



# Django 101

By Sebastián Lagomarsino. May 2021.



Facts



# What is Django?

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. It's free and open source.

Although Django was created in the fall of 2003, it is constantly being updated. The latest stable version is 3.2, released on April 6th 2021.

The three key values of Django are:

- Ridiculously fast.
- Reassuringly secure.
- Exceedingly scalable.



# Django vs Django REST Framework

In simple words:

- **Django:** It provide features for standard web app.
- **Django REST Framework:** It provide features to build standard REST API.



# Architecture

Django follows MVC pattern very closely but it uses slightly different terminology. Django is essentially an MTV framework.

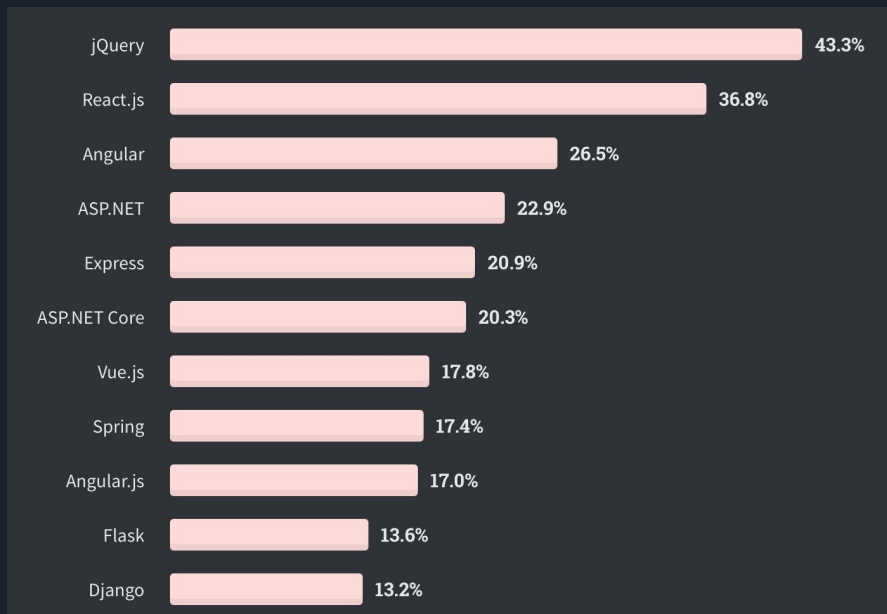
Django uses the term **T**emplates for Views and **V**iews for Controller. In other words, in Django views are called templates and controllers are called views.

Hence HTML code will be in **T**emplates and Django code will be in **V**iews and **M**odels.



# Django in the world

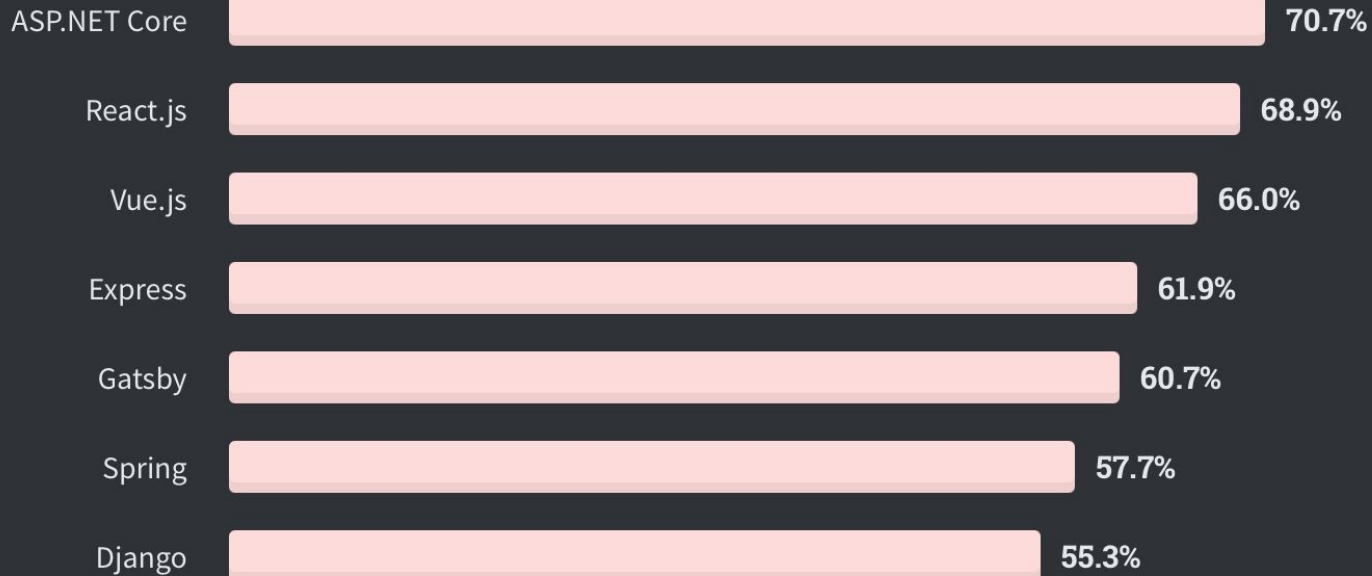
According a survey made by StackOverflow with 36,291 responses.





