Django - Intranet2

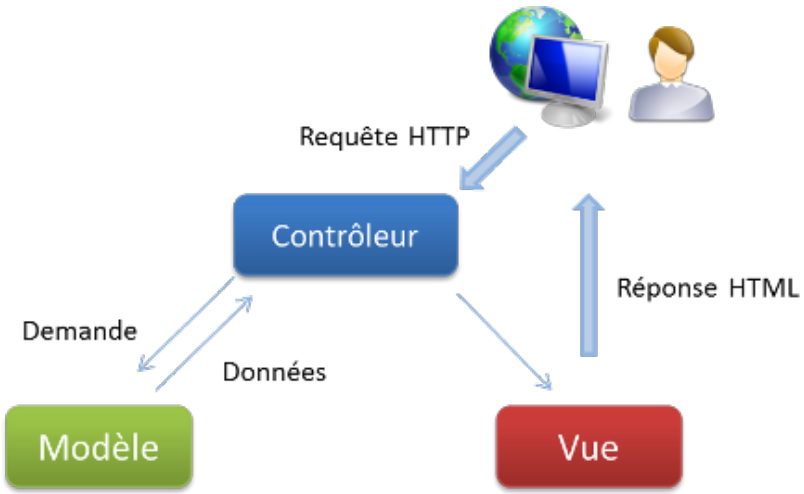
Ariane SORET

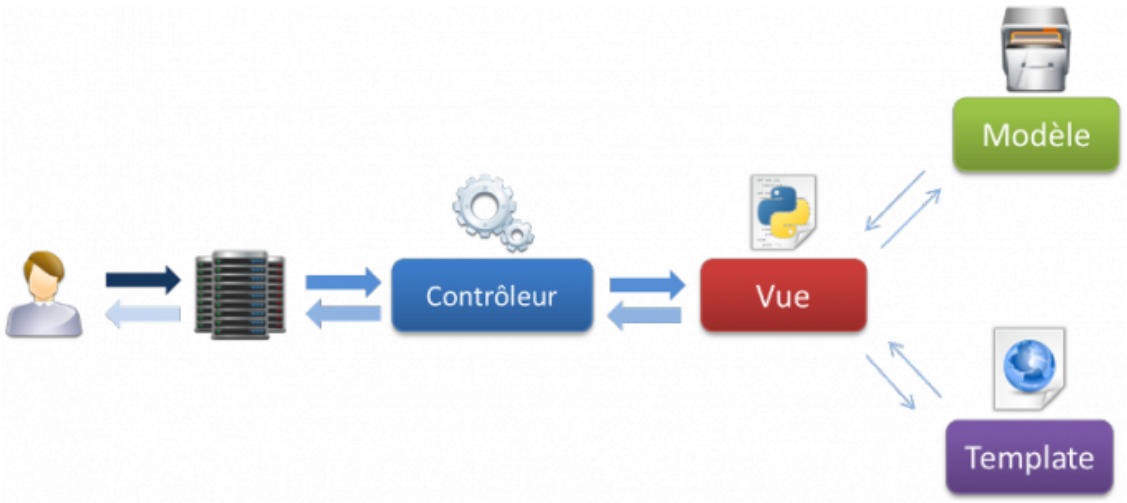
Encadrée par Vincent LE GALLIC et Daniel STAN

23 avril 2013

- L'univers de Django
- Comment ça marche ?
- Intranet2

- Framework (cadre de travail) python, pour créer (rapidement) de (belles) applications
- Architecture MVT (Modèle Vue Template), pas MVC (Modèle Vue Contrôleur)
- Modèle : information ; interface entre code et base de donnée (sqlite)
  Vue : visualisation de l'information ; présente données et recueille actions de l'utilisateur (page web = vue)
  Contrôleur : prend en charge les requêtes de l'utilisateur
  Template : (fichier HTML) affichage des données


Requête HTTP
Contrôleur
Réponse HTML
Demande
Données
Modèle
Vue


Modèle
Contrôleur
Vue
Template

- Django permet de créer des projets (blog, intranet)
- Un projet comporte plusieurs applications
- Une application peut servir à plusieurs projets

- Créer un projet Django :

  `django-admin.py startproject blogtest`
- Ce qui crée dans le dossier blogtest : manage.py blogtest
- Dans le projet blogtest :

  `__init__.py  settings.py  urls.py  wsgi.py`

```
python manage.py startapp blog
```

Dans blog :

```
admin.py  __init__.py  models.py  tests.py  views.py
```

Il faut ajouter blog à la liste des applications du projet dans

```
settings.py
```

Les modèles

- Ils sont dans

  `models.py`

- Liaison entre modèles :

  `ForeignKey`

- Pour dire à Django de créer les bases de données correspondantes :

  `python manage.py syncdb`

Les Vues

- Sont dans

  `views.py`
- Fait le lien entre base de données et le template.

Routage url

- Dans le urls.py du projet
- Mieux vaut stocker les urls dans un fichier pour chaque application (créer urls.py dans blog).
  Dans urls.py du projet, ajouter :

```
url(r'^blog/', include('blog.urls'))
```

Testons !

```
python manage.py runserver
```

- Pour appeler un template et générer une réponse html : fonction

  `render()`
- Il faut lier template et vue :

  `/plop/templates/article_form.html`
- Et créer la vue
- Requête POST et GET

Les formulaires :

- ont une déclaration très similaire à celle d'un modèle
- héritent, comme les modèles, d'une classe mère Form
- n'ont pas de fichiers qui leur est dédié (comme models.py) ; en général, on crée dans chaque application un

  `forms.py`

  pour accueillir les déclrations de formulaire

On distingue deux types de requêtes pour les formulaires

- requête GET : aucune donnée n'est fournie par l'utilisateur, envoi d'un formulaire vierge ; par ex : demande du formulaire d'impression
- requête POST : envoi de données (formulaire déjà rempli) ; par ex : demander à lancer l'impression après avoir renseigné les champs du formulaire (les données sont transmises à l'imprimante)

Les données sont stockées dans un dictionnaire.

```
python manage.py shell
from blog.models import Article, Categorie
art=Article() # création d'un objet de type Article
art.titre = "Tout sur les korats"
cat=Categorie()
cat.nom="Les chats"
cat.save()
art.categorie = cat  # attribution d'une catégorie (ForeignK
art.save() # on enregistre les modifications
Article.objects.all()
[<Article: Tout sur les korats>]
```

- Sur o2
- L'authentification utilise la base ldap

- Interface d'impression
- Digicode
- Machines adhérents
- ...

Questions ?