**COOKIES - CARRITO**

**Fecha: 22 / 04 / 2020**

**Objeto: implementación de cookies**

**Framework: Flask**

**Autor: Manuel Semeco**

**Introducción**

En el presente documento se comentara a detalle la implementación de un carrito en el proyecto Minitienda usando cookies, provenientes de Flask, además del procedimiento de cómo se setea una cookie, como se llama y como se elimina, además se explica sus posibles uso en controladores para implementar una lógica funcional para el carrito de la tienda.

**Cookie-Flask:**

Es un fichero de texto que se define gracias al comando proveniente de un aplicación de servidor flask, es decir para que una cookie sea creada el servidor tiene que comandarlo a través de la respuesta o **response**. Las cookies son accesibles (una vez creadas) en el código del “back” o código de la aplicación de servidor con el uso del **request.cookies** para obtener las lista de cookies y otros elementos que van por defecto y para obtener una cookie especifica se usa la siguiente sentencia **request.cookies.get(key).**

**Procedimientos Básicos para las cookies en flask**

1. **CREAR COOKIES**

Para crear cookies se usa el método **Response.set\_cookies(key: string , value: string, expires: default(forever) )**, donde **Response** es la clase “respuesta” de un controlador todo lo que resulte valido para retornar en un controlador en flask adopta esta clase automáticamente, pero también se tiene la oportunidad de crear este tipo (**Response**) previamente en el código del controlador, con aplicación\_flask.make\_response(object: Optional(redirect, render\_template, str, bytes etc)) siendo el retorno de este método, un objeto de clase **Response,** de esta manera se puede usar sobre el objeto creado el método **set\_cookie.**

**Palabras claves: crear , setear , set\_cookie.**

**Caso de uso Nro.1 - Controlador – set cookie**

@app.route("/SetCookies")

def set\_cookies():

    art=Articulos.query.get(1)

    content=json.dumps([{"nombre":art.nombre, "precio":art.precio, "cantidad":8}])

    print(content)

    content=bytes(content,"UTF-8")

    response=app.make\_response(redirect(url\_for("view\_inicio")))

    response.set\_cookie("1",content)

    return response

**\_\_ comentario:** se usa ***json.dumps(object)***para que el objeto pase a formato string requisito para el value de cookie.

­\_\_ **comentario:** también acepta byestring (experimento) sigue siendo una cadena.

\_\_ **comentario:** se crea el objeto response con lo que se desea retornar desde el controlador , como se puede observar sea hace uso de **make\_response** sobre **redirect** y además y muy **IMPORTANTE** se usa el método **set\_cookie(key:string, value: string)** como un método con seteo inplace este retorna None si se pretende igualar a la etiqueta response por lo que se hace en una línea aparte.

**Caso de uso Nro.2 – controlador del enlace comprar**

@app.route("/add/articulo/<int:id>", methods=["POST","GET"])

@login\_required

def comprar(id):

... #(código omitido)

else:

                content=bytes(json.dumps(articulo\_c), "UTF-8")

                print("creando cookie")

                response=app.make\_response(redirect(url\_for("view\_inicio")))

                response.set\_cookie(f"{current\_user.id}",content)

                return response

**Comentario:** se crea el contenido de la cookie**,** con json.dumps (lo de bytes es experimental e igual funciona), luego se crea el objeto response con una redirección a la página de inicio. Y finalmente se setea la cookie con el id del usuario que tiene por valor los el articulo:dict en una lista. Ver **código línea 535**.

1. **ACTUALIZAR COOKIES**

Para actualizar una cookie básicamente consiste en volverla a **setear(set\_cookie)** con las modificaciones en Value es decir si se tiene una lista de cookies (request.cookies) y se desea cambiar la cookie de llave str(11) entonces se debe llamar esa cookie usando json.loads(cookie: str) agregando o modificando lo que se debe modificar y luego setearlo bajo la misma **key** usando **Response.set\_cookie(“11”, json.dumps(datos\_modificados)).**

**Palabras claves:**actualizar,setear.

**Caso de uso Nro.1 – Sobrescribiendo la cookie de usuario**

# cookie lista del usuario en str.

            cookie=request.cookies.get(str(current\_user.id)

try:

                cookie=json.loads(cookie)

            except:

                cookie=None

#… código omitido

# se sustituye el articulo repetido

                else:

                    print(f"se sobreescribe cookie:{articulo\_c}")

                    cookie.pop(indice)

                    cookie.insert(indice,articulo\_c[0])

                    art\_json=json.dumps(cookie)

                    print(f"sobrescribiendo con : {art\_json}")

                    redirectx=app.make\_response(redirect(url\_for("view\_inicio")))

                    redirectx.set\_cookie(str(current\_user.id),art\_json)

                    print(f"el response {redirectx}")

                    return redirectx

#codigo omitido

\_\_ **Comentarios**: se elimina el articulo/cookie del objeto cookies que coincide en una key ( “nombre” en este caso )de la cookie entrante (un artículo nuevamente definido en el carrito).

