CptS 322- Programming Language Design

Flask Overview

Instructor: Sakire Arslan Ay

Fall 2023

WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Flask

- Small, lightweight, straightforward web framework
- Adapts easily to your way of thinking
- Plays well with third party Python packages.
- Lots of extensions for Flask (easy to build extensions).

Setup

- Install Python Interpreter
  - Python 3.4.x or higher
- (Initial) Flask class examples:
  - https://github.com/WSU-CptS-arslanay/FlaskLectureExample
- Student enrollment app
  - Flask exercise – Student App (Canvas)

“Hello World” app

hello.py

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

@app.route('/')
def index():
    return "<h1>Hello!</h1>"

@app.route('/course')
def course():
    return '<h1>Hello class!</h1>'

@app.route('/course/<name>')
def mycourse(name):
    return '<h1>Hello, {0} class!</h1>'.format(name)

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

Creates the flask app.

- Flask constructor takes the module name as argument. To create the application, flask needs to know where your application files are.

Creates a mapping between URL and the decorator function.

View function for the route ‘/course’

Starts the Flask server.

Open the following URL on browser:

- http://localhost:5000/
- http://localhost:5000/course
- http://localhost:5000/course/CptS322

Flask Templates

- Why?
  - Separate presentation from logic.
- How?
  - Create HTML templates (`02_FlaskForms`)
  - Create links between pages (`03_FlaskForms_linkpages`)

Flask Templates

hello.py

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

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

@app.route('/course/<name>')
def course(name):
    return render_template('course.html', name=name)

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

Import "render_template" module.

Render the HTML code in index.html.

Pass the name variable.

Open the following URL on browser:

- http://localhost:5000/
- http://localhost:5000/course/CptS322

Flask Templates – Link pages

Index.html

03_FlaskForms_linkpages

```
<h1>Hello!</h1>
<p>Please choose your class: </p>
<ul>
    <li><a href="{{ url_for('course',name='CptS322')}}"> CptS 322 </a></li>
    <li><a href="{{ url_for('course',name='CptS355')}}"> CptS 355</a></li>
    <li><a href="{{ url_for('course',name='CptS321')}}"> CptS 321</a></li>
    <li><a href="{{ url_for('course',name='CptS451')}}"> CptS 451</a></li>
</ul>
```

Open the following URL on browser:

- http://localhost:5000/

Web Forms

- Why?
  - To accept input from users
- How?
  - Add a form to the application (`04_FlaskForms`)
    - When form is submitted a 'POST' request is submitted.
  - The POST-Redirect-GET pattern (`05_FlaskForms_Redirect`)
  - Using Flask-WTF and WTForms (`06_FlaskWTF`)

Flask Forms

index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Flask Example</title>
  </head>
  <body>
    <h1> Hello!</h1>
    <form method="POST" action="{{url_for('index')}}">
      Please choose your class: <input type="text" name="name">
      <input type="submit" name="submit" value="Submit">
    </form>

    {% if name %}
      <h2>Your course is
         <a href="{{ url_for('course',name=name)}}">  {{name}} </a> !</h2>
    {% endif %}
  </body>
</html>
```

A POST request will be issued to this URL when the form is submitted.

HTML form

Text input box

Submit button

Jinja2 “if” statement

Open the following URL on browser:

- http://localhost:5000/

Flask Forms (cont.)

app.py

```
from flask import Flask, render_template, request
app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    name = None
    if request.method == 'POST' and 'name' in request.form:
        name = request.form['name']
    return render_template('index.html', name=name)

@app.route('/course/<name>')
def course(name):
    return render_template('course.html', name=name)

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

"request" is a built in object in Flask. It corresponds to the request message received from the client.

Open the following URL on browser:

- http://localhost:5000/

Web Forms

- Why?
  - To accept input from users
- How?
  - Add a form to the application (`04_FlaskForms`)
    - When form is submitted a 'POST' request is submitted.
  - The POST-Redirect-GET pattern (`05_FlaskForms_Redirect`)
  - Using Flask-WTF and WTForms (`06_FlaskWTF`)

Flask Forms – “POST-Redirect-GET” pattern

app.py

```
from flask import Flask, render_template, request, redirect, url_for
app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    name = None
    if request.method == 'POST' and 'name' in request.form:
        name = request.form['name']
        return redirect(url_for('course',name=name))
    return render_template('index.html')

@app.route('/course/<name>')
def course(name):
    return render_template('course.html', name=name)

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

When POST request is received, if “name” is provided in the request, the page is redirected to “course”.

Open the following URL on browser:

- http://localhost:5000/

Web Forms

- Why?
  - To accept input from users
- How?
  - Add a form to the application (`04_FlaskForms`)
    - When form is submitted a 'POST' request is submitted.
  - The POST-Redirect-GET pattern (`05_FlaskForms_Redirect`)
  - Using Flask-WTF and WTForms (`06_FlaskWTF`)
    - Separation of concerns : move `CourseNameForm` to "`forms.py`" (`07_FlaskWTF_ver2`)
    - Make sure to install flask-wtf, i.e., `pip install flask-wtf`

Flask Forms using Flask-WTF

index.html

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Flask Example</title>
  </head>
  <body>
    <h1> Hello!</h1>
    <form method="POST">
      {{ form.hidden_tag() }}
      <p>{{ form.name.label}}
         {{ form.name()}} </p>
    <p>{{ form.submit()}}</p>
    </form>
  </body>
</html>
```

HTML can render the WTForm fields automatically.

