Python Programming

Nov 3, 2017

" Da Vinci Code World "
A Information Encryption Blog System based on Python

Eve Thullen, Claremont Graduate University & Harvey Mudd College

1. Project Background:

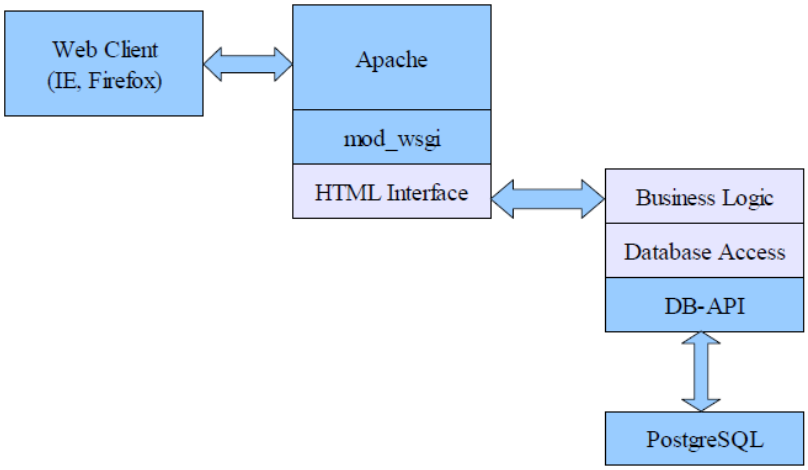
Let's called this project Da Vinci Code World. It's website that the user could post text message, that the user wants to hidden real message behind those text for different reasons, such as avoiding web censorship and monitoring system, or the user want to deliver important message to specific people. In a world, it's a website for specific people share secrets.

2. Project Description:

1. A website written by python and connected to database, that the user can register an account and login.
2. User can post text or articles. Those text is encrypted, and the same time an encryption key been created for each post. The algorithm written by python
3. Readers can add comments to the article.
4. Readers can require key to decrypt the message.

3. Project Platform and Structure:

1. Python code to connect Apache and database Postgres together
2. Python and Pesto to generate HTML to display to the user



- Project Language:

Python
HTML
SQL

- Platform and Frame:

Database: PostgreSQL 9.6.5
Server: Apache 2
Python: 2.7

- Python Library to Use:

mod_wsgi (Python based Web Application Run on Apache)
Pesto (HTML interface)
pgdb (database interface)

4. Project Installation:

- Platform and Frame Installation:

1. **PostgreSQL 9.6.5** installation: https://www.postgresql.org/download/macosx/

Set up pg_config path: /users/yingfenhuang/PostgreSQL/pg96/bin

```
$ export PG_HOME=/users/yingfenhuang/PostgreSQL/pg96
$ export PATH=$PATH:$PG_HOME/bin
```

,

2. **Apache 2:** MacOS built in Apache 2

3. **Python 2.7.14**, https://www.python.org/downloads/

- Python Library Installation:

1. mod_wsgi 4.5.20

The mod_wsgi package provides an Apache module that implements a WSGI compliant interface for hosting Python based web applications on top of the Apache web server.

1) Install mod_wsgi to python: pip install mod_wsgi
or download and install to Python: https://pypi.python.org/pypi/mod_wsgi
python setup.py install

2) Run the test.py: rom terminal cd into the test.py folder

```
mod_wsgi-express start-server
```

or

```
mod_wsgi-express start-server test.py
```

or

```
mod_wsgi-express start-server test.py --port 8080
```

3) Open the web:

http://localhost:8000/

2. pesto/ Python 2.7

1) Install pesto: compile with Python 2.7

```
$ pip install pesto
```

or download and install to Python:

```
python setup.py install
```

https://pypi.python.org/pypi/pesto/25

2) Run pesto

```
$ python test_pesto.py
```

3) Open the web:

http://localhost:8000/

* test_pesto.py

```
from pesto import to_wsgi, Response
from wsgiref import simple_server
def handler(request):
    return Response([
        "<html>",
        “<body><h1>Hello World!</h1></body>",
        "</html>",
    ])
if __name__ == "__main__":
    httpd = simple_server.make_server('', 8000, to_wsgi(handler))
```

