Django: web-based software development made (more) simple. Example of a clinical trial management system.

Who Am I

Ghislain Bidaut

(ghislain.bidaut@inserm.fr)

IR Cibi Platform (CRCM Integrative Bioinformatics)

Web: http://cibi.marseille.inserm.fr

Forge: http://forgecrcm.marseille.inserm.fr/projects/cibi

Biologist/informatician dialog...

- (Biologist) Could you make me a software or a database to store my clinical trials data ?
- (Informatician) Cound you explain me what do you want exactly ?
- (B) Not sure yet, I just want to store some *important* data that I have on an Excel spreadsheed and share it with other people.
- (I) And everyone could modify it ?
- (B) No, of course, only certain people. And of course, this data is **absolutely vital for my organization** and must be kept secure.
- (I) So you need a Web-based system along a database.
- (B) Maybe... And yes, it would be nice if we could access it from my iPad and from my collegue's phone!

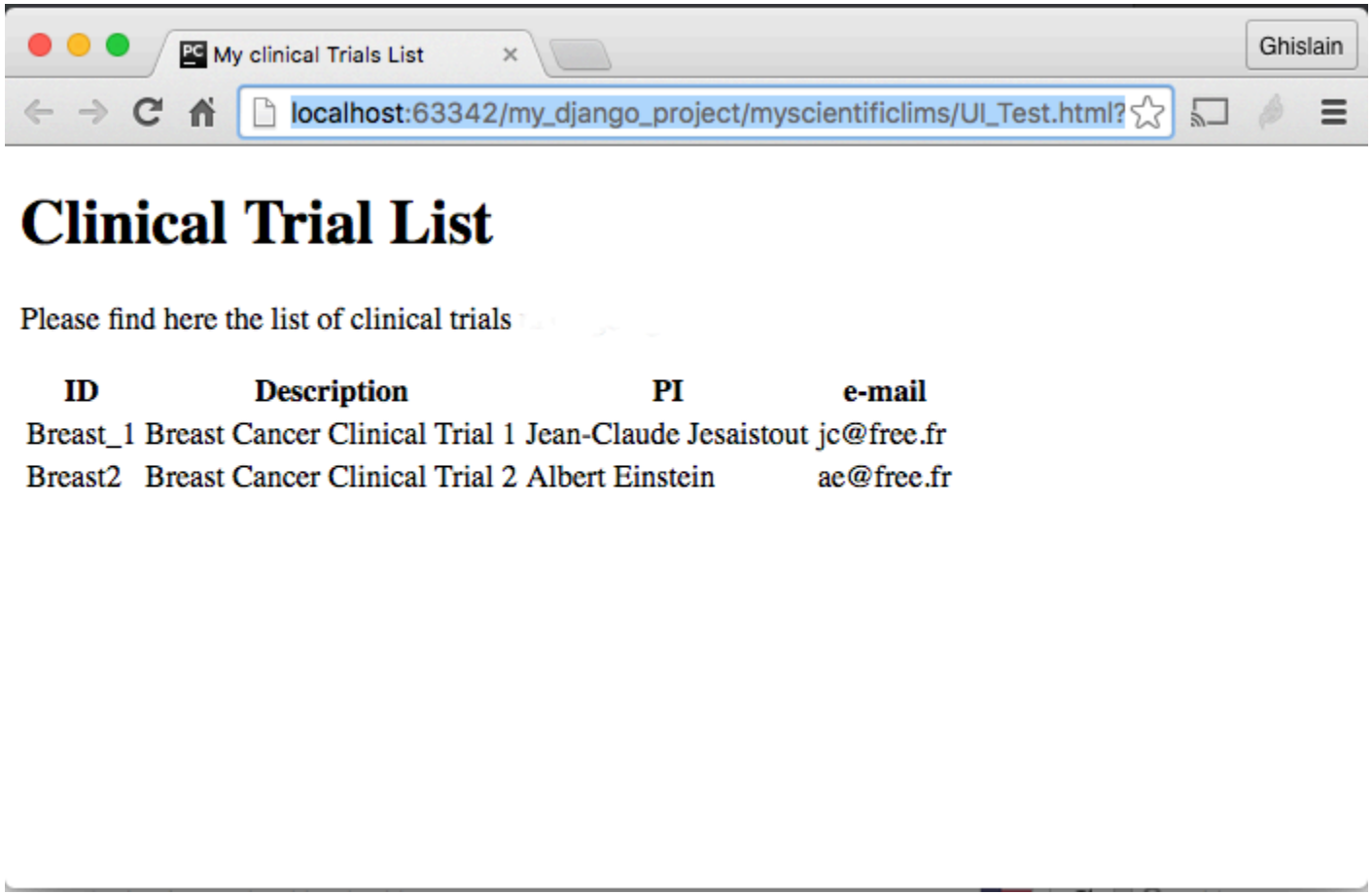
Quick Specs

- Data to be stored: Clinical trial related data: **ID**, **description**, **Principal Investigator(s)**, **Disease Category(ies)**.
- Clinical department staff can **modify** it, others can only access it **read-only** (autorizations managment).
- App must be accessible from **all systems type**, and **responsive.**
- App must be easily **prototypable** and **maintainable.** Specs may change overnight!
- And... **I dont' have time** to learn PHP/MySQL/CSS, but I know *Python* !

