TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

Introducción al desarrollo de páginas web

Trabajo de investigación Django

Prof. Ericka Solano Fernández

Integrantes:

Andrés Sánchez Gómez - 2017239278

Diego Velásquez Castro - 2017107723

María Venegas Berrocal - 2017097489

Rodrigo Venegas Vega - 2017047627

Verano 2019-2020

Introducción

Django es un framework multiplataforma basado en Python enfocado en el desarrollo web. Este framework cuenta con todo el stack de desarrollo web, es decir, base de datos, modelos, backend y frontend o en otras palabras, el fullstack. Debido a lo anterior y a cómo se estructura este framework, el desarrollo en el mismo es ágil y sencillo. Según sus propios autores, Django es ideal en situaciones donde se quiera desarrollar software de manera rápida sin perder los principios básicos de programación.

En este documento se analizarán las principales características de este framework, ventajas, importancia y su arquitectura. Esto, con el objetivo de conocer cuando y porqué se debería utilizar Django en un proyecto, cuáles son sus implicaciones y el porqué de su alta popularidad.

Historia

Django fue creado en el año 2003 por los programadores Adrian Holovaty y Simon Willison, mientras trabajaban para el equipo The World Online, quienes eran responsables de producir y mantener sitios web de noticias. En esa época, los periodistas y directivos de los noticieros exigían a los programadores agregar nuevas características a las aplicaciones, e incluso, crear nuevos programas usualmente con muy poco tiempo de preaviso. De esta manera, Simon y Adrian se vieron en la necesidad de crear un framework que les facilitara esta tarea y así poder cumplir con los tiempos de entrega.

Luego, el equipo que ahora contaba con la colaboración del desarrollador Jacob Kaplan-Moss, decidió liberar el desarrollo del framework bajo una licencia BSD en el año 2005, con el fin de que desarrolladores de todo el mundo aportaran con su trabajo para lograr ampliar y mejorar Django.

Actualmente, a pesar de que el desarrollo del framework es libre, los desarrolladores originales aportan una guía orientada a su crecimiento. Además, The World Online facilita aspectos como marketing y hosting del sitio web de Django.

Principales características

Django cuenta con todo lo necesario para realizar el desarrollo del full stack, es decir, es un framework de desarrollo completo. Por ejemplo, Django proporciona de manera oficial conectores a distintas bases de datos como PostgreSQL, MariaDB, MySQL, Oracle y SQLite. Sin embargo, gracias a su comunidad, existen incluso más conectores que no serán mencionados, mantenidos por la comunidad. Además, gracias a su arquitectura que se ampliará más adelante, la comunicación cliente servidor está implícita en el framework.

Debido a que este framework es de código abierto y alta popularidad, Django cuenta con comunidad altamente activa, por lo que encontrar soluciones a problemas o ayuda para el desarrollador no es una tarea difícil. Una clara evidencia de lo anterior es su repositorio oficial en GitHub el cual, al momento de ser escrito este documento cuenta con 46300 estrellas, 27760 commits, 1841 contribuidores, 20000 forks y más de 341000 proyectos usando este repositorio.

Gracias a su arquitectura de shared-nothing, Django se considera bastante escalable pues dicha arquitectura consiste en aislar la aplicación en distintas capas, por ejemplo, la base de datos y la aplicación en sí, y proporcionarles los recursos necesarios conforme las distintas capas lo vayan necesitando. Además, según la documentación oficial de Django, el framework posee un poderoso sistema de cache, que en aplicaciones grandes tales como washingtonpost.com o slashdot.org, miles de solicitudes podrían consumir todos los recursos del servidor en cuestión. Es por ello que el framework ahorra tiempo de procesado almacenando las páginas pre generadas en la caché.

```
given a URL, try finding that page in the cache
if the page is in the cache:
    return the cached page
else:
    generate the page
    save the generated page in the cache (for next time)
    return the generated page
```

Pseudo lógica del funcionamiento básico del cache en Django.

