

No sabemos si podemos cambiar el mundo,
pero si estamos convencidos de que
podemos transformar vidas. Queremos
ayudar a la gente a encontrar eso que ama, y
conectar gente talentosa con otras personas
y/o oportunidades que conlleven a resultados
aún más GRANDES.

CURSO DJANGO

Curso de Django 1.8 y Python



Arturo Jamaica García

Linux : `sudo apt-get install python easy_install django`

Windows : <http://www.activestate.com/activepython/downloads>

Mac : `easy_install django` , `easy_install django`

PYTHON

Python es un lenguaje de programación creado por **Guido van Rossum** a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “**Monty Python**”.

Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script.

Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora

VENTAJAS

- **Legible**: Sintaxis intuitiva y estricta
- **Productivo** : Entre 1/3 y 1/5 más rápido que Java o C++
- **Portable** : GNU/Linux, Windows, Mac OS X, ...
- **Recargado** : Standard Library, Third parties

PRINT

```
print "Hola Mundo!"  
print "Hola de Nuevo"  
print "Hola Platzi"  
print "Hola Colombia."  
print 'Hola.'  
print "Saludos'."  
print 'Hola amigos.'
```

COMO LO USO?

holamundo.py

```
#!/usr/bin/env python  
print "hola mundo!"
```

Consola

```
$ python holamundo.py  
hola mundo!
```

```
$ python  
Python 2.6.5 (r265:79063, Apr 16 2010, 13:09:56)  
[GCC 4.4.3] on linux2  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print "hola mundo!"  
hola mundo!
```

TIPOS DE DATOS

- **int** : 15 float : 14.3 Long : 35L
- **bool** : True False
- **str** : "Hola Mundo"
- **list** : [1, [2, 'three'], 4]
- **dict** : { 'food' : 'spam', 'taste': 'yum' }
- **tuple** : (1 , 'spam' , 4 , 'U')

OPERADORES

Aritméticos

$a + b$ $a - b$ $a * b$ a / b
 $a \% b$ $-a$ $a ** b$

Lógicos

a **and** b a **or** b **not** b

Comparadores

$a \geq b$ $a == b$ $a != b$ $a > b$

COMENTARIOS

```
print("Not a comment")  
#print("Am a comment")
```

```
'''
```

```
print("We are in a comment")  
print ("We are still in a comment")  
'''
```

```
print("We are out of the comment")
```

FUNCIONES

```
def my_first_function(p1, p2):  
    return "Hello World!"
```

VARIABLES

```
a = 3  
b = int()
```

```
c = objeto()  
b = c
```

LISTAS

`L = [22, True, "una lista", [1, 2]]`

TUPLAS

`T = (22, True, "una tupla", (1, 2))`

DICCIONARIOS

`d = {"Kill Bill": "Tarantino",
"Amélie": "Jean-Pierre Jeunet"}`

```
def factorial(x):  
    if x == 0:  
        return 1  
    else:  
        return x * factorial(x - 1)
```

INPUT

```
ingreso = raw_input("Que nos quieres decir? ")  
print "Usted Ingreso" + ingreso
```

FORMATO

```
print "Usted %s" % ingreso  
print "El valor es %d" % 5
```

PROBLEMA

```
age = raw_input("Cual es tu edad? ")
height = raw_input("Cual es tu altura? ")
weight = raw_input("Cuanto pesas? ")

print "Entonces, tienes %r anhos, %r altura y %r kg." % (
    age, height, weight)
```



IF

```
nombre = "Arturo"
if nombre == "Arturo":
    nombre = "Arturo Rifa!"
elif nombre == "Youtube":
    nombre = "Hola Youtube"
else:
    nombre = "Quien eres?"
```

```
$nombre = "Arturo";
if ($nombre == "Arturo"){
    $nombre = "Arturo Rifa!";
}else if (nombre == "Youtube"){
    nombre = "Hola Youtube"
}else{
    nombre = "Quien eres?"
}
```


WHILE

```
contador = 0
while contador < 5:
    print "numero %i" % contador
    contador += 1
```

```
$count = 0;
while ($count < 5) {
    echo "Number ".$count;
    $count+=1;
}
```

FOR

```
for i in range(4):  
    print "Numero %i" % i
```

```
for ($i=0; $i < 5; $i++) {  
    echo "Numero ".$i;  
}
```

Code is read much more often than it is written.