OK, lets' try with HTML !

```
 1 <!DOCTYPE html>
 2 <html lang="en">
 3 <head>
 4     <meta charset="UTF-8">
 5     <title>My clinical Trials List</title>
 6 </head>
 7 <body>
 8     <h1>Clinical Trial List</h1>
 9     <p>Please find here the list of clinical trials</p>
10 <table>
11     <tr><th>ID</th><th>Description</th><th>PI</th><th>e-mail</th></tr>
12     <tr><td>Breast_1</td><td>Breast Cancer Clinical Trial 1</td>
13         <td>Jean-Claude Jesaistout</td><td>jc@free.fr</td></tr>
14     <tr><td>Breast2</td><td>Breast Cancer Clinical Trial 2</td>
15         <td>Albert Einstein</td><td>ae@free.fr</td></tr>
16 </table>
17 </body>
18 </html>
```

OK, lets' try with HTML !



General principle 1

OK this data is accessible by all but, it would be better if:

```
- user interface
- data
- program that links interface and data
```

Were *separated*.

```
- Also better if display was prettier.
```

- We need to apply the **Model, View, Template** (MVT) Model for development.

General principle 2

- **Never** recode twice the same thing: This is the **Don't Repeat Yoursef** (DRY) principle.
- Better: *Simple, clean, redable, maintainable*.
- Worse: *complex, dirty, unreadable, unmaintainable*.
- **Django** adheres **naturally** to the *best* principles.

OK, but what is Django ?

- **Django Reinhardt** is a French guitarist [1910-1953].
- **Django** is a **free** and **open-source** web development framework.
- Follows the **MVT** (Model, View, Template) architectural pattern (**NOT** Model, View, Controller MVC).
- Initially developed in 2003 for **news publication** (Lawrence Journal-World).
- Developed and maintained since 2008 by the **Django Software Foundation** (DSF).
- Pinterest, Instagram, Mozilla, The Washington Time, etc...
- **Multi environment** (for the user and the developer). Operating system independant since based on *Python*.
- A lot of web hosting services **naturally** allow for deployment of Django-based websites.

OK, let's start with Django

```
 1 pip install Django
 2 django-admin startproject myscientificlims .
 3 ls -l
 4 -rw-r--r--  1 bidaut  staff  36864  8 jui 09:33 db.sqlite3
 5 -rwxr-xr-x  1 bidaut  staff    259  8 jui 09:29 manage.py
 6 drwxr-xr-x  9 bidaut  staff    306 13 jui 14:23 myscientificlims/
 7 ls -l myscientificlims
 8 -rw-r--r--  1 bidaut  staff     0  8 jui 09:29 __init__.py
 9 -rw-r--r--  1 bidaut  staff  2668  8 jui 09:29 settings.py
10 -rw-r--r--  1 bidaut  staff   766  8 jui 09:29 urls.py
11 -rw-r--r--  1 bidaut  staff   409  8 jui 09:29 wsgi.py
```

Manage.py allows for managing the project, starting the server, deploying the database, etc...

setting.py

In settings.py, change *Timezone* and *Database* parameters, then issue the following commands.

```
1 python manage.py migrate
2 python manage.py runserver
```

Then, see the development server in action at: http://127.0.0.1:8000/

Great, now let's add some content!

We have created the **global project**. Functionalities (actions) are added through **applications**. Each functionality must correspond to a single application and action.

Create App

```
 1 python manage.py startapp clinicalmanager
 2 ls -l clinicalmanager/
 3
 4 total 32
 5 -rw-r--r--  1 bidaut  staff    0 13 jui 15:19 __init__.py
 6 -rw-r--r--  1 bidaut  staff   63 13 jui 15:19 admin.py
 7 drwxr-xr-x  3 bidaut  staff  102 13 jui 15:19 migrations/
 8 -rw-r--r--  1 bidaut  staff   57 13 jui 15:19 models.py
 9 -rw-r--r--  1 bidaut  staff   60 13 jui 15:19 tests.py
10 -rw-r--r--  1 bidaut  staff   63 13 jui 15:19 views.py
```

And add 'clinicalmanager' in 'INSTALLED_APPS'.' (settings.py).

OK, now we have a complete skeleton for our app.

Objects

Clinical Trial

- Title (e.g. "My Breast Cancer Clinical trial")
- Internal Identifier (e.g. "Breast01")
- Description
- Opening date
- Category ("Breast Cancer")
- Principal Investigator (PI)

