

UAA12 Flask TP3 – Sessions et cookies

1 Test1 - Les cookies : découverte

Un **cookie** est un **petit fichier** (de l'ordre de quelques kilo-octets) que l'on **enregistre temporairement sur l'ordinateur du visiteur**. En général, il ne stocke qu'une information à la fois.

Les cookies sont utilisés pour 3 raisons principales :

- gestion des **sessions** : login, panier d'achat, score d'un jeu, ou tout autre chose dont le serveur doit se souvenir.
- **personnalisation** : préférences utilisateur, thèmes, et autres paramètres.
- suivi : enregistrement et analyse du comportement utilisateur.

Les données du cookie sont envoyées par le serveur dans l'en-tête de la réponse à la requête http.

Pour en savoir plus sur comment Firefox gère les cookies :

<https://support.mozilla.org/fr/kb/cookies-informations-sites-enregistrent>

1. Copie le dossier flask_template et renomme-le xx_test2_cookies .

Dans le dossier **templates**, crée 2 fichiers :

- **index.html**
- **page1.html**

1. Dans index.html et page1.html

Affiche les pages ci-dessous, les nom, prénom et âge provenant de variables Jinja

Page index.html dans le navigateur:

Salut Louis !
Tu es à l'accueil de mon site. Tu veux aller sur une autre page ?
[Lien vers page1.php](#)

Page page1.html dans le navigateur:

Re-bonjour Louis De Funes , sois la bienvenue
Tu es né en 1914

2. Dans view.py, fonction index() :

- Prépare la page à retourner index.html avec la fonction **make_response()**
- Crée les cookies pour 5 sec et rajoute-les dans l'en-tête de la réponse par la fonction **set_cookie()**
- Retourne la page index.html avec les cookies

Code dans view.py:

```
@app.route('/')
def index():
    # Préparation de la page index.html à retourner :
    resp = make_response(render_template("index.html", prenom_jin='Louis'))
    # création des cookie et rajout à index.html :
    resp.set_cookie('prenom_cook', 'Louis', 5)
    resp.set_cookie('nom_cook ', 'De Funes', 5)
    resp.set_cookie('age_cook ', '107', 5)
    # retourne index.html et les cookies :
    return resp
```

3. Dans view.py, fonction page1() :

- Lis les cookies grâce `request.cookies.get('nom_cook')` (applique `get()` à la variable `request.cookies`) ou directement en accédant à la variable dans la liste `request.cookies['nom_cook']`
- Affiche dans le navigateur le contenu des cookies

Code dans view.py:

```
@app.route('/page1.')
def page1():
    return request.cookies # ou dans le terminal avec print('debug : ', request.cookies)
```

- Retourne la page1 en affectant les variables Jinja nom, prenom, age par le contenu des cookies.

Code dans view.py:

```
@app.route('/page1')
def page1():
    cookies = request.cookies
    print("debug: ", cookies)
    p_py = request.cookies['prenom_cook '] # lecture des cookies
    n_py = request.cookies['nom_cook ']
    a_py = request.cookies['age_cook ']
    print("debug: ", p_py, n_py, a_py)
    return render_template("page1.html", prenom_jin=p_py, nom_jin=n_py, age_jin=107)
```

2 Test2 - Les cookies : nombre de visites

Affiche dans une page le nombre de fois qu'elle a été vue (nombre de visiteurs).

Pour cela tu dois :

- Vérifier l'existence du cookie avec le code suivant :

Code dans view.py

```
if 'nom_du_cookie' not in request.cookies: # si le cookie n'existe pas
    print("le cookie n'existe pas")
else:                                     # si le cookie existe
    print("le cookie existe")
```

- Si le cookie n'existe pas (1^{ère} connexion à cette page), il faut le créer avec nombre de visiteurs = 1.
- Si le cookie existe, il faut lire son contenu, incrémenter le nombre de visiteur et mettre à jour le cookie (c'est-à-dire re-crée un cookie).
- Pour tester tout ton programme, crée le cookie pour une durée de quelques secondes.

