# Django Content Management System Evaluation and Integration with Apache Airavata

Stephen Paul Adithela
Indiana University
Bloomington, Indiana
stephenpaul2727@gmail.com

Marcus Christie
Science Gateways Research Center
Bloomington, Indiana
machrist@iu.edu

Suresh Marru
Science Gateways Research Center
Bloomington, Indiana
smarru@iu.edu

Marlon Pierce
Science Gateways Research Center
Bloomington, Indiana
marpierc@iu.edu

## ABSTRACT

Apache Airavata is an open-source software framework that enables scientific researchers to compose, manage, execute and monitor large-scale applications and workflows on distributed computing resources. Airavata is currently leveraged by many science gateways to perform computations on shared clusters. Currently, Gateway Administrators managing content on their websites will require the assistance of the Airavata Developer Team to make the slightest of change to their website. This paper will overcome this challenge by presenting the benefits of integrating a content management system. It will also briefly evaluate various options available for choosing a Content Management Platform which complies with the Airavata Architecture Standards. This feature will enable researchers with minimal web design knowledge to easily manage content across their gateway. It is also poised to drastically increase the productivity of the Airavata developer team and the gateway administrators.

## CCS CONCEPTS

• **Software and its engineering → Integrated and visual development environments**; Object oriented frameworks;

## KEYWORDS

ACM proceedings, text tagging, Apache Airavata, Science Gateway, Wagtail CMS, Django Framework

**ACM Reference Format:**
Stephen Paul Adithela, Marcus Christie, Suresh Marru, and Marlon Pierce. 2018. Django Content Management System Evaluation and Integration with Apache Airavata. In *PEARC '18: Practice and Experience in Advanced Research Computing, July 22–26, 2018, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3219104.3229272

## 1 INTRODUCTION

Apache Airavata Django Portal is a web portal for Apache Airavata [4] which is currently provides user interface for gateway administrators to create and manage resources for their applications. It is based on the popular Python-based web framework called Django. This portal is now equipped with a Content Management System (CMS) which internally uses Wagtail CMS. With the integration of the CMS, the portal is now capable of providing gateway administrators, the ability to manage content on their own without any assistance from the Airavata Developer team. It will also provide them the ability to deploy changes on their own and the ability to style the content on their website without writing a single piece of code. We will discuss why the CMS based approach will help researchers who are leveraging Apache Airavata and how we arrived at choosing a CMS for Apache Airavata.

## 2 PROBLEM STATEMENT

Apache Airavata is currently being used by many science gateways as a middleware between their job submissions and the Grid systems. One of the Airavata components is the Airavata Django Portal (User Interface) which will be used by researchers to create, submit and manage jobs.

Researchers who leverage Airavata Django Portal request a unique theme for their gateway which adheres to their organization's web standards. To solve this problem, Airavata Developer team designed unique themes for each gateway with the help of gateway administrators. The following are the problems that are still left unanswered by this approach.

(1) Gateway administrators with no knowledge of web design won't be able to make changes and maintain their website on their own.
(2) Gateway administrators with no knowledge of web design will delegate the task of making changes to the Airavata Developer team.
(3) Each Gateway theme is maintained in a separate Github repository which will add additional management responsibilities for the Airavata Developer team.

## 3 CONTENT MANAGEMENT SYSTEM

A content management system[6] is an application which supports creation and modification of digital content. It typically supports multiple users being able to manage content in a collaborative environment.

CMS features vary widely. CMS functionalities include format management, history editing, version control, indexing, search, and retrieval. By their nature, content management systems support the separation of content and presentation. A typical CMS consists of two major components:

(1) **CMA**: A CMA is a content management application which is a front-end user interface which lets users even with limited knowledge of web applications to add, modify and remove content from a website without the intervention of the software creators.
(2) **CDA**: A CDA is a content delivery application that compiles the information modified by the user and updates the web application.