PI

- Salutation (Dr, Pr, Mrs, Mr, ...)
- First Name
- Last Name
- email (optional)

models.py

```
1 class ClinicalTrial(models.Model):
2
3     title = models.CharField(max_length=100, unique=True)
4     internal_identifier = models.CharField(max_length=100, unique=True,
5         validators=[validate_alphanumerical])
```

models.py

```
 1 class ClinicalTrial(models.Model):
 2
 3     title = models.CharField(max_length=100, unique=True)
 4     internal_identifier = models.CharField(max_length=100, unique=True,
 5         validators=[validate_alphanumerical])
 6
 7     isConfidential = models.BooleanField(default=True)
 8
 9     description = models.TextField(max_length=10000)
10     opening_date = models.DateField()
```

models.py

```
 1 class ClinicalTrial(models.Model):
 2
 3     title = models.CharField(max_length=100, unique=True)
 4     internal_identifier = models.CharField(max_length=100, unique=True,
 5         validators=[validate_alphanumerical])
 6
 7     isConfidential = models.BooleanField(default=True)
 8
 9     description = models.TextField(max_length=10000)
10     opening_date = models.DateField()
11
12     category = models.ManyToManyField(Category)
13     pi = models.ManyToManyField(PI)
```

models.py

```
 1 class ClinicalTrial(models.Model):
 2
 3     title = models.CharField(max_length=100, unique=True)
 4     internal_identifier = models.CharField(max_length=100, unique=True,
 5         validators=[validate_alphanumerical])
 6
 7     isConfidential = models.BooleanField(default=True)
 8
 9     description = models.TextField(max_length=10000)
10     opening_date = models.DateField()
11
12     category = models.ManyToManyField(Category)
13     pi = models.ManyToManyField(PI)
14
15 class PI(models.Model):
16     salutation = models.CharField(max_length=10)
17     first_name = models.CharField(max_length=400)
18     last_name = models.CharField(max_length=400)
19     email = models.EmailField(blank=True) # Optional
20
21 class Category(models.Model):
22     category = models.CharField(max_length=1000)
```

Object string representation

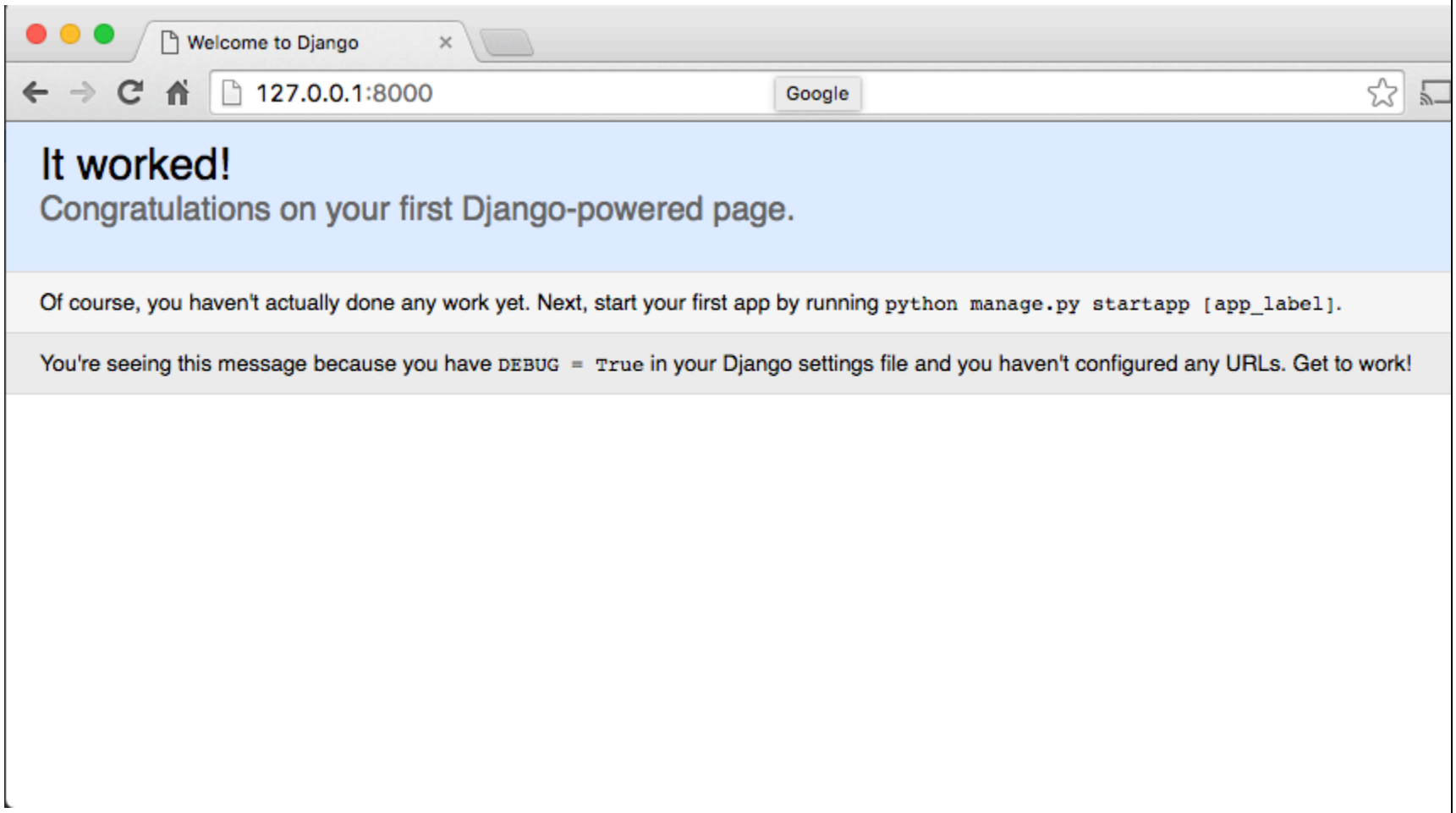
```
 1 class PI(models.Model):
 2
 3     salutation = models.CharField(max_length=10)
 4     first_name = models.CharField(max_length=400)
 5     last_name = models.CharField(max_length=400)
 6     email = models.EmailField(blank=True)  # Optional
 7
 8     # proper string representation
 9     def __unicode__(self):
10     return '%s %s' % (self.first_name, self.last_name)
```

Migrate model in db

```
 1 python manage.py makemigrations clinicalmanager
 2 0001_initial.py:
 3 - Create model Category
 4 - Create model PI
 5 - Create model ClinicalTrial
 6 python manage.py migrate clinicalmanager
 7 Operations to perform:
 8 Apply all migrations: clinicalmanager
 9 Running migrations:
10 Rendering model states... DONE
11 Applying clinicalmanager.0001_initial... OK
12 Applying clinicalmanager.0002_auto_20160613_1342... OK
13 Applying clinicalmanager.0003_auto_20160613_1342... OK
14 python manage.py runserver
```

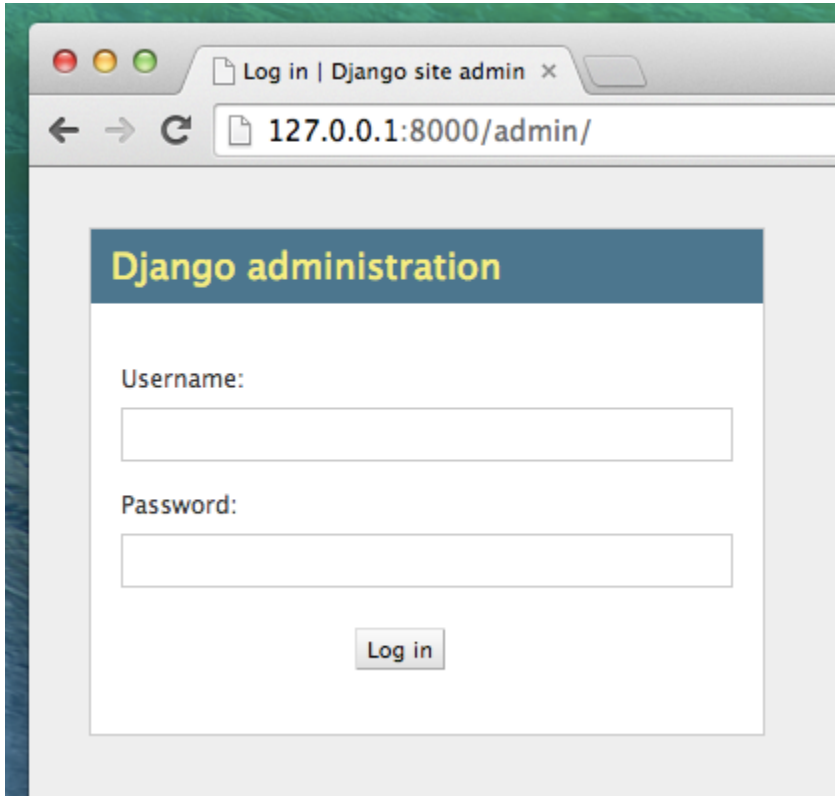
Great, I've got my web server!

http://127.0.0.1:8000/



Great, I've got my admin interface!

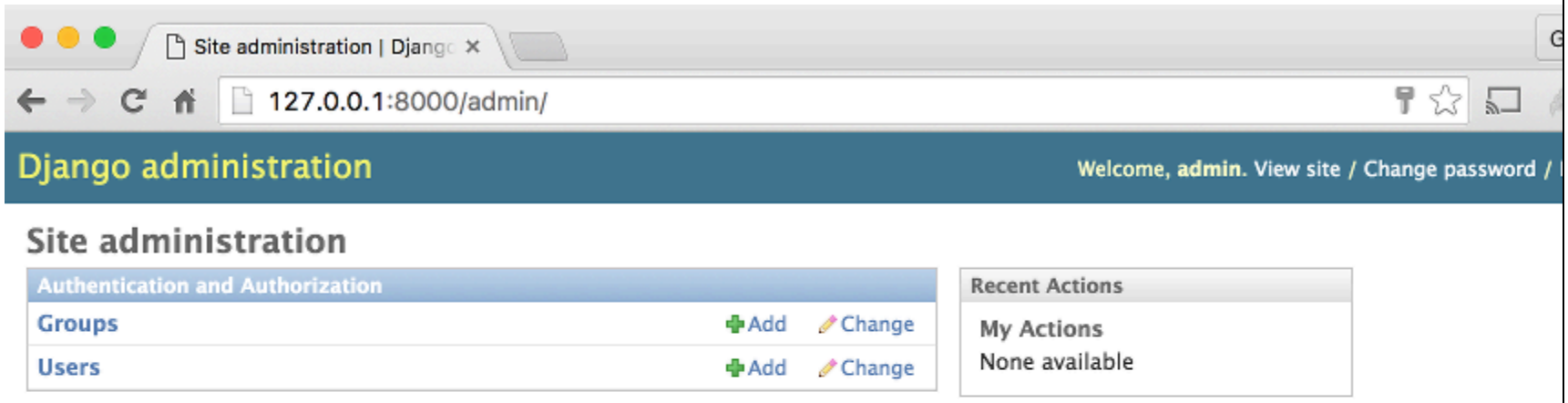
http://127.0.0.1/admin



Create Super User

```
1 $ python manage.py createsuperuser
2 Username: admin
3 Email address: admin@admin.com
4 Password:
5 Password (again):
6 Superuser created successfully.
```

