Desenvolupament ràpid d'aplicacions amb Django - I (8195 lectures)

Per **Antoni Aloy López**, ***aaloy*** (http://trespams.com)

Creado el 29/12/2005 00:57 modificado el 29/12/2005 01:38

Django[1] és un bastiment per al desenvolupament ràpid d'aplicacions web fet amb i per Python.[2].

Aquest bastiment és un dels més potents que m'he trobat i a l'hora també un dels més elegants i simples. Com que la millo provar-ho és fent-hi coses he començat a desenvolupar una mini comptabilitat seguint el tutorial de Django i adaptant-ho a necessitats. El que he fet ho he anat posant a un wiki per a us propi, i ara el que us present és un volcat d'aquest wiki en el primera part d'una sèrie d'articles que han d'acabar amb una mini-comptabilitat.

Introducció

L'objectiu d'aquest tutorial és proporcionar un punt de partida per a la utilització de Django[1] com a eina per al desenvolupament d això el que farem serà començar a construir una mini-aplicació comptable a la que anomenarem **bulconta**.

L'aplicació serà prou senzilla per a que es pugui completar a un tutorial però a la vegada esper que sigui prou complexa com per ve Django a l'hora de desenvolupar aplicacions de gestió.

Per fer aquesta aplicació aniré seguint el tutorial de la pàgina de Django[3] encara que m'ho aniré botant o ampliant quan ho conside

La primera cosa que hem de fer és instal·lar Django, una manera molt senzilla i recomanada a la propia documentació és baixar-no de Subversion i fer un link cap a *site-packages* de la nostra instal·lació de Python. La instal·lació és força senzilla:

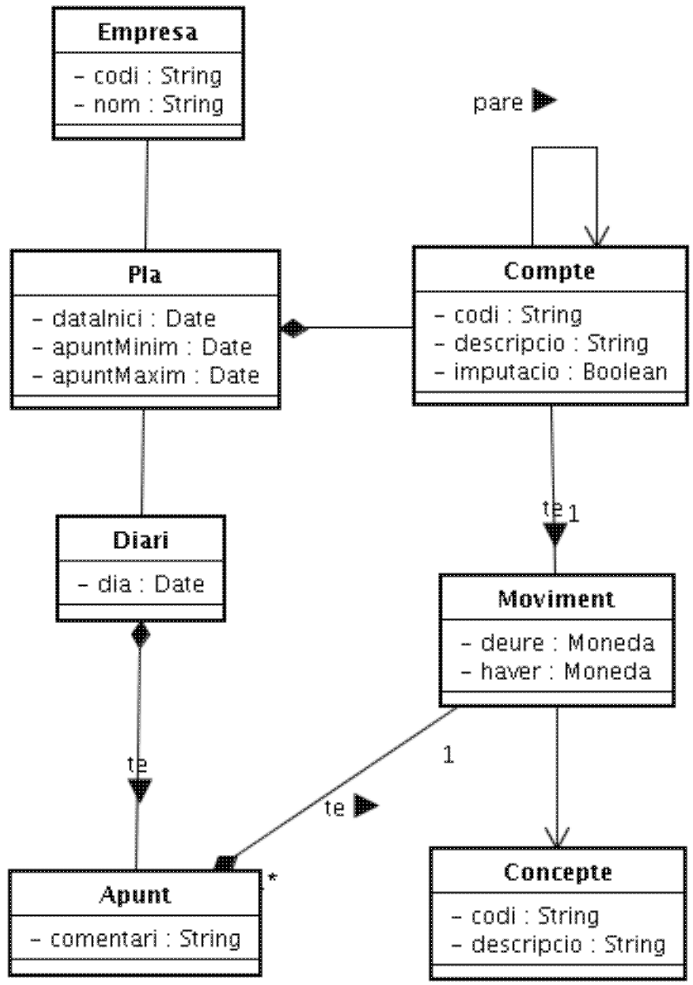
1. Instal·lar Subversion si no ho tenin
2. Desde el directori on volguem deixar el codi feim svn co http://code.djangoproject.com/svn/django/trunk/ django_src
3. Cream un enllaç simbòlic des del directori django_src/django a directori site-packages de la nostra instal·lació de Python a
4. També és recomanable fer un enllaç simbòlic de l'arxiu django-admin, per exemple

```
>> sudo ln -s /usr/lib/python2.4/site-packages/django/django/bin/django-admin.py /usr/local/bin/django-ad
```

Django ve amb el seu propi servidor web encara que sols es recomana per al desenvolupament. Quan posem l'aplicatiu en producci *mod_python* per exemple.

Definició del problema

Volem crear una minicomptabilitat que serà més o manco funcional però que hem de tenir ben clar que no és el que hom podria esp comptabilitat "com cal". Senzillament ho farem servir perquè el model ja me'l conec prou i puc adaptar-ho sense problemes. Farem



en comptes d'un model més ric com podria ser aquest[4]

Anem a descriure un poc el model.