Not only does a Content Management System(CMS) solve all the issues with the current Airavata User Interface, It will also provide many advantages as shown below:

- Ability to create a unified look and feel
- Reusable components
- Version control
- Easy Deployment
- Reduced need to code from scratch
- Secure Permission Management
- SEO Friendly URLs
- Support for various databases
- Media Management

Although a Content Management System (CMS) has its advantages, using it requires the user to have a good understanding of how the it works. A limited amount of training is required for the content editors before they are given access to the system itself.

## 4 DJANGO ARCHITECTURE

Airavata Djagno Portal is currently utilizing Python-based web framework called Django which encourages rapid development. Django supports the MVT Pattern which is similar to the very familiar MVC Pattern.

**MVC Pattern:** [3] MVC Pattern is a common architectural pattern followed in designing software applications. This pattern divides the modules into three parts. They are model, view, and controller. This is done to separate internal representations with the information that is presented to the users.

**MVT Pattern:** [3] The model-view-template pattern is slightly different from the MVC Pattern. Here the controller part is taken care of by the framework itself. This leaves us with the ability to configure models and templates. The template is an HTML file mixed with Django Template Language (DTL). The models are the Django representations of tables inside the database.

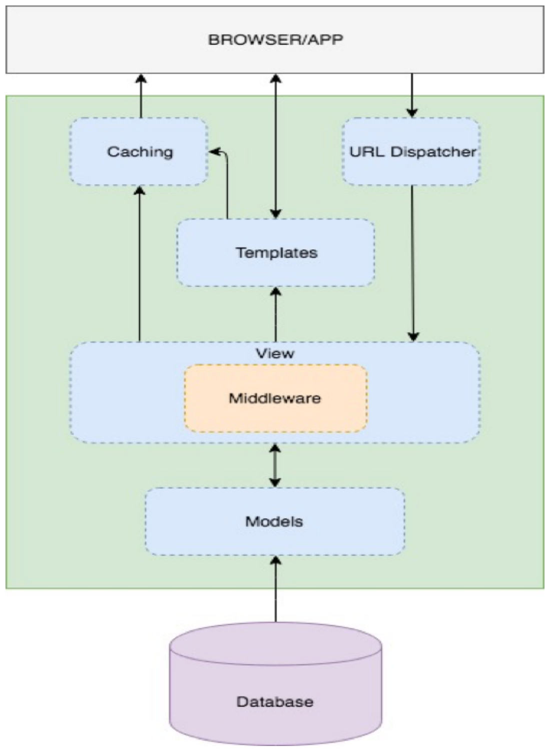

**Figure 1: A Normal Django Application's internal Architecture Diagram**

[1] Django is open source and free. Out of the box, it provides with the application structure and files everything pre-configured. So, the developer only needs to worry about the important aspects of the software he is making.

Django framework is structured such that it is very modular and versatile. A new module can be easily added into the Django application in the form of an app. Each app inside the Django application is modularized to perform a particular task. This versatility and extensibility of Django makes it a perfect framework for implementing a content management system.

## 5 CHOOSING A CMS

Many websites around the world are using a Content Management System to facilitate editors of their website easily manage content without going through the meticulous and arduous routines of Software Development. Some of the most used Content Management Systems (CMS) around the world are as follows:

(1) WordPress
(2) Drupal
(3) Joomla
(4) TextPattern

Although these content management systems are versatile and are used by many major companies, they are not suitable for the current Airavata Django Portal Architecture. The following are some reasons why these content management systems would be bad choices for the current Airavata Django Portal.

(1) **Integration Problem**: Airavata Django Portal is just missing CMS capabilities. These content management systems are written in various programming languages and don't have a way of integration with an existing Django Application.
(2) **BlackBox**: If the developer want to change the underlying code of these content management systems according to his requirements. Due to the large codebase understanding underlying implementations can be an arduous task.
(3) **Database Constraints**: Some of these content management systems are restricted to using a particular database. Integrating with the existing Airavata Django Portal will make the application use two databases which can carry a lot of overhead.
(4) **Language Constraints**: Python's just-in-time compiler makes it faster to compile and run scripts when compared to PHP. Django Framework by itself checks for security vulnerabilites like SQL Injection, CSRF, etc. where the developer is given the task of making these security advancements on PHP which makes Django Python more secure.
(5) **Heavyweight**: By default, these CMS's come all the features which are not required for the application under consideration. This will result in bloated code and will occupy a huge amount of disk space.
(6) **Additional Dependencies**: Because these frameworks are built using different languages and different software dependencies, the dependencies list which already includes the dependencies for the Airavata Django Portal will increase even more which in turn makes the application heavy and puts a lot of load on the hosted web server.

From the above analysis, it is clear that if we leverage an existing CMS which itself is a Django application, we can reduce a lot of overhead. Currently, there are two powerful well-maintained open-source content management systems implemented using Django framework, they are **Django CMS** & **Wagtail CMS**.

## 5.1 Django CMS

Django CMS[7] is a free and open source content management system platform for publishing content on the World Wide Web and intranets. It is written in Django and Python. Django CMS released in 2007. Currently, it is backed by Divio AG a Swiss software company. Django CMS is a mature content management system which can provide CMS capabilities to complex web apps built using Django. It also has great community support both for developers and users. The project is open sourced and under active development. Django CMS is very versatile, easily upgradeable, extendable and secure.

## 5.2 Wagtail CMS

Wagtail CMS[8] is a free and open source content management system written in Python. It is currently maintained by a team of open source contributors. Wagtail is still a developing project which started back in 2014. It is not as mature as Django CMS, but it has gained traction very quickly and development process is also faster. It offers support for all the content management functionalities. It also provides support for some important features like Revision Control which are not present in Django CMS

The Following is a table which will differentiate between the two CMS platforms based on the following categories:[5]

| **Features** | **Content Management Systems** | |
|---|---|---|
| | **Django CMS** | **Wagtail CMS** |
| WYSIWYG Editor | Yes | Yes |
| Revision Management | No | Yes |
| App Integration | Yes | Yes |
| Media Asset Management | Yes | Yes |
| Pypi Installable | Yes | Yes |
| 508 Compliant[2] | Yes | Yes |
| Granular Permissions | Yes | Yes |
| Multi-Device Support | Partial Yes | Yes |
| Plugin Support | Yes | Yes |
| Multi-site Support | Yes | Yes |
| Tagging | No | Yes |
| Migrations | Yes | Yes |
| Internationalization | Yes | Yes |
| SEO Friendly | Yes | Yes |
| Community Support | Great | Good |

**Table 1: Differences between wagtail and Django CMS based on the evaluation criteria.**

From the above comparison, both Wagtail and Django CMS are pretty promising and integrate well with the existing Airavata Django Portal. Both of them offer all the features of a typical content management system.

Django CMS excels over Wagtail on the following aspects:

(1) None to little knowledge of Django Framework is required by the developers for implementing Django CMS as opposed to Wagtail which requires the developer to have good knowledge of Django.
(2) Configuring and Initializing Django CMS is effortless compared to configuring Wagtail CMS.
(3) Integrating an existing Django application is straightforward and creating custom plugins is uncomplicated.

But Wagtail stands out as the best future-proof option for implementing CMS functionalities into the existing Airavata Django Portal. The following are the main reasons:

(1) Configuring Wagtail will result in one additional dependency whereas configuring Django CMS requires multiple additional dependencies. This makes Django CMS hard to

manage with new updates and the complexity will keep increasing in the future.

(2) Django CMS does not have Revision Control and Version management integrated. It also doesnâĂŹt have third-party plugins which can perform this essential task. On the other hand, Wagtail has this feature inbuilt.
(3) Django CMS is a black box where internal implementations cannot be customized easily. While wagtail offers customization to its core models. In this sense, Django CMS performs all the essential tasks but might be limited in scope as the software progress towards the future.
(4) Django CMS provides an admin panel which is integrated with the default Django Admin. Integration will be difficult for a Django application which already has a custom admin application which is the case for Airavata Django Portal.
(5) Wagtail is database-centric and doesn't disturb the file or the folder structure of the Airavata Django Portal code base.

## 6 AIRAVATA DJANGO PORTAL WITH WAGTAIL

Airavata Django Portal is now configured to provide content management functionalities to the researchers and Gateway administrators with the integration of Wagtail. Wagtail comes with a login interface for accessing the CMS dashboard.

