



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

Build a Flask application to validate CAPTCHA

Last Updated : 26 May, 2022

In this article, we are going to build a web application that can have a CAPTCHA. CAPTCHA is a tool you can use to differentiate between real users and automated users. CAPTCHAs are usually found on the login pages or on payment pages.

What is CAPTCHA?

Captcha is a strategy used to ensure sites against spam. Objective is to prevent intuitive sites from being spammed by sifting through naturally created input. Abbreviation CAPTCHA means 'Totally Automated Public Turing test to distinguish Computers and Humans'. Captchas are generally utilized when web applications require client input. Envision you are running an online store and need to offer your clients a chance to compose item surveys in the remarks segment. For this situation, you need to guarantee that passages are really from your clients or if nothing else from human site guests. You will regularly go over naturally produced spam remarks – in most pessimistic scenarios connecting to your opposition.

Note: For more information refer to [What is CAPTCHA code?](#)

Requirements

- **[Flask framework](#):** Flask is an API of Python that allows us to build up web-applications. Flask's framework is more explicit than Django's framework and is also easier to learn because it has less base code to implement a simple web-Application.
- **Flask-session-captcha library:** A captcha implementation for flask using flask-sessionstore and captcha packages. Each captcha challenge answer is saved in the server-side session of the challenged client.
- **[MongoDB](#):** MongoDB, the most popular NoSQL database, is an open-source document-oriented database. The term 'NoSQL' means 'non-relational'. It

means that MongoDB isn't based on the table-like relational database structure but provides an altogether different mechanism for storage and retrieval of data.

Note: The database depends on your application. Flask-session-captcha library supports multiple databases so if you are using any other database be sure to check out their documentation.

Build the application

You can install all the required dependencies using the following command

```
pip install -U Flask flask-session-captcha Flask-Sessionstore pymongo
```

It is recommended that you install all the dependencies in the virtual environment.

Stepwise Implementation

Step 1: Create the UI

Create a folder called templates and inside it create an HTML file. You can save the HTML file as form.html.

HTML

```
<form method="post">

    <!-- The following line creates the captcha -->
    {{ captcha() }}
    <input type="text" name="captcha">
    <input type="submit" name="submit">
</form>
```

Step 2: Create app.py

Let's create a simple app that shows the above form without captcha.

Example: Initialize Flask Application

Python3

```
from flask import Flask, render_template

app = Flask(__name__)

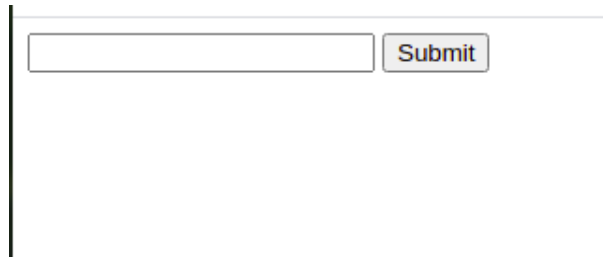
@app.route('/')
def index():
    return render_template('form.html')

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

Run the server using the following command to make sure that the application is running successfully and the form.html page is displayed.

```
python app.py
```

Output:

A screenshot of a web browser displaying a simple form. The form consists of a single text input field followed by a button labeled "Submit". The input field is empty, and the button is a light gray with a thin border. The form is centered on the page, and there is a vertical line to its left.

Step 3: Initialize PyMongoClient

To use Python in MongoDB, we are going to import PyMongo. From that, MongoClient can be imported which is used to create a client to the database.

Python3

```
from flask import Flask, render_template
from pymongo import MongoClient

app = Flask(__name__)
```

```
# Database Config
# If your mongodb runs on a different port
# change 27017 to that port number
mongoClient = MongoClient('localhost', 27017)

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

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

Step 4: Configure Captcha

There are various configurations required to use the captcha in the application. Here we will be using flask_session_captcha module which implements captcha using flask-sessionstore and captcha packages.

Python3

```
import uuid
from flask import Flask, render_template
from flask_sessionstore import Session
from flask_session_captcha import FlaskSessionCaptcha
from pymongo import MongoClient

app = Flask(__name__)

# Database Config
# If your mongodb runs on a different port
# change 27017 to that port number
mongoClient = MongoClient('localhost', 27017)

# Captcha Configuration
app.config["SECRET_KEY"] = uuid.uuid4()
app.config['CAPTCHA_ENABLE'] = True

# Set 5 as character length in captcha
app.config['CAPTCHA_LENGTH'] = 5

# Set the captcha height and width
app.config['CAPTCHA_WIDTH'] = 160
app.config['CAPTCHA_HEIGHT'] = 60
app.config['SESSION_MONGODB'] = mongoClient
```

```
app.config['SESSION_TYPE'] = 'mongodb'

# Enables server session
Session(app)

# Initialize FlaskSessionCaptcha
captcha = FlaskSessionCaptcha(app)

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

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

Step 5: Configure Logging

Logging is a means of tracking events that happen when some software runs. Here we have used the `logging.getLogger()` method which returns a logger with a specified name otherwise returns the root logger,

Python3

