# django

http://www.djangoproject.com/


Fco Javier Lucena Lucena

22 de Abril de 2010

# Índice

# FrameWork

- Esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

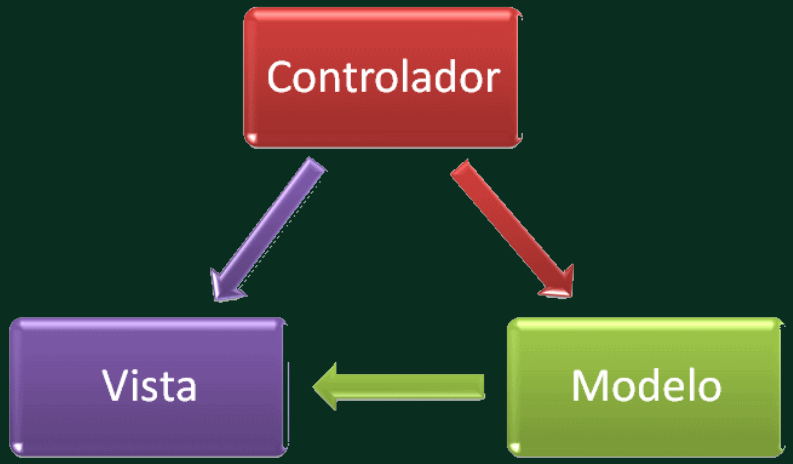
Framework Wikipedia

# FrameWork

- Ventajas
  - El programador no necesita plantearse una **estructura global** de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar"
  - Facilita la **colaboración**. Definir y estandarizar va a ahorrar tiempo y trabajo a los desarrollos colaborativos.
  - Es más fácil encontrar herramientas (utilidades, librerías) adaptadas al framework concreto para **facilitar el desarrollo**

# Patrón MVC

- Es una forma de desarrollar software en la que el código para definir y acceder a los datos (**Modelo**) esta separada de la lógica de envió de solicitudes (**Controlador**), que a su vez esta separado de la interfaz de usuario (**Vista**)



# ¿Qué es Django?

- Es un **Framework** de alto nivel desarrollado en **Python** que permite desarrollar aplicaciones rápidamente de forma clara y estructurada.

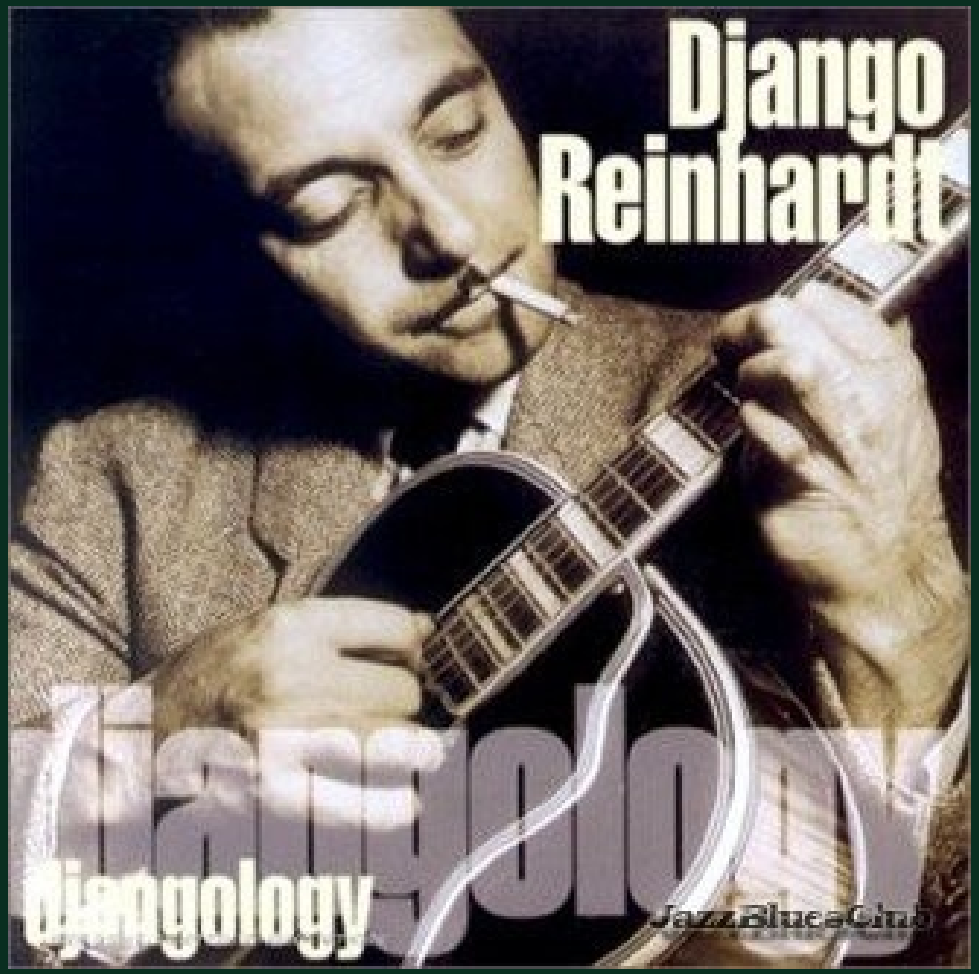
# Historia

1. Escribir una aplicación web desde cero
2. Escribir otra aplicación web desde cero
3. Comprender que la aplicación del paso 1 **comparte** varios elementos con la aplicación del paso 2
4. **Reestructurar** el código para que la aplicación del paso 1 comparta código con la aplicación del paso 2
5. Repetir los pasos 2-4 varias veces
6. Comprender que hemos inventado un marco de trabajo

# Historia

- **Otoño de 2003**, los programadores del rotativo Lawrence Journal-World **Adrian Holovaty y Simon Willinson** comenzaron a utilizar Python para crear aplicaciones
- Este entorno periodístico requería que las aplicaciones se entregaran **en dias u horas**.
- **Verano de 2005**, se publico como código abierto, con el nombre de Django. El equipo de trabajo ya incluía a **Jacob Kaplan-Moss**

# Historia



# ¿Why use it?

- Reutilizar código
- Desarrollo rápido
- Automatización de tareas
- DRY (Don't repeat yourself)
- Conectividad y Extensibilidad de componentes

# Instalación

## Python

# Instalación

http://www.djangoproject.com/download/

```
>>> import django
>>> dango.VERSION
```

# Proyecto

- django-admin.py
  - startproject
- manage.py
  - startapp
  - runserver
  - syncdb
  - shell

# Proyecto

- Proyecto
  - Contiene nuestro sitio web
  - Almacena las aplicaciones
  - Se establece el **controlador**
- Aplicación
  - Se establece el **modelo** y la **vista**

# Proyecto

- python django-admin.py startproject website
- Crea un directorio website que contiene:
  - __init.py__ : Archivo para tratar el directorio website como un paquete.
  - manage.py: Utilidad para interactuar con el proyecto. *python manage.py help*
  - settings.py: Configuración y opciones
  - urls.py: “tabla de contenidos” del website

# Proyecto

- ruta/python manage.py startapp app1
  - __init.py__: Archivo para tratar el directorio website como un paquete
  - views.py: vistas para las urls de nuestra web
  - models.py: objetos mapeados a la base de datos

# Proyecto

```
fran@lucena:~/adjango$ tree
.
|-- __init__.py
|-- __init__.pyc
|-- app1
|   |-- __init__.py
|   |-- models.py
|   |-- tests.py
|   `-- views.py
|-- manage.py
|-- settings.py
|-- settings.pyc
`-- urls.py

1 directory, 10 files
fran@lucena:~/adjango$
```

# Proyecto

- Servidor de desarrollo

  python manage.py runserver

```
fran@lucena:~/adjango$ python manage.py runserver
Validating models...
0 errors found

Django version 1.1.1, using settings 'adjango.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
[21/Apr/2010 15:21:16] "GET / HTTP/1.1" 200 2055
```

# Vistas y URLs

- ¿Os resulta cansino lo de “Hola Mundo !” cuando aprendéis un nuevo lenguaje ?
  - Con todos mis respetos al “Hola Mundo” ehh
- ¿ Alguna idea innovadora para sustituirlo ?
  - Y lo hacemos con django.. por ejemplo en turco:

**Merhaba Dünya!**

# Vistas y URLs

- Vistas
  - Fichero views.py en adjango
- Urls
  - Fichero urls.py en adjango

# Expresiones Regulares

http://www.cheat-sheets.org/

- carácter: .
- dígito: \d
- [A-Z]
- [A-Z][a-z]
- +
- ?
- *
- {1,3}

# Vistas y URLs

"Una vista es simplemente una función python que recibe un objeto **HttpRequest** como primer parámetro y devuelve una instancia de **HttpResponse**"

# Ejemplo

- URL Estática
  - import datetime
  - variable html
  - nueva url

# Ejemplo

- URL Dinámica
- URLS.py
  - Nueva url: expresión regular
    - Entre paréntesis indicamos parámetro
- VIEWS.py
  - nuevo parámetro a la vista
  - controlar excepciones
    - hours es int (python fuertemente tipado)
    - url no cumple exp. reg
  - import Http404
    - ValueError

# Vistas y URLs

- HttpRequest
  - Encapsula la petición del un cliente, la sesión, el contenido de un POST
- HttpResponse
  - Respuesta que se devuelve al cliente: text/html
- Http404
  - Django muestra una pagina informativa del error

# Plantillas

- Los ejemplos anteriores no son adecuados:
  - Una modificación en el diseño de la web requiere una modificación en el código Python.
  - Escribir código Python y diseñar HTML son disciplinas diferentes
  - Es mucho mas eficiente que los programadores trabajen sobre el código python y los diseñadores sobre el HTML al mismo tiempo

# Plantillas

- Son ficheros de texto plano, se usan para generar ficheros de texto html xml
- Contienen:
  - Texto estático
  - Variables
    - Se asignan valores al evaluar la plantilla
  - Marcas
    - Manejan el flujo de control de la plantilla
- Se reutiliza código y diseño.

# Plantillas

Ejemplo.html

# Plantillas

- Ejemplo: Uso en vistas
- settings.py template_dirs
- Variables, filtros y marcas
- render_to_response

# Modelos

- **ORM (Object Relational Mapper)**, permite acceder a datos almacenados en una base de datos relacional con una interfaz orientada a

  Objetos.
- Las **tablas** de la base de datos serán **clases**.
- Las **tuplas** serán instancias, **objetos**.
- Se define en **models.py**

# Modelos

- django.db.models
- Si no se define una clave primaria django la creara automaticamente “id”
- Tipos de campos:  AutoField, CharField,

  BooleanField, DateField, TimeField, DateTimeField, (Positive)IntegerField, File/ImageField, DecimalField,

  IPAddressField, ...

# Modelos

- Pueden representarse las 3 relaciones típicas entre tablas del modelo relacional mediante campos en django.db.models
  - Uno a muchos: ForeignKey
  - Muchos a muchos: ManyToManyField
  - Uno a uno: OneToOneField

  Se declaran sólo en uno de los modelos involucrados en la relación.

# Modelos

- Python manage.py syncdb
- Python manage.py slqall

# Modelos

- Ejemplo aplicación encuestas

# Admin

- Activar
  - settings.py
    - installed_app
- Crear tablas del modelo syncdb
  - La primera vez solicitar crear superusuario
    - Python manage.py createsuperuser
- URLs.py
  - Eliminar comentarios
- htttp://localhost:8000/admin/

# Admin

- Para que aparezcan nuestras aplicaciones
  - Añadir modulo admin.py
- Ejemplo en aplicación encuestas
- Gestión Usuarios

# Fuentes

- http://docs.djangoproject.com/
- http://djangobook.com
- Python Web development with Django
  Jeff Forcier, Paul Bissex, Wesley Chun.
- The Definitive Guide to Django: Web Development Done Right
  Adrian Holovaty, Jacob Kaplan-Moss
- Practical Django Projects
  James Bennett
- Sams Teach Yourself Django in 24 Hours
  Brad Dayley

# Licencia

Fco Javier Lucena Lucena

fran.lucena@gmail.com

http://www.franlucena.es/