IMPORTS

```
import sound.effects.echo  
sound.effects.echo.echofilter(...)
```

```
from sound.effects import echo  
echo.echofilter(...)
```

```
from sound.effects import echo as rev  
rev.echofilter(...)
```

```
class Estudiante(object):  
— def __init__(self, nombre, edad):  
—— self.nombre = nombre  
—— self.edad = edad  
  
— def hola(self):  
—— if self.edad > 18 :  
——— return '%s es mayor' % self.nombre  
—— else:  
——— return '%s es menor' % self.nombre
```

CONVERSIONES

```
>>> int(4.3)  
4
```

```
>>> float(4)  
4.0
```

```
>>> str(4.3)  
"4.3"
```

```
>>> list((4,4,5))  
[4,4,5]
```

COMUNES

```
>>> len("hey")  
3
```

```
>>> type(4)  
< type int >
```

```
>>> map(str,[5,2,1])  
['5','2','1']
```

```
>>> round(6.3243,1)  
6.3
```

```
>>> range(5)  
[0,1,2,3,4,5]
```

```
>>> sum([1,2,4])  
7
```

```
>>> sorted([5,2,1])  
[1,2,5]
```

```
>>> dir([5,2,1])
```

```
>>> help(sorted)
```

CLASES

```
class Estudiante(object):  
    def __init__(self, nombre, edad):  
        self.nombre = nombre  
        self.edad = edad  
    def hola(self):  
        return 'Mi nombre es is %s' % self.nombre
```

```
e = Estudiante("Arturo", 21)
```


MÉTODOS

`__cmp__(self, otro)`

Método llamado cuando se utilizan los operadores de comparación para comprobar si nuestro objeto es menor, mayor o igual al objeto pasado como parámetro.

`__len__(self)`

Método llamado para comprobar la longitud del objeto. Se utiliza, por ejemplo, cuando se llama a la función `len(obj)` sobre nuestro objeto. Como es de suponer, el método debe devolver la longitud del objeto.

EXCEPCIONES

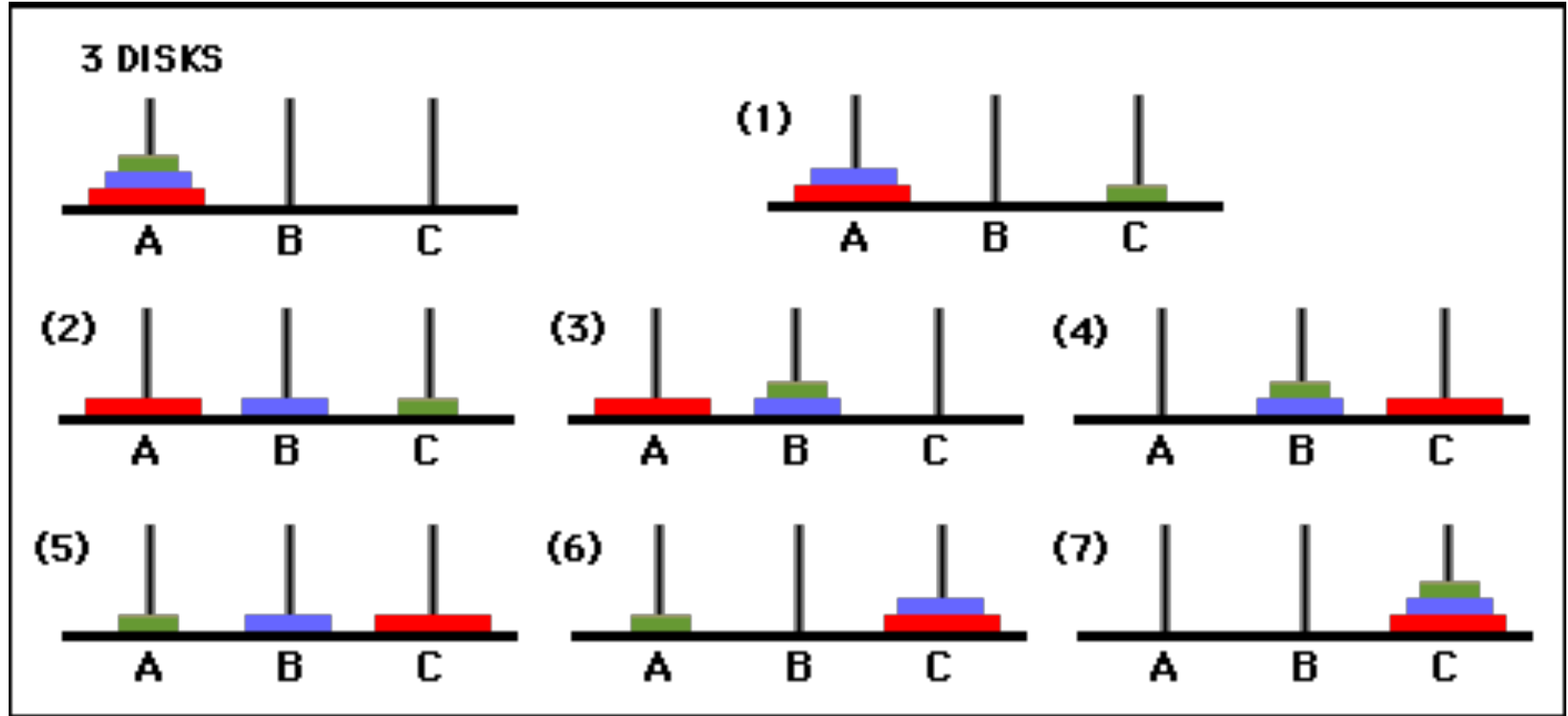
```
try:  
    r = 3 / 0  
except:  
    print "Division entre 0"
```

```
try {  
    $r = 3 / 0;  
} catch (Exception $e) {  
    echo "Hoyo Negro";  
}
```

