

# **PROYECTO DE SOFTWARE**

**Cursada 2021**

# TEMAS DE HOY

- Sesiones

**COOKIES**

# COOKIES

- Básicamente, son “tokens” en el requerimiento HTTP que permite identificar de alguna manera al cliente en el servidor.
- Se almacenan en el cliente.
- Muy usado por ser HTTP un protocolo sin estado.
- Formato: **nombreCookie=valor;expires=fecha;**
- En flask: Es un atributo del objeto request (diccionario)

# COOKIES EN FLASK

- Obteniendo cookies del requerimiento del cliente:

```
from flask import request

@app.route('/')
def index():
    contador = request.cookies.get('contador_visitas')
```

- Seteando cookies en el cliente:

```
from flask import make_response

@app.route('/')
def index():
    resp = make_response(render_template(...))
    resp.set_cookie('contador_visitas', '0')
    return resp
```

- Referencia de [uso de cookies en flask](#)
- Veamos un ejemplo aplicado en nuestra app <http://localhost:5000/>.



# USO DE COOKIES

Las cookies se utilizan principalmente con tres propósitos:

- **Gestión de Sesiones:** Inicios de sesión, carritos de compras, puntajes de juegos o cualquier otra cosa que el servidor deba recordar.
- **Personalización:** Preferencias de usuario, temas y otras configuraciones.
- **Rastreo:** Guardar y analizar el comportamiento del usuario.

## SEGURIDAD DE COOKIES: SECURE, HTTPONLY Y SAMESITE

- **Secure**: Una cookie segura sólo se envía al servidor con una petición cifrada sobre el protocolo HTTPS. Incluso con Secure, no debería almacenarse **NUNCA** información sensible en la cookies.
- **HttpOnly**: Para prevenir ataques cross-site scripting (XSS), las cookies HttpOnly son **inaccesibles desde la API de Javascript** Document.cookie; Solamente se envían al servidor.
- **SameSite**: Permiten a los servidores requerir que una cookie **no sea enviada con solicitudes cross-site**, lo que proporciona algo de protección contra ataques cross-site request forgery (CSRF).

# SESIONES

# MANEJO DE SESIONES TRADICIONAL

Es un mecanismo para conservar ciertos datos a lo largo de varios accesos.

- Permite registrar un número arbitrario de variables que se conservarán en las siguientes peticiones.
- Identificador: A cada visitante se le asigna un identificador único, llamado **session id** (identificador de sesión).
- Tradicionalmente (en lenguajes como PHP) la sesión es almacenada en un archivo en el servidor, el cliente SÓLO posee el sessionID para identificarla.
- Hay dos formas de propagar un identificador de sesión:
  - Mediante cookies
  - A través de la URL.

## SESIONES EN FLASK (CLIENT SIDE)

- Por defecto Flask usa sesiones basadas en cookies (**session cookie**).
- La información de sesión se almacena **en el cliente** en una cookie firmada con una **secret key**.
- Cualquier modificación a la cookie queda invalidada por su firma. Pero es **visible en todo momento** en el cliente.
- No es aconsejable guardar información sensible en una session cookie.

Veamos una de estas sesiones en [http://localhost:5000/iniciar\\_sesion](http://localhost:5000/iniciar_sesion) decodificadas con <https://github.com/noraj/flask-session-cookie-manager>

# SESIONES EN FLASK UTILIZANDO FLASK-SESSION (SERVER SIDE)

- Flask posee extrensiones como **Flask-Session** que permiten un mejor manejo de las sesiones.
- Con Flask-Session podemos elegir diferentes lugares donde almacenar la sesión en el servidor:
  - redis
  - memcached
  - **filesystem**
  - mongodb
  - sqlalchemy
- Se instala con pip: **pip install Flask-Session**.

# USO DE FLASK-SESSION

```
from flask_session import Session
# Configuración inicial de la app
app = Flask(__name__)
app.config.from_object(Config)
#Server Side session
app.config['SESSION_TYPE'] = 'filesystem'
Session(app)
```

- Modifiquemos la app y veamos de nuevo...  
**[http://localhost:5000/iniciar\\_sesion](http://localhost:5000/iniciar_sesion)**

## **MANEJO DE SESIONES SERVER-SIDE**

- Los archivos con los datos de la sesión se generan en el servidor
- El cliente sólo guarda el sessionid

# EJEMPLO DE USO DE SESIONES: NO SOLO PARA LOGIN

```
from flask import Flask, render_template, request, session
from flask_session import Session

app = Flask(__name__)

app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"

Session(app)

@app.route("/")
def index():
    if session.get("pos_equipos") is None:
        session["pos_equipos"] = { "A" : "Campeón", "B":
"Sub Campeón",
                                "C": "Semi finalista",
                                "D": "Semi finalista"}

    return render_template ("index.html")

@app.route("/equipos")
def equipos():
    return render_template("equipos.html",
contenido=session["pos_equipos"])

@app.route("/agregar", methods=["POST"])
def agregar():
    equipo = request.form.get("equipo")
```

```
    posicion = request.form.get("posicion")
    session["pos_equipos"][equipo] = posicion
    return render_template("equipos.html",
    contenido=session["pos_equipos"])
```

## REPASANDO: MANEJO DE SESIONES

- ¿Qué es una sesión?
- ¿Porqué son necesarias?
- ¿Qué son las cookies?
- ¿Para qué sirven las cookies?
- ¿Dónde se alojan las cookies?
- ¿Dónde se aloja la sesión?

## REFERENCIAS:

- Cookies en Flask:  
<https://flask.palletsprojects.com/en/2.0.x/quickstart/#cookies>
- HTTP Cookies:  
<https://developer.mozilla.org/es/docs/Web/HTTP/Cookies>
- Local Storage vs Session Storage vs Cookie:  
<https://krishankantsinghal.medium.com/local-storage-vs-session-storage-vs-cookie-22655ff75a8>
- Sesiones en Flask:  
<https://flask.palletsprojects.com/en/2.0.x/quickstart/#sessions>
- Artículo sesiones en flask: <https://overiq.com/flask-101/sessions-in-flask/>
- Documentación de Flask-Session: <https://flask-session.readthedocs.io/en/latest/>

**FIN**