



FIG. 2.
PONY "MAGIC"

Sistemas Web Desconectados

Release 1

van Haaster, Diego Marcos; Defossé, Nahuel

August 12, 2009

Índice general

1. Tecnologías del servidor	3
1.1. CGI	3
1.2. WSGI	3
1.3. Lenguajes interpretados	3
1.4. Frameworks	3
1.5. Django	5
1.6. El Mapeador Objeto-Relacional de Django	8
2. Glosario	11
3. Indices, glosario y tablas	13
Índice	15

Índice:

Tecnologías del servidor

1.1 CGI

CGI es bla

1.2 WSGI

WSGI es CGI para Python.

1.3 Lenguajes interpretados

1.3.1 PHP

1.3.2 Ruby

1.3.3 Python

Python es un lenguaje interpretado.

1.4 Frameworks

Un framework web es una abstracción en la cual un código común que provee una funcionalidad genérica puede ser personalizadas por el programador de manera selectiva para brindar una funcionalidad específica.

Se suele decir que los frameworks son similares a las bibliotecas de software (a veces llamadas librerías) dado que proveen abstracciones reusables de código a las cuales se accede mediante una API bien definida.

Sin embargo, podemos encontrar ciertas características que diferencian al framework de una librería o aplicaciones normales de usuario:

- **Inversion de control** Al contrario que las bibliotecas en las aplicaciones de usuario, en un framework, el flujo de control no es manejado por el llamador, sino por el framework. Es decir, cuando se utilizan bibliotecas o programas de usuario como soporte para brindar funcionalidad, estas son llamados o invocados en el código de aplicación principal que es definido por el usuario. En un framework, el flujo de control principal está definido por el framework.
- **Comportamiento por defecto definido** Un framework tiene un comportamiento por defecto definido. En cada componente del framework, existe un comportamiento genérico con alguna utilidad, que puede ser redefinido con funcionalidad del usuario.
- **Extensibilidad** Un framework suele ser extendido por el usuario mediante redefinición o especialización para proveer una funcionalidad específica.
- **No modificabilidad del código del framework** En general no se permite la modificación del código del framework. Los programadores pueden extender el framework, pero no modificar su código.

Los diseñadores de frameworks tienen como objetivo facilitar el desarrollo de software, permitiendo a los programadores enfocarse en cumplimentar los requerimientos del análisis y diseño, en vez de dedicar tiempo a resolver los detalles comunes de bajo nivel. En general la utilización de un framework reduce el tiempo de desarrollo.

Por ejemplo, en un equipo donde se utiliza un framework web para desarrollar un sitio de banca electrónica, los desarrolladores pueden enfocarse en la lógica necesaria para realizar las extracciones de dinero, en vez de la mecánica para preservar el estado entre las peticiones del navegador.

Sin embargo, se suele argumentar que los frameworks pueden ser una carga, debido a la complejidad de sus APIs o la incertidumbre que genera la existencia de varios frameworks para un mismo tipo de aplicación. A pesar de tener como objetivo estandarizar y reducir el tiempo de desarrollo, el aprendizaje de un framework suele requerir tiempo extra en el desarrollo, aunque posteriores desarrollos pueden verse beneficiados de este aprendizaje inicial.

1.4.1 Frameworks web

Los frameworks web surgen como evolución histórica a CGI.

1.4.2 Model View Controller

Modelo-Vista-Controlador, MVC ¹, es un patrón de diseño arquitectural que separa la aplicación del modelo de datos, de la lógica de negocio y la interfase de usuario.

Muchos frameworks web se respetan en mayor o menor grado el patrón MVC, aliviando tareas comunes en cada una de estas capas.

En la capa de *modelo de datos*, muchos frameworks proveen lo que se conoce como Mapeador Objeto Relacional o ORM. Su función es transformar una representación de objetos en tablas relacionales. El ORM se encarga de presentar una interfase orientada a objetos (API) para representar el modelo y brindar mecanismos para creación, actualización y eliminación de entidades, así como también una API para consultas. Algunos ORM permiten escribir el modelo utilizando la API OO ² y generar las tablas relacionales a partir de esta representación.

