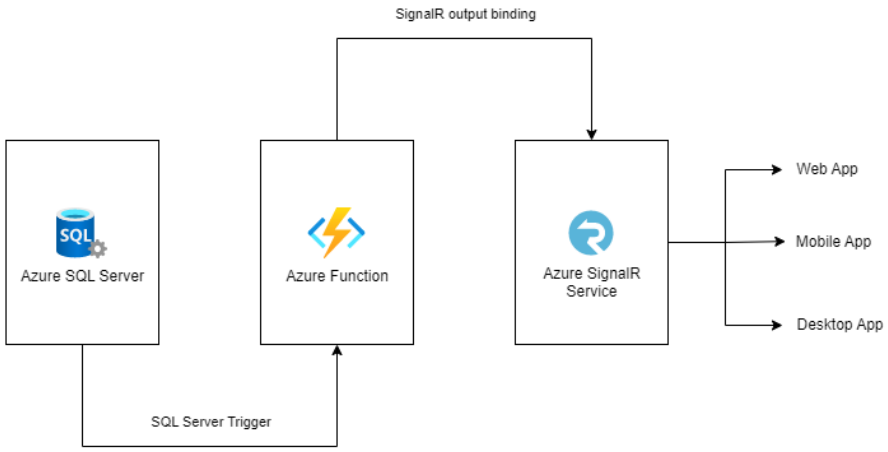
Implementing database change notification using SignalR and Azure Functions

This post is about implementing database change notification using SignalR and Azure Functions. In this post we will using Azure Function with SQL trigger for change identification and Azure SignalR binding for notifications.

Architecture

Here is the basic architecture of the implementation.



When a record added to database table – which will trigger the Azure Function – which is implemented using the SQL Trigger (preview) feature of Azure Functions. Then the Azure SignalR service will be invoked – using the Azure Function SignalR binding.

Implementation

We will be using a simple note taking application - in the application if someone adding a note, it will be automatically populated to all the users who are online.

Database Schema

Here is the database schema.

```
CREATE TABLE [dbo].[Notes](
        [Id] [int] NOT NULL IDENTITY(1,1) PRIMARY KEY,
        [Content] [nvarchar](max) NOT NULL,
        [CreatedOn] [date] NOT NULL,
        [CreatedBy] [nvarchar](255) NOT NULL,
        [UpdatedOn] [date] NULL,
        [IsDeleted] [bit] NOT NULL DEFAULT 0
)
```

For the SQL trigger to function, we need to enable change tracking in the database. We can do this using the following code.

```
ALTER DATABASE [NotesDb]
SET CHANGE_TRACKING = ON
(CHANGE_RETENTION = 2 DAYS, AUTO_CLEANUP = ON);

ALTER TABLE [dbo].[Notes]
ENABLE CHANGE_TRACKING;
```

Azure Function implementation

Next, we need to create an Azure Function with SQL trigger. To do this, we need to create an Azure Function with Http Trigger and add the `Microsoft.Azure.WebJobs.Extensions.Sql` nuget package. And we need to modify the code like this.

```
public static class WriteDataNotifications
{
    [FunctionName("WriteDataNotifications")]
    public static async Task Run(
        [SqlTrigger("[dbo].[Notes]", ConnectionStringSetting =
"NotesDbConnection")]
        IReadOnlyList<SqlChange<Note>> noteChanges,
        ILogger logger)
    {
        foreach (var noteChange in noteChanges)
        {
            var note = noteChange.Item;
            logger.LogInformation($"Change operation: {noteChange.Operation}");
            logger.LogInformation($"Id: {note.Id}, Content: {note.Content}, " +
                $"Created By: {note.CreatedBy}, Created On: {note.CreatedOn}");
        }
    }
}
```

And here is the Note class.

```
public class Note
{
  public int Id { get; set; }
  public string Content { get; set; }
  public DateTime CreatedOn { get; set; }
  public DateTime UpdatedOn { get; set; }
  public string CreatedBy { get; set; }
  public bool IsDeleted { get; set; }
}
```

Once we configured it, then we need to create an Azure SignalR service and configure Azure Function SignalR binding.

After created the Azure SignalR service, we need to add reference of `Microsoft.Azure.WebJobs.Extensions.SignalRService` nuget package.

And modify the Azure Function like this.

```
[FunctionName("WriteDataNotifications")]
public static async Task Run(
  [SqlTrigger("[dbo].[Notes]", ConnectionStringSetting = "NotesDbConnection")]
  IReadOnlyList<SqlChange<Note>> noteChanges,
```

```
  [SignalR(ConnectionStringSetting = "AzureSignalRConnectionString", HubName =
"serverless")]
  IAsyncCollector<SignalRMessage> signalRMessages,
  ILogger logger)
{
  foreach (var noteChange in noteChanges)
  {
    var note = noteChange.Item;
    logger.LogInformation($"Change operation: {noteChange.Operation}");
    logger.LogInformation($"Id: {note.Id}, Content: {note.Content}, Created By:
{note.CreatedBy}, Created On: {note.CreatedOn}");

    await signalRMessages.AddAsync(new SignalRMessage
    {
      Target = "NewNote",
      Arguments = new[] { $"Id: {note.Id}, Content: {note.Content}, Created By:
{note.CreatedBy}, Created On: {note.CreatedOn}" }
    });
  }
}
```

To configure SignalR in the Azure Function, we need to configure one more Azure Function - `Negotiate` - which will be called by SignalR client. Here is the implementation.

```
[FunctionName("negotiate")]
public static SignalRConnectionInfo Negotiate(
  [HttpTrigger(AuthorizationLevel.Anonymous)] HttpRequest req,
  [SignalRConnectionInfo(HubName = "serverless")] SignalRConnectionInfo
connectionInfo)
{
  return connectionInfo;
}
```

Now we have completed the implementation. Next, we can configure a client application which will show the updates when a user inserts a note to the table. For making the implementation simple, I am creating a new function which returns a HTML file.

```
[FunctionName("web")]
public static IActionResult GetHomePage([HttpTrigger(AuthorizationLevel.Anonymous)]
HttpRequest req, ExecutionContext context)
{
  var path = Path.Combine(context.FunctionAppDirectory, "content", "index.html");
  return new ContentResult
  {
    Content = File.ReadAllText(path),
    ContentType = "text/html",
  };
}
```

And in the index.html page we need to add the following code.

```
<!DOCTYPE html>

<html>

<body>
  <h1>Azure SignalR Serverless Sample</h1>
  <div id="messages"></div>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-
signalr/3.1.7/signalr.min.js"></script>
  <script>
```

```
    let messages = document.querySelector('#messages');
    const apiBaseUrl = window.location.origin;
    const connection = new signalR.HubConnectionBuilder()
      .withUrl(apiBaseUrl + '/api')
      .configureLogging(signalR.LogLevel.Information)
      .build();
    connection.on('NewNote', (message) => {
      document.getElementById("messages").innerHTML = message;
    });

    connection.start()
      .catch(console.error);
  </script>
</body>

</html>
```

The index.html file created in a content folder. And to copy the file to the output directory we can add the following code in the Project file.

```
<None Update="content\index.html">
 <CopyToOutputDirectory>Always</CopyToOutputDirectory>
</None>
```

This way we can implement Database changes notifications using Azure SignalR service and Azure Functions.