AngD001 Angular\_Django-CMS

# Prerequisites and envs\AngD001\backend\.venv:

* Windows **Python** 3.11.x from <https://www.python.org/downloads/windows/>. Install for all users in windows. Python Virtual Environments use python 3.11. Later we try to upgrade to 3.12.
* **VSCode** from <https://code.visualstudio.com/Download>. Start by typing *code in Win-Search Field.* Open New Window and open New Terminal by [Ctrl+Shift-` ]
* **Windows Git** from <https://gitforwindows.org/>.
* **NodeJS-lts** from <https://nodejs.org/en/download>. JasaScript for Angular frontend.
* **Virtual Environment** will be created for our MP50\_project AbgCMD000.
* **Envs folder** will be created for all envs, at [C:\Users\%username%\envs\AngCMS000](file:///C:\Users\%25username%25\envs\AngCMS000).
* **Open VSCode** [code & [Ctrl+Shift+` ] and copy-paste in Terminal: cmd.
* **Installation of Django-CMS by Hand,** [**view link**](https://docs.django-cms.org/en/release-3.4.x/how_to/install.html#:~:text=How%20to%20install%20django%20CMS%20by%20hand%20%C2%B6,%C2%B6%20...%208%20Launch%20the%20project%20%C2%B6%20)**. On purpose I use this mode of installation because I want to understand the packages in detail. Open VScode > File > new Windows & Close old Windows > [Ctrl+Shift+` ] to open the Terminal. Copy part by part (rows between (1)’s etc. and past by part and follow exactly the rows in order:**

|  |
| --- |
| cmd  cd c:\Users\%username%\ md envs & cd envs md AngD001 & cd AngD001  md backend & cd backend python -m pip install --upgrade pip  python -m venv .venv  .venv\Scripts\activate  C:\Users\hansh\envs\AngD001\backend\.venv\Scripts\activate  REM check: (.venv) C:\Users\hansh\envs\AngD001>  pip install pip-tools black  pip install --upgrade pip-tools black cd C:\Users\%username%\envs\AngD001\backend\  .venv\Scripts\activate  pip install django==5.0.2 pyyaml uritemplate markdown pygments django-filter django-guardian psycopg2 djangocms\_text\_ckeditor django-countries[pyuca] psycopg2 django-sekizai  djangorestframework django-countries[pyuca]  pip install djangocms-text-ckeditor django-filer djangocms-versioning djangocms-alias djangocms\_admin\_style  pip install django-cms django-admin startproject MP50 --template <https://github.com/django-cms/cms-template/archive/4.1.tar.gz>  pip freeze >requirements.in  pip-compile **--no-strip-extras**  python manage.py makemigrations  python manage.py migrate  python manage**.**py cms check  REM check: if all packages are installed |

Check python version, I use python 3.12.2. I had error that cms-cmsplugin was not available. That is resolved by migration of db models.

# Check runserver

|  |
| --- |
| cd C:\Users\%username%\envs\AngD001\backend\  .venv\Scripts\activate  cd mp50  python manage.py createsuperuser  REM to check I use simple hansh [hansh@gmail.com](mailto:hansh@gmail.com) Admin1234!@# Admin1234!@#  python manage.py runserver  REM check: [django CMS](http://127.0.0.1:8000/en/) == <http://127.0.0.1:8000/en/> & [**Tutorials - django cms 4.1.0 documentation (django-cms.org)**](https://docs.django-cms.org/en/latest/introduction/index.html) |

# Chapeau!

# 

# Create a Desktop-MP-Starter-Icon and Starter .venv in VScode

|  |  |
| --- | --- |
| **FOR USERS**: Explanation of using .bat files. in de .venv folder we have startvenvserverINVSCODE.bat to place on your desktop. Use the icon as well (added to your codes). This .bat opens also the browser.  **FOR DEVELOPERS**: The second startvenserver startvenv.bat starts in the VSCode Terminal the .venv and brings you in the MP50 folder for programming. |  |

# Use the second .bat code in .vscode folder to start .venv

|  |  |
| --- | --- |
| REM startvenvserver.bat  Rem this server is used to start (1) .venv VE (2) Open VScode en (3) runserver  cd /d c:\Users\%username%\envs\ANgD001\backend\call .venv\Scripts\activate.bat  cd MP50\ start cmd /k code . # start cmd /k python manage.py runserver  start cmd /k python manage.py runserver |  |

|  |  |
| --- | --- |
| cd /d c:\Users\%username%\envs\ANgD001\backend\  call .venv\Scripts\activate.bat  cd MP50\ |  |

Before we continue, we create a simple cloning process from github. View AngD002.