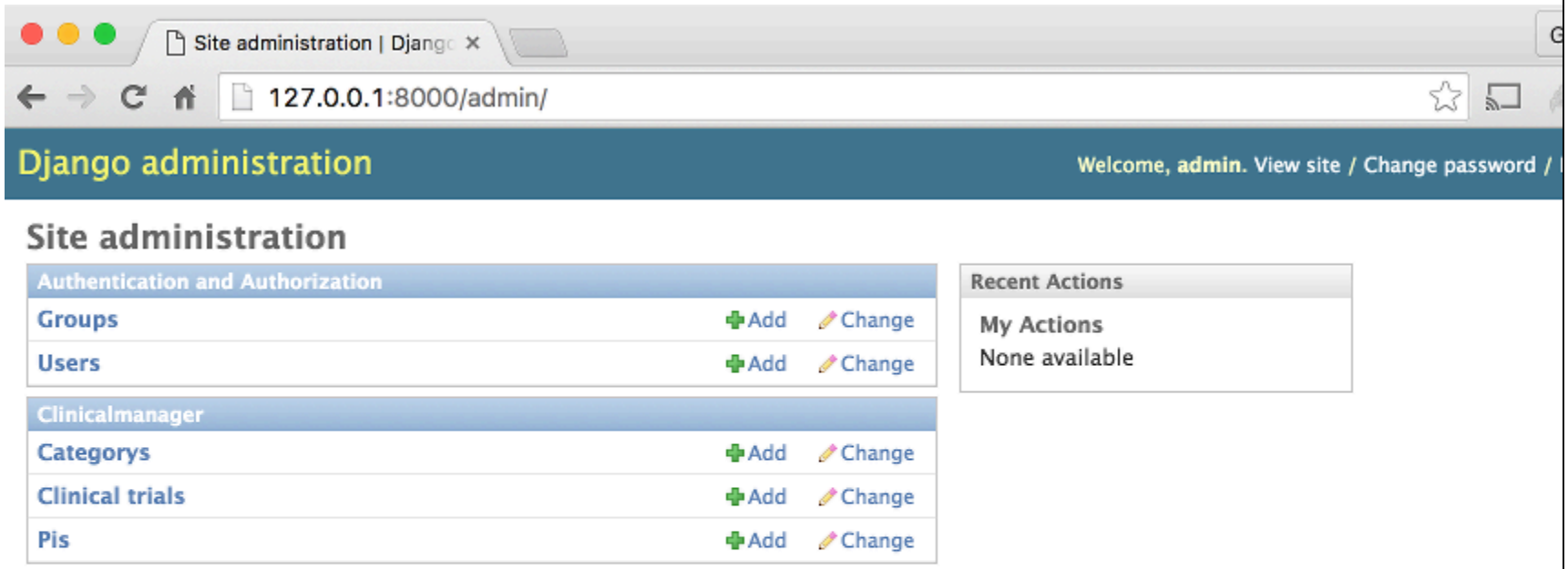
Great, I've got my admin interface!



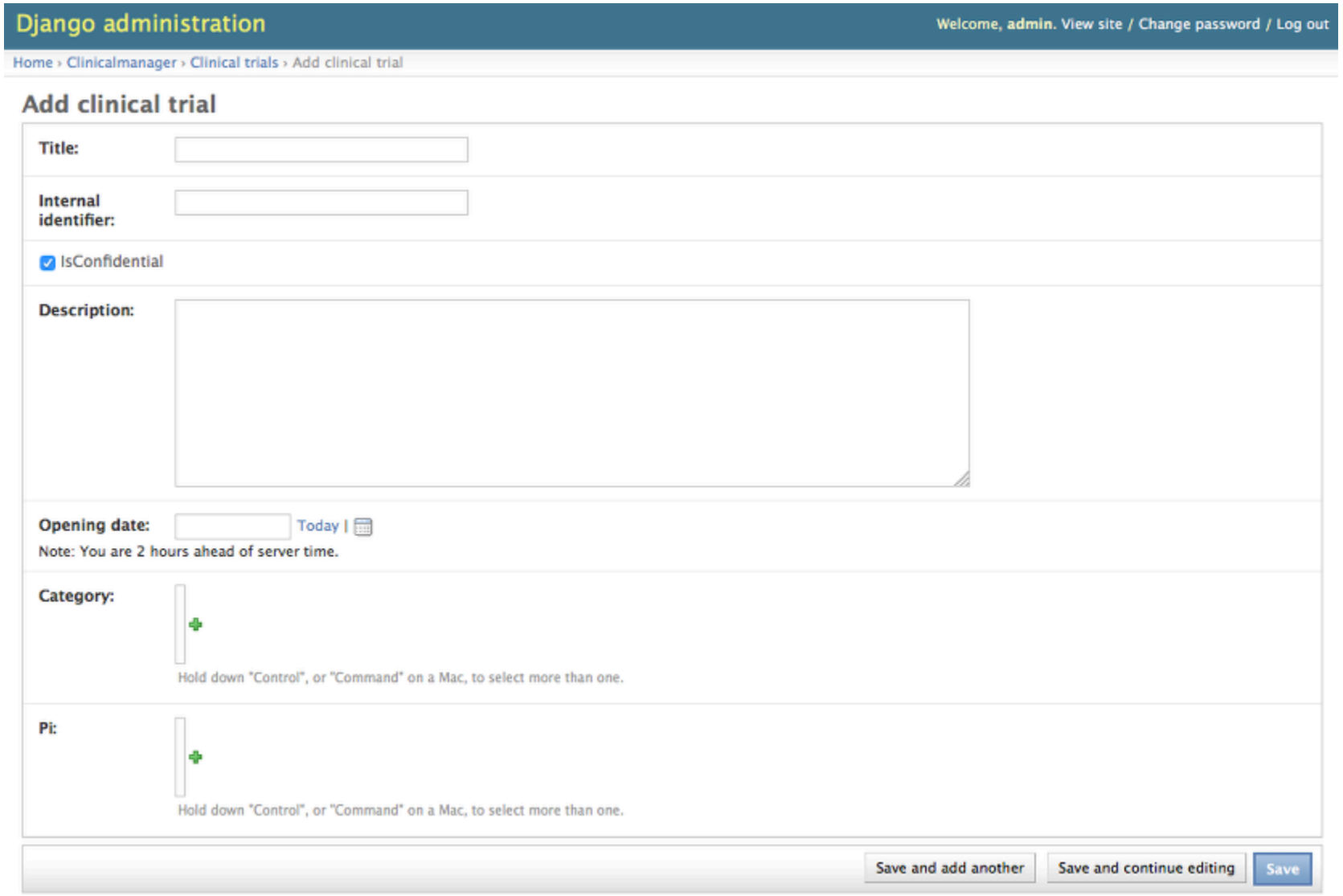
Add model in admin interface
Format Data in admin interface (file admin.py)

```
 1 class ClinicalTrialAdmin(admin.ModelAdmin):
 2 list_display = ('title', 'internal_identifier', 'isConfidential', 'opening_date', 'status
 3 list_filter = ('title', 'internal_identifier', 'isConfidential', 'opening_date', 'status'
 4
 5 class PIAdmin(admin.ModelAdmin):
 6 list_display = ('last_name', 'first_name', 'salutation', 'email')
 7 list_filter = ('first_name', 'last_name')
 8
 9 class CategoryAdmin(admin.ModelAdmin):
10 list_display = ('category', )
11 list_filter = ( 'category', )
12
13 # Register your models here.
14 admin.site.register(ClinicalTrial, ClinicalTrialAdmin)
15 admin.site.register(PI, PIAdmin)
16 admin.site.register(Category, CategoryAdmin)
```

Great, I've got my admin interface!



Great, I've got my admin interface!



URLs (clinicalmanager/urls.py)

What's an URL ? Uniform Resource Locator (e.g http://www.debian.org).

Django manages URLs in a smart fashion in order to retrieve information using the url.py file.

Examples:

List all clinical trials:

http://128.0.0.1:8000/list

Display clinical trial 'Breast_1':

http://128.0.0.1:8000/trial/Breast_1

Default url.py

```
 1 from django.conf.urls import include, url
 2 from django.contrib import admin
 3
 4 urlpatterns = [
 5 # Examples:
 6 # url(r'^$', 'mysite.views.home', name='home'),
 7 # url(r'^blog/', include('blog.urls')),
 8
 9 url(r'^admin/', include(admin.site.urls)),
10 ]
```

Regular Expression (RegEx)

- '^': Beginning of text
- '$': end of text
- '\d': integer
- '\w': any alphanumerical character

Examples:

http://128.0.0.1:8000/list

http://128.0.0.1:8000/trial/Breast_1

```
1 url(r'^list/$', clinicaltrials_list, name='list'),
2 url(r'^trial/(\w+)/$', clinicaltrial_view, name='clinicaltrial'),
```

Views (clinicalmanager/views.py)

Let's do the http://128.0.0.1:8000/trial/Breast_1 case.

The code in views.py connects data from the database to the viewable content.

```
1 def clinicaltrial_view(request, clinicaltrial_id):
2
3 # Get clinical trial with clinical ID
4 clinicaltrial = ClinicalTrial.objects.get(
5     internal_identifier=clinicaltrial_id)
6
7 return render_to_response("clinicaltrials/trial.html", {
8 'clinicaltrial': clinicaltrial,
9 }, RequestContext(request))
```

Templates, inheritance and variables

base.html

```
<DOCTYPE html>

  <head>
    <title>{% block title %}{% endblock %}</title>
  </head>

  <body>
        {% block content %}
       {% endblock %}
    <footer>
       (c) Clinicaltrial System - Copyright Ghislain Bidaut 2016 - Droits réservés.
    </footer>
    </body>
  </html>
```

Templates, inheritance and variables

base.html

```
<DOCTYPE html>

  <head>
    <title>{% block title %}{% en
  </head>

  <body>
        {% block content %}
       {% endblock %}
    <footer>
       (c) Clinicaltrial System - Co
    </footer>
    </body>
  </html>
```

trial.html

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

{% block title %}
        Essai [ {{ clinicaltrial.title }} ]
{% endblock %}

{% block content %}

  <h2>Essai clinique {{ clinicaltrial.title }} [ {{ clinicaltrial.internal_identifier }}]</h2>

