**Notas de Django**

**¿Qué es Django?**

**Django** es un **framework** de desarrollo web de **código abierto**, escrito en **Python**, que respeta el patrón de diseño conocido como **Modelo–vista–template**. Fue desarrollado en origen para gestionar varias páginas orientadas a noticias de la World Company de Lawrence, Kansas, y fue liberada al público bajo una **licencia BSD** en julio de 2005; el framework fue nombrado en alusión al guitarrista de jazz gitano Django Reinhardt. En junio de 2008 fue anunciado que la recién formada Django Software Foundation se haría cargo de Django en el futuro.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio **No te repitas** (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

**Arquitectura**

Aunque Django está fuertemente inspirado en la filosofía de desarrollo **Modelo Vista Controlador**, sus desarrolladores declaran públicamente que no se sienten especialmente atados a observar estrictamente ningún paradigma particular, y en cambio prefieren hacer "lo que les parece correcto". Como resultado, por ejemplo, lo que se llamaría "controlador" en un "verdadero" framework MVC se llama en Django "vista", y lo que se llamaría "vista" se llama "plantilla".

Gracias al poder de las capas mediator y foundation, Django permite que los desarrolladores se dediquen a construir los objetos Entity y la lógica de presentación y control para ellos.

**Presentación:**

Aquí se maneja la interacción entre el usuario y el computador. En Django, esta tarea la realizan el motor de plantillas y el cargador de plantillas que toman la información y la presentan al usuario (vía HTML, por ejemplo). El sistema de configuración de URLs es también parte de la capa de presentación.

**Control:**

En esta capa reside el programa o la lógica de aplicación en sí. En Django son representados por las vistas y los manipuladores. La capa de presentación depende de ésta y a su vez ésta depende de la capa de dominio.

**Mediator:**

Es el encargado de manejar la interacción entre el subsistema Entity y foundation. Aquí se realiza el mapeo objeto-relacional a cargo del motor de Django.

**Entity:**

El subsistema entity maneja los objetos de negocio. El mapeo objeto-relacional de Django permite escribir objetos de tipo entity de una forma fácil y estándar.

**Foundation:**

La principal tarea del subsistema foundation es la de manejar a bajo nivel el trabajo con la base de datos. Se provee soporte a nivel de foundation para varias bases de datos y otras están en etapa de prueba.

**Soporte de bases de datos**

Respecto a la base de datos, la recomendada es **PostgreSQL**, pero también son soportadas **MySQL** y **SQLite 3**. Se encuentra en desarrollo un adaptador para Microsoft SQL Server. Una vez creados los modelos de datos, Django proporciona una abstracción de la base de datos a través de su API que permite crear, recuperar, actualizar y borrar objetos. También es posible que el usuario ejecute sus propias consultas SQL directamente. En el modelo de datos de Django, una clase representa un registro de una tabla en la base de datos y las instancias de ésta serán las tuplas en la tabla.

**Soporte de servidores Web**

Como mencionamos en los requisitos, Django incluye un servidor web liviano para realizar pruebas y trabajar en la etapa de desarrollo. En la etapa de producción, sin embargo, se recomienda **Apache 2** con **mod\_python**. Aunque Django soporta la especificación **WSGI**, por lo que puede correr sobre una gran variedad de servidores como **FastCGI** o SCGI en Apache u otros servidores (particularmente **Lighttpd**).

***Fuente:***[*https://es.wikipedia.org/wiki/Django\_(framework)*](https://es.wikipedia.org/wiki/Django_(framework))

*Septiembre de 2018*

**Documentación de Django**

Todo lo que necesita saber sobre Django.

**Como está organizada la documentación**

Django tiene mucha documentación. Una descripción general de alto nivel de cómo se organiza le ayudará a saber dónde buscar ciertas cosas:

* [Tutoriales](https://docs.djangoproject.com/es/2.0/intro/tutorial01/) lo llevan de la mano a través de una serie de pasos para crear una aplicación Web. Comience aquí si es principiante en Django o en el desarrollo de aplicaciones Web. Eche un vistazo también al «[Los primeros pasos](https://docs.djangoproject.com/es/2.0/#index-first-steps)» a continuación.
* [Guías temáticas](https://docs.djangoproject.com/es/2.0/topics/) discuten los temas y conceptos fundamentales en un nivel bastante alto y proporcionan información y explicación de antecedentes valiosa.
* [Guías de referencia](https://docs.djangoproject.com/es/2.0/ref/) contienen la referencia técnica para las APIs y otros aspectos del sistema de Django. Ellas describen cómo funciona y cómo utilizarla, pero da por sentado que usted tiene un conocimiento básico de los conceptos claves.
* [Las guías de uso](https://docs.djangoproject.com/es/2.0/howto/) son fórmulas. Lo guian a través de los pasos involucrados en abordar los problemas principales y los casos de uso. Son más avanzadas que los tutoriales y dan por sentado el tener algo de conocimiento sobre cómo funciona Django.

**Resumen:** Para iniciar, leer primero los tutoriales, después las guías de uso, tener en cuenta las guías temáticas y las guías de referencia para resolver dudas muy puntuales.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Los primeros pasos**

¿Es usted nuevo en Django o con la programación? ¡Este es el lugar para comenzar!

* **Desde cero:** [Visión general](https://docs.djangoproject.com/es/2.0/intro/overview/) | [Instalación](https://docs.djangoproject.com/es/2.0/intro/install/)
* **Tutorial:** [Parte 1: Peticiones y respuestas](https://docs.djangoproject.com/es/2.0/intro/tutorial01/) | [Parte 2: Modelos y el sitio administrativo](https://docs.djangoproject.com/es/2.0/intro/tutorial02/) | Parte 3: Vistas y plantillas | [Parte 5: Pruebas](https://docs.djangoproject.com/es/2.0/intro/tutorial05/) | [Parte 6: Archivos estáticos](https://docs.djangoproject.com/es/2.0/intro/tutorial06/) | [Parte 7: Personalizando el sitio administrativo](https://docs.djangoproject.com/es/2.0/intro/tutorial07/)
* **Tutoriales avanzados:** [Cómo escribir aplicaciones reutilizables](https://docs.djangoproject.com/es/2.0/intro/reusable-apps/) | [Creando tu primer parche para Django](https://docs.djangoproject.com/es/2.0/intro/contributing/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**La capa del modelo**

Django proporciona una capa de abstracción (los «modelos») para estructurar y manipular los datos de su aplicación Web. Aprenda más sobre esta a continuación:

* **Modelos:** [Introducción a los modelos](https://docs.djangoproject.com/es/2.0/topics/db/models/) | [Tipos de campos](https://docs.djangoproject.com/es/2.0/topics/db/models/) | [Índices](https://docs.djangoproject.com/es/2.0/ref/models/indexes/) | [Opciones Meta](https://docs.djangoproject.com/es/2.0/ref/models/options/) | [Clase Model](https://docs.djangoproject.com/es/2.0/ref/models/class/)
* **Querysets:** [Haciendo consultas](https://docs.djangoproject.com/es/2.0/topics/db/queries/) | [Referencia de métodos de QuerySet](https://docs.djangoproject.com/es/2.0/ref/models/querysets/) | [Expresiones de búsqueda](https://docs.djangoproject.com/es/2.0/ref/models/lookups/)
* **Instancias de modelos:** [Métodos de las instancias](https://docs.djangoproject.com/es/2.0/ref/models/instances/) | [Accediendo a objetos relacionados](https://docs.djangoproject.com/es/2.0/ref/models/relations/)
* **Migraciones:** [Introducción a las migraciones](https://docs.djangoproject.com/es/2.0/topics/migrations/) | [Referencia de operaciones](https://docs.djangoproject.com/es/2.0/ref/migration-operations/) | [SchemaEditor](https://docs.djangoproject.com/es/2.0/ref/schema-editor/) | [Escribiendo migraciones](https://docs.djangoproject.com/es/2.0/howto/writing-migrations/)
* **Avanzado** [Manejadores](https://docs.djangoproject.com/es/2.0/topics/db/managers/) | [Raw SQL](https://docs.djangoproject.com/es/2.0/topics/db/sql/) | [Transacciones](https://docs.djangoproject.com/es/2.0/topics/db/transactions/) | [Agregación](https://docs.djangoproject.com/es/2.0/topics/db/aggregation/) | [Búsqueda](https://docs.djangoproject.com/es/2.0/topics/db/search/) | [Campos personalizados](https://docs.djangoproject.com/es/2.0/howto/custom-model-fields/) | [Múltiples bases de datos](https://docs.djangoproject.com/es/2.0/topics/db/multi-db/) | [Búsquedas personalizadas](https://docs.djangoproject.com/es/2.0/howto/custom-lookups/) | [Expresiones de consulta](https://docs.djangoproject.com/es/2.0/ref/models/expressions/) | [Expresiones condicionales](https://docs.djangoproject.com/es/2.0/ref/models/conditional-expressions/) | [Funciones de Base de Datos](https://docs.djangoproject.com/es/2.0/ref/models/database-functions/)
* **Otros:** [Bases de datos soportadas](https://docs.djangoproject.com/es/2.0/ref/databases/) | [Bases de datos heredadas](https://docs.djangoproject.com/es/2.0/howto/legacy-databases/) | [Proporcionando los datos iniciales](https://docs.djangoproject.com/es/2.0/howto/initial-data/) | [Optimizar el acceso a la base de datos](https://docs.djangoproject.com/es/2.0/topics/db/optimization/) | [Características específicas de PostgreSQL](https://docs.djangoproject.com/es/2.0/ref/contrib/postgres/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**La capa vista**

