Last Updated : 30 Dec, 2022

Flask is a lightweight WSGI framework that is built on Python programming. WSGI simply means Web Server Gateway Interface. Flask is widely used as a backend to develop a fully-fledged Website. And to make a sure website, templating is very important. Flask is supported by inbuilt template support named Jinja2. Jinja2 is one of the most used Web template engines for Python. This Web template engine is a fast, expressive, extensible templating engine. Jinja2 extensively helps to write Python code within the HTML file. Further, it also includes:

- Async support for generating templates that automatically handle sync and async functions without extra syntax.
- Template inheritance and inclusion.
- The Template engine makes debugging easier.
- Support of both High-level and Low-level API support.

Install the required package

To install the Jinja2 package in Python, check your latest pip version and stay updated. Install Jinja2 using the following command:

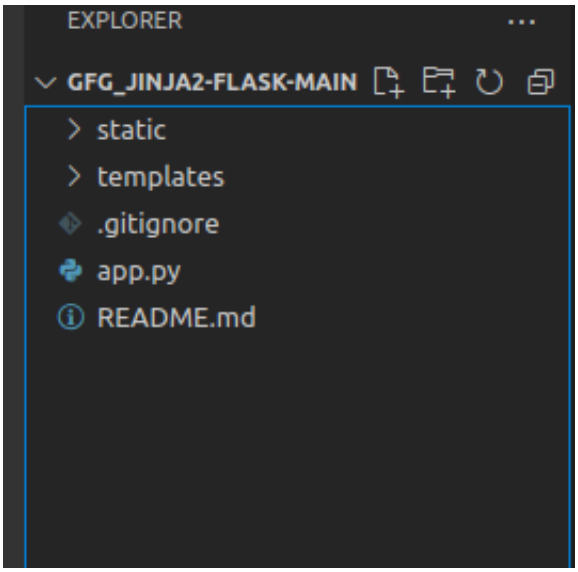
```
pip install Jinja2
```

But since we are dealing with the Templating with Jinja2 in Flask, there is no need to separately install Jinja2. When you install the Flask framework, the Jinja2 comes installed with it.

```
pip install flask
```

Templating with Jinja2 in Flask

Before we proceed with the coding part, this is how our project directory should look like:



Main Python File

Here is the common app.py file that interfaces with all the HTML files.

Python3

```
from flask import Flask, render_template, redirect, url_for

app = Flask(__name__)


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


@app.route("/default")
def default():
    return render_template("layout.html")


@app.route("/variable")
def var():
    user = "Geeksforgeeks"
    return render_template("variable_example.html", name=user)


@app.route("/if")
def ifelse():
```

```
    user = "Practice GeeksforGeeks"
    return render_template("if_example.html", name=user)


@app.route("/for")
def for_loop():
    list_of_courses = ['Java', 'Python', 'C++', 'MATLAB']
    return render_template("for_example.html", courses=list_of_courses)


@app.route("/choice/<pick>")
def choice(pick):
    if pick == 'variable':
        return redirect(url_for('var'))
    if pick == 'if':
        return redirect(url_for('ifelse'))
    if pick == 'for':
        return redirect(url_for('for_loop'))


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

Jinja Template Variables

To declare the variable using Jinja Template we use **{{variable_name}}** within the HTML file. As a result, the variable will be displayed on the Website.

Syntax of Jinja Template Variables

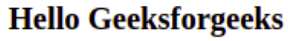
```
{{any_variable_name}}
```

variable_example.html

HTML

```
<html>
    <head>
        <title>Variable Example</title>
    </head>
    <body>
        <h3>Hello {{name}}</h3>
    </body>
</html>
```

Output



Jinja Template if Statements

Just like declaring the variable in the Jinja2 template, if conditions have almost similar syntax. But here we specify the beginning and end of the if block.

Syntax of Jinja Template if Statements

```
{% if conditions %}
 ...
 ...
 ...
{% endif %}
```

Our app.py will stay the same. Just only one change instead of Geeksforgeeks tries giving Geeks so that we can verify the else block.

if_example.html

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>If example</title>
    </head>
    <body>
        {% if(name == "Geeksforgeeks") %}
            <h3> Welcome </h3>
```

```
        {% else %}
            <h3> Unknown name entered: {{name}} </h3>
        {% endif %}
    </body>
</html>
```

Output:

Unknown name entered: Practice GeeksforGeeks

Jinja Template for Loop

Jinja for loop syntax is similar to the if statements, the only difference is for loop requires sequence to loop through.

