Django

O que é Django?

É um framework para aplicações web, escrito em python. Pode ser comparado ao TurboGears e Ruby on Rails no tocante ao rápido desenvolvimento que esses frameworks oferecem. Entretando a sua filosofia de desenvolvimento é bem diferente se comparado aos outros dois frameworks.

Por que usar?

Porque é simples de programar, tudo no Django é extremamente simples de se fazer.
A maioria dos problemas que você encontra no desenvolvimento de aplicações para web já tem alguma solução no Django, e o que não é tão genérico assim é facilmente programável usando todo o poder e facilidade da linguagem python.

Características

- Mapeamento objeto-relacional**:** os modelos são definidos em Python e possuem uma API que facilita a manipulação dos dados. Apesar disso é possível escrever comandos SQL caso seja necessário.
- Interface de administração: gera facilmente interfaces para inserção e modificação de dados (CRUD, Scaffolding)
- Urls fáceis: usando-se o recursode reescrita de URLs facilita o acesso às aplicações.
- Sistema de templates: possui uma linguagem de template para separar a lógica de programação do design.
- Sistema de cache: aumento de performance pois as páginas são geradas uma vez e acessadas do cache
- Internacionalização: possui suporte a desenvolver aplicações de forma a facilitar o processo de internacionalização.

Instalação

Para instalar o Django em seu sistema operacional é preciso os seguintes requisitos:

- Python 2.3 ou superior
- psycopg (PostgreSQL) ou MySQLdb (MySQL) ou pysqlite (Sqlite)

Em termos práticos, você precisa ter o python instalado e um adaptador para o seu banco de dados.

Instalação no Windows

Nossa primeira visita vai ser o site do python, na sessão de downloads baixe a versão mais nova do python compilada para o windows (geralmente a opção python 2.x windows installer). Para instalar não tem nenhum problema, basta seguir o famoso next, next, next.
Agora você tem o python instalado na pasta C:\Python24.

Próxima visita é o site do MySQLdb, baixe o binário para windows da versão 2.x, lembre-se de pegar o binário para a versão 2.4 do python. Nenhum segredo para instalar, basta seguir o mesmo modelo do python installer.

Baixe o Django. Descompacte o conteúdo do .tar.gz em um diretório e abra o prompt do windows (iniciar->executar->cmd) e entre no diretório onde foi descompactado o Django, digite **python setup.py install** para instalar (é necessário uma conexão com a internet para baixar uma dependência).

Se tudo der certo o Django vai estar instalado na pasta C:\Python24\Lib\site-packages\django

Instalação no Linux

Eu utilizarei a distribuição Ubuntu nessa série de artigos.

A instalação no Linux é mais fácil, isso porque praticamente todas as distribuições já trazem o python instalado, assim não vai ser preciso cobrir a instalação do python. Quanto a conexão com o MySQL é muito fácil de instalar, apenas rode o comando **sudo apt-get install python-mysqldb** em um terminal.

No CentOS e Fedora é executar o comando **yum install MySQL-python**

A instalação do Django 0.96 segue os mesmo passos da instalação no windows:

```
tar xfvz Django-0.96.tar.gz
cd Django-0.96
python setup.py install
```

Agora temos o Django instalado pronto para operar com o banco de dados MySQL

Com todos os programas instalados no sistema é hora de começar nosso primeiro projeto usando o framework **Django**.

Criando o projeto

django-admin.py

O script `django-admin.py` é copiado para o diretório /usr/bin quando instalado no Linux, se você usar o windows coloque esse arquivo dentro de algum diretório visível a variável PATH do sistema.

Criando nosso projeto:

```
django-admin.py startproject aula
cd aula/
ls
__init__.py  manage.py  settings.py  urls.py
```

Explicando melhor o comando:

`django-admin.py`
: Esse é o script responsável por tarefas administrativas do Django.

`startproject`
: Esse parâmetro avisa ao script `django-admin.py` que será criado um novo projeto.

`aula`
: O nome do novo projeto, um novo diretório com esse mesmo nome é criado.

Agora temos os seguintes arquivos dentro do diretório aula:

- `__init__.py` Arquivo que indica ao Python que esse diretório vai ser considerado um pacote.
- `manage.py` Script semelhante ao `django-admin.py`, para tarefas administrativas do seu projeto.
- `settings.py` Configurações do projeto.
- `urls.py` Configurações de url para o projeto.