3. Festo

1) Install fresco

$ pip install fresco gunicorn

2) Run fresco

$ gunicorn Test_fresco:app

3) Open the web:

http://localhost:8000/

* test_fresco.py

```
from fresco import FrescoApp, GET, Response
def helloworld():
    return Response(["<h1>Hello World!</h1>"])
app = FrescoApp()
app.route('/', GET, helloworld)
```

4. pgdb

1) Install PyGreSQL, need install PostgreSQL first, then set up pg_config path: /users/yingfenhuang/PostgreSQL/pg96/bin

$ export PG_HOME=/users/yingfenhuang/PostgreSQL/pg96

$ export PATH=$PATH:$PG_HOME/bin

Install PyGreSQL

$ pip install PyGreSQL

* test_pgdb.py

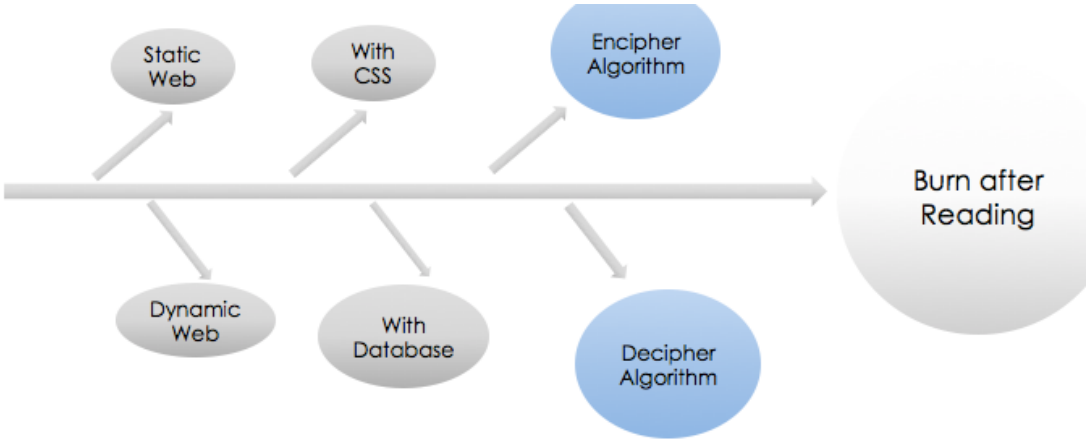
```
import os
os.environ['PATH'] += ";/users/yingfenhuang/PostgreSQL/pg96/bin"
import pgdb
#Connect to database
try:
    connection = pgdb.connect(user="postgres", password="testpgdb",
database="postgres")
    cursor = connection.cursor()
except:
```

```
print "\n failed to connect to Database"
print "\n Exception:", sys.exc_value
```

Eve Thullen
IST 431 Python Programming
Dec 16, 2017

5. Project Process

- Initial vision and its evolution



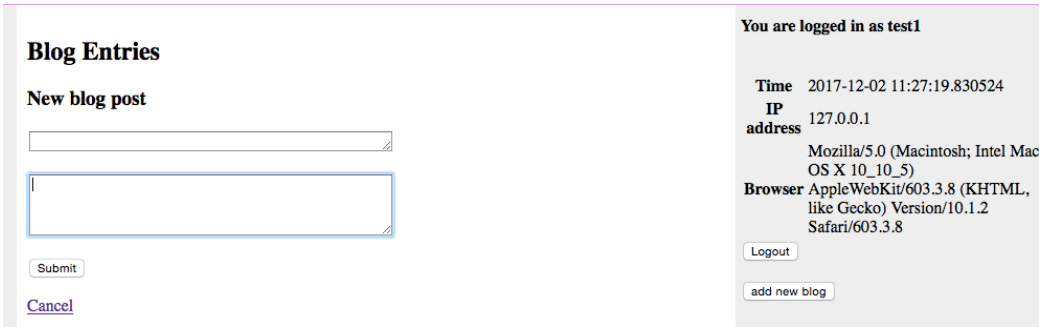
- The progress and the final system's capabilities