Django tiene el concepto de «vistas» para encapsular la lógica responsable de procesar una petición del usuario y devolver la respuesta. Encuentre todo lo que necesita saber sobre las vistas a través de los enlaces que siguen:

* **Lo básico:** [URLconfs](https://docs.djangoproject.com/es/2.0/topics/http/urls/) | [Funciones de vista](https://docs.djangoproject.com/es/2.0/topics/http/views/) | [Atajos](https://docs.djangoproject.com/es/2.0/topics/http/shortcuts/) | [Decoradores](https://docs.djangoproject.com/es/2.0/topics/http/decorators/)
* **Referencia:** [Vistas incorporadas en Django](https://docs.djangoproject.com/es/2.0/ref/views/) | [Objetos petición/respuesta](https://docs.djangoproject.com/es/2.0/ref/request-response/) | [Objetos TemplateResponse](https://docs.djangoproject.com/es/2.0/ref/template-response/)
* **Carga de archivos:** [Visión general](https://docs.djangoproject.com/es/2.0/topics/http/file-uploads/) | [Objetos de archivo](https://docs.djangoproject.com/es/2.0/ref/files/file/) | [API de almacenamiento](https://docs.djangoproject.com/es/2.0/ref/files/storage/) | [Gestión de archivos](https://docs.djangoproject.com/es/2.0/topics/files/) | [Almacenamiento personalizado](https://docs.djangoproject.com/es/2.0/howto/custom-file-storage/)
* **Vistas basadas en clases:** [Visión general](https://docs.djangoproject.com/es/2.0/topics/class-based-views/) | [Vistas incorporadas para visualización](https://docs.djangoproject.com/es/2.0/topics/class-based-views/generic-display/) | [Vistas incorporadas para edición](https://docs.djangoproject.com/es/2.0/topics/class-based-views/generic-editing/) | [Usando Mixins](https://docs.djangoproject.com/es/2.0/topics/class-based-views/mixins/) | [Referencia de la API](https://docs.djangoproject.com/es/2.0/ref/class-based-views/) | [Índice estático](https://docs.djangoproject.com/es/2.0/ref/class-based-views/flattened-index/)
* **Avanzado:** [Generación de archivos CSV](https://docs.djangoproject.com/es/2.0/howto/outputting-csv/) | [Generación de archivos PDF](https://docs.djangoproject.com/es/2.0/howto/outputting-pdf/)
* **Middleware:** [Visión general](https://docs.djangoproject.com/es/2.0/topics/http/middleware/) | [Clases de Middleware incorporadas](https://docs.djangoproject.com/es/2.0/ref/middleware/)

**Resumen:** Hacer énfasis en carga de archivos y en Avanzado, se va a necesitar posteriormente.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**La capa de plantillas**

La capa de plantillas provee una sintaxis amigable para los diseñadores que permite mostrar la información que se presentará al usuario. Aprenda cómo esta sintaxis puede ser usada por los diseñadores y extendida por los programadores:

* **Lo básico:** [Visión general](https://docs.djangoproject.com/es/2.0/topics/templates/)
* **Para los diseñadores:** [Información general del lenguaje](https://docs.djangoproject.com/es/2.0/ref/templates/language/) | [Etiquetas y filtros incorporados](https://docs.djangoproject.com/es/2.0/ref/templates/builtins/) | [Humanización](https://docs.djangoproject.com/es/2.0/ref/contrib/humanize/)
* **Para los programadores:** [API de las plantillas](https://docs.djangoproject.com/es/2.0/ref/templates/api/) | [Etiquetas y filtros personalizados](https://docs.djangoproject.com/es/2.0/howto/custom-template-tags/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Formularios**

Django provee un poderoso framework para facilitar la creación de formularios y la manipulación de sus datos.