99 BOTTLES OF BEER

```
verse = ""\  
{some} bottles of beer on the wall  
{some} bottles of beer  
Take one down, pass it around  
{less} bottles of beer on the wall  
""  
  
for bottles in range(99,0,-1):  
    print verse.format(some=bottles, less=bottles-1)
```

TORRES HANOI



RESUMEN TÉCNICO.

Una función recursiva es aquella que se repite a si misma hasta que valida una condición final. Esta condición final es casi siempre la solución del problema mas sencilla/mínima/básica. En las torres de hanoi esta solución es cuando no hay mas discos que mover puesto que hemos movido todos.

Cuando lo resolvemos con python la condición se representa con 0 (zero piezas que mover). Tampoco podríamos mover -1 piezas. Es por eso que tenemos que ir revisando en cada vuelta si aun hay piezas que mover. Si logras entender la estructura de cualquier problema de modo recursivo notarás que todos se resuelven con la misma.

El algoritmo de Hanoi cuando tenemos piezas que mover funciona de la siguiente manera.

- Mover el n-1 disco de el punto inicial a el pilar auxiliar.
- Mover el disco n del inicio a la torre final.
- Mover el n-1 disco de la torre auxiliar a la torre final.
- Repetir hasta que no existan discos en la torre inicial.

CARTA ANÓNIMA A PAQUITA GALLEGO



LECTURA DE ARCHIVOS

CONEXIÓN DE API

```
from urllib2 import urlopen

placeholder = urlopen('http://
loempixel.com/400/200/sports/Dummy-
Text/')

f = open('holder.jpg', 'wb')
f.write(placeholder.read())
f.close()
```