```
import uuid
import logging
from flask import Flask, render_template
from flask_sessionstore import Session
from flask_session_captcha import FlaskSessionCaptcha
from pymongo import MongoClient

app = Flask(__name__)

# Database Config
# If your mongodb runs on a different port
# change 27017 to that port number
mongoClient = MongoClient('localhost', 27017)

# Captcha Configuration
app.config["SECRET_KEY"] = uuid.uuid4()
app.config['CAPTCHA_ENABLE'] = True

# Set 5 as character length in captcha
app.config['CAPTCHA_LENGTH'] = 5
```

```

# Set the captcha height and width
app.config['CAPTCHA_WIDTH'] = 160
app.config['CAPTCHA_HEIGHT'] = 60
app.config['SESSION_MONGODB'] = MongoClient
app.config['SESSION_TYPE'] = 'mongodb'

# Enables server session
Session(app)

# Initialize FlaskSessionCaptcha
captcha = FlaskSessionCaptcha(app)

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

if __name__ == "__main__":
    app.debug = True
    logging.getLogger().setLevel("DEBUG")
    app.run()

```

Step 6: Code the index route

Here we have used the POST method which returns checks the input against the captcha. If the captcha matches success method is displayed otherwise fail method is displayed.

Python3

```

import uuid
import logging
from flask import Flask, request, render_template
from flask_sessionstore import Session
from flask_session_captcha import FlaskSessionCaptcha
from pymongo import MongoClient

app = Flask(__name__)

# Database Config
# If your mongodb runs on a different port
# change 27017 to that port number
mongoClient = MongoClient('localhost', 27017)

```

```
# Captcha Configuration
app.config["SECRET_KEY"] = uuid.uuid4()
app.config['CAPTCHA_ENABLE'] = True

# Set 5 as character length in captcha
app.config['CAPTCHA_LENGTH'] = 5

# Set the captcha height and width
app.config['CAPTCHA_WIDTH'] = 160
app.config['CAPTCHA_HEIGHT'] = 60
app.config['SESSION_MONGODB'] = MongoClient
app.config['SESSION_TYPE'] = 'mongodb'

# Enables server session
Session(app)

# Initialize FlaskSessionCaptcha
captcha = FlaskSessionCaptcha(app)

@app.route('/', methods=['POST', 'GET'])
def index():
    if request.method == "POST":
        if captcha.validate():
            return "success"
        else:
            return "fail"

    return render_template("form.html")

if __name__ == "__main__":
    app.debug = True
    logging.getLogger().setLevel("DEBUG")
    app.run()
```

Run the app

Start server with:

```
python app.py
```

Then visit:

```
http://localhost:5000/
```

Output:



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!



gupta...



9

Previous Article

[Todo list app using Flask | Python](#)

Next Article

[How to write a simple Flask API for hello world?](#)

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

Build an Application to Search Installed Application using Python

In this article, we are going to write python scripts to search for an installed application on Windows and bind it with the GUI application. We are using...

6 min read

Generate Captcha Using Python

In this article, we are going to see how to generate a captcha using Python package captcha to generate our own CAPTCHA (Completely Automated Public...

2 min read

How to Build a Simple Android App with Flask Backend?

Flask is an API of Python that allows us to build up web applications. It was developed by Armin Ronacher. Flask's framework is more explicit than Django's...

8 min read

Python | Build a REST API using Flask

Prerequisite: Introduction to Rest API REST stands for REpresentational State Transfer and is an architectural style used in modern web development. It define...

3 min read

Build a Text Translator Web App using Flask and Azure Cognitive Services

We're going to create a website that will translate text into multiple languages using Artificial Intelligence(AI). We'll use Flask framework for the Front end and...

7 min read

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

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](#)

Languages

[Python](#)
[Java](#)
[C++](#)
[PHP](#)
[GoLang](#)
[SQL](#)
[R Language](#)
[Android Tutorial](#)
[Tutorials Archive](#)

Data Science & ML

[Data Science With Python](#)

[Algorithms](#)[DSA for Beginners](#)[Basic DSA Problems](#)[DSA Roadmap](#)[Top 100 DSA Interview Problems](#)[DSA Roadmap by Sandeep Jain](#)[All Cheat Sheets](#)[Data Science For Beginner](#)[Machine Learning](#)[ML Maths](#)[Data Visualisation](#)[Pandas](#)[NumPy](#)[NLP](#)[Deep Learning](#)

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](#)[OOAD](#)[System Design Bootcamp](#)[Interview Questions](#)

School Subjects

[Mathematics](#)[Physics](#)[Chemistry](#)[Biology](#)[Social Science](#)[English Grammar](#)[Commerce](#)[World GK](#)

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](#)[Puzzles](#)

GeeksforGeeks Videos

[DSA](#)[Python](#)[Java](#)[C++](#)[Web Development](#)[Data Science](#)[CS Subjects](#)

@GeeksforGeeks, Sanchhaya Education Private Limited, All rights reserved