

Azure SQL Database – Azure Search Code Review

Level-300 Demonstration

Script

Version 1.0

09/3/2015

Prepared for: Darmadi Komo

Prepared by: Prowess Consulting

|  |  |
| --- | --- |
| **Form Factor** | Demo script (Microsoft Word) |
| **Target Audience** | TDMs and database administrators |
| **Goals/Objectives** | Increase confidence among Microsoft SQL Server administrators that Azure SQL Database is easy to configure and can still meet the needs of their production databases. |

Overview

This demonstration explores the search functionality in Microsoft Azure SQL Database in the context of a software-as-a-service (SaaS) provider, Wingtip Tickets, which provides ticketing software to artists and groups. This demonstration centers on the tenant Julie and the Plantes (a fictitious pop-music tenant).

Other Tenants that will be discussed in future labs, will include the following:

* The Archie Boyle Band (a fictitious rock-music tenant)
* Walla Walla Symphony (a fictitious classical-music tenant)

## Demo Architecture

## 

**Figure 1** Overall architecture of the various demo components

## Dependencies

This demonstration requires running the deployment and configuration PowerShell scripts from the Level-200 demonstration in order to work. The demo also requires the Azure Search code, which comes in a compressed archive file distinct from this script or the PowerShell scripts.

|  |  |  |
| --- | --- | --- |
|  | **Section 1: Azure Search Code Review** |  |
|  | Using Microsoft Visual Studio, you can see what is going on behind the scenes to make Azure Search work. | |
|  | 1. From Visual Studio, with the **WingTipTickets solution** open, and within the **Tenant.MVC** project, browse to the **App\_Start** folder, and then open **DataConfig.cs**. |  |
|  | 1. Point out the configure and CreateIndex methods to the audience. | Notice the configure method initializes the Azure Search service using the SearchServiceName from web.config, along with the SearchServiceKey.  Next notice the CreateIndex method, which creates a new Index called “Concerts.” |
|  | 1. Point out that the CreateIndexer method is used to create a new Azure SQL Database Indexer, which will use a view called "ConcertSearch." | Lastly, notice how the CreateIndexer method is used to create a new Azure SQL Database Indexer, which will use a view called "ConcertSearch." |
|  | 1. Open SSMS. 2. Expand the **ConcertSearch** **View** under the Primary Database. 3. Right-click **dbo.ConcertSearch** and then select **Edit Top 200 Rows**. | Let’s see the results of the view, in SQL Server Management Studio. |
|  | 1. Open the Azure Portal 2. Click **Browse** > **Search services** under your subscription on the Azure Portal. 3. Point out to the audience that the Azure Search service that corresponds to the name of your application. | Now let’s look at the Azure Search Service and the related Index in the Azure Portal.  Notice the Azure Search service that corresponds to the name of your application. |
|  | 1. Click on your Azure Search Service. 2. Point out the index Concerts. | Notice the index Concerts under the Azure Search Service. |
|  | 1. Click on the index **Concerts**. | When I click on the Concerts index, notice the fields that correspond to the fields in the CreateIndex method |
|  | 1. Switch to Visual Studio. 2. Browse to the **HomeController.cs** file under the **Controllers** folder. 3. Scroll down to public async **Task<ActionResult> AutoCompleteEvents(*string term*)**. This is what drives autocomplete, which you saw at the beginning of this exercise. | Let’s look at how this is used, in Visual Studio. We will browse to the HomeController.cs file under the Controllers folder.  Once there, we’ll scroll down to public async Task<ActionResult> AutoCompleteEvents(string term). This is what drives autocomplete, which we saw at the beginning of this part of the demo. |
|  | 1. Scroll up to public async Task<ActionResult> Index(). | And back here at the public async Task<ActionResult> Index(), notice that the first “else” statement block is used to match a search result to a single event, returning the result to the "ViewSearchResults" view |
|  |  | Notice that the second “else” statement block is used to match a search result to multiple events, and also returns the result to the "ViewSearchResults" view. |

|  |  |  |
| --- | --- | --- |
|  | **Section 2: Conclusion** |  |
|  | The search service in Azure provides power, built-in search capabilities for your users and customers. To see Azure Search in action from a customer point of view, ask your Microsoft sales representative about seeing the all-up, level-200 Azure SQL Database demonstration (particularly Section 6). | |