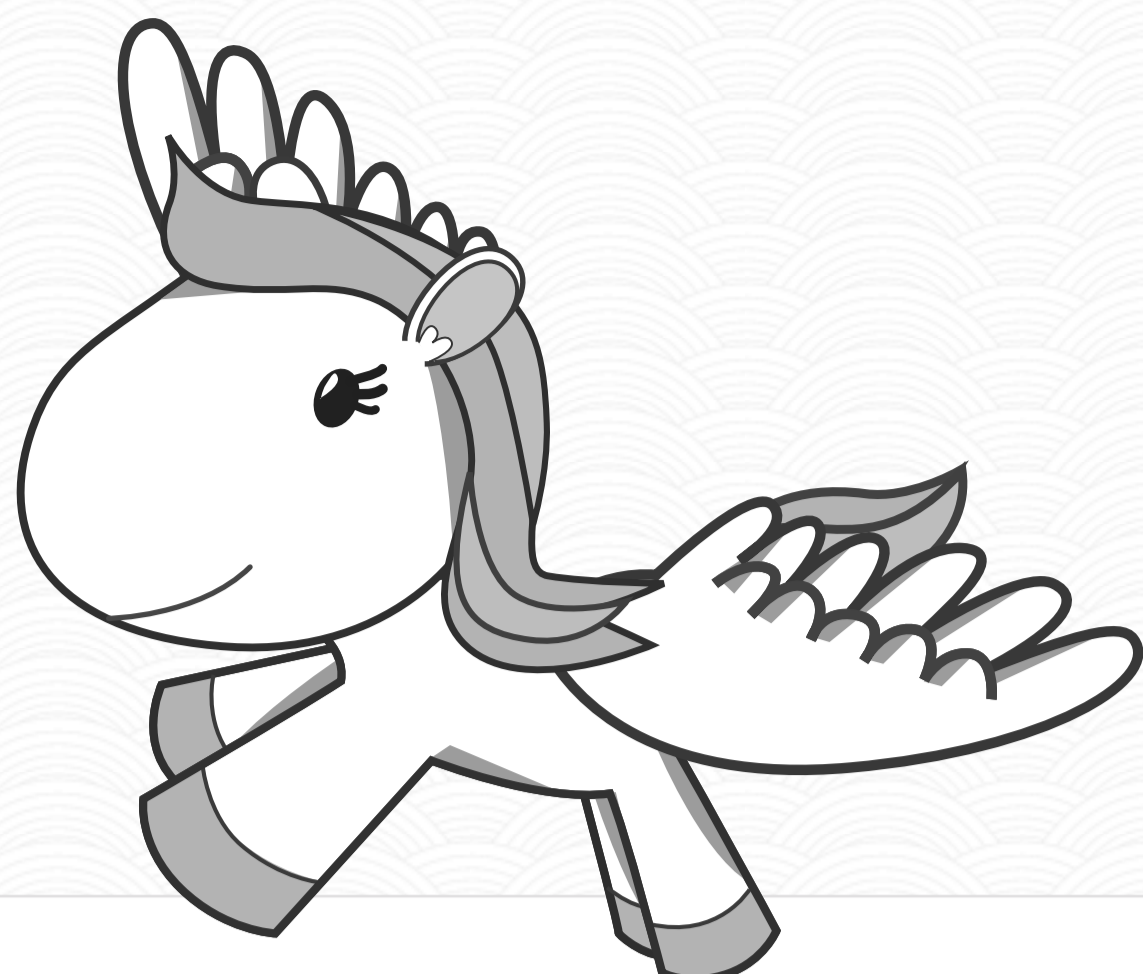


django

The Web framework for perfectionists with deadlines

Django makes it easier to build better Web apps more quickly and with less code. [Learn More](#).



- [Quick Start](#)
- [Get Django v1.4](#)
- [Documentation](#)

Who's using Django?



Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. It lets you build high-performing, elegant Web applications quickly.