La nostra comptabilitat serà multiempresa i cada empresa podrà tenir definit un pla de comptes. El pla de comptes ens defineix la c
format per les comptes comptables. La comptabilitat es fa a partir per apunts, que no són més que un conjunt de moviments de tal r
les quantitats del deure i les de l'haver són iguals. Els apunts es guarden dins el llibre diari donant-los una data de comptabilització

Cada moviment fa referència a una compte comptable i tendrà un concepte associat.

Per fer la cosa més interessant farem que un compte pugui estar actiu, bloquejat o amb marca d'avís. El primer és el valor per defec
permetrà apunts i el tercer estat permetrà apunts però avisarà a l'usuari.

El pla de comptes segueix una estructura jeràrquica, es a dir, llevat de les compte de primer nivell (1 fins a 9) la resta són filles del
comptes que poden rebre apunts s'anomenen d'imputació i el nivell normalment ve determinat pel propi pla comptable. Per ara no a
tema, podeu trobar un petit anàlisi a Hispalinux[5] i n'hauria d'haver un en format pdf a algún lloc de la xarxa...

Cream l'estructura de l'aplicació

Django necessita que els objectes estiguis creats amb una estructura de directoris concreta amb un noms d'arxiu determinats. Això
especificacions de Sun pels servlets per exemple. La diferència és que Django proporciona un mecanisme automàtic per crear aque
esquelet que ens permet començar a fer feina.

```
  >>> django-admin.py startproject bulconta
```

Aquesta instrucció ens crea el projecte bulconta

```
aaloy@G5:~/devel$ du -a -h bulconta
4,0K    bulconta/manage.py
0       bulconta/__init__.py
4,0K    bulconta/settings.py
4,0K    bulconta/urls.py
4,0K    bulconta/__init__.pyo
4,0K    bulconta/manage.pyo
4,0K    bulconta/settings.pyo
4,0K    bulconta/urls.pyo
0       bulconta/apps/__init__.py
4,0K    bulconta/apps/__init__.pyo
8,0K    bulconta/apps
40K     bulconta
```

La primera cosa que hem de fer és editar l'arxiu *settings.py* i adaptar la configuració a la nostra base de dades i a la codificació que
meu cas faré servir Postgres com a base de dades, amb l'usuari *bulconta* i el password *bulconta*. Així

```
DATABASE_ENGINE = 'postgresql' # 'postgresql', 'mysql', 'sqlite3' or 'ado_mssql'.
DATABASE_NAME = 'bulconta'             # Or path to database file if using sqlite3.
DATABASE_USER = 'bulconta'             # Not used with sqlite3.
DATABASE_PASSWORD = 'bulconta'         # Not used with sqlite3.
DATABASE_HOST = 'localhost'             # Set to empty string for localhost. Not used with sqlite3.
DATABASE_PORT = ''             # Set to empty string for default. Not used with sqlite3.
```

A més per la codificació i la localització canviarem:

```
TIME_ZONE = 'unknown' # millor que l'agafi per defecte
LANGUAGE_CODE = 'es-es'
```

Per tal de fer les proves necessitarem que l'estructura creada estigui al nostre path de Python, per això podem fer simplement

```
>> export PYTHONPATH=.:$PYTHONPATH
```

desde el directori superior del nostre projecte. Això ens permetrà importar el paquet *bulconta*. des de la línea de comandaments

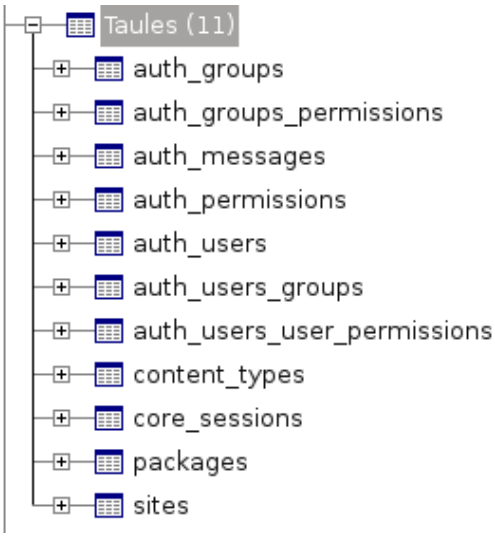
Ara el que farem és inicialitzar la nostra base de dades amb les estructures que Django necessita, però abans farem un pas previ per
gestió de projectes quin és el nostre projecte. Així desde el directori superior a bulconta farem

```
>> export DJANGO_SETTINGS_MODULE=bulconta.settings
```

La qual cosa ens ha de permetre fer

```
>> django-admin init
```

Si tot ha anat bé ja tindrem la nostra base de dades inicialitzada amb les taules que es mostren a la figura



Definició del nostre model

Per crear el nostre model anirem dins el directori *bulconta/apps* i teclejarem

```
>> django-admin startapp bulconta
```

Això ens crea un paquet Python amb els esquelets dels objetes que farem servir per a la nostra aplicació. El que ara ens interessa, e
models/bulconta.py. El que farem és editar aquest arxiu amb el nostre editor preferit i modelar les nostres classes de negoci.

