

## **ASP.NET Core MVC Web Application CRUD operations using Azure Cosmos DB**

In this tutorial I'm going to show how to:

- Create an Azure Cosmos DB Account
- Create an Azure Cosmos DB
- Add a Container to the Cosmos DB
- Create an ASP.NET Core MVC Application
- Add Azure Cosmos DB NuGet package to the Project
- Set up the code for ASP.NET Core MVC application for CRUD Operations
- CRUD Operations Screen Captures

Source code for this Project can be found here:

<https://github.com/patricksameerajayalath/CosmosDBTutorial>

## Step 01: Create an Azure Cosmos DB Account

Click + Add to add new Cosmos DB Account.

The screenshot shows the 'Azure Cosmos DB accounts' page in the Azure portal. At the top, there are navigation links for 'Home' and 'Azure Cosmos DB accounts'. Below the header are several action buttons: '+ Add', 'Edit columns', 'Refresh', 'Export to CSV', 'Assign tags', and 'Feedback'. There are also search and filter options: 'Filter by name...', 'Subscription == all', 'Resource group == all', 'Location == all', and 'Add filter'. The main content area displays a message: 'Showing 0 to 0 of 0 records.' Below this is a table with three columns: 'NAME', 'STATUS', and 'SUBSCRIPTION'. A large, stylized planet icon with a ring is centered in the middle of the page. At the bottom, there is a message: 'No Azure Cosmos DB accounts to display' and a call-to-action button: 'Create Azure Cosmos DB account'.

Provide relevant details and click Review + Create.

- Resource group: patsam-rg
- Account name: patsam-cosmosaccount
- API: Core (SQL)
- Location: Central US

The screenshot shows the 'Create Azure Cosmos DB Account' wizard, step 1: Basics. At the top, there is a banner with the text: 'Create a new Azure Cosmos DB account with multi-region writes in North Europe, West Europe, South Central US, or North Central US by November 30, 2019 and receive up to 33% off for the life of your account. Restrictions apply.' Below the banner, there are tabs for 'Basics', 'Network', 'Tags', and 'Review + create'. The 'Basics' tab is selected. It contains fields for 'Project Details': 'Subscription' (Visual Studio Enterprise - MPN) and 'Resource Group' (patsam-rg). Under 'Instance Details', there are fields for 'Account Name' (patsam-cosmosaccount), 'API' (Core (SQL)), and 'Location' ((US) Central US). There are also sections for 'Geo-Redundancy' (Enable/Disable) and 'Multi-region Writes' (Enable/Disable). At the bottom of the form, there is a note about the offer and a 'Review + create' button.

*Note: We have used Core (SQL) as the API for this Cosmos DB Account.*

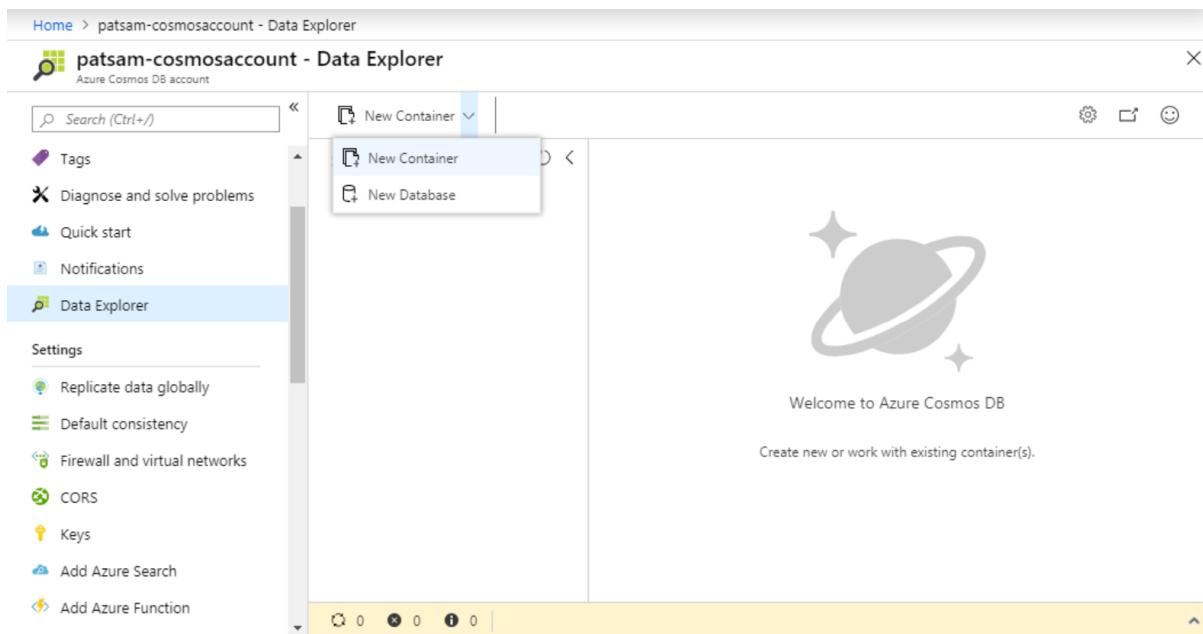
Once the deployment finishes, we can see the newly created Cosmos DB Account.