En la capa de *controlador* los frameworks proveen una forma de

1.4.3 Mapeador Objeto-Relacional

Rails

Symfony

¹ Del inglés *Model View Controller*

² API Orientada a Objetos

1.5 Django

Acá tenemos que justificar por que django

Django es un framework web escrito en Python el cual sigue vagamente el concepto de Modelo Vista Controlador. Ideado inicialmente como un administrador de contenido para varios sitios de noticias, los desarrolladores encontraron que su CMS era lo suficientemente genérico como para curbir un ámbito más aplio de aplicaciones.

En honor al músico Django Reinhart, fue liberado el código base bajo la licencia *BSD* en Julio del 2005 como Django Web Framework. El slogan del framework fue “Django, Él framework para perfeccionistas con fechas límites”³.

En junio del 2008 fue anunciada la cereación de la Django Software Fundation, la cual se hace cargo hasta la fecha del desarrollo y mantenimiento.

Los orígenes de Django en la administración de páginas de noticias son evidentes en su diseño, ya que proporciona una serie de características que facilitan el desarrollo rápido de páginas orientadas a contenidos. Por ejemplo, en lugar de requerir que los desarrolladores escriban controladores y vistas para las áreas de administración de la página, Django proporciona una aplicación incorporada para administrar los contenidos que puede incluirse como parte de cualquier proyecto; la aplicación administrativa permite la creación, actualización y eliminación de objetos de contenido, llevando un registro de todas las acciones realizadas sobre cada uno (sistema de logging o bitácora), y proporciona una interfaz para administrar los usuarios y los grupos de usuarios (incluyendo una asignación detallada de permisos).

Con Django también se distribuyen aplicaciones que proporcionan un sistema de comentarios, herramientas para syndicar contenido via RSS y/o Atom, “páginas planas” que permiten gestionar páginas de contenido sin necesidad de escribir controladores o vistas para esas páginas, y un sistema de redirección de URLs.

Django como framework de desarrollo consiste en un conjunto de utilidades de consola que permiten crear y manipular proyectos y aplicaciones.

1.5.1 Estructuración de un proyecto en Django

Durante la instalación del framework en el sistema del desarrollador, se añade al PATH un comando con el nombre `django-admin.py`. Mediante este comando se crean proyectos y se los administra.

Un proyecto se crea mediante la siguiente orden:

```
$ django-admin.py startproject mi_proyecto # Crea el proyecto mi_proyecto
```

Un proyecto es un paquete Python que contiene 3 módulos:

- **manage.py** Interfase de consola para la ejecución de comandos
- **urls.py** Mapeo de URLs en vistas (funciones)
- **settings.py** Configuración de la base de datos, directorios de plantillas, etc.

En el ejemplo anterior, un listado gerárquico del sistema de archivos mostraría la siguiente estructura:

```
mi_proyecto
|-- __init__.py
|-- manage.py
|-- settings.py
'-- urls.py
```

El proyecto funciona como un contenedor de aplicaciones que ser rigen bajo la misma base de datos, los mismos templates, las mismas clases de middleware entre otros parámetros.

Analicemos a continuación la función de cada uno de estos 3 módulos.

³ Del ingles “The Web framework for perfectionists with deadlines”

1.5.2 Módulo settings

Este módulo define la configuración del proyecto, siendo sus atributos principales la configuración de la base de datos a utilizar, la ruta en la cual se encuentran los medios estáticos, cuál es el nombre del archivo raíz de urls (generalmente `urls.py`). Otros atributos son las clases middleware, las rutas de los templates, el idioma para las aplicaciones que soportan *i18n*, etc.

Al ser un módulo del lenguaje python, la configuración se puede editar muy facilmente a diferencia de configuraciones realizadas en XML, además de contar con la ventaja de poder configurar en caliente algunos parametros que así lo requieran.

