Sesiones HTTP

Sesiones

- Definición:
  - Información que podemos recordar entre peticiones para los usuarios de nuestra aplicación
  - Los datos de la sesión podrían **no** viajar en las preguntas/respuestas entre cliente servidor
    - Requiere almacenamiento en el lado del servidor
- En el servidor:
  - La información se puede mantener en archivos o en bases de datos
  - Cada sesión es identificada por un identificador único de sesión (SID)
  - El cliente recibe el SID desde el servidor y lo usa en peticiones posteriores

Sesiones

- SID es enviado desde el servidor al cliente de varias formas:
  - Incluyéndolo como parámetro en un enlace
  - Valor de elemento de tipo “`hidden`” de un formulario
  - Como una cabecera HTTP `Set-Cookie`
- El cliente puede enviar el SID al servidor dependiendo de la forma que el servidor lo asigna

Sesiones con Cookies

- Ventajas
  - Dura hasta que la cookie expira
  - Solo se transmite el SID entre el cliente y servidor:
    - Es más rápido y seguro
- Desventaja
  - El cliente debe tener habilitado el uso de Cookies
  - Ojo con "*Cookieless*"
- En el servidor se debe generar un identificador único para cada sesión

Flask: Sesiones

- Adicional al objeto `request`, tenemos el objeto `session`
  - Permite almacenar información entre peticiones del usuario
  - Utiliza cookies con contenido firmado digitalmente
    - El cliente puede mirar las cookies pero no podrá modificarlas
  - Para firmar contenido es necesario definir una llave secreta
  - Podemos generar llaves con el siguiente comando:

```
$ python -c 'import secrets; print(secrets.token_hex())'
'192b9bdd22ab9ed4d12e236c78afcb9a393ec15f71bbf5dc987d54727823bcbf'
```

Flask: Sesiones

- Sesiones basadas en cookies
  - Flask serializará los valores que se agregan a la sesión y los agrega en el contenido de la cookie
  - En caso de no encontrar valores agregados a la sesión, se debe revisar el tamaño de la cookie
    - Puede que esté sobrepasando el tamaño soportado por el navegador

Browser	Cookie count limit per domain	Total size of cookies
Chrome	180	4096
Firefox	150	4097
Opera	60	4096
Safari	600	4093

Fuente: https://docs.devexpress.com/AspNet/11912/common-concepts/cookies-support

Flask: Sesiones

6. Implementation Considerations

6.1. Limits

Practical user agent implementations have limits on the number and size of cookies that they can store. General-use user agents SHOULD provide each of the following minimum capabilities:

- At least 4096 bytes per cookie (as measured by the sum of the length of the cookie's name, value, and attributes).
- At least 50 cookies per domain.
- At least 3000 cookies total.

Servers SHOULD use as few and as small cookies as possible to avoid reaching these implementation limits and to minimize network bandwidth due to the Cookie header being included in every request.

Servers SHOULD gracefully degrade if the user agent fails to return one or more cookies in the Cookie header because the user agent might evict any cookie at any time on orders from the user.

Fuente: https://www.ietf.org/rfc/rfc6265.txt

Flask: Sesiones

```python
from flask import Flask, session, request, render_template, redirect, url_for

app = Flask(__name__)

# Set the secret key to some random bytes. Keep this really secret!
app.secret_key = b'940b414bb69fb679d5cd7f071a5e2012e89a6319b0e298f5b9de2eeeafbdb69f'

@app.route('/')
def index():
    tareas = []
    if 'tareas' in session:
        tareas = session["tareas"].split("###")
    return render_template("tareas.html", tareas=tareas)

@app.route('/tareas', methods=['POST'])
def tareas():
    if "tareas" in session:
        session['tareas'] = session["tareas"] + "###" + request.form['tarea']
    else:
        session['tareas'] = request.form['tarea']

    return redirect(url_for('index'))

if __name__ == "__main__":
    app.run(debug=True)
```

Flask: Sesiones

- Podemos usar sesiones almacenando la información en el lado del servidor
  - Evitamos problemas con el tamaño de Cookie
  - Podemos usar archivos, base de datos u otro servicio de almacenamiento
- Podemos usar la extensión `Flask-Session`
  - Documentación e instalación:

    https://flask-session.readthedocs.io/en/latest/
  - Permite tener “server-side session”
  - Solo transmite Cookie que identifica la sesión
    - No transmite contenido que puede ir creciendo en el tiempo
    - No necesitamos definir un “secret key”

Flask: Flask-Session

```python
from flask import Flask, session, request, render_template, redirect, url_for
from flask_session import Session

app = Flask(__name__)
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

@app.route('/')
def index():
    tareas = []
    if 'tareas' in session:
        tareas = session["tareas"].split("###")
    return render_template("tareas.html", tareas=tareas)

@app.route('/tareas', methods=['POST'])
def tareas():
    if "tareas" in session:
        session['tareas'] = session["tareas"] + "###" + request.form['tarea']
    else:
        session['tareas'] = request.form['tarea']

    return redirect(url_for('index'))

if __name__ == "__main__":
    app.run(debug=True)
```

Sesiones con Cookies

- Autenticación de usuarios:
  - Podemos asignar un SID en la autenticación exitosa de usuarios
  - Cada script de la aplicación se debe revisar la existencia de la Cookie de sesión
  - Si no está presente, se debería re-direccionar a página de `login`
  - Si está presente, continuamos considerando al usuario autenticado
  - Con el SID tenemos una *llave* segura que enviamos al cliente y el cliente la retorna
    - En el servidor podemos tener un archivo o registro en base de datos asociado al SID