IO Lab: Python Web Frameworks Flask

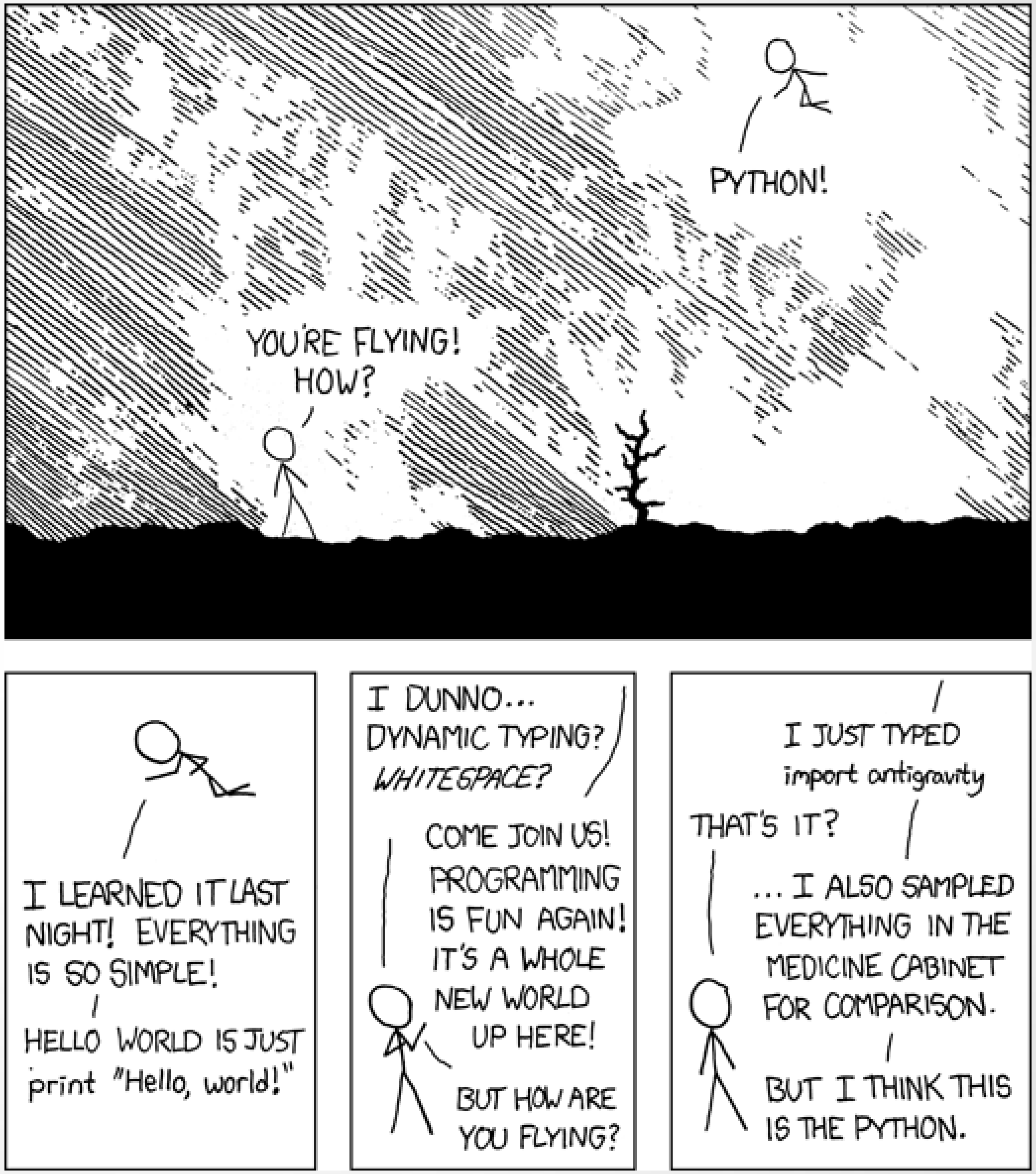
November 18, 2013

Info 290TA (Information Organization Lab)
Kate Rushton & Raymon Sutedjo-The

Python

Python

Is an interactive, object-oriented, extensible programming language.



PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Syntax

- Python has semantic white space.

```
//JavaScript
if (foo > bar)
           { foo = 1;
 bar = 2;                }

# python
if foo > bar:
    foo = 1
    bar = 2
```

Indentation Rules

- Relative distance
- Indent each level of nesting

```
if foo > bar:
   for item in list:
      print item
```

- Required over multiple lines; single line uses semi-colons

```
if a > b: print "foo"; print "bar";
```

Variables

- Must begin with a letter or underscore, can contain numbers, letters, or underscores

```
foo = 0
_bar = 20
another1 = "test"
```

- Best practice – separate multiple words with underscores, do not use camel case

```
api_key = "abcd12345" #not apiKey
```

Comments

- Single-line comments are marked with a #

```
# this is a single line comment
```

- Multi-line comments are delineated by three "

```
" " " this is a comment
  that spans more than one line
" " "
```

Data Types

Type	Example
int	14
float	1.125
bool	True/False
str	“hello”
list	[“a”,”b”,”c”]
tuple	(“Oregon”, 1, False)
dict	{ “name”: “fred”, “age”, 29 }

Strings

- Defined with single or double quotes

```
fruit =  "pear"
name = 'George'
```

- Are immutable

```
fruit[0] = "b"   # error!
```

Strings

"hello"+"world“	-->	"helloworld"
"hello"*3	-->	“hellohellohello"
"hello"[0]	-->	“h"
"hello"[-1]	-->	“o“
"hello"[1:4]	-->	“ell”
"hello" < "jello“	-->	True

More String Operations

len(“hello”)	-->	5
“g” in “hello”	-->	False
"hello”.upper()	-->	“HELLO"
"hello”.split(‘e’)	-->	[“h”, “llo”]
“ hello ”.strip()	-->	“hello”
"hello" < "jello“	-->	True

Lists

- Usually defined with brackets

```
fruit =  ["apple", "cherry", "kiwi"]
```

- Can contain different data types

```
stuff = [3, “blind”, “mice”]
```

- Are mutable

```
stuff[0] = “three”
```

Lists

```
fruit = [“apple”, “cherry”, “kiwi”]

fruit[0]                    -->   “apple”

fruit[1] = “pear”           -->    [“apple”, “pear”, “kiwi”]

fruit.append(“grape”)       -->   [“apple”, “cherry”, “kiwi”, “grape”]

fruit.insert(1, “grape”)    -->   [apple”, “grape”, “cherry”, “kiwi”]

fruit.index( “cherry”)      -->   1

fruit.index( “orange”)      -->   ValueError
```

Looping Through Lists

- With **for..in**

```
for letter in [“a”, “b”, “c”]:
      print letter
```

- With **range**

```
for i in range(len(list)):
      print list[i]
```

Looping Through Lists

- Use **enumerate** to get both index and item

```
for i,letter in enumerate([“a”, “b”, “c”]):
     print letter * (i+1)
```

List Comprehension

- When performing an action on every item in a list, can do something like

```
doubled = []
for x in [2, 3, 4, 5]:
     doubled.append(x*2)
```

- **List comprehensions** make this simpler

```
doubled = [x*2 for x in [2, 3, 4, 5]]:
```

Dictionaries

- Defined with braces { } or **dict()** constructor
- Key-value pairs – keys must be immutable (most often strings/numbers)

```
pet = { "name": "fido",
      "type": "dog" }
```

Standard dictionaries are unordered.

Dictionaries

```
employee = {  "id": 133, "name": "Bob",
"location" : "Chicago" }
```

len(employee)	-->	3
employee["id"]	-->	133
employee.get("id")	-->	133
employee.values()	-->	[133, "Bob", "Chicago"]
del employee["location"]	-->	{ "id" : 133, "name": "Bob"}
"Bob" in d	-->	False

Looping through Dictionaries

- With **for..in**

```
for key in mydict:
     print mydict[key]
```

- With **iteritems/items**

```
for key, val in mydict.items():
     print key + ":" + val
```

Functions

- Defined with the keyword **def** followed by the function name and then arguments in parens
- Return values with the **return** keyword

```
def add_two_things(arg1, arg2):
    return arg1 + arg2

sum = add_two_things(1,2)   #sum is 3
```

Functions

- Can provide default values for arguments in the function definition

```
def add_two_things(arg1=2, arg2=2):
    return arg1 + arg2;


sum = add_two_things(1,2)    # sum is 3

sum = add_two_things()       # sum is 4
```

Classes and Objects

- Define a class with the **class** keyword

```
class Car:
    #instance data
    #and function definitions
```

- Define instance data with **self**

```
self.make = "Geo"
self.model = "Prism"
self.year = 1997
self.max_speed = 60
```

Classes and Objects

- All functions in a class have the parameter **self**

```
def stop(self):
     self.speed = 0
def go(self, speed):
     self.speed = speed
```

- Implement the __init___ constructor function

```
def __init__(self, model, year):
     self.model = model
     self.year = year
     #other setup tasks
```

Classes and Objects

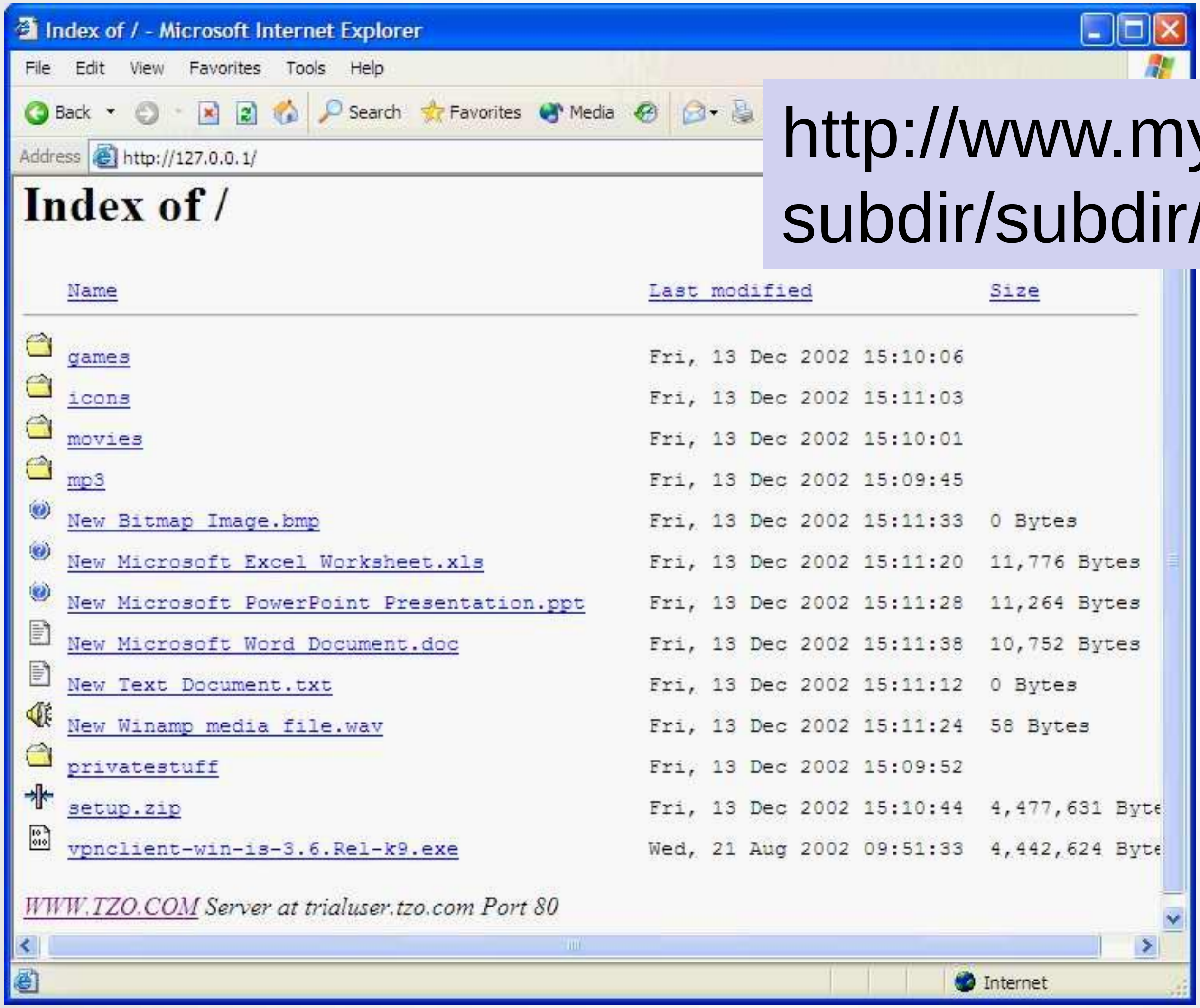
```
class Employee:
'Represents one employee'
  def __init__(self, first, last, salary):
      self.first_name = first
      self.last_name = last
      self.salary = salary

  def give_raise(self, amount=100):
      self.salary += amount

  def get_name(self):
      return first + " " + last
```

Classes and Objects

```
new_hire= Employee("Bob", "Jones", 30000)

new_hire.give_raise(1000)
#salary is now 31K

new_hire.full_name()
#outputs Bob Jones
```

Web Frameworks

File-Folder Web


Index of / - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://127.0.0.1/
Internet

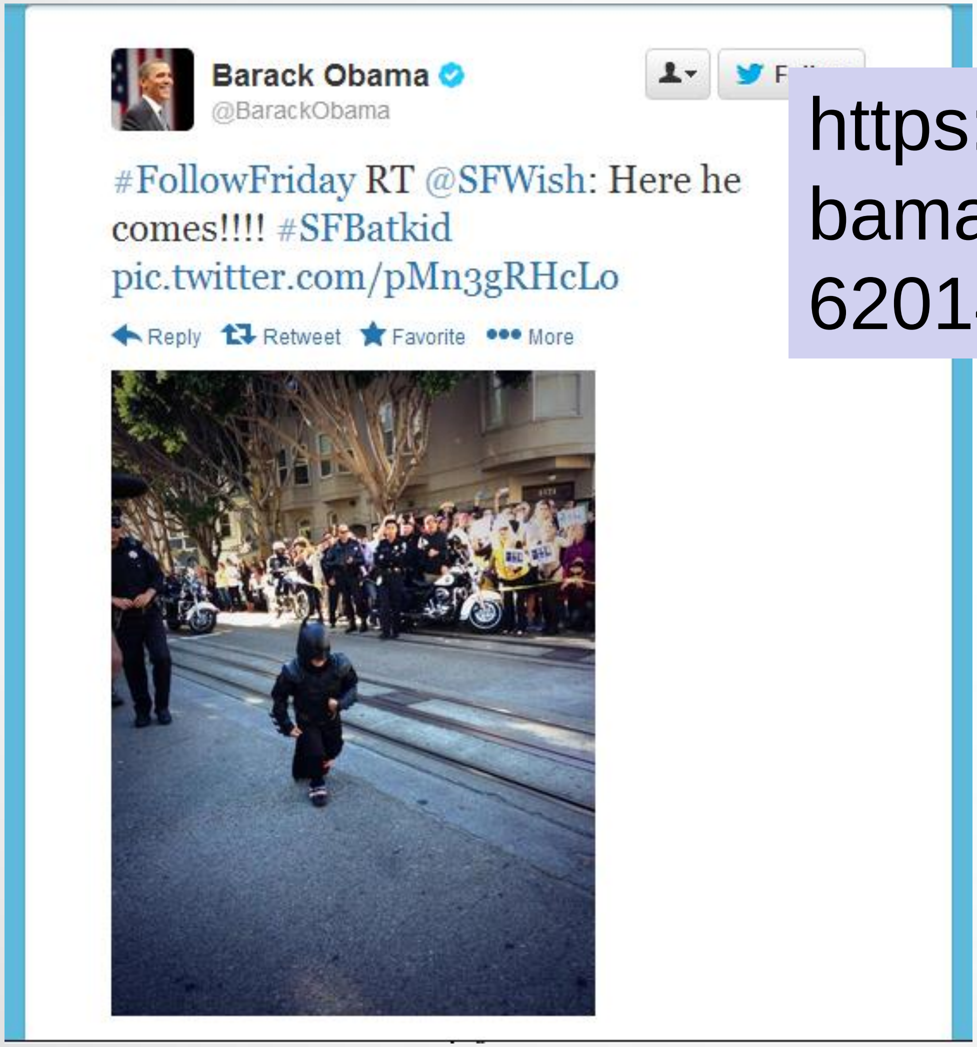
Index of /

```
Name                                      Last modified                 Size
games                                     Fri, 13 Dec 2002 15:10:06
icons                                     Fri, 13 Dec 2002 15:11:03
movies                                    Fri, 13 Dec 2002 15:10:01
mp3                                       Fri, 13 Dec 2002 15:09:45
New Bitmap Image.bmp                      Fri, 13 Dec 2002 15:11:33    0 Bytes
New Microsoft Excel Worksheet.xls         Fri, 13 Dec 2002 15:11:20    11,776 Bytes
New Microsoft PowerPoint Presentation.ppt Fri, 13 Dec 2002 15:11:28    11,264 Bytes
New Microsoft Word Document.doc           Fri, 13 Dec 2002 15:11:38    10,752 Bytes
New Text Document.txt                     Fri, 13 Dec 2002 15:11:12    0 Bytes
New Winamp media file.wav                 Fri, 13 Dec 2002 15:11:24    58 Bytes
privatestuff                              Fri, 13 Dec 2002 15:09:52
setup.zip                                 Fri, 13 Dec 2002 15:10:44    4,477,631 Byte
vpnclient-win-is-3.6.Rel-k9.exe           Wed, 21 Aug 2002 09:51:33    4,442,624 Byte
```

WWW.TZO.COM Server at trialuser.tzo.com Port 80

http://www.mysite.com/dir/subdir/subdir/page.html

Modern Web



Barack Obama
@BarackObama

#FollowFriday RT @SFWish: Here he comes!!!! #SFBatkid pic.twitter.com/pMn3gRHcLo

https://twitter.com/barackobama/status/401427096276201474

Web Frameworks

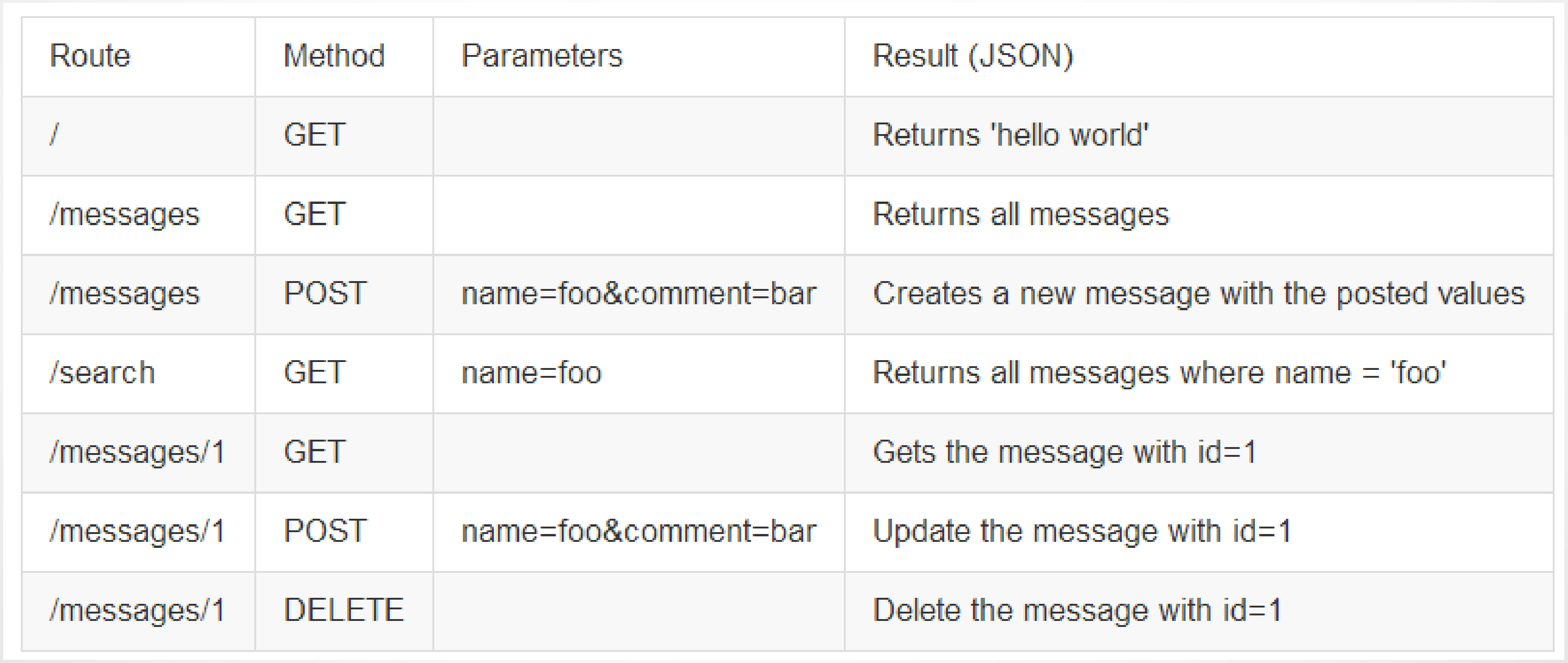
php	CakePHP, Symfony, CodeIgniter
python	Django, web2py, Bottle, Flask
Ruby A Programmer's Best Friend	Ruby on Rails, Sinatra
Java	Spring, Struts

Web Framework Features

- Routing
- Templates
- Database integration / ORM
- Unit testing
- Client or server-side validation
- Localization
- Scaffolding
- Asset management

