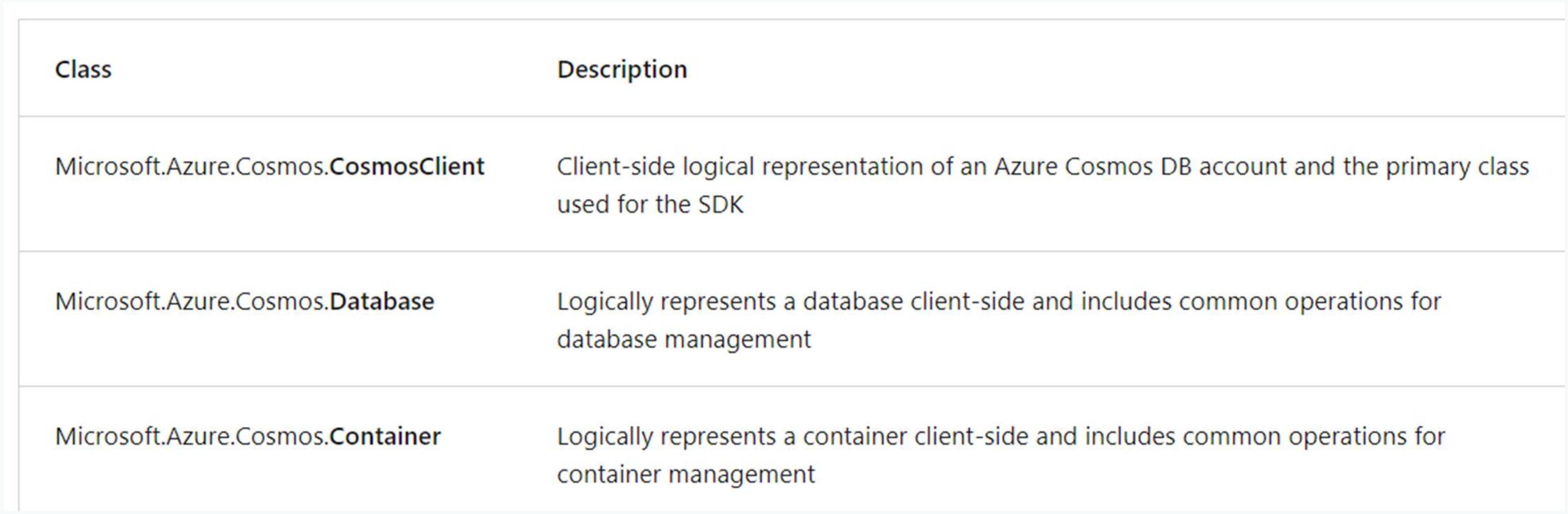
Use the Azure Cosmos DB for NoSQL SDK

Introduction

- There are various SDKs available to connect to the Azure Cosmos DB for NoSQL from many popular programming languages including, but not limited to:
  - .NET (C#)
  - Java
  - Python
  - JavaScript (Node.js)

Understand the SDK

- The Microsoft.Azure.Cosmos library is the latest version of the .NET SDK for Azure Cosmos DB for NoSQL.
- The library is open-source and hosted online on GitHub at azure/azure-cosmos-dotnet-v3.



Class	Description
Microsoft.Azure.Cosmos.**CosmosClient**	Client-side logical representation of an Azure Cosmos DB account and the primary class used for the SDK
Microsoft.Azure.Cosmos.**Database**	Logically represents a database client-side and includes common operations for database management
Microsoft.Azure.Cosmos.**Container**	Logically represents a container client-side and includes common operations for container management

Import from package manager

- dotnet add package Microsoft.Azure.Cosmos

Connect to an online account

- Will see in hands-on. Below is the code

```
– using Microsoft.Azure.Cosmos;
– string connectionString = "AccountEndpoint=https://dp420.documents.azure.com:443/;AccountKey=fDR2ci9QgkdkvERTQ==";
– CosmosClient client = new (connectionString);
– string endpoint = "https://dp420.documents.azure.com:443/";
– string key = "fDR2ci9QgkdkvERTQ==";
– CosmosClient client = new (endpoint, key);
– AccountProperties account = await client.ReadAccountAsync();
– Database database = client.GetDatabase("cosmicworks");
– Database database = await client.CreateDatabaseAsync("cosmicworks");
– Container container = database.GetContainer("products");

– Container container = await database.CreateContainerAsync(
–     "cosmicworks",
–     "/categoryId",
–     400
– );
```

CosmosClientOptions options

- The CosmosClientOptions class provides a range of options that you can configure for the client when it connects to an account
- These options include, but are not limited to:
  - The mode used to connect to the account
  - Custom consistency level used specifically for the client instance
  - The preferred account region[s]

Overriding default client options

- When connecting to an Azure Cosmos DB account using the CosmosClient class, there are a few assumptions that the client makes:
  - That you will want to connect to the first writable (primary) region of your account
  - That you will use the default consistency level for the account with your read requests
  - That you will connect directly to data nodes for requests
- To configure the client, you will need to create an instance of the CosmosClientOptions class
  - CosmosClientOptions options = new ();
  - CosmosClient client = new (endpoint, key, options);

Configure connectivity mode

- CosmosClientOptions options = new ()
- {
-     ConnectionMode = ConnectionMode.Direct
- };

- **<u>OR</u>**

- CosmosClientOptions options = new ()
- {
-     ConnectionMode = ConnectionMode.Gateway
- };

Changing the current consistency level

- CosmosClientOptions options = new ()
- {
-     ConsistencyLevel = ConsistencyLevel.Eventual
- };

Setting the preferred application region[s]

- The ApplicationRegion property sets the single preferred region that the client will connect to for operations
  - CosmosClientOptions options = new ()
  - {
  -     ApplicationRegion = "westus"
  - };
- If you would like to create a custom failover/priority list for the client to use for operations, you can use the ApplicationPreferredRegions property with a list of regions.
  - CosmosClientOptions options = new CosmosClientOptions()
  - {
  -     ApplicationPreferredRegions = new List<string> { "westus", "eastus" }
  - };

Exercise

- Connect to Azure Cosmos DB for NoSQL with the SDK

Thank You