Ohjelmallinen sisällönhallinta

Aleksi Tuominen

3. vaihe: Datarajapinnan käyttöönotto

Harjoitustyössä on tarkoituksena hyödyntää TKL:n avointa dataa. Alun perin suunnitelmissa oli hakea bussilinjoja ja tarkastella, ovatko ne myöhässä tai edellä aikataulusta. Useamman bussilinjan tiedoista olisi näin saanut mielenkiintoista vertailudataa ja niiden visuaalinen mallintaminenkin olisi ollut järkevää. Tutustuttuani tarkemmin TKL:n avoimen datan rajapintaan totesin kuitenkin, ettei tätä ole helppo/järkevä toteuttaa tässä harjoitustyössä. Päädyin siis toteuttamaan jotakin hieman helpompaa: listauksia bussipysäkeistä.

2.3 Stops in area

Tampere API can be used to search for stops within a square with a given diameter. Given point is the center of the square. Square's side is the same as the length of the given diameter in meters..Alternatively it is also used to search for stops within a bbox.

Parameter	Description	Type
center_coordinate	Coordinate	Coordinates separated by comma (e.g. <x,y>)
bbox	Box coordinates	Coordinates separated by comma (e.g. <minx,minY,maxX,maxY>)
limit	Limit the amount of stops.	Optional, default 20.
diameter	Length of the side of the square.	Optional, default 1500, max 5000 meters.

Pysäkkilistaus TKL-APIssa

Aiemmin toteutettua Django-palvelinta voidaan käyttää tässäkin vaiheessa. Lisätään vain urls.py-tiedostoon uusi hakemisto

```
path('stops/', stops, name='stops'),
```

Kirjoitetaan views.py-tiedostoon funktio, joka hakee datan rajapinnasta XML-formaatissa ja tallentaa sen Python-sanakirjana. URL on tässä vaiheessa staattinen ja kysely tuottaa aina saman lopputuloksen. Hakuparametrit lisätään myöhemmin.

```
file = urllib.request.urlopen('http://api.publictransport.tampere.fi/prod/?request=stops_area&user=
data = file.read()
file.close()

data = xmltodict.parse(data)
return render_to_response('stops.html', {'data' : data})
```

Luodaan tiedosto stops.html ja tulostetaan rajapinnasta haettu data jinjaa ja HTML:ää käyttäen.

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

{% block content %}

<ul>

{% with pysakit=data.response.node %}

    {% for pysakki in pysakit %}

        <li>{{pysakki.name}}</li>

    {% endfor %}

{% endwith %}

</ul>

{% endblock %}
```

Nyt localhost:8000/stops/ näyttää seuraavalta:

- Aakkula
- Aakkula
- Messukylänkatu
- Kivikirkko
- Kivikirkko
- Messukylänkatu
- Monttilanpolku
- Monttilanpolku
- Vuohenoja
- Janka
- Alasniitynkatu
- Vuohenoja
- Messukylän kirkko
- Jankanraitti
- Kyläojankatu
- Messukylän kirkko
- Palvaanniemi
- Palvaanniemi
- Kyläojankatu
- Vuohensilta

Datarajapinta on nyt otettu käyttöön. Seuraavaan vaiheeseen toteutetaan interaktiivisuutta rajapintaan ja visualisaatioita datalle.

Rajapinnan käyttöönotto oli helppoa tähän asti. Luultavasti interaktiivisuuden tarjoaminen käyttäjälle tuottaa enemmän haasteita. Vaikeinta tässä kohtaa on Jinjan syntaksi, joka ei todellisuudessa ole kovin Pythonmaista. Pienet erot syntaksissa aiheuttavat harmillisia huolimattomuusvirheitä.