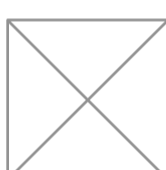
Django focuses on automating as much as possible and adhering to the [DRY principle](#).

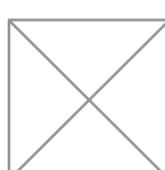
[Dive in by reading the overview](#) →

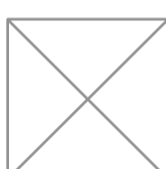
When you're ready to code, read the [installation guide](#) and [tutorial](#).

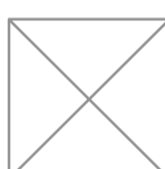
The Django framework

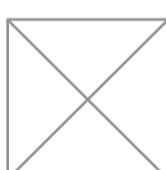
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin neque arcu, venenatis et tincidunt tempus, consectetur imperdiet justo.

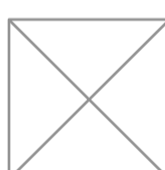
 **Object-relational mapper**
Define your [data models](#) entirely in Python. You get a rich, [dynamic database-access API](#) for free — but you can still write SQL if needed.

 **Template system**
Use Django's powerful, extensible and designer-friendly [template language](#) to separate design, content and Python code.

 **Automatic admin interface**
Save yourself the tedious work of creating interfaces for people to add and update content. [Django does that automatically](#), and it's production-ready.

 **Cache system**
Hook into memcached or other cache frameworks for [super performance](#) — caching is as granular as you need.

 **Elegant URL design**
Design pretty, [cruff-free URLs](#) with no framework-specific limitations. Be as flexible as you like.

 **Internationalization**
Django has full support for [multi-language applications](#), letting you specify translation strings and providing hooks for language-specific functionality.

Weblog

DjangoCon Europe 2012 is just 2 months away
by Daniele Procida and Russell Keith-Magee

Apr. 6, 2012 - DjangoCon Europe 2012 in Zürich, Switzerland is only two months away now. The conference runs from 4th to 6th June, followed by two days of sprints.

[Read more](#)

Django 1.4 released
by James Bennett

Mar. 23, 2012 - The Django team is pleased to announce the release of Django 1.4.

[Read more](#)

Django 1.4 release candidate 2 issued
by James Bennett

Mar. 14, 2012 - Today the Django team has issued Django 1.4 release candidate 2, a preview/testing package for the upcoming Django 1.4 release.

[Read more](#)

Community

[Mailing list django-users on Google Groups](#) ▶

[IRC channel #django on freenode.net](#) ▶

[Contributing to Django](#) ▶

[Community blog posts](#) ▶

[Django Q&A](#) ▶







[New / updated Django packages](#) ▶

[Django jobs](#) ▶

[Django links](#) ▶

django

- [Download](#)
- [Documentation](#)
- [Weblog](#)
- [Community](#)
- [Code](#)

-  **About**
A technical overview of Django
-  **Installation Guide**
Getting Django installed
-  **Documentation**
Everything you need to know
-  **Tutorial**
Writing your first Django app
-  **Community**
Mailing lists, blogs, and more
-  **More**
There's a lot of info here

Django
Quick Start Guide

```

$ pip install django
$ django-admin.py startproject myproject
$ cd myproject
$ django-admin.py startapp myapp
$ django-admin.py runserver

```

- 1 Download and install the Django
- 2 Create a new project
- 3 Create a new app inside the project
- 4 Run the development server

[Continue to the tutorial →](#)

Who's using Django?



Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. It lets you build high-performing, elegant Web applications quickly.

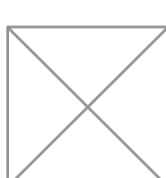
Django focuses on automating as much as possible and adhering to the [DRY principle](#).

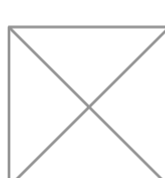
[Dive in by reading the overview →](#)

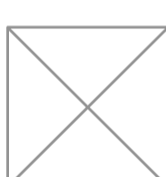
When you're ready to code, read the [installation guide](#) and [tutorial](#).