Open the following URL on browser:

- http://localhost:5000/

Flask Forms (cont.)

app.py

```
from flask import Flask, render_template, request, redirect, url_for
from flask_wtf import FlaskForm
from wtforms.fields import StringField, SubmitField
from wtforms.validators import  ValidationError, DataRequired
app = Flask(__name__)
app.config['SECRET_KEY'] =  'secret!'

class CourseNameForm(FlaskForm):
    name=StringField(label="Please choose your class:",validators=[DataRequired()])
    submit = SubmitField('Submit')

@app.route('/', methods=['GET', 'POST'])
def index():
    form = CourseNameForm()
    if form.validate_on_submit():
        if form.name.data is not None:
            return redirect(url_for('course',name=form.name.data))
    return render_template('index.html', form=form)

@app.route('/course/<name>')
def course(name):
    return render_template('course.html', name=name)

……
```

Import FlaskForm and wtforms fields and validators.

Define form as a class.

Create form instance.

Business Logic

- How?
  - Database Model
  - Flask SQLAlchemy

(`08_Database_SQLAlchemy`) and
(`09_Database_SQLAlchemy_ver2`)

```
pip install Flask-SQLAlchemy
```

- We revise the form and add new fields.

08_Database_SQLAlchemy

forms.py

```
class CourseForm(FlaskForm):
    major = SelectField(label= "Please select major:", choices =
                ['CptS', 'EE','MATH','ME', 'CHE'],validators=[DataRequired()])
    coursenum = StringField(label= "Please enter course number:",
                validators=[DataRequired()])
    title = StringField(label= "Please enter course title:",
                validators=[DataRequired()])
    submit = SubmitField('Submit')
```

Dropbox for selecting major.

String input field to enter course number.

String input filed to enter course title.

Database – SQL Alchemy

app.py

08_Database_SQLAlchemy

```
import os
basedir = os.path.abspath(os.path.dirname(__file__))

from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SECRET_KEY'] =  'secret!'
app.config['SQLALCHEMY_DATABASE_URI']='sqlite:///'+os.path.join(basedir,'course.db')
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

from forms import CourseForm

class Course(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    coursenum = db.Column(db.String(3))
    title = db.Column(db.String(150))
    major = db.Column(db.String(20))
    def __repr__(self):
        return '<Course {},{},{},{} >'.format(self.id, self.coursenum, self.title,
self.major)
```

Configure Flask-SQLAlchemy

Create database instance.

Define database model. Tables are defined as classes in the model.

Database – SQL Alchemy (cont.)

- Let's create some courses:

```
> python
>>> from app import db
>>> from app import Course
>>> db.create_all()
>>> newCourse = Course(major='CptS',coursenum='322', title='Software
Engineering')
>>> db.session.add(newCourse)
>>> db.session.commit()
```

- Query the course table:

```
> python
>>> from app import db
>>> from app import Course
>>> Course.query.all()
>>> Course.query.filter_by(major='CptS').all()
>>> Course.query.filter_by(major='CptS').first()
>>> Course.query.filter_by(major='CptS').order_by(Course.title).all()
>>> Course.query.filter_by(major='CptS').count()
```

Database – SQL Alchemy (cont.)

app.py

08_Database_SQLAlchemy

```
...
@app.route('/', methods=['GET', 'POST'])
def index():
    form = CourseForm()
    if form.validate_on_submit():
        if (form.major.data is not None) and (form.coursenum.data is not None):
            #check if course already exists
            _coursecount = Course.query.filter_by(major=form.major.data).filter
_by(coursenum=form.coursenum.data).count()
            if _coursecount < 1:
                # add new course to the database
                newcourse = Course(major = form.major.data,coursenum = form.cou
rsenum.data,title = form.title.data)
                db.session.add(newcourse)
                db.session.commit()
                return redirect(url_for('course',name="{}{}-
{}".format(form.major.data,form.coursenum.data, form.title.data)))

    return render_template('index.html', form=form)
```

Search the Course table and get all courses that has the same major and coursenum.

If no such course exists…

Add the new course to the database.

After the course is created, redirect to the course page…

Database – SQL Alchemy (cont.)

09_Database_SQLAlchemy_ver2

app.py

```
@app.route('/', methods=['GET', 'POST'])
def index():
    form = CourseForm()
    if form.validate_on_submit():
    …..

    # display existing courses
    _courses = Course.query.order_by(Course.coursenum).all()
    return render_template('index.html', form=form, courses = _courses)
```

Get all courses in the Course table.

index.html will display all course.

index.html

```
……
<h2>Current Courses:</h2>
  <ul>
    {% for course in courses %}
    <li>
<a href="{{url_for('course',name='{}{}-{}'.format(course.major,course.coursenum,course.title))}}">
        {{ course.major}}{{course.coursenum}}-{{course.title}}  </a> </li>
    {% endfor %}
  </ul>
```

Display each course , link course names to their course pages.

Application structure

- Separation of concerns

(10-AppStructure)

```
app/
    static/
    templates/
        index.html          HTML template for the main page
        course.html         HTML template for the course page
    __init__.py             Python script that initializes the app.
    forms.py                Flask-WTF forms
    models.py               Database model
    routes.py               Flask routes
.flaskenv
config.py                   Defines app's configuration variables
app.py                      Main application script
```

Database – Relationships

11_SQLAlchemy_ManytoOne

12_SQLAlchemy_ManytoOne_Alternative

13-SQLAlchemy_ManytoMany

14-SQLAlchemy_ManytoMany_AssociationObject

- Demo in class

Flask-SqlAlchemy Resources

- https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-viii-followers
- https://flask-sqlalchemy.palletsprojects.com/en/2.x/models/