Templates and Static Content

Reindert-Jan Ekker

@rjekker http://nl.linkedin.com/in/rjekker

Overview

Model Template View

Player App with player homepage

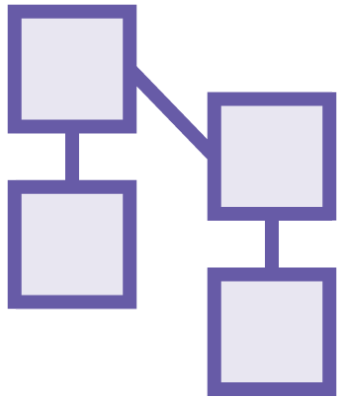
Templates

- Template Language syntax
- Calling Template from View
- Sending data from View to Template
- Displaying data using Template

Static content

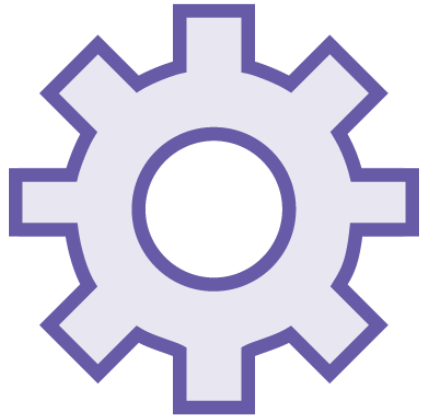
- Styling with CSS
- Using static content in templates
- Shared layout: Template inheritance
- Templates and static outside Apps

Model Template View



Model
Represents your data;
Maps model classes to database tables

Template
Generates HTML;
Presentation logic only



View
Takes HTTP request and returns response;
May call Template and/or Model

Demo

Add a “player” App

- Add a home View
- Add a URL mapping
- Add a Template

Django Templates

Can render HTML or any kind of text-based data

Templates go in `templates/` dir in your app

Best practice

- Use a dir under `templates/`
- Name it after your app
- `project/app/templates/app/templt.html`

```
def home(request):
    return render(request, “player/home.html")
```

Calling a Template

Django will search inside template folders by default

So we can omit that part of the path

Use `django.shortcuts.render`

Mapping URLs for Apps (1)

```
# urls.py inside the app
from django.conf.urls import url
from .views import home


# These urls do NOT start with a caret (^)
urlpatterns = [
    url(r'home$', home)
]
```

Mapping URLs for Apps (2)

```
# In project-wide urls.py
from django.conf.urls import url, include
urlpatterns = [
    # Include urlpatterns from player/urls.py
    # First argument: URL prefix
    url(r'^player/', include('player.urls')),
    # ...
]
```

Demo

Templates and data

- Retrieve games for user
- Show them using a Template
- Custom QueryManager
- Q()

```
def home(request):
    return render(request, “player/home.html“,
                  { message: “Hi, There”,
                    games: Game.objects.all() }
)
```

Passing Data to a Template

Pass data as a dictionary as third argument to `render()`

Data will be available in the *template context*

Context contains other data too (like logged in user)

Template Language: Variables

```
<!- Double curly braces: retrieve and output data
    From the template context -->
<ul>
  <li>
    Get data from the template context: {{ message }}
  <li>
    Access object attributes: {{ user.username }}
</ul>
```

Template Language: Tags

```
<!- Curly braces and percent signs:
    Execute a template tag -->
<ul>
  {% for g in games %}
      <li> Game id is: {{ g.id }}
  {% endfor %}
</ul>
```

```
Game.objects.filter(first_player=request.user, status="F")
```

Filter

`filter()` and `exclude()` can take multiple criteria

This will return objects that match/exclude all criteria (AND)

```
from django.db.models import Q
Game.objects.exclude(Q(status="F") | Q(status="S"))

Game.objects.filter(
    Q(first_player=request.user) & ~Q(status="F"))
```

Q()

If you need to combine criteria with OR: use `Q()` and the `|` operator

`Q()` also supports & (AND) and ~(NOT)

Custom QuerySet

```
# Define custom operations on a QuerySet
# By inheriting from models.QuerySet
class GamesQuerySet(models.QuerySet):
    def draw(self):
        return self.filter(status='D')
```

Custom Model Manager

```
# To make the custom operation available for use
# We override the manager object for our Model class
class Game(models.Model):
    objects = GamesQuerySet.as_manager()

# We can now call our custom method through Game.objects
drawed_games = Game.objects.draw()
```

Demo

Styling with static content and CSS

Using a bootstrap layout

Template inheritance

Demo

Moving content out of apps

- Fixing our welcome view
- Configure template locations
- Configure static content locations

Static Files

For non-dynamic content (CSS, JavaScript, Images)

Go in `static/` folder in your app

- Or top-level folder (needs configuration)

Files are served as-is

May be hosted separately

Static Files and HTML

```
{% load staticfiles %}
<!DOCTYPE html><html lang="en"><head>
    <link href="{% static ‘player/style.css' %}"
          rel="stylesheet">
    <script src=“{% static ‘player/script.js' %}“></script>
    <img src=“{% static ‘player/image.png' %}“>
```

Template Inheritance

```
<!-- Include all HTML from the parent template
     (This should be the first tag! -->
{% extends “base.html” %}

{% block block_name %}
   But this replaces the content of the block
   with the same name
{% endblock %}
```

Top-level Templates Folder

```
# in settings.py
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends...',
        'DIRS': [os.path.join(BASE_DIR, "templates")],
        # etc.
```

Top-level Static Folder

```
# in settings.py - add this as a new variable!
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static")
]
```

Overview

Templates

- Calling Template with `render()`
- Sending data from View to Template
- `{{ variables }}` and `{% tags %}`

Static content

- Using static content in templates
- Shared layout: Template inheritance
- Templates and static outside Apps