Lauri Larjo 63433N
Joona Olkkola 66708W
T-106.3105 Ohjelmoinnin harjoitustyö
Dokumentaatio

Veji Multimedia Library – http://veji.appspot.com/

Ohjelman kuvaus

Ohjelmamme nimeltä Veji Multimedia Library on oma aiheemme, joka vastannee vähintään laajuudeltaan muita kurssin harjoitustyöaiheita. Se on nettipohjainen sovellus, jolla voi ylläpitää (tällä hetkellä) omaa elokuvakirjastoaan. Ohjelma toimii Googlen sovelluspalvelimella ja hyödyntää Pythonin lisäksi mm. seuraavia teknologioita: HTML, XML, Javascript, CSS. Google App Engine on Googlen kehittämä avoin Internet-sovellusympäristö, joka tukee Pythonia.

Toteutustavan ansiosta ohjelma on käytettävissä mistä päin maailmaa tahansa, kunhan käyttäjällä on vain mahdollisuus päästä Internetiin. Vejin on tarkoitus toimia käyttäjilleen ensisijaisesti elokuvatietokantana, jonne voi lisätä kaikki ne elokuvat, jotka omistaa tai on joskus nähnyt. Ratkaisumme tarjoaa erittäin hyvät mahdollisuudet laajentaa ohjelman toimintaa mm. tukemaan useita erilaisia mediaformaatteja tai yhteisöllisiä palveluita.

Ohjelman käyttöohje

- Mikäli sinulla ei vielä ole omaa Google ID:tä, joudut luomaan sen ennen kuin pääset käyttämään ohjelmaa. Tilin luonti onnistuu esimerkiksi https://www.google.com/accounts/ kautta.
- Kun olet saanut tilisi luotua, kirjaudu sisään ohjelman käyttöliittymän kautta osoitteessa http://veji.appspot.com/ omalla Google ID:lläsi.
- Olet päässyt tietokannan etusivulle. Ylärivistä löytyvät erilaiset toiminnot: voit lisätä elokuvia kohdasta "Add - New movie". Muutkin linkit toimivat, mutta niiden takaa ei löydy minkäänlaista logiikkaa.
- Elokuvan lisääminen tapahtuu kirjoittamalla elokuvan nimi (tai osa siitä) aukeavaan dialogi-ikkunaan kohtaan ”Movie title” ja tämän jälkeen valitsemalla elokuvan kentän alle ilmestyvästä valintaboksista.
- Elokuvan poistaminen omasta listasta onnistuu valitsemalla yhden tai useamman elokuvan (klikkaa valintaboksia) ja valitsemalla "Delete". Klikkaamalla ylintä valintaruutua valinta kohdistetaan kaikkiin listan alkioihin. Ohjelma kysyy vielä vahvistuksen poistokomennolle ennen sen suorittamista.

Ratkaisutapa

- elokuvien tiedot haetaan suoraan IMDB:stä (Internet Movie Database) html-sivuna
- IMDB:stä saatu data parsitaan tietokantaan tallennettavaksi rakenteeksi
- kaikkien elokuvien kaikki tiedot poster-kuvaa lukuunottamatta sijaitsevat Google App Enginen tarjoamassa Datastore-tietokantapilvessä
- elokuvalista esitetään käyttäjälle web-käyttöliittymän avulla
- ohjelmaa on mahdollista ajaa myös omalla koneella, mikäli Google App kehitysympäristö on asennettu – erityisen hyödyllistä tämä on silloin, kun ohjelmaa halutaan debuggailla tai muuten testailla
- ohjelma tulostaa runsaasti ajonaikaista tietoa palvelimen tilasta konsoliin ja lokeihin

Elokuvasta on mahdollista tallentaa seuraavat tiedot:

# Title	movie title
# Year	release year
# Director	movie director
# Plot	few line description of move plot
# UserRating	this is popularity rating from 1/10
# MovieRating	this is a MPAA rating
# Runtime	length in minutes
# Cast	a comma separated list of cast members
# Genres	a comma separated list of genres
# Countries	a comma separated list of countries
# PosterUrl	link to movie poster image

Lisäksi käytössä on taulut käyttäjäkohtaisille elokuvalistoille ja elokuvalistan alkioille. Käytännössä siis jo kerran tietokantaan lisättyä elokuvaa ei poisteta sieltä koskaan, vaikka kukaan ei omistaisi kyseistä elokuvaa.

Ohjelman rakenne

- ohjelman ydin koostuu Python-luokista, jotka käyttävät html:ää käyttöliittymäsivujen piirtämiseen
- tietokantaan kirjoittaminen ja sieltä lukeminen on toteutettu kokonaan Googlen Datastore-rajapinnan kautta, joten se on täysin Python-kielistä
- rakenne on jaettu useampaan pakettiin, jotka ovat toiminnallisuuksiltaan koherentteja
- viestintä Python-luokkien ja käyttöliittymän välillä on toteutettu post-kutsuilla (Ajax), joita luovat javascript-metodit sekä html-formit

Tärkeimmät metodit tietoineen moduleittain				
Moduli	*Metodi ja luokka*	*Parametrit*	*Palautus-arvo*	*Merkitys*
veji	BaseRequestHandler.generate	self, template_name, template_values={}	-	Supplies a common generation function for web page creation.
	MainPage.get	self	-	Lists the movies for logged user and acts as a main page.
	ShowMovieDetailsAction.post	self	-	Retrieves movie information from database and puts it to a dialog shown to the user.
	AddNewMovieAction.post	self	-	Adds a new movie for the current user.