Un parametro fundamental es la lista denominada `INSTALLED_APPS` que contiene los nombres de las aplicaciones instaladas en le proyecto.

1.5.3 Módulo manage

Esta es la interfase con el framework. Éste módulo es un script ejecutable, que recibe como primer argumento un nombre de comando de django.

Los comandos de django pemiten, permiten entre otras cosas:

- **startapp** <nombre de aplicación> Crear una aplicación
- **runserver** Correr el proyecto en un servidor de desarrollo.
- **syncdb** Generar las tablas en la base de datos de las aplicaciones instaladas

El resultado de el comando **startapp** en el ejemplo anterior genera el siguiente resultado:

```
mi_proyecto
|-- mi_aplicacion
|   |-- __init__.py
|   |-- models.py
|   |-- tests.py
|   |-- views.py
|-- __init__.py
|-- manage.py
|-- settings.py
'-- urls.py
```

1.5.4 Módulo urls

Este nombre de módulo aparece a nivel proyecto, pero también puede aparecer a nivel aplicación. Su misión es definir las asociaciones entre URLs y vistas, de manera de que el framework sepa que vista utilizar en función de la URL que está requiriendo el clinete. Las URLs se escriben mediante expresiones regulares. Se suele aprovechar la posibilidad del modulo de expresiones regulares del lenguaje python, que permite recuperar gurpos nombrados (en contraposición al enfoque ordinal tradicional).

La asociación url-vistas se define en el módulo bajo el nombre *urlpatterns*. También es posible derivar el tratado de una parte de la expresión reuglar a otro módulo de urls. Generalmente esto ocurre cuando se desea delegar el tratado de las urls a una aplicación particular.

Ej: Derivar el tratado de todo lo que comience con la cadena `personas` a al módulo de urls de la aplicación `personas`.

```
(r'^personas', include('mi_proyecto.personas.urls'))
```

1.5.5 Estructura de una aplicación Django

Una aplicación es un paquete python que consta de un módulo models y un módulo views.

1.5.6 Módulo models

Cada vez que se crea una aplicación, se genera un módulo models.py, en el cual se le permite al programador definir modelos de objetos, que luego son transformados en tablas relacionales ⁴.

1.5.7 Módulo views

Cada aplicación posee un módulo views, donde se definen las funciones que atienden al cliente y son activadas gracias a el mapeo definido en el módulo urls del proyecto o de la aplicación.

Las funciones que trabajan como vistas deben recibir como primer parámetro el request y opcionalmente parámetros que pueden ser recuperados del mapeo de urls.

Dentro del módulo de urls

```
# Tras un mapeo como el siguiente
(r'^persona/(?P<id_persona>\d)/$', mi_vista)
# la vista se define como
def mi_vista(request, id_persona):
    persona = Personas.objects.get(id=id_persona)
    datos = {'persona': persona, }
    return render_to_response('plantilla.html', datos)
```

1.5.8 El ciclo de una petición

Cada vez que un browser realiza una petición a un proyecto desarrollado en django, la petición HTTP pasa por varias capas.

Inicialmente atraviesa los Middlewares, en la cual, el middleware de Request, empaqueta las variables del request en una instancia de la clase Request.

Luego de atravesar los middlewares de request, mediante las definiciones de URLs, se selecciona la vista a ser ejecutada.