# Loved

According a survey made by StackOverflow. % of developers who are developing with the language or technology and have expressed interest in continuing to develop with it.





# Basics





# Models

A model is the single, definitive source of information 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.

```
class HelloWorldModel(models.Model):
    """Model Example to show some Django model fields"""
    title = models.CharField(max_length=255)
    price = models.IntegerField()
    date = models.DateTimeField(null=True, blank=True)
    image = models.ImageField(null=True, upload_to=image_file_path)
    user = models.ForeignKey(
        settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE,
    )

    def __str__(self):
        return self.title
```



# Serializers

Django's serialization framework provides a mechanism for “translating” Django models into other formats.

```
class HelloWorldSerializer(serializers.ModelSerializer):  
    """Serializer for HelloWorldModel objects"""  
    class Meta:  
        model = HelloWorldModel  
        fields = ('id', 'title')  
        read_only_fields = ('id', )
```



# Views

A view is a callable which takes a request and returns a response.



Using Django:


```
from django.core.serializers import serialize
from django.http import HttpResponse
from django.views import View

class HelloWorldView(View):
    """Django view"""
    def get(self, request, *args, **kwargs):
        queryset = HelloWorldModel.objects.all()
        json_data = serialize('json', queryset, fields=('id', 'title'))
        return HttpResponse(json_data, content_type='application/json')
```

Using Django REST Framework:

```
from rest_framework import viewsets

class HelloWorldDRFViewSet(viewsets.ModelViewSet):
    """Django REST Framework view"""
    queryset = HelloWorldModel.objects.all()
    serializer_class = serializers.HelloWorldSerializer
```

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with faint, lighter blue diagonal stripes.

# Creating our first Django project

A decorative graphic on the left side of the slide. It consists of a blue parallelogram and a light green parallelogram, both tilted at an angle. The blue shape is in the foreground, and the green shape is partially behind it. They are set against a dark blue background with diagonal stripes of varying shades.

# Docker



# Creating DockerFile

```
FROM python:3.7-alpine
```

```
COPY ./requirements.txt /requirements.txt
```

```
RUN pip install -r /requirements.txt
```

```
RUN mkdir /django
```

```
WORKDIR /django
```

```
COPY ./django /django
```

```
RUN adduser -D user
```

```
USER user
```



# Creating docker-compose.yml

```
version: "3"

services:
  django:
    build:
      context: .
    ports:
      - "8000:8000"
    volumes:
      - ./django:/django
    command: >
      sh -c "python manage.py runserver 0.0.0.0:8000"
```





# Useful Commands



# Creating Django project

```
→ django-hello-world git:(main) docker-compose run django django-admin.py startproject django .
```



# Creating Django app

```
→ django-hello-world git:(main) docker-compose run django python manage.py startapp hello_world
```



# Creating and running migrations

```
→ django-hello-world git:(main) docker-compose run django python manage.py makemigrations  
→ django-hello-world git:(main) docker-compose run django python manage.py migrate
```



Demo



# About

The demo contains one Django project with three Django apps:

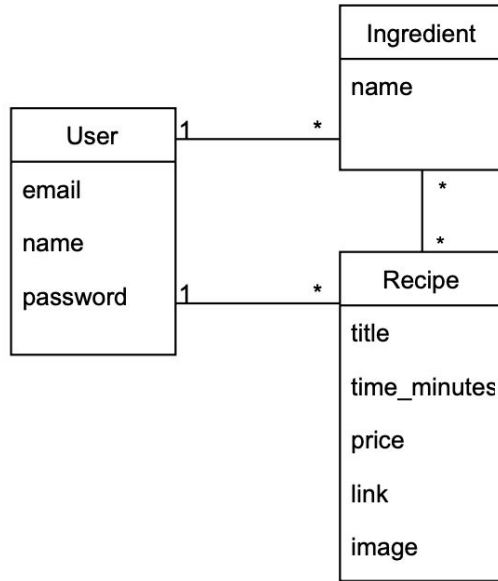
- `hello_world`: has a view that renders a Django template.
- `recipe`: where you can manage Recipes with Ingredients.
- `users`: where you can manage Users.

It provides a REST Api that we can make request in order to:

- `Recipes`: create, list, update and delete.
- `Ingredients`: create and list.
- `Users`: create, get token, update.

It also has a Django Administration web page.

# MER





# Useful references

- Views: <https://docs.djangoproject.com/en/3.2/topics/class-based-views/>
- Serializers: <https://docs.djangoproject.com/en/3.2/topics/serialization/>
- Models: <https://docs.djangoproject.com/en/3.2/topics/db/models/>
- ViewSets: <https://www.django-rest-framework.org/api-guide/viewsets/>
- Generic views: <https://www.django-rest-framework.org/api-guide/generic-views/>
- Mixins: <https://www.django-rest-framework.org/api-guide/generic-views/#mixins>
- Demo GitHub repository: <https://github.com/slagomarsino/django-hello-world>





Thank you :hug: