

TLO-32400 Ohjelmallinen sisällönhallinta

Harjoitustyön vaihe 4

Tommi Honkanen

tommi.honkanen@tuni.fi

Käytetyt teknologiat:

Web-framework: Django

Tietokanta: Sqlite3

Ohjelmointikielet: Python, JS

Editori: MS Visual Studio

Datan visualisointi

Sovellukseen aikaisemmin ladattu data visualisoidaan D3.js-kirjaston avulla. Palvelun kotisivulla on yksinkertainen HTML-form, johon käyttäjä syöttää haluamansa liigan sekä haluamansa esineen. Alla on kuva HTML-sivusta ja sivun lähdekoodista.

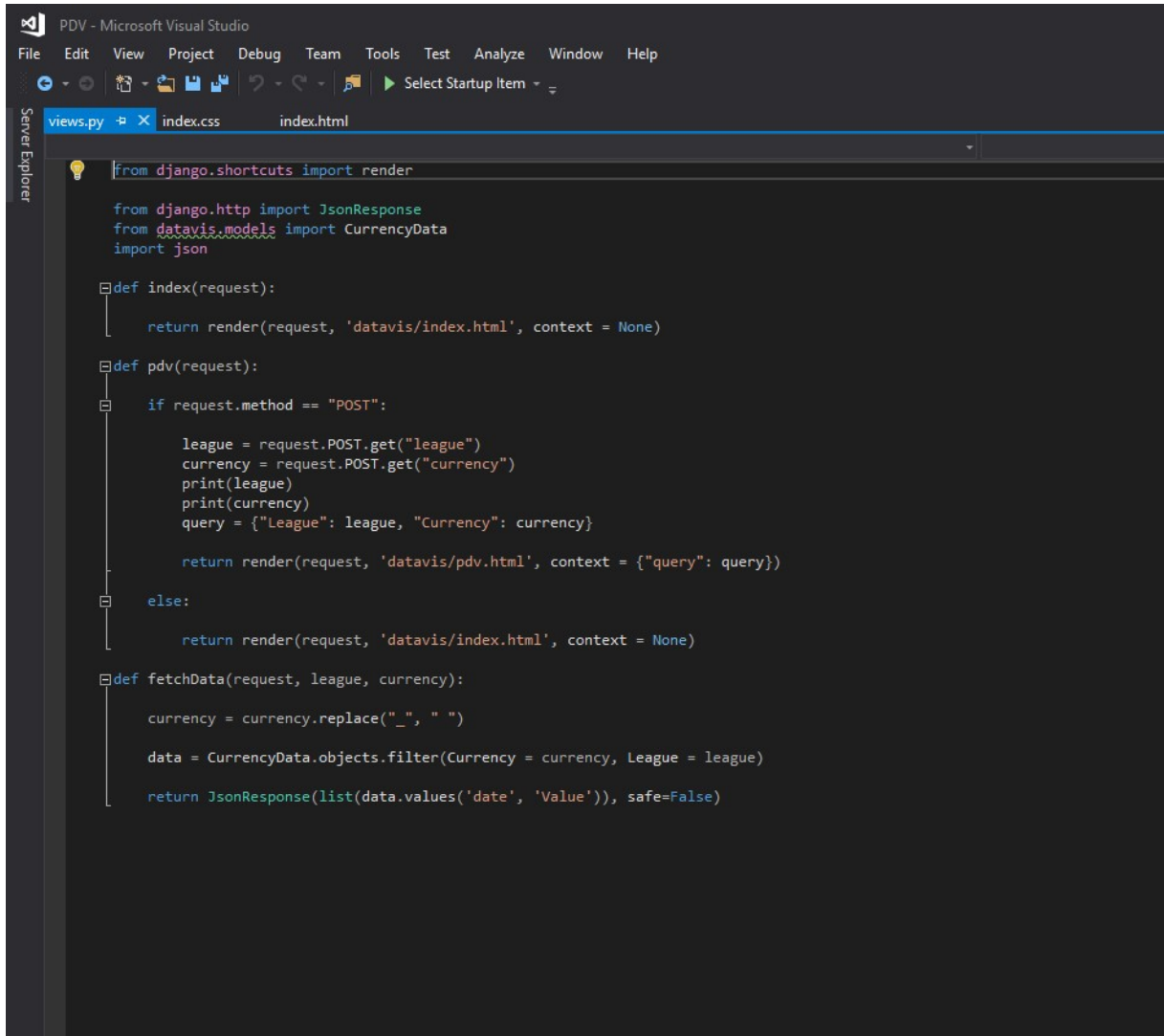
Welcome to PoE price data

Select league: Select currency:

```
PDV - Microsoft Visual Studio
File Edit View Project Debug Team Tools Test Analyze Window Help
Select Startup Item...
Server Explorer
index.css index.html
22
23
24 <div id="page-container">
25   <!--
26   <div id="image">
27     
28   </div>
29   -->
30   <div id="header">
31     <h1>Welcome to PoE price data</h1>
32   </div>
33
34   <div id="content-wrap">
35
36     <form class="form-inline" action="{% url 'pdv' %}" method="post">
37       <div class="form-group">
38         <label for="league">Select league:</label>
39         <select class="form-control" name="league" id="league">
40           <option>Abyss</option>
41           <option>Bestiary</option>
42           <option>Incursion</option>
43           <option>Delve</option>
44           <option>Betrayal</option>
45         </select>
46       </div>
47       <div class="form-group">
48         <label for="currency">Select currency:</label>
49         <select class="form-control" name="currency" id="currency">
50           <option>Exalted Orb</option>
51           <option>Orb of Annulment</option>
52           <option>Orb of Transmutation</option>
53           <option>Orb of Chance</option>
54           <option>Silver Coin</option>
55           <option>Orb of Alchemy</option>
56           <option>Chromatic Orb</option>
57           <option>Orb of Regret</option>
58           <option>Orb of Scouring</option>
59           <option>Orb of Alteration</option>
60           <option>Blacksmith's Whetstone</option>
61           <option>Orb of Augmentation</option>
62         </select>
63       </div>
64       <button type="submit" class="btn btn-default">Fetch price data</button>
65     </form>
66
67   </div>
68
```

Form suorittaa POST-operaation serverille tiettyyn Django näkymään, jonka kautta kyselyn parametrit viedään Django template-contextin avulla toiselle sivulle. Tällä sivulla D3.js-kirjasto tekee parametreilla kutsun yhteen palvelun näkymään, josta palvelun tietokantaan tehdään kysely annetuilla parametreilla. Alla olevassa kuvassa fetchData-näkymä kuvaa tarvittavaa rajapintaa D3.js-kirjastolle, mutta tätä rajapintaa voi aivan hyvin hyödyntää kuka tahansa

kyseistä dataa JSON-muodossa tarvitseva käyttämällä polkua `pricedata/league/currency`, jossa `league` ja `currency` ovat hakuparametreja.



```
from django.shortcuts import render

from django.http import JsonResponse
