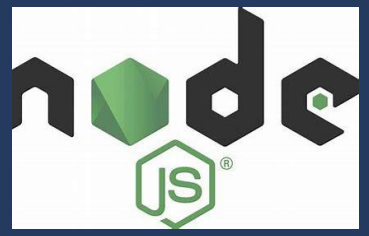
Microsoft

Azure Cosmos DB web apps with Node.js and TypeScript

Chris Joakim, Microsoft, Cosmos DB Global Black Belt (GBB)
https://www.linkedin.com/in/chris-joakim-4859b89



Presentation Outline

- Meet Chris
- Meet Azure Cosmos DB
- The Azure Cosmos DB JavaScript/TypeScript SDK
- Why TypeScript?
- The azu-js Package @ NPM
- The TypeScript/Express/Cosmos DB Web Application - code walkthrough and demo
- Summary

Meet Chris

- 2021 - **Microsoft - Cosmos DB Global Black Belt (GBB)**
- 2016 - Microsoft - Cloud Solution Architect
- 1986 - 2016 - Software Developer, CTO, Owner
- Languages: COBOL -> Smalltalk -> Java -> Ruby on Rails (RoR) -> Node.js (MEAN) -> Python -> TypeScript. Used **CoffeeScript** with RoR and MEAN stacks.
- Databases: IMS/DB, Relational (DB2, MySQL, PostgreSQL), **MongoDB (2008),** -> **Cosmos DB (2017)**
- Preferences: Dynamic Languages and Schemaless JSON-based NoSQL
- Davidson/Charlotte, NC, USA
- LinkedIn: https://www.linkedin.com/in/chris-joakim-4859b89/
- GitHub: https://github.com/cjoakim
- **Why** did I Highlight **CoffeeScript** and **MongoDB & Cosmos DB?**

Meet Azure Cosmos DB

- Cloud-Native family of NoSQL Databases
  - Born in the cloud
  - High Performance – single-digit ms queries
  - High Availability – multiple copies of your data, 99.99 to 99.999% SLA
  - Globally Replicated, on the high-speed Azure fiber network
- APIs: **NoSQL**, Mongo, Mongo vCore, Cassandra, Cassandra MI, Gremlin
- Dynamic Throughput Model with **Request Units (RUs)**
  - Enables you to "right size" your Cosmos DB account and costs
- NoSQL API stores JSON Documents, but uses SQL for queries
- Enables high-performance **"end-to-end JSON" applications**
- **Change-Feed** for "event driven" apps with serverless Azure Functions
- **Synapse Link** for analytics and batch
- https://learn.microsoft.com/en-us/azure/cosmos-db/

Azure Cosmos DB : Data Explorer in Azure Portal

Dashboard > gbbcjcosmos > gbbcjcdbnosql

gbbcjcdbnosql | Data Explorer

Azure Cosmos DB account

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Cost Management
- Quick start
- Notifications
- Data Explorer

Settings

- Features
- Replicate data globally
- Default consistency
- Backup & Restore
- Networking
- CORS
- Dedicated Gateway
- Keys
- Advisor Recommendations
- Microsoft Defender for Cloud

New Item | Update | Discard | Delete | Upload Item

NOSQL API

MY DATA

- dev
  - airports
  - baseballplayers
    - Items
    - Scale & Settings
    - Stored Procedures
    - User Defined Functions
    - Triggers
    - Conflicts
  - events
  - global
  - GlobalThoughputController
  - GlobalThoughPutController
  - imdb_graph
  - imdb_seed
  - leases
  - local
  - npm graph

SAMPLE DATA

- CopilotSampleDb

NOTEBOOKS

Home | baseballplayers... | Scale & Settin...

SELECT * FROM c | Edit Filter

id	/pla...
ecb908e...	aardsda01
cfacb778...	aaronha01
9a8b178f...	aaronto01
7cd6857...	aasedo01
e0baa20...	abadan01
0fd9a25f...	abadfe01
4ed2f532...	abbated01
8ed040f5...	abbotco01
8bf518f9...	abbotfr01
a57b5bb...	abbotgl01
2f710098...	abbotje01
6823136...	abbotji01
06d0ad5...	abbotku01
902a192...	abbotky01
e05f2e1e...	abbotod01
5a7d7b7...	abbotpa01


```
{
    "playerID": "aaronha01",
    "birthYear": 1934,
    "birthCountry": "USA",
    "deathYear": "2021.0",
    "nameFirst": "Hank",
    "nameLast": "Aaron",
    "weight": 180,
    "height": 72,
    "bats": "R",
    "throws": "R",
    "debut": "1954-04-13",
    "finalGame": "1976-10-03",
    "teams": {
        "total_games": 3298,
        "teams": {
            "ML1": 1806,
            "ATL": 1270,
            "ML4": 222
        },
        "primary_team": "ML1"
    },
    "primary_position": "RF",
    "batting": {
        "G": 3298,
        "AB": 12364,
        "R": 2174,
        "H": 3771,
        "2B": 624,
        "3B": 98,
        "HR": 755,
        "RBI": 2297,
```