  <h3>Date d'ouverture</h3>
  <p>{{ clinicaltrial.opening_date }}</p>

  <h3>Description</h3>
  <p>{{ clinicaltrial.description }}</p>

{% endblock %}
```

Templates, inheritance and variables

File clinicalmanager/template/base.html

```
 1 <!DOCTYPE html>
 2 <html lang="en">
 3 <head>
 4     <meta charset="utf-8">
 5     <title>{% block title %}{% endblock %}</title>
 6 </head>
 7
 8 <body>
 9     <h1 class="title">Clinical Trial Management System</h1>
10     <div class="container">
11         {% block content %}{% endblock %}
12     </div>
13     {% block footer %}
14     <footer>
15     <p class="footer">(c) Clinicaltrial System - Copyright Ghislain Bidaut 2016 - Droits
16     </footer>
17     {% endblock %}
18     </body>
19 </html>
```

Templates, inheritance and variables

File clinicalmanager/templates/trial.html

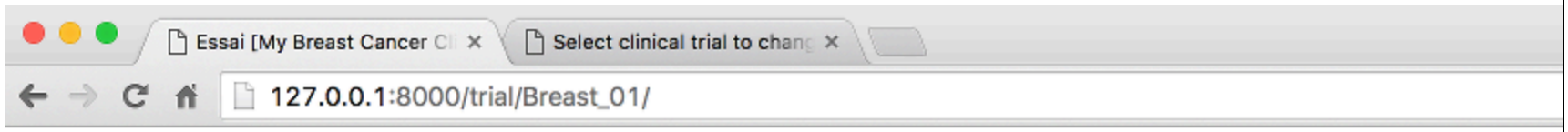
```
 1 {% extends "base.html" %}
 2 {% block title %}Essai [{{ clinicaltrial.title }}]{% endblock %}
 3
 4 {% block content %}
 5
 6 <h2>Essai clinique {{ clinicaltrial.title }} [{{ clinicaltrial.internal_identifier }}]</h
 7
 8 {% if clinicaltrial.isConfidential %}
 9 <p class="error">Essai Confidentiel</p>
10 {% else %}
11 <p class="success">Essai Non Confidentiel</p>
12 {% endif %}
13
14 <h3>Date d'ouverture</h3>
15 <p>{{ clinicaltrial.opening_date  }}</p>
16 ...
```

Templates, inheritance and variables

File clinicalmanager/templates/trial.html

```
 1 ...
 2 <h3>Pathologies(s)</h3>
 3 <p>
 4 <ul>
 5 {% for cate in clinicaltrial.category.all %}
 6 <li>{{ cate }}</li>
 7 {% endfor %}
 8
 9 </ul>
10 </p>
11
12 <h3>Description</h3>
13 <p>{{ clinicaltrial.description }}</p>
14
15
16 {% endblock %}
```

Result (no CSS)



Clinical Trial Management System

Essai clinique My Breast Cancer Clinical Trial [Breast_01]

Essai Non Confidentiel

Date d'ouverture

June 13, 2016

Pathologies(s)

- Category object

Investigateurs

- Jean-Claude Jesaistout

Titre

This is a breast cancer CT.

CSS boilerplates

To make things prettier.

Example: Skeleton.

- Download skeleton from http://getskeleton.com/.
- create a 'static' dir and move Skeleton code inside.

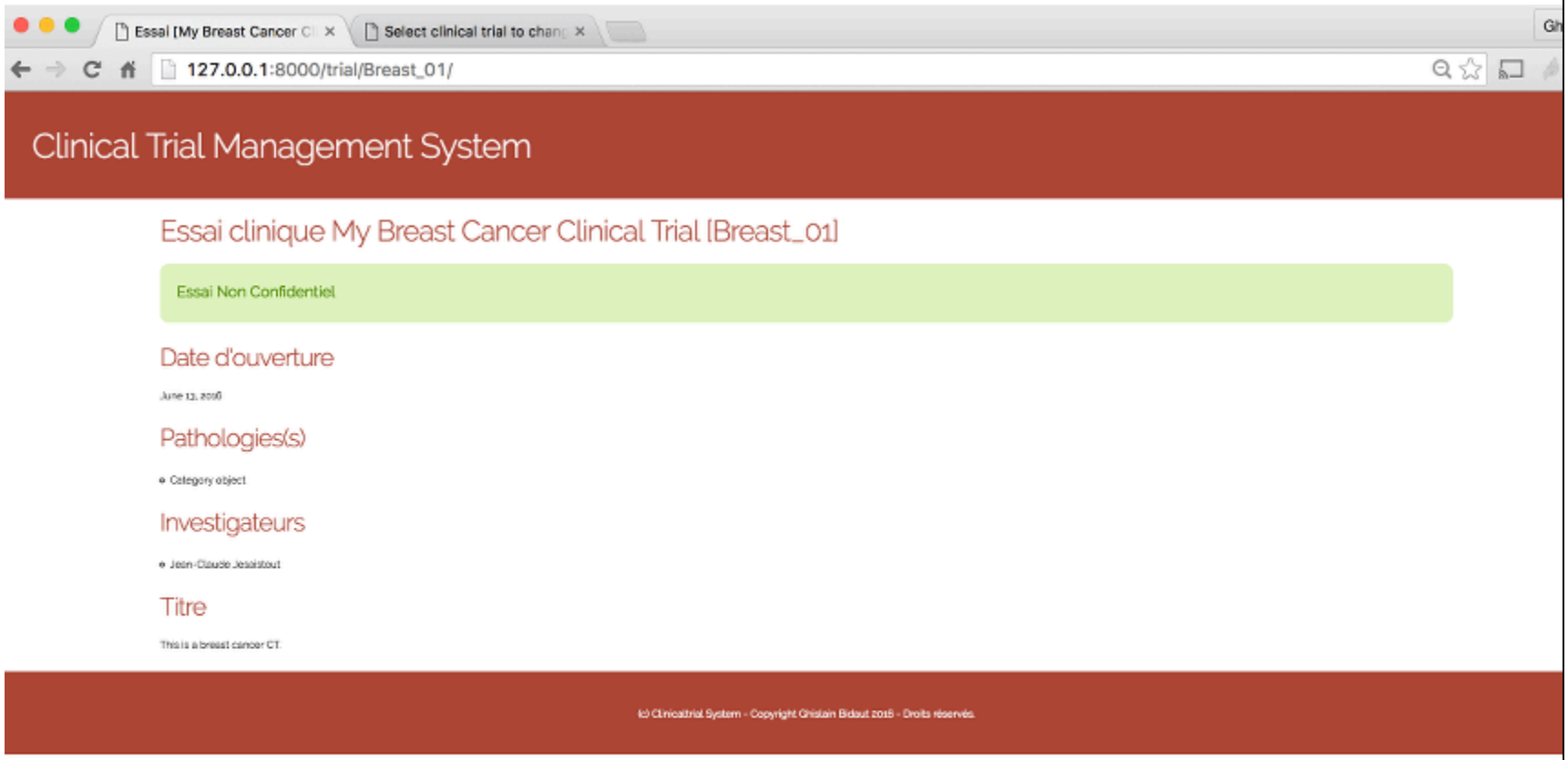
Add following in settings.py