3 Test3 - les sessions

La variable de session permet de conserver des variables sur toutes les pages du site pendant toute la durée de la présence du visiteur (GET et POST étant plutôt utilisés pour transmettre des informations une seule fois d'une page à l'autre).

Elle est stockée **sur le serveur** dans un dossier temporaire comme un cookie, mais **encrypté**.

1. Copie le dossier flask_template et renomme-le xx_test3_session .

Dans le dossier **templates**, copie les 2 fichiers ci-dessous de xx_test1_cookies :

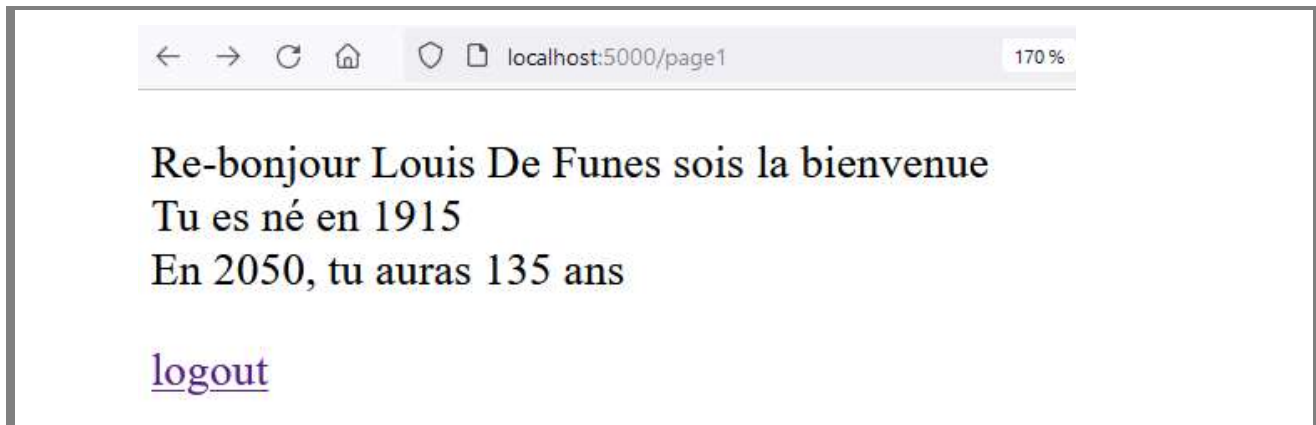
- index.html
- page1.html

2. Dans index.html et page1.html

Page index.html dans le navigateur:



Page page1.html dans le navigateur:



3. Dans view.py, fonction index():

- Ecris le code secret pour éviter que l'utilisateur puisse modifier le cookie de session.
- Crée 3 variables de session prenom_sess, nom_sess, age_sess
- Affiche à l'écran les variables de session

Code dans view.py:

```
app.secret_key = "codesecretintrouvable"
@app.route('/')
def index():
    session['prenom_sess']='Louis'                # création var de session
    session['nom_sess']='De Funes'
    session['age_sess']=107
    return render_template("index.html")
```

4. Dans view.py, fonction page1():

- Vérifie l'existence des variables de session et affiche-les à l'écran

Code dans view.py:

```
@app.route('/page1')
def page1():
    p_py=session['prenom_session ']              # lecture session
    n_py=session['nom_session ']
    a_py=int(session['age_session '])
    print("debug: ", session,p_py,n_py,a_py)
    return ...
```

- Lis les variables de session nom, prenom, age et affecte leur contenu aux variables Jinja dans page1.html

5. Dans view.py, fonction logout():

- Efface les variables de session (elles seront de tout façon effacées lorsque l'utilisateur quitte le site)

Code dans view.py:

```
@app.route('/logout')
def logout():
    session.pop('prenom', None)
    session.pop('nom', None)
    session.pop('age', None)
    return "Var de session effacées"
```

4 Test4 Synthèse Get / Post / Sessions / Cookies

Le but de cet exercice est de faire une synthèse des différentes méthode.