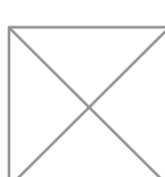
The Django framework

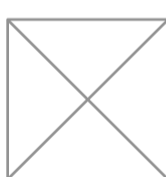
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin neque arcu, venenatis et tincidunt tempus, consectetur imperdiet justo.

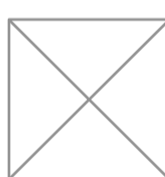
 **Object-relational mapper**
Define your [data models](#) entirely in Python. You get a rich, [dynamic database-access API](#) for free — but you can still write SQL if needed.

 **Template system**
Use Django's powerful, extensible and designer-friendly [template language](#) to separate design, content and Python code.

 **Automatic admin interface**
Save yourself the tedious work of creating interfaces for people to add and update content. [Django does that automatically](#), and it's production-ready.

 **Cache system**
Hook into memcached or other cache frameworks for [super performance](#) — caching is as granular as you need.

 **Elegant URL design**
Design pretty, [cruff-free URLs](#) with no framework-specific limitations. Be as flexible as you like.

 **Internationalization**
Django has full support for [multi-language applications](#), letting you specify translation strings and providing hooks for language-specific functionality.

Weblog

DjangoCon Europe 2012 is just 2 months away
by Daniele Procida and Russell Keith-Magee

Apr. 6, 2012 - DjangoCon Europe 2012 in Zürich, Switzerland is only two months away now. The conference runs from 4th to 6th June, followed by two days of sprints.

[Read more](#)

Django 1.4 released
by James Bennett

Mar. 23, 2012 - The Django team is pleased to announce the release of Django 1.4.

[Read more](#)

Django 1.4 release candidate 2 issued
by James Bennett

Mar. 14, 2012 - Today the Django team has issued Django 1.4 release candidate 2, a preview/testing package for the upcoming Django 1.4 release.

[Read more](#)

Community

[Mailing list django-users on Google Groups](#)

[IRC channel #django on freenode.net](#)

[Contributing to Django](#)

[Community blog posts](#)

[Django Q&A](#)







[New / updated Django packages](#)

[Django jobs](#)

[Django links](#)

django

- Download
- Documentation
- Weblog
- Community
- Code

-  **About**
A technical overview of Django
-  **Documentation**
Everything you need to know
-  **Community**
Mailing lists, blogs, and more
-  **Installation Guide**
Getting Django installed
-  **Tutorial**
Writing your first Django app
-  **More**
There's a lot of info here

Django Download Documentation Weblog Community Code Jobs

django

The Web framework for perfectionists with deadlines
Django makes it easier to build better Web apps more quickly and with less code. [Learn More.](#)

