



Get the value of a checkbox in Flask

Last Updated : 02 Aug, 2024

The checkbox is a typical widget with selected options within a list by a user. They are employed in web forms whereby they are applied in the selection of preferences, settings, or a survey. Here in this article, we will see how to add the checkboxes in the [Flask](#) application right from the installation of Flask to the handling of inputs.

Get the value of a checkbox in Flask

Here is the most basic requirement before we start doing the checkboxes, we require a flask application. Flask is a framework for creating web applications based on [Python](#), it is micro, which is why it doesn't contain a lot of components, but it provides the features needed to start an application. Follow these steps to set up your Flask environment:

Follow these steps to set up your Flask environment:

Step 1: Install Flask

If you haven't already, [install Flask](#) using pip. Open your terminal or command prompt and run:

```
pip install Flask
```

Step 2: Create a Flask Application

Create a new directory for your project and navigate to it. Inside this directory,

[Flask Templates](#) [Jinja2](#) [Flask-REST API](#) [Python SQLAlchemy](#) [Flask Bcrypt](#) [Flask Cookies](#) [Json](#) [Postman](#)

application.

Step 3: Create a Template Directory

Create a folder named templates in the same directory as app.py. Inside the templates folder, create a file named index.html.

Creating Checkboxes

In the **index.html** file, we will create a form with checkboxes. Each checkbox will allow the user to select an option, and when the form is submitted, the selected options will be sent to the server.

Code Implementation

Here is the complete example code for the Flask application with checkboxes:

index.html

In this example, we have created a form with three checkboxes and a submit button. Each checkbox has a unique id and value attribute. The name attribute for all checkboxes is the same (checkbox), which allows us to handle them as a group on the server side.

HTML



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
6     <title>Checkbox Example</title>
7 </head>
8 <body>
9     <form action="/submit" method="post">
10         <label for="option1">Option 1</label>
11         <input type="checkbox" id="option1" name="checkbox"
    value="Option 1"><br>
12         <label for="option2">Option 2</label>
13         <input type="checkbox" id="option2" name="checkbox"
    value="Option 2"><br>
14         <label for="option3">Option 3</label>
15         <input type="checkbox" id="option3" name="checkbox"
    value="Option 3"><br>
16         <input type="submit" value="Submit">
```

```
        </form>
18    </body>
19    </html>
```

app.py

Python



```
1  from flask import Flask, render_template, request
2
3  app = Flask(__name__)
4
5  @app.route('/')
6  def index():
7      return render_template('index.html')
8
9  @app.route('/submit', methods=['POST'])
10 def submit():
11     selected_options = request.form.getlist('checkbox')
12     return f'Selected options: {", ".join(selected_options)}'
13
14 if __name__ == '__main__':
15     app.run(debug=True)
```

Output

Best Practices to ensure your Application File

When working with checkboxes in Flask, consider the following best practices to ensure your application is robust and user-friendly:

Use Descriptive Labels: Ensure that each checkbox has a clear and descriptive label. This improves the user experience and accessibility, making it easier for users to understand the options available to them.

Validate User Input: Always validate the user input on the server side to ensure that the data received is as expected. This prevents malicious input and

ensures data integrity. For example, you can check if the selected options are within the allowed set of values:

Handle Multiple Checkboxes: Use `request.form.getlist('checkbox')` to handle multiple checkboxes with the same name attribute. This method returns an array of all selected options, which is convenient on selection of multiple options.

Maintain State: If you need to preserve the state of the checkboxes (e.g., when a form submission fails and needs to be corrected), consider passing the selected values back to the template and pre-selecting the checkboxes based on these values. Here's an example of how you can achieve this:

Use CSRF Protection: When working with form submissions user should take Care against Cross-Site Request Forgery attack. Flask-WTF is an extension that assists Flask applications to add CSRF protection. To use it, install Flask-WTF and configure it in your application:

```
pip install Flask-WTF
```

FAQs

1. How do I pre-select checkboxes based on existing data?

To pre-select checkboxes based on existing data, pass the selected values to the template and use conditional statements in your HTML to check the relevant checkboxes. Here's an example:

In `app.py`:

Python

```
1 @app.route('/')
2 def index():
3     selected_options = ['Option 1', 'Option 3'] # Example
      selected options
4     return render_template('index.html',
      selected_options=selected_options)
```

In index.html:

HTML



```
1 <input type="checkbox" id="option1" name="checkbox"
  value="Option 1" {% if 'Option 1' in selected_options
    %}checked{% endif %}>
2 <input type="checkbox" id="option2" name="checkbox"
  value="Option 2" {% if 'Option 2' in selected_options
    %}checked{% endif %}>
3 <input type="checkbox" id="option3" name="checkbox"
  value="Option 3" {% if 'Option 3' in selected_options
    %}checked{% endif %}>
```

2. How can I style checkboxes?

You can style checkboxes using CSS. Here's an example:

CSS



```
1 input[type="checkbox"] {
2     width: 20px;
3     height: 20px;
4     accent-color: #007bff; /* Change the color of the
   checkbox */
5 }
```

Add the CSS to a `<style>` tag in your HTML file or to a separate CSS file linked in your HTML.

3. How can I handle dynamic checkbox lists?

If you need to generate checkboxes dynamically (e.g., from a database), pass the list of options to the template and loop through them in your HTML. Here's an example:

In app.py:

Python



```
1 @app.route('/')
2 def index():
3     options = ['Option 1', 'Option 2', 'Option 3', 'Option
4         4']
5     return render_template('index.html', options=options)
```

index.html:

Python



```
1 <form action="/submit" method="post">
2     {% for option in options %}
3         <label for="{{ option }}">{{ option }}</label>
4         <input type="checkbox" id="{{ option }}"
5         name="checkbox" value="{{ option }}"><br>
6     {% endfor %}
7     <input type="submit" value="Submit">
8 </form>
```

Looking to dive into the world of programming or sharpen your Python skills? Our [Master Python: Complete Beginner to Advanced Course](#) is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to advanced techniques. With **hands-on projects**, real-world examples, and expert guidance, you'll gain the confidence to tackle complex **coding challenges**. Whether you're starting from scratch or aiming to enhance your skills, this course is the perfect fit. Enroll now and master Python, the language of the future!

A adity...



Previous Article

Python | Checkbox widget in Kivy

Next Article

Similar Reads

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

Flask HTTP methods, handle GET & POST requests

In this article, we are going to learn about how to handle GET and POST requests of the flask HTTP methods in Python. HTTP Protocol is necessary for data...

6 min read

GET Request Query Parameters with Flask

In this article, we will learn how we can use the request object in a flask to GET Request Query Parameters with Flask that is passed to your routes using Pytho...

4 min read

How to get data from 'ImmutableMultiDict' in flask

In this article, we will see how to get data from ImmutableMultiDict in the flask. It is a type of Dictionary in which a single key can have different values. It is used...

2 min read

Get the Data Received in a Flask request

In this article, we will learn how we can use the request object in a flask to Get the Data Received that is passed to your routes. and How To Process Get Reque...

6 min read

How to Get Data from API in Python Flask

In modern web development, APIs (Application Programming Interfaces) play a crucial role in enabling the interaction between different software systems. Flas...

2 min read

Article Tags : [Python](#) [Python Flask](#) [Flask Projects](#)

Practice Tags : [python](#)



Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305



Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Interview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation

[OOAD](#)[Puzzles](#)[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](#)

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved