Accelerating Information Technology

http://aiti.mit.edu

Ghana Summer 2012
Lecture DJ05 – Django Templates
Django Architecture

MVC
(Traditional)

MTV
(Django)
Today’s weather in {{ city }} is {{ description }}.

{% for day in thisWeek %}
<li> On {{ day.date }}, the temperature will be {{ day.temperature }}.
</li>
{% endfor %}

Click on these ads!

{% block ads %}
{% endblock %}
Today’s weather in Accra is sunny.

- On Thursday, the temperature will be 20.
- On Friday, the temperature will be 25.
- On Saturday, the temperature will be 22.
Templates

• A text-based template for HTML, CSS, XML, JavaScript, etc.
• Mixture between hard-coded text and abstractions
• Abstractions
  – Variables
  – Tags
• Re-useable and extensible
<html>
    <head>
        <title>Weather</title>
    </head>
    <body>
        <p>Today's weather in {{ city }} is {{ description }}.</p>
        <div id="temperature">
            {% for day in thisWeek %}
                <li>On {{ day.date }}, the temperature will be {{ day.temperature }}. </li>
            {% endfor %}
        </div>
        <div id="ads">
            {% block ads %}
                Click on these ads!
            {% endblock %}
        </div>
    </body>
</html>
Variables

- {{ variable }}
  - If variable doesn’t exist, then output
    `TEMPLATE_STRING_IF_INVALID` (default: empty string"

- {{ variable.attribute }}

  1. Dictionary Lookup. `variable [“attribute”]`
  2. Attribute Lookup. `variable.attribute`
  3. Method Call. `variable.attribute()`
  4. List–index Call. `variable[attribute]`
Variables in weather.html

<html>
<head>
<title>Weather</title>
</head>
<body>
<p>Today's weather in {{ city }} is {{ description }}.</p>
<div id="temperature">
{% for day in thisWeek %}
<li> On {{ day.date }}, the temperature will be {{ day.temperature }}. </li>
{% endfor %}
</div>
<div id="ads">
{% block ads %}
Click on these ads!
{% endblock %}
</div>
</body>
</html>
Filters

• Modify the output of variables
• `{{ variable|filter }}`

foo := “Hello World”
bar := [‘a’, ‘b’, ‘c’]

`{{ foo|lower }}` --> hello world
`{{ bar|length }}` --> 3
`{{ bar|slice:“:2” }}` --> [‘a’, ‘b’]
`{{ baz|default:“error!” }}` --> error!
Tags

• for loops
• if clauses
• comments
• blocks
• and many more built-in tags (look them up!)

• {% tag %} ... {% endtag %}
Tags in weather.html

<html>
    <head>
        <title>Weather</title>
    </head>
    <body>
        <p>Today's weather in {{ city }} is {{ description }}.</p>
        <div id="temperature">
            {% for day in thisWeek %}
                <li> On {{ day.date }}, the temperature will be {{ day.temperature }}. </li>
            {% endfor %}
        </div>
        <div id="ads">
            {% block ads %}
            Click on these ads!
            {% endblock %}
        </div>
    </body>
</html>
For loops

{% for x in y %}
    ... logic ...
{% endfor %}

fruit_basket := {'apples', 'oranges', 'pineapples'}

{% for fruit in fruit_basket %}
    <li>{{ fruit }}</li>
{% endfor %}

--> <li>apples</li>
    <li>orange</li>
    <li>pineapples</li>
If clauses

{% if <condition> %}
    ... logic ...
{% else %}
    ... logic ...
{% endif %}

{% if rain > 1 %}
    Buy an umbrella for {{ price1 }}
{% else %}
    Buy sunglasses for {{ price2 }}
{% endif %}
Comments

{% comment %}
This comment won’t be displayed!
{% endcomment %}

- Ignore everything inside tag
  - For inline comments, use {# blah blah blah blah #}
Template Inheritance

• Define extensible parts of a template with block tags
  
  ```
  {% block name %}
  ...
  {% endblock %}
  ```

• Create child templates that can extend blocks

• Load parent template with
  
  ```
  {% extends "parent_template" %}
  ```
weather.html

<html>
<head>
    <title>Weather</title>
</head>
<body>
    <p>Today's weather in {{ city }} is {{ description }}.</p>
    <div id="temperature">
        {% for day in thisWeek %}
        <li> On {{ day.date }}, the temperature will be {{ day.temperature }}. </li>
        {% endfor %}
    </div>
    <div id="ads">
        {% block ads %}
        Click on these ads!
        {% endblock %}
    </div>
</body>
</html>
{% extends "weather.html" %}
{% block ads %}
{% if rain > 1 %}
    Buy an umbrella!
{% else %}
    Buy sunglasses!
{% endif %}
{% endblock %}
Today’s weather in Accra is sunny.

- On Thursday, the temperature will be 20.
- On Friday, the temperature will be 25.
- On Saturday, the temperature will be 22.

Buy an umbrella!
Template Inheritance

• In child template, redefine contents of the parent’s block tag
  – similar to overriding methods in class inheritance

• If a block tag is not redefined, then use contents of block tag in parent

• {{ block.super }} explicitly refers to contents of block tag in parent
def get_titles(request, limit=100):
    book_list = Book.objects.all()[:limit]
    response = 'List of titles is:
    for b in book_list:
        response+=str(b.title)
    return HttpResponse(response)
Views before

List of titles: Crime and Punishment, Harry Potter and the Order of Phoenix
def get_titles(request, limit=100):
    book_list = Book.objects.all()[:limit]
    response = HttpResponse()
    response.write('<html><body>')</n    response.write('<h1>List of titles</h1>')
    response.write('<ul>')</n    for b in book_list:
        response.write('<li>Title: ' + b.title + '</li>')
    response.write('</ul>')
    response.write('</body></html>')
    return response
Views with more HTML

List of titles:

- Crime and Punishment
- Harry Potter and the Order of Phoenix
But it’s messy and all together and I don’t like it!

def get_titles(request, limit=100):
    book_list = Book.objects.all()[:limit]
    response = HttpResponse()
    response.write('<html><body>')
    response.write('<h1>List of titles</h1>')
    response.write('<ul>
        for b in book_list:
            response.write('<li>Title: ' + b.title + '</li>"
        response.write('</ul>
    response.write('</body></html>')
    return response
Instead: Template!

```html
display_titles.html

<html>
<h1> List of titles </h1>
<ul>
  {% for book in books %}
    <li>{{ book.title }}</li>
  {% endfor %}
</ul>
</html>
```
def get_titles(request, limit=100):

    book_list = Book.objects.all()[:limit]

    my_temp=get_template('display_titles.html')

    my_context = Context('books', book_list)

    return my_temp.render(my_context)
def get_titles(request, limit=100):
    book_list = Book.objects.all()[:limit]
    my_context = Context('books', book_list)
    return render_to_response('display_titles.html', my_context)
Templates

Remember to specify where your templates are in TEMPLATE_DIRS in settings.py
Templates

• Mixture of hard-coded text and abstractions

• Abstractions often look like and function like Python code, but you can’t run arbitrary Python code
  – Lookup list of built-in filters and tags in Django
  – Customize your own filters and tags

• Complex logic with arbitrary Python should be performed by views.py and only the processed variables should be passed to a template
ads.html

{% extends "weather.html" %}
Today's weather in Accra is sunny.

- On Thursday, the temperature will be 20.
- On Friday, the temperature will be 25.
- On Saturday, the temperature will be 22.

Click on these ads!