FRAMEWORKS

1

HTML CGI

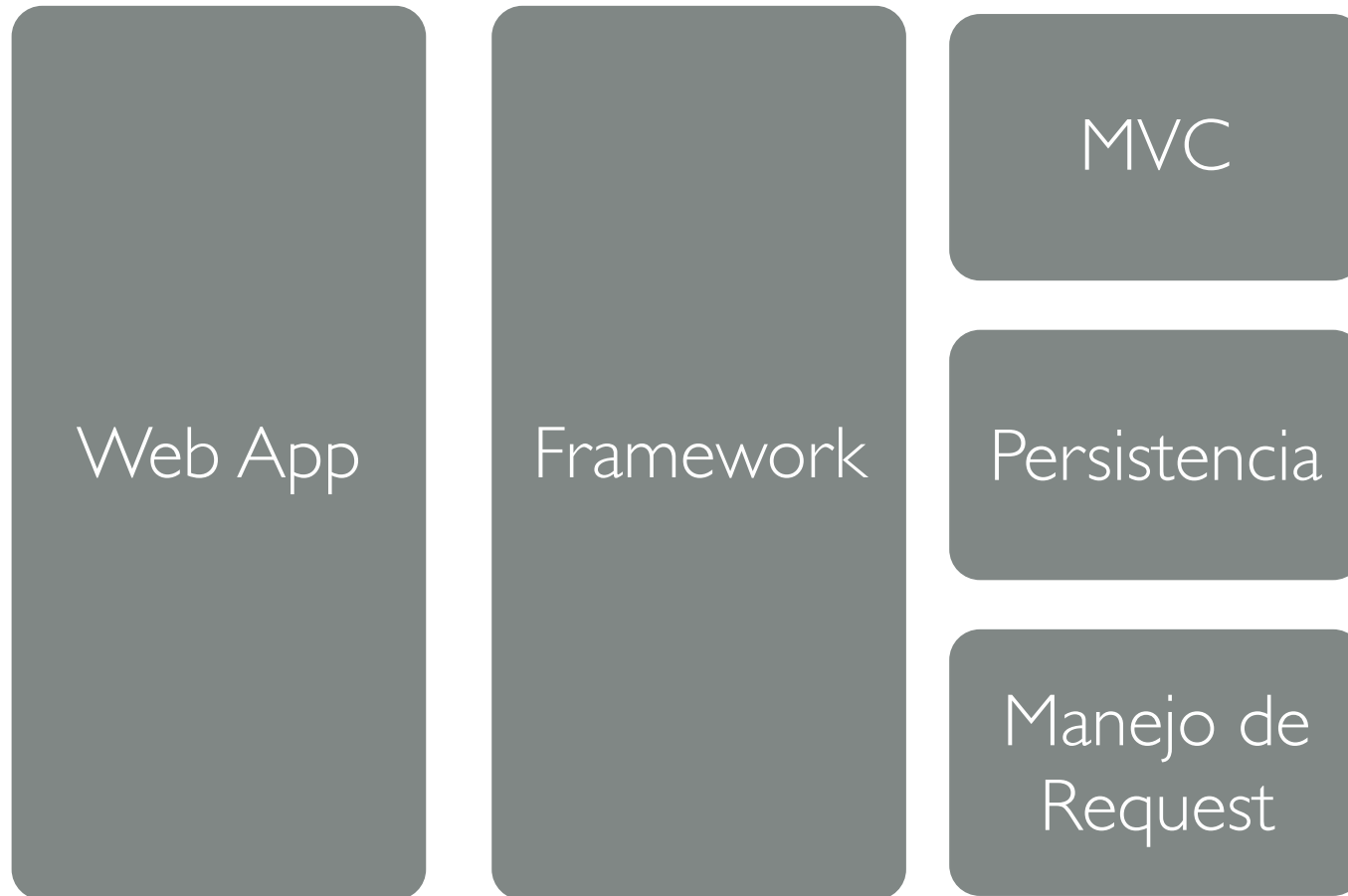
2

PHP JSP ASP

3

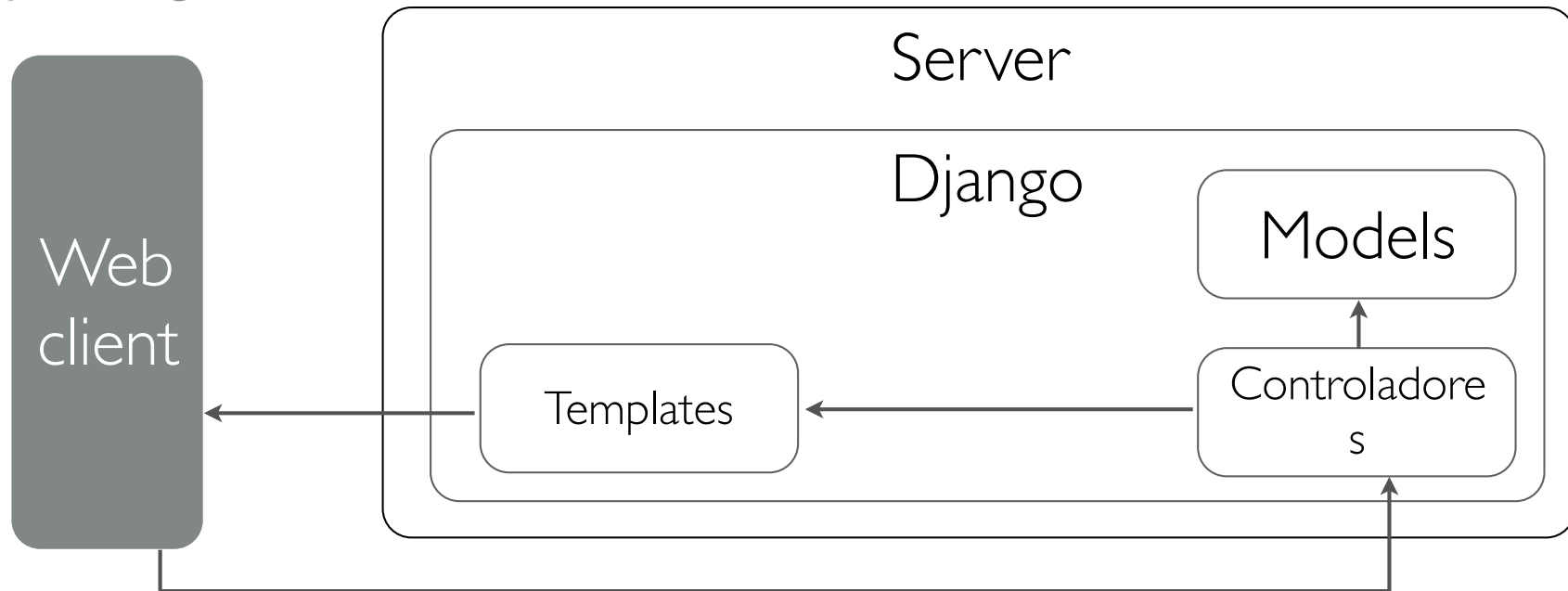
Django Rails

FRAMEWORK



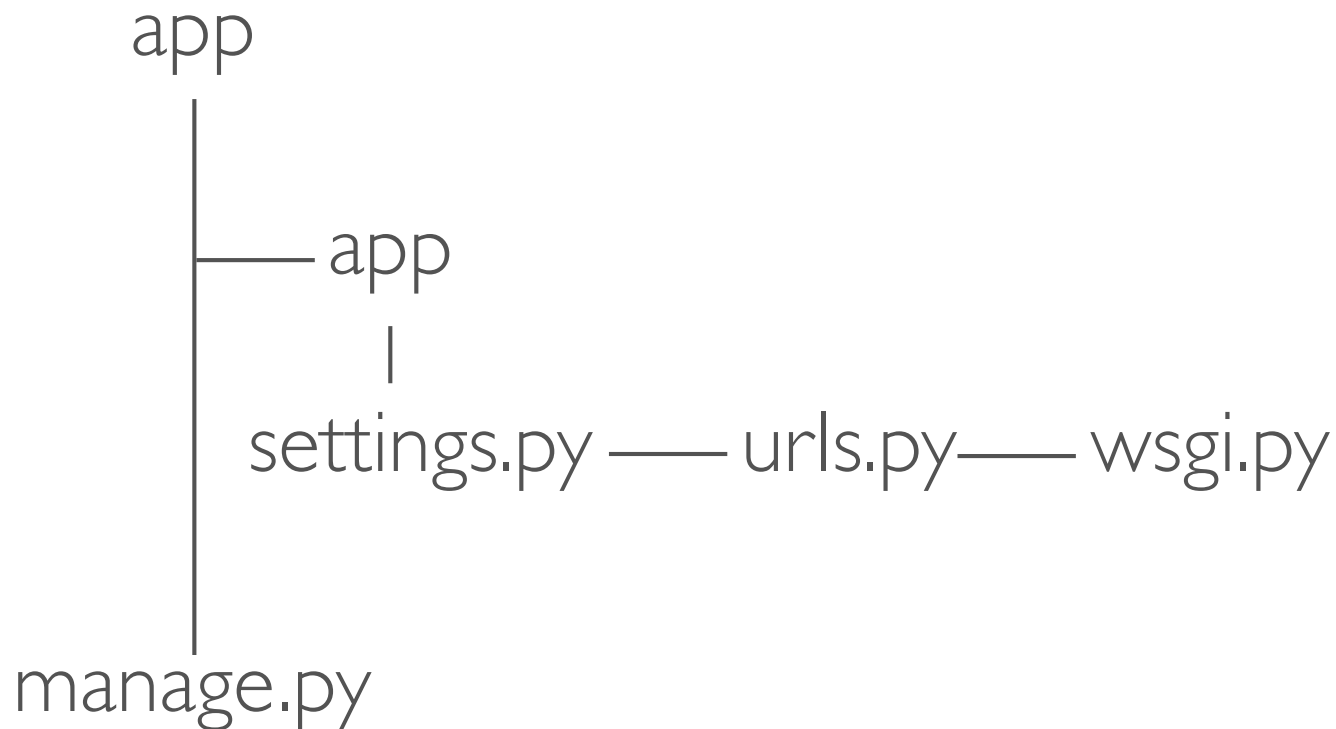
DJANGO

Django es un framework de desarrollo web de código abierto, escrito en Python, que cumple en cierta medida el paradigma del Modelo Vista Controlador.