Servidor web

O Django vem com um pequeno servidor web embutido para ajudar no desenvolvimento.

Iniciando o servidor:

```
python manage.py runserver
Validating models...
0 errors found.

Django version 0.96, using settings 'aula.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Acesse a url htpp://localhost:8000/ e confira a página de boas vindas do Django.

Configurando o projeto

Agora chegou a hora de configurar o seu projeto, abra o arquivo `settings.py` no seu editor preferido e confira as mudanças que eu fiz no meu projeto.

`settings.py` padrão(Apenas as partes queeu vou editar):

```
DATABASE_ENGINE = 'mysql'           # 'postgresql_psycopg2', 'postgresql',
'mysql', 'sqlite3' or 'ado_mssql'.
DATABASE_NAME = 'django'             # Or path to database file if using
sqlite3.
DATABASE_USER = 'root'             # Not used with sqlite3.
DATABASE_PASSWORD = ''         # 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.
.....
LANGUAGE_CODE = 'pt_br'
.....
TEMPLATE_DIRS = (
    # Put strings here, like "/home/html/django_templates" or
"C:/www/django/templates".
    # Always use forward slashes, even on Windows.
    # Don't forget to use absolute paths, not relative paths.
    '/var/www/aula/aula/templates',
)
....
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.admin',
    'aula.meuslivros'
)
....
```

As mudanças foram:

- `DATABASE_ENGINE` para mysql.
- `DATABASE_NAME` para django
- `DATABASE_USER`, `DATABASE_PASSWORD` e `DATABASE_HOST` para as configurações da base de dados.
- `LANGUAGE_CODE` para pt-br (Português do Brasil).

Outra configuração importante são as últimas linhas:

```
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
)
```

Cada entrada do `INSTALLED_APPS` é um aplicativo para seu projeto:

- `django.contrib.auth` Sistema de autenticação.
- `django.contrib.contenttypes` Framework para content-types.
- `django.contrib.sessions` Framework para sessões.
- `django.contrib.sites` Framework para configurar vários sites com apenas uma instalação do Django.

Por exemplo, se você não quiser o sistema de autenticação que vem com o Django basta comentar a linha e desenvolver o seu próprio. E assim por diante, nos meus documentos eu sempre uso todos esses aplicativos. Adicionamos também as duas últimas linhas, que indicam a utilização da interface de administração e a utilização do aplicativo que iremos desenvolver. Cada novo aplicativo criado dentro do projeto deve ser adicionado aqui.

Para mais informações sobre as configurações visite a documentação

Criando seu primeiro aplicativo

Como visto anteriormente, um projeto é composto de uma série de aplicativos, isso é uma característica que torna o Django bastante modular.

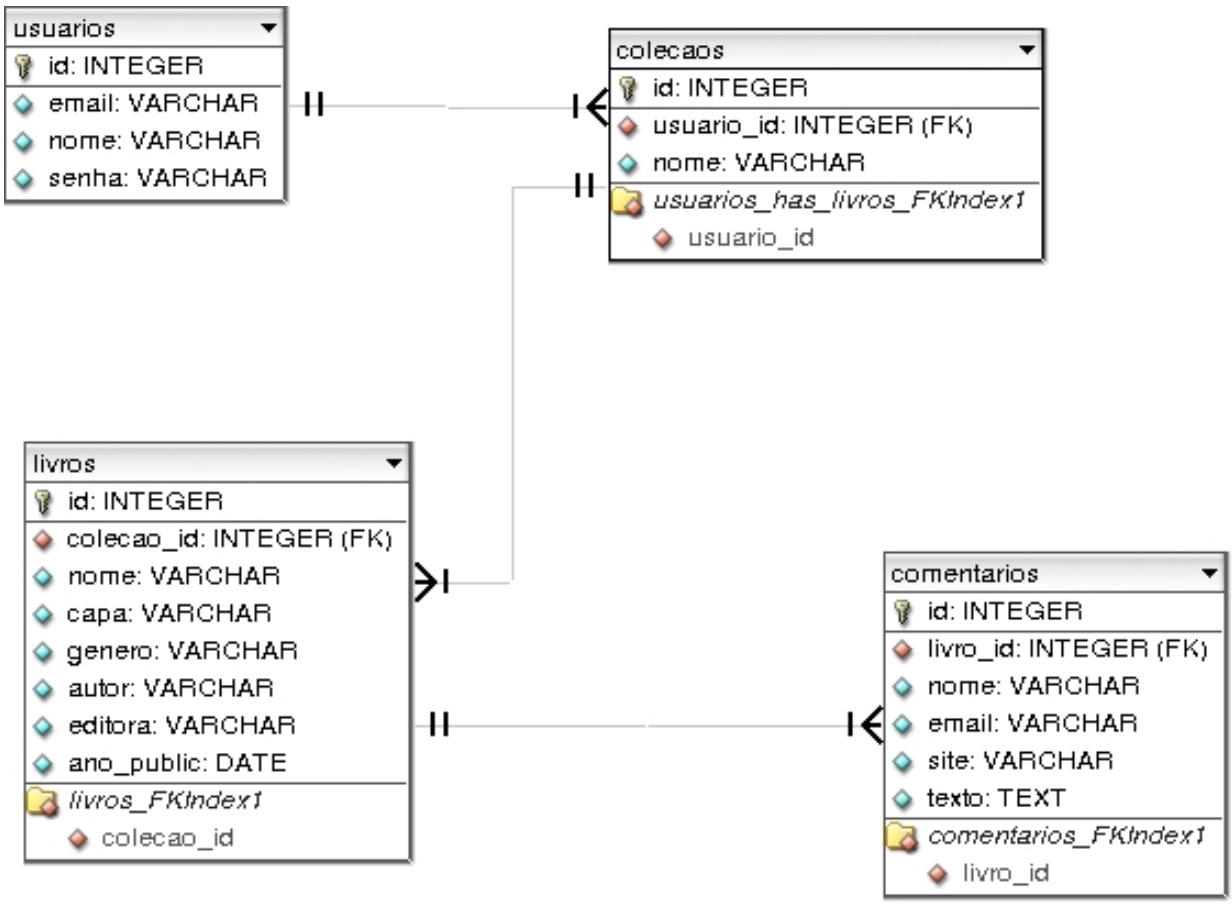
Criando seu primeiro aplicativo:

```
python manage.py startapp meuslivros
ls meuslivros/
__init__.py  models.py  views.py
```

Foi criado um novo diretório com o mesmo nome do aplicativo, e dentro desse diretório temos os seguintes arquivos:

- `__init__.py` Arquivo que indica ao Python que esse diretório vai ser considerado um pacote.
- `models.py` Arquivo onde você define seus modelos.
- `views.py` Arquivo onde você define as views do seu projeto.

Vamos escrever nosso modelo. Ele vai refletir a seguinte modelagem:



A única alteração é que iremos criar os modelos no singular. Para isto devemos editar o arquivo meuslivros/models.py:

```
from django.db import models

class Usuario(models.Model):
    class Admin:
        pass
    def __str__(self):
        return self.email
    nome =  models.CharField(maxlength=100)
    email = models.CharField(maxlength=100)
    senha = models.CharField(maxlength=100)

class Colecao(models.Model):
    class Admin:
        pass
    def __str__(self):
        return self.nome
    usuario_id = models.ForeignKey(Usuario)
    nome = models.CharField(maxlength=100)

class Livro(models.Model):
    class Admin:
        list_display = ('colecao_id','nome', 'capa', 'genero','autor','editora')
        list_filter = ['ano_public']
        search_fields = ['nome']
    def __str__(self):
        return self.nome
    colecao_id = models.ForeignKey(Colecao)
    nome = models.CharField(maxlength=100)
    capa = models.CharField(maxlength=100)
```

