

Clase 6: Desarrollo de aplicaciones web en Python

Módulo 2: Programación Avanzada en Python.

Apoyado por:

CORFO

Clase de hoy

01

Introducción a Flask

Creación de aplicación básica, rutas y vistas.

02

Templating y formularios

Crear contenido dinámico y controlar flujo de formularios.



6A

Introducción a Flask

Bloque A

Qué veremos en Bloque A

- Introducción a Flask
- Rutas y vistas
- HTML renderizado

Introducción a Flask

¿Qué es Flask?

- Flask es un **microframework** web en Python que permite crear aplicaciones web de manera rápida y sencilla.
- Es adecuado para proyectos pequeños y medianos, aunque también es escalable.
- Proporciona gran flexibilidad y permite añadir funcionalidades adicionales mediante extensiones.
- Características clave:
 - Ligero y modular: es minimalista y da al desarrollador la libertad de estructurar la aplicación.
 - Soporte para templates con Jinja2: facilita la generación de contenido dinámico en HTML.

Instalación y configuración básica de Flask

- **Instalación de Flask:**

```
pip install Flask
```

- **Estructura básica de un proyecto Flask:**

- a. Archivo principal (`app.py`) que define la aplicación y rutas.
- b. Directorio `templates/` para almacenar las plantillas HTML.
- c. Directorio `static/` para archivos estáticos como CSS y JavaScript.

```
mi_aplicacion_flask/  
├── app.py  
├── templates/  
│   └── index.html  
├── static/  
│   └── style.css
```

Creación de una aplicación básica

Ejemplo código inicial (app.py)

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return "¡Bienvenido a mi aplicación Flask!"

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

Rutas y vistas en Flask

- **Rutas:** son las URLs que Flask asocia con funciones de vista. Se definen con el decorador `@app.route('/ruta')`.
- **Vistas:** son las respuestas que se envían al navegador (texto o HTML renderizado).

Ejemplo:

```
@app.route('/usuario/<nombre>')  
def usuario(nombre):  
    return f"¡Bienvenido, {nombre}!"
```

¡Pueden incluir parámetros dinámicos!

HTML renderizado en Flask

Flask permite **renderizar HTML** directamente, y acoplarlo al retorno de una vista. Estos son conocidos como “templates” o “plantillas”.

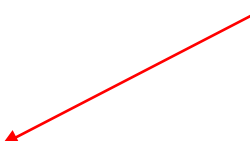
Ejemplo:

```
from flask import Flask, render_template
#...

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

#...
```

Por defecto, las plantillas
deberán estar en el directorio
templates.



HTML renderizado en Flask

Además, podemos pasar **valores** a los templates.

Ejemplo:

```
from flask import Flask, render_template
#...

@app.route('/')
def home():
    return render_template('home.html', nombre='Juan')

#...
```

HTML renderizado en Flask

Así se vería el home.html con valores pasados desde la vista:

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Home</title>
</head>
<body>
  <h1>Hola</h1>
  <p>Bienvenido {{ nombre }} a nuestro sitio.</p>
</body>
</html>
```

HTML renderizado en Flask

- `url_for()`: Construye rutas dinámicas basadas en funciones de vista, útil para mantener enlaces actualizados si las rutas cambian.
- `redirect()`: Redirige al usuario a otra URL, ideal para mostrar una página diferente tras una acción (como enviar un formulario o iniciar sesión).

```
from flask import Flask, redirect, url_for
#...
@app.route('/')
def home():
    return "Página de Inicio"

@app.route('/redireccionar')
def redireccionar():
    # Redirige a la página de inicio usando `url_for`
    return redirect(url_for('home'))
```

Query String en Flask

Es la parte de la URL que permite enviar parámetros opcionales en solicitudes HTTP GET.