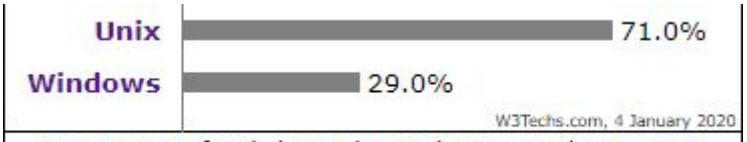
[Django's cache framework]

Al seguir las buenas practicas de programacion y filosofías sugeridas para este framework, el código resulta ser reutilizable y altamente mantenible. En general, las filosofías del diseño de Django son:

- Loose coupling: Por palabras de sus desarrolladores, esta es la principal meta de Django. Este principio se refiere a que las distintas capas del framework no deberían (a menos de que sea estrictamente necesario) "conocerse" las unas a las otras.
- DRY: Viene de la frase en inglés "Don't Repeat Yourself" que se traduce al espanol como no te repitas. Este principio predica que cada pieza o concepto de dato debe estar en solo y solo un solo lugar. La redundancia está mal y la normalización y generalización es mejor.
- Explícito es mejor que implícito: Este principio fue heredado del lenguaje en el cual Django se basa: Python y significa que Django no debería hacer mucha "magia". Es decir, abusar de la automatización de procesos a menos de que exista una buena razón para ello. De esta manera, se evita confundir a los programadores que están aprendiendo a utilizar cierta funcionalidad del framework.

Por mencionar los más importantes.

Gracias a que Python está implementado en los sistemas operativos más importantes y populares que se utilizan actualmente, Django puede ser utilizado en la mayoría de los sistemas y además, asegura ser portable entre las distintas plataformas. Segun la documentacion oficial de Python, este puede ser compilado en Unix, Linux, BSD, macOS, Cygwin y Windows, por lo que Django esta disponible en todas las plataformas previamente mencionadas. Según un reporte de W3Techs, un portal dedicado a hacer encuestas de tecnologías web, un 71% de los sitios web utilizan servidores Unix, mientras que el resto utiliza Windows.



Percentages of websites using various operating systems
Note: a website may use more than one operating system

Lo que quiere decir que Django puede ser utilizado en prácticamente cualquier sistema.

Django automatiza una gran variedad de procesos, lo que ayuda al desarrollador a enfocarse en la aplicación en sí y no a la configuracion desde cero de los sistemas. Es por ello que el desarrollo es rapido ademas de seguro, ya que en los procesos más cruciales se minimiza la configuración del programador y sus posibles errores. Por ejemplo, Django posee un poderoso creador de proyectos en sus funcionalidades que crean toda la configuración necesaria para el servidor y evitar las configuraciones que este podría implicar, sin embargo, es posible configurar todo el servidor una vez este haya sido creado. Además, Django posee herramientas de creador de sitio admin, que ahorra al programador tener que crear dicho portal y validaciones.

Arquitectura

Cuando se habla de la arquitectura de Django se deben abarcar dos puntos importantes que son la manera en la que Django guarda la información en la base de datos y el patrón de diseño bajo el cual se crean los proyectos en este framework. Estos dos puntos se detallan a continuación:

ORM(Object Relational Mapping)

Django utiliza la técnica de programación ORM (Object Relational Mapping), la cual consiste en relacionar los datos entre un lenguaje de programación orientado a objetos y un sistema de bases de datos. En otras palabras, un ORM permite transformar los datos contenidos en los objetos para guardarlos en la base de datos.

Patrón Model - Template - View (MTV)

El desarrollo en Django está basado en la arquitectura MVC (Model-View-Controller), sin embargo, los nombres de esta arquitectura varían en Django. En éste, al View se le conoce como Template y al Controller se le conoce como View, por lo que su arquitectura se denomina MTV(Model-Template-View).

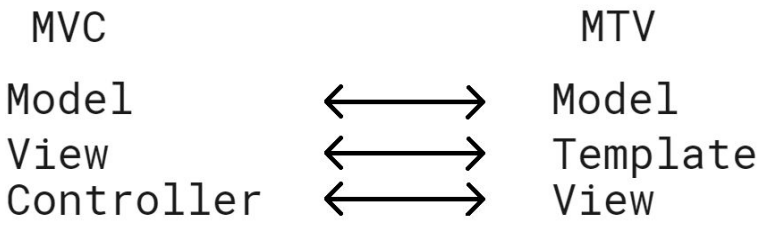


Figura 1: Nomenclatura MVC y MTV

El uso de esta arquitectura permite a los desarrolladores dividir la lógica de negocios, el acceso a los datos y lo que perciben los usuarios en una aplicación. Esto con el fin de

modularizar las partes de la aplicación y para que a la hora de realizar una modificación, no sea necesario cambiar todas las partes de la misma.

A continuación, se indica la función de cada una de las partes del patrón MTV:

- Model

  El Model consiste en la capa de acceso a datos, es decir, un objeto en el que se establece la estructura de los datos, la manera de acceder a los mismos, cómo validarlos y el comportamiento de los mismos, así como su relación con otros objetos y modelos.

- Template

  El Template consiste en la capa de presentación, la cual se encarga de indicar la manera en la que se le presentan al usuario los datos definidos en el Model. Por ejemplo, en esta capa se indica si los datos que se muestran son únicamente de lectura o si pueden ser editados por el usuario.

- View

  El View consiste en la capa de la lógica de negocios, la cual tiene la función de acceder al Model y proporcionar los datos al Template según la interacción que tenga el usuario con la aplicación. En otras palabras, el View sirve como intermediario entre el Model y el Template.

Ventajas y desventajas

Django es un entorno de trabajo de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. Este framework se hace cargo de gran parte de los obstáculos a la hora de desarrollar sitios web, cuenta con una amplia documentación y muchas opciones de soporte. Con Django puedes escribir **software completo**, provee casi todo lo que los desarrolladores esperaría tener a mano, sigue principios de diseño consistentes y cuenta con amplia y actualizada documentación. Django puede ser y ha sido utilizado para desarrollar casi cualquier tipo de sitio web, por lo que se dice que es **versátil**, se puede trabajar en conjunto con cualquier framework del lado del cliente y puede devolver contenido gran variedad de formatos. Se concibe como una apuesta de **seguro**, pues ayuda a los desarrolladores a evitar varios errores comunes de seguridad ya que Django es un framework diseñado para "hacer lo correcto", para proteger el sitio web de forma automática. Se considera que es muy **estable**, pues utiliza un componente que se basa en la arquitectura "shared-nothing" (cada componente es independiente de los otros). Se desarrolló utilizando principios y patrones de diseño para fomentar la creación de código **mantenible** y **reutilizable**. Debido a que está escrito en Python, no está sujeto a ninguna plataforma, es **portable**.

Además, por su arquitectura podemos mencionar algunas ventajas como:

- **Desarrollo rápido:** Al separarse en distintos componentes, se le facilita a los desarrolladores trabajar en diferentes aspectos de la misma aplicación, de manera simultánea.
- **Emparejamiento:** Esta arquitectura tiene componentes que se requieren uno al otro en ciertas partes de la aplicación, lo cual incrementa la seguridad del sitio web en general, ya que el archivo modelo sólo se almacenará en el servidor en lugar de almacenarse en el sitio web.

- **Facilidad de modificación:** Si sucede un cambio en componentes distintos, no es necesario cambiarlo en otros componentes. Esta es una ventaja especial de Django, ya que provee mucha más adaptabilidad del sitio web que en otros entornos de trabajo.

