee) esimekiksi komentokehotteella tai powershellillä.
2. Suorita komento python manage.py shell.
3. Suorita alla oleva

```
from django.contrib.auth.models import User

superusers = User.objects.filter(is_superuser=True)

for superuser in superusers:

    print(superuser.username)
```