Airavata Django Portal uses Keycloak which is an open-source software product to allow single sign-on with identity management and access management. The Wagtail Login functionality is configured to authenticate by re-using the existing credentials in the Keycloak instead of creating new credentials again for each gateway researcher and maintaining a separate database. 'Admin' Privilege is the highest access level for a CMS user. An admin will be able to create/edit/update/delete any page in the database. Other groups of CMS users include 'Editors' and 'Moderators'. Moderators will have higher access rights than an editor. An Editor will responsible for editing content and making changes on the website. Moderator will have the right to approve those changes to be published live or suggest any modifications. Any Authorized user with Access Level 'Admin', 'Moderator' or an 'Editor' can access the CMS Dashboard.

The CMS dashboard hosts the content which reflects Airavata Django Portal live website. An authorized CMS user can perform some/all of the following actions based on his access level in the dashboard:

(1) Create, Update, Delete and Modify a Page
(2) Publish/Unpublish a page
(3) Add new users, new groups of CMS users
(4) Add Custom CSS
(5) Add/Delete Images
(6) Add/Delete Documents
(7) Update Site Details
(8) Revert a page back to previously published version
(9) Add permissions to a Page/Image/Document.

With the integration of the CMS, all the Science Gateways which leverage Apache Airavata Middleware are converted from using themes in separate GitHub Repositories to use the CMS directly to style their website. This integration has rendered the process of hosting website themes in separate repositories obsolete. Sufficient amount of documentation is provided to the Gateway administrators to make them knowledgeable of using CMS.

The revamped architecture will promote easy management of the Airavata Django Portal for the Airavata Developer team. Python Scripts are created which can load a particular gateway theme with one single command. For example, **python manage.py load_seagrid_data** will load the Seagrid Gateway website on the same host environment without again having to configure the environment for the other website. This makes the process of switching between gateways easy and intuitive. This functionality will make the current Airavata Django Portal Architecture very simple, powerful and versatile.

## 7 CONCLUSION

Integrating a Content Management System into the existing Airavata Django Portal has proved to be far more intuitive, simple and easy to manage than the previous versions. Gateway administrators and researchers who generally don't have much knowledge of web development can now add images, documents and static pages to their website with full control.

Airavata Developer Team will not be burdened with the arduous tasks of making changes to the gateway portals upon request and deploying them to production. This process can now be accomplished by the Gateway administrators/researchers themselves which will make the overall deployment process faster and simple. Content Editors for each gateway can now effortlessly modernize their website or build upon an existing gateway theme by loading the desired gateway and building on top of it.

Integrating CMS into the existing Airavata Django Portal has simplified the Software Development Life-Cycle of the Apache Airavata powered Science Gateways. It also enhanced the productivity of Airavata Developer team and the gateway administrators.

## REFERENCES

[1] 2013. The web framework for perfectionists with deadlines | Django. https://www.djangoproject.com/
[2] 2018. 508 Compliance in the United States of America. https://www.section508.gov/manage/laws-and-policies
[3] 2018. Django Overview. https://www.tutorialspoint.com/django/django_overview.htm
[4] Suresh Marru, Lahiru Gunathilake, Chathura Herath, Patanachai Tangchaisin, Marlon Pierce, Chris Mattmann, Raminder Singh, Thilina Gunarathne, Eran Chinthaka, Ross Gardler, et al. 2011. A high-level approach to computer document formatting. In *Proceedings of the 2011 ACM workshop on Gateway computing environments*. ACM, 21–28.
[5] Daniel Roy-Greenfeld and Audrey Roy-Greenfeld. 2018. Django Content Management Systems evaluation criteria and differences. https://djangopackages.org/grids/g/cms/
[6] Wikipedia. 2018. Content Management System Definition and Features. https://www.wikipedia.org/
[7] Wikipedia. 2018. Django CMS Introduction. https://en.wikipedia.org/wiki/Django_CMS
[8] Wikipedia. 2018. Wagtail CMS Introduction. https://en.wikipedia.org/wiki/Wagtail_(CMS)