Web Framework Features

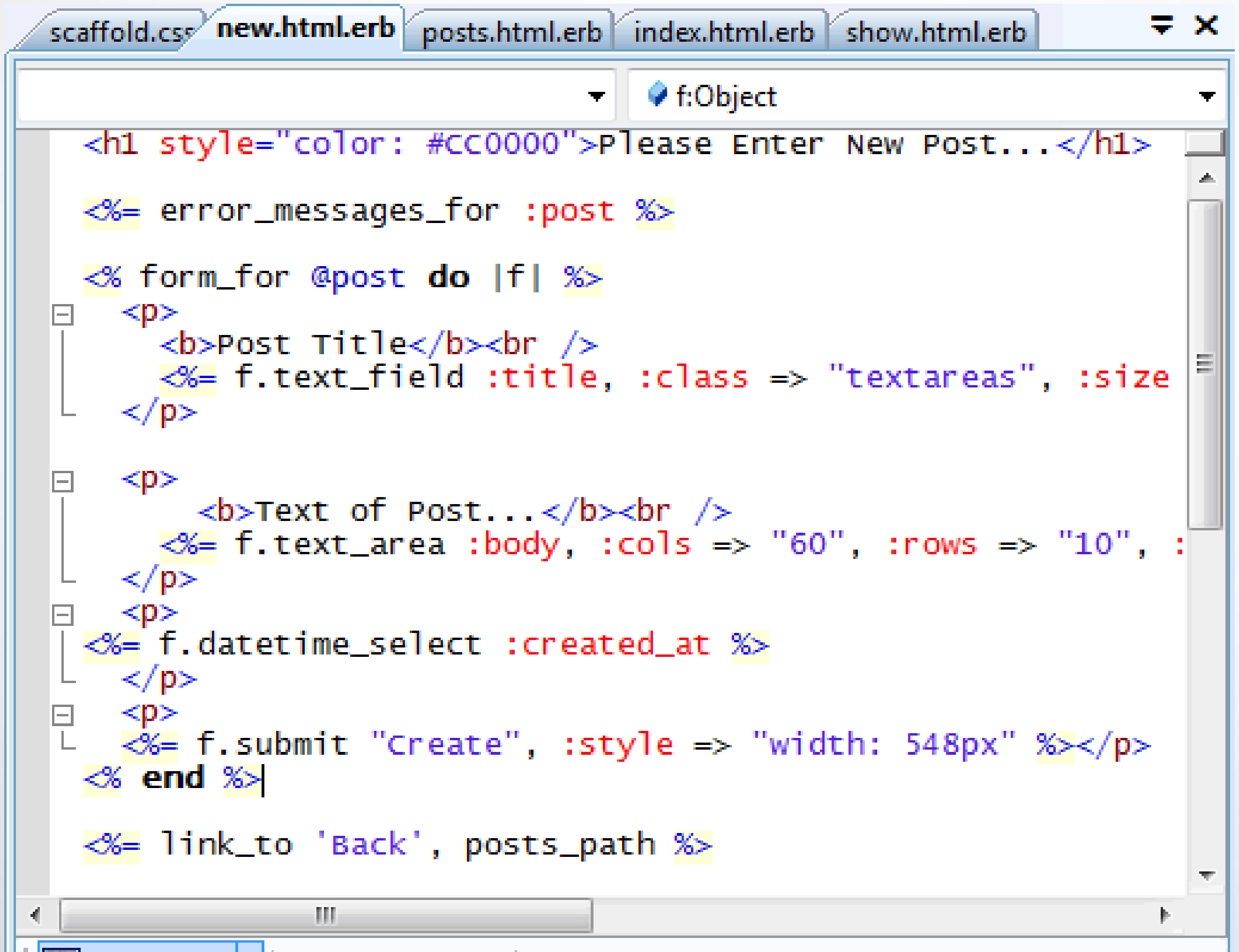
Routing



Route	Method	Parameters	Result (JSON)
/	GET		Returns 'hello world'
/messages	GET		Returns all messages
/messages	POST	name=foo&comment=bar	Creates a new message with the posted values
/search	GET	name=foo	Returns all messages where name = 'foo'
/messages/1	GET		Gets the message with id=1
/messages/1	POST	name=foo&comment=bar	Update the message with id=1
/messages/1	DELETE		Delete the message with id=1

Web Framework Features

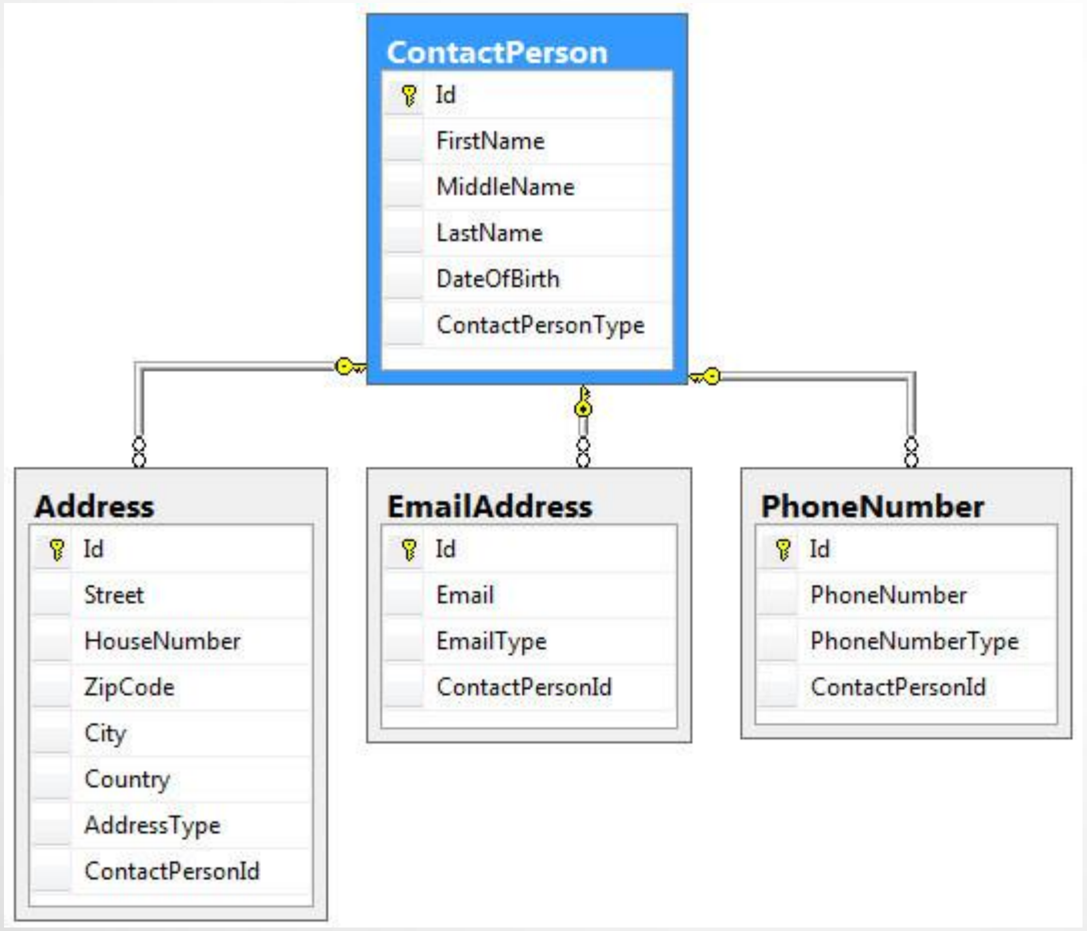
Templates



```
<h1 style="color: #CC0000">Please Enter New Post...</h1>

<%= error_messages_for :post %>

<% form_for @post do |f| %>
  <p>
    <b>Post Title</b><br />
    <%= f.text_field :title, :class => "textareas", :size
  </p>

  <p>
      <b>Text of Post...</b><br />
    <%= f.text_area :body, :cols => "60", :rows => "10", :
  </p>
  <p>
<%= f.datetime_select :created_at %>
  </p>
  <p>
  <%= f.submit "Create", :style => "width: 548px" %></p>
<% end %>

<%= link_to 'Back', posts_path %>
```

