

Περιγραφή του EcoTruck

Νίκος Μπεζιργιάννης

Πανεπιστήμιο Μακεδονίας

16 Μαΐου 2010



- Online εφαρμογή για τη διαχείριση ανακύκλωσης και επαναχρησιμοποίησης της χάρτινης ύλης
- Η ιδέα δημιουργήθηκε για τη συμμετοχή στον Πανελλήνιο Διαγωνισμό Λογισμικού xirafia.gr
- Θέμα του Διαγωνισμού: Ψηφιακή Προστασία του Περιβάλλοντος
- 1ο Βραβείο

- Χρήση της web τεχνολογίας
- Συντονισμός της διαδικασίας
- Επιτάχυνση της διαδικασίας
- Θα μπορούν να συμμετέχουν ιδιώτες και επιχειρήσεις με μικρότερη προσφορά ύλης
- Άρα η ανακύκλωση γίνεται πιο μαζικά
- Δεν θα χρειάζονται κάδοι!

- Εταιρίες, ιδιώτες και φορτηγά μπορούν να κάνουν εγγραφή
- Δηλώνουν την τοποθεσία τους
- Αποστέλλουν τη προσφορά τους για ανακύκλωση ή κάνουν αίτηση για επαναχρησιμοποίηση
- Τα φορτηγά συλλέγουν την ύλη
- Μεταφορά της στα εργοστάσια

- Η εφαρμογή λειτουργεί online σε έναν κεντρικό server
- Real-time ενημέρωση της εξέλιξης της διαδικασίας ανακύκλωσης
- Real-time ενημέρωση του μποτιλιαρίσματος στους δρόμους
- Επιλογή από τα φορτηγά της βέλτιστης διαδρομής (με χρήση A*)

- Πτυχιακή Εργασία
- Η εφαρμογή να είναι πιο “real”
- Τα φορητά να γίνουν agents. Με αυτό τον τρόπο θα υπάρχει αυτονομία, αξιοπιστία και ταχύτητα (οι υπολογισμοί θα γίνονται client-side)
- Τα real φορητά μέσω της εφαρμογής θα στέλνουν δεδομένα για το μποτιλιάρισμα και ο server θα εξάγει στατιστικά στοιχεία

- Αρκετή Python (simplejson , python-graph)
- Λίγη Javascript
- Django
- SQLite
- AJAX
- JSON
- Google Maps API

- Αρκετή Python (simplejson , python-graph)
- Λίγη Javascript
- Django
- SQLite
- AJAX
- JSON
- Google Maps API

django

- Web framework γραμμένο στην Python
- Ακολουθεί την αρχή DRY (Don't Repeat Yourself)
- Τα Django projects χρησιμοποιούν το μοτίβο MTV (Model Template View)
- Προσφέρει έτοιμες πακεταρισμένες εφαρμογές για reuse
- Πολλές επεκτάσεις με τη χρήση middleware κλάσεων

Κάποια βασικά μέρη-συστατικά του:

- ORM (Object-Relational Mapper)
- Admin interface
- URL dispatcher βάση κανονικών εκφράσεων
- Template σύστημα (Jinja)
- Cache σύστημα

Τι είναι το MTV;

- Βασικά είναι πιο γνωστό ως MVC (Model View Controller)
- Το Django το ονομάζει MTV (Model Template View)
- Κάθε web εφαρμογή που γράφουμε - ανεξαρτήτως framework - μπορεί να χωριστεί σε 3 βασικά μέρη τα models , views και templates

- Τα δεδομένα της εφαρμογής θα αποθηκεύονται σε κάποια βάση δεδομένων (δεν μας ενδιαφέρει τι είδους βάση)
- Αλλά θα αναπαριστώνται με τον ίδιο ακριβώς τρόπο
- Η μοντελοποίηση είναι αντικειμενοστρεφής
- Έτσι δημιουργείται μια εικονική βάση αντικειμένων

Listing 1: Αρχείο models.py

```
class Post(models.Model):  
    author = models.CharField(max_length=100)  
    title = models.CharField(max_length=30, unique=True)  
    text = models.CharField(max_length=1000)  
    date = models.DateField()  
    active = models.BooleanField(default=True)
```

- Δημιουργεί τη σύνδεση μεταξύ της εικονικής βάσης αντικειμένων και της σχεσιακής βάσης δεδομένων μας (Object-Relational Mapping)
- Κλάση \leftrightarrow Πίνακας
- Attributes \leftrightarrow Πεδία
- Αντικείμενα \leftrightarrow Καταχωρήσεις

