How the portal's importation, generation and transformation services work

Last updated by | Joyclyn Vincent | 10 Jun 2024 at 11:33 GMT+10

The portal's importation, generation and transformation services leverage Azure functions, factories, and GitHub integrations.
This system ensures efficient content reception, transformation, validation, and publication across multiple platforms.

System overview

The system starts with an Azure function and involves several key components:

- Azure Function for entry point.
- Factories for service creation.
- Services for specific content handling tasks.
- GitHub for storage and automation.
- Validation steps to ensure content integrity.

Data flow

1. Entry Point: Azure function

ContentService:

- Acts as the main entry point for incoming content.
- Interacts with Azure Cosmos DB for configuration data.

2. Content factories

Factories are responsible for creating the necessary service instances based on the type of content received:

- ContentReceiverFactory: Generates services for receiving content.
- ContentTransformationFactory: Generates services for transforming content.
- ContentGenerationFactory: Generates services for creating new content.

3. Service processing

Once the content factories produce the necessary services, the content undergoes specific handling by these services:

- DevopsWikiReceiverService: Receives content from DevOps Wiki.
- GithubReceiverService: Receives content from a GitHub repository.
- ContentmanagerReceiverService: Receives content from Content Manager.
- ContentGenerationService: Generates new content based on the received data.

4. GitHub integration

GitHub plays a pivotal role in storing content and automating workflows:

- GitHub repository: Serves as a storage hub for the received content.

- GitHub action: Automates workflows triggered by repository events, including validation steps before publishing to the final destination.

5. Validation steps

To ensure the content's integrity and authenticity, the system incorporates validation steps:

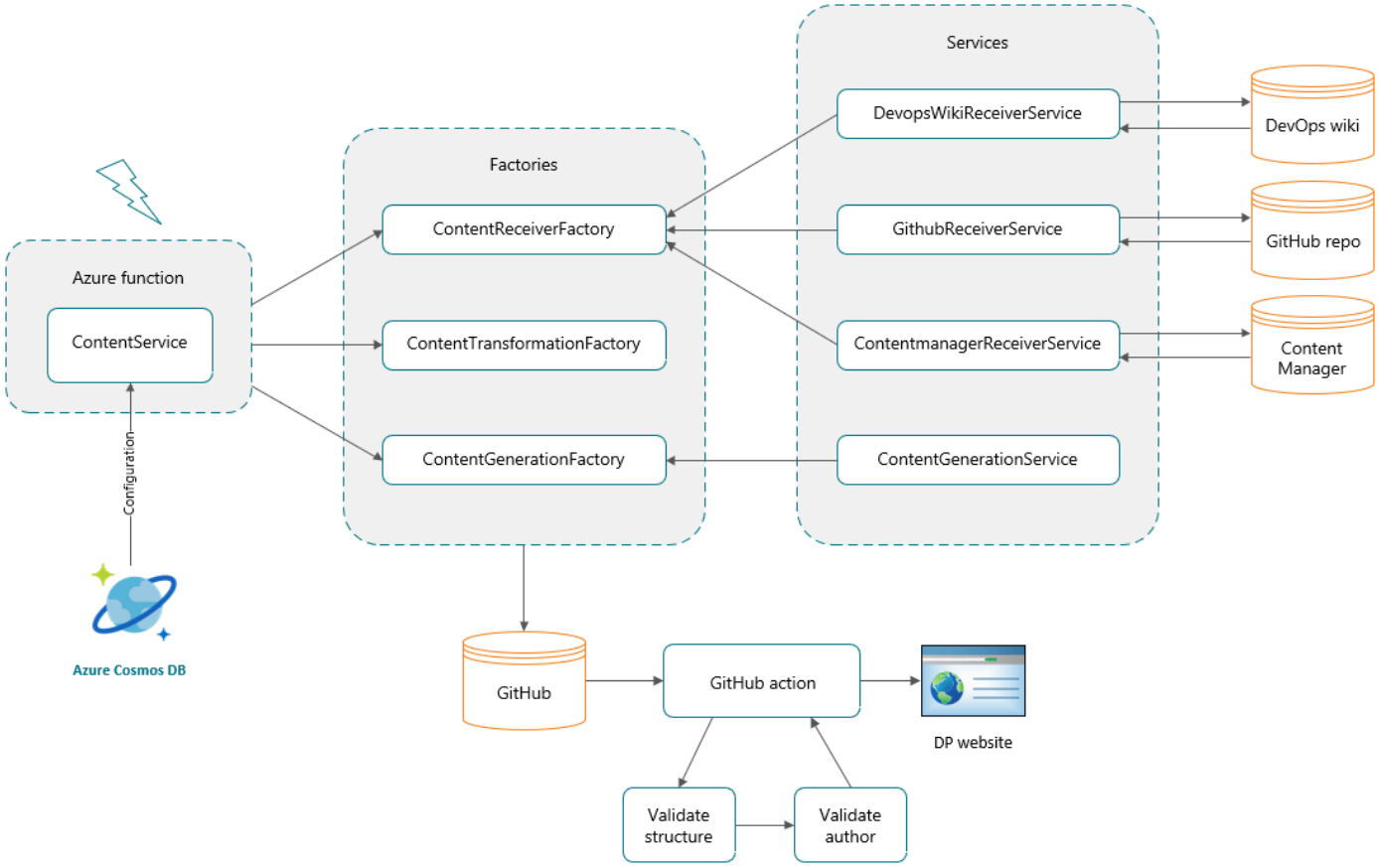
- Validate structure: Ensures the content adheres to the required format.
- Validate author: Verifies the authenticity and validity of the content author.

6. Publishing the content

After successful validation, the content is published to the Developer Portal site.

Configuration

Azure Cosmos DB: Stores configuration data for the system.



This system showcases a robust architecture that efficiently handles content from reception to publication. By integrating Azure functions, various service factories, and GitHub automation, the system ensures that content is processed, validated, and published seamlessly. This setup not only maintains content integrity but also optimises the workflow through automation and validation steps.

This post was written in conjunction with ChatGPT.