Quick Start Get Django v1.4 Documentation

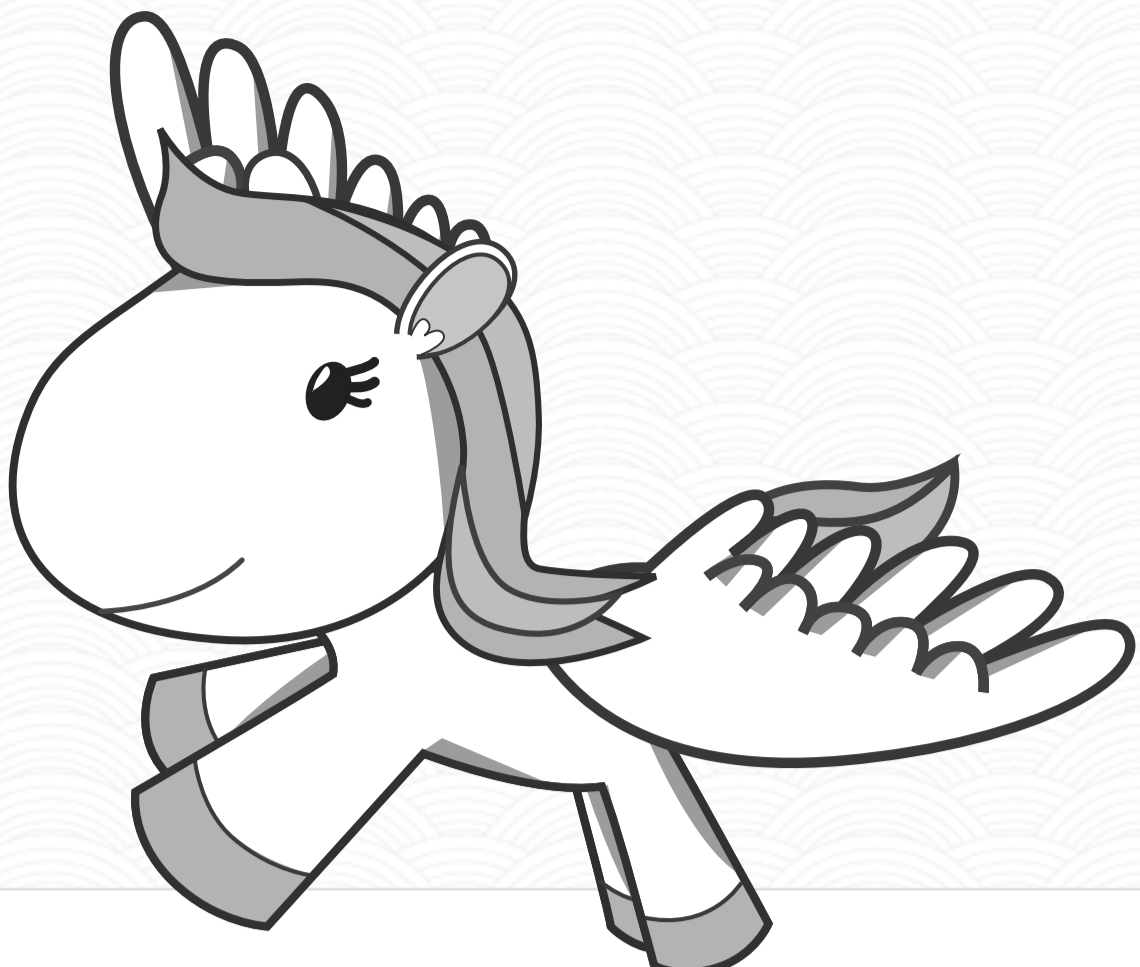
PIP Install from terminal
\$ pip install django

Source Tarball
Click to download latest release 1.4

Github Repository
\$ git clone https://github.com/django/django.git

Latest version **1.4** Released on **Mar. 23, 2012** Open sourced under [BSD license](#)

Washington Post mozilla DISQUS



Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. It lets you build high-performing, elegant Web applications quickly.

Django focuses on automating as much as possible and adhering to the [DRY principle](#).

[Dive in by reading the overview](#) →

When you're ready to code, read the [installation guide](#) and [tutorial](#).

The Django framework

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin neque arcu, venenatis et tincidunt tempus, consectetur imperdiet justo.

Object-relational mapper
Define your [data models](#) entirely in Python. You get a rich, [dynamic database-access API](#) for free — but you can still write SQL if needed.

Template system
Use Django's powerful, extensible and designer-friendly [template language](#) to separate design, content and Python code.

Automatic admin interface
Save yourself the tedious work of creating interfaces for people to add and update content. [Django does that automatically](#), and it's production-ready.

Cache system
Hook into memcached or other cache frameworks for [super performance](#) — caching is as granular as you need.

Elegant URL design
Design pretty, [cruff-free URLs](#) with no framework-specific limitations. Be as flexible as you like.

Internationalization
Django has full support for [multi-language applications](#), letting you specify translation strings and providing hooks for language-specific functionality.

Weblog

DjangoCon Europe 2012 is just 2 months away
by Daniele Procida and Russell Keith-Magee

Apr. 6, 2012 - DjangoCon Europe 2012 in Zürich, Switzerland is only two months away now. The conference runs from 4th to 6th June, followed by two days of sprints.