# Black - Knox - Environ - Angular17

* To run Black just type in the root black . with on purpose “ .”. I installed black, view [psf/black: The uncompromising Python code formatter (github.com).](https://github.com/psf/black)
* Test the cms-code by python manage**.**py cms check. Running python manage.py runserver is a good test of code as well.

|  |
| --- |
| cd C:\Users\hansh\envs\AngD001\  black .  cd C:\Users\hansh\envs\AngD001\backend\MP50  python manage.py cms check  python manage.py test  python manage.py runserver |

1. Unit Tests: Writing unit tests for your Django-CMS project is essential for ensuring the functionality of individual components. You can create test cases using Django's testing framework and run them with:

python manage.py test

1. Integration Tests: In addition to unit tests, consider writing integration tests to validate the interaction between different parts of your Django-CMS application. Integration tests help ensure that all components work together as expected.
2. Manual Testing: While automated tests are valuable, manual testing is also crucial for verifying the user experience and overall functionality of your Django-CMS project. Test different features, pages, and interactions manually to catch any issues that automated tests may miss.
3. Code Review: Collaborate with team members or peers to review your code changes before testing. Code reviews help identify potential bugs, improve code quality, and ensure adherence to best practices.

By incorporating these testing practices into your workflow, you can effectively evaluate and validate your Django-CMS project codebase, ensuring its reliability, functionality, and adherence to coding standards before deployment or further development.

* Knox
* Environ
* Angular17

While Django-CMS provides a robust content management system for building web applications with Django, integrating Angular17 into your project can offer a powerful frontend framework for dynamic and interactive user interfaces. Here are some considerations and steps to replace Django-CMS frontend with Angular17:

1. Separation of Concerns: Django-CMS primarily handles backend logic and content management, while Angular17 focuses on frontend development. By replacing the Django-CMS frontend with Angular17, you can leverage Angular's capabilities for building single-page applications (SPAs) and enhancing user experience.
2. API Integration: Since Angular17 operates independently from Django-CMS, you will need to set up APIs or endpoints in your Django backend to communicate with the Angular frontend. This allows data exchange between the two frameworks.
3. Angular Project Setup: Create a new Angular project within your existing Django project structure or in a separate directory. You can use Angular CLI to generate components, services, and modules for your frontend application.
4. Frontend Routing: Define routing in your Angular application to handle different views and components. This will replace the URL routing functionality provided by Django-CMS.
5. Replace Templates: Instead of using Django templates for rendering frontend components, Angular uses its own templating system based on HTML and TypeScript. You will need to migrate your existing frontend components to Angular components.
6. Styling: Angular17 allows you to use CSS frameworks like Bootstrap or Material Design for styling your frontend components. You can customize the look and feel of your application using these frameworks.
7. Build Process: Configure the build process for your Angular application to generate optimized production-ready code. You can use tools like Webpack or Angular CLI for bundling and minification.
8. Deployment: Once you have integrated Angular17 into your project, deploy the frontend application separately from the Django backend. You can host the Angular app on a different server or serve it through a CDN.

By following these steps and considering the differences between Django-CMS and Angular17, you can successfully replace the Django-CMS frontend with Angular17 to create a modern and dynamic user interface for your web application

A screenshot of a computer program

Description automatically generated

When installing and working with packages like Knox, Environ, and Angular17 in your Django-CMS project, it's important to follow best practices for integration, configuration, and testing. Here are some suggestions for each of these packages:

1. Knox:
   * Installation: Knox is a popular authentication library for Django REST framework. You can install Knox using pip:

pip install django-rest-knox

* + Configuration: Follow the setup instructions provided in the Knox documentation to integrate token-based authentication into your Django project.
  + Usage: Utilize Knox to handle user authentication and token management in your Django-CMS project. Refer to the Knox documentation for detailed usage guidelines.

1. Environ:
   * Installation: Environ is a package for managing environment variables in Python projects. Install Environ using pip:

pip install environ

* + Configuration: Use Environ to manage sensitive information like API keys, database credentials, and other configuration settings in your Django-CMS project. Store environment variables in a .env file or system environment variables.
  + Usage: Access environment variables using Environ in your Django settings.py file or other parts of your project where configuration values are needed.