Crée un fichier index.html dans lequel tu peux passer :

- Le nom par la méthode GET via un lien vers le 2^{ème} fichier page.html
- Le prénom par la méthode POST via un formulaire.

Lors du chargement de cette page, l'âge sera sauvegardé dans un cookie, la rue dans une variable de session.

Crée un fichier page.html dans lequel tu afficheras le contenu des 4 variables et un lien de retour la page d'index.

- ⇒ Pense à écrire et tester ton code étape par étape en affichant le contenu de des variables dans le terminal :

Code dans view.py : debug et lecture des variables

```
print("debug : ", request.form, request.args, request.cookies, session)
```

- ⇒ Selon si on remplit le formulaire ou si on click sur le lien, soit une variable POST, soit une variable GET sera créée. Le cookie sera effacée après 5 sec. Il faut donc vérifier dans view.py l'existence de ces variables.

Code dans view.py : test existence des variables

```
if request.method == 'GET':  
    ...  
if request.method == 'POST':  
    ...  
if nom_var_cook in request.cookies:  
    ...
```

Page index.html dans le navigateur:

[lien vers la page \(nom par méthode GET\)](#)

Prénom: (par méthode POST):

Page page.html dans le navigateur:

nom par get:

prenom par post: Charlie

age par cookie: 107

rue par session: Félix Happ

[Retour vers la page d'accueil](#)

5 Test5 Post / Sessions / Cookies - Fiche de renseignements

Reprends l'exercice test4 de uaa12_flask_tp2_get_post_from sur la fiche de renseignements.

Dans le fichier **cible.html**, sauvegarde toutes les données rentrées dans des cookies, la qualité et le nom dans des variables de sessions.

Rajoute un fichier **page1.html** qui affiche les variables de session et cookies

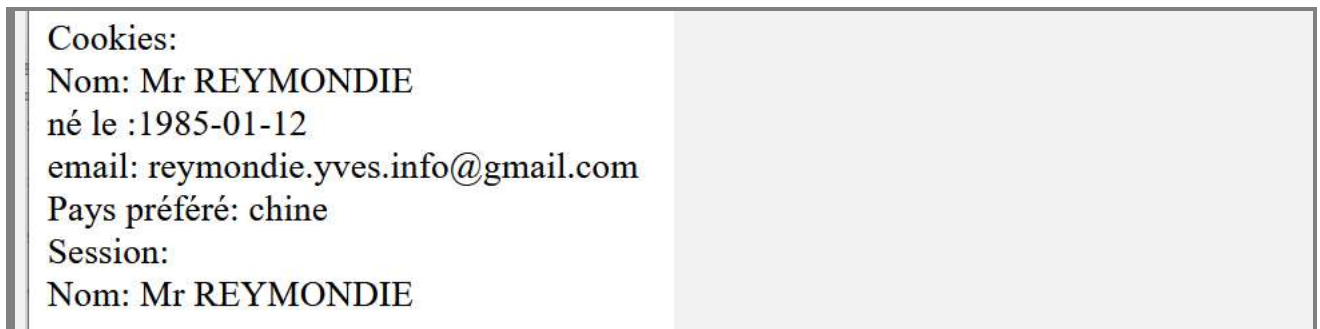
Page Index.html dans le navigateur:



Page cible.html dans le navigateur:

page du site' and '[Retour vers le formulaire](#)'. The text is on a light gray background." data-bbox="80 457 911 541"/>

Page page1.html dans le navigateur:



Pour aller plus loin ...

Challenge

Regroupe les codes des 3 fichiers dans un seul fichier index.html

Il te faudra vérifier si le formulaire a été rempli, si les cookies et les variables de session existent...