from datavis.models import CurrencyData
import json

def index(request):
    return render(request, 'datavis/index.html', context = None)

def pdv(request):
    if request.method == "POST":
        league = request.POST.get("league")
        currency = request.POST.get("currency")
        print(league)
        print(currency)
        query = {"League": league, "Currency": currency}

        return render(request, 'datavis/pdv.html', context = {"query": query})
    else:
        return render(request, 'datavis/index.html', context = None)

def fetchData(request, league, currency):
    currency = currency.replace("_", " ")

    data = CurrencyData.objects.filter(Currency = currency, League = league)

    return JsonResponse(list(data.values('date', 'Value')), safe=False)
```

Kysely palauttaa kannasta kyseisen esineen kurssin halutun liigan aikana, ja tämä data palautetaan JSON-pakettina, jota D3.js-kirjasto pystyy hyödyntämään. Data muotoillaan oikeaksi ja siitä piirretään kurssikaavio sivulle.



Tulevat muutokset/lisäykset

Toistaiseksi visualisoinnin aika-akseli on suhteellisen tiheä, ja tarkkoja hetkiä on vaikeaa erottaa käyrältä. Tämän voi korjata lisäämällä kaavioon D3.js-kirjaston erillisen interaktiivisen cursorin, jolla käyrällä voi liikkua tarkasti hiiren avulla ja datan tarkat arvot voi esittää kaaviossa tekstinä.

Palvelu on myös tarkoitus siirtää Herokuun, jossa kuka tahansa voi katsoa visualisointeja sekä hyödyntää palveluun luotua JSON-rajapintaa datalle. Palvelua voisi myös laajentaa luomalla käyttäjille omat tilit ja antamalla heille mahdollisuuden pitää yllä omaa tämänhetkistä ”tiliään” esineistä ja laskemalla näille historiadataan avulla arvion tilin arvosta nykyisessä liigassa.

Helpot ja vaikeat asiat

- Helppoa: Palvelun sivujen ja komponenttien luominen ja hallinta. Djangon näkymien liittäminen eri sivuihin ja liikenteen ohjaaminen sivujen välillä on yksinkertaista ja tehokasta, sekä laajasti dokumentoitua ja aiheesta löytyy paljon esimerkkejä.
- Vaikeaa: D3.js-kirjaston komponenttien asettaminen sekä datan liittäminen niihin. D3.js on kirjastona melko monimutkainen, sillä komponentit rakennetaan täysin alusta asti, jolloin käsiteltäviä parametreja ja funktioita on paljon. Toisaalta D3:n etu on dynaaminen grafiikan luonti pelkän datan avulla, ja D3 lukee dataa verkon yli rajapinnasta, mikä pakottaa kirjaston käyttäjän luomaan toimivan rajapinnan datalle tai käyttämään ulkopuolista sellaista, edesauttaen järkevää palvelumuotoilua.
- Vaikeaa: Palvelun sivujen muotoilu nopeasti ulkomuistista. Käytettävän D3.js-kirjaston elementtien ja muiden HTML-elementtien muotoilu yhteen siistiksi kokonaisuudeksi vaati odotettua enemmän aikaa ja CSS:n kertaamista.

Hyödylliset linkit

D3.js-kirjastoesimerkki:

<https://medium.freecodecamp.org/how-to-build-historical-price-charts-with-d3-js-72214aaf6ba3>

Satunnaiset ohjeet muotoiluun ja HTML-elementteihin <https://www.w3schools.com>