

Harjoitustyön vaihe 2: Esimerkkipalvelu

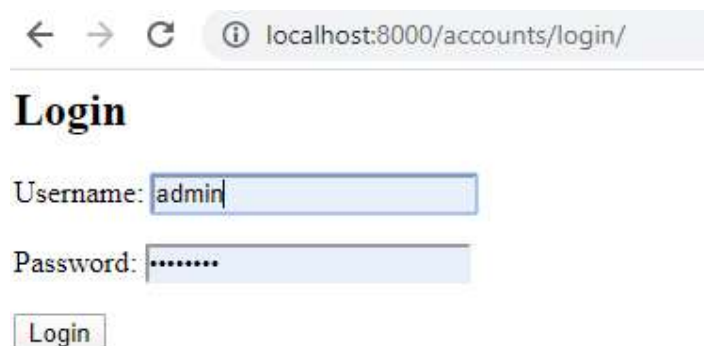
Toisessa vaiheessa toteutetaan tuki kirjautumiseen ja rekisteröitymiseen. Lisäksi lisätään yksinkertaiset CRUD, eli create, read, update ja delete -toiminnot. Käytetyt teknologiat ovat samat, jotka esiteltiin vaiheessa 1.

Kirjautuminen ja rekisteröinti

Kirjautumisen luominen onnistui melko kevyesti Django dokumentaatiota apuna käyttäen. Djangossa oli valmiiksi asennetuissa apeissa auth -app, joka tarjoaa valmiiksi kirjautumiseen vaadittavat [URL:t](#). Näille piti siis enää luoda omat templatet. Template-kansio luotiin project1 -kansion rinnalle ja sen sisälle luotiin registration- kansio, jonka sisään login.html, joka näyttää tältä:

```
templates > registration > login.html > form > button
1  {% extends 'base.html' %}
2
3  {% block title %}Login{% endblock %}
4
5  {% block content %}
6      <h2>Login</h2>
7      <form method="post">
8          {% csrf_token %}
9          {{ form.as_p }}
10         <button type="submit">Login</button>
11     </form>
12  {% endblock %}
```

Tämän jälkeen muokattiin project1 kansion settings.py tiedostoa, jossa TEMPLATES -kohtaan lisättiin "os.path.join(BASE_DIR, 'templates')". Lisäksi settings.py tiedostoon lisättiin loginin jälkeen tapahtuva uudelleenohjaus. Nyt login toimi ja se näytti tältä.



← → ↻ ⓘ localhost:8000/accounts/login/

Login

Username:

Password:

Tässä kohtaa luotiin vielä superuser komennolla **python manage.py createsuperuser**, jotta pystyttiin tarkastelemaan sivuille luotuja käyttäjiä ja testaamaan rekisteröinnin toimivuutta.

Seuraavaksi luotiin kotisivusto, josta selviää, onko käyttäjä kirjautunut vai ei. Lisäksi tältä sivulta voi kirjautua sisään tai rekisteröityä. Kotisivun lisääminen tapahtui lisäämällä templates kansioon tiedostot base.html ja home.html, jotka näyttävät tältä:

```
templates > <> base.html > html
1  <html>
2  <head>
3      <meta charset="utf-8">
4      <title>{% block title %}Django Auth Tutorial{% endblock %}</title>
5  </head>
6  <body>
7      <main>
8          {% block content %}
9          {% endblock %}
10     </main>
11 </body>
12 </html>
```

```
templates > <> home.html > a
1  {% extends 'base.html' %}
2
3  {% block title %}Home{% endblock %}
4
5  {% block content %}
6  {% if user.is_authenticated %}
7      Hi {{ user.username }}!
8      <p><a href="{% url 'list_cars' %}">Selaa autoja</a></p>
9      <p><a href="{% url 'logout' %}">Logout</a></p>
10 {% else %}
11     <p>You are not logged in</p>
12     <a href="{% url 'login' %}">Login</a>
13     <div> </div>
14     <p> No account? </p><a href="{% url 'signup' %}">Register here</a></p>
15 {% endif %}
16
17
18 {% endblock %}
```

Näiden tiedostojen luomisen jälkeen lisättiin vielä project1/urls.py tiedostoon polku `path("", TemplateView.as_view(template_name='home.html'), name='home')`, jotta kotisivu voitiin tuoda näkyviin.



You are not logged in

[Login](#)

No account?

