Implement Azure Cosmos DB for NoSQL point operations

Introduction

- The NoSQL API SDK for Azure Cosmos DB is used to perform various point operations
  - Perform transactions, and
  - To process bulk data

Understand point operations

- Retrieving a document based on this document ID (or partition key) is called a point read
- At the most foundational level, you can create a C# class that represents an item in your container that, at a minimum, contains two members:
  - a string property named id with a public getter and setter
  - a string property with the same name as your partition key path with a public getter and setter
- Example
  - public class Product
  - {
  -     public string id { get; set; }
  -
  -     public string name { get; set; }
  -     public string categoryId { get; set; }
  -     public double price { get; set; }
  -     public string[] tags { get; set; }
  - }

Create documents

```
•   Product saddle = new()
•   {
•       id = "027D0B9A-F9D9-4C96-8213-C8546C4AAE71",
•       categoryId = "26C74104-40BC-4541-8EF5-9892F7F03D72",
•       name = "LL Road Seat/Saddle",
•       price = 27.12d,
•       tags = new string[]
•       {
•           "brown",
•           "weathered"
•       }
•   };

•   ItemResponse<Product> response = await container.CreateItemAsync<Product>(saddle);

•   HttpStatusCode status = response.StatusCode;
•   double requestUnits = response.RequestCharge;

•   Product item = response.Resource;
```

Read a document

- To do a point read of an existing item from the container, we need two things.
- First, we need the unique id of the item. Here, we store that id in a variable of the same name.
  - string id = "027D0B9A-F9D9-4C96-8213-C8546C4AAE71";
- Second, we need to create a variable of type PartitionKey with the string value at the partition key path for the item we are seeking.
  - string categoryId = "26C74104-40BC-4541-8EF5-9892F7F03D72";
  - PartitionKey partitionKey = new (categoryId);
- Once we have both items, we can invoke the asynchronous and generic ReadItemAsync<> method, which will return an item of the given generic type, Product, in this example.
  - Product saddle = await container.ReadItemAsync<Product>(id, partitionKey);

Update documents

- saddle.price = 35.00d;
- await container.UpsertItemAsync<Product>(saddle);

Configure time-to-live (TTL) value for a specific document

- [JsonProperty(PropertyName = "ttl", NullValueHandling = NullValueHandling.Ignore)]
- public int? ttl { get; set; }

- saddle.ttl = 1000;

- await container.UpsertItemAsync<Product>(saddle);

Delete documents

- string id = "027D0B9A-F9D9-4C96-8213-C8546C4AAE71";
- string categoryId = "26C74104-40BC-4541-8EF5-9892F7F03D72";
- PartitionKey partitionKey = new (categoryId);

- await container.DeleteItemAsync<Product>(id, partitionKey);

Exercise

- Create and update documents with the Azure Cosmos DB for NoSQL SDK

Thank You