Syntax of Jinja Template for Loops

```
{% for variable_name in sequence%}
 ...
 ...
 ...
{% endfor %}
```

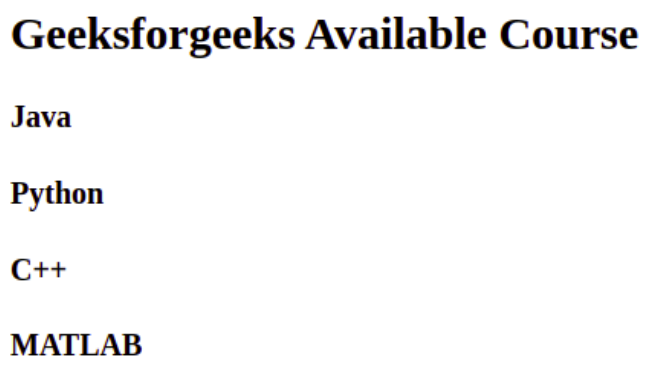
for_example.html

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>For Example</title>
    </head>
```

```
    <body>
        <h2> Geeksforgeeks Available Course </h2>
        {% for course in  courses%}
          <h4> {{course}} </h4>
        {% endfor %}
    </body>
</html>
```

Output:



Jinja Template Inheritance

If you closely check the project files, you will find the index.html and layout.html. In this example, we gonna take look into Template Inheritance. In most of the websites, if you notice, the footer and header remain the same, which means they share similar formats. In this example, layout.html will contain the default design that is common to all the pages, but here we will keep it specifically for index.html to understand how it works.

The syntax for layout.html contains the default text, along with the block contain, that will be inherited by other HTML files. You can think of layout.html as the parent and index.html as a child.

Syntax of Jinja Template Inheritance

layout.html

```
{% block content %}
{% endblock %}
```

index.html

```
{% extends "layout.html" %}
    {% block content %}
        ....
{% endblock %}
```

Example

In **layout.html** we define the top block and specify a template to insert block content that acts as parent HTML files.

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Jinja2 and Flask</title>
    </head>
    <body>
        <h1>Welcome to Geeksforgeeks</h1>
        <h4>A Computer Science portal for geeks.</h4>
        {% block content %}
        {% endblock %}
    </body>
</html>
```

In **index.html** using the layout.html as the parent file, we shall derive all its content and add the block content to the existing HTML file.

Note: No need to define the HTML, head, and body tag in the child HTML file.

HTML

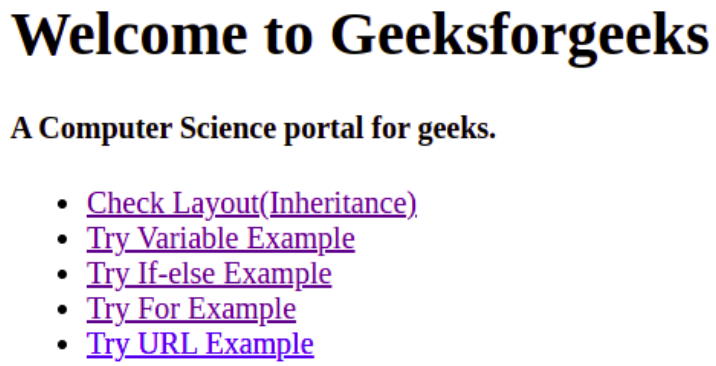
```
{% extends "layout.html" %}
        {% block content %}
        <ul>
            <li><a href="default"> Check Layout(Inheritance) </a></li>
            <li><a href="/variable"> Try Variable Example </a></li>
            <li><a href="/if"> Try If-else Example </a></li>
            <li><a href="/for"> Try For Example </a></li>
            <li><a href="/url"> Try URL Example </a></li>
        </ul>
```

```
{% endblock %}
```

Output:

layout.html

index.html



Jinja Template url_for Function

To build a dynamic website you need multiple re-direction within the website. url_for function is a very handy method that helps in re-direction from one page to another. url_for is also used to link HTML templates with static CSS or JavaScript files.

In our example since we have multiple choice for example, i.e., variable, if and for. Using url_for, we can create a custom function in which the user can alter the URL to get the specific result. For example, we shall define a function inside app.py and in example 2 we will take link HTML with CSS.

Syntax of Jinja Template url_for Function

```
url_for(function_name)
```

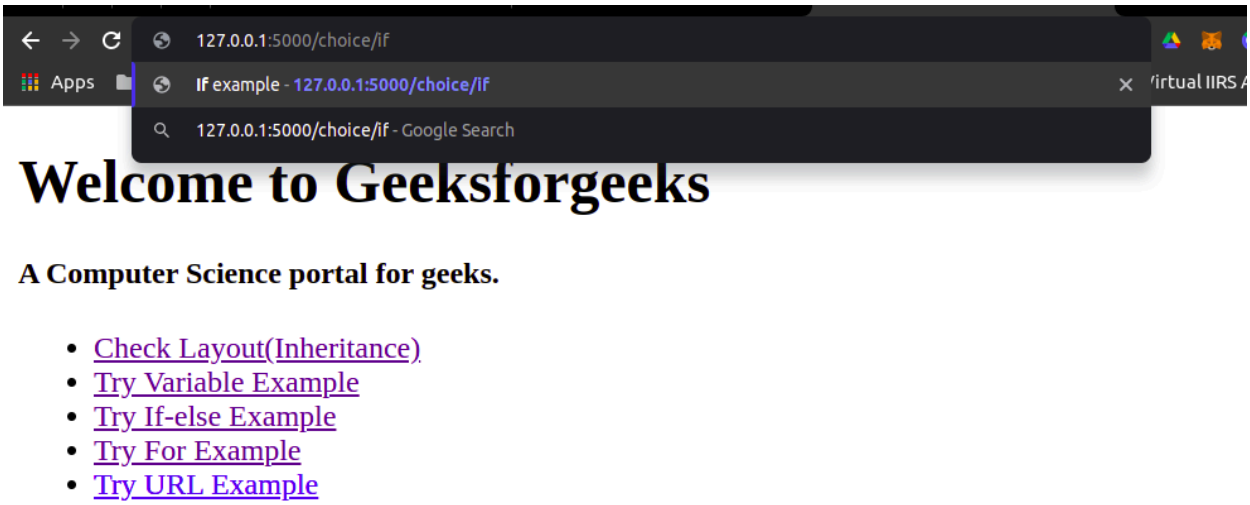
Example 1:

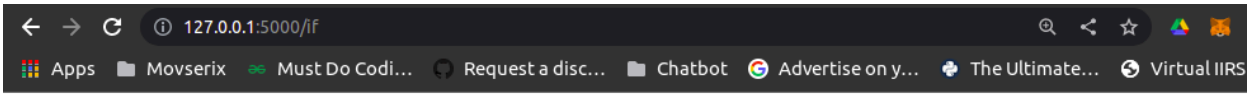
In the below example, if the user enters choice/<his choice> then it will redirect to that HTML file. Make sure redirect and url_for are imported.

Python3

```
@app.route("/choice/<pick>")
def choice(pick):
    if pick == 'variable':
        return redirect(url_for('var'))
    if pick == 'if':
        return redirect(url_for('ifelse'))
    if pick == 'for':
        return redirect(url_for('for_loop'))
```

Output:





Unknown name entered: Practice GeeksforGeeks

Example 2:

In example 1 we used url_for inside a Python file. Now we shall use url_for inside the layout.html (parent file) HTML file, it will follow the variable define syntax i.e., to be enclosed within {{}}. Just like templates, create a static file for CSS.

```
{{ url_for('static', filename='<path of the file>') }}
```

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Template with Jinja2 and Flask</title>
        <link rel="stylesheet" type="text/css" href="{{ url_for('static', filenam
    </head>
    <body>
        <h1>Welcome to Geeksforgeeks</h1>
        <h4>A Computer Science portal for geeks.</h4>
        {% block content %}
        {% endblock %}
    </body>
</html>
```

Output

Next Article

Unit Testing Jinja Templates with Flask

Similar Reads

Placeholders in jinja2 Template - Python

Web pages use HTML for the things that users see or interact with. But how do we show things from an external source or a controlling programming language...

5 min read

Python Falcon - Jinja2 Template

Python Falcon is a lightweight and minimalist web framework designed for building web APIs, with a particular emphasis on simplicity, speed, and efficienc...

6 min read

Documenting Flask Endpoint using Flask-Autodoc

Documentation of endpoints is an essential task in web development and being able to apply it in different frameworks is always a utility. This article discusses...

4 min read

How to use Flask-Session in Python Flask ?

Flask Session - Flask-Session is an extension for Flask that supports Server-side Session to your application.The Session is the time between the client logs in to...

4 min read

How to Integrate Flask-Admin and Flask-Login

In order to merge the admin and login pages, we can utilize a short form or any other login method that only requires the username and password. This is know...

8 min read

Minify HTML in Flask using Flask-Minify

Flask offers HTML rendering as output, it is usually desired that the output HTML should be concise and it serves the purpose as well. In this article, we would...

12 min read

Flask URL Helper Function - Flask url_for()

In this article, we are going to learn about the flask url_for() function of the flask URL helper in Python. Flask is a straightforward, speedy, scalable library, used f...

11 min read

Best Practices for Jinja Templating

Jinja templating is a powerful tool for generating dynamic content in Python web applications. In this article, we will discuss some best practices related to Jinja...

4 min read

Comparing Jinja to Other Templating Engines

When it comes to web development, templating engines play a crucial role in separating the logic from the presentation layer of an application. They allow...

5 min read

Python Flask - ImmutableMultiDict

MultiDict is a sub-class of Dictionary that can contain multiple values for the same key, unlike normal Dictionaries. It is used because some form elements ha...

2 min read

Article Tags : Python Technical Scripter Technical Scripter 2022

Practice Tags : python

System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar
Commerce
World GK

GeeksforGeeks Videos

DSA
Python
Java
C++
Web Development
Data Science
CS Subjects