```
    genero = models.CharField(maxlength=100)
    autor = models.CharField(maxlength=100)
    editora = models.CharField(maxlength=100)
    ano_public = models.DateField('Ano de publicacao')

class Comentario(models.Model):
    class Admin:
        pass
    def __str__(self):
        return self.nome
    livro_id = models.ForeignKey(Livro)
    nome = models.CharField(maxlength=100)
    email = models.CharField(maxlength=100)
    site = models.CharField(maxlength=100)
    texto = models.TextField(maxlength=255)
```

Antes que você se desespere, leia a documentação model_api.

Deu para perceber que cada modelo é uma classe em Python, o método __str__ foi definido para que o nome do objeto seja retornado. A classe interna chamada Admin que foi definida em cada classe significa que ela será visualizada pela interface de administração.

Temos um também relacionamentos entre as tabelas, o que é indicado pelo método models.ForeignKey. Estes relacionamentos serão criados nas tabelas no banco de dados.

Antes de criar as tabelas é necessário criar a base de dados no Mysql:

mysql -uroot

mysql> create database django;

mysql>exit

Agora criando as tabelas:

```
python manage.py syncdb
Creating table auth_message
Creating table auth_group
Creating table auth_user
Creating table auth_permission
Creating table django_content_type
Creating table django_session
Creating table django_site
Creating table django_admin_log
Creating table meuslivros_colecao
Creating table meuslivros_livro
Creating table meuslivros_comentario
Creating table meuslivros_usuario

You just installed Django's auth system, which means you don't have any
superusers defined.
Would you like to create one now? (yes/no): yes
Username (Leave blank to use 'elm'): admin
E-mail address: eminetto@gmail.com
Password:
Password (again):
Superuser created successfully.
Installing index for auth.Message model
Installing index for auth.Permission model
Installing index for admin.LogEntry model
Installing index for meuslivros.Colecao model
Installing index for meuslivros.Livro model
Installing index for meuslivros.Comentario model
Loading 'initial_data' fixtures...
No fixtures found.
```

Todas as tabelas são criadas na nova base de dados. Também é identificado que não existe um usuário de administração por isso é solicitado sua criação.

Para testar precisamos alterar mais um arquivo, o urls.py e alterar a linha:

```
#     (r'^admin/', include('django.contrib.admin.urls')),
```

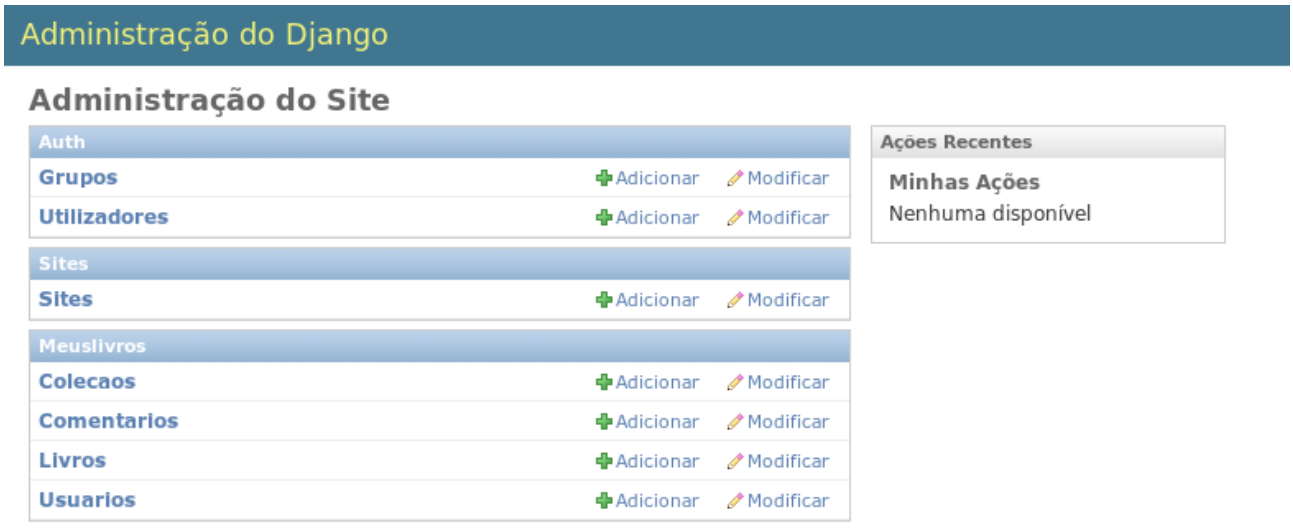
para

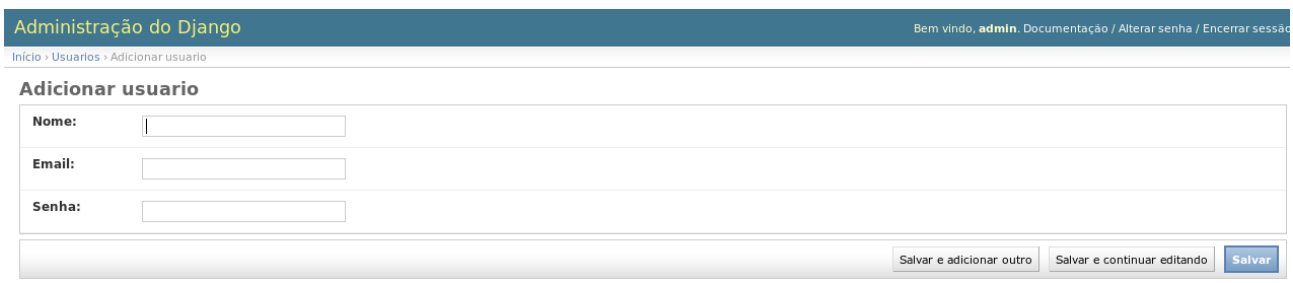
```
    (r'^admin/', include('django.contrib.admin.urls')),
```

Depois é só reiniciar o servidor, parando com CTRL+C e rodando novamente

```
python manage.py runserver
```

E acessando a url http://localhost:8000/admin no navegador:





Na interface de administração podemos criar novos usuários e grupos com permissão de manipular os dados nas tabelas.

Criando uma página inicial

Vamos criar uma página inicial onde sejam apresentados os dados já cadastrados na tabela.

Para isto precisamos inicialmente configurar a nova aplicação na lista de URLs reconhecidas pelo projeto. Vamos alterar o arquivo urls.py e deixá-lo da seguinte forma:

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    # Example:
    # (r'^aula/', include('aula.foo.urls')),
     (r'^index/$', 'aula.meuslivros.views.index'),
    # Uncomment this for admin:
     (r'^admin/', include('django.contrib.admin.urls')),
)
```

A última linha, a da interface de administração já havíamos alterado, só adicionamos a quinta linha para indicar que o endereço index/ irá executar a função chamada index, contida no arquivo aulas/meuslivros/views.

Vamos criar esta função. No arquivo meuslivros/views iremos adicionar o seguinte conteúdo:

```
# Create your views here.
from django.http import HttpResponse
from aula.meuslivros.models import Colecao
from django.template import Context, loader

def index(request):
    lista = Colecao.objects.all() #busca as colecoes
    t = loader.get_template('index.html') #indica o template
    c = Context ({'lista': lista}) #cria uma variável "lista" para o template
    return HttpResponse(t.render(c)) #renderiza o template
```

Precisamos agora criar o diretório dos templates e dentro deste diretório o arquivo index.html. O caminho do diretório de templates foi configurado no settings.py.

```
mkdir /var/www/aula/aula/templates
```

O django possui uma linguagem de marcação especial para os templates. Ela é baseada em Python, por isso é simples. O conteúdo do arquivo index.html é:

```
<htm>
<body>
<h1>Coleções</h2>
{% for i in lista %}
   {{i.nome}}
   <h3>Livros</h3>
   {% for j in i.livro_set.all %}
       {{j.nome}}<br>
   {% endfor %}
{% endfor %}
</body>
</htm>
```

É um arquivo HTML com marcações especiais. Quando é necessário executar algum comando é preciso colocá-lo entre os caracteres **{%** e **%}**. Não é preciso usar o **:** para indicar início de blocos, mas é preciso finalizá-los, como o exemplo do **for..endfor**. Para imprimir variáveis é só colocá-las entre os caracteres **{{** e **}}**, como no exemplo acima.

O resultado do acesso a URL http://localhost:8000/index é a listagem das coleções e seus respectivos livros.

Links

- http://allisson.wordpress.com/
- http://www.djangoproject.com/documentation/
- http://www.djangoproject.com/documentation/tutorial1/
- http://www.plugmasters.com.br/sys/materias/252/1/Hello-Django!
- http://www.plugmasters.com.br/sys/materias/275/1/Django,-Primeiros-passos,-parte-1
- http://www.plugmasters.com.br/sys/materias/498/1/Django%2C-Primeiros-Passos-2