Django : monter une application web en Python

INTRODUCTION

- Présentations
  - Davin Baragiotta
  - Montréal-Python (MP, sprints/hackathons, ateliers, PyCon)
- Objectifs
  - connaître l'architecture d'un projet : principaux fichiers
  - gérer data dans **backend**
  - présenter data dans **frontend**
- Documentation
  - doc : bonne version de django
    - officielle : https://docs.djangoproject.com/en/1.4/
    - français : http://docs.django-fr.org/
  - tutoriel
    - Part 1 : créer un projet, créer une app, jouer avec ORM (API)
      couvert ici
    - Part 2 : backend (admin)
      couvert ici
    - Part 3 : frontend
      couvert ici, plus straightforward même ;)
    - Part 4 : forms et vues génériques
      pas couvert ici
- Environnement technique

```
>>> import django
>>> django.get_version()
'1.4'
>>> import south
>>> south.__version__
'0.7.4.'
```

  - Python >= 2.5
  - Django 1.4
    - Mac OS X : voir tutoriel Part 1 pour manip supplémentaire
  - South 0.7.4 (optionnel)

PARTIE 1 : DÉVELOPPEMENT WEB AVEC DJANGO : ARCHITECTURE D'UN PROJET

Développement web

- schéma : le développement web
  - rôle d'un framework web : aider à construire réponse à une requête
- environnements
  - développement (DEV)
  - déploiement (TEST, PROD)

Django

Pourquoi Django vs autre framework (ex.: CherryPy, Pyramid, web2py)

- tout-en-un
- admin : meilleur vendeur ;)
- docs
- communauté : utilisation répandue

Principaux fichiers

- `urls.py`
- `views.py`
- `models.py`
- `templates` (HTML)
- `admin.py`

Projet : définition du besoin

Montréal-Python a une ligue de rugby *(ce n'est pas vrai : DoesNotExist)*.
On vous contacte pour créer un site web qui pourrait gérer les joueurs et les équipes...
... et planifier les matchs de la saison.

Projet : modélisation

- Joueur
- Equipe
- Saison
- Match

PARTIE 2 : HANDS-ON : PROJET ET APPLICATIONS

Création du projet

```
django-admin.py startproject ligue_rugby_mp
cd ligue_rugby_mp
```

> *Pour certains système d'exploitation, il pourrait être nécessaire de fournir le chemin complet vers le script `django-admin.py`. Ex.: Windows*

- survol des fichiers générés

```
python manage.py runserver
```

- `settings.py`

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3', # Add 'postgresql_psycopg2', 'mysql', 'sqlite3' or 'oracle'.
        'NAME': 'ligue_rugby_mp.db',                      # Or path to database file if using sqlite3.
        'USER': '',                      # Not used with sqlite3.
        'PASSWORD': '',                  # Not used with sqlite3.
        'HOST': '',                      # Set to empty string for localhost. Not used with sqlite3.
        'PORT': '',                      # Set to empty string for default. Not used with sqlite3.
    }
}
```

Création d'une application

```
python manage.py startapp ligue
```

- survol fichiers
- `ligue/models.py`
  - documentation :
    https://docs.djangoproject.com/en/1.4/topics/db/models/
    https://docs.djangoproject.com/en/1.4/ref/models/fields/
  - yeah! on code!
    - Equipe.nom
    - Joueur.nom

```
from django.db import models

class Equipe(models.Model):
    pass  # code fields here
```

```
class Joueur(models.Model):
    pass  # code fields here
```

- settings.py

```
INSTALLED_APPS = (
    # ...
    'ligue',
)
```

South

- settings.py

```
INSTALLED_APPS = (
    # ...
    'south',
)
```

- créer une migration pour une nouvelle application

```
python manage.py schemamigration ligue --initial
```

  - *créer une migration pour un application existante (pas le cas ici)*

```
python manage.py schemamigration --auto
```

- créer les tables de south

```
python manage.py syncdb
```

- appliquer les migrations

```
python manage.py migrate ligue
```

Backend : gérer les données dans l'admin

- urls.py
  activer admin : décommenter

