

Cute cats web app (Django + VueJs) — Settings



Jrmym

Follow

Jan 23, 2018 · 6 min read

I can't find any friendly sources about building project using Django as backend service and VueJs as frontend. The goal is to keep advantages of the two frameworks.

- Django to manage database, computing and serving datas
- VueJS to create powerfull web components and reuse them anywhere and making request to the Django API Rest

Summary

Part I—[Settings](#)

Part II—[Create VueJs views](#)

Part III—[Create Rest API with DRF](#)

Part IV—[Get datas from API with VueJs](#)

Starting here !



What do you need now?

— Django—VueJs 2.0—Webpack —

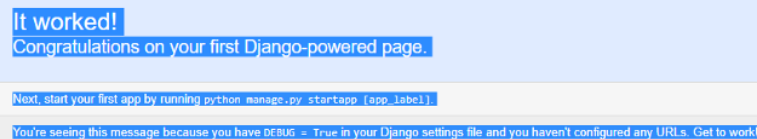
Is required to install Python and NodeJs before.

Create Django Project

Open command shell and create your django project

```
python pip install django //installing django
django-admin startproject <project-name> //creating
django project
cd <project-name>
manage.py runserver //starting local server
```

Open your browser and go to <http://localhost:8000>. You should see this text on the page.

A screenshot of a web browser displaying the Django default page. The page has a light blue header with the text "It worked!" and "Congratulations on your first Django-powered page." Below the header, there is a light gray section with two lines of text: "Next, start your first app by running python manage.py startapp [app_label]." and "You're seeing this message because you have DEBUG = True in your Django settings file and you haven't configured any URLs. Get to work!"

It worked!
Congratulations on your first Django-powered page.

Next, start your first app by running python manage.py startapp [app_label].

You're seeing this message because you have DEBUG = True in your Django settings file and you haven't configured any URLs. Get to work!

In your explorer, you should have a new folder named with your *project-name*.

```
myproject
|_ myproject
|_ manage.py
```

Job is done ! Your project render a template. Now we want to integrate VueJS to our project

Integrate VueJS part

Assuming NodeJs is install on your computer. Run following command in the shell. You must be in *myproject* folder

```
npm install -g vue-cli
vue init webpack-simple
```

The last command will create a VueJs project configured to work with Webpack. The shell will prompt some questions

```
Generate project in current directory ? Yes
? Project name : myproject
? Project description : use default description
? Author : default
? Lincense : default
? Use Sass : Yes
```

If you are looking to your *myproject* folder, new files are present.

```
-myProject
|_ myProject
|_ src
|_ assets
|_ App.vue
|_ main.js

|_ .babelrc
|_ .editorconfig
|_ index.html
|_ manage.py
|_ package.json
|_ README.md
|_ webpack.config.js
```

The package.json file contains all informations about dependencies your application need to run correctly. It's look like this :

```
"name": "myproject",
"description": "A Vue.js project",
"version": "1.0.0",
"scripts": {
  "dev": "cross-env NODE_ENV=development webpack-dev-server --open --hot",
  "build": "cross-env NODE_ENV=production webpack --progress --hide-modules"
},
"dependencies": {
  "vue": "^2.5.11"
},
...
"devDependencies": {
  "babel-core": "^6.26.0",
```

```
"babel-loader": "^7.1.2",
"babel-preset-env": "^1.6.0",
"babel-preset-stage-3": "^6.24.1",
"cross-env": "^5.0.5",
"css-loader": "^0.28.7",
"file-loader": "^1.1.4",
"node-sass": "^4.5.3",
"sass-loader": "^6.0.6",
"vue-loader": "^13.0.5",
"vue-template-compiler": "^2.4.4",
"webpack": "^3.6.0",
"webpack-dev-server": "^2.9.1"
}
}
```

When you are running *npm install*, all dependencies will be installed in a *node_module* folder.

```
npm install //and wait
```