Listing 2: ORM - Παράδειγμα

```
class Post(models.Model):
    author = models.CharField(max_length=100)
    title = models.CharField(max_length=30, unique=True)
    text = models.CharField(max_length=1000)
    date = models.DateField()
    active = models.BooleanField(default=True)

BEGIN;
CREATE TABLE "main_post" (
    "id" integer NOT NULL PRIMARY KEY,
    "author" varchar(100) NOT NULL,
    "title" varchar(30) NOT NULL UNIQUE,
    "text" varchar(1000) NOT NULL,
    "date" date NOT NULL,
    "active" bool NOT NULL
);
COMMIT;
```

Listing 3: ORM - Παράδειγμα

```
from main.models import Post
from datetime import date
P1 = Post("author1","title1","text here", date.today())
P1.title = "New Title"
p1.save()
p1.author = "New Author"
p1.save()

INSERT INTO "main_post" VALUES
(1,"author1","title1","text here","2010-04-16",1);

UPDATE "main_post" SET "author"="New Author"
WHERE "id"=1;
```

Listing 4: ORM - Παράδειγμα

```
from main.models import Post
all_posts = Post.objects.all()
oldest_ten_posts = Post.objects.all().order_by("date")[:10]
active_posts = Post.objects.filter(active=True)
nicks_newest_posts = Post.objects.filter(author="Nick").order_by(-date)
some_post = Post.objects.get(title="Nice Title")

SELECT * FROM "main_post";
SELECT * FROM "main_post" ORDER BY "date" LIMIT 10;
SELECT * FROM "main_post" WHERE "active"=1;
SELECT * FROM "main_post" WHERE "author"="Nick"
ORDER BY DATE DESC;
SELECT * FROM "main_post" WHERE "title"="Nice Title";
```

- Είναι html αρχεία συνήθως
- Τα οποία περιέχουν κενά
- Η template μηχανή γεμίζει αυτά τα κενά με δεδομένα που τις δίνουμε (επεξεργασμένα δεδομένα από την βάση)
- Template inheritance
- Έχει δικιά της σύνταξη , μοιάζει αρκετά με Python

Listing 5: Αρχείο main.html

```
<html>
  <head><title>Super Blog</title></head>
  <body>
    <h1>Welcome to the Super Blog</h1>
    {% block main %}
    {% endblock %}
  </body>
</html>
```

Listing 6: Αρχείο main_home.html

```
{% extends main.html %}
{% block main %}
{% for post in posts %}
<h1>{{ post.title }}</h1>
<h3><i>{{ post.author }}</i>, {{ post.date }}</h3>
<p>{{ post.text }}</p>
{% endfor %}
{% endblock %}
```

- Python συναρτήσεις
- Δέχονται ως είσοδο την HTTP αίτηση μαζί με παραμέτρους και επιστρέφουν την HTTP απάντηση
- Επεξεργάζονται τα δεδομένα της βάσης μέσω των models και τα δίνουν στην template μηχανή για να τα προβάλλει με τον επιθυμητό τρόπο

Listing 7: Αρχείο views.py

```
from django.http import HttpResponseRedirect
from django.shortcuts import render_to_response
def homepage(req):
    templ = "main_home.html"
    return render_to_response(templ, {
        "posts": Post.objects.all().order_by("-date")[:5]
    })

def postpage(req, post_id):
    templ = "main_post.html"
    return render_to_response(templ, {
        "post": Post.objects.get(id=post_id)
    })
```

- Πρόκειται για μία association list που συνδέει κανονικές εκφράσεις διευθύνσεων με τα views
- Όταν γίνεται μια HTTP αίτηση ο dispatcher προσπαθεί να κάνει match τη διεύθυνση της αίτησης με κάποια από τις κανονικές εκφράσεις της λίστας σύμφωνα με την προτεραιότητά τους
- Αν γίνει match τότε εκτελείται το view που ταιριάστηκε
- Αν δεν γίνει match επιστρέφεται 404

Listing 8: Αρχείο urls.py

```
urlpatterns = patterns("",
    (r"^admin/", include(admin.site.urls)),
    (r"^$", "blog.main.views.homepage"),
    (r"^post/(\d+)$", "blog.main.views.postpage"),
)
```

<http://djangoproject.com>

<http://djangobook.com>

<http://djangosnippets.org>

Ευχαριστώ! Ερωτήσεις;