Web Framework Features

Database Integration/ORM



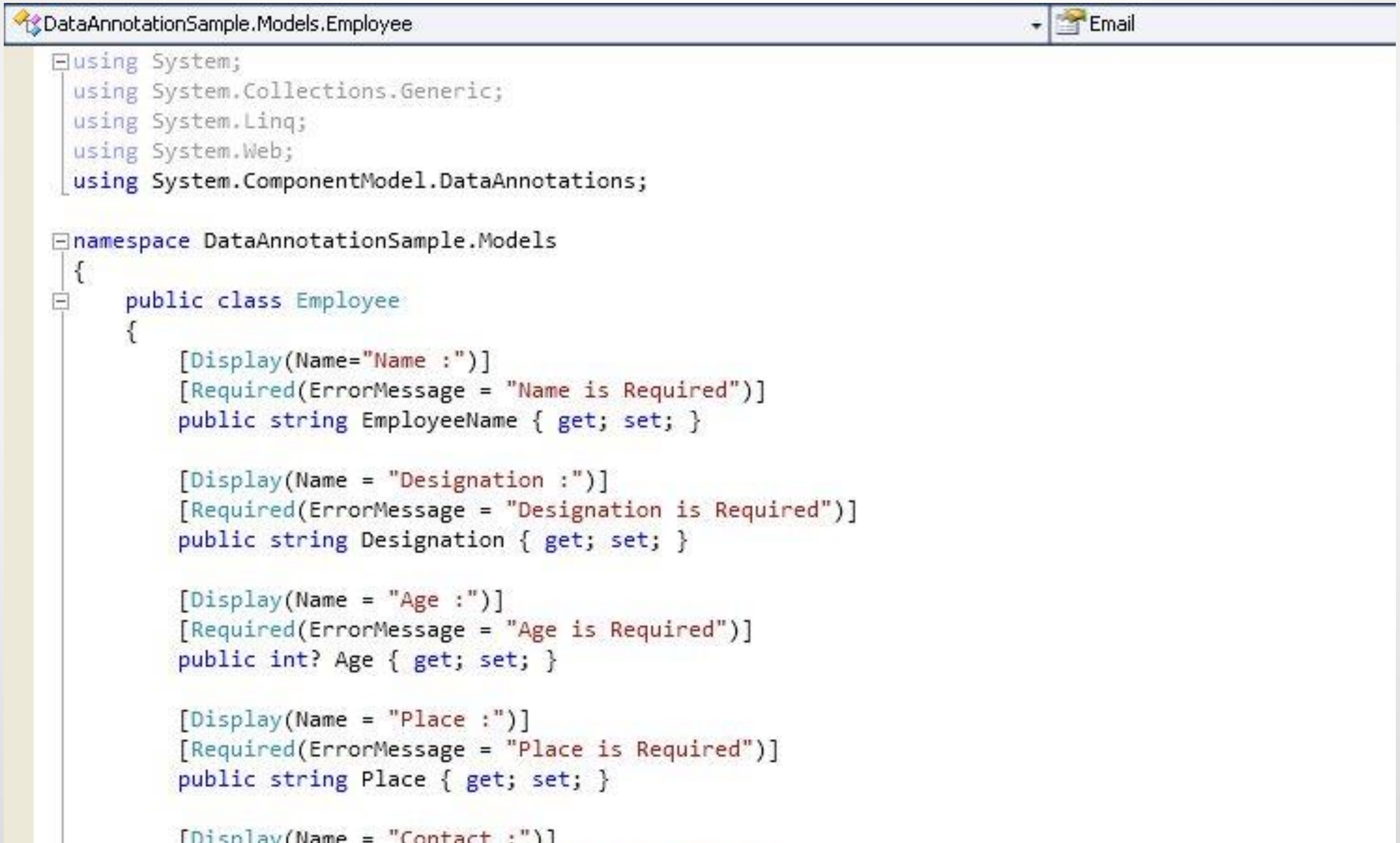
Web Framework Features

Unit Testing

```python
import  unittest
import  quotes

class  MyTests (unittest .TestCase ):

    def  test_add_get_quote  (self):
        quotes .add ("Confucius ",  "A journey of a thousand miles ...
        q = quotes .get ("Confucius ",  contains ="step ")
        self.assertEqual (q, [  "A journey of a thousand miles ...  "

    def  test_add_get_quote_no_contains   (self):
        quotes .add ("Confucius ",  "A journey of a thousand miles ...
        q = quotes .get ("Confucius ")
        self.assertEqual (q, [  "A journey of a thousand miles ...  "

if  __name__  == "__main__ ":
    unittest .main ()
```

