Inlämningsuppgift 2

Mikael Svahnberg*

2024-09-13

1 Översikt över Inlämningsuppgifterna

1. Samarbete och Konfigurationshantering
   - Lite att göra i början (sätt upp repo, börja använda `git`)
   - Resten kommer "på köpet" när ni arbetar med uppgift 2.
   - **Mål**: alla bidrar med *någonting* med hjälp av git.
2. *Implementation och Dokumentation*
   - (se resten av presentationen för mer detaljer)
   - **Mål**: Att inse att applikationer kan bestå av flera olika (och separata) delar som samarbetar.
   - **Mål**: Att dokumentera er programkod och reflektera över ert arbete.
3. Testning och Debuggning
   - Planering av hur och vad som skall testas
   - Testa *någon* del av systemet ni implementerar
   - Debugga *någon* del av systemet ni implementerar
   - Se exempel från övningsuppgifterna; Ni skall göra ungefär likadant.
   - **Mål**: Att se testning som en naturlig och självklar del av implementationen
   - **Mål**: Att inse att det finns ett kraftfullt verktyg rakt framför er för att hjälpa er förstå vad som händer i era datorprogram.

2 Implementation och Dokumentation

Uppgiften i Stort

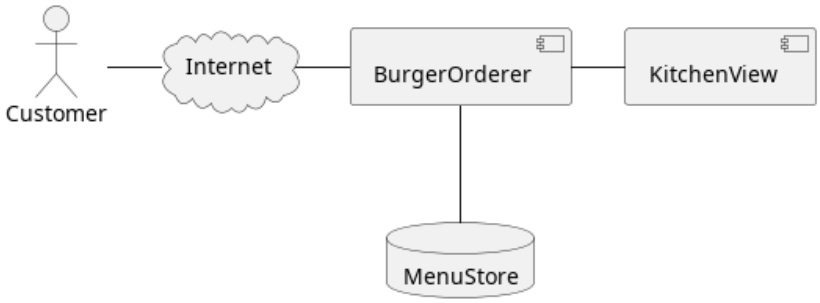
- Ni skall skriva två datorprogram: *BurgerOrderer* och *KitchenView*
- *BurgerOrderer* och *KitchenView* är egentligen ganska lika: De lyssnar efter anrop och reagerar på dessa.

*Mikael.Svahnberg@bth.se

  - Har ni löst *BurgerOrderer* så är *KitchenView* till 80% färdigt redan.

- **Mål**: Att inse att applikationer kan bestå av flera olika (och separata) delar som samarbetar.
- **Mål**: Att dokumentera er programkod och reflektera över ert arbete.

3 BurgerOrderer



- BurgerOrderer skall *lyssna på en nätverkskoppling*, t.ex. `http://localhost:8000/`
  - `localhost` är din egen dator. Kan också heta `127.0.0.1`
  - `:8000` i det här fallet är en *nätverksport*.
  - `/` är en "API-ändpunkt": Den här delen av adressen berättar vad man vill att BurgerOrderer skall göra.
    * Det här kan ni planera nu, exempel:

API	Beskrivning
`/`	Visa förstasidan. Lista alla hamburgare.
`/burger/name`	Visa en viss hamburgare
`/buy/name`	Köp en viss hamburgare
`/buy/name?cfg=extraBacon`	Köp en viss konfiguration av en hamburgare

Exempelkod: `https://codeberg.org/mickesv/BO-py.git`

4 Steg Ett: Lyssna på Nätverkskoppling

- Enligt uppgiftsbeskrivningen: lyssna på uppkoppling från en webläsare
  - Kan göras mycket mer lågnivå, men det täcker vi inte i den här kursen.

Exempel:

- JavaScript/Node.js: `express.io`

```
const express = require('express');
const app = express();
const PORT = 8000;
```

```
app.get('/', startPage);
app.listen(PORT, () => { console.log('Listening on port', PORT); });

function startPage(req, res) {
    console.log('Loading start page');
    return res.send('Welcome!');
}
```

- Python: `flask` (≈ version 3.0.3)

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def frontpage():
    print('Loading start page', flush=True)
    return 'Welcome!'

if __name__ == "__main__":
    app.run(debug=True, host="0.0.0.0", port=8000)
```

5 Vad Webläsaren gör

Webläsaren (Firefox, Chrome, Safari, Edge, ...)

1. kopplar sig mot en öppen nätverksport
2. skickar upp eventuella parametrar
3. ber att få *någonting* tillbaka.

"Någonting" kan vara

- en kod som t.ex. säger om allt är ok (200)
- en text
- en html-sida
- en bild, en css-fil, eller vad som helst.

Webläsaren *tolkar* sedan det den får, och försöker visa det.

- Det kan finnas instruktioner om att hämta andra filer också (t.ex. css-filer). Då gör webläsaren det.
- Om det är html så formatterar den texten
- Om det är något annat så försöker den så gott den kan visa det eller be ett annat program om hjälp.

6 HTML

- Beskriver bara hur saker skall visas
  - Man beskriver innehållet med en start-tag, e.g. `<h1>`, och en slut-tag `</h1>` .
  - ex `<h1>innehåll</h1>`

Exempel

```
<h1>Rubrik</h1>
<h2>Underrubrik</h2>
<p>Stycke</p>

<ul> <!-- En punktlista -->
<li>Punkt 1
<li>Punkt 2
</ul>

Text med <i>italics</i> och <b>fetstil</b>

<a href="/buy/heartstopperBurger">En klickbar länk</a>
<a href="https://zombo.com/">En länk till en annan websida</a>

Man kan också stoppa in bilder bland texten: <img src="/images/blommor.png">
(Men man behöver se till att webservern hittar
till /images/blommor.png och vet vad den skall göra ).
```

7 Bygg vidare på BurgerOrderer

- Vi började med minsta möjliga; vi returnerade texten `'Welcome!'` och inget annat.
- Nu kan vi lägga till lite html-kod:

```
@app.route('/')
def frontpage():
    print('Loading start page', flush=True)
    return '<h1>Welcome!</h1>'
```

- Det blir lite fult att ha allt i samma metod när det blir större
- *Refactor* och skapa en ny metod

```
def renderFrontpage():
  pg = '<h1>Welcome!</h1>'
  pg += '<p>lite mer text'
  return pg

@app.route('/')
def frontpage():
    return renderFrontpage()
```

8 Flera Olika Hamburgare

- Så småningom vill vi hämta från en databas
- Men just nu vill vi bara ha en lista
- Vi kan hårdkoda, men det kostar bara lite extra att förbereda för framtiden.

```
staticBurgers= [{"name":"fettburgare"},
                {"name":"gnuttburgare"},
                {"name":"isterburgare"}]

def getBurgers():
    return staticBurgers;

def renderFrontpage():
    pg = "<h1>Welcome to BurgerOrderer</h1>"
    pg += "<P><UL>"

    for burger in getBurgers():
        pg += "<LI>" + burger["name"]

    pg += "</UL>"
    return pg
```

9 KitchenOrderer

- Är precis samma: lyssna på en nätverkskoppling, skriv ut (till konsolen)
- Kan lägga lite tid på att göra det någorlunda snyggt.

Exempel:

```
kitchenview-1    | One gnuttburgare ordered with the following options:
kitchenview-1    |  - noOnion
kitchenview-1    |  - extraBacon
kitchenview-1    | 172.19.0.1 - - [13/Sep/2024 10:47:10] "GET /buy/gnuttburgare?noOnion&ex
```

10 Skicka från BurgerOrderer till KitchenView

- Behövs ett ramverk till: `requests` (≈ v 2.32.3)

```
import requests

def sendToKitchen(burgerName, args):
    requrl = makeURL(burgerName)
    requrl = addOptions(requrl, args)

    print('Using KitchenView URL: ' + requrl)
    # exempel på requrl: http://kitchenview:5000/buy/gnuttburgare?noOnion&extraBacon&
    requests.get(requrl);
    return
```

11 Sammanfattning

- Exempelkod: `https://codeberg.org/mickesv/BO-py.git`
- Skriv applikationerna *BurgerOrderer* och *KitchenView*
  - BurgerOrderer skapar html-sidor som kan returneras till användarens webläsare.
  - Använd så lite HTML som nödvändigt för att få klickbara länkar
    - Olika länkar (ex. till `/`, `/buy/` , `/info/`) ger olika svar.
  - KitchenView tar bara emot information och skriver ut till skärmen.
- Jobba med detta ett tag, så småningom kommer vi
  - Stoppa in BurgerOrderer och KitchenView i varsin liten Container (med *Docker* eller *podman*)
  - Hänga på en databas som BurgerOrderer använder sig av.