web application frameworks
server side structures

static site
› same request, same response
› just a tree of files, including Javascript

dynamic site
› same request, may get different response
› simplest structure: files, with some PHP or Perl (eg)
php example

<form action="action.php" method="post">
<p>Your name: <input type="text" name="name" /></p>
<p>Your age: <input type="text" name="age" /></p>
<p><input type="submit" /></p>
</form>

form.html

Hi <?php echo htmlspecialchars($_POST['name']); ?>.
You are <?php echo (int)$_POST['age']; ?> years old.

action.php

› unlike Javascript, PHP executed server side
› note access to submitted form data
full blown web app

application includes
- application code files in Java, Python, Ruby, etc
- HTML template files, CSS, Javascript
- database schema files
- application framework (Rails, Django, Flask, etc)
- database system (SQLite, MySQL, Postgres, Oracle, etc)
web server calls application code via gateway (eg, WSGI)
one call per HTTP request
flask: a microframework

why did we choose it?
› small and easy to get going
› nicely documented
› has essentials (routing, templates)
› many extensions (mail, forms, ORM)
› uses Python, which you learnt in 6.01

also, not too much magic
› easier to understand what’s going on
  than in more elaborate frameworks (Django, Rails, etc)
structure of a simple flask app

myapp/
  myapp.py
static/
  style.css
templates/
  layout.html
  index.html
  login.html
...

myapp/
  myapp.py
util/
  pretty_time.py
model/
  users.py
  purchases.py
static/
  style.css
templates/
  layout.html
  index.html
  login.html
...
flaskr: a sample app
add_entry is a view function according to Flask docs

I’d call it an action, and I’d call flaskr.py a controller
{% extends "layout.html" %}
{% block body %}
{% if session.logged_in %}
  <form action="{{ url_for('add_entry') }}" method=post class=add-entry>
    <dl>
      <dt>Title:
      <dd><input type=text size=30 name=title>
      <dt>Text:
      <dd><textarea name=text rows=5 cols=40></textarea>
      <dd><input type=submit value=Share>
    </dl>
  </form>
{% endif %}
{% endif %}
<ul class=entries>
{% for entry in entries %}
  <li><h2>{{ entry.title }}</h2>{{ entry.text|safe }}</li>
{% else %}
  <li><em>Unbelievable. No entries here so far</em>
{% endfor %}
</ul>
{% endblock %}
session state

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != app.config['USERNAME']:
            error = 'Invalid username'
        elif request.form['password'] != app.config['PASSWORD']:
            error = 'Invalid password'
    else:
        session['logged_in'] = True
        flash('You were logged in')
        return redirect(url_for('show_entries'))
    return render_template('login.html', error=error)
```

- session encapsulates set-cookie/cookie
- flash adds message to list; in template, get_flashed_messages()
what a typical action does

get data from request
› query string passed as args to action

```python
@app.route('/subjects/<sid>/reviews/<rid>', methods=['GET'])
def show_review(sid, rid):
    review = Review.get_by_id(rid)
```
› form data passed in request.form object

read and write state
› database state, session state

generate response
› create flash messages, set error variables
› instantiate template and return
› or redirect to another action or call
MVC structure

- **Model**: what data is stored? what operations?
- **Controller**: what happens next? what if not logged in? session state?
- **View**: what data is displayed? how is data displayed? appearance, themes?

*Acceptable dependence*:

*Undesirable but common dependence*:

What if not logged in? session state?
MVC and Flask

not a very good fit?

aim of MVC
› separate concerns
› user interface/application logic/underlying data

most important that
› data layer does not depend on view

ideally
› nothing depends on user interface
› so can access as API, eg
› but Flask not really designed for this