[Read more](#)

Django 1.4 released
by James Bennett

Mar. 23, 2012 - The Django team is pleased to announce the release of Django 1.4.

[Read more](#)

Django 1.4 release candidate 2 issued
by James Bennett

Mar. 14, 2012 - Today the Django team has issued Django 1.4 release candidate 2, a preview/testing package for the upcoming Django 1.4 release.

[Read more](#)

Community

Mailing list django-users on Google Groups ▶

IRC channel #django on freenode.net ▶

Contributing to Django ▶

Community blog posts ▶

Django Q&A ▶

New / updated Django packages ▶

Django jobs ▶

Django links ▶

django

- Download
- Documentation
- Weblog
- Community
- Code

- About**
A technical overview of Django
- Installation Guide**
Getting Django installed
- Documentation**
Everything you need to know
- Tutorial**
Writing your first Django app
- Community**
Mailing lists, blogs, and more
- More**
There's a lot of info here

Documentation for Django 1.4

Models

A model is the single, definitive source of data about your data. It contains the essential fields and behaviors of the data you're storing. Generally, each model maps to a single database table.

The basics:

- Each model is a Python class that subclasses `django.db.models.Model`.
- Each attribute of the model represents a database field.
- With all of this, Django gives you an automatically-generated database-access API; see Making queries.

Quick example

This example model defines a `Person`, which has a `first_name` and `last_name`:

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

`first_name` and `last_name` are [fields](#) of the model. Each field is specified as a class attribute, and each attribute maps to a database column.

The above `Person` model would create a database table like this:

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Some technical notes:

The name of the table, `myapp_person`, is automatically derived from some model metadata but can be overridden. See [Table names](#) for more details.. An id field is added automatically, but this behavior can be overridden. See [Automatic primary key fields](#). The CREATE TABLE SQL in this example is formatted using PostgreSQL syntax, but it's worth noting Django uses SQL tailored to the database backend specified in your settings file.

Using models

Once you have defined your models, you need to tell Django you're going to use those models. Do this by editing your settings file and changing the `INSTALLED_APPS` setting to add the name of the module that contains your `models.py`.

For example, if the models for your application live in the module `mysite.myapp.models` (the package structure that is created for an application by the `manage.py startapp` script), `INSTALLED_APPS` should read, in part:

```
INSTALLED_APPS = (
    #...
    'mysite.myapp',
    #...
)
```

When you add new apps to `INSTALLED_APPS`, be sure to run `manage.py syncdb`.

Fields

The most important part of a model -- and the only required part of a model -- is the list of database fields it defines. Fields are specified by class attributes.

Example:

```
class Musician(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    instrument = models.CharField(max_length=100)

class Album(models.Model):
    artist = models.ForeignKey(Musician)
    name = models.CharField(max_length=100)
    release_date = models.DateField()
    num_stars = models.IntegerField()
```

Fields

Each field in your model should be an instance of the appropriate Field class. Django uses the field class types to determine a few things:

- The database column type (e.g. INTEGER, VARCHAR).
- The widget to use in Django's admin interface, if you care to use it (e.g. `<input type="text">`, `<select>`).
- The minimal validation requirements, used in Django's admin and in automatically-generated forms.

Django ships with dozens of built-in field types; you can find the complete list in the [model field reference](#). You can easily write your own fields if Django's built-in ones don't do the trick; see [Writing custom model fields](#).

Field options

Each field takes a certain set of field-specific arguments (documented in the [model field reference](#)). For example, `CharField` (and its subclasses) require a `max_length` argument which specifies the size of the VARCHAR database field used to store the data.

There's also a set of common arguments available to all field types. All are optional. They're fully explained in the [reference](#), but here's a quick summary of the most often-used ones:

null

If `True`, Django will store empty values as `NULL` in the database. Default is `False`.

blank

If `True`, the field is allowed to be blank. Default is `False`.

