Password Hashing with Bcrypt in Flask

Last Updated : 21 Mar, 2023

In this article, we will use Password Hashing with Bcrypt in Flask using Python. Password **hashing** is the process of converting a plaintext password into a hashed or encrypted format that cannot be easily reverse-engineered to reveal the original password. Bcrypt is a popular hashing algorithm used to hash passwords. It is a password-hashing function that is based on the Blowfish cipher and is designed to be slow and computationally expensive, making it more difficult for attackers to guess or crack passwords.

Key Terminologies:

- **Password Hashing:** The process of converting a plaintext password into a hashed or encrypted format.
- **Bcrypt:** A password-hashing function based on the Blowfish cipher.
- **Salt:** Random data that is used as additional input to a one-way function that hashes a password or passphrase.
- **Hashing Algorithm:** A mathematical function that converts a plaintext password into a fixed-length hash value.
- **Iterations:** The number of times a password is hashed using the bcrypt algorithm.

Stepwise Implement with Bcrypt in Flask

Step 1: Install Flask-Bcrypt

To use Bcrypt in Flask, we need to install the Flask-Bcrypt extension. We can install it using pip.

```
pip install flask-bcrypt
```

Step 2: Import Flask-Bcrypt

We need to import the Bcrypt module from Flask-Bcrypt in our Flask app.

Python3

```
from flask_bcrypt import Bcrypt
```

Step 3: Create a Bcrypt Object

We need to create a Bcrypt object and pass our Flask app as an argument.

Python3

```
bcrypt = Bcrypt(app)
```

Step 4: Hash a Password

We need to decode the hashed password using Python decode('utf-8') as the generate_password_hash() function returns a bytes object. We can hash a password using the **generate_password_hash()** function of the Bcrypt object.

Python3

```
hashed_password = bcrypt.generate_password_hash
                ('password').decode('utf-8')
```

Step 5: Verify a Password

The check_password_hash() function returns True if the password matches the hashed password, otherwise, it returns False. We can verify a password using the **check_password_hash()** function of the Bcrypt object.

Python3

```
is_valid = bcrypt.check_password_hash(hashed_password, 'password')
```

Complete Code

Here is an example of how to implement Bcrypt in a Flask app.

Python3

```
from flask import Flask
from flask_bcrypt import Bcrypt

app = Flask(__name__)
bcrypt = Bcrypt(app)

@app.route('/')
def index():
    password = 'password'
    hashed_password = bcrypt.generate_password_hash
                            (password).decode('utf-8')
    is_valid = bcrypt.check_password_hash
                            (hashed_password, password)
    return f"Password: {password}<br>Hashed Password:
                          {hashed_password}<br>Is Valid: {is_valid}"

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

Output:

When we run the Flask app, we will see the following output.

Output

Related Articles:

- Hashing Passwords in Python with BCrypt
- How To Hash Passwords In Python

Previous Article

Add User and Display Current Username in Flask

Next Article

How to store username and password in Flask

Similar Reads

Hashing Passwords in Python with BCrypt

In this article, we will see how to hash passwords in Python with BCrypt. Storing passwords in plain text is a bad practice as it is vulnerable to various hacking...

4 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

How to store username and password in Flask

We'll discuss how to save a Username and password on a Flask website in this article. We may view the welcome message and the username we chose when...

9 min read

Implementation of Hashing with Chaining in Python

Hashing is a data structure that is used to store a large amount of data, which can be accessed in O(1) time by operations such as search, insert and delete. Variou...

3 min read

Compare two files using Hashing in Python

In this article, we would be creating a program that would determine, whether the two files provided to it are the same or not. By the same means that their...

3 min read

Full domain Hashing with variable Hash size in Python

A cryptographic hash function is a special class of hash function that has certain properties which make it suitable for use in cryptography. It is a mathematical...

5 min read

Article Tags : Python 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