Now, your VueJs application is ready to work.

```
npm run dev
```

Your app is build in developpment mode. Your browser will enter by *index.html*. In this one, he find an instruction to load *src/main.js*

```
src/main.js

import Vue from 'vue'
import App from './App.vue'

new Vue({
  el: '#app',
  render: h => h(App)
})
```

Here, we are creating a Vue and rendering the App content from *src/App.vue*

```
src\App.vue

<template>
  <div id="app">
    ...
  </div>
</template>

<script>
export default {
  name: 'app',
  data () {
    return {
      msg: 'Welcome to Your Vue.js App'
    }
  }
}
</script>

<style lang="scss">
#app {
  // some scss rules
}
</style>
```

So when you running your application, on localhost://8080, you should see the *App.vue* template in your browser



Welcome to Your Vue.js App

Essential Links

[Core Docs](#) [Forum](#) [Community Chat](#) [Twitter](#)

Ecosystem

[vue-router](#) [vuex](#) [vue-loader](#) [awesome-vue](#)

Cool hum ?! Here we are. We have a back end part working on a localhost, and a front end part on an another localhost. Just take a time to enjoy it.



Well, now we will see how to join VueJs App component to a Django template. It's mean that when we're running the local server with Django, we will see the App template.

Fusion

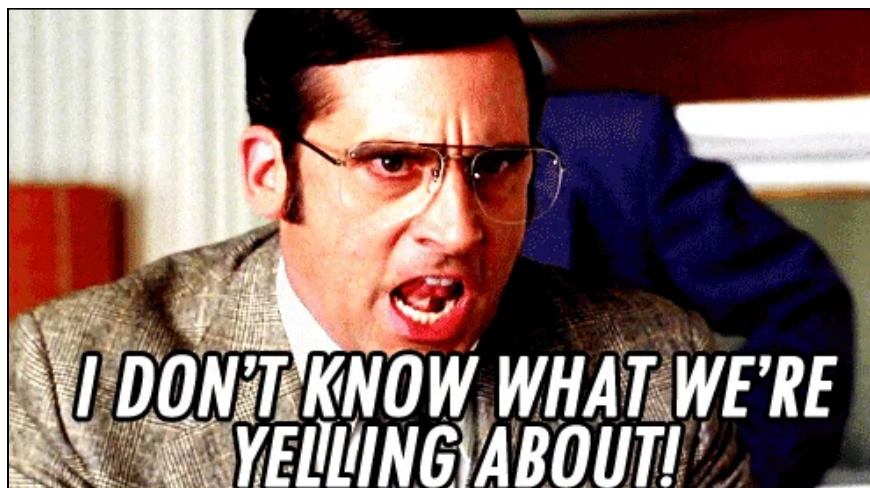
We need to add some tools to our project.

```
npm install webpack-bundle-tracker --save-dev  
npm install write-file-webpack-plugin --save-dev
```

it should be adding to your *package.json devdependencies*.

Compile frontend part

What do we want now ?



My idea was to compile the front end part, get this in a file. In this way, I could call him anywhere I want. To do that, we have to modify a little bit the Webpack configuration file. So open it and see what we can do.

```
var path = require('path')
var webpack = require('webpack')
module.exports = {
  entry: 'main.js',
  output: {
    path: path.resolve(__dirname, './dist'),
    filename: 'bundle.js'
  },
  module: {
    rules: [
      {
        test: /\.css$/,
        use: [
          'vue-style-loader',
          'css-loader'
        ],
      },
      ...
    ]
  },
  resolve: {
    alias: {
      'vue$': 'vue/dist/vue.esm.js'
    },
    extensions: ['*', '.js', '.vue', '.json']
  }
}
```

- entry: is your starting point.
- path: where to find compiled file
- filename: ...

- rules: tell to Webpack which tool to use when he meet those extensions

As we saw before, that config works but we don't have any file at the end. Let's add some require. One is tracking and save in a file the location of the compiled file. Unfortunately, the compiled file is kept in memory. So, we add an another tool to save this one on computer.