```
GET /perfil?nombre=Juan
```

```
@app.route('/perfil')
```

```
def perfil():
```

```
    nombre = request.args.get('nombre', 'invitado')
```

```
    return f"Hola, {nombre}!"
```



Trabajo grupal – Bloque A

Paralelo 2

G1	G2	G3	G4
Nicolas Mardones	Víctor Meza Herrera	Daniela Méndez Gándara	Claudia Blanco
Manuel Denis	Estefania Manriquez	Álvaro Pérez	Ariel Inostroza
Bryan Castillo	Patricio Vera	Pedro Nahum	Héctor Aguayo
GERALDY SUAREZ	Oscar Torres	Javier Gajardo	Ruben Sanhueza Ramirez
Scarlett Espinoza	Braulio Quiroz	Angela Proboste Neira	Félix González
Ulises Campodónico	Yerko Gallardo	Nicolás Guzmán	Ariel Mora
Carol Leiva	Rodrigo Araya		
G5	G6	G7	G8
Fabian Díaz	Camila Oyarzún	Mayerlyn Rodriguez	Daniela Porto
Natalia Rivera	Stefanya Pulgar	Sebastian Vega	Cristian Chavez Jara
Juan Salinas	Carlos Emilio Azócar Riquelme	Efrain Duarte Campos	Juan Rodrigo Vega
Rodrigo Pastén Cortés	Nicolas Rojas	Bianel Bianchini	Rodolfo Cantillana
Flavio Jara R.	luis.paillan.cnc@gmail.com	Bastián Gamboa Labbé	Abraham Ruiz
Daniel García	Cristóbal Gajardo	Pablo Uribe	Rodrigo Álvarez

Trabajo grupal – Ejercicio #1

1. Configuración inicial y rutas básicas

Objetivo: Configurar la estructura básica de la aplicación Flask y crear rutas principales con HTML **estático para crear un sitio web de su grupo.**

1. Instale Flask.
2. Cree una aplicación Flask con las rutas iniciales:
 1. `/`: Página de inicio con una bienvenida al sitio informativo.
 2. `/about`: Página "Acerca de", que contenga una breve descripción del grupo y los nombres de los integrantes.
3. Cree un archivo `app.py` y una carpeta `templates/` con templates HTML para cada ruta (`index.html` y `about.html`).

Trabajo grupal – Ejercicio #2

2. Uso de templates y datos dinámicos

Objetivo: renderizar contenido **dinámico** en los templates del sitio web.

1. Modificar la ruta /about para pasar datos dinámicos a about.html
 - a. Nombre del grupo
 - b. Breve descripción de su objetivo en el curso.
 - c. Lista de integrantes
2. *Extra:* mostrar la fecha y hora actualizadas cada vez que se cargue la página usando Python.



Break!



6B

Templating y formularios

Bloque B

Qué veremos en Bloque B

- Introducción a Jinja2
- Uso de condicionales, ciclos y filtros en los templates
- Métodos HTTP comunes
- Formularios

Introducción a Jinja2

- **Jinja2** es el motor de templates de Flask que permite generar HTML dinámico utilizando una sintaxis similar a Python.
- Facilita el uso de **variables**, **estructuras de control** (bucles, condicionales) y **filtros** para manipular el contenido en las páginas web.

Sintaxis básica:

- **Uso de variables:** `<p>¡Hola, {{ nombre }}!</p>`
- **Comentarios:** Los comentarios se escriben entre `{# ... #}`, y no se muestran en el HTML resultante. `{# Este es un comentario en Jinja2 #}`

Control de Flujo en Templates

Uso de Condicionales y Bucles en Jinja2

- Los templates en Jinja2 (HTML) permiten condicionales (`if`) y bucles (`for`).
- Se escriben entre `{% ... %}`.

Ejemplo: Mostrar una lista de integrantes en el template, mostrando el número de iteración.

```
<ul>
  {% for integrante in integrantes %}
    <li>{{ loop.index }}: {{ integrante }}</li>
  {% endfor %}
</ul>
```

Control de Flujo en Templates

Uso de Filtros en Templates con Jinja2

Los filtros se aplican con el símbolo `|` y permiten manipular variables dentro de los templates.

- **capitalize**: Capitaliza la primera letra.

```
<p>{{ nombre | capitalize }}</p>
```

- **lower** y **upper**: convierte el texto a minúscula o mayúscula

```
<p>{{ mensaje | upper }}</p>
```

- **default**: Muestra un valor predeterminado si la variable es None.

```
<p>{{ nombre | default('Invitado') }}</p>
```

Métodos HTTP

Métodos comunes de HTTP

Los métodos HTTP son instrucciones que definen cómo el servidor y el cliente (e.g. navegador) deben comunicarse.

1. **GET:** Para solicitar datos del servidor, se envían a través de la URL (visible para usuario).
2. **POST:** Para enviar o modificar datos, se envían en el cuerpo de la solicitud, no en URL.

Métodos HTTP en Flask

Métodos comunes de HTTP en rutas

Ejemplo de ruta con GET y POST en archivo `app.py`:

```
from flask import request

@app.route('/saludo', methods=['GET', 'POST'])
def saludo():
    if request.method == 'POST':
        return "Esto fue enviado por método POST"
    if request.method == 'GET':
        return "Esto fue enviado por método GET"
```

Formularios en Flask

Creación de Formularios HTML

- Flask permite usar formularios HTML para recolectar datos de los usuarios.
- Configurar el formulario con métodos HTTP (GET o POST).

```
<form action="/contacto" method="post">
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre">

  <label for="mensaje">Mensaje:</label>
  <textarea id="mensaje" name="mensaje"></textarea>

  <button type="submit">Enviar</button>
</form>
```

Formularios en Flask

Procesamiento de Formularios en Flask

Los datos enviados desde un formulario pueden capturarse usando `request.form`.