Azure Cosmos DB : Request Units and Autoscale

Dashboard > gbbcjcosmos > gbbcjcdbnosql

gbbcjcdbnosql | Data Explorer

Azure Cosmos DB account

Search

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Cost Management
- Quick start
- Notifications
- Data Explorer

Settings

- Features
- Replicate data globally
- Default consistency
- Backup & Restore
- Networking
- CORS
- Dedicated Gateway
- Keys

New Container | New Notebook | Connect to GitHub | New SQL Query | Open Query | New Stored Procedure

NOSQL API

MY DATA

- dev
  - airports
  - baseballplayers
    - Items
    - Scale & Settings
    - Stored Procedures
    - User Defined Functions
    - Triggers
    - Conflicts
  - events
  - global
  - GlobalThoughputController
  - GlobalThoughPutController
  - imdb_graph
  - imdb_seed
  - leases
  - local
  - npm_graph

SAMPLE DATA

- CopilotSampleDb

Home | baseballplayers... | Scale & Settin... ×

Scale | Settings | Indexing Policy | Conflict Resolution

Throughput (autoscale)

(•) Autoscale ( ) Manual

Maximum RU/s required by this resource *

4000

1,000 10,000 100,000

Instant 4-6 hrs

Based on usage, your container throughput will scale from **400 RU/s (10% of max RU/s) - 4000 RU/s**

Estimate your required RU/s with capacity calculator

Storage capacity

Unlimited

Azure Cosmos DB : TTL, Synapse Link, Partition Key Configuration

Dashboard > gbbcjcosmos > gbbcjcdbnosql

gbbcjcdbnosql | Data Explorer

Azure Cosmos DB account

Search

Save Discard

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Cost Management
- Quick start
- Notifications
- Data Explorer

Settings

- Features
- Replicate data globally
- Default consistency
- Backup & Restore
- Networking
- CORS
- Dedicated Gateway
- Keys
- Advisor Recommendations
- Microsoft Defender for Cloud

NOSQL API

MY DATA

- dev
  - airports
  - baseballplayers
    - Items
    - Scale & Settings
    - Stored Procedures
    - User Defined Functions
    - Triggers
    - Conflicts
  - events
  - global
  - GlobalThoughtputController
  - GlobalThoughPutController
  - imdb_graph
  - imdb_seed
  - leases
  - local
  - npm graph

SAMPLE DATA

- CopilotSampleDb

NOTEBOOKS

Home | baseballplayers... | Scale & Settin... ×

Scale | **Settings** | Indexing Policy | Conflict Resolution

Time to Live

- (●) Off
- ( ) On (no default)
- ( ) On

Geospatial Configuration

- (●) Geography
- ( ) Geometry

Analytical Storage Time to Live

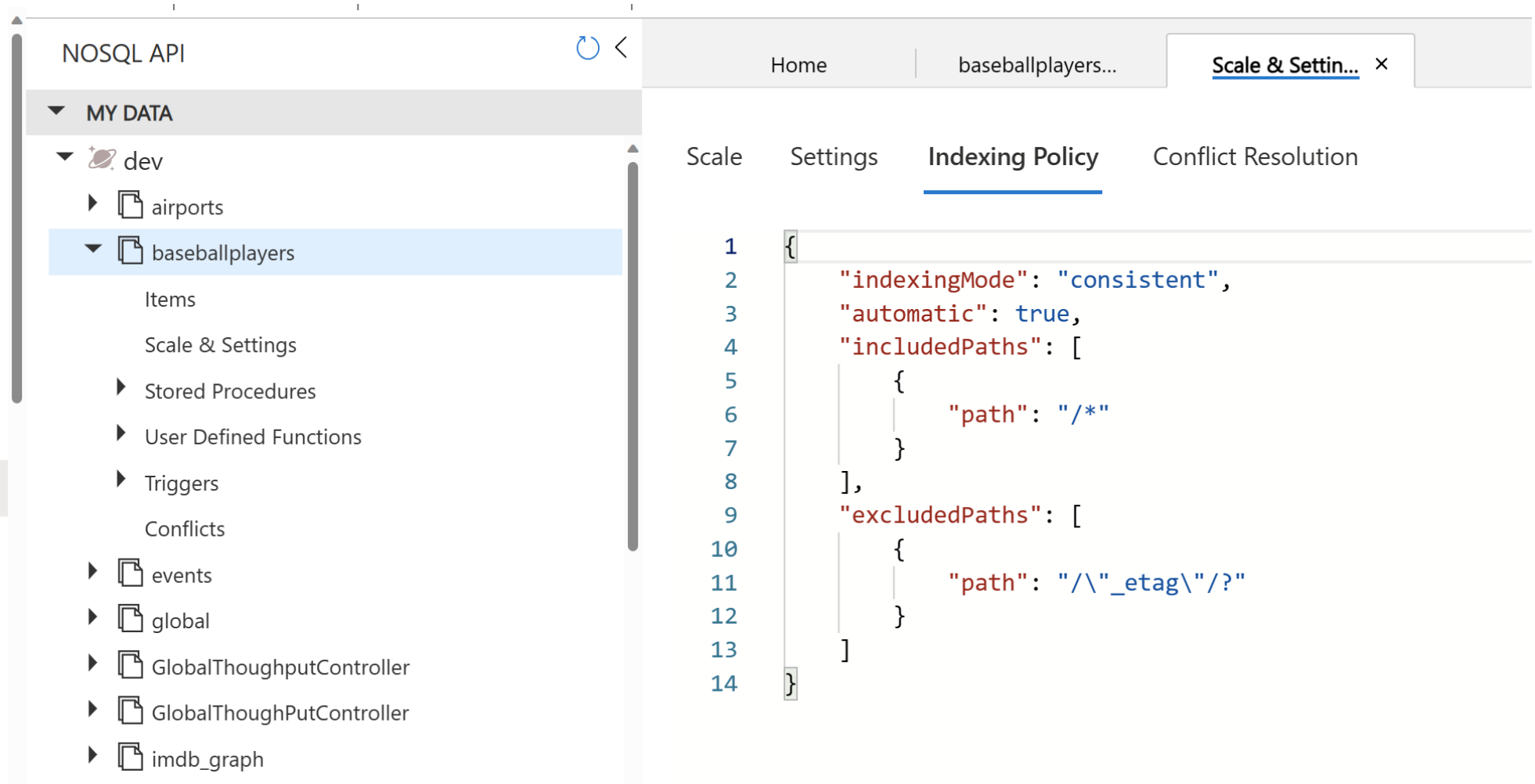
- ( ) Off
- (●) On (no default)
- ( ) On

Partition key

/playerID

Large partition key has been enabled.

Azure Cosmos DB : Indexing – Defined with JSON, Defaults to /*



Azure Cosmos DB : Query with SQL

Home | baseballplayers... | Scale & Settin... | Query 1 ×

```
1   SELECT * FROM c where c.playerID = "aaronha01"
```

Results | Query Stats

1 - 1


```
[
    {
        "playerID": "aaronha01",
        "birthYear": 1934,
        "birthCountry": "USA",
        "deathYear": "2021.0",
        "nameFirst": "Hank",
        "nameLast": "Aaron",
        "weight": 180,
        "height": 72,
        "bats": "R",
        "throws": "R",
        "debut": "1954-04-13",
        "finalGame": "1976-10-03",
        "teams": {
            "total_games": 3298,
            "teams": {
                "ML1": 1806,
                "ATL": 1270,
```

The Azure Cosmos DB JavaScript/TypeScript SDK

- https://github.com/Azure/azure-sdk-for-js
- https://github.com/Azure/azure-sdk-for-js/tree/main/sdk/cosmosdb/cosmos
- https://www.npmjs.com/package/@azure/cosmos

- The Cosmos DB JavaScript SDK is part of the Azure JavaScript SDK
- **It is written in TypeScript**, and is transpiled into JavaScript
- Current Version is **4.0.0** – adds excellent features
  - Throughput Control, Hierarchical Partition keys, Async
- Supports **Asynchronous** Operations – **await/async**
- npm install `@azure/cosmos`

- Other Links:
- https://learn.microsoft.com/en-us/javascript/api/overview/azure/cosmos-readme?view=azure-node-latest
- https://learn.microsoft.com/en-us/**samples**/azure/azure-sdk-for-js/cosmos-typescript/

SDK Code on GitHub

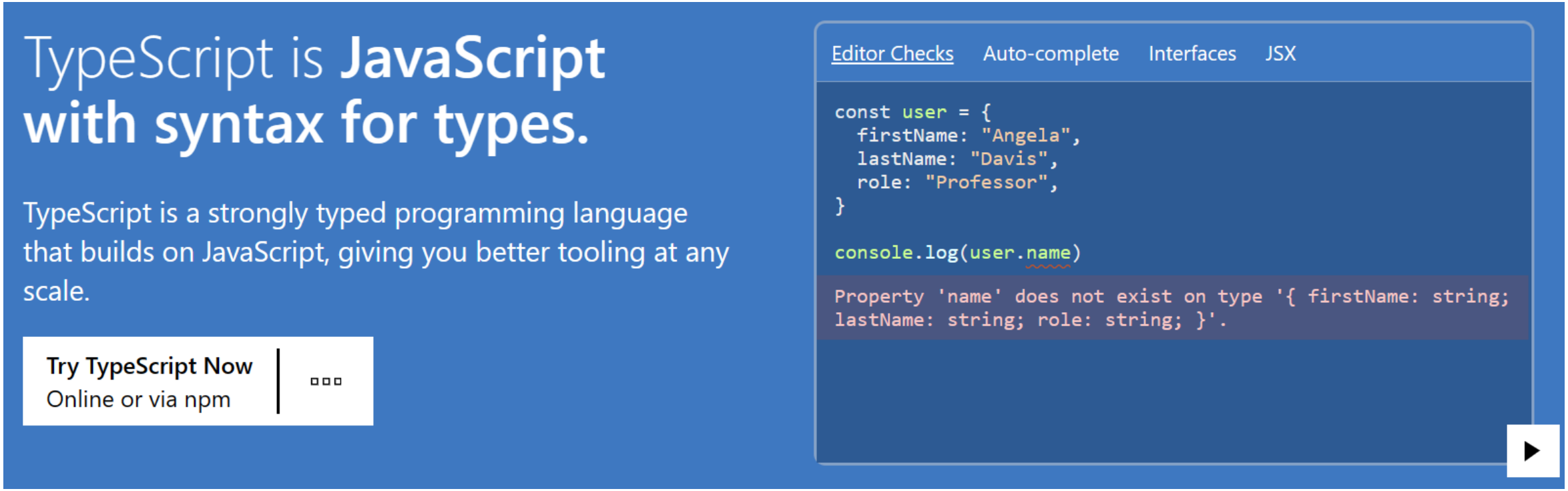
main azure-sdk-for-js / sdk / cosmosdb / cosmos / src / client / Item / Items.ts ↑ Top

Code Blame 622 lines (591 loc) · 21.7 KB Raw

```
61  export class Items {
67    */
68    constructor(public readonly container: Container, private readonly clientContext: ClientContext) {
69      this.partitionKeyRangeCache = new PartitionKeyRangeCache(this.clientContext);
70    }
71
72    /**
73     * Queries all items.
74     * @param query - Query configuration for the operation. See {@link SqlQuerySpec} for more info on how to configure a c
75     * @param options - Used for modifying the request (for instance, specifying the partition key).
76     * @example Read all items to array.
77     * ```typescript
78     * const querySpec: SqlQuerySpec = {
79     *   query: "SELECT * FROM Families f WHERE f.lastName = @lastName",
80     *   parameters: [
81     *     {name: "@lastName", value: "Hendricks"}
82     *   ]
83     * };
84     * const {result: items} = await items.query(querySpec).fetchAll();
85     * ```
86     */
87    public query(query: string | SqlQuerySpec, options?: FeedOptions): QueryIterator<any>;
88    /**
89     * Queries all items.
90     * @param query - Query configuration for the operation. See {@link SqlQuerySpec} for more info on how to configure a c
91     * @param options - Used for modifying the request (for instance, specifying the partition key).
92     * @example Read all items to array.
```

Why TypeScript?

- https://www.typescriptlang.org/
- Type Safety, High Quality, and High Productivity!



Why TypeScript?

Development

- Author your Code as TypeScript (with types, *.ts)
- "**Transpile**" the *.ts into regular JavaScript with the **tsc** program
- Catch many coding errors during development, not deployment
- Great IDE Support – **Visual Studio Code** and GitHub Copilot

Deployment

- At runtime it's just the JavaScript that executes (*.js), all types are eliminated
- The **Node.js** runtime is performant, mature, lightweight
- Excellent for Docker Containers and Microservices
- Azure Kubernetes Service (AKS), Azure Container Instances (ACI),
- Azure Functions, Azure Container Apps (ACA)

Why TypeScript? Open-Source, Loved, Excellent Docs, Easy to Learn

- Open-Source, created and developed by Microsoft
- https://www.typescriptlang.org/docs/
- Type Safety and High Productivity!

Loved by Developers

2020 Developer Survey

Rust
TypeScript
Python

Voted **2nd most loved programming language** in the Stack Overflow 2020 Developer survey

TypeScript was **used by 78%** of the 2020 State of JS respondents, with **93% saying they would use it again**.

TypeScript was given the award for **"Most Adopted Technology"** based on year-on-year growth.

TypeScript: What does it look like?

Items of note:

- import/export statements
- classes
- constructor
- method args with types
- optional args with ?
- return type defined
- use of duck-type interfaces
- also public/private/static
- tsc compiler will complain

```
1   import {
2       SqlQuerySpec,
3       SqlParameter
4     } from "@azure/cosmos";
5
6   // This class is used to create instances of the SqlQuerySpec interface.
7   // See https://learn.microsoft.com/en-us/javascript/api/@azure/cosmos/sqlqu
8
9   export class CosmosNoSqlQuerySpecUtil {
10
11      constructor() {}
12
13      querySpec(sql: string, parameters?: string[]) : SqlQuerySpec {
14          let params = [];
15          if (parameters) {
16              params = parameters;
17          }
18          return { query: sql, parameters: params };
19      }
20  }
21
```

The azu-js Package @ NPM

- Provides an easy to use **wrapper** for some Azure SDK functions
  - Azure Cosmos DB NoSQL API
  - Azure Blob Storage
  - Azure Cognitive Search (now Azure AI Search)
  - Azure OpenAI
  - Also File System, Config, Logging
- Provides easy to find **samples** of the Cosmos DB JS SDK
- I use it for my many TypeScript & Cosmos DB projects
- Includes Unit Tests using the Jest test framework
- The Unit Tests also provide code examples
- https://www.npmjs.com/package/azu-js

The TypeScript/Express/Cosmos DB Web Application : Code

- https://github.com/cjoakim/azure-cosmos-db-ts-web
- This repo provides a working demonstration application
- As well as a “starter codebase” for developing your own apps

- Also in the repo:
  - Dockerfile for building a container
  - compose-up and compose-down scripts for Docker Compose
  - Az CLI provisioning script to deploy container to ACI

- See https://www.npmjs.com/package/tsg-js for a minimal “scaffolding code generator” like in Ruby on Rails

The TypeScript/Express/Cosmos DB Web Application : Code

- **index.ts** entry point – using the Express Web Framework
- https://github.com/cjoakim/azure-cosmos-db-ts-web/blob/main/src/IndexRouter.ts

- **CosmosRouter.ts** – uses the azu-js package
- https://github.com/cjoakim/azure-cosmos-db-ts-web/blob/main/src/CosmosRouter.ts

- **ejs HTML templates** in the views/ directory
- https://github.com/cjoakim/azure-cosmos-db-ts-web/tree/main/views

- **package.json** – defines the Node.js project and NPM dependencies
- **tsconfig.json** – defines the behavior of the **tsc** transpiler program
- **web.ps1 and web.sh** – scripts to start the application on Developer workstation

The TypeScript/Express/Cosmos DB Web Application : Demo

See the live or recorded demo,
or the screen shots on the following pages

Or, deploy it yourself using the az CLI deployment script in this repo in the az/ directory
az/provision-webapp-aci-prod.ps1

DockerHub image:
cjoakim/azure-cosmos-db-ts-web-prod:latest

Web App Screen Shots : About

cosmos db + About | Configuration | Cosmos DB Metadata | Cosmos DB Upload | Cosmos DB Query | Cosmos DB CRUD | OpenAI | Cognitive Search | Logoff

About this Application

Highlights

- Intended for both demonstration purposes, and also as a working **starter app** to accelerate application development
- This web application is implemented in **Node.js and TypeScript**
- It uses the **Azure JavaScript SDK** and **azu-js** NPM packages to interact with **Azure Cosmos DB and other Azure PaaS services**
- The azu-js NPM package builds upon the Azure JavaScript SDK, and is also implemented in **TypeScript**
- It uses the popular **Express** JavaScript web framework - for both UIs and Microservices
- Developed with Visual Studio Code and **GitHub Copilot** - a highly productive dev environment
- Deployable as a **Docker container** - to Docker compose, **Azure Container Instances, Azure Container Apps, Azure Kubernetes Service**
- This application can also be used **to explore Cosmos DB and Azure** in a programming language agnostic way

Implementation Details

This app is built with the following **modern tech stack**, using popular Azure PaaS services:

Node.js	https://nodejs.org/en	link
TypeScript	https://www.typescriptlang.org/	link
Express Web Framework	https://expressjs.com/	link
EJS HTML Templates	https://ejs.co/	link
Visual Studio Code with GitHub Copilot	https://code.visualstudio.com/blogs/2023/03/30/vscode-copilot	link
Docker	https://www.docker.com/	link
Azure Cosmos DB	https://learn.microsoft.com/en-us/azure/cosmos-db/	link
Azure OpenAI	https://learn.microsoft.com/en-us/azure/ai-services/openai/	link
Azure Cognitive Search	https://learn.microsoft.com/en-us/azure/search/	link

Web App Screen Shots : Configuration

cosmos db + About | Configuration | Cosmos DB Metadata | Cosmos DB Upload | Cosmos DB Query | Cosmos DB CRUD | OpenAI | Cognitive Search | Logoff

Application Configuration

- This application uses the following specific **environment variables** listed below
- The **actual environment variable values for this deployment are shown below**; the secret values are truncated with "..."
- The application is easily deployable to Azure Docker Container PaaS services - such as **Azure Container Instances (ACI)**
- See the /az directory in the GitHub repository for **az CLI** deployment scripts
- See the **docker-compose-web.yml** file in the repo to run the Docker image locally with compose

Environment Variable Details

Required Environment Variable Name	Deployment Environment Variable Value
AZURE_COSMOSDB_NOSQL_RW_KEY1	ba9O...
AZURE_COSMOSDB_NOSQL_URI	https://gbbcjcdbnosql.documents.azure.com:443/
AZURE_OPENAI_EMBEDDINGS_DEPLOYMENT	embeddings
AZURE_OPENAI_KEY1	b910...
AZURE_OPENAI_URL	https://cjz5mxhd2ciwy-openai.openai.azure.com/
AZURE_SEARCH_ADMIN_KEY	j9bA...
AZURE_SEARCH_NAME	gbbcjsearch
AZURE_SEARCH_QUERY_KEY	Smeu...
AZURE_SEARCH_URL	https://gbbcjsearch.search.windows.net
AZURE_WEB_AUTH_USERS	gues...
AZURE_WEB_COOKIE_AGE	undefined
AZURE_WEB_COOKIE_KEYS	unde...

The current HTTP session database/container pair is .
HTTP session id: 6f8aa43f-667c-4386-95e0-c903e9c50c95

Web App Screen Shots : Cosmos DB Metadata

cosmos db + About | Configuration | Cosmos DB Metadata | Cosmos DB Upload | Cosmos DB Query | Cosmos DB CRUD | OpenAI | Cognitive Search | Logoff

Azure Cosmos DB NoSQL Account - Metadata

URI: https://gbbcjcdbnosql.documents.azure.com:443/

- Click the "Get Account Metadata" button below
- The application will gather and display database, container, and throughput metadata
- This metadata is **required for the other Cosmos DB pages of this application**
- Thus, the other Cosmos DB pages will redirect to this page if your database metadata hasn't been read yet
- The metadata is stored as a session-specific file on the server

Get Account Metadata

Database	Container	Partition Key(s)	Document TTL	Analytic TTL	Throughput Configuration
dev	GlobalThoughPutController	/groupId	-1		container level: { "offerThroughput": 4000, "offerIsRUPerMinuteThroughputEnabled": false, "offerMinimumThroughputParameters": { "maxThroughputEverProvisioned": 4000, "maxConsumedStorageEverInKB": 0 } }
dev	GlobalThoughputController	/groupId	-1		container level: { "offerThroughput": 4000, "offerIsRUPerMinuteThroughputEnabled": false, "offerMinimumThroughputParameters": { "maxThroughputEverProvisioned": 4000, "maxConsumedStorageEverInKB": 0 } }
dev	airports	/pk			container level: { "offerThroughput": 400, "offerMinimumThroughputParameters": { "maxThroughputEverProvisioned": 4000, "maxConsumedStorageEverInKB": 0 }, "offerAutopilotSettings": { "maxThroughput": 4000 }

Web App Screen Shots : Cosmos DB JSON File Upload

cosmos db + About | Configuration | Cosmos DB Metadata | Cosmos DB Upload | Cosmos DB Query | Cosmos DB CRUD | OpenAI | Cognitive Search | Logoff

Azure Cosmos DB NoSQL Account - Upload Documents (Bulk Load)

Documents Upload Form

Database & Container => Partition Key

dev | unittests => /pk

JSON File containing an Array of Objects/Documents

Choose File | No file chosen

☐ Generate new id values for each document?

Upload

Upload Results


```
  {
 "inputDocumentCount": 197,
 "startTime": 1702917378269,
 "endTime": 1702917378654,
 "elapsedTime": 385,
 "batchSize": 50,
 "batchCount": 4,
 "totalRUs": 5751.047619047621,
 "responseCodes": {
   "201": 197
 }
}
```

Web App Screen Shots : Queries and Point Reads

Azure Cosmos DB NoSQL Account - Queries and Point-Reads

Database & Container => Partition Key

dev | unittests => /pk

Enter a SQL Query, or just id and partition key values for a point-read

select * from c offset 0 limit 1

Submit

Query found 1 documents, 2.28 RU


```
[
{
  "playerID": "abreubo01",
  "birthYear": 1974,
  "birthCountry": "Venezuela",
  "deathYear": "",
  "nameFirst": "Bobby",
  "nameLast": "Abreu",
  "weight": 220,
  "height": 72,
  "bats": "L",
  "throws": "R",
  "debut": "1996-09-01",
  "finalGame": "2014-09-28",
  "teams": {
    "total_games": 2425,
    "teams": {
      "HOU": 74,
      "PHI": 1353,
      "NYA": 372,
      "LAA": 456,
      "LAN": 92,
      "NYN": 78
    },
```


Azure Cosmos DB NoSQL Account - Queries and Point-Reads

Database & Container => Partition Key

dev | unittests => /pk

Enter a SQL Query, or just id and partition key values for a point-read

7c3fcd74-b446-455a-b454-f09b28106f6d abreubo01

Submit

Point-read, document found, 1.05 RU


```
[
{
  "playerID": "abreubo01",
  "birthYear": 1974,
  "birthCountry": "Venezuela",
  "deathYear": "",
  "nameFirst": "Bobby",
  "nameLast": "Abreu",
  "weight": 220,
  "height": 72,
  "bats": "L",
  "throws": "R",
  "debut": "1996-09-01",
  "finalGame": "2014-09-28",
  "teams": {
    "total_games": 2425,
    "teams": {
      "HOU": 74,
      "PHI": 1353,
      "NYA": 372,
      "LAA": 456,
      "LAN": 92,
      "NYN": 78
    },
```

Web App Screen Shots : CRUD operations, including Patch

Azure Cosmos DB NoSQL Account - CRUD Operations - Create, Upsert, and Delete

Document CRUD Form

Database & Container => Partition Key

dev | unittests => /pk

Enter a well-formed JSON Document, then select an operation - Create, Upsert, Patch, or Delete.
A sample document is shown below, but you can paste in your own JSON Document.
The JSON will be validated in the browser before being sent to the server.

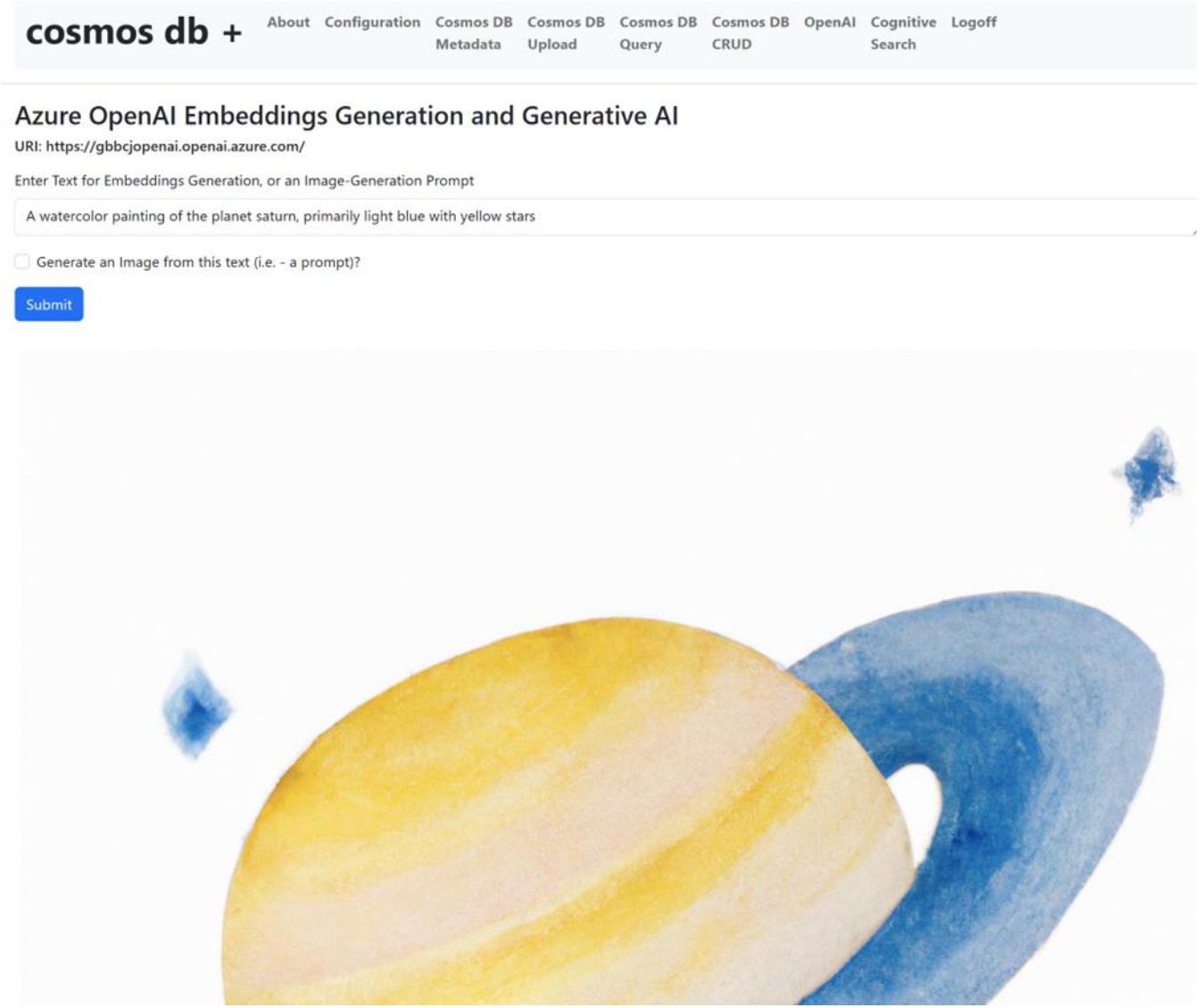

```
{
  "id": "e2dad0a9-601a-4967-9dfa-2870d1c272a6",
  "pk": "123456",
  "band": "U2",
  "tour": "UV",
  "song": "Ultraviolet",
  "duration": 331,
  "band_members": [
    "Bono",
    "Edge",
    "Adam",
    "Larry",
    "Chris"
  ],
  "date": "2023-12-18T18:45:17.313Z",
```