DJANGO

django-admin.py startproject app

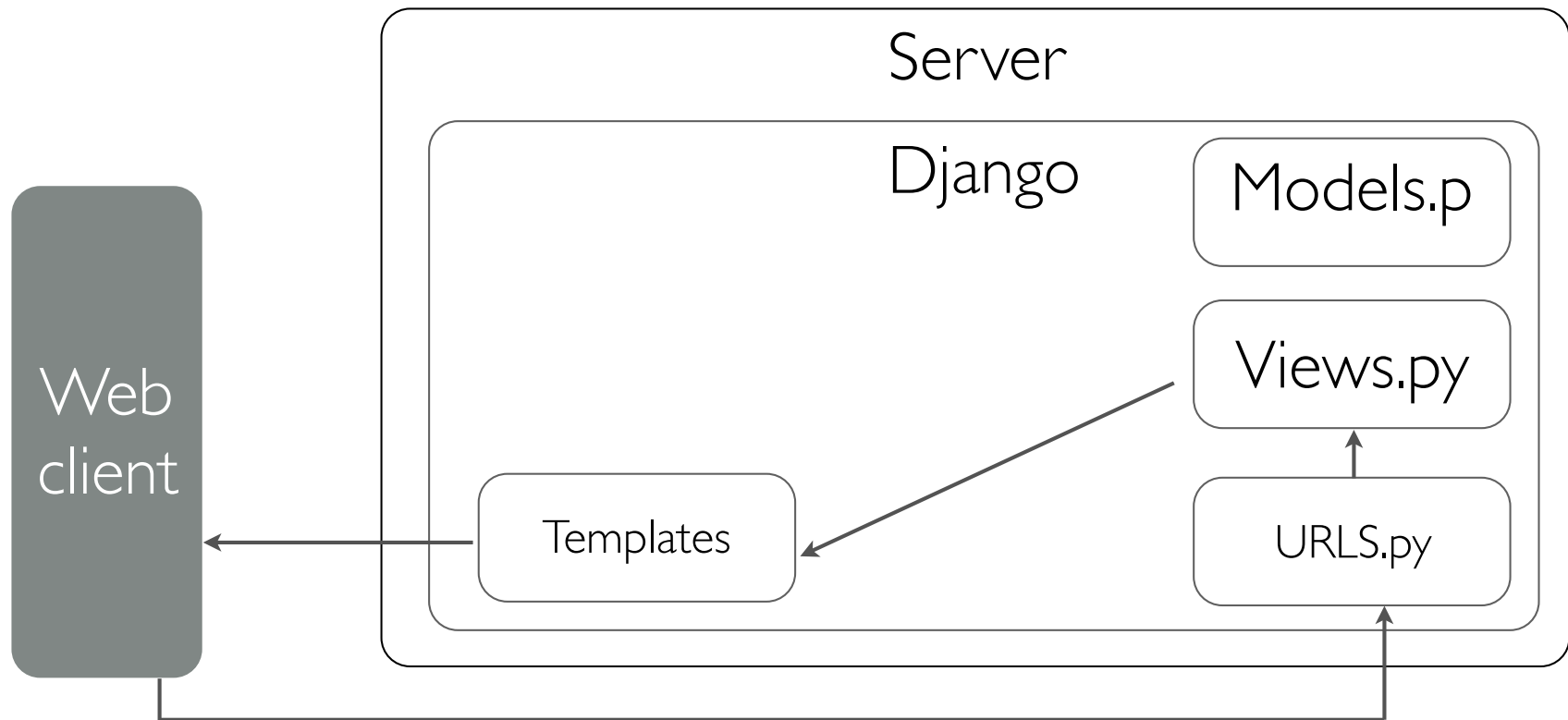


DJANGO

```
$cd app
```

```
$ python manage.py runserver
```

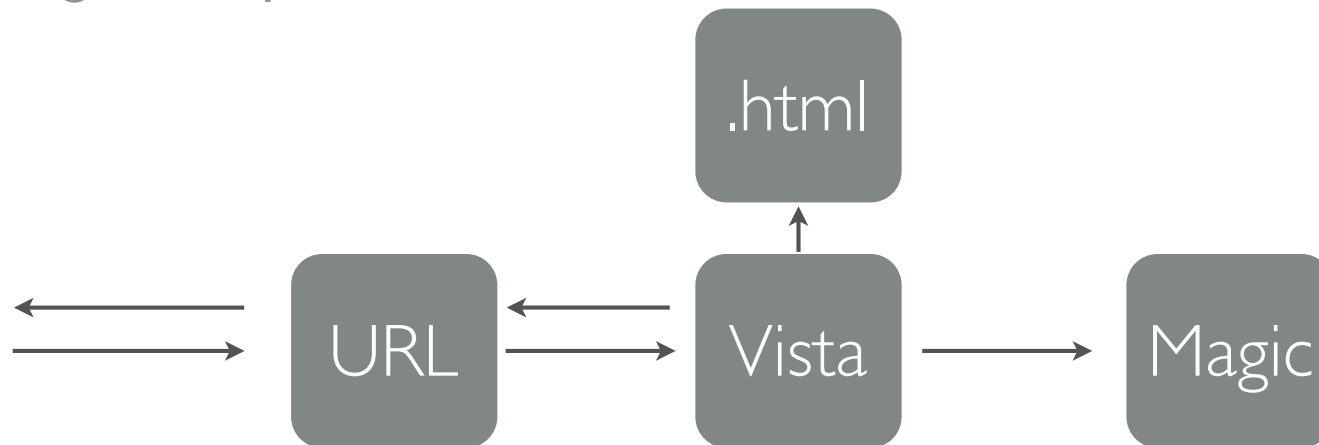
A DETALLE



URL Y VISTAS

Las Urls actuan como entrada a las peticiones y estas se resuelven con expresiones regulares.