L'order de definició importa És a dir, primer hem de definir les dependències abans que les classes que les fan servir.

Per començar una de fàcil, la definició d'empresa

```
class Empresa(meta.Model):
    """Defineix l'empresa que fara servir la comptabilitat"""
    codi = meta.CharField(maxlength=12, unique=True, blank=False, db_index=True)
    nom = meta.CharField(maxlength=100, blank=False)
```

Definim empresa com una classe filla de meta.Model i començam a definir els camps. Com podeu veure el codi s'explica per ell ma
Python. La única cosa rellevant és el fet que automàgicament Django ens crearà l'identificador i la sequència associada.

Per complicar-ho un poc més he definit codi com a unic, que no es pot deixar en blanc a l'hora d'editar-ho i com que sé que s'hi acc
doncs ja defineixo un índex damunt el camp.

La definició del pla comptable és també prou senzilla

```
class Pla(meta.Model):
    """Defineix el pla comptable lligat a l'empresa
        dataInici és la data de a partir de la qual considerarem obert l'any
        apuntMinim és la data mínima que pot tenir un apunt per ser vàlid
        apuntMaxim és la data màxima que pot tenir un apunt per ser vàlid
    """
    empresa = meta.OneToOneField(Empresa)
    dataInici = meta.DateField(blank=False)
    apuntMinim = meta.DateField()
    apuntMaxim = meta.DateField()
```

Aquí hem introduït dos conceptes nous, el camp *DateField* que ens serveix per definir dates i com mapejan una relació un a un fen
meta.OneToOneField

Anem a crear l'estructura de taules del que hem fet fins ara i anam a veure què podem fer des de la línea de comandes. Primer anem model genera codi SQL i quin codi genera. Per això guardarem el que hem fet i direm a Django que incorpori l'aplicació que acaba que sols sigui el model l'estructura és de tota una aplicació) a la configuració. Per això editam *settings.py* de manear que quedi

```
INSTALLED_APPS = (
    'bulconta.apps.bulconta',
)
```

Si hem arribat fins aquí

```
>> django-admin sql bulconta
BEGIN;
CREATE TABLE "bulconta_empresas" (
    "id" serial NOT NULL PRIMARY KEY,
    "codi" varchar(12) NOT NULL UNIQUE,
    "nom" varchar(100) NOT NULL
);
CREATE TABLE "bulconta_plas" (
    "empresa_id" integer NOT NULL PRIMARY KEY REFERENCES "bulconta_empresas" ("id"),
    "dataInici" date NOT NULL,
    "apuntMinim" date NOT NULL,
    "apuntMaxim" date NOT NULL
);
COMMIT;
```

Si volem veure els indexs que es crearan ho podem fer

```
>> django-admin sqlindexes bulconta
CREATE UNIQUE INDEX bulconta_empresas_codi ON "bulconta_empresas" ("codi");
CREATE INDEX bulconta_plas_empresa_id ON "bulconta_plas" ("empresa_id");
```

Com es pot comprovar Django ens dona la feina de crear l'SQL que d'adapti a la nostra base de dades. Els plurals catalans no van m arribarà :) . Una opció seria definir els noms de les classes en anglès però com que tanmateix no tocarem massa les taules és millor més autodocumentat millor. Ara el que farem és crear aquestes estructures dins la base de dades. Ho podem fer manualment o bé d faci per nosaltres. Triaré la darrera opció:

```
>> django-admin install bulconta
```

Ara ja podriem començar a fer feina amb el nostre model. Anem a a línea de comandaments i executem Python

```
>>> from django.models.bulconta import empresas, plas
>>> empresas.get_list()
[]
>>> empresa = empresas.Empresa(codi='000', nom="Empresa de proves")
>>> empresa.save()
>>> empresa.id
1L
```

El que hem fet aquí es primer de tot importar les classes que farem servir. Django automàticament crea les classes a partir de la def afegint una s al nom de la classe que hem creat a la definició del nostre model. És també el que fa amb les taules així que no costa r Una de les característiques més sorprenents del bastiment és que es capaç de generar els noms de les funcions al vol, en tenim una com s'ha construït l'objete empresa, si ho recordau no hem necessitat definir cap constructor.

Entrant en la part de persistència veim que crear una nova empresa és realment senzill:

1. Cream l'objecte
2. Guardam l'objecte

Una vegada hem guardat l'objecte aquest agafa l'identificador (l'id) que per cert també ha creat Django per nosaltres.

Ara crearem un pla comptable i ho assignarem a l'empresa que acabam de crear. Aquí hem de fer un poc de feina amb el tractamen però res de l'altre mon

```
>>>import datetime
>>>data = datetime.datetime(2005,12,28,0,0)
>>>dataMaxima = datetime.datetime(2005,1,31,0,0)
>>>pla = plas.Pla(empresa=empresa, dataInici=data, apuntMinim=data, apuntMaxim=dataMaxima)
>>>pla.save()
```

En aquest punt és interessant veure de quines funcions i atributs disposam per l'objecte *pla*, per això basta fer

```
dir(pla)
['__class__', '__delattr__', '__dict__', '__doc__', '__eq__', '__getattribute__', '__hash__', '__init__',
'__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__str__', '
'_empresa_cache', '_meta', 'apuntMaxim', 'apuntMinim', 'dataInici', 'delete', 'empresa_id', '
get_empresa', 'get_next_by_apuntMaxim', 'get_next_by_apuntMinim', 'get_next_by_dataInici',
'get_previous_by_apuntMaxim', 'get_previous_by_apuntMinim', 'get_previous_by_dataInici', 'save']
```

Seguirem fent feina amb la línea de comandes un poc més tard. Ara el que farem serà arreglar un poc aquestes dues classes que her poguem fer la representació textual (el típic toString() de Java per exemple i afegirem un poc de la màgia de Django, seguirem els que ens indica la segona part del tutorial de Django tutorial2[6]

- A l'arxiu settings.py a INSTALLED_APPS hi afegim "django.contrib.admin", la cosa ha de quedar així:

```
INSTALLED_APPS = (
    'bulconta.apps.bulconta',
    'django.contrib.admin',
)
```

- Des de la línea de comandaments executam *django-admin.py install admin.* per tal que es crein les taules de l'aplicació d'a
- Editam bulconta/urls.py i descomentam la línea inferior a "Uncomment this for admin:"
- Cream el superusuari amb

```
django-admin createsuperuser
```

- Per tal de seguir amb el tutorial hem creat el superusuari bulconta amb clau bulconta, sóm així d'originals

Això ja ens permetria executar el servidor web de Django i anar al mòdul d'administració. Per fer les coses més interessants, però a nostre model per a que pugui figurar dins el modul d'administració. Això es fa afegint una classe a

```
class META:
        admin = meta.Admin()
```

De manera que tindriem

```
class Empresa(meta.Model):
    """Defineix l'empresa que fara servir la comptabilitat"""
    codi = meta.CharField(maxlength=12, unique=True, blank=False, db_index=True)
    nom = meta.CharField(maxlength=100, blank=False)
    def __repr__(self):
        return "%s\t%s" % (self.codi, self.nom)

    class META:
        admin = meta.Admin()

# Pla comptable
class Pla(meta.Model):
    """Defineix el pla comptable lligat a l'empresa
    dataInici és la data de a partir de la qual considerarem obert l'any
    apuntMinim és la data mínima que pot tenir un apunt per ser vàlid
    apuntMaxim és la data màxima que pot tenir un apunt per ser vàlid
    """
    empresa = meta.OneToOneField(Empresa)
    dataInici = meta.DateField(blank=False)
    apuntMinim = meta.DateField()
    apuntMaxim = meta.DateField()
```

```
    def __repr__(self):
        return "Pla comptable de %s" % self.get_empresa().codi

    class META:
        admin = meta.Admin()
```

Fixem-nos com __repr__ defineix la representació textual. Mentre retorni texte hi podem posar el que volguem, ei! millor que té al l'objecte!

Obrim una nova consola, ens situam al directori superior del nostre projecte, establim les variables d'entorn i ja podem executar el

```
>>export DJANGO_SETTINGS_MODULE=bulconta.settings
>>export PYTHONPATH=.:$PYTHONPATH
>>django-admin.py runserver
```

Si tot a anat bé

```
http://localhost:8000/
```

ens ha de presentar la pantalla d'entrada, ens autenticam i ja podrem veure que Django ens ha creat els manteniments d'usuaris (de manteniments. Haurem de tocar algunes cosestes per a deixar-ho bén polit i funcional però com podeu veure anam per bon camí.

Lista de enlaces de este artículo:

1. http://www.djangoproject.org
2. http://www.python.org
3. http://www.djangoproject.com/documentation/tutorial1/
4. http://bulma.net/~aaloy/django/Bulconta.png
5. http://wiki.hispalinux.es/moin/M_f3dulodeContabilidad
6. http://www.djangoproject.com/documentation/tutorial2/

E-mail del autor: aaloy _ARROBA_ bulma.net