Flask

web development,
one drop at a time

Microframework for Python

Hammad Ahmad and Emily Carden

What is it?

"Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine."

What is it?

"Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine."

Buzzwords

- **"Micro web framework":**
  - Micro: designed for simple projects with few tasks
  - Web framework: supports web site and web application development. Makes the developer's life easier.
- **"Werkzeug toolkit"**: a utility library for Python, which means it is a an interface between web servers and web applications. Werkzeug is what takes the python code and makes it into a website.
- **"Jinja2"**: a templating language for Python. Flask uses it to format their web pages. HTML and PHP are other templating languages.

What does Flask look like?

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

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

Import Flask

Name of the app's module

The URL that will trigger the function

Message to display

```
emilycardensmbp:~ emilycarden$ python3 hello.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

MiniTwit

- A small-scale, Flask and SQLite powered Twitter clone
- Features:
  - "Tweeting"
  - User timelines, Public timelines
  - Finding other users
  - Following users
- https://github.com/pallets/flask/tree/master/examples/minitwit

What does the code look like?

```
@app.route('/')
def timeline():
    """Shows a users timeline or if no user is logged in it will
    redirect to the public timeline.  This timeline shows the user's
    messages as well as all the messages of followed users.
    """
    if not g.user:
        return redirect(url_for('public_timeline'))
    return render_template('timeline.html', messages=query_db('''
        select message.*, user.* from message, user
        where message.author_id = user.user_id and (
            user.user_id = ? or
            user.user_id in (select whom_id from follower
                                    where who_id = ?))
        order by message.pub_date desc limit ?''',
        [session['user_id'], session['user_id'], PER_PAGE]))


@app.route('/public')
def public_timeline():
    """Displays the latest messages of all users."""
    return render_template('timeline.html', messages=query_db('''
        select message.*, user.* from message, user
        where message.author_id = user.user_id
        order by message.pub_date desc limit ?''', [PER_PAGE]))
```

Testing

- MiniTwit uses pytest for testing
- pytest is a Python framework for testing
  - Supports unit testing
  - Supports over 150 external plugins
  - Unit testing very similar to JUnit
    - Provides details about failed assert statements

Testing

```python
def test_login_logout(client):
    """Make sure logging in and logging out works"""
    rv = register_and_login(client, 'user1', 'default')
    assert b'You were logged in' in rv.data
    rv = logout(client)
    assert b'You were logged out' in rv.data
    rv = login(client, 'user1', 'wrongpassword')
    assert b'Invalid password' in rv.data
    rv = login(client, 'user2', 'wrongpassword')
    assert b'Invalid username' in rv.data
```

pytest

```java
@Test
public void testFreeSpace() {
    Disk tmp = new Disk();