* **Lo básico:** [Visión general](https://docs.djangoproject.com/es/2.0/topics/forms/) | [API de formularios](https://docs.djangoproject.com/es/2.0/ref/forms/api/) | [Campos incorporados](https://docs.djangoproject.com/es/2.0/ref/forms/fields/) | [Widgets incorporados](https://docs.djangoproject.com/es/2.0/ref/forms/widgets/)
* **Avanzado:** [Formularios para modelos](https://docs.djangoproject.com/es/2.0/topics/forms/modelforms/) | [Integración de medios](https://docs.djangoproject.com/es/2.0/topics/forms/media/) | [Juegos de formularios](https://docs.djangoproject.com/es/2.0/topics/forms/formsets/) | [Personalización de la validación](https://docs.djangoproject.com/es/2.0/ref/forms/validation/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**El proceso de desarrollo**

Aprenda sobre los diversos componentes y herramientas para ayudarse en el desarrollo y pruebas de aplicaciones Django:

* **Ajustes:** [Visión general](https://docs.djangoproject.com/es/2.0/topics/settings/) | [Lista completa de ajustes](https://docs.djangoproject.com/es/2.0/ref/settings/)
* **Aplicaciones:** [Información general](https://docs.djangoproject.com/es/2.0/ref/applications/)
* **Excepciones:** [Visión general](https://docs.djangoproject.com/es/2.0/ref/exceptions/)
* **django-admin y manage.py:** [Información general](https://docs.djangoproject.com/es/2.0/ref/django-admin/) | [Añadiendo comandos personalizados](https://docs.djangoproject.com/es/2.0/howto/custom-management-commands/)
* **Pruebas:** [Introducción](https://docs.djangoproject.com/es/2.0/topics/testing/) | [Confección y ejecución de pruebas](https://docs.djangoproject.com/es/2.0/topics/testing/) | [Herramientas de pruebas incluidas](https://docs.djangoproject.com/es/2.0/topics/testing/tools/) | [Temas avanzados](https://docs.djangoproject.com/es/2.0/topics/testing/advanced/)
* **Implementación:** [Información general](https://docs.djangoproject.com/es/2.0/howto/deployment/) | [Servidores WSGI](https://docs.djangoproject.com/es/2.0/howto/deployment/wsgi/) | [Implementando archivos estáticos](https://docs.djangoproject.com/es/2.0/howto/static-files/deployment/) | [Rastreando errores de código por correo electrónico](https://docs.djangoproject.com/es/2.0/howto/error-reporting/)

**Resumen:** Hacer énfasis en este tema, ya que es de vital importancia para una buena ingeniería de software y un buen desarrollo. (Consultar si hay forma de aplicar Scrum o algún marco de trabajo de softare).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**El sitio administrativo**

Encuentre todo lo que necesita saber acerca de la interfaz administrativa automatizada, una de las funcionalidades más populares de Django:

* [Sitio administrativo](https://docs.djangoproject.com/es/2.0/ref/contrib/admin/)
* [Acciones del sitio administrativo](https://docs.djangoproject.com/es/2.0/ref/contrib/admin/actions/)
* [Generador de documentación del sitio administrativo](https://docs.djangoproject.com/es/2.0/ref/contrib/admin/admindocs/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Seguridad

La seguridad es un tema de suma importancia en el desarrollo de aplicaciones Web y Django provee múltiples herramientas y mecanismos de protección:

* [Información general de la seguridad](https://docs.djangoproject.com/es/2.0/topics/security/)
* [Problemas de seguridad en Django revelados](https://docs.djangoproject.com/es/2.0/releases/security/)
* [Protección contra Clickjacking](https://docs.djangoproject.com/es/2.0/ref/clickjacking/)
* [Protección de Falsificación de Petición entre Sitios](https://docs.djangoproject.com/es/2.0/ref/csrf/)
* [Firma criptográfica](https://docs.djangoproject.com/es/2.0/topics/signing/)
* [Middleware de seguridad](https://docs.djangoproject.com/es/2.0/ref/middleware/#security-middleware)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Internacionalización y localización**

Django ofrece un robusto framework de internacionalización y localización para ayudarle en el desarrollo de aplicaciones para múltiples idiomas y regiones del mundo:

* [Información general](https://docs.djangoproject.com/es/2.0/topics/i18n/) | [Internacionalización](https://docs.djangoproject.com/es/2.0/topics/i18n/translation/) | [Localización](https://docs.djangoproject.com/es/2.0/topics/i18n/translation/#how-to-create-language-files) | [formato localizado de la interfaz Web y entrada de formulario](https://docs.djangoproject.com/es/2.0/topics/i18n/formatting/)
* [Husos horarios](https://docs.djangoproject.com/es/2.0/topics/i18n/timezones/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Rendimiento y optimización**

Existe una variedad de técnicas y herramientas que lo pueden ayudar a ejecutar el código de forma más eficiente y rápida y con menos uso de recursos del sistema.

* [Reseña sobre rendimiento y optimización](https://docs.djangoproject.com/es/2.0/topics/performance/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Framework geográfico**

[GeoDjango](https://docs.djangoproject.com/es/2.0/ref/contrib/gis/) tiene como propósito ser un framework para el desarrollo geográfico de clase mundial. Su meta es hacer lo más fácil posible construir aplicaciones GIS Web y aprovechar el poder de los datos habilitados de forma espacial.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Herramientas comunes para aplicaciones Web**

Django ofrece múltiples herramientas que son necesarias para el desarrollo de aplicaciones Web:

* **Autenticación:** [Información general](https://docs.djangoproject.com/es/2.0/topics/auth/) | [Usando el sistema de autenticación](https://docs.djangoproject.com/es/2.0/topics/auth/default/) | [Gestión de contraseñas](https://docs.djangoproject.com/es/2.0/topics/auth/passwords/) | [Personalizando la autenticación](https://docs.djangoproject.com/es/2.0/topics/auth/customizing/) | [Referencia de la API](https://docs.djangoproject.com/es/2.0/ref/contrib/auth/)
* [Almacenamiento en caché](https://docs.djangoproject.com/es/2.0/topics/cache/)
* [Registro](https://docs.djangoproject.com/es/2.0/topics/logging/)
* [Enviando correos](https://docs.djangoproject.com/es/2.0/topics/email/)
* [Fuentes de redifusión (RSS/Atom)](https://docs.djangoproject.com/es/2.0/ref/contrib/syndication/)
* [Paginado](https://docs.djangoproject.com/es/2.0/topics/pagination/)
* [Framework de mensajes](https://docs.djangoproject.com/es/2.0/ref/contrib/messages/)
* [Serialización](https://docs.djangoproject.com/es/2.0/topics/serialization/)
* [Sesiones](https://docs.djangoproject.com/es/2.0/topics/http/sessions/)
* [Mapas de sitio](https://docs.djangoproject.com/es/2.0/ref/contrib/sitemaps/)
* [Gestión de archivos estáticos](https://docs.djangoproject.com/es/2.0/ref/contrib/staticfiles/)
* [Validación de datos](https://docs.djangoproject.com/es/2.0/ref/validators/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Otras funcionalidades básicas**

Aprenda sobre otras funcionalidades básicas del framework Django:

* [Procesamiento condicional de contenido](https://docs.djangoproject.com/es/2.0/topics/conditional-view-processing/)
* [Tipos de contenido y relaciones genéricas](https://docs.djangoproject.com/es/2.0/ref/contrib/contenttypes/)
* [Páginas estáticas](https://docs.djangoproject.com/es/2.0/ref/contrib/flatpages/)
* [Redirecciones](https://docs.djangoproject.com/es/2.0/ref/contrib/redirects/)
* [Señales](https://docs.djangoproject.com/es/2.0/topics/signals/)
* [Framework de comprobación de sistema](https://docs.djangoproject.com/es/2.0/topics/checks/)
* [El framework de sitios](https://docs.djangoproject.com/es/2.0/ref/contrib/sites/)
* [Unicode en Django](https://docs.djangoproject.com/es/2.0/ref/unicode/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**El proyecto de código abierto Django**

Aprenda sobre el proceso de desarrollo para el proyecto Django en sí y sobre cómo puede contribuir:

* **Comunidad:** [Cómo involucrarse](https://docs.djangoproject.com/es/2.0/internals/contributing/) | [El proceso de liberación](https://docs.djangoproject.com/es/2.0/internals/release-process/) | [Organización del equipo](https://docs.djangoproject.com/es/2.0/internals/organization/) | [El repositorio de código fuente de Django](https://docs.djangoproject.com/es/2.0/internals/git/) | [Políticas de seguridad](https://docs.djangoproject.com/es/2.0/internals/security/) | [Listas de correo](https://docs.djangoproject.com/es/2.0/internals/mailing-lists/)
* **Filosofías de diseño:** [Información general](https://docs.djangoproject.com/es/2.0/misc/design-philosophies/)
* **Documentación:** [Sobre esta documentación](https://docs.djangoproject.com/es/2.0/internals/contributing/writing-documentation/)
* **Distribuciones de terceros:** [Visión general](https://docs.djangoproject.com/es/2.0/misc/distributions/)
* **Django en el tiempo:** [Estabilidad de la API](https://docs.djangoproject.com/es/2.0/misc/api-stability/) | [Notas de versión e instrucciones de actualización](https://docs.djangoproject.com/es/2.0/releases/) | [Línea de tiempo de funcionalidades en desuso](https://docs.djangoproject.com/es/2.0/internals/deprecation/)