Create

Patch Attributes for the above Document - +attrName to add, -attrName to delete, attrName to change.
Only root level attributes are supported in this demo app.

Submit

Web App Screen Shots : OpenAI Integration



Web App Screen Shots : Cognitive Search (now AI Search)

Azure Cognitive Search of Indexed Cosmos DB Documents

URI: https://gbbcjsearch.search.windows.net

Search Index Name

baseballplayers

Enter a Search as well-formed JSON. Or enter the value "reset" to show the sample query.

```
{
  "queryType": "simple",
  "search":    "primary_position eq 'SS' +USA +derek",
  "searchMode": "any",
  "orderby":   "playerID",
  "select":    "id,playerID,nameFirst,nameLast,primary_position,birthCountry,embeddings_str",
  "count":     "true"
}
```

Submit

Search Results


```
{
"url": "https://gbbcjsearch.search.windows.net/indexes/baseballplayers/docs/search?api-version=2023-07-01-Preview",
"method": "POST",
"body": {
  "queryType": "simple",
  "search": "primary_position eq 'SS' +USA +derek",
  "searchMode": "any",
  "orderby": "playerID",
  "select": "id,playerID,nameFirst,nameLast,primary_position,birthCountry,embeddings_str",
  "count": "true"
},
"status": 200,
"respData": {
  "@odata.context": "https://gbbcjsearch.search.windows.net/indexes('baseballplayers')/$metadata#docs(*)",
  "@odata.count": 3,
  "value": [
    {
      "@search.score": 9.381344,
      "id": "00aff935-9180-4f77-b991-7ec6b1c76213",
      "playerID": "jeterde01",
      "birthCountry": "USA",
      "nameFirst": "Derek",
      "nameLast": "Jeter",
      "primary_position": "SS",
      "embeddings_str": "fielder primary_position_ss total_games_2747 bats_r throws_r hits_3465 hr_260 batting_avg_310 runs_per_ab_172 2b_avg_49 3b_avg_6 hr_avg_23 rb:
    },
```

Summary

- **Cosmos DB** is a high performance, highly available, distributed, JSON-based NoSQL database that allows for low costs due to the Request Unit cost model
- **TypeScript** is a high-productivity programming language that transpiles to regular JavaScript
- The Azure Cosmos DB **SDKs** provide excellent support as well as modern features such as Asynchronous processing
- The **azu-js** NPM library provides an easy-to-use SDK wrapper as well as code examples for using the SDK
- **Node.js** is a very performant runtime environment
- **Azure** has multiple deployment options for TS/JS apps

Microsoft

Thank you!

Questions?