Del lado de las desventajas, aunque pocas, podemos encontrar que a pesar de su completa y excelente documentación, es muy extensa y tiende a ser **confusa**. Además, encontramos que a la hora de crear un API REST, Django suele ser más **difícil** de implementar que otros frameworks como Flask, aunque ciertamente esta afirmación está sujeta a la subjetividad.

Aplicaciones actuales que incorporan Django

Algunas de las aplicaciones implementan Django como parte de su WEB Stack, entre ellas podemos mencionar:

- **Instagram:** Según el equipo de ingenieros de Instagram, esta plataforma cuenta con la implementación más grande de Django en la actualidad. Además, mencionan que decidieron utilizar este framework porque se pueden desarrollar aplicaciones web de manera simple y rápida. Además, debido a la cantidad de usuarios que utiliza la plataforma, necesitaban una tecnología que fuera capaz de manejar servidores eficientes que respondiesen rápidamente a las necesidades de los usuarios.
- **Youtube:** En un inicio, esta popular plataforma utilizada para compartir videos estaba construída sobre PHP. Sin embargo, cuando la aplicación comenzó a ganar popularidad los ingenieros decidieron optar por Python y Django, debido a la necesidad de incrementar su rendimiento y realizar cambios de manera sencilla.

- **Dropbox:** Dropbox es una aplicación multiplataforma utilizada para el almacenamiento de imágenes, videos, documentos, entre otros. Dicha aplicación está construída con Python tanto del lado del servidor, como de la la aplicación de escritorio. Además,, utiliza Django para habilitar múltiples opciones para compartir, almacenar y sincronizar archivos.
- **BitBucket:** BitBucket es una aplicación web para el manejo de repositorios con versionamiento basados en Git. Esta es, según djangostars, una de las aplicaciones que usan Django con más carga, es decir, que más estresan al sistema. Esto debido a que se realizan más de 17 millones de requests y se cargan más de 6 millones de repositorios en un año. La razón por la cual Django es utilizado por BitBucket es por su comunidad de desarrolladores altamente activa además de la velocidad con la cual se puede desarrollar una solución con Django, ya que este cuenta con todo lo necesario para realizar la implementación.

También vale la pena mencionar algunas Webs como el periódico **The New York Times**, la página **NASA Science** y la aplicación **Pinterest**.

Referencias

Django (Version 3.0) [Computer Software]. (2020). Recuperado de https://djangoproject.com.

Holovary, H., Moss, J. (2013). *The Django Book*. Recuperado de https://buildmedia.readthedocs.org/media/pdf/djangobook/latest/djangobook.pdf

10 Popular Websites Built With Django. (2019). Retrieved 18 December 2019. Recuperado de https://djangostars.com/blog/10-popular-sites-made-on-django/

Django (web framework). (2019). Retrieved 18 December 2019. Recuperado de https://en.wikipedia.org/wiki/Django_(web_framework)#History

How we rolled out one of the largest Python 3 migrations ever. (2019). Retrieved 18 December 2019. Recuperado de https://blogs.dropbox.com/tech/2018/09/how-we-rolled-out-one-of-the-largest-python-3-migrations-ever/

Introducción a Django. (2019). Retrieved 18 December 2019. Recuperado de https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducci%C3%B3n

Team, D. (2019). Django Architecture - 3 Major Components of MVC Pattern - DataFlair. Retrieved 18 December 2019. Recuperado de https://data-flair.training/blogs/django-architecture/

The Web framework for perfectionists with deadlines | Django. (2019). Retrieved 18 December 2019. Recuperado de https://www.djangoproject.com

Top 10 Django Apps and Why Companies Are Betting on This Framework. (2019). Retrieved 18 December 2019. Recuperado de https://www.netguru.com/blog/top-10-django-apps-and-why-companies-are-betting-on-this-framework

Web Service Efficiency at Instagram with Python. (2019). Retrieved 18 December 2019. Recuperado de https://instagram-engineering.com/web-service-efficiency-at-instagram-with-python-4976d078e366