Timebox1 – user controller

Oversigt

OpgaveNavn	Implementering af en user controller funktion		
Implementering af krav	N/A		
Udført af	Jan	**Dato**	29-09-2021
Timebox	2	**Område**	website

Contents

INTRODUKTION **1**
ANALYSE **2**
IMPLEMENTERING **2**
VERIFIKATION **3**
TESTRESULTAT **3**
KONKLUSION **3**

Introduktion

For at sørge for at ”field user” ikke kan gå ind på ”office user” siderne, og at ”office user” ikke går ind på ”field user” siderne, bliver der lavet en funktion der holder de respektive bruger typer, på de sider der er ment til dem.

Analyse

For at gøre funktionen nemmere at implementere, vil der blive lavet en python dekoratør, i stedet for en standard python funktion.

En dekoratør, er en python funktion der tager en funktion som argument, og returnere så en ”ny” funktion eller udvider en eksisterende funktion, baseret på argumentfunktionen og dekorator funktionen.

For at gøre det nemmere at implementere en dekoratør kan python bibliotekets ”functools” modulet ”wraps” importeres.

Wraps modulet er en dekoratør der gør det muligt at passere doc strengen, fra en argumentfunktion, med til den ”nye” funktion.

Implementering

Dekoratøren er lavet så admin brugere har adgang til alle typer hjemmesider.
Dekoratøren for både ”field users” og ”office users”, er den samme, med undtagelse af hvem der har lov til at se siden.

På Figur 1, kan der ses den udviklede custom dekoratør.

- Linje 26: wraps dekoratøren tager også argumentfunktionen som argument, for at kunne passere doc strengen med til den nye funktion.
- Linje 27: den indre funktion i dekoratøren, tager ”request” fra hjemmesiden som argument, for at kunne tjekke bruger typen, af den nuværende bruger, gennem et kald til Djangos user model objekt.
- Linje 29: hvis bruger typen på en ”office user” page er enten ”admin” eller ”office user”, kan de få lov til at se siden, ved at passere dem videre til argumentfunktionen.
- Linje 32: hvis brugeren er ”field user”, skal brugeren dirigeres hen til ”field userhomepagen”.
- Linje 34: hvis brugeren ikke kan genkendes, skal der dirigeres hen til loginforsiden.

```
18  # custom decorator
19  def user_controller(function):
20      """Checks the user type, and redirects if wrong type..."""
26      @wraps(function)
27      def wrap(request, *args, **kwargs):
28          usertype = request.user.userType
29          if usertype == "Admin" or usertype == "Office user":
30              return function(request, *args, **kwargs)
31          elif usertype == "Field user":
32              return redirect('field_user_home')
33          else:
34              return redirect('front_page')
35
36      return wrap
```

Figur 1 backend decorator

Verifikation

Der skal testes:

- en bruger bliver dirigeret hen til deres pågældende side

Det er forventet at:

- Serveren er oppe at køre.
- De forskellige bruger typer er oprettet.
- Testene startes fra login siden.
- Dekoratørene er sat på forholdsvis
    - Field_user_homepage
    - Office_user_homepage

Tabel 1: Tests til verifikation af opgave

Test	Test Steps	Pre-requisites	Pass-betingelser	Resultat
Office user	1. log ind som ”office user” 2. indsæt url’en: /field_user_home	N/A	1. der bliver dirigeret hen til /office_user_home	Bestået
Field user	1. Log ind som ”field user” 2. indsær url’en: 3. /office_user_home	N/A	1. der bliver dirigeret hen til /field_user_home	Bestået

Testresultat

Ved begge test’s blev brugeren dirigeret hen til deres respektive homepage.

Konklusion

Det er nu muligt at sikre den rigtige type bruger kommer ind på brugerens rigtige sider, og det er nu ikke muligt at komme ind på en anden brugertype end ens egen.
Brugertypen ”admin” kan tilgå alle sider for alle brugertyper.
Implementeringen af dekoratørene virkede som det skulle, samt var det nemt at implimentere.