Version 0: Built in Static Web Page with html in local host

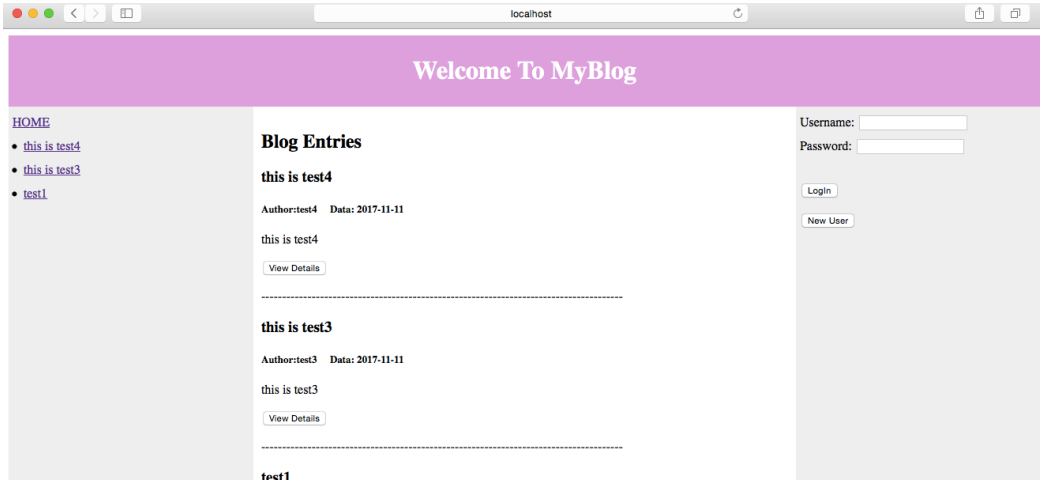
```
import test

if __name__ == "__main__":
        from wsgiref import simple_server
        httpd = simple_server.make_server('', 8000, test.application)
        httpd.serve_forever()
```

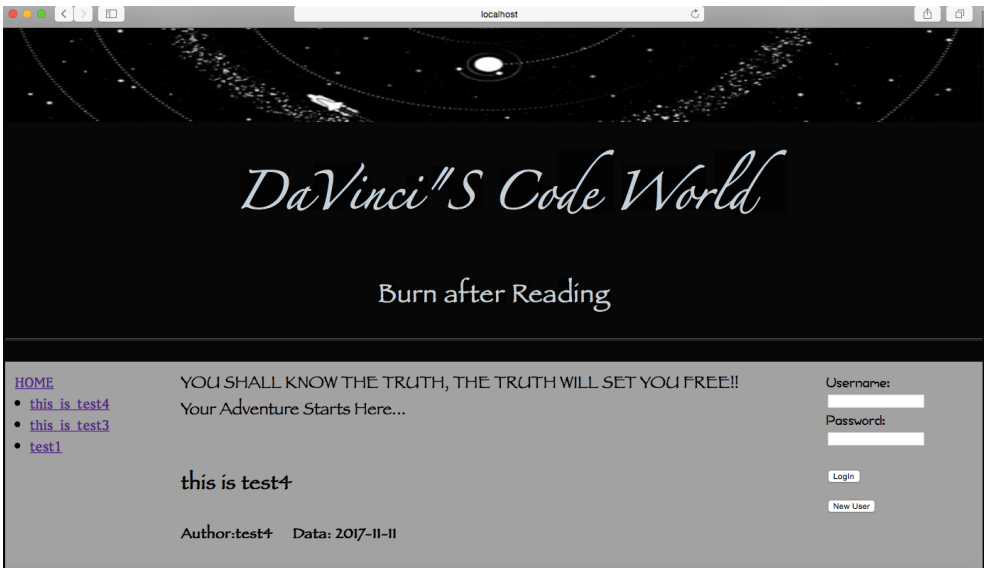
Version 1: Built in Dynamic Web Page with html



Version 2: Built in Dynamic Web Page with Database



Version 3: Built in Dynamic Web Page with CSS



Version 4: Built in Cryptographic Dynamic Web Page

System capabilities:

- User Register and Error Validation
- Front Page Posts and Page navigation
- Add New and Delete Post
- Add Comments to the Post
- Encipher Post and Decipher Post

Tackle the project differently:

One of the most important part of this the project is about encipher and decipher part. I probably use python Cryptographic library to encrypt the text. However, I designed my own encipher algorithm to encrypt the text for this project, just for security reason.

Future Version:

Version 5: Burn after Reading

- The user could set up the time for the post, that their post will disappear itself at specific time.
- Other language version, such as Chinese Version