Web Framework Features

Validation

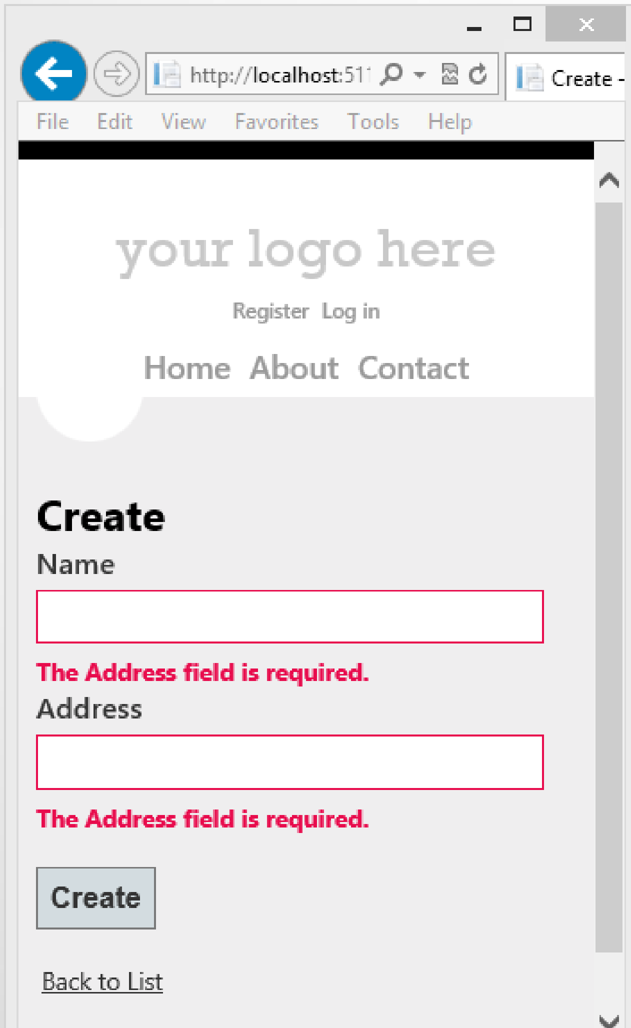


DataAnnotationSample.Models.Employee | Email

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ComponentModel.DataAnnotations;

namespace DataAnnotationSample.Models
{
    public class Employee
    {
        [Display(Name="Name :")]
        [Required(ErrorMessage = "Name is Required")]
        public string EmployeeName { get; set; }

        [Display(Name = "Designation :")]
        [Required(ErrorMessage = "Designation is Required")]
        public string Designation { get; set; }

        [Display(Name = "Age :")]
        [Required(ErrorMessage = "Age is Required")]
        public int? Age { get; set; }

        [Display(Name = "Place :")]
        [Required(ErrorMessage = "Place is Required")]
        public string Place { get; set; }