Note that this is different than `null`. `null` is purely database-related, whereas `blank` is validation-related. If a field has `blank=True`, validation on Django's admin site will allow entry of an empty value. If a field has `blank=False`, the field will be required.

choices

An iterable (e.g., a list or tuple) of 2-tuples to use as choices for this field. If this is given, Django's admin will use a select box instead of the standard text field and will limit choices to the choices given.

Quick example

Using models

Fields

- Field types
- Field options
- Automatic primary key fields
- Verbose field names
- Relationships
 - Many-to-one relationships
 - Many-to-many relationships
 - Extra fields on many-to-many relationships
 - One-to-one relationships
- Models across files
- Field name restrictions
- Custom field types

Meta options

Model methods

Model inheritance

Documentation for Django 1.4

Models

A model is the way you think of data about your data. It contains the essential fields and behaviors of the data. Generally, each model maps to a single database table.

The basics:

- Each model is a Python class that subclasses `django.db.models.Model`.
- Each attribute of the model represents a database field.
- With all of this, Django gives you an automatically-generated database-access API; see Making queries.

Quick example

This example model defines a `Person`, which has a `first_name` and `last_name`:

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

`first_name` and `last_name` are **fields** of the model. Each field is specified as a class attribute, and each attribute maps to a database column.

The above `Person` model would create a database table like this:

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
    "last_name" varchar(30) NOT NULL
);
```

Some technical notes:

The name of the table, `myapp_person`, is automatically derived from some model metadata but can be overridden. See [Table names](#) for more details..

An `id` field is added automatically, but this behavior can be overridden. See [Automatic primary key fields](#).

The `CREATE TABLE` SQL in this example is formatted using PostgreSQL syntax, but it's worth noting Django uses SQL tailored to the database backend specified in your settings file.

Using models

Once you have defined your models, you need to tell Django you're going to use those models. Do this by editing your settings file and changing the `INSTALLED_APPS` setting to add the name of the module that contains your `models.py`.

For example, if the models for your application live in the module `mysite.myapp.models` (the package structure that is created for an application by the `manage.py startapp` script), `INSTALLED_APPS` should read, in part:

```
INSTALLED_APPS = (
    #...
    'mysite.myapp',
    #...
)
```

When you add new apps to `INSTALLED_APPS`, be sure to run `manage.py syncdb`.

Fields

The most important part of a model -- and the only required part of a model -- is the list of database fields it defines. Fields are specified by class attributes.

Example:

```
class Musician(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    instrument = models.CharField(max_length=100)