    assertEquals(GIGABYTE, tmp.freeSpace());
    tmp.add(GIGABYTE / 2);
    assertEquals(GIGABYTE / 2, tmp.freeSpace());
    tmp.add(GIGABYTE / 2);
    assertEquals(0, tmp.freeSpace());
}
```

JUnit

Debugging

Without debugger (default)

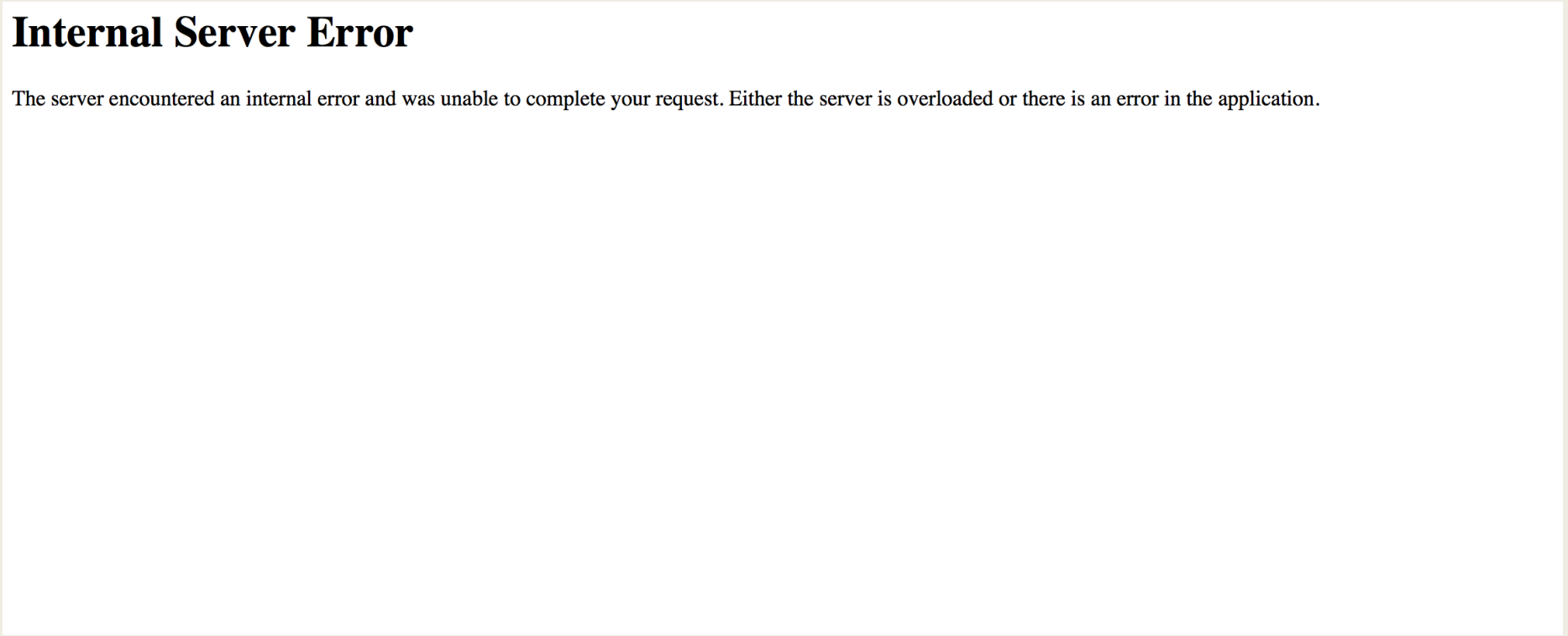
```
Hammads-MacBook-Pro:minitwit hammad$ export FLASK_APP=minitwit
Hammads-MacBook-Pro:minitwit hammad$ flask initdb
Initialized the database.
Hammads-MacBook-Pro:minitwit hammad$ flask run
 * Serving Flask app "minitwit"
 * Forcing debug mode off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
^[
```

With debugger

```
[Hammads-MacBook-Pro:minitwit hammad$ export FLASK_DEBUG=1                    ]
[Hammads-MacBook-Pro:minitwit hammad$ flask run                               ]
 * Serving Flask app "minitwit"
 * Forcing debug mode on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 510-040-896
```

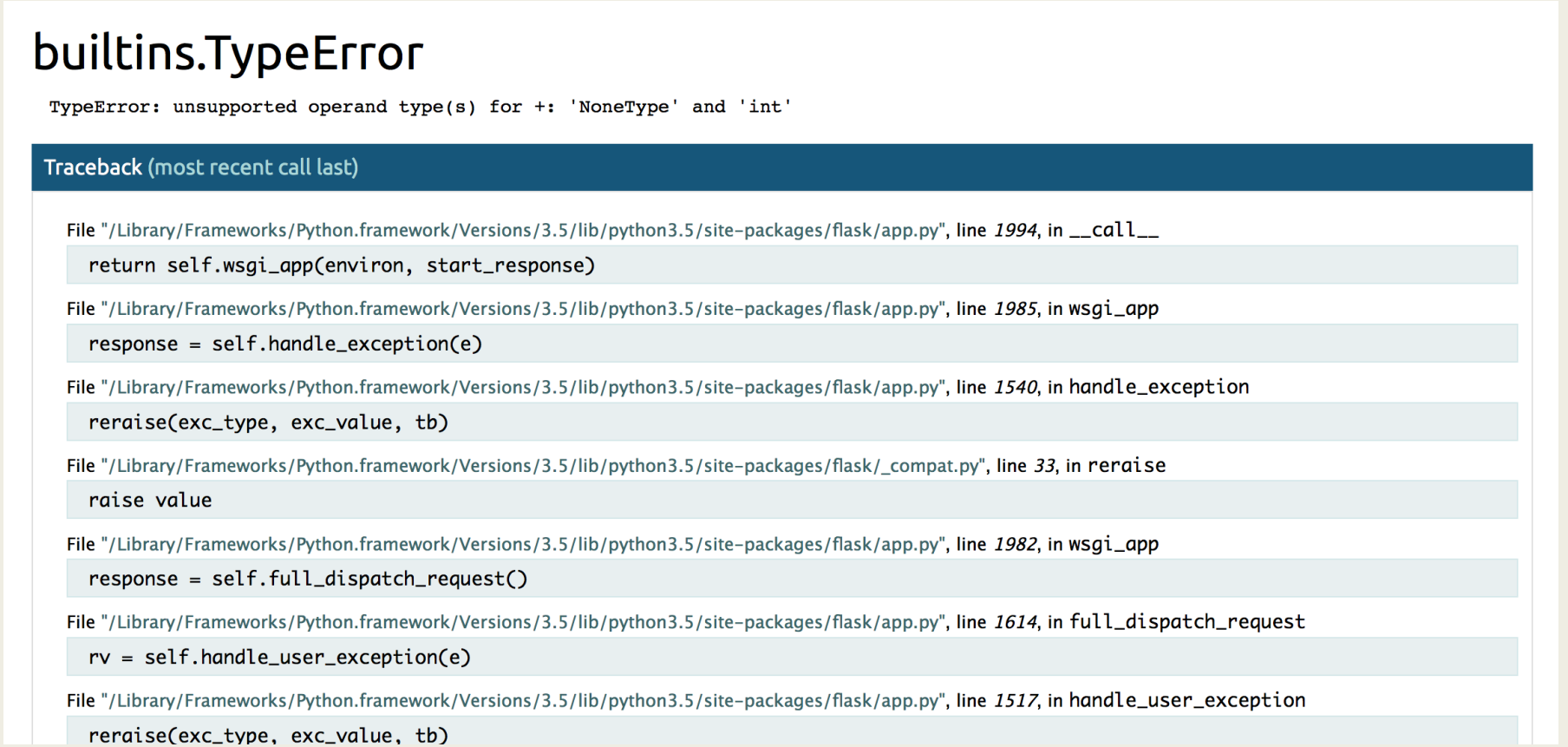
Debugging

What happens if there is an error and the debugger is off?



Debugging

What happens if there is an error and the debugger is on?



builtins.TypeError

