Configure the Azure Cosmos DB for NoSQL SDK

Introduction

- Configure the SDK for offline development
- Troubleshoot common connection errors
- Implement parallelism in the SDK
- Configure logging using the SDK

Enable offline development

- The Azure Cosmos DB emulator is a great tool for common Dev+Test workflows that developers may need to implement on their local machine.
- The emulator is available to run in Windows, Linux, or as a Docker container image.

Configuring the SDK to connect to the emulator

- string endpoint = "https://localhost:8081/";
- string key = "C2y6yDjf5/R+ob0N8A7Cgv30VRDJIWEHLM+4QDU5DE2nQ9nDuVTqobD4b8mGGyPMbIZnqyMsEcaGQy67XIw/Jw==";
- CosmosClient client = new (endpoint, key);

Handle connection errors

- There are some scenarios where a request can fail for a temporary reason
- Built-in retry
  - The Azure Cosmos DB for NoSQL SDK for .NET has built-in logic to handle common failures for read and query requests.
- If you are writing an application that experiences a write failure, it is up to your application code to implement retry logic which is considered a best practice.
- As an application developer, it's important to understand the HTTP status codes where retrying your request makes sense
- These codes include, but are not limited to:
  - 429: Too many requests
  - 449: Concurrency error
  - 500: Unexpected service error
  - 503: Service unavailable

Implement threading and parallelism

- The SDK implements thread-safe types and some degrees of parallelism
- Still there are best practices that you can implement in your application code to ensure the best performance

Use async/await in .NET

- Database database = await client.CreateDatabaseIfNotExistsAsync("cosmicworks");

Use built-in iterators

- container.GetItemLinqQueryable<T>()
- .Where(i => i.categoryId == 2)
- .ToFeedIterator<T>();

Configure logging

- There are many scenarios where you wish to log the HTTP requests that the Azure Cosmos DB for NoSQL SDK performs "under the hood.“
- The SDK includes a fluent client builder class that simplifies the process of injecting custom handlers into the HTTP requests and responses
- You can take advantage of this functionality to build a simple logging mechanism.

Thank You