class Album(models.Model):
    artist = models.ForeignKey(Musician)
    name = models.CharField(max_length=100)
    release_date = models.DateField()
    num_stars = models.IntegerField()
```

Fields

Each field in your model should be an instance of the appropriate Field class. Django uses the field class types to determine a few things:

- The database column type (e.g. `INTEGER`, `VARCHAR`).
- The widget to use in Django's admin interface, if you care to use it (e.g. `<input type="text">`, `<select>`).
- The minimal validation requirements, used in Django's admin and in automatically-generated forms.

Django ships with dozens of built-in field types; you can find the complete list in the [model field reference](#). You can easily write your own fields if Django's built-in ones don't do the trick; see [Writing custom model fields](#).

Field options

Each field takes a certain set of field-specific arguments (documented in the [model field reference](#)). For example, `CharField` (and its subclasses) require a `max_length` argument which specifies the size of the `VARCHAR` database field used to store the data.

There's also a set of common arguments available to all field types. All are optional. They're fully explained in the reference, but here's a quick summary of the most often-used ones:

null

If `True`, Django will store empty values as `NULL` in the database. Default is `False`.

blank

If `True`, the field is allowed to be blank. Default is `False`.

Note that this is different than `null`. `null` is purely database-related, whereas `blank` is validation-related. If a field has `blank=True`, validation on Django's admin site will allow entry of an empty value. If a field has `blank=False`, the field will be required.

choices

An iterable (e.g., a list or tuple) of 2-tuples to use as choices for this field. If this is given, Django's admin will use a select box instead of the standard text field and will limit choices to the choices given.

Search

Django 1.4

Quick example

Using models

Fields

- Field types
- Field options
- Automatic primary key fields
- Verbose field names
- Relationships
 - Many-to-one relationships
 - Many-to-many relationships
 - Extra fields on many-to-many relationships
 - One-to-one relationships
- Models across files
- Field name restrictions
- Custom field types

Meta options

Model methods

Model inheritance

django

The Web framework for perfectionists with deadlines

Django makes it easier to build better Web apps more quickly and with less code. [Learn More.](#)

Quick Start

Documentation

Latest version Django v1.4

Who's using Django?

Latest version Django v1.4

Who's using Django?

Instagram

Pinterest

The Washington Post

DISQUS

Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Meet Django

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

Developed by a fast-moving online-news operation, Django was designed to handle two challenges: the intensive deadlines of a newsroom and the stringent requirements of the experienced Web developers who wrote it. It lets you build high-performing, elegant Web applications quickly.

Django focuses on automating as much as possible and adhering to the [DRY principle](#).

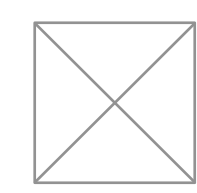
[Dive in by reading the overview](#) →

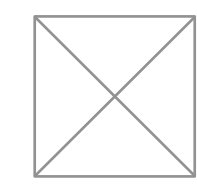
When you're ready to code, read the [installation guide](#) and [tutorial](#).

When you're ready to code, read the [installation guide](#) and [tutorial](#).

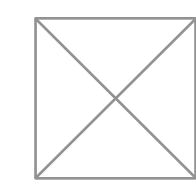
The Django framework

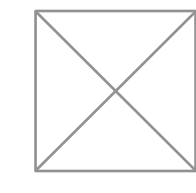
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin neque arcu, venenatis et tincidunt tempus, consectetur imperdiet justo.

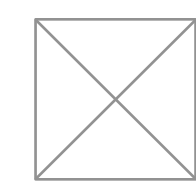
 **Object-relational mapper**
Define your [data models](#) entirely in Python. You get a rich, [dynamic database-access API](#) for free — but you can still write SQL if needed.

 **Automatic admin interface**
Save yourself the tedious work of creating interfaces for people to add and update content. [Django does that automatically](#), and it's

 **Elegant URL design**
Design pretty, [craft-free URLs](#) with no framework-specific limitations. Be as flexible as you like.

 **Template system**
Use Django's powerful, extensible and designer-friendly [template language](#) to separate design, content and Python code.

 **Cache system**
Hook into memcached or other cache frameworks for [super performance](#) — caching is as granular as you need.

 **Internationalization**
Django has full support for [multi-language applications](#), letting you specify translation strings and providing hooks for language-specific functionality.

Weblog

DjangoCon Europe 2012 is just 2 months away

by Daniele Procida and Russell Keith-Magee

Apr. 6, 2012 - DjangoCon Europe 2012 in Zürich, Switzerland is only two months away now. The conference runs from 4th to 6th June, followed by two days of sprints.

Read More

Django 1.4 released

by James Bennett

Mar. 23, 2012 - The Django team is pleased to announce the release of Django 1.4.

Read More

announce the release of Django 1.4.

Read More

Community

Mailing list [django-users on Google Groups](#)

IRC channel [#django on freenode.net](#)

Contributing to Django


Community blog posts


Django Q&A


New / updated Django packages


Django links


django

 **About**
A technical overview of Django

 **Documentation**
Everything you need to know

 **Community**
Mailing lists, blogs, and more

 **Installation Guide**
Getting Django installed

 **Tutorial**
Writing your first Django app

 **More**
There's a lot of info here

django

Django was originally developed at World Online, the Web department of a newspaper in Lawrence, Kansas, USA. Django's now run by an international team of volunteers; you can read all about them over at the [list of committers](#).

© 2005-2012 Django Software Foundation unless otherwise noted. Django is a registered trademark of the Django Software Foundation. Linux Web hosting graciously provided by Media Temple.

Close