Estructura de proyectos en django

Estructura de la aplicación

- los típicos: models.py/urls.py/views.py/admin.py
- forms.py
- middleware.py
- context_processors.py
- managers.py
- feeds.py
- signals.py
- **init**.py <-- inicialización de signals
- tasks.py
- templates/
- templatetags/ <- nunca usar un template tag con el nombre de una aplicación
- migrations/ <- si usas south
- management/commands
- doc/ <-- :)

Reglas generales

- DRY
  - urls: url('^ruta$', view, name="url_name")
  - template : {% url url_name %}
  - views: reverse
- logging (aunque no se saque)
- aprovechar vistas genericas de django
- requirements.txt
- PEP-8
- comentarios
- test!!

Queryset - models

- __unicode__
- métodos especiales: *get_absolute_url*
- son lazy, evitar *list(query)*
- aprovechar el operador [] -> *query[:10]* => *LIMIT 10*
- *query.cont()* en vez de *len(query)*

se pueden enlazar:

```
1 query = Model.objects.filter(nombre='john')
2 if blah:
3     query = query.filter(apellido='rambo')
```

- los objetos se cachean
- usar .get() cuando se quiera un solo objeto, nunca [0]
    - recoger DoesNotExist

Views

- FAT models, intentar usar el menos código posible en las views, pasar todo a modelo
  - más testeable y mantenible
- usar RequestContext cuando sea necesario
- vistas genéricas cuando se pueda
  - template configurable
  - add/edit

Forms

- convención: form.save()
- usar help_text en los campos

Estructura del deploy

- /home/rambot/
  - app.wsgi
  - src/
    - urls.py
    - manage.py
    - ...
    - app1/
  - env/ <- entorno virtual env
  - logs/ <- apache & app logs

FIN