LINKÖPINGS UNIVERSITET

TDP003 Projekt: Egna datormiljön

TDP003 - Systemdokumentation

Författare

Anton Sköld, antsk320@student.liu.se
William Utbult, wilut499@student.liu.se

INSTITUTIONEN FÖR DATAVETENSKAP

Höstterminen 2017
Version 1.3

2017-10-10

Innehåll

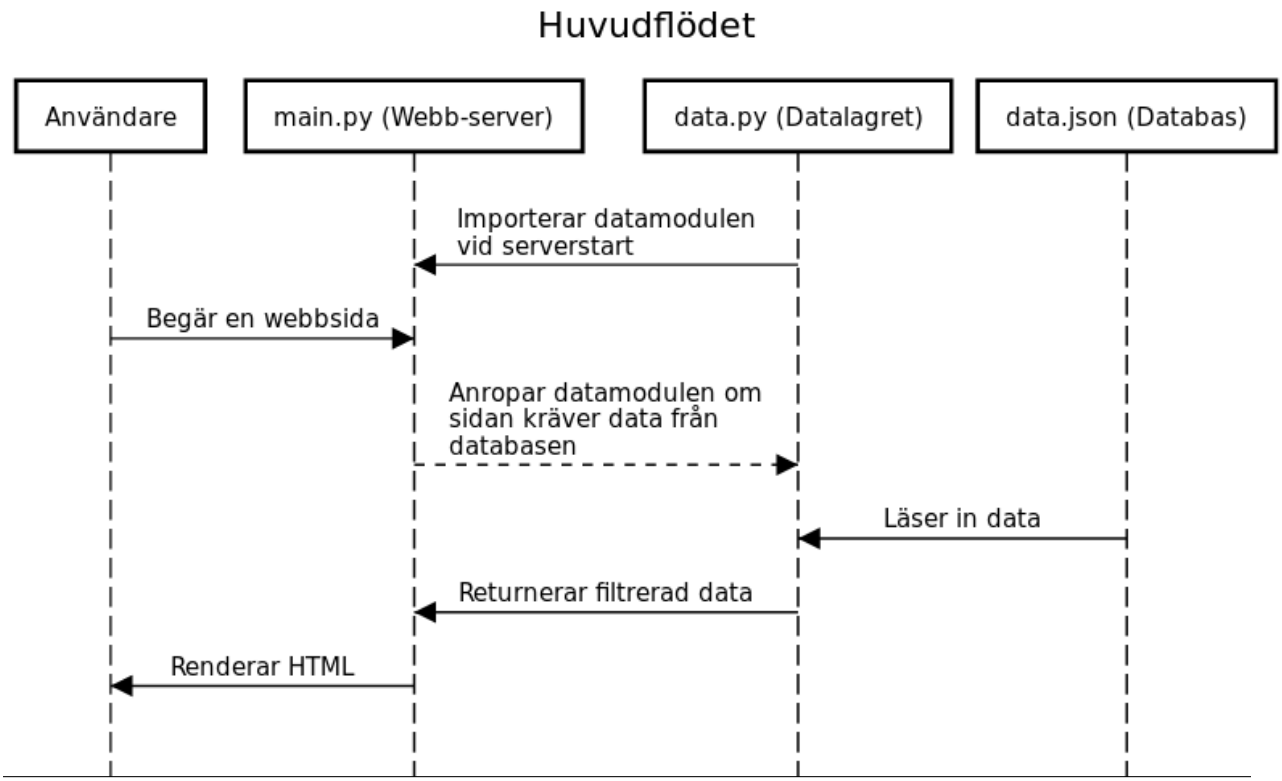
1 Revisionshistorik

Ver.	Revisionsbeskrivning	Datum
1.0	Grundläggande mall och Överblick tillagd	2017-10-03
1.1	Lade till dokumentation för funktioner	2017-10-09
1.2	Redigerad dokumentation över funktioner och tillagd sektion för tester, fel och loggning	2017-10-10
1.3	Databasformat och språkversion tillagt	2017-10-10

2 Krav

Se systemspecifikationsdokumenten:
`http://www.ida.liu.se/~TDP003/current/projekt/dokument/systemspecifikation.pdf`
`http://www.ida.liu.se/~TDP003/current/portfolio-api_python3/`

3 Översikt



1. Servern startar och importerar datamodulen
2. Användaren gör en http-begäran till servern
3. Om sidan kräver data från databasen anropas datamodulen
4. Databasen läses in
5. Datan filtreras och skickas tillbaka till servern
6. Servern renderar HTML med rätt inehåll och presenterar det för användaren

Steg 2-6 repeteras för varje http-begäran användaren gör.
Systemet är skrivet i Python 3.5.2 & (Flask + Jinja2) för formatering och presentation i HTML5 & CSS3.
Databasen i JSON-format ser ut som följande:

```
[
  {
    "project_id": 1,

    "project_name": "Portfolio",

    "start_date": "2017-09-15",

    "end_date": "YYYY-MM-DD",

    "course_id": "TDP003",
```

```
    "course_name": "Projekt: Egna datormiljön",

    "techniques_used": ["HTML", "CSS", "Python"],

    "description": "Short description",

    "long_description": "Long description",

    "image": "https://url",

    "external_link": "https://url"
  },
  {...}
]
```

4 Funktionförklaringar

4.1 main.py - Webbservern

4.1.1 `index()`

Hämtar HTML för indexsidan och dess mallar
Anropas via Flask route ”/”

Returnerar: sträng
HTML för indexsidan

4.1.2 `search()`

Läser in all input via HTML-form för sökningen Hämtar sökresultaten från data.py och HTML för listsidan och dess mallar
Anropas via Flask route ”/list/”

Returnerar: sträng
HTML för listsidan (med projekt matchande sökningen om sådan gjordes)

4.1.3 `techniques()`

Hämtar unika använda tekniker från data.py och HTML för techniquessidan och dess mallar
Anropas via Flask route ”/techniques/”

Returnerar: sträng
HTML för techniquessidan (med alla unika använda tekniker i databasen)

4.1.4 `project(id)`

Hämtar projektinformation från data.py och HTML för projectsidan och dess mallar
Anropas via Flask route ”/project/<id>där id är projektets id

Parametrar:
id (heltal) - Projektets ID angivet i databasen

Returnerar: sträng
HTML för projectsidan (med projekt matchande sökningen om sådan gjordes)

4.1.5 `page_not_found(e)`

Hämtar HTML för 404sidan och dess mallar
Anropas via Flask errorhandler(404)
e har felkod och felmeddelande

Parametrar:
e (sträng) - Felkod och felmeddelande (ges av Flask)

Returnerar: sträng
HTML för 404sidan (e används inte för tillfället)

4.2 data.py - Datalagret

4.2.1 `load(filename)`

Läser in JSON-formaterad projektdata från den givna filen och returnerar en lista av projekt sorterade efter ID.

Parametrar:
filename (sträng) - filnamn/sökväg

Returnerar: lista
Lista innehållande en tabell per projekt (se databasen)

4.2.2 `get_project_count(db)`

Returnerar antalet projekt i en projektlista.
db är den givna listan.

Parametrar:
db (lista) - databas/projektlista

Returnerar: heltal
Antalet proejekt i databasen/listan

4.2.3 `get_project(db, id)`

Returnerar ett projekt med givet id från projektlista.
db är den givna listan.
id är ID får projektet som ska returneras.

Parametrar:
db (lista) - databas/projektlista
id (heltal) - projekt ID

Returnerar: tabell
Projektet inehållande projektinformation

4.2.4 `search(db, sort_by, sort_order, techniques, search, search_fields`

Hittar och returnerar projekt från projektlista som matchar givna kriterier.

Parametrar:
`db` (lista) - databas/projektlista
`sort_by` (sträng) - fält som resultaten sorteras efter (se databas). Standard: "`start_date`"
`sort_order` (sträng) - Sorteringsordning. 'asc' för stigande, 'desc' för fallande. Standard: 'desc'
`techniques` (lista) - List över använda tekniker (strängar) där alla måste matcha med projektet
`search` (sträng) - sökord att matcha med projektinformation
`search_field` (lista) - Lista över fält (strängar) att matcha sökord med (se databas).
None ger alla. Standard: None

Returnerar: tabell
Projektet inehållande projektinformation

4.2.5 `get_techniques(db)`

Returnerar en lista av alla unika använda tekniker i den givna databasen

Parametrar:
`db` (lista) - databas/projektlista

Returnerar: lista
Tekniker (strängar)

4.2.6 `get_technique_stats(db)`

Samlar statistik över alla tekniker i den specificerade projektlistan.
Returnerar en tabell med alla unika använda tekniker och listor över vilka projekt som använde dem

Parametrar:
`db` (lista) - databas/projektlista

Returnerar: tabell
Tabell av tekniker (listor) som i sin tur innehåller tabeller med 'id' = projektid, 'name' = projektnamn

5 Tester, felhantering och loggning

5.1 Tester

5.1.1 Datalagret

Datalagret fungerar just nu precis som det ska. Om ändringar skulle ske i det så finns det flera sätt att testa så att funktionerna fortfarande beter sig som de ska.

Första sättet är att kolla i källkoden, längst ner i data.py så finns det en hel del utkommenterade tester som vi skapade under utveckling av datalagret. De testar olika funktioner, och många kombinationer av sökparametrar. Man måste manuellt jämföra returdatat ur testerna med data.json-filen för att få reda på om det är det förväntade datat som returneras.

Andra sättet är genom de gemensamma testfallen som skapades dels av klassen och dels av kursledare.
De gemensamma testerna finns tillgängliga i `https://gitlab.ida.liu.se/filst04/tdp003-2017-database-tests/tree/master`.
För att använda de gemensamma testerna så klonar du repositoryt till lokal maskin, sedan kopiera in `data.py` (Namn är viktigt) i mappen, och sedan kör `data_test.py` med valfri modern Python-interpretator.
Sedan skrivs det ut information i terminalen om hur många tester som godkändes/misslyckades och hänvisar till vilka tester det var.

5.1.2 Presentationslagret

Presentationslagret har inte några specifika tester så som datalagret har. Omfattningen av möjliga tester skulle vara att göra sina ändringar, ladda om webbsidan, och läsa eventuella felmeddelanden.

5.2 Felhantering

5.2.1 Datalagret

Fel som uppstår inom datalagret är aldrig dödliga. De fel som kan uppstå, som att en fil inte finns eller att man har matat in dålig information, returnerar antingen tomma listor, eller ett None-objekt. När det sker så får man dubbelkolla sin input, och sedan stegvis gå igenom källkoden för att hitta de logiska buggarna.

5.2.2 Presentationslagret

Det fel som hanteras i presentationslagret just nu är 404-fel, att sidorna inte kan hittas.
Om man vill hantera ännu fler fel och man vet felkoden för felen så finns en @app.errorhandler(felkod) funktion tillgänglig i Flask. Den kan användas före en funktion för att få felhantering för det felet inuti funktionen.
De fel som kan uppstå under utveckling när man har dålig data/ofullständig kod med mera är oftast Jinja2-fel. Jinja2 har en inbyggd debugger som visar de senaste funktionsanropen som skedde innan felet uppstod.

5.3 Loggning

Datalagret och presentationslagret saknar loggning.
OpenShift (som systemet ligger på) har inbyggd loggning över vissa fel och händelser.
För att se webbserverns senaste log, kör kommandot `"rhc tail portfolio"`, detta rekommenderas att vara körande vid uppdatering av sidan.