***Fuente:***[*https://docs.djangoproject.com/es/2.0/*](https://docs.djangoproject.com/es/2.0/)

*Septiembre de 2018*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Django de un vistazo**

Debido a que Django fue desarrollado en un ambiente de sala de redacción de ritmo rápido, fue diseñado para hacer tareas comunes de desarrollo web de forma rápida y fácil. Aquí se incluye una introducción informal de cómo escribir una aplicación web que utiliza una base de datos con Django.

El objetivo de este documento es darle suficiente información técnica para entender cómo funciona Django, pero no pretende ser un tutorial o referencia - ¡Aunque tenemos ambas cosas! Cuando esté listo para empezar un proyecto con Django, puede [comenzar con el tutorial](https://docs.djangoproject.com/es/2.0/intro/tutorial01/) o: doc:doc:ir directamente a la documentación más detallada </topics/index>.

**Diseñe su modelo**

Aunque es posible usar Django sin una base de datos, Django cuenta con un [mapeador objeto-relacional](https://en.wikipedia.org/wiki/Object-relational_mapping) en el que es posible definir la estructura de la base de datos utilizando código Python.

La [sintaxis de modelo de datos](https://docs.djangoproject.com/es/2.0/topics/db/models/) ofrece muchas formas de representar sus modelos – hasta ahora, ha resuelto una cantidad equivalente a años de problemas de esquemas de base de datos. Aquí hay un ejemplo rápido:

**mysite/news/models.py**

**from** **django.db** **import** models

**class** **Reporter**(models.Model)**:**

full\_name = models.CharField(max\_length=70)

**def** \_\_str\_\_(self)**:**

**return** self.full\_name

**class** **Article**(models.Model):

pub\_date = models.DateField()

headline = models.CharField(max\_length=200)

content = models.TextField()

reporter = models.ForeignKey(Reporter, on\_delete=models.CASCADE)

**def** \_\_str\_\_(self)**:**

**return** self.headline

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Instálelo**

A continuación ejecute la utilidad de la línea de comandos de Django para crear las tablas de base de datos de forma automática:

**$** python manage.py migrate

El comando **migrate** analiza todos sus modelos disponibles y crea cualquier tabla faltante en su base de datos, así como ofrece opcionalmente un [control de esquema mucho más rico](https://docs.djangoproject.com/es/2.0/topics/migrations/).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Disfrute de la API sin restricciones**

Con esto, usted tiene una [API Python](https://docs.djangoproject.com/es/2.0/topics/db/queries/) rica y sin restricciones para acceder a sus datos. La API se crea sobre la marcha y no es necesario generar ningún código:

# Import the models we created from our "news" app

>>> **from** **news.models** **import** Article, Reporter

# No reporters are in the system yet.

>>> Reporter.objects.all()

<QuerySet []>

# Create a new Reporter.

>>> r = Reporter(full\_name='John Smith')

# Save the object into the database. You have to call save() explicitly.

>>> r.save()

# Now it has an ID.

>>> r.id

1

# Now the new reporter is in the database.

>>> Reporter.objects.all()

<QuerySet [<Reporter: John Smith>]>

# Fields are represented as attributes on the Python object.

>>> r.full\_name

'John Smith'

# Django provides a rich database lookup API.

>>> Reporter.objects.get(id=1)

<Reporter: John Smith>

>>> Reporter.objects.get(full\_name\_\_startswith='John')

<Reporter: John Smith>

>>> Reporter.objects.get(full\_name\_\_contains='mith')

<Reporter: John Smith>

>>> Reporter.objects.get(id=2)

Traceback (most recent call last):

...

DoesNotExist: Reporter matching query does **not** exist.

