2023-Spring
Advanced Computer Programming (Week 14)

CSIE, Asia Univ.

Course schedule

- W1-Introduction
- W2-Python libraries
- W3-BeautifulSoup(1)
- W4-BeautifulSoup(2)
- W5-
- W6-Scrapy(1)
- W7-Scrapy(2)
- W8-Storing Data
- W9-Midterm project
- W10-Web & HTTP
- W11-Flask
- W12-Flask Routes
- W13-Jinja template
- W14-Flask-form
- W15-Flask-mail
- W16-REST API
- W17-Project development(2)
- W18-Final presentation


python
Flask
web development,
one drop at a time

Assignment 3

- Choose one of Bootstrap Personal Templates from https://bootstrapmade.com/bootstrap-personal-templates/
- Replace the personal data and photos on the webpage with your own information.
- Refer to the step-by-step tutorial for constructing the pythonanywhere website
- submit the url of your pythonanywhere website

(14) Flask-WTF

- Creating Forms
- Validating Forms
- CSRF
  - Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

https://flask-wtf.readthedocs.io/en/1.0.x/quickstart/

Example 4-1. hello.py: Flask-WTF configuration

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'hard to
guess string'
```

Example 4-2. hello.py: form class definition

```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired

class NameForm(FlaskForm):
    name = StringField('What is your name?', validators=[DataRequired()])
    submit = SubmitField('Submit')
```

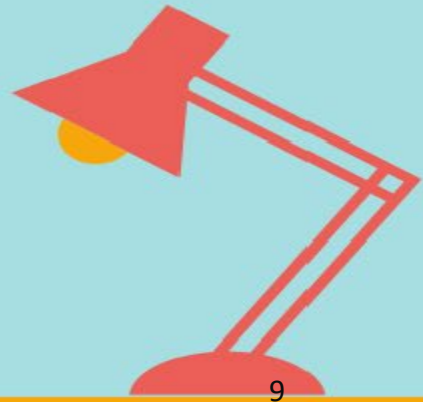
Table 4-1. WTForms standard HTML fields

Field type	Description
`BooleanField`	Checkbox with `True` and `False` values
`DateField`	Text field that accepts a `datetime.date` value in a given format
`DateTimeField`	Text field that accepts a `datetime.datetime` value in a given format
`DecimalField`	Text field that accepts a `decimal.Decimal` value
`FileField`	File upload field
`HiddenField`	Hidden text field
`MultipleFileField`	Multiple file upload field
`FieldList`	List of fields of a given type

Field type	Description
`FloatField`	Text field that accepts a floating-point value
`FormField`	Form embedded as a field in a container form
`IntegerField`	Text field that accepts an integer value
`PasswordField`	Password text field
`RadioField`	List of radio buttons
`SelectField`	Drop-down list of choices
`SelectMultipleField`	Drop-down list of choices with multiple selection
`SubmitField`	Form submission button
`StringField`	Text field

Table 4-2. WTForms validators

Validator	Description
`DataRequired`	Validates that the field contains data after type conversion
`Email`	Validates an email address
`EqualTo`	Compares the values of two fields; useful when requesting a password to be entered twice for confirmation
`InputRequired`	Validates that the field contains data before type conversion
`IPAddress`	Validates an IPv4 network address
`Length`	Validates the length of the string entered
`MacAddress`	Validates a MAC address
`NumberRange`	Validates that the value entered is within a numeric range
`Optional`	Allows empty input in the field, skipping additional validators
`Regexp`	Validates the input against a regular expression
`URL`	Validates a URL
`UUID`	Validates a UUID
`AnyOf`	Validates that the input is one of a list of possible values
`NoneOf`	Validates that the input is none of a list of possible values



Example 4-3. templates/index.html

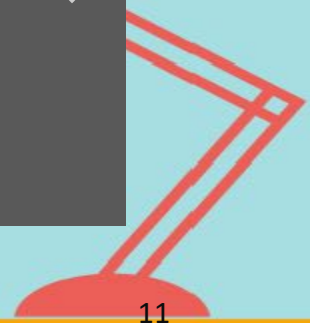
```
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}

{% block title %}Flasky{% endblock %}

{% block page_content %}
<div class="page-header">
    <h1>Hello, {% if name %}{{ name }}{% else %}Stranger{% endif %}!</h1>
</div>
{{ wtf.quick_form(form) }}
{% endblock %}
```

Example 4-4. hello.py: handle a web form with GET and POST request methods

```
@app.route('/', methods=['GET', 'POST'])
def index():
    name = None
    form = NameForm()
    if form.validate_on_submit():
        name = form.name.data
        form.name.data = ''
    return render_template('index.html', form=form, name=name)
```



Example 4-5. hello.py: redirects and user sessions

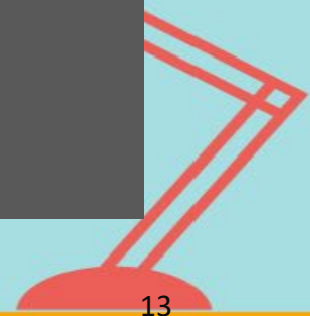
```
from flask import Flask, render_template, session, redirect, url_for

@app.route('/', methods=['GET', 'POST'])
def index():
    form = NameForm()
    if form.validate_on_submit():
        session['name'] = form.name.data
        return redirect(url_for('index'))
    return render_template('index.html', form=form, name=session.get('name'))
```

Example 4-6. hello.py: flashed messages

```
from flask import Flask, render_template, session, redirect, url_for, flash

@app.route('/', methods=['GET', 'POST'])
def index():
    form = NameForm()
    if form.validate_on_submit():
        old_name = session.get('name')
        if old_name is not None and old_name != form.name.data:
            flash('Looks like you have changed your name!')
        session['name'] = form.name.data
        return redirect(url_for('index'))
    return render_template('index.html',
        form = form, name = session.get('name'))
```



Example 4-7. templates/base.html: rendering of flashed messages

```
{% block content %}
<div class="container">
    {% for message in get_flashed_messages() %}
    <div class="alert alert-warning">
        <button type="button" class="close" data-
dismiss="alert">&times;</button>
        {{ message }}
    </div>
    {% endfor %}

    {% block page_content %}{% endblock %}
</div>
{% endblock %}
```

Thanks!

Q&A