Las URLs apuntan a una función en las vistas que definen el Django comportamiento



VIEWS.PY

Cada función de Views recibe como parametro un HttpRequest y devuelve un objeto

EJEMPLO

<http://mejorando.la/index>

```
from django.conf.urls.defaults import *  
from mysite.views import index_view  
urlpatterns = patterns('',  
    (r'^index/$', index_view),  
)
```

url.py

```
from django.http import HttpResponseRedirect  
def index_view(request):  
    html = "Bienvenido al curso"  
    return HttpResponseRedirect(html)
```

views.py

EJEMPLO

<http://mejorando.la/post/4>

```
from django.conf.urls.defaults import *
from mysite.views import index_view
urlpatterns = patterns('',
    (r'^post/(\d{1,2})/$', post),
)
```

```
from django.http import HttpResponse
from mysite.models import Post
def post(request,id):
    html = post.object.get(pk=id).title
    return HttpResponse(html)
```

TEMPLATES

Se basan en dos tipos de objetos: `Template()` y `Context()`.

- `Template()` contiene el string de salida que queremos devolver en el `HttpResponse` (normalmente HTML)
- `Context()` contiene un diccionario con los valores que dan contexto a una plantilla

Context `{'user': 'Freddier'}` \longrightarrow "Bienvenido, Freddier."

Template `"Bienvenido, {{ user }}."`

TEMPLATES

Settings.py



```
TEMPLATE_DIRS = (  
    '/home/django/templates',  
)
```

```
from django.http import HttpResponse  
from django.template.loader import get_template  
from django.template import Context  
from datetime import datetime  
def hora_actual(request):  
    ahora = datetime.now()  
    t = get_template('hora.html')  
    c = Context({'hora': ahora})  
    html = t.render(c)  
    return HttpResponse(html)
```

HOT TRICK

```
from django.shortcuts import render_to_response
from datetime import datetime
def hora_actual(request):
    now = datetime.now()
    return render_to_response('hora.html', {'hora': now})
```

TAGS Y FILTROS

filter {{ variable|filter }}

inline tag {% tag var1 var2 %}

block tag
{% tag var1 %}

...
{% endtag %}

TAGS

```
{% for elemento in lista %}  
    <li>{{ elemento }}</li>  
{% endfor %}
```

```
{% if username == "Juan" %}  
    Hola Juan  
{% else %}  
    Hola {{ usuario }},  
{% endif %}
```

FILTROS

```
{'fecha': datetime.datetime(2011, 9, 11, 17, 1, 59, 385323) }
```

```
{{ username|length }}      {{ username|wordcount }}
```

```
{{ username|upper }}
```

```
{{ fecha|date:"d M Y" }}      {{ fecha|timesince }}
```


TAGS

```
{% for elemento in lista %}  
    <li class="{% cycle 'rojo' 'azul' %}">{{ elemento }}</li>  
{% endfor %}
```

```
{% include "menu_bar.html" %}
```

```
{% for elemento in lista %}  
    <li >{{ forloop.counter }} - {{ elemento }}</li>  
{% endfor %}
```

MODELOS

```
from django.db import models
class Libro(models.Model):
    nombre = models.CharField(blank=True, max_length=100)
    creado = models.DateTimeField(blank=False)
    disponible = models.BooleanField(default=True)
```

TIPOS DE DATOS

- BigIntegerField
- BooleanField
- CharField
- CommaSeparatedIntegerField
- DateField
- DateTimeField
- DecimalField
- EmailField
- FileField
- FilePathField
- FloatField
- ImageField
- IntegerField
- NullBooleanField
- PositiveIntegerField
- PositiveSmallIntegerField
- SlugField
- SmallIntegerField
- TextField
- TimeField
- URLField
- XMLField
- ForeignKey
- ManyToManyField
- OneToOneField

PROPIEDADES

- null (True|False)
- blank (True|False)
- choices (lista)
- default (valor)
- editable (True|False)
- help_text (String)
- unique (True|False)
- primary_key
- unique_for_date
- unique_for_month
- unique_for_year

BASES DE DATOS

```
DATABASE_ENGINE = 'sqlite3'  
DATABASE_NAME = 'db.sqlite'  
DATABASE_USER = ''  
DATABASE_PASSWORD = ''  
DATABASE_HOST =
```

EJECUTAR

```
$ python manage.py syncdb
```

ORM

Query

```
books = Book.objects.all()  
books = Book.objects.all()[:100]  
books = Book.objects.all()[100:]
```

Insert

```
book = Book(nombre ='Art of war')  
book.save()
```

ORM

Update

```
Book.Objects.all().update( disponible= False)
```

Update

```
book.id  
book.disponible = False  
book.save()
```


ORM

Delete

```
Book.Objects.all().delete()
```

Update

```
book.id  
book.delete()
```

ORM

Get

```
Book.Objects.get(id='36')
```

```
Book.Objects.get(nombre='Art of war')
```

Filtros

```
Book.Objects.filter(disponible=True)
```

```
Book.Objects.exclude(disponible=True)
```

FILTROS

campo__lt=0
campo__lte=0
campo__in=[,]
campo__month=12
campo__startswith="
campo__startswith="
campo__endswith="
campo__iendswith="
campo__range=(,)
campo__year=2010
campo__exact="

campo__iexact="
campo__contains="
campo__icontains="
campo__isnull=TIF
campo__day=31
campo__gt=0
campo__gte=0

RELACIONES

OneToOneField

```
class Libro(models.Model):  
    autor = OneToOneField(Autor)
```

```
class Autor(models.Model):  
    ...
```

```
>> l.autor  
<Autor: Autor object>
```

```
>> a.libro  
<Libro: Libro object>
```

RELACIONES

ForeignKeyField

```
class Blog(models.Model):  
...
```

```
class Post(models.Model):  
    blog = ForeignKey(Blog)
```

```
>>b.post_set.all()  
[<Post: Post object>,...]
```

```
>> p.blog  
<Blog: Blog object>
```

RELACIONES

ManyToMany

```
class Post(models.Model):  
    tags = ManyToMany(Tags)
```

```
class Tags(models.Model):
```

```
tags = ManyToMany(Tags, related_name='tags')
```

```
>>p.tags.all()  
[<Tags: Tags object>, ...]
```

```
>>t.post_set.all()  
[<Post: Post object>, ...]
```

PROFILES

Problema: El modelo User de django.contrib.auth no puede contener toda la información que necesitamos.

- Username, Password, Name.... y poco más.

Solución: Definir un Profile (Un Modelo Agregado) para guardar esa información.

PROFILE

```
class Profile(models.Model):  
    user = models.OneToOneField(User, unique=True)  
    bio = models.CharField(blank=True, max_length=200)  
    AUTH_PROFILE_MODULE = "website.Profile"
```


ADMIN

1. Quitar los comentarios en las url
2. Instalar 'django.contrib.admin'
3. Registrar modelos con

```
from django.contrib import admin  
admin.site.register
```

FORMAS

Las Formas son objetos que nos permiten manejar datos ingresados por los usuarios.

En Django el framework se encarga de Pintar a HTML, Validar y guardar los datos ingresados

Esta basado en la idea de los modelos

FORMAS

```
from django import forms
```

```
class ContactoForm(forms.Form):  
    titulo = forms.CharField(max_length=100, label='Titulo')  
    mail = forms.EmailField(required=False)  
    mensaje = forms.CharField(widget= forms.Textarea)
```

FORMAS EN

```
<html> <body>
<h1>Título</h1>
{% if form.errors %}
    <p style="color: red;">
        {{ form.errors|pluralize }}.
    </p>
{% endif %}
<form action="" method="post"> <table>
    {{ form.as_table }}
</table>
<input type="submit" value="Submit"> </form>
</body> </html>
```

FORMS EN VIEWS

```
from django.shortcuts import render_to_response
from site.app.forms import ContactoForm
```

```
def contact(request):
    if request.method == 'POST':
        form = ContactoForm(request.POST)
        if form.is_valid():
            cd = form.cleaned_data
            send_mail(cd['subject'], cd['message'],)
            return HttpResponseRedirect('/contacto/enviado/')
    else:
        form = ContactoForm()
    return render_to_response('contact_form.html', {'form': form})
```

TIPOS DE DATO

BooleanField

CharField

ChoiceField

TypedChoiceField

DateField

DateTimeField

DecimalField

EmailField

FileField

FilePathField

FloatField

ImageField

IPAddressField

MultipleChoiceField

NullBooleanField

RegexField

SlugField

TimeField

URLField

ComboField

MultiValuefield

SplitDateTimeField

ModelChoiceField

ModelMultipleChoiceField

MODELFORMS

```
from django.db import models
```

```
class Libro(models.Model):  
    nombre = models.CharField(blank=True, max_length=100)  
    creado = models.DateTimeField(blank=False)  
    disponible = models.BooleanField(default=True)
```

```
from django import forms  
from books.models import Author
```

```
class AuthorForm(forms.ModelForm):  
    class Meta:  
        model = Author  
        exclude = ('country',)
```

DEPLOYMENT

DJANGO

Curso Python y Django



Arturo Jamaica
jamaica@brounie.com