	QueryMoviesAction.post	self	-	Handles the movie search and add query.
	MainFormAction.post	self	-	Performs an action in the user's MovieList. Only 'Delete' action is supported.
	main	-	-	Creates the program and starts it.
imdbConnector	Iconnector.getMoviePoster	movieid (string)	url (string)	Queries IMDB for this specific movie's poster picture URI. Returns it as a string
	Iconnector.getMovieList	movieid (string)	array (string)	searches IMDB with the given query and returns results using the array format
	Iconnector.getMovieData	movieid (string)	array (string)	Fetches moviedata from IMDB and returns it in a defined array
parser	Parser.parseBetween	self, data, start, stop	string	Receives data-string which will be searched, start and stop strings to search for. Everything between these tags will be returned as a result string
datastore	Movie.getValueStringByName	valueName (string)	value (string)	Returns the value with name ‘valueName’.
	Movie.movieConstruct	movieTable, posterUrl	Movie object	Creates a movie from given data.
	MovieList.getMovieList	user	MovieList	Returns the

			object	equivalent MovieList for the specified user and adds it to the database provided that it's a new user
	MovieList.deleteMovie	self, movieid	-	Deletes a single movie from this list by given id provided that it exists
	MovieList.getMovies	self	Movie object array	Return the Movies on this list.
	MovieList.getMovie	self, movieid	Movie object	Returns the Movie that has the specified movieid if it's on this list
	MovieList.addMovie	self, movieid	True, False	Adds a new Movie to this list if the Movie does not exist in the database, it's added there
	MovieListItem	-	-	This class represents one to many -relationships, which maps user's movies to the actual movie-objects

Arvio projektityöstä

- Kerrankin tehtiin jotain hyödyllistä, oikein opettavaista ja ajankohtaista ohjelmointikurssilla. Google App on julkaistu noin 6 kuukautta sitten ja Django-framework on sekin vain muutaman kuukauden ikäinen.
- Olimme asettaneet itsellemme turhan tiukan aikataulun, mutta saatiin projekti silti valmiiksi, ja olemme lopputulokseen tyytyväisiä.
- Ohjelman erityinen ansio on se, että se on toteutettu olemassa olevan palvelun päälle ja että se on täysin käytettävissä internet-yhteyden yli.
- Ohjelma käyttää lukuisia oikeita web-tekniikoita kokonaisuuden aikaansaamiseksi.
- Projektissa oli lukuisia uusia tai muuten vain opeteltavia asioita, ja siten paljon haasteita. Kaikesta huolimatta saavutimme mainion lopputuloksen.
- Projekti oli varsin intensiivinen, mutta mukava ja opettavainen koodauskokemus.

- Valmiin Google App –esimerkin päälle rakennettaessa oppi myös oikean tavan toteuttaa tämän tyyppisiä sovelluksia. Lisäksi se auttoi useampaan tekniikkaan tutustumista.
- Tekniikoita, joita opimme projektin aikana: html, javascript, css, ajax, python, django, web 2.0-tyyppinen sivujen toteutus, Google App Engine ja oliotietokannat
- Projektin versionhallinta hoidettiin koululle pystytetyn SVN:n avulla. Lisäksi kehitysympäristön pystytys vaati tavallista enemmän paneutumista (ssh, svn, googleapp_sdk, eclipse + pydev, python, firefox + firebug)

Ajoesimerkit

Käyttöliittymä on itsensä selittävä ja sillä voi poistaa tai lisätä elokuvia omaan listaansa.

Parannusehdotuksia

- tehokkuus ei ole optimaalinen
    - jos uusi elokuva jo listassa/db:ssä, ei lähdetä etsimään sitä imdb:stä
    - "reaaliaikainen" haku vain jo löydettyihin tuloksiin (ohjelma tallentaa sisäiseen välimuistiinsa jo tehdyt haut)
    - elokuvan tietojen esittäminen tulisi tehdä yhdellä postilla, nyt käytössä 1 post / haluttu kenttä
- oman databasen kentät voisivat olla myös jotain muuta kuin stringejä
    - mahdollistaisi kivoja nice-to-have –ominaisuuksia helpommin
    - toisi hieman nopeuslisää
- interfacet olisivat tämän tyyppiseen ohjelmointiin hyviä, mutta Python ei tue niitä
- testikattavuutta tulisi parantaa, ohjelman crash-testausta ei ole toteutettu, lisäksi unit-testit puuttuvat (tätä yritetty korjata runTests-metodeilla, jotka testaavat tiettyjen luokkien kriittisiä toimintoja)
- käyttöliittymätestaus on tehty käsin
- parserissa on vielä puutteita
    - osa kentistä on kovakoodattuja, sillä Pythonissa ei toiminut pattern-regexpien käyttö

Lähdeviitteet

- Pythonilla tehty valmis Tasklist, saatavissa: http://code.google.com/p/google-app-engine-samples/downloads/list, tasks_20080409.tar.gz
- Perl-skripti IMDB-tiedon parsimiseen, saatavissa: http://svn.mythtv.org/trac/browser/trunk/mythplugins/mythvideo/mythvideo/scripts/imdb.pl
- Google App Enginen tutorialit
- Nettisivujen tekoon liittyviä tutorialeita: www.w3schools.org
- Ulkoisia kirjastoja: Google App Engine APIt, Prototype.js (javascripteillä Ajaxin käsittelyä)
- Lukuisia sähköpostilistojen viestejä, forumeita, nettisivuja ym. googletusta