
.UBAfiuba
FACULTAD DE INGENIERÍA

Introducción al Desarrollo de Software I - Lanzillotta

Capra - Chaves - Di Matteo - Sosa - Villegas - Palavecino

.UBAfiuba
FACULTAD DE INGENIERÍA

Agenda

- Organizar un proyecto Flask (repaso)
- Modificación de urls
- Reutilización de código con block
- Creación de un formulario GET-POST
- Enviar un listado y utilizarlo en el template
- Error handler
- Uso del Inspector en los navegadores

Organizar un proyecto en flask

Del diseño a la realidad

- Todo proyecto web comienza con un relevamiento de requerimientos con los usuarios del futuro sistema.
- Posteriormente se puede solicitar a un diseñador gráfico o a especialista en diseño que haga realidad el diseño del sitio web.
- Muchas veces se puede acceder ya sea en forma gratuita o comprándolo a un template en HTML
- Nosotros como programadores podemos importar el diseño a nuestro proyecto.

Repaso de la estructura de archivos

/mi_proyecto

app.py

/.venv

/static

style.css

/templates

layout.html

index.html

login.html

...

Jinja nos permite agregar funcionalidad a estos templates, haciendo que puedan integrar data desde el template dinámicamente.

Creación de rutas

- Dentro del archivo app.py

```
@app.route("/about")
def about():
    return render_template('about.html')

@app.route("/contact")
def contact():
    return render_template('contact.html')

@app.route("/gallery")
def gallery():
    return render_template('gallery.html')
```

Modificación de las url :: Links

- Toda referencia a links dentro del sitio o cualquier redirección se deberá crear mediante:

  url_for('user', name='john', _external=True)

equivale a:

http://localhost:5000/user/john

_external=True
Crea la dirección absoluta, ideal cuando necesitamos enviar un link por mail por ejemplo.

Modificación de las url :: archivos estáticos

- Toda referencia a recursos estáticos como ser archivos javascript, imágenes, CSS, etc se deberá crear mediante:

  url_for('static', filename='css/styles.css', _external=True)

equivale a:

http://localhost:5000/static/css/styles.css

El primer parámetro static
Hará referencia a la carpeta static y luego filename, informará la carpeta y el archivo particular a acceder.

Block :: Reutilización de código HTML

- Todo sitio web posee partes de código HTML (header, footer, scripts) que se repiten en cada una de las páginas.

  Para esto se puede utilizar los bloques **{% block header %} {% endblock %}**

- Para la utilización de los bloques es necesario crear un archivo base.html, ubicarlo dentro de la carpeta template.

  Este archivo tendrá todo lo que consideremos como código que se pueda repetir, y dentro de este se colocarán los bloques para ser editables a posteriori.

Creación de un formulario

- Los formularios dentro de un sitio web son fundamentales para poder interactuar con el usuario.
- Si bien Flask posee una dependencia que agiliza y permite crear formularios más robustos y seguros, en la materia vamos a utilizar los métodos tradicionales de etiquetas <form> y métodos POST para enviar la información.

Definición de un form en HTML

- Supongamos que queremos crear un formulario de Login

action => indicará el *form-handler*, en general es el script del lado del servidor

```
1  <form action="{{ url_for('example2')}}" method="post">
2      <label for="firstname">First Name:</label>
3      <input type="text" id="firstname" name="fname" placeholder="firstname">
4      <label for="lastname">Last Name:</label>
5      <input type="text" id="lastname" name="lname" placeholder="lastname">
6      </br>
7      <button type="submit">Login</button>
8  </form>
```

Utilización de ids para identificar los campos

button submit => Define el botón que enviará la información al form-handler del lado del servidor.

Definición de form-handler (lado servidor)

Debemos definir el tipo de método que recibirá la ruta, en este caso GET(por defecto) + POST (para el envío de la información)

```
19  @app.route('/example3', methods =["GET", "POST"])
20  def example3():
21      if request.method == "POST":
22          nombre = request.form.get("fname")
23          return redirect(url_for("show",nombre=nombre))
24
25      return render_template("example3.html")
```

Si recibimos información con el método POST, entonces redirigimos a un sitio de respuesta, sino volvemos a mostrar el form

Ejemplo

Crear un ejemplo que utilice en formulario de contacto del sitio del bar Brooklyn y en caso de contestar ser exitoso, abrir una nueva página con el mensaje:

“Su mensaje de contacto ha sido enviado exitosamente”.

Uso de ciclos

- Como hemos visto en clases anteriores, podemos utilizar un ciclo for embebido dentro del código HTML que **reciba información del servidor y la pueda desplegar en el front** una porción de código repetida tantas veces como se desee

```
<ul>
    {% for comment in comments %}
        <li>{{ comment }}</li>
    {% endfor %}
</ul>
```

Ejemplo

Utilizar el proyecto bar Brooklyn y refactorizar el código del menú, creando uno que liste las 4 columnas con información hardcodeada proveniente del servidor.

Errorhandler

- Flask posee un manejador de errores que pueden capturar errores que sucedan en el servidor y permitirnos crear una vista que lo muestre de una forma adecuada a la estética de nuestro proyecto web.

```
30  @app.errorhandler(404)
31  def page_not_found(e):
32      return render_template('404.html'), 404
```

- El template 404.html deberá ser creado por el programador.

Ejemplo

Crear utilizando nuestro proyecto de bar Brooklyn errorhandler para error 404 y 500.

- El error 404, deberá desplegar el mensaje “La ruta a la que quiere acceder no existe. Haga click en Home para volver

- El error 500, deberá desplegar un mensaje “Se ha producido un error en el servidor. Haga click en Home para volver

Uso del inspector

- Los navegadores, poseen una funcionalidad “Inspector” que es súmamente útil para los desarrolladores, con la misma se podrá al DOM y ver el estado del Front.

https://developer.chrome.com/docs/devtools/dom?hl=es-419

Ejercitación

Ejercicio

Descargar el proyecto .zip que les enviaremos por slack.

a- Crear una estructura base.html que permita reutilizar código mediante bock

b- Reemplazar las rutas, utilizando url_for()

c- Crear un formulario que al ser cargado utilizando método POST, envié un mensaje de éxito

d- Usar el método GET para pasar un parámetro por url con el nombre de un usuario y darle la bienvenida al sitio

e- Crear los errorhandler que crea necesarios.