1. Angular17:
   * Installation: Angular17 is a powerful frontend framework for building dynamic web applications. Install Angular CLI to create and manage Angular projects:

npm install -g @angular/cli

* + Integration: Create a new Angular project within your Django-CMS project structure or as a standalone frontend application. Configure API endpoints to communicate with your Django backend.
  + Development: Use Angular17 features like components, services, and modules to build interactive user interfaces for your Django-CMS project.

For testing strategies with these packages, consider the following approaches:

1. Unit Testing: Write unit tests for backend components that interact with Knox authentication or Environ configuration. Use tools like Django's TestCase class for testing Django views, models, and serializers.
2. Integration Testing: Test the integration between Knox authentication and your Django REST API endpoints. Verify that token-based authentication works correctly and securely.
3. End-to-End Testing: Consider implementing end-to-end tests using tools like Selenium or Cypress to test the functionality of your Angular17 frontend components interacting with the Django backend.
4. Security Testing: Perform security testing to ensure that sensitive information managed by Environ is properly secured and not exposed in production environments.

By following these suggestions for installing and working with Knox, Environ, and Angular17, as well as implementing specific testing strategies tailored to each package, you can enhance the functionality, security, and reliability of your Django-CMS project while ensuring seamless integration with frontend technologies like Angular17.

# Python manage.py cms check, run until all are okay!

|  |
| --- |
| **(.venv) C:\Users\hansh\envs\AngD001\backend\MP50>python manage.py cms check**  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Checking django CMS 4.1.0 installation  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  Sekizai  =======  - Sekizai is installed [OK]  - Sekizai template context processor is not installed, could not find 'sekizai.context\_processors.sekizai' in TEMPLATES option context\_processors [ERROR]  Sekizai configuration has errors [ERROR]  Internationalization  ====================  - New style CMS\_LANGUAGES [OK]  Middlewares  ===========  - django.middleware.locale.LocaleMiddleware middleware must be in MIDDLEWARE\_CLASSES [ERROR]  - cms.middleware.user.CurrentUserMiddleware middleware must be in MIDDLEWARE\_CLASSES [ERROR]  - cms.middleware.page.CurrentPageMiddleware middleware must be in MIDDLEWARE\_CLASSES [ERROR]  - cms.middleware.toolbar.ToolbarMiddleware middleware must be in MIDDLEWARE\_CLASSES [ERROR]  - cms.middleware.language.LanguageCookieMiddleware middleware must be in MIDDLEWARE\_CLASSES [ERROR]  Context processors  ==================  - cms.context\_processors.cms\_settings context processor must be in TEMPLATES option context\_processors [ERROR]  - django.template.context\_processors.i18n context processor must be in TEMPLATES option context\_processors [ERROR]  Plugin instances  ================  - Plugin instances of 0 types found in the database [OK]  The plugins in your database are in good order [OK]  Presence of "copy\_relations"  ============================  All plugins and page/page content extensions have "copy\_relations" method if needed. [OK]  OVERALL RESULTS  ===============  9 errors!  5 checks successful!  Please check the errors above  CommandError: |

# 

# 

# ANGCMS000 Django-CMS Step 1 + Frontend Demo

View django-cms documentation and lessons. [git](https://github.com/django-cms/django-cms), [home](https://www.django-cms.org/en/), [tutorials-How to’s](https://user-guide.django-cms.org/en/latest/). Open VScode and [Ctrl+Shift+`] is button under the sign below Esc at the left upper corner of keyboard. I open at C:\Users\hansh> In the codes I use C:\Users\%UserName%\ or >. Open VSCode and new Terminal.

|  |
| --- |
| py manage.py migrate  cd C:\Users\hansh\envs\ANGCMS000\backend\MP50\  **python manage.py runserver**  pip install git+https://github.com/psf/black  cd C:\Users\hansh\envs\ANGCMS000\backend\  black MP50 |

pip install djangocms\_frontend ?????? <http://127.0.0.1:8000/en/admin/login/?next=/en/> & <http://127.0.0.1:8000/en/admin/>

Close the server by [Ctrl+C] and the VE by *deactivate*. If not succesfull remove all folders and try again. <http://127.0.0.1:8000/> and <http://127.0.0.1:8000/en/admin/> en relates to the default language.