```
# ...

from django.contrib import admin
admin.autodiscover()

urlpatterns = patterns('',
    # ...
    url(r'^admin/', include(admin.site.urls)),
)
```

- http://127.0.0.1:8000/
- settings.py

```
INSTALLED_APPS = (
    # ...
    'django.contrib.admin',
    # ...
)
```

- créer les tables de l'admin

```
python manage.py syncdb
```

- http://127.0.0.1:8000/admin/
- ligue/admin.py
  enregistrer modèles
  voir tutoriel Part 2
  https://docs.djangoproject.com/en/1.4/intro/tutorial02/

```
from django.contrib import admin

from ligue.models import *
```

```
admin.site.register(Equipe)
admin.site.register(Joueur)
```

- ligue/models.py

```
def __unicode__(self):
    return ""  # coder ici la chaîne unicode désirée que l'on veut retourner
```

  - Joueur.prenom
  - Joueur.date_naissance

> *Si south est installé, créer et appliquer la migration :*
>
> ```
> python manage.py schemamigration ligue --auto
> python manage.py migrate ligue
> ```
>
> *Sinon :*
>
> - *supprimer ligue_rugby_mp.db*
>
> ```
> python manage.py syncdb
> ```

- ligue/models.py
  - ajouter une relation entre Joueur et Equipe :
    - Joueur.equipe
  - modifier la base de données en conséquence :
    - south : créet et appliquer la migration
    - else : supprimer la DB et syncdb
- ajouter quelque Equipes et Joueurs dans l'admin

Frontend : présenter les données

- urls.py
  https://docs.djangoproject.com/en/1.4/topics/http/urls/

```
urlpatterns = patterns('',
    url(r'^$', 'ligue_rugby_mp.views.home', name='home'),
    # ...
)
```

- views.py
  https://docs.djangoproject.com/en/1.4/topics/http/views/
  voir tutoriel Part 3 (version shortcut)
  https://docs.djangoproject.com/en/1.4/intro/tutorial03/

```
from django.shortcuts import render

def home(request):
    c = { }
    return render(request, 'home.html', c)
```

- templates
  https://docs.djangoproject.com/en/1.4/topics/templates/
  - créer répertoire templates dans ligue_rugby_mp
  - settings.py

```
import os

# ...

TEMPLATE_DIRS = (
    os.path.join(os.path.dirname(__file__), "templates"),
)
```

  - templates/home.html

- views.py
  passer une variable au template
- templates/home.html
  utiliser une variable :

```
{{ var }}
```

- views.py

endroit où on code la logique en Python
plus simple si explore interactivement...

```
python manage.py shell
```

- ORM (API) : object relation mapping

```
from ligue.models import *
equipes = Equipe.objects.all()
for e in equipes: print e
e = equipes[0]
e.joueur_set.all()
j = Joueur.objects.get(id=1)
j.id
j = Joueur.objects.get(id=314)
e.id
joueurs = Joueur.objects.filter(equipe__nom__startswith='Pyth')
```

- `ligue/models.py`

```
related_name = "joueurs"
```

- relancer shell

```
e.joueurs.all()
e.joueurs.count()
```

- `views.py`
  passer les variables pertinentes pour accueil
  suite à exploration interactive via l'ORM dans le shell

- `templates/home.html`
  boucle for dans template :

```
{% for e in equipes %}
{% endfor %}
```

PARTIE 3 : PROJET LRMP PIMPÉ

Héritage de templates

- `base.html`

```
{% block main %}
{% endblock %}
```

- `templates`

```
{% extends "base.html" %}

{% block main %}
{% endblock %}
```

URL avec paramètres

- pages de détail : capter l'id de l'objet
  - `urls.py`
    import des urls de l'app ligue

```
urlpatterns = patterns('',
    # ...
    url(r'^', include('ligue.urls')),
)
```

- `ligue/urls.py`

```
urlpatterns = patterns('ligue.views',
    url(r'^equipes/(?P<id>\d+)$', 'equipe_detail', name="equipe"),
)
```

- `ligue/views.py`

```
from django.shortcuts import render

from ligue.models import Equipe
```

```
def equipe_detail(request, id):
    equipe = Equipe.objects.get(id=id)
    c = {
        'equipe': equipe,
    }
    return render(request, "ligue/equipe_detail.html", c)
```

- templates/ligue/equipe_detail.html

```
{{ equipe.name }}
```

Admin pimpé : ModelAdmin

https://docs.djangoproject.com/en/1.4/ref/contrib/admin/

- ligue/admin.py
- classes JoueurAdmin, EquipeAdmin
  héritent de ModelAdmin

```
class EquipeAdmin(admin.ModelAdmin):
    pass

class JoueurAdmin(admin.ModelAdmin):
    pass
```

- enregistrer Modele avec ModeleAdmin

```
admin.site.register(Equipe, EquipeAdmin)
admin.site.register(Joueur, JoueurAdmin)
```

- config ModeleAdmin

```
list_display
search_fields
list_filter
```

- plus?
  https://docs.djangoproject.com/en/1.4/ref/contrib/admin/
  - fields
  - fieldsets
  - ...

Saut dans le temps : télécharger les sources

- Télécharger les sources du projet pimpé
- Extraire
- Pimpé? Quoi de neuf?

Fichiers statiques : CSS, images et js

https://docs.djangoproject.com/en/1.4/howto/static-files/

- répertoire : static
  - css
  - images
  - js
- settings.py

```
PROJECT_ROOT = os.path.dirname(__file__)
SITE_ROOT = os.path.dirname(PROJECT_ROOT)

MEDIA_ROOT = os.path.join(PROJECT_ROOT, 'media')
MEDIA_URL = '/media/'

STATIC_ROOT = os.path.join(SITE_ROOT, 'site_static')
STATIC_URL = '/static/'
STATICFILES_DIRS = (
    os.path.join(PROJECT_ROOT, 'static'),
)
```

- urls.py

```
from django.contrib.staticfiles.urls import staticfiles_urlpatterns

# ...

urlpatterns += staticfiles_urlpatterns()
```

- templates

```
{{ STATIC_URL }}
```

Templates pimpés

- templates/base.html

Connexion du user

- templates
  - connexion.html
  - deconnexion.html
- urls.py

```
urlpatterns = patterns('',
    # ...
    url(r'^login/$', 'django.contrib.auth.views.login',
        dict(template_name='connexion.html',),
        'connexion'
    ),
    url(r'^logout/$', 'django.contrib.auth.views.logout',
        dict(template_name='deconnexion.html',),
        'deconnexion'
    ),
    # ...
)
```

- settings.py

```
LOGIN_URL = "/login/"
LOGIN_REDIRECT_URL = "/"
```

PARTIE 4 : HANDS-ON : CRÉER L'APPLICATION CALENDRIER POUR LE PROJET LRMP

Créer une application calendrier

Ajouter les modèles Match et Saison

- Match
  - date
  - lieu
  - equipe1
  - equipe2
  - score1
  - score2
- Saison
  - ...

Autre exercice : âge d'un joueur

- Joueur.date_naissance
- Joueur.age()
  ajouter une methode age() sur la classe Joueur utilisant la date de naissance

CONCLUSION : POUR CONTINUER

- Autres aspects non couverts (quelques uns)
  - permissions et décorateurs
  - manage.py inspectdb
    https://docs.djangoproject.com/en/1.4/howto/legacy-databases/
  - fixtures : données initiales et de test
    https://docs.djangoproject.com/en/1.4/howto/initial-data/
  - manage.py test
  - forms
  - generic views
  - templates : tags, filtres

- Mailing list des utilisateurs Django :
  http://groups.google.com/group/django-users
- Channel IRC #django :
  irc://irc.freenode.net/django
- Contribs, plugins
  https://docs.djangoproject.com/en/dev/ref/contrib/
  - south
  - reversion
  - Autre : pypi
    http://pypi.python.org/
- Enjoy!

SONDAGE DE SATISFACTION

- http://www.surveymonkey.com/s/LKRWS9V