Contents

[MVP Blog Search Feature 1](#_Toc379385262)

[User Stories 2](#_Toc12121524)

[User Guide 2](#_Toc1335446366)

[Adding Publication Urls 2](#_Toc279255529)

[ Indexing publications 2](#_Toc462967235)

[Searching publications 3](#_Toc1355579068)

[Design & Implementation 3](#_Toc830406356)

[Search Service Implementation 4](#_Toc1580846696)

[Component Design 7](#_Toc627494108)

[Setting up solution locally 8](#_Toc1626049653)

# MVP Blog Search Feature

As many users referring to the MVP site could be interested to search and browse the publications of Sitecore MVPs, we decided to extend the MVP site by adding a search feature so that users can search and read through publications easily

## User Stories

* *As a user I like to have a search box on the MVP site so that I can search the publications of MVP holders based on the contents*
* *As a user I like to search and find all publications of MVP holders by entering their name in the search box*
* *As a user I want to search for information related to Sitecore from a list of reliable posts.*
* *As a content editor, I like to add the urls of the publications in MVP data items in the content tree so that the publications at these urls can be searched*

# User Guide

## Adding Publication Urls

A list of publication urls can be added to the People items in the content editor. To add the urls, navigate to the right MVP data item and populate BlogPost field with the urls of publications.

Graphical user interface, application

Description automatically generated with medium confidenceText

Description automatically generated with medium confidence

## Indexing publications

*MVPBlogsReader* scheduled task runs periodically to read the People items and extract the urls from BlogPost field. Reads the contents of the blog article and uploads them as html documents to a storage account to be indexed and searched

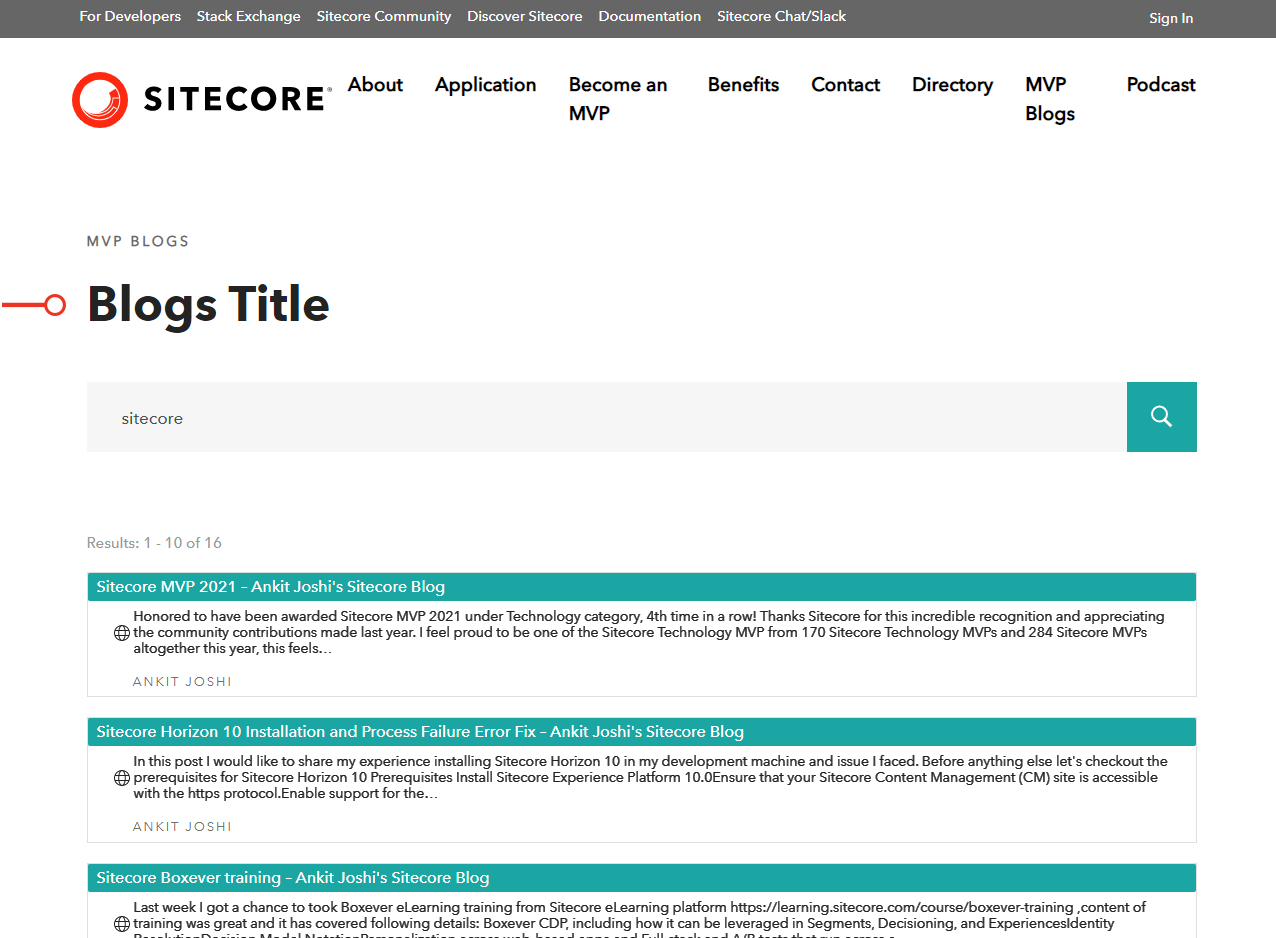
Graphical user interface, text, application, chat or text message