# Create in backend file start.bat. To activate the VE.

Open VSCode, run start.bat: runserver opens website. Place start.bat on Desktop and in ANGCMS000.

|  |
| --- |
| @ECHO OFF  cmd  cd /d C:\Users\%UserName%\envs\ call code .  Rem Right-click and run the code by low end of drop down “run”  cd /d C:\Users\%UserName% \envs\ANGCMS000\.venv\Scripts\  call activate.bat  cd /d C:\Users\%UserName% \envs\ANGCMS000\backend\MP50\  call python manage.py runserver  Rem You need to wait to show the server address <http://127.0.0.1:8000/> & <http://127.0.0.1:8000/admin/>  Rem To quit the server use [Ctrl+C]  Rem Open new Terminal by [ctrl+Shift+`]  Rem Test this file by r-click and run (at end of drop down)  Node cd /d C:\Users\%UserName% \envs\ANGCMS000\frontend\ ng serve |

Save zip in folder ANGCMS000 and create PDF in there as well of this page.

|  |
| --- |
| @ECHO OFF  cmd  cd /d C:\Users\%UserName%\envs\ call code .  Rem Right-click and run the code by low end of drop down “run”  cd /d C:\Users\%UserName% \envs\ANGCMS000\.venv\Scripts\  call activate.bat  cd /d C:\Users\%UserName% \envs\ANGCMS000\backend\MP50\  call python manage.py runserver  Rem You need to wait to show the server address <http://127.0.0.1:8000/> & <http://127.0.0.1:8000/admin/>  Rem To quit the server use [Ctrl+C]  Rem Open new Terminal by [ctrl+Shift+`]  Rem Test this file by r-click and run (at end of drop down) Node  cd /d C:\Users\%UserName% \envs\ANGCMS000\frontend\ ng serve |

# ANGCMS000 Django-CMS Base. Step 2 Frontend Angular

Install Node <https://nodejs.org/dist/v20.11.1/node-v20.11.1-x64.msi>   
[Angular - Setting up the local environment and workspace](https://angular.io/guide/setup-local)

For npm you need to use Powershell!

|  |
| --- |
| cmd  deactivate  cd C:\users\%UserName%\envs\ANGCMS000  cmd  npm install -g @angular/cli  ~~Set-ExecutionPolicy -Scope Process -ExecutionPolicy RemoteSigned Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned~~  npm install -g @angular/cli  npm fund ANGCMS000  ng new frontend  REM Yes (share data?)  ~~REM ng analytics disable --global~~  CSS  Rem Choose CSS  Rem ? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No  cd frontend  ng serve --open  ~~ng analytics disable~~ |

Test port: netstat -ano | findstr :4200 Empty cache [Ctrl+Shift+delete]

SWITCH OFF VPN!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Global setting: enabled

Local setting: enabled

Effective status: enabled

⠹ Building...

Initial chunk files | Names | Raw size

polyfills.js | polyfills | 83.60 kB |

main.js | main | 22.09 kB |

styles.css | styles | 95 bytes |

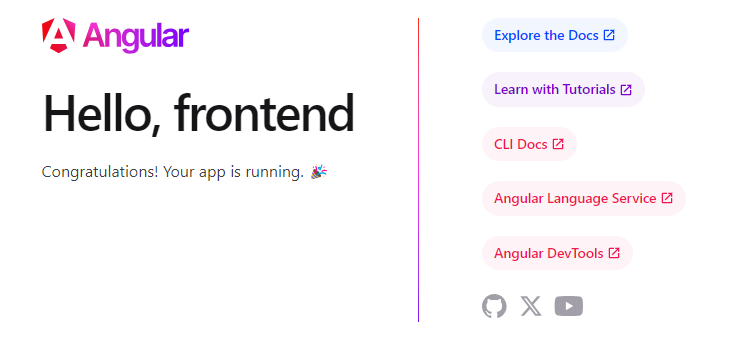
| Initial total | 105.78 kB

Application bundle generation complete. [1.219 seconds]

Watch mode enabled. Watching for file changes...

➜ Local: http://localhost:4200/

➜ press h + enter to show help



[Explore the Docs](https://angular.dev/)[Learn with Tutorials](https://angular.dev/tutorials)[CLI Docs](https://angular.dev/tools/cli)[Angular Language Service](https://angular.dev/tools/language-service)[Angular DevTools](https://angular.dev/tools/devtools) @ <https://angular.dev/>

<https://angular.dev/>

