Projektportfölj

Portfölj projektet är baserat på ramverket Flask.

Upphovsmän: Warhell Nasim och Henrik Björs

Installationsmanual

Innehållsförteckning

System beskrivning

Portfölj projektet är en mallbaserad webserver skapad utav Warhell Nasim och Henrik Björs.

Portföljsystemet förenklar hantering utav mindre portfolio hemsidor för användare med begränsad kunskap inom HTML och programmering överlag.

Instruktioner för installation

Installationsinstruktionerna går igenom installation för Windows 7 och Linux Mint 17. Andra versioner eller operativsystem garanteras inte fungera med instruktionerna.

Windows 7

1. Ladda ner och installera Python 3.4 från
   https://www.python.org/ftp/python/3.4.1/python-3.4.1.msi

2. Ladda ner portföljsystemet från denna
   https://github.com/warna720/TDP003/releases/download/1.0/portfolio.zip.

3. Öppna upp zip filen i valfritt program. Extrahera innehållet till en lämplig plats.

4. Öppna upp kommandotolken och navigera dig till mappen som du extraherade innehållet från zip filen till.

5. För att skapa en virtuell miljö, skriv följande kommando i kommandotolken:
   C:\Python34\python.exe –m venv portfolio

6. Nu har vi skapat en virtuell miljö. För att aktivera miljön, skriv in följande i kommandotolken:
   portfolio\scripts\activate

7. Till slut installerar vi även Flask som används utav portföljsystemet. Skriv in följande i kommandotolken:
   pip install flask

8. För att starta portföljsystemet skriver vi in följande i kommandotolken:
   python portfolio.py

Linux Mint 17

1. Ladda ner och installera Python 3.4, virtualenv samt git genom att skriva in följande i terminalen:
   sudo apt-get update && sudo apt-get install –y python3 python3-pip python-virtualenv

2. Ladda ner portföljsystemet genom att skriva in följande i terminalen:
   svn checkout https://svn-und.ida.liu.se/courses/TDP003/2014-1-PRA1/ip1-2-5/TDP003/portfolio --username <användarnamn> ~/portfolio

   Byt ut ”<användarnamn>” mot ett giltigt användarnamn, lösenord skriver du in när terminalen ber om det. Observera att du kan ladda ner portföljsystemet till en annan plats genom att byta ut ’~/portfolio’.

3. Navigera dig till mappen som du valt för lagring av portföljsystemet, har du valt ’~/portfolio’ kan du skriva in följande i terminalen:
   cd ~/portfolio

4. För att skapa en virtuell miljö, skriv följande kommando i terminalen:
   virtualenv portfolio

5. Nu har vi skapat en virtuell miljö. För att aktivera miljön, skriv in följande i terminalen:
   . portfolio/bin/activate

6. Till slut installerar vi även Flask som används utav portföljsystemet. Skriv in följande i terminalen:
   pip install flask

7. För att starta portföljsystemet skriver vi in följande i terminalen:
   python3 portfolio.py

Lägga till projekt

För att lägga till ett projekt behöver du redigera filen ”data.json” som ligger i roten av projektmappen.

När du öppnar filen ”data.json” för redigering ser du att filen börjar med en hakparantes och slutar även med en hakparantes, dessa hakparanteser behöver du inte redigera.
Ett projekt ligger mellan två måsvingar. Det innebär alltså att för varje måsvinge är det ett nytt projekt.

Vi börjar med att kopiera det sista projektet, inkludera även måsvingarna i kopieringen.
Sedan lägger vi till kommatecken efter det kopierade projektet, detta för att indikera att det inte är det sista projektet.
Därefter klistrar vi in det kopierade projektet mellan kommatecknet och måsvingen.

Du kan nu ändra värdet på parametrarna allt eftersom.
När du är färdig med det nya projektet måste du spara filen för att det nya projektet skall sättas i produktion.

Redigera projekt

Byta ut stor bild

Om vi exempelvis vill byta ut en stor bild i ett projekt så börjar vi med att lägga in bilden vi tänkt använda under mappen ”static” som vi finner i rotmappen till projektet.

När du lagt in bilden öppnar vi upp filen ”data.json” för redigering. Filen ”data.json” befinner sig i rotmappen till projektet.

Lokalisera projektet som du vill byta bild på. Ändra värdet, innanför citationstecknen, för parametern ”big_image” till ”static/images/” ihopsatt med filnamnet för bilden. Spara filen för att ändringarna skall sättas i produktion.

Ifall ingen ändring sker, pröva med att använda kortkommando CTRL + F5 eller SHIFT + F5, detta för att rensa cacheminnet i webbläsaren.

Lägga till teknik

Börja med att öppna upp filen ”data.json” för redigering. Filen ”data.json” befinner sig i rotmappen till projektet.

Lokalisera projektet som du vill lägga till ytterligare tekniker på. Värdet på parametern ”techniques_used” är en två hakparanteser med dem olika teknikerna emellan. För att lägga till en teknik börjar vi med att lägga till ett kommatecken efter den sista tekniken, därefter ett mellanslag och till slut skriver vi in den nya tekniken. Spara filen och fascineras av tekniken.

Felsökning

Loggning

Felsökning sker främst genom analys av loggfilen som portföljsystemet genererar, samt genom de felmeddelanden som portföljsystemet rapporterar när ett eller flera fel sker.

Portföljsystemet loggar funktionsanrop, parametrar, url (webbadress) samt datum och tid till en loggfil. Loggningsfilen befinner sig i roten av portföljsystemets mapp och är döpt till ”log.txt”. För avancerade användare kan det vara lämpligt vid felsökning att kolla igenom loggfilen.

Loggfilen är formaterad i sektioner och är sorterad efter datum/händelse i stigande ordning, senaste händelserna hittas längst ner på loggfilen.

Sektionerna innehåller ett antal rader med information.

Exempel #1

```
Request, GET, /project/6955
load, filename=data.json
get_project, id=6955, db=data.json
Page not found, 404: Not Found
```

Första raden innehåller information om vilken typ av begäran som gjordes och vilken webbadress det var. I detta fall är det begäran utav typen ”`GET`” och webbadressen är ”`www.*.*/project/6955`”.

Nästa rad innehåller information om vilka funktioner respektive argument som anropats ifrån datalagret. I detta fall var det funktionerna ”`load`” med argumentet ”`data.json`” och ”`get_project`” med argumenten ”`6955`” och ”`data.json`”.

Till slut, på sista raden, avslutas sektionen med ett meddelande. Meddelandet innehåller information om allt gick som det ska eller ifall ett fel inträffade. Ifall ett fel inträffade loggas en beskrivning utav det inträffade felet. I detta fall har ett fel inträffat med HTTP felkoden ”`404`”, som innebär att sidan inte kunde hittas.

Exempel #2

```
Request, POST, /search
load, filename=data.json
get_technique_stats, db=data.json
get_techniques, db=data.json
search, db=data.json, sort_by=project_no, sort_order=desc,
techniques=[], search=python, search_fields=[”project_name”]
Success.
```

I denna sektion ser vi att det har skett en begäran utav typen ”`POST`” på webbadressen ”`www.*.*/search`”.

Därefter förekommer ett antal rader med funktioner tillsammans med dess argument som har anropats ifrån datalagret. Vi ser att funktionerna ”`load`”, ”`get_technique_stats`” och ”`get_techniques`” har anropats tillsammans med argumentet ”`data.json`”. ”`Search`” funktionen har anropats med ett antal olika argument.

Till slut ser vi att sektionen avslutas med ”`Success`” vilket innebär att inget fel inträffade.