Description automatically generated

## Searching publications

Navigate to “MVP Blogs” and type the keywords in the search box to see the MVP publication results.

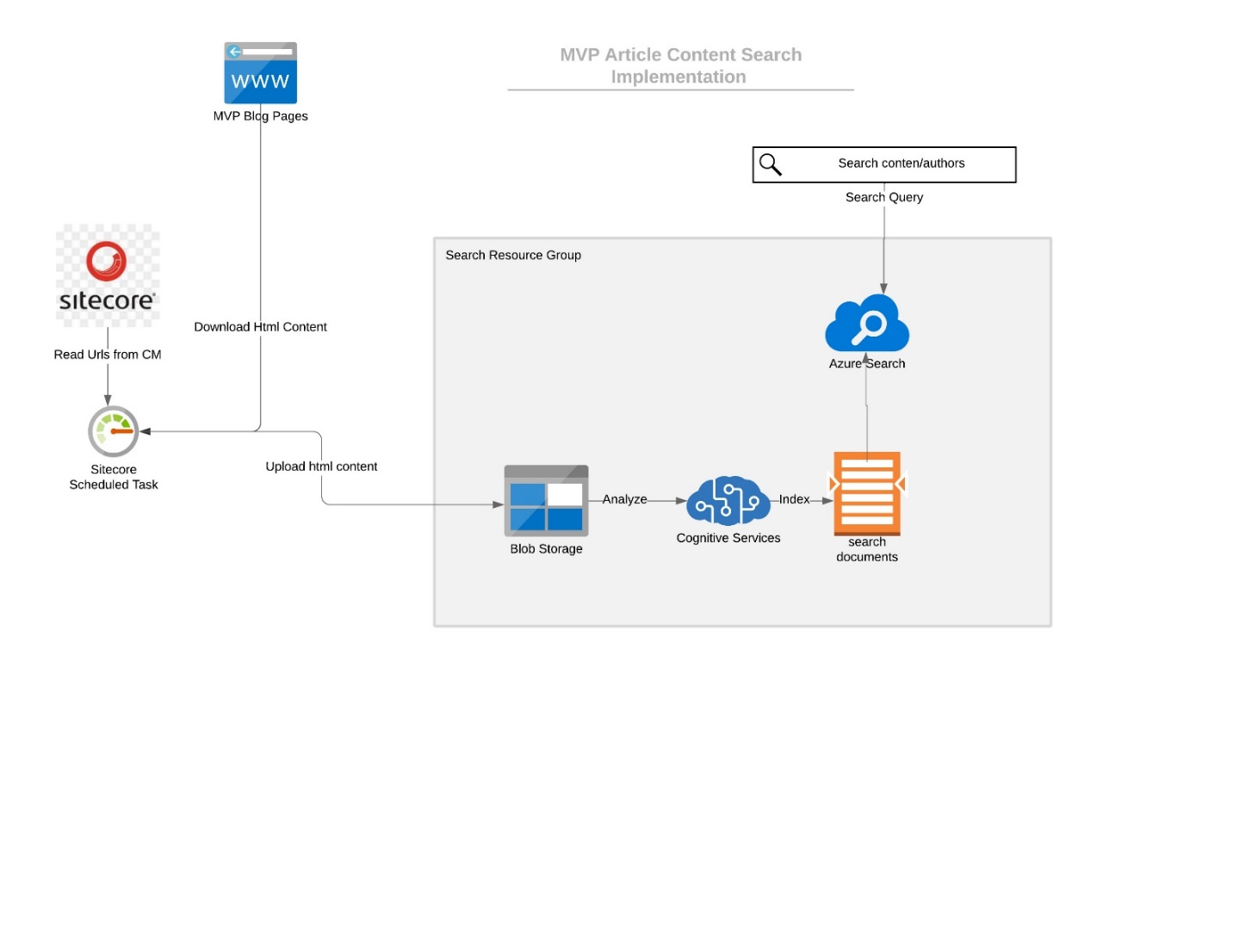
Click on each result item to navigate to the article page



# Design & Implementation

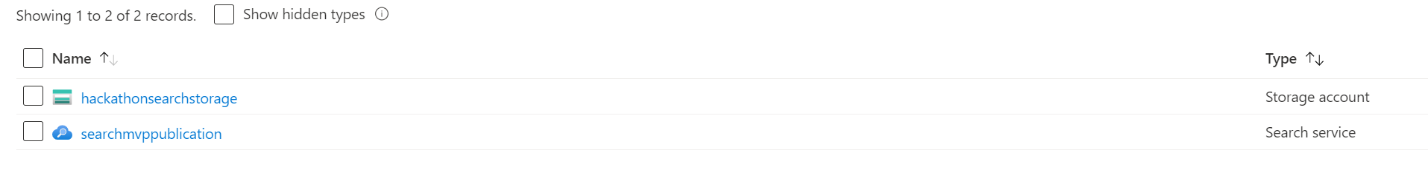
## Search Service Implementation

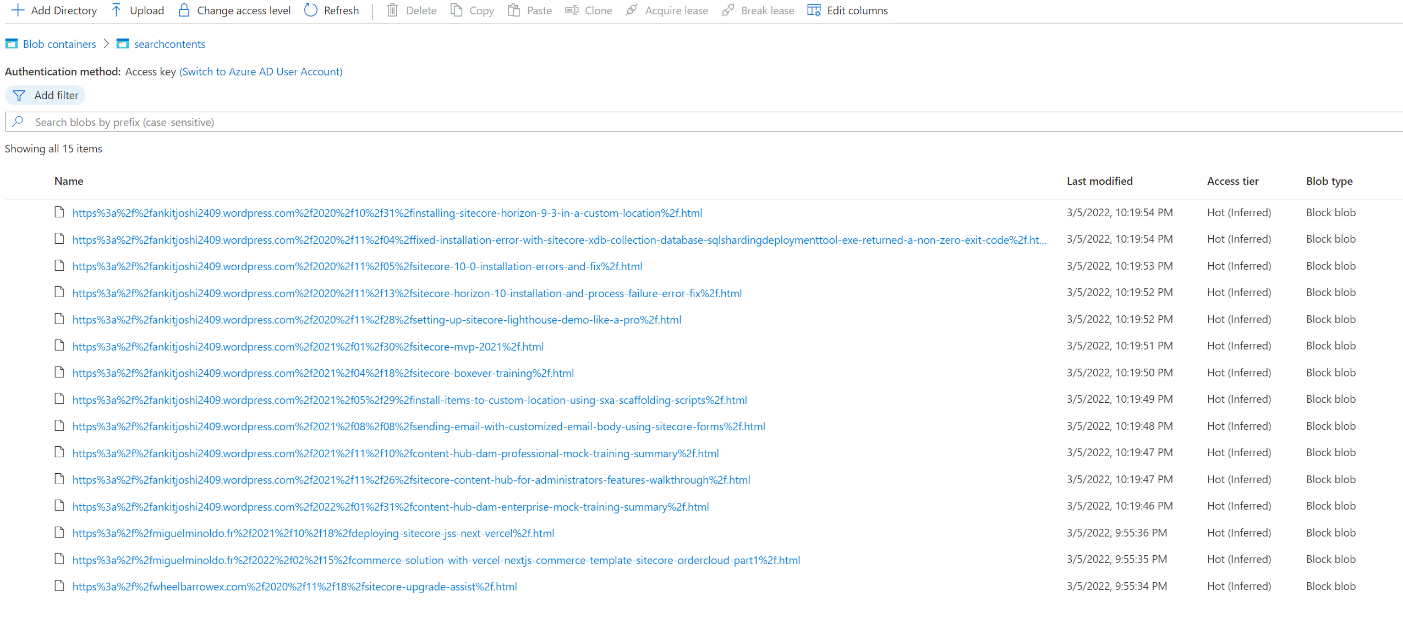
To search among the contents of the publications of the MVPs, Azure Cognitive Search was used which enables deeper analysis of the contents and extracting information using AI enrichment.



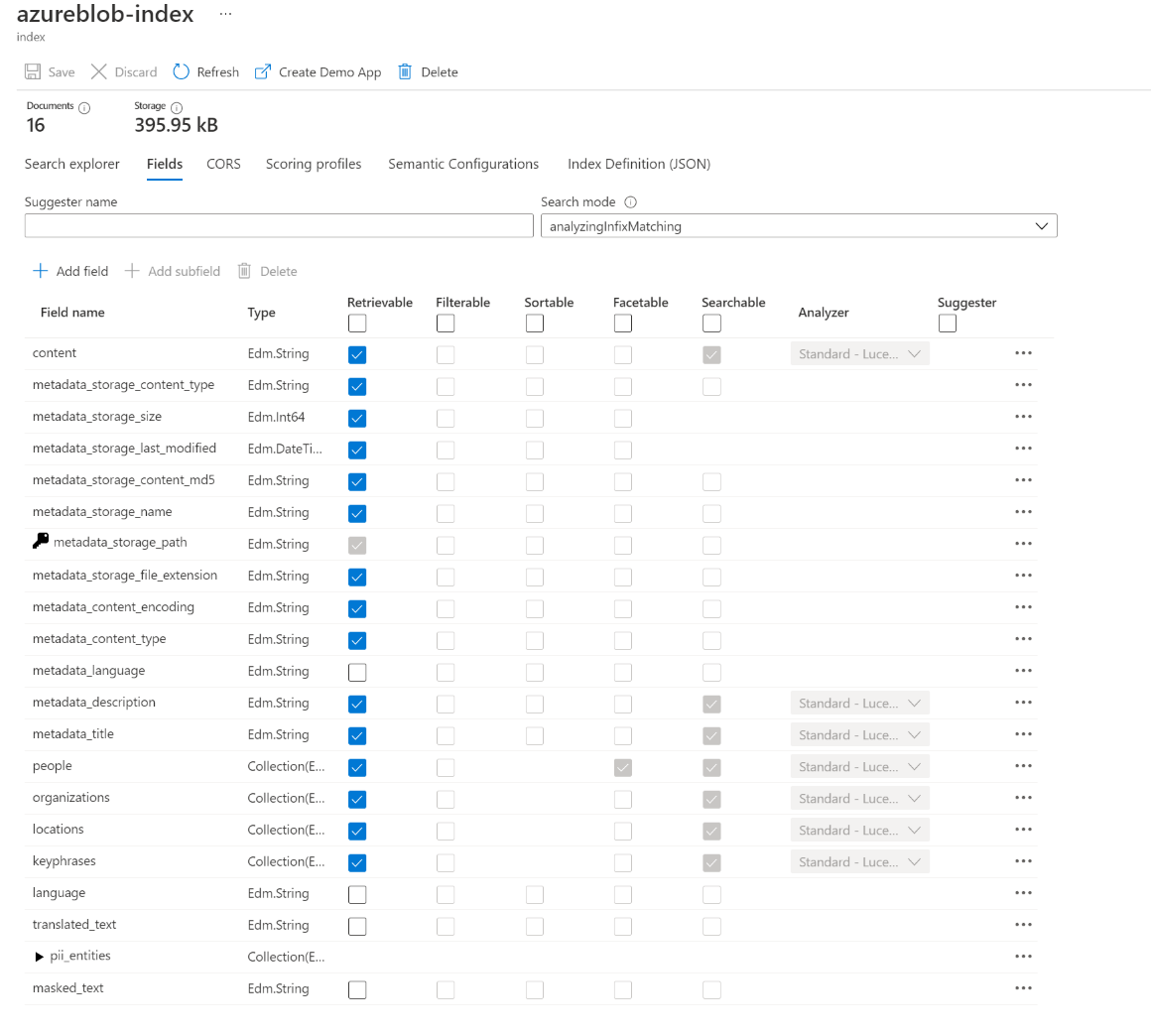
The Search Service implementation includes:

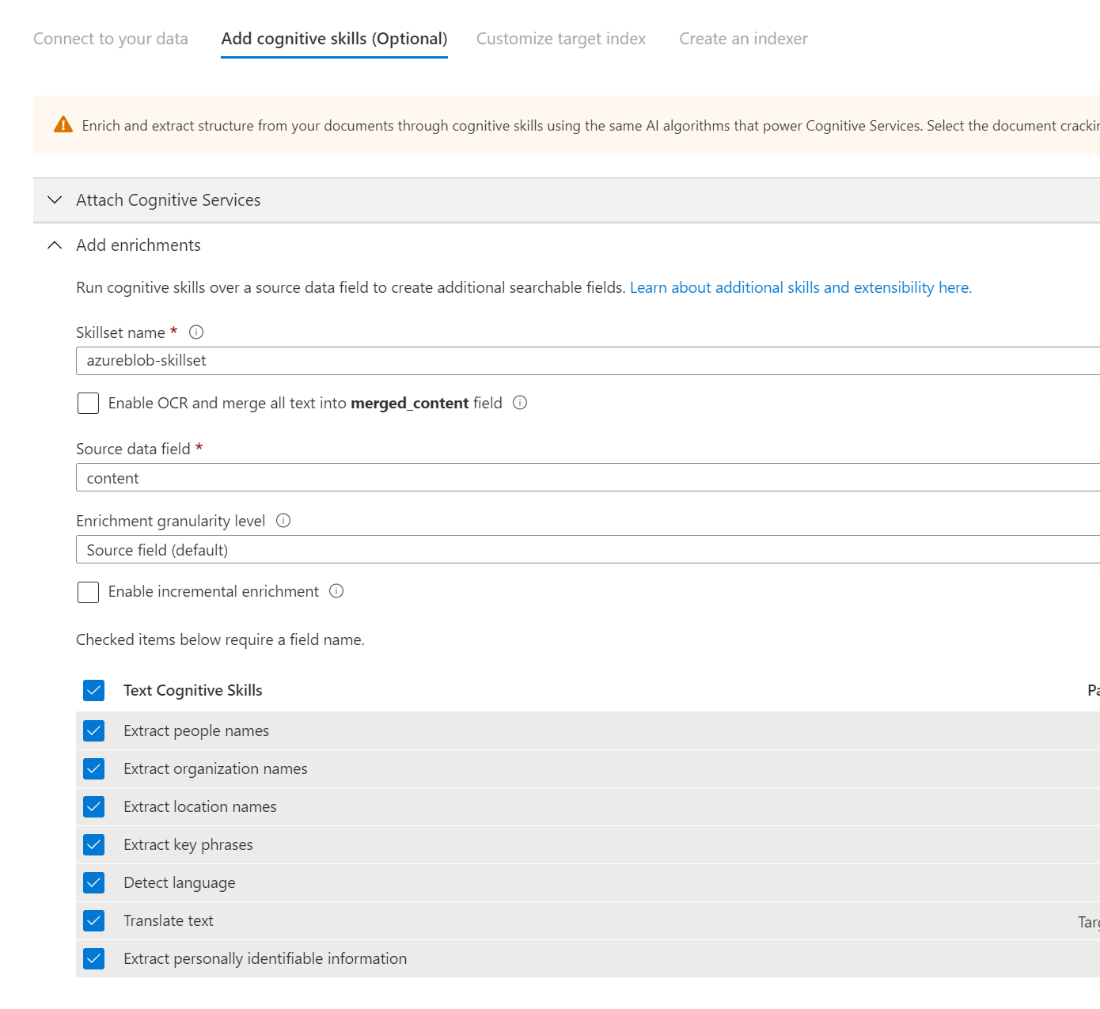
* **Storage Account** : This storage will store all the html documents that will be used by indexer to extract Json documents which will be added to the index
* **Search Service** : Azure Search service which searches the documents indexed from the storage account
* **Cognitive Services** : cognitive services use AI to extract the content out of uploaded html blobs and prepares them for indexing

