

DJANGO OVERVIEW





WHAT IS DJANTO?

- **Django is a high-level Python Web Framework.**
- **Rapid development**
- **Free and open source**
- **Has a thriving and active community.**
- **Great documentation.**

The Django logo is displayed within a dark green square. The word "django" is written in a white, lowercase, sans-serif font. The letter 'j' is stylized with a long, curved tail that extends downwards and to the left, ending under the 'a'.

WHERE DID IT COME FROM?

- Django was initially developed between 2003 and 2005 by a web team who were responsible for creating and maintaining newspaper websites.
- After creating a number of sites, the team began to factor out and reuse lot of common code and design patterns.
- This common code evolved into a generic web development framework, which was open-sourced as the "Django" project in **July 2005**.

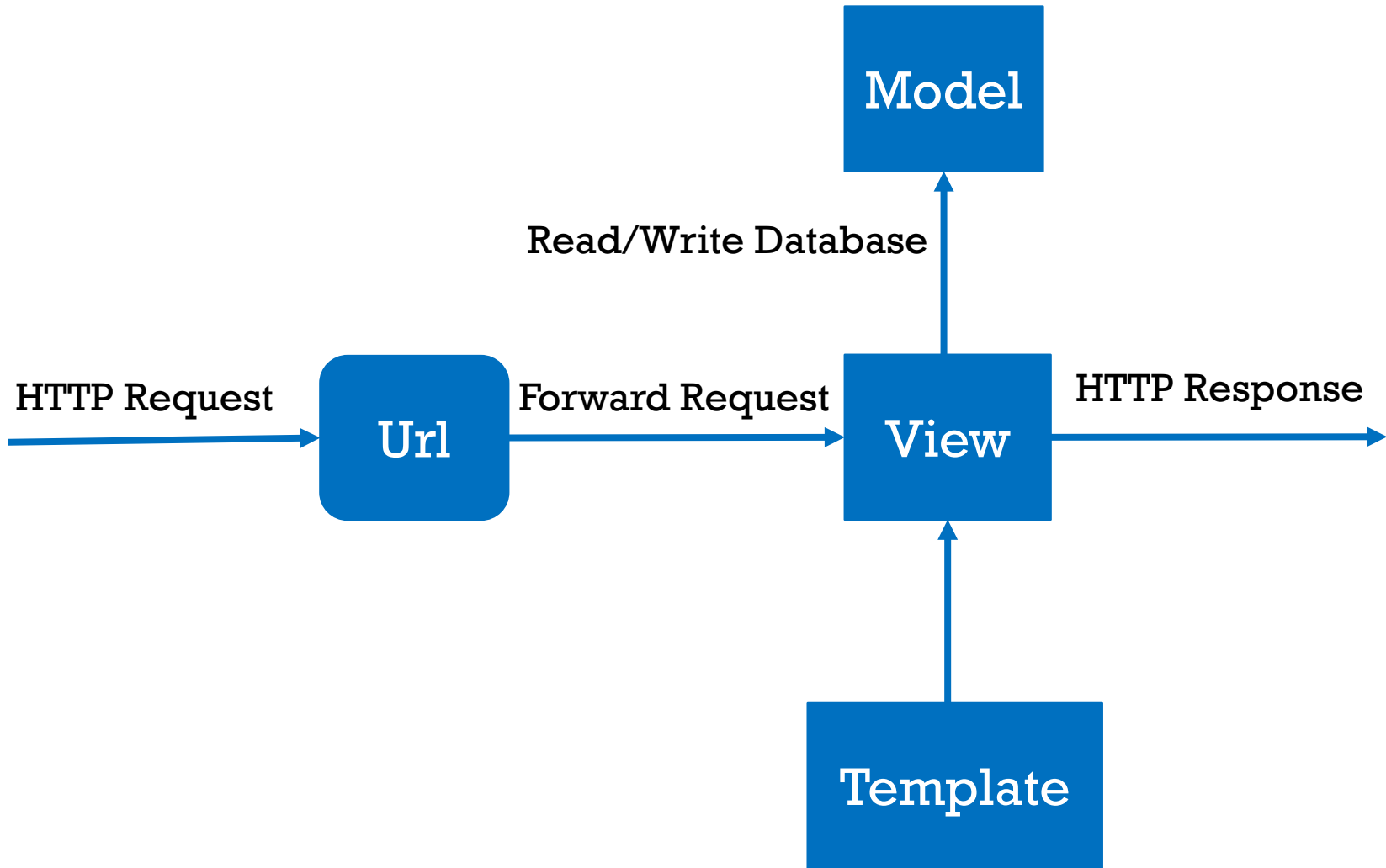


DJANGO MVT PATTERN

- **Models**: manage(add, modify, delete) records in database
- **View**: A view is a request handler function, which receives HTTP requests and returns HTTP responses.
- **Template**: is a text file defining the structure or layout
- **URLs**: process requests from every single URL



DJANGO MVT PATTERN



MVC – MVT PATTERN

- **When talking about web development, we usually talk about MCV architecture.**
- **MVC pattern is based on three components: Model, View and Controller.**



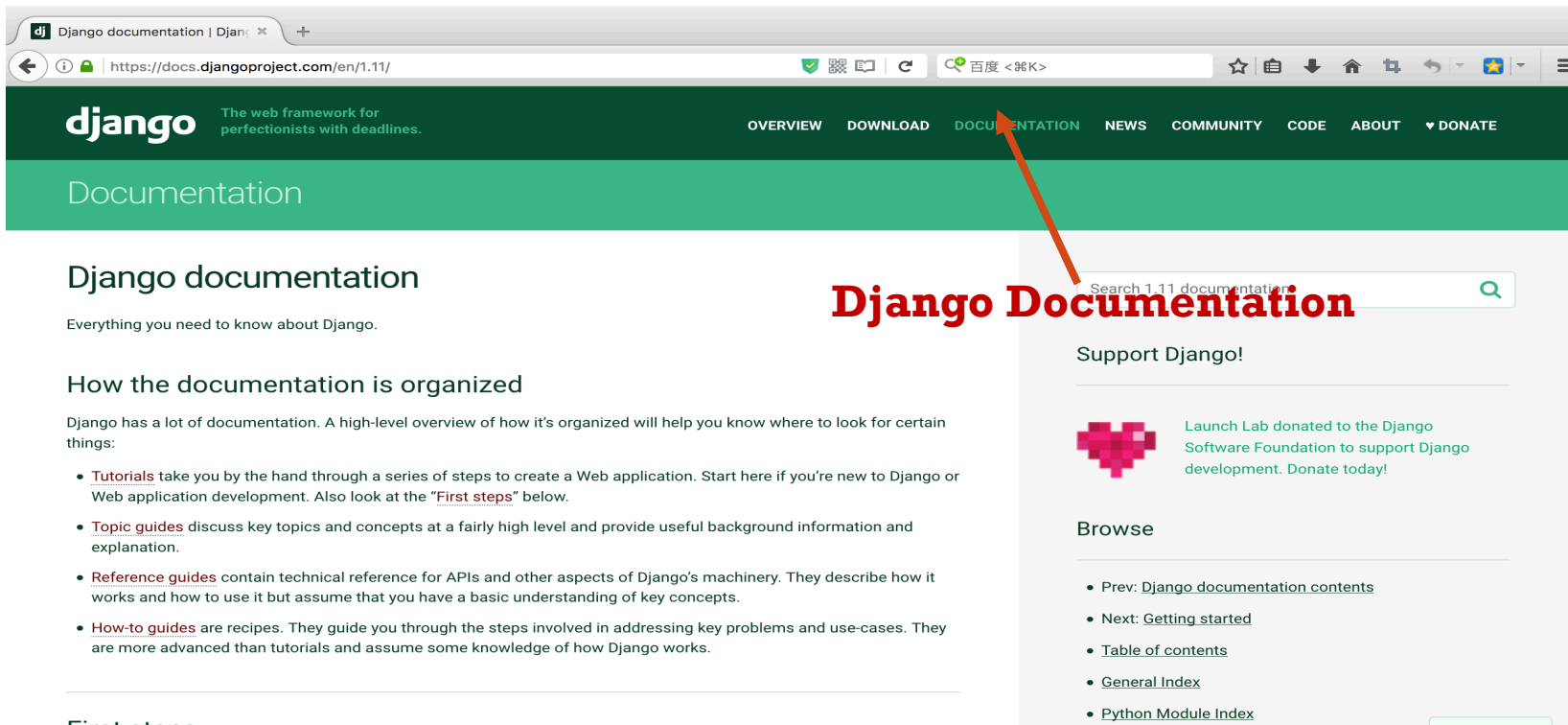
MVC – MVT PATTERN

- **Model-View-Template (MVT)**
- **The main difference between the two patterns is that Django itself takes care of the Controller part, leaving us with the template.**
- **The template is HTML file mixed with Django Template Language (DTL)**



DJANGO

■ <https://www.djangoproject.com/>



The screenshot shows the Django documentation website. The browser's address bar displays the URL `https://docs.djangoproject.com/en/1.11/`. The navigation bar is dark green with the Django logo and the tagline "The web framework for perfectionists with deadlines." on the left, and a series of links (OVERVIEW, DOWNLOAD, DOCUMENTATION, NEWS, COMMUNITY, CODE, ABOUT, DONATE) on the right. A red arrow points from the word "Django Documentation" (written in red text over the image) to the "DOCUMENTATION" link in the navigation bar. Below the navigation bar is a green banner with the word "Documentation". The main content area has the heading "Django documentation" and the subtext "Everything you need to know about Django." Below this is a section titled "How the documentation is organized" which explains the structure of the documentation and lists four types of guides: Tutorials, Topic guides, Reference guides, and How-to guides. On the right side of the page, there is a search bar labeled "Search 1.11 documentation", a section titled "Support Django!" with a pink heart icon and text about Launch Lab's donation, and a "Browse" section with a list of links: Prev: Django documentation contents, Next: Getting started, Table of contents, General Index, and Python Module Index.

Django documentation

Everything you need to know about Django.

How the documentation is organized


Django has a lot of documentation. A high-level overview of how it's organized will help you know where to look for certain things:

- [Tutorials](#) take you by the hand through a series of steps to create a Web application. Start here if you're new to Django or Web application development. Also look at the "[First steps](#)" below.
- [Topic guides](#) discuss key topics and concepts at a fairly high level and provide useful background information and explanation.
- [Reference guides](#) contain technical reference for APIs and other aspects of Django's machinery. They describe how it works and how to use it but assume that you have a basic understanding of key concepts.
- [How-to guides](#) are recipes. They guide you through the steps involved in addressing key problems and use-cases. They are more advanced than tutorials and assume some knowledge of how Django works.

First steps

Search 1.11 documentation

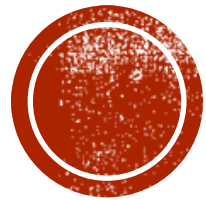
Support Django!

 Launch Lab donated to the Django Software Foundation to support Django development. Donate today!

Browse

- Prev: [Django documentation contents](#)
- Next: [Getting started](#)
- [Table of contents](#)
- [General Index](#)
- [Python Module Index](#)





DJANGO ENVIRONMENT



DJANGO SETUP – STEP 1

- **Check Python**

- Being a Python web framework, Django require Python
- Before install Django, check the version of Python

```
$ python -V
```

```
Python 3.6.1
```



DJANGO SETUP – STEP 2

■ Install Django

- `pip install django` // install the newest version
- If want to install previous version
 - `pip install django==1.10.7`

```
$ pip install django
```

```
# install previous version
```

```
$ pip install django==1.10.7
```



DJANGO SETUP – STEP 3

- **Verify**

- **Open terminal and type below command:**

```
$ python
>>> import django
>>> print(django.get_version())
1.11.4
```



LET'S START A PROJECT!

- **Creating Django project**
 - Before creating, cd to the directory where you want to store your code
 - `django-admin startproject <project_name>`

```
# create a django project named library  
$ django-admin startproject library
```



LET'S START A PROJECT!

```
library
├── library
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py
```



WHAT IS CREATED?

- **__init__.py** treat this folder as package, just for python
- **settings.py** contains all the website settings, such as database, static file and so on.
- **urls.py** defines the site url-to-view mapping.
- **wsgi.py** is used to help your Django application communicate with the web server.



WHAT IS CREATED?