­­\_\_ **Comentarios**: esto es un objeto lista que se crea con una lista que contiene un diccionario con el artículo enviado para actualizar o crear las cookies del usuario logueado. Ver línea 489.

1. **Eliminar cookie.**

Para eliminar una cookie consiste en usar el argumento **expires**  de **set\_cookie,**  el cual básicamente seteado a int: 0 hace que la cookie indicada por el key argumento quede expirada de inmediato, un simil a eliminarla. La forma seria como sigue **Response.set\_cookie(key,value=””, expires=0). Pero** en el caso particular de Minitienda se usó la “actualización” para eliminar sencillamente se pide a cookie del usuario y se busca el articulo y se elimina y luego se vuelve a cargar la cookie.

**Palabras claves:**actualizar,setear.

**Caso de uso Nro.1 – controlador para eliminar un artículo del carrito**

@app.route("/delete/carrito/<int:id>", methods=["GET","POST"])

@login\_required

def carro\_delete(id):

    try:

        # se intenta obtener el objeto de la cookie del usuario, una lista de juegos.

        cookie=json.loads(request.cookies.get(str(current\_user.id)))

        # se recorre la lista de juego

        for juego in cookie:

            # ... Y se pregunta si el id del juego actual es el id del juego que se quiere eliminar.

            if juego.get("id")==id:

                # se pregunta el indice del juego en la lista.

                indice=cookie.index(juego)

                # se usa el indice del juego para eliminarlo con pop.

                cookie.pop(indice)

        # se crea la respuesta

        respuesta= redirect(url\_for("carro"))

        # se setean las nuevas cookies resultado del proceso anterior, se usa json.dumps para hacerlo

        # una cadena

        respuesta.set\_cookie(str(current\_user.id), json.dumps(cookie))

        return respuesta

    except:

        return redirect(url\_for("carro"))

**Caso de uso Nro.2- ejemplo de flask**

@app.route('/carrito\_delete/<id>')

@login\_required

def carrito\_delete(id):

try:

datos = json.loads(request.cookies.get(str(current\_user.id)))

except:

datos = []

new\_datos=[]

for dato in datos:

if dato["id"]!=id:

new\_datos.append(dato)

resp = make\_response(redirect(url\_for('carrito')))

resp.set\_cookie(str(current\_user.id),json.dumps(new\_datos))

return resp

**Comentario:** Se llama la cookie del usuario y se crea el objeto. Luego se itera y se pregunta si el id del articulo a eliminar coincide con un artículo en la cookies del usuario si es el caso se omite y si no es el caso se agrega en una nueva lista. Terminada la iteración se agrega en la cookie del usuario.

**Caso de uso Nro.3 – Pedido realizado: eliminar cookies y descontar inventario en base de datos.**

@app.route("/inicio/FinalizarPedido/", methods=["POST","GET"])

@login\_required

def pedido():

    try:

        # lista de juegos del carrito del usuario.

        print(1)

        articulos\_carrito= json.loads(request.cookies.get(str(current\_user.id)))

        #descuento de los juegos del carrito de los articulos en la pagina(base de datos)

        print(2)

        for articulo\_carrito in articulos\_carrito:

            print("hee")

            Articulos.query.get(articulo\_carrito["id"]).stock-=articulo\_carrito["cantidad"]

            db.session.commit()

        print(3)

        respuesta= app.make\_response(render\_template("pedido.html", precio=total))

        respuesta.set\_cookie(str(current\_user.id),"",expires=0)

        return respuesta

    except:

        abort(404, "No hay pedido")

\_\_ **Comentario:** se pide la lista de artículos en el carrito

\_\_ **Comentario:** se descuenta (acumulativamente por iteración por cada artículo) todos los artículos del inventario (tabla de artículos stock).

\_\_ **Comentario:** se elimina directamente la cookie haciéndola expirar de inmediato.