Each html document at the storage account corrsponds to a Index document. AI enrichment services enabled by Cognitive Services, analyzes the contents of the html files (containing the MVP article) and extracts keywords, text content, people and locations to create json index documents.



After analysis The search index on Azure Search Service is updated by indexer. Index fields are automatically poulated by the enrichment AI services selected when setting up the index





## Component Design

The Blog Search component is designed using headless approach using Rendering Host.

|  |  |
| --- | --- |
| **Rendering Type** | Json Rendering |
| **Rendering Path** | /sitecore/layout/Renderings/Feature/Blogs/Blogs List |

*Mvp.Feature.Blog.Components.SearchBlogsListViewComponent* invokes the search service query and binds the view with the returned results.

The *SearchService* class exposes *SearchAsync* method that allows a paginated search in the publications index

# Setting up solution locally

1. Before you can run the solution, you will need to prepare the following for the Sitecore container environment:
   * A valid/trusted wildcard certificate for \*.sc.localhost
   * Hosts file entries for
     + mvp-cd.sc.localhost
     + mvp-cm.sc.localhost
     + mvp-id.sc.localhost
     + mvp.sc.localhost
   * Required environment variable values in .env for the Sitecore instance
     + (Can be done once, then checked into source control.)

* See Sitecore Containers documentation for more information on these preparation steps. The provided init.ps1 will take care of them, but **you should review its contents before running.**
* You must use an elevated/Administrator Windows PowerShell 5.1 prompt for this command, PowerShell 7 is not supported at this time.
* .\init.ps1 -InitEnv -LicenseXmlPath "C:\path\to\license.xml" -AdminPassword "DesiredAdminPassword"

1. At the bottom of the .env file you'll find the section for your Okta developer account details. You will need to populate the following values:
   * OKTA\_DOMAIN (*must* include protocol, e.g. OKTA\_DOMAIN=https://dev-your-id.okta.com)
   * OKTA\_CLIENT\_ID
   * OKTA\_CLIENT\_SECRET

Note that DOCKER\_RESTART defaults to no but can point to always or other values as per this page - <https://docs.docker.com/config/containers/start-containers-automatically/>

1. After completing this environment preparation, run the startup script from the solution root: ps1 .\up.ps1 Note that the up.ps1 script now automatically detects:

* if running Docker linux daemon and switches to Windows
* and stops IIS if it is running in the machine

1. When prompted, log into Sitecore via your browser, and accept the device authorization.
2. Wait for the startup script to open browser tabs for the rendered site and Sitecore Launchpad.