```
1 STATIC_URL = '/static/'
2
3 STATICFILES_DIRS = (
4     os.path.join(BASE_DIR, "static"),
5     '/var/www/static/',
6 )
```

Data is then served with instruction as follows

```
1 <link rel="stylesheet" href="{% static "Skeleton-2.0.4/css/normalize.css"
2     %}"/>
```

Result (with CSS)



Conclusion

- Model, View and Controllers are separated.
- In the view itself, content (*HTML*) and page layout (*CSS*) are separated
- Project structured in applications
- Project settings described in *settings.py*
- Data described in *models.py*
- URLs described in *url.py*
- Generation of view described in *views.py*
- Templates written in HTML (*templates* dir)
- Applicable to lots of potential development in the lab (Lims, data visualisation, etc...)

Putting it all together

CLAM

Clinical Trial Management System

LISTE RECHERCHE STATS A PROPOS/CONTACT ADMINISTRATION (RÉSERVÉ) CONNECTION

Filtrer par pathologie(s)

Toutes

Filtrer par status

Toutes

Tri par

date

Ouverts après

Ouverts avant

Recherche

MISE À JOUR

VERSION IMPRIMABLE

Afficher 10 éléments

Essai Clinique	Statut	Pathologie	Date d'ouverture	Titre
Aucune donnée disponible dans le tableau				

Affichage de l'élement 0 à 0 sur 0 éléments

Going Further

- PyCharm IDE https://www.jetbrains.com/pycharm/
- SQLite browser http://sqlitebrowser.org/
- Data entry through web interface (Django Forms)
- REST server for client-server apps
- Security
- Non-regression testing and continuous integration (Jenkins)
- More Javascript (jQuery, jqPlot, JSON...)
- More advanced databases (e.g.MySQL) or No-SQL (e.g. MongoDB) or other
- Deployment (Git, OpenShift, PythonAnywhere)

Presentation created with Python Landslide

Quizz

- What is Django ?

Quizz

- What is Django ?
- What does MVT stands for ?

Quizz

- What is Django ?
- What does MVT stands for ?
- What is an URL ?

Quizz

- What is Django ?
- What does MVT stands for ?
- What is an URL ?
- In what file do we store the models ?

Quizz

- What is Django ?
- What does MVT stands for ?
- What is an URL ?
- In what file do we store the models ?
- In what file do we store the URLs ?

Quizz

- What is Django ?
- What does MVT stands for ?
- What is an URL ?
- In what file do we store the models ?
- In what file do we store the URLs ?
- In what file do we store the code in response to URLs ?

Quizz

- What is Django ?
- What does MVT stands for ?
- What is an URL ?
- In what file do we store the models ?
- In what file do we store the URLs ?
- In what file do we store the code in response to URLs ?
- What is a 'responsive' web site ?

Quizz

- What is Django ?
- What does MVT stands for ?
- What is an URL ?
- In what file do we store the models ?
- In what file do we store the URLs ?
- In what file do we store the code in response to URLs ?
- What is a 'responsive' web site ?
- Can I run a production server from my desktop ?

Thanks !