Accelerating Information Technology Innovation

http://aiti.mit.edu

India Summer 2012

Lecture 7 – Django: Templates and Forms

Django: Review of Views

Making a Simple View

Make a view method that gets the list of all `City` *objects.*

Making a Simple View

Add a URL to view the list of `City` *objects.*

Making a Simple View

Now let's view the list of `City` *objects.*

(You'll need to use the "Django" menu to run "startapp"!)

Making a Simple View

Make a view method that gets a `City` *object by its* `id`.

Making a Simple View

Make a view method that gets a `City` *object by its* `name`.

(`[A-Za-z ]+` *matches a string of letters and spaces)*

Templates

Templates

- Templates build on (are used by) Views
- HTML, with special curly brace tags:
  - `{% keyword arguments %}` – Tags
    - control flow
    - page properties
  - `{{ expression[|filter|...] }}` – Variables
    - variable/value substitution (with optional filters)
    - expression can refer to methods without using `()`

Templates

```
{% extends "base.html" %}

{% block title %}States of India{% endblock %}

{% block content %}
<ul>
{% for state in states %}
  <li><a href="/states/{{ state.id }}/">{{ state }}</a></li>
{% endfor %}
</ul>
{% endblock %}
```

Common Tags

- `{% extends "file" %}`
  - Specifies the "base" template file ("file") (e.g. including header, footer, sidebars, etc.)
- `{% block name %}...{% endblock %}`
  - In the "base" template file: default values
  - In the "extending" file: value substituted for default values.

Common Tags

- `{% if expr %}...[{% else %}...]`
  `{% endif %}`
  - If expression is true, display what comes after if
  - Otherwise, display what comes after else (if any)
- `{% for item in list %}...`
  `{% endfor %}`
  - Print contents for each item in list (item = list[i])

Common Filters

- `{{ value|upper }}{{ value|lower }}`
  - Uppercase/lowercase value
- `{{ value|default:"string" }}`
  - Print “string” if value is false or empty (otherwise value)
- `{{ value|join:", " }}`
  - Join each item in the list value with “, ”

Common Filters

- `{{ value|length }}`
  - The number of items in the list value.
- `{{ value|linebreaks }}`
  - Convert string line-breaks “\n” to HTML “
”

Common Filters

- `{{ value|urlencode }}`
  - Make value safe for a URL parameter
- `{{ value|striptags }}`
  - Strip HTML tags from the output (USE THIS!!)
- `{{ value|escape }}`
  - Escape string for HTML printing
  - Django does this automatically (a good thing)

Common Filters

- `{{ value|pluralize[:"[1,]2+"] }}`
  - Output “s” if `value != 1`.
  - If an argument is given without a comma, output the argument if `value != 1`.
  - If an argument is given with a comma, output string before comma if `value == 1`
    output string after comma if `value != 1`

Applying Templates

1. Use a loader to load the template:

```
t = loader.get_template('file')
```

2. Populate a context (like a dictionary):

```
c = Context({'tpl_var': view_var})
```

3. Render the template with the context:

```
html = t.render(c)
```

Applying Templates

```
from places.models import State
# Need the next line for templates!
from django.template import loader, Context
from django.http import HttpResponse  # Needed for HttpResponse


def list_states(request):
    state_objects = State.objects.all().order_by("name")
    t = loader.get_template('file')
    c = Context({'states': state_objects})
    html = t.render(c)
    return HttpResponse(html)
```

Applying Templates

```
from places.models import State
# Need the next line for templates!
from django.template import render_to_response

def list_states(request):
    state_objects = State.objects.all().order_by("name")
    # The following is shorthand for the four template commands:
    # t = loader.get_template('file')
    # c = Context({'states': state_objects})
    # html = t.render(c)
    # return HttpResponse(html)
    return render_to_response('file', {
        'states': state_objects,
    })
```

Making a Template

Making a Template

Make a template for the list of `City` *objects.*

Make it list them using an HTML list.

Making a Template

Add the template to your list view.

Making a Template

Make the city names in the list appear in all upper case.

Forms

Forms

- Make it easy to construct and take input
- Constructed like models, included in views
- Can be used in templates

Forms

```
from django import forms

class StateForm(forms.Form):
    name = forms.CharField(max_length=255)
    capital = forms.CharField(max_length=255)
    population = forms.IntegerField()
    area = forms.DecimalField(max_digits=11, decimal_places=2)
    languages = forms.ModelMultipleSelectField(
      queryset=languages.models.Language.objects.all())
```

Forms

- Fields often same as Models
  - BigIntegerField not supported (use IntegerField)

- Some extra fields:
  - `ChoiceField` – with `choices=["a","b","c"]` argument, generates dropdown with “a”,“b”,“c”
  - `FileField` – for file uploads

Forms

- More extra fields for relationships:
  - `ModelChoiceField` – with `queryset=...` argument, generates dropdown of models.
  - `ModelMultipleChoiceField` – with `queryset=` argument, generates multiple-selector of models.

Using Forms

- `request.method` says if form submitted (`"GET"` is a request, `"POST"` a submission)
- Create form with `request.POST`
- `form.is_valid()` is true if the form validates against the field params
- `form.cleaned_data["field"]` gets data
- Add `{{ form.as_p }}` to template

Forms

```
def new_state(request):
    if request.method == 'POST':
        form = StateForm(request.POST)
        if form.is_valid():
            state = State()
            state.name=form.cleaned_data["name"]
            # And so on for the other properties...
            state.save()
            # Redirect after POST
            return HttpResponseRedirect('/thanks/')
    else:
        form = StateForm()  # An unbound form

    return render_to_response('new_state.html', {
        'form': form,
    })
```

Templates

```
{% extends "base.html" %}

{% block title %}India – New State{% endblock %}

{% block content %}
{# csrf_token prevents session/data hijacks! #}
{# Lookup “cross-site request forgery” #}
<form action="/new_state" method="post">{% csrf_token %}
{{ form.as_p }}
<input type="submit" value="submit"/>
</form>
{% endblock %}
```

Making a Form

Making a Form

Make a form to make a new `City` *object.*

Making a Template

Add a view to add new cities.

References

- Django documentation:
  - "The Django template language" <https://docs.djangoproject.com/en/1.3//topics/templates/>
  - "Built in tags and filters" <https://docs.djangoproject.com/en/1.3/ref/templates/builtins/>
  - "Working with forms" <https://docs.djangoproject.com/en/1.3//topics/forms/>
  - "Form fields" <https://docs.djangoproject.com/en/1.3//ref/forms/fields/>