[Register here](#)

Rekisteröinnin luominen lähti siitä, että luotiin uusi appi nimeltä **accounts** komennolla **python manage.py startapp accounts**. Project1 -kansion url polkuihin lisättiin nyt `path('accounts/', include('accounts.urls'))`. Accounts kansion sisälle luotiin `urls.py` -tiedosto, joka näytti lopulta tältä:

```
from django.urls import path
from . import views
urlpatterns = [
    path('signup/', views.SignUp.as_view(), name='signup'),
]
```

`views.py` -tiedostoa muokattiin seuraavanlaiseksi:

```
from django.contrib.auth.forms import UserCreationForm
from django.urls import reverse_lazy
from django.views import generic
class SignUp(generic.CreateView):
    form_class = UserCreationForm
    success_url = reverse_lazy('login')
    template_name = 'signup.html'
```

Lopuksi luotiin vielä `templates` -kansioon `signup.html`, joka näytti tältä:

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

{% block title %}Sign Up{% endblock %}
{% block content %}
    <h2>Sign up</h2>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Sign up</button>
    </form>
{% endblock %}
```

Lopputuloksena saatiin seuraavanlainen rekisteröitymissivusto.

← → ↻ ⓘ localhost:8000/accounts/signup/

Sign up

Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Password:

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation: Enter the same password as before, for verification.

CRUD- toimintojen lisääminen

CRUD- toimintojen lisäämistä varten tehtiin jo harjoitustyön ensimmäisessä vaiheessa luotuun app - kansioon muutoksia. Esimerkkinä käytän autojen tietojen lisäämistä, sillä lopullinen aihe ei ole vielä tarkkaan selvillä.

Aluksi models.py tiedostoon luotiin luokka.

```
class Car(models.Model):
    merkki = models.CharField(max_length=100)
    hinta = models.IntegerField()

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

Tämän jälkeen lisättiin views.py -tiedostoon jokaiselle toiminnolle omat funktionsa.

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from .models import Car
from .forms import CarForm

def list_cars(request):
    cars = Car.objects.all()
    return render(request, 'cars.html', {'cars': cars})

def create_car(request):
    form = CarForm(request.POST or None)
    if form.is_valid():
        form.save()
        return redirect('list_cars')
```

```
        return render(request, 'cars-form.html', {'form': form})

def update_car(request, id=None):
    pass
    car = Car.objects.get(id=id)
    form = CarForm(request.POST or None, instance=car)
    if form.is_valid():
        form.save()
        return redirect(list_cars)
    return render(request, 'cars-form.html', {'form': form, 'car': car})

def delete_car(request, id=None):
    pass
    car = Car.objects.get(id=id)

    if request.method == 'POST':
        car.delete()
        return redirect('list_cars')
    return render(request, 'car-delete-confirm.html', {'car': car})
```

Funktiot lisättiin myös urls.py -tiedostoon

```
from django.urls import path
from .views import list_cars, create_car, update_car, delete_car

urlpatterns = [
    path('list', list_cars, name='list_cars'),
    path('new', create_car, name='create_cars'),
    path('update/<int:id>/', update_car, name='update_car'),
    path('delete/<int:id>/', delete_car, name='delete_car'),
]
```

Tämän jälkeen forms.py -tiedostoon lisättiin luokka uusien autojen lisäämistä varten.

```
from django import forms
from .models import Car

class CarForm(forms.ModelForm):
    class Meta:
        model = Car
        fields = ['merkki', 'hinta']
```

Lopuksi luotiin vielä car-delete-confirm.html, cars-form.html ja cars.html -tiedostot app -kansion alla olevaan templates -kansioon. Näiden sisältö selviää git-repositoriostani.

← → ↻ ⓘ localhost:8000/app/list

- [BMW 116](#)
- [Volvo S80](#)
- [AUDI](#)
- [Pökötti 206 1.01](#)

[New car](#)

← → ↻ ⓘ localhost:8000/app/update/7/

Update car

Merkki: Hinta:
[Delete](#)

← → ↻ ⓘ localhost:8000/app/new

Update car

Merkki: Hinta:

Helppoja asioita

- Login ja logout toimintojen tekeminen oli helppoa hyvän dokumentaation ansiosta.
- Djangoissa on valmiiksi paljon toiminnallisuuksia, jokaista asiaa (esim. auth), ei tarvitse luoda itse.
- Kokonaisuuden käsittäminen projektista onnistui hyvällä tasolla.

Haasteita

- Omien virheiden löytäminen koodista.
- Kun on valmiiksi paljon toiminnallisuuksia, on vaikea välillä hahmottaa, mitä itse on tehnyt väärin.
- Templates -kansion toiminta ja sen ymmärtäminen.

Työssä hyödynnetyt lähteet:

<https://learndjango.com/tutorials/django-login-and-logout-tutorial>

<https://learndjango.com/tutorials/django-signup-tutorial>

https://www.youtube.com/watch?v=Kf9KB_TZY5U

Linkki Gittiin:

<https://github.com/matiasee77/OHSIHA>