# Create an article.

>>> **from** **datetime** **import** date

>>> a = Article(pub\_date=date.today(), headline='Django is cool',

... content='Yeah.', reporter=r)

>>> a.save()

# Now the article is in the database.

>>> Article.objects.all()

<QuerySet [<Article: Django **is** cool>]>

# Article objects get API access to related Reporter objects.

>>> r = a.reporter

>>> r.full\_name

'John Smith'

# And vice versa: Reporter objects get API access to Article objects.

>>> r.article\_set.all()

<QuerySet [<Article: Django **is** cool>]>

# The API follows relationships as far as you need, performing efficient

# JOINs for you behind the scenes.

# This finds all articles by a reporter whose name starts with "John".

>>> Article.objects.filter(reporter\_\_full\_name\_\_startswith='John')

<QuerySet [<Article: Django **is** cool>]>

# Change an object by altering its attributes and calling save().

>>> r.full\_name = 'Billy Goat'

>>> r.save()

# Delete an object with delete().

>>> r.delete()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Una interfaz administrativa dinámica: no es sólo andamios; es toda una casa**

Una vez definidos sus modelos, Django puede crear automáticamente una [interfaz administrativa](https://docs.djangoproject.com/es/2.0/ref/contrib/admin/) profesional y lista para la producción; un sitio Web que le permita a los usuarios autenticados añadir, modificar y eliminar objetos. Es tan fácil como registrar su modelo en el sitio administrativo:

**mysite/news/models.py**

**from** **django.db** **import** models

**class** **Article**(models.Model)**:**

pub\_date = models.DateField()

headline = models.CharField(max\_length=200)

content = models.TextField()

reporter = models.ForeignKey(Reporter, on\_delete=models.CASCADE)

**mysite/news/admin.py**

**from** **django.contrib** **import** admin

**from** **.** **import** models

admin.site.register(models.Article)

La filosofía aquí es que su sitio es editado por un equipo, un cliente o tal vez sólo por usted y usted no quiere tener que lidiar con la creación de interfaces de backends, sólo quiere administrar el contenido.

Un flujo de trabajo típico en la creación de aplicaciones Django es crear modelos y poner a funcionar los sitios administrativos lo más rápido posible para que su personal (o clientes) puedan empezar a ingresar los datos y luego, perfeccionar la forma como los datos se presentan al público.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Diseñe sus URLs**

Un esquema de URL limpio y elegante es un detalle importante en una aplicación web de alta calidad. Django fomenta un diseño de URL hermoso y no añade ningún elemento innecesario en las URLs, como **.php** o **.asp**.

Para diseñar las URLs de una aplicación, usted debe crear un módulo de Python llamado URLconf. Esta es una tabla de contenidos para su aplicación que contiene un mapeo sencillo entre patrones URLs y funciones callback de Python. Las tablas URLconf también sirven para separar las URLs del código Python.

Así es como se vería una URLconf para el ejemplo **Reporter**/**Article** anterior:

**mysite/news/urls.py**

**from** **django.urls** **import** path

**from** **.** **import** views

urlpatterns = [

path('articles/<int:year>/', views.year\_archive),

path('articles/<int:year>/<int:month>/', views.month\_archive),

path('articles/<int:year>/<int:month>/<int:pk>/', views.article\_detail),

]

El código anterior mapea las URLs a funciones callback de Python («vistas»). Las cadena de caracteres de la ruta usa etiquetas para «capturar» los valores de las URLs. Cuando un usuario solicita una página Django pasa por cada patrón en orden y se detiene en el primero que coincide con la URL solicitada. (Si ninguno de ellos coincide Django llama una vista del caso especial 404). Esto es increíblemente rápido debido a que las rutas son compiladas como expresiones regulares en tiempo de carga.

Una vez que una de las expresiones regulares coincide, Django llama a la vista determinada, que es una función Python. Cada vista recibe un objeto de consulta – que contiene metadatos de consulta – y los valores capturados en la expresión regular.

Por ejemplo si un usuario solicitara la URL «/articles/2005/05/39323/» Django llamaría la función **news.views.article\_detail(request, year=2005, month=05, pk=39323)**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Escriba sus vistas**

Cada vista es responsable de hacer una de dos cosas: retornar un objeto HttpResponse que incluye el contenido de la página solicitada, o levantar una excepción como Http404. El resto depende de ti.

Generalmente, una vista recupera los datos de acuerdo con los parámetros, carga una plantilla y crea la plantilla con los datos recuperados. Aquí está una vista de ejemplo para el year\_archive de arriba:

**mysite/news/views.py**

**from** **django.**shortcuts **import** render

**from** **.models** **import** Article

**def** year\_archive(request, year):

a\_list = Article.objects.filter(pub\_date\_\_year=year)

context = {'year': year, 'article\_list': a\_list}

**return** render(request, 'news/year\_archive.html', context)

Este ejemplo utiliza el [sistema de plantillas de Django](https://docs.djangoproject.com/es/2.0/topics/templates/) que tiene varias características potentes, pero se esfuerza por mantenerse lo suficientemente simple como para que los que no son programadores lo puedan utilizar.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Diseñe sus plantillas**

El código anterior carga la plantilla **news/year\_archive.html**.