```
from flask import request, render_template

@app.route('/contacto', methods=['GET', 'POST'])
def contacto():
    if request.method == 'POST':
        nombre = request.form['nombre']
        mensaje = request.form['mensaje']
        return f"Gracias por tu mensaje, {nombre}."
    return render_template('contacto.html')
```

Validación de Formularios

Validar Datos en formulario

Ejemplo de Validación: Verificar que los campos nombre y mensaje no estén vacíos

```
@app.route('/contacto', methods=['GET', 'POST'])
def contacto():
    error = None
    if request.method == 'POST':
        nombre = request.form['nombre']
        mensaje = request.form['mensaje']
        if not nombre or not mensaje:
            error = "Por favor, completa todos los campos."
        else:
            return f"Gracias por tu mensaje, {nombre}."
    return render_template('contacto.html', error=error)
```

Validación de Formularios

Uso de mensajes Flash

- Los mensajes flash permiten enviar retroalimentación temporal a los usuarios, como confirmaciones o errores.
- Se usan para proporcionar mensajes breves que desaparecen después de la próxima solicitud.
- Para configurar mensajes Flash en Flask, es necesario definir una clave secreta de la aplicación:

```
app.secret_key = 'mi_clave_secreta'
```

Validación de Formularios

Ejemplo mensaje Flash en la Vista

```
from flask import flash, redirect, url_for

@app.route('/contacto', methods=['POST'])
def contacto():
    nombre = request.form['nombre']
    if not nombre:
        flash("Por favor, ingresa tu nombre.")
    else:
        # hacer algo con nombre
        flash("Gracias por tu mensaje.")
    return redirect(url_for('contacto'))
```

Validación de Formularios

Mostrando mensajes Flash en el Template

Los mensajes flash se recuperan en el template usando `get_flashed_messages()`

```
{% with messages = get_flashed_messages() %}
  {% if messages %}
    <ul>
      {% for message in messages %}
        <li>{{ message }}</li>
      {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
```



Trabajo grupal – Bloque B

Paralelo 2

G1	G2	G3	G4
Nicolas Mardones	Víctor Meza Herrera	Daniela Méndez Gándara	Claudia Blanco
Manuel Denis	Estefania Manriquez	Álvaro Pérez	Ariel Inostroza
Bryan Castillo	Patricio Vera	Pedro Nahum	Héctor Aguayo
GERALDY SUAREZ	Oscar Torres	Javier Gajardo	Ruben Sanhueza Ramirez
Scarlett Espinoza	Braulio Quiroz	Angela Proboste Neira	Félix González
Ulises Campodónico	Yerko Gallardo	Nicolás Guzmán	Ariel Mora
Carol Leiva	Rodrigo Araya		
G5	G6	G7	G8
Fabian Díaz	Camila Oyarzún	Mayerlyn Rodriguez	Daniela Porto
Natalia Rivera	Stefanya Pulgar	Sebastian Vega	Cristian Chavez Jara
Juan Salinas	Carlos Emilio Azócar Riquelme	Efrain Duarte Campos	Juan Rodrigo Vega
Rodrigo Pastén Cortés	Nicolas Rojas	Bianel Bianchini	Rodolfo Cantillana
Flavio Jara R.	luis.paillan.cnc@gmail.com	Bastián Gamboa Labbé	Abraham Ruiz
Daniel García	Cristóbal Gajardo	Pablo Uribe	Rodrigo Álvarez

Trabajo grupal – Ejercicio #1

1. Creando un Perfil de integrante del grupo con Parámetros en el Query String

Objetivo: Practicar el uso de `query string` en Flask para personalizar la página de perfil de un integrante del grupo.

1. Crea una ruta `/perfil` que reciba los siguientes parámetros opcionales en el query string; nombre, edad, profesión.
2. Si un parámetro no se proporciona en el query string, establece un valor predeterminado.
3. Muestra una respuesta que salude al usuario y describa su perfil con la información proporcionada. Use un template para mostrar los datos formateados.

Trabajo grupal – Ejercicio #2

2. Publicación de Anuncios para el Grupo

Objetivo: Crear una funcionalidad en el sitio web del grupo que permita publicar y visualizar anuncios importantes.

1. **Ruta de publicación de Anuncio (/anuncio).** La ruta /anuncio debe aceptar dos métodos; GET: Muestra un formulario para agregar un anuncio y POST: Procesa el formulario y guarda el anuncio en una lista en memoria. El formulario debe tener los siguientes campos; título (obligatorio), contenido (obligatorio), prioridad (alta, media, baja) con valor predeterminado media.
2. **Página de Anuncios (/anuncios):** La ruta /anuncios muestra todos los anuncios publicados. Usa **ciclos** en el template para mostrar los anuncios en forma de lista.
 1. Los anuncios de **prioridad Alta** deben resaltarse visualmente (por ejemplo, con un color diferente).
 2. Si no hay anuncios publicados, muestra un mensaje indicándolo.

¿Preguntas?

¡Hemos llegado al final de la clase!