Let's add some require

```
var BundleTracker = require('webpack-bundle-tracker')
var WriteFilePlugin = require('write-file-webpack-plugin')
```

And tell to Webpack we want to use them

```
entry: '...',
output: {
  ...
},
plugins: [
  new BundleTracker({filename: 'webpack-stats.json'}),
  new WriteFilePlugin()
],
module: {
  ...
}
```

Before running compilation, we have to decide where is our starting point and where to store the output compiled file.

```
entry: './src/main.js',
output: {
  path: path.resolve(__dirname, './dist/'),
  filename: 'bundle.js'
},
```

Go to the command shell and execute *npm run build*. Wait for some seconds and TADA ! We have a new file *webpack-stats.json*. If we open it, we can see that the *bundle.js* file is located is *./dist/* folder.


```
-myProject
|_ dist
|_ assets
|_ bundles.js
|_ myProject
|_ src
|_ App.vue
|_ main.js
|_ index.html
|_ package.json
|_ webpack-stats.json
|_ webpack.config.js
```



How to use this bundle file with Django

We are going to create a Django view to call the existing *index.html* file. In this file we will use the `<app>` tag defined by *App.vue* and render it.

```
index.html

<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    <div id="app">
      <app> </app>
    </div>
  </body>
</html>
```

First, we have to install a tool to permit to Django using bundle.js file

```
pip install django-webpack-loader

# Then in settings.py
# Add 'webpack_loader' to INSTALLED_APP

# Add path to STATICFILES_DIRS
# paste it

STATIC_URL = '/public/'

STATIC_ROOT = os.path.join(BASE_DIR, 'public')

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'dist'),
)

WEBPACK_LOADER = {
    'DEFAULT': {
        'CACHE': not DEBUG,
        'BUNDLE_DIR_NAME': '',
        'STATS_FILE': os.path.join(BASE_DIR, 'webpack-
stats.json'),
        'POLL_INTERVAL': 0.1,
        'TIMEOUT': None,
        'IGNORE': ['.+\\.hot-update.js', '.+\\.map']
    }
}

# More settings
# Add path into TEMPLATES_DIR
TEMPLATES = [
    {
        'DIRS': [BASE_DIR],
        ...
    },
]
```

Now you can load this bundle.js file like a static file in index.html

```
index.html

<!DOCTYPE html>
<html lang="en">
  <head>
    ...
  </head>
  <body>
    {% load render_bundle from webpack_loader %}
```

```

<div id="app">
  <app> </app>
</div>

{% render_bundle 'main' %}
</body>
</html>

```

Finally, we have to route those *index.html* in *urls.py*.

```

-myProject
|_ dist
|_ myProject
|_ settings.py
|_ urls.py
|_ src
|_ App.vue
|_ main.js
|_ index.html

```

In *urls.py*

```

from django.conf.urls import url
from django.contrib import admin
from django.contrib.staticfiles.urls import
staticfiles_urlpatterns
from django.views.generic import TemplateView

urlpatterns = [
    url(r'^$',
        TemplateView.as_view(template_name='index.html'),
        name='uHome'
    ),
    url(r'^admin/', admin.site.urls),
]

urlpatterns += staticfiles_urlpatterns()

```

Main.js change a little bit to define the *<app>* tag.

```

import Vue from 'vue'
import App from './App.vue'

```

```
new Vue({  
  el: '#app',  
  components: {  
    'app': App  
  }  
})
```

Guess what !? we're done !

```
//Compile VueJS App  
npm run build
```

```
#collect static files in public folder  
python manage.py collectstatic
```

```
#run Django server  
python manage.py runserver
```



Go to the page and you should see the Vue.Js App template display on your Django localhost port.

Good Job ! We did it together. In the next part, i'll show you how to render your own Vue with Django.

Next part—custom VueJs template