```
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Traceback (most recent call last)

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/app.py", line *1994*, in __call__

```
return self.wsgi_app(environ, start_response)
```

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/app.py", line *1985*, in wsgi_app

```
response = self.handle_exception(e)
```

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/app.py", line *1540*, in handle_exception

```
reraise(exc_type, exc_value, tb)
```

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/_compat.py", line *33*, in reraise

```
raise value
```

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/app.py", line *1982*, in wsgi_app

```
response = self.full_dispatch_request()
```

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/app.py", line *1614*, in full_dispatch_request

```
rv = self.handle_user_exception(e)
```

File "/Library/Frameworks/Python.framework/Versions/3.5/lib/python3.5/site-packages/flask/app.py", line *1517*, in handle_user_exception

```
reraise(exc_type, exc_value, tb)
```

Debugging

A closer look at the function that raised the exception:

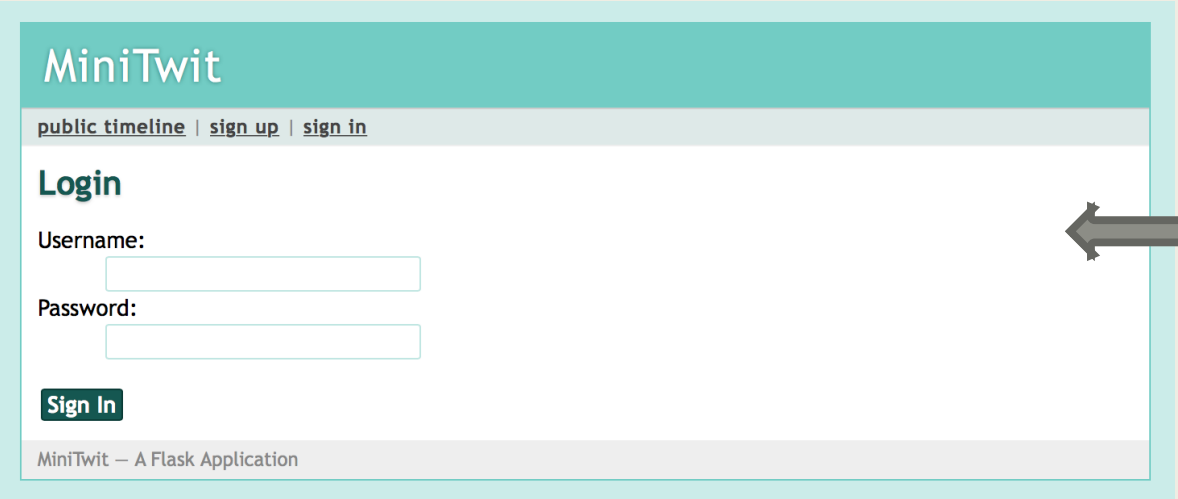
File "/Users/hammad/Desktop/CSCI 397-01/flask-master/examples/minitwit/minitwit/minitwit.py", line *264*, in stupid_feature

```
@app.route('/stupid_feature')
def stupid_feature():
    """A useless method that should raise an exception."""
    item = None
    return item + 1

# add some filters to jinja
app.jinja_env.filters['datetimeformat'] = format_datetime
app.jinja_env.filters['gravatar'] = gravatar_url
```

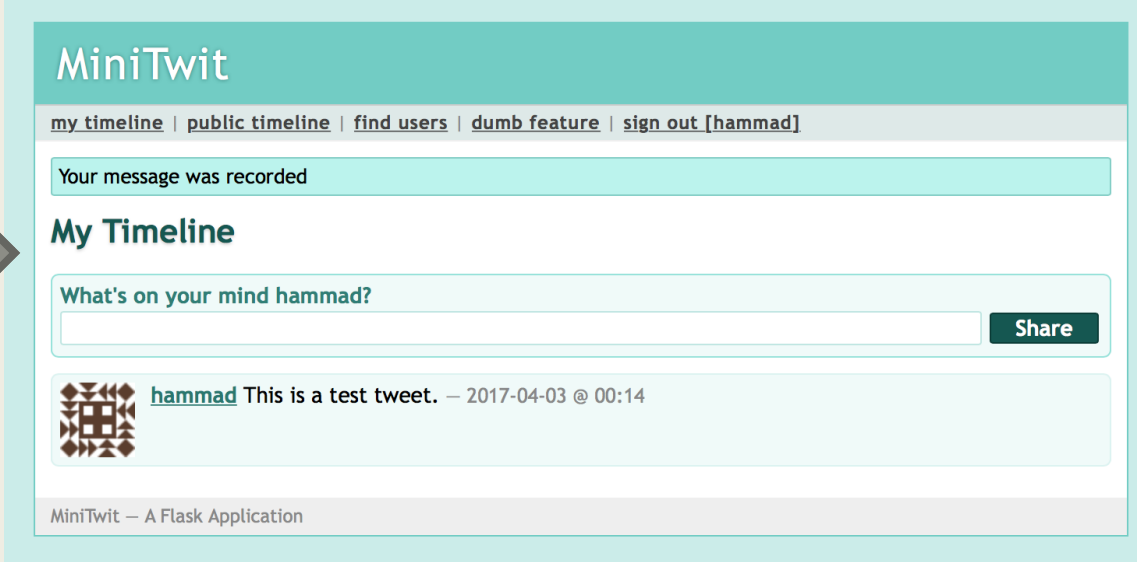
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'

Let us show you how MiniTwit works!



Login Screen

User Timeline



MiniTwit Demo

- In the terminal:
  - ssh -XY carl
  - firefox
- Go to: https://127.0.0.1:5000/
- Create an account
- Tweet!

Comparison

Scalability
Extensibility
Flexibility
Usability

Django: "The web framework for perfectionists with deadlines"

Pyramid: "The start small, finish big, stay finished framework"

When You Need **Pyramid**

Megaframeworks make decisions for you. But if you don't fit their viewpoint, you end up fighting their decisions.
Microframeworks force no decisions, making it easy to start. But as your application grows, you're on your own.

In both cases, the focus is on the **start**: either too much or too little. Either way, finishing and staying finished is hard. You need a **finishing-focused** framework with an architectural design that scales down to getting started, then up as your application grows.

Pyramid was made for just this. It's a Goldilocks Solution: not too small, not too big, just right.

Pyramid The Start **Small**, Finish **Big**, Stay **Finished** Framework.

Scalability

- Microframework
- Small applications with simple requirements
- One or two functions

- Microframework(?)
- Mid-large sized projects
- Instagram, Pinterest, Washington Post

- "Goldilocks"
- Any size
- Cars.com, Survey Monkey, Dropbox

Extensibility

- Newest framework
- Meant for small applications
- Good for a program with a simple task

- Older framework
- Lots of extensions and plug-ins

- Stems from Pylons projects
- Lots of extensions and plug-ins

Flexibility

- Developer chooses templating, database, security, etc.
- Developer chooses how to store data

- Built-in templating, database, security, etc.
- Includes ORM out of the box

- Developer chooses templating, database, security, etc.
- Developer chooses how to store data

Usability

- Quickest to start
- Confusing documentation

- Quick because of built-in
- Well documented
- Most widely used

- Advertises quick start up
- Have to choose different settings

When should *you* use Flask?

- You want to get started fast
- You are a beginner
- You don't like to make decisions
- You don't have huge plans for extensions
- You don’t need a lot of bells and whistles

Learn more about Flask!

⟹ http://flask.pocoo.org/ ⟸

Sources:

- https://www.airpair.com/python/posts/django-flask-pyramid
- https://github.com/pallets/flask/tree/master/examples/minitwit
- http://flask.pocoo.org
- https://scotch.io/tutorials/getting-started-with-flask-a-python-microframework