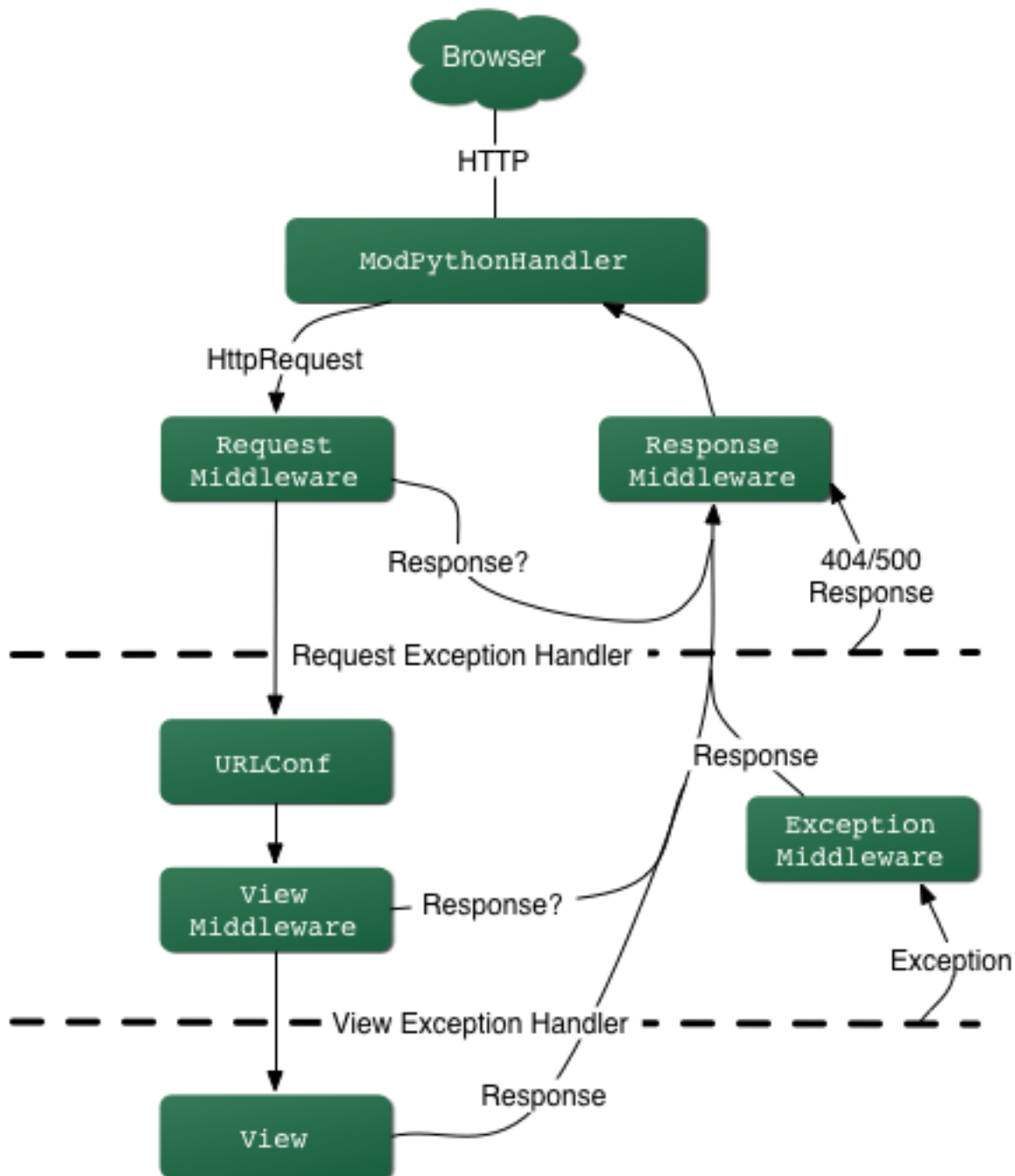
Una vista es una función que recibe como primer argumento el request y opcionalmente una serie de parámetros que puede recuperar de la propia URL.

Dentro de la vista se suelen hacer llamadas al ORM, para realizar consultas sobre la base de datos. Una vez que la vista a completado la lógica, genera un mapeo que es transferido a la capa de templates.

El template rellena sus comodines en función de los valores del mapeo que le entrega la vista. Un template puede poseer lógica muy básica (bifurcaciones, bucles de repetición, formateo de datos, etc).

El template se entrega como un HttpResponse. La responsabilidad de la vista es entregar una instancia de esta clase.