- **manage.py** is kind of your project local django-admin for interacting with your project via command line.
- **To get full list of command:**
 - **python manage.py help**



VIRTUAL ENVIRONMENTS

- **A virtual environment is a tool to keep the dependencies required by different projects in separate places, by creating virtual Python environments from them.**
- **It solves the “project X depends on version 1.x but project Y need version 4.x”.**



VIRTUALENV

- **Virtualenv is a tool to create isolated Python environments.**
- **Virtualenv creates a folder which contains all necessary executables packages that a Python project would need.**



VIRTUALENV -- INSTALL

- **Install**
 - **pip install virtualenv**
- **Test your installation**
 - **virtualenv --version**

```
$ virtualenv --version  
15.1.0
```



VIRTUALENV -- CREATE

- Create a virtual environment for a project
 - `$ cd project_folder`
 - `$ virtualenv <env_name>`
- Using the Python interpreter of your choice
 - `$ virtualenv -p python3 <env_name>`

```
$ cd <project_folder>  
$ virtualenv env
```

```
# define python version  
$ virtualenv -p python2.7 env
```



VIRTUALENV -- ACTIVATE

- To begin using the virtual environment, it needs to be *activated*.

```
# OS or Linux
```

```
$ source env/bin/activate
```

```
# Windows
```

```
$ env\Scripts\activate
```



VIRTUALENV -- DEACTIVATE

- If you are done working in the virtual environment for the moment, you can *deactivate*

```
(env) $ deactivate  
$
```



VERIFY

- **Let's verify your Django project works.**

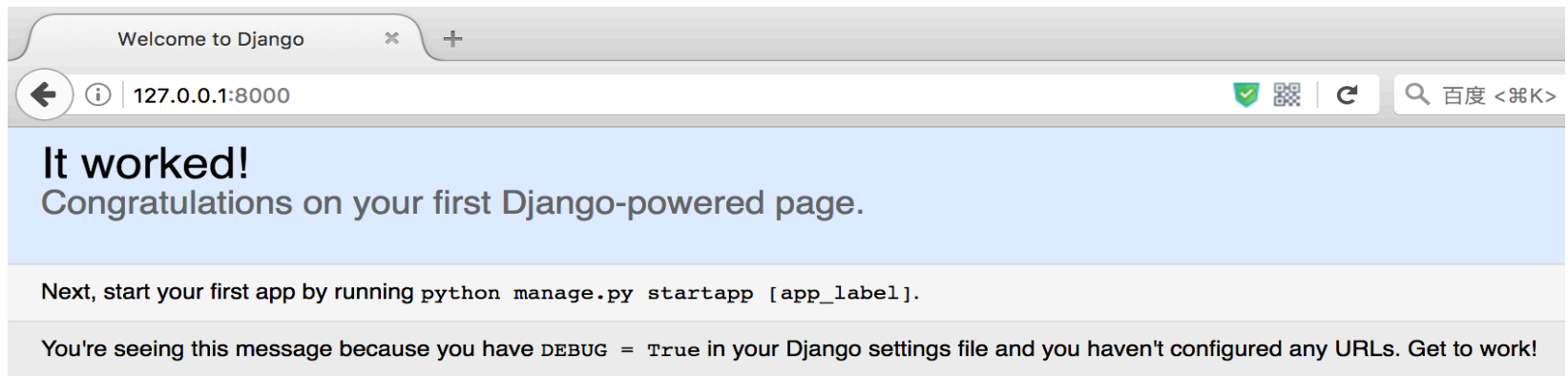
```
$ cd library  
$ env\Scripts\activate
```

```
(env)$ python manage.py runserver
```

- **Now, open the web page at <http://127.0.0.1:8000>**
- **By default, the runserver command starts the development server on the internal IP at port 8000.**



IT WORKED!



CHANGING THE PORT

- **If you want to change the server's port, pass it as a command-line argument. For instance, this command starts the server on port 8080:**

```
(env)$ python manage.py runserver 8080
```



CHANGING THE PORT

- **If you want to change the server's IP, pass it along with the port.**

```
(env)$ python manage.py runserver 0.0.0.0:8000
```

- **Open settings.py**

```
ALLOWED_HOSTS = [ '*' ]
```



PYCHARM

- **PyCharm**
 - **Python IDE for Professional Developers**



Questions?