Django tiene una ruta de búsqueda de plantillas lo que le permite a usted minimizar la redundancia entre las plantillas. En su configuración de Django, usted especifica una lista de directorios para verificar si hay plantillas con **DIRS**. Si una plantilla no existe en el primer directorio, verifica en el segundo y así sucesivamente.

Digamos que se encontró la plantilla news/year\_archive.html`. Así es como podría verse:

**mysite/news/templates/news/year\_archive.html**

{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}

<h1>Articles for {{ year }}</h1>

{% for article in article\_list %}

<p>{{ article.headline }}</p>

<p>By {{ article.reporter.full\_name }}</p>

<p>Published {{ article.pub\_date|date:"F j, Y" }}</p>

{% endfor %}

{% endblock %}

Las variables están rodeadas por llaves dobles. **{{ article.headline }}** significa «Enviar el valor del atributo headline del artículo». Sin embargo, los puntos no sólo se utilizan para las operaciones de búsqueda de atributo. También pueden hacer búsqueda de pares diccionario-clave, búsqueda de índice y llamadas a funciones.

Tenga en cuenta que **{{ article.pub\_date|date:"F j, Y" }}** utiliza una «tubería» al estilo Unix (el carácter «|»). Esto se conoce como filtro de plantilla, y es una manera de filtrar el valor de una variable. En este caso, el filtro de fecha formatea un objeto datetime de Python en el formato determinado (como se encuentra en la función de fecha de PHP).

Usted puede encadenar tantos filtros como desee. Usted puede escribir [filtros de plantillas personalizados](https://docs.djangoproject.com/es/2.0/howto/custom-template-tags/#howto-writing-custom-template-filters). Usted puede escribir [etiquetas de plantillas personalizadas](https://docs.djangoproject.com/es/2.0/howto/custom-template-tags/) que ejecutan el código Python personalizado tras bastidores.

Por último, Django usa el concepto de «herencia de plantillas». Eso es lo que hace el **{% extends "base.html"%}**. Esto quiere decir «Primero cargue la plantilla llamada “base”, que ha definido un grupo de bloques, y llene los bloques con los siguientes bloques.» En resumen, eso le permite reducir drásticamente la redundancia en las plantillas: cada plantilla tiene que definir sólo lo que es exclusivo de esa plantilla.

Aquí esta la plantilla «base.html», incluyendo el uso de los [archivos estáticos](https://docs.djangoproject.com/es/2.0/howto/static-files/) de como podría verse:

**mysite/templates/base.html**

{% load static %}

<html>

<head>

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

</head>

<body>

<img src="{% static "images/sitelogo.png" %}" alt="Logo" />

{% block content %}{% endblock %}

</body>

</html>

De forma simplista, define el aspecto del sitio (con el logo del sitio) y proporciona «agujeros» a ser llenados por las plantillas hijas. Esto hace un rediseño del sitio tan fácil como cambiar un solo archivo: la plantilla base.

También le permite crear múltiples versiones de un sitio, con diferentes plantillas bases, a la vez que reutiliza las plantillas hijas. Los creadores de Django han usado esta técnica para crear versiones móviles de sitios sorprendentemente distintas, simplemente creando una nueva plantilla base.

Tenga en cuenta que usted no tiene que usar el sistema de plantillas de Django si prefiere otro sistema. Si bien el sistema de plantillas de Django está particularmente bien integrado con la capa modelo de Django, nada lo obliga a usarlo. No tiene tampoco que usar la API de base de datos de Django. Puede usar otra capa de abstracción de base de datos, leer archivos XML, leer archivos desde el disco o lo que desee. Cada parte de Django – modelos, vistas, plantillas, está separada de la otra.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Esto es solo la superficie**

Esta ha sido solamente una rápida introducción a la funcionalidad de Django. Otras características útiles:

* Un [framework de caché](https://docs.djangoproject.com/es/2.0/topics/cache/) que se integra con memcached u otros backends.
* Un [framework de sindicación](https://docs.djangoproject.com/es/2.0/ref/contrib/syndication/) que hace que la creación de feeds RSS y Atom sea tan fácil como escribir una pequeña clase de Python.
* Funciones del sitio administrativo más atractivas generadas automáticamente – esta descripción general apenas arañó la superficie.

*Fuente:* [*https://docs.djangoproject.com/es/2.0/intro/overview/*](https://docs.djangoproject.com/es/2.0/intro/overview/)

*Septiembre de 2018*