⁴ Mediante el comando syncdb del módulo manage del proyecto



1.6 El Mapeador Objeto-Relacional de Django

1.6.1 Modelos

Los modelos son la fuente de información sobre los datos de la aplicación. Esencialmente están compuestos de campos y comportamiento propio de los datos almacenados. Generalmente, un modelo se corresponde con una tabla en la base de datos.

Dentro de un proyecto los modelos se definen por aplicacion en el modulo `models.py`.

Un modelo es una clase Python que hereda de `django.db.models.Model` y cada atributo representa un campo requerido por el modelo de datos de la aplicación. Con esta informacion Django genera automaticamente una *API* de acceso a los datos en la base.

Este modelo de ejemplo define una `Persona` que encapsula los datos correspondientes al nombre y el apellido.

```
from django.db import models

class Persona(models.Model):
    nombre = models.CharField(max_length = 30)
    apellido = models.CharField(max_length = 30)
```

nombre y apellido son atributos de clase

```
CREATE TABLE miapp_persona (
    "id" serial NOT NULL PRIMARY KEY,
    "nombre" varchar(30) NOT NULL,
    "apellido" varchar(30) NOT NULL
);
```

1.6.2 Consultas

bala

1.6.3 Administradores de consultas

Estos objetos representan la interfase de comunicacion con la base de datos. Cada modelo tiene por lo menos un administrador para acceder a los datos almacenados.

Glosario

API [Application-Programming-Interface](#); conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

DOM [Document-Object-Model](#); interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, un modelo estándar sobre cómo pueden combinarse dichos objetos, y una interfaz estándar para acceder a ellos y manipularlos.

JSON [JavaScript-Object-Notation](#); formato ligero para el intercambio de datos.

RPC [Remote-Procedure-Call](#); es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

field An attribute on a [model](#); a given field usually maps directly to a single database column.

generic view A higher-order [view](#) function that abstracts common idioms and patterns found in view development and abstracts them.

model Models store your application's data.

MTV hola

MVC [Model-view-controller](#); a software pattern.

project A Python package – i.e. a directory of code – that contains all the settings for an instance of Django. This would include database configuration, Django-specific options and application-specific settings.

property Also known as “managed attributes”, and a feature of Python since version 2.2. From [the property documentation](#):

Properties are a neat way to implement attributes whose usage resembles attribute access, but whose implementation uses method calls. [...] You could only do this by overriding `__getattr__` and `__setattr__`; but overriding `__setattr__` slows down all attribute assignments considerably, and overriding `__getattr__` is always a bit tricky to get right. Properties let you do this painlessly, without having to override `__getattr__` or `__setattr__`.

queryset An object representing some set of rows to be fetched from the database.

slug A short label for something, containing only letters, numbers, underscores or hyphens. They're generally used in URLs. For example, in a typical blog entry URL:

`http://www.djangoproject.com/weblog/2008/apr/12/spring/`

the last bit (`spring`) is the slug.

template A chunk of text that separates the presentation of a document from its data.

view A function responsible for rendering a page.

BSD ve ese de

i18n La internacionalización es el proceso de diseñar software de manera tal que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar cambios de ingeniería ni en el código. La localización es el proceso de adaptar el software para una región específica mediante la adición de componentes específicos de un locale y la traducción de los textos, por lo que también se le puede denominar regionalización. No obstante la traducción literal del inglés es la más extendida.

Indices, glosario y tablas

- *Índice*
- *Índice de Módulos*
- *Glosario*

Índice

A

API, [11](#)

B

BSD, [12](#)

D

DOM, [11](#)

F

field, [11](#)

G

generic view, [11](#)

I

i18n, [12](#)

J

JSON, [11](#)

M

model, [11](#)

MTV, [11](#)

MVC, [11](#)

P

project, [11](#)

property, [11](#)

Q

queryset, [11](#)

R

RPC, [11](#)

S

slug, [11](#)

T

template, [12](#)

V

view, [12](#)