The screenshot shows two views of the Azure portal. The top view is the 'Azure Cosmos DB accounts' list, showing one record: 'patsam-cosmosaccount' (Status: Online, Location: Central US, Subscription: Visual Studio Enterprise – MPN). The bottom view is the detailed 'Overview' page for the 'patsam-cosmosaccount'. It displays basic account information: Status (Online), Resource group (patsam-rg), Subscription (Visual Studio Enterprise – MPN), and Subscription ID (3822b9c0 bf7c 455c a1d1 8717b1b4150b). It also shows sections for Containers (empty), Monitoring (last 24 hours), and Requests (empty).

## **Step 02: Create an Azure Cosmos DB**

Next, we need to add a Cosmos DB. Click add New Database.

Data Explorer → New Database



The screenshot shows the Azure Data Explorer interface for the 'patsam-cosmosaccount' database. On the left, there's a sidebar with various settings like Tags, Diagnose and solve problems, Quick start, Notifications, and Data Explorer (which is selected). The main area has a search bar at the top. Below it, there's a dropdown menu with 'New Container' and 'New Database' options. A large, stylized planet icon with a ring is centered in the main area, with the text 'Welcome to Azure Cosmos DB' below it. At the bottom, there are some status indicators.

Provide relevant details and Click OK.

- Database id: patsam-cosmosdb

### New Database

**i** Start at \$24/mo per database, multiple containers included  
[More details](#)

\* Database id ⓘ  
patsam-cosmosdb

Provision throughput ⓘ

\* Throughput (400 - 100,000 RU/s) ⓘ  
400 —  
+

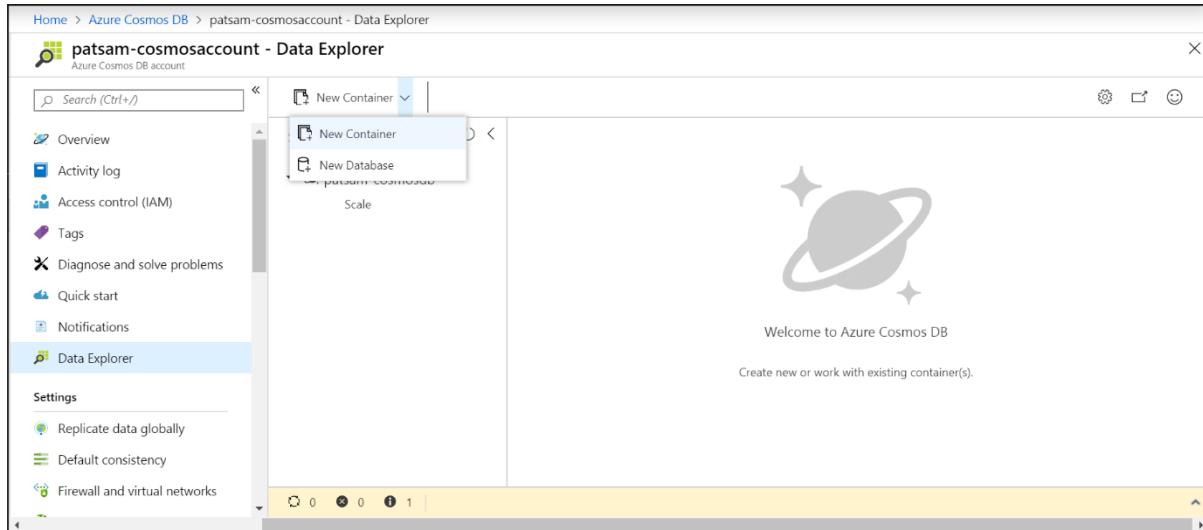
Estimated spend (USD): **\$0.032 hourly / \$0.77 daily** (1 region, 100RU/s, 100000RU/s)

**OK**

### **Step 03: Add a Container to the Cosmos DB**

Next, we need to add a Container. Click on New Container.

Data Explorer → New Container



Provide relevant data and Click OK.

- Database id: patsam-cosmosdb (select the Database we created earlier)
- Container id: Employee
- Partition key: /id

## Add Container

X



Start at \$24/mo per database, multiple containers included  
[More details](#)

\* Database id ⓘ

Create new  Use existing

patsam-cosmosdb



\* Container id ⓘ

Employee

\* Partition key ⓘ

/id

My partition key is larger than 100 bytes

Provision dedicated throughput for this container ⓘ

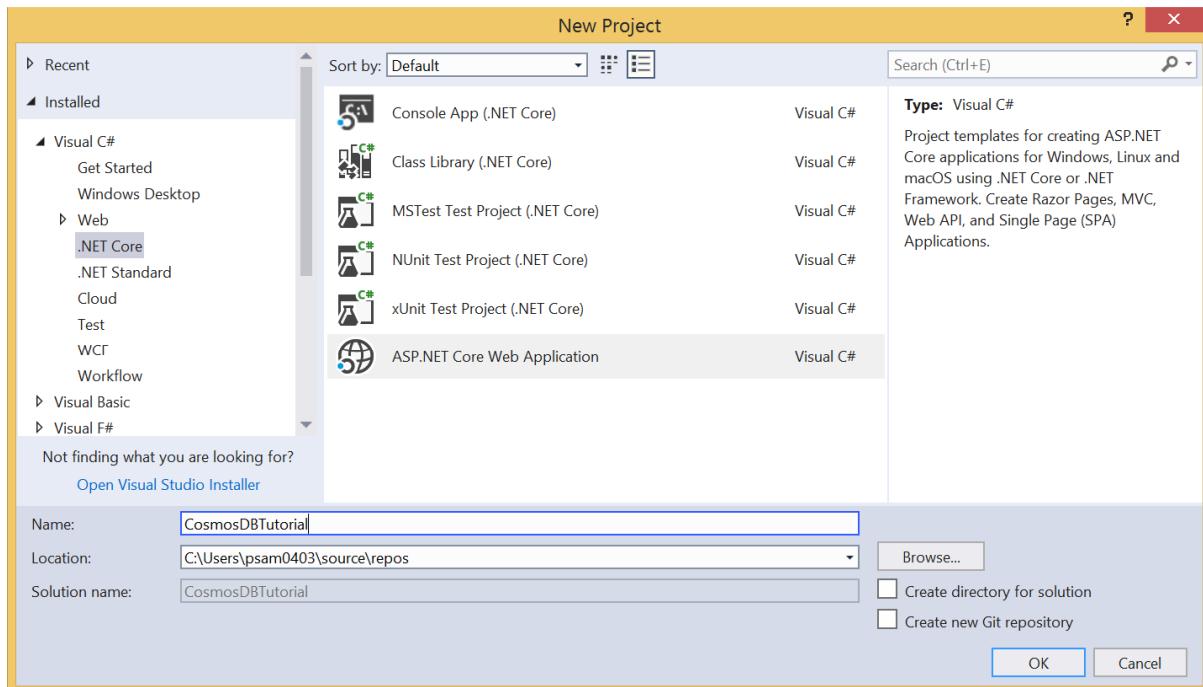
Unique keys ⓘ

+ Add unique key

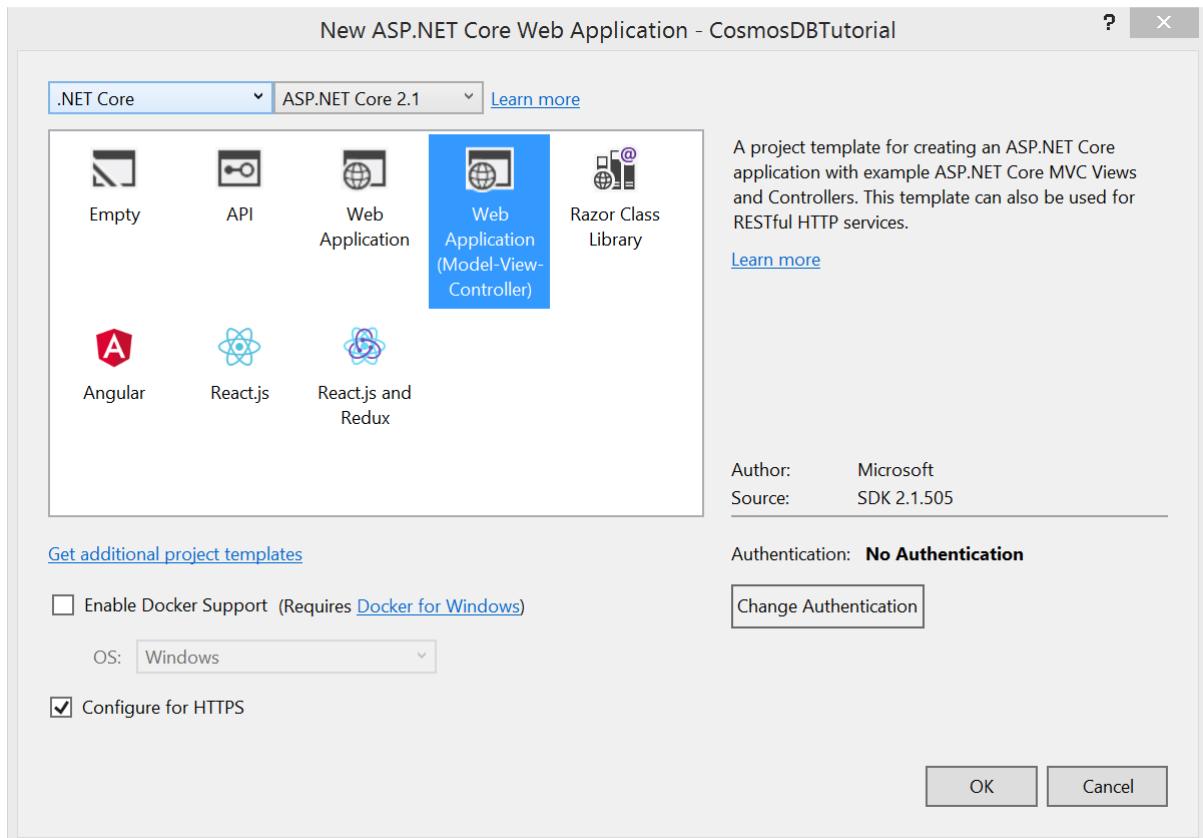
OK

## Step 04: Create an ASP.NET Core MVC Application

Create a new .Net Core Web Project.



Select MVC.



Once Project gets created Click F5 to run the Project, just to make sure it compiles properly.

The screenshot shows the Microsoft Visual Studio interface for a project named "CosmosDBTutorial". The left pane displays the "Overview" of the ASP.NET Core project, featuring sections for "Build Your App", "Connect To The Cloud", and "Learn Your IDE", along with links to documentation and productivity guides. The right pane shows the "Solution Explorer" with a tree view of the project files, including "Connected Services", "Dependencies", "Properties", "wwwroot", "Controllers", "Models", "Views", "appsettings.json", "Program.cs", and "Startup.cs". Below the main window, the status bar shows "Ready".