        [Display(Name = "Contact :")]
```

Web Framework Features

Localization



http://localhost:51

File Edit View Favorites Tools Help

your logo here

Register Log in

Home About Contact

Create

Name

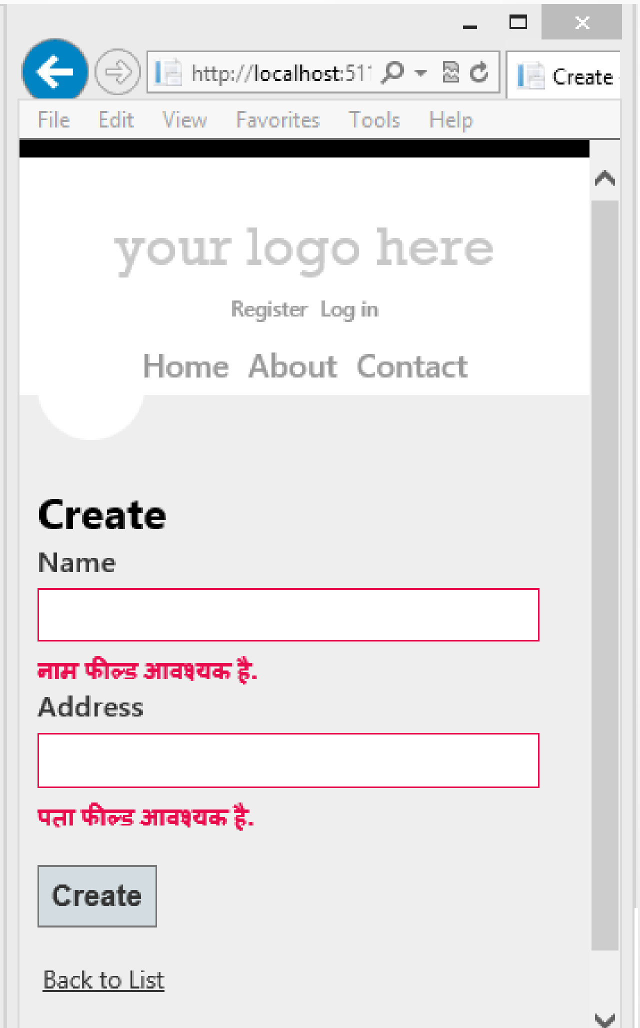
The Address field is required.

Address

The Address field is required.

Create

Back to List



http://localhost:51

File Edit View Favorites Tools Help

your logo here

Register Log in

Home About Contact

Create

Name

नाम फील्ड आवश्यक है.

Address

पता फील्ड आवश्यक है.

Create

Back to List

Web Framework Features

Scaffolding

Create **R**ead **U**pdate **D**estroy

- Create/Add resource
- Edit/Update resource
- View/Show resource
- List/All resources
- Delete resource

Web Framework Features

Asset management

Model-View-Controller

Application architecture that emphasizes separation of business logic and user interaction

- Model – the data and rules that apply to it
- View – a representation of that data in a UI
- Controller – handles requests and mediates between model and view

MVC Grocery Store

Model FoodItem

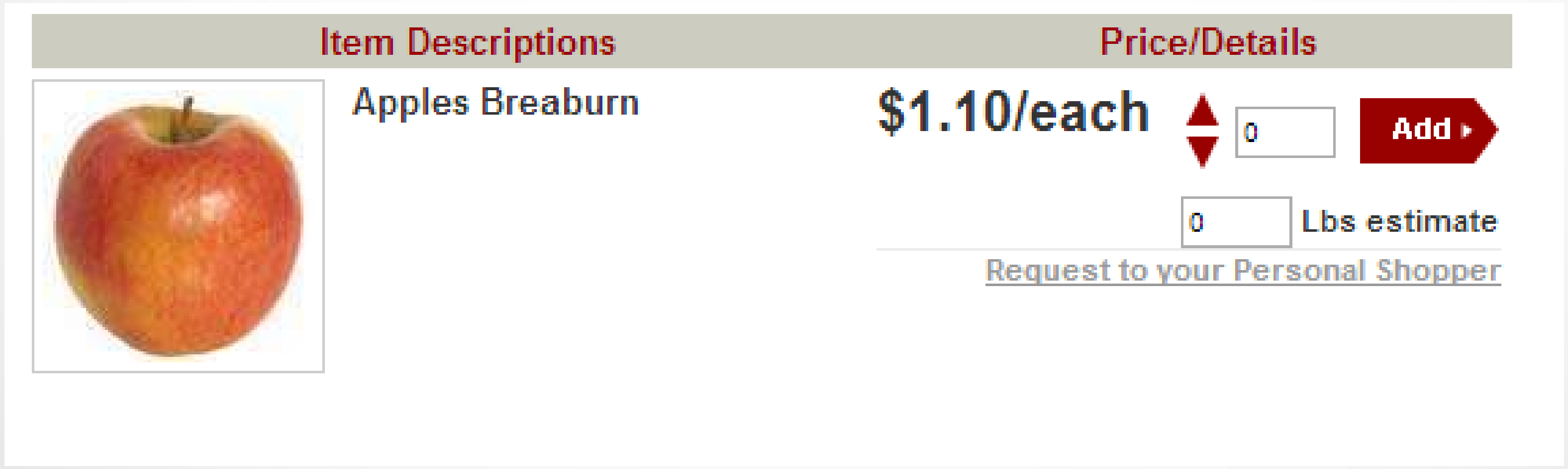
- Attributes: id, name, brand, price, units, whether on sale, number in stock, etc.
- Functions: getPrice(), updateInventory(), etc.

ID: 1093
Name: Apples Breaburn
Brand: Dole
Price: 1.10
Units: each
Number In Stock:1200

MVC Grocery Store

View Item Details

- HTML template for a food item detail view
- Model may change in response to user action



MVC Grocery Store

Controller

- Look for incoming requests, eg a GET request to http://mvcgrocery.com/item/1093
- Retrieve item #1093 from db and use it to populate an item details template page

Django

- Routing ✓
- Templates ✓
- Database integration / ORM ✓
- Unit testing ✓
- Client or server-side validation ✓
- Localization ✓
- Scaffolding ✓
- Asset management ✓

Flask – A Microframework

- Routing ✓
- Templates ✓
- Database integration / ORM ✗
- Unit testing ✗
- Client or server-side validation ✗
- Localization ✗
- Scaffolding ✗
- Asset management ✗

Flask – Hello World

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World!'


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

Flask – Sample Route 1

```
@app.route('/search')
def search():
     q = request.args.get('query')
     #implement search for query,
     #return search results page
```

Flask – Sample Route 2

```
@app.route('/items/<id>')
def food_item(id):
    #get details about food item with
    #specified id from db
    #return with details template
```

Flask – Sample Route 3

```
@app.route('/api/items')
def get_all_items():
    #get all items from db
    #return in JSON array
```

Flask – Route Method

```
@app.route('/items', methods=['GET'])
def get_all_items():
    #get all items from db
    #return in item list page template

@app.route('/items', methods=['POST'])
def post_item():
    params = request.form
    #use params to add a new item to db
    #return success message
```

Flask – Response HTML

```
@app.route('/bad', methods=['GET'])
def bad_html_page():
     message = "Don't do this."
     page = "<html><body><p>"
     + message + "</p></body></html>"
     return page
```

Flask – Route with Template

```
@app.route('/items/<id>')
def food_item(id):

    food = get_item_object_from_db()
    return render_template('item.html',
    item=food)
```

Templates

- Templates are HTML with special markup for server-side rendering
- There are many template engines, but Flask uses jinja2
- Braces/percent signs separate code and html

```
<ul>
{% for book in library %}
     <li>{{ book.title }}</li>
{% endfor %}
</ul>
```

Flask – Directory Structure

```
app.py
static (directory)
     --css and js files
     --images
templates (directory)
     --template files
```

Sample Item Detail Template

```
<html>
     <head>
       <title>{{item.name}}</title>
     ...</head>
     <body>
        <div id="item-details">
          <p>Name: {{item.name}}</p>
          <p>Price: {{item.price}}</p>
        </div>
     </body>
<html>
```

Jinja2 Template Logic

- Conditionals

```
{% if book.available %}
    <p>{{ book.title }}</p>
{% endif %}
```

- Loops

```
<ul>
{% for book in library %}
    <li>{{ book.title }}</li>
{% endfor %}
</ul>
```

Jinja2 Template Logic

- Math

```
<p>Price with sales tax:
{{ book.price * .08 }}</p>
```

- Filters

```
<p>{{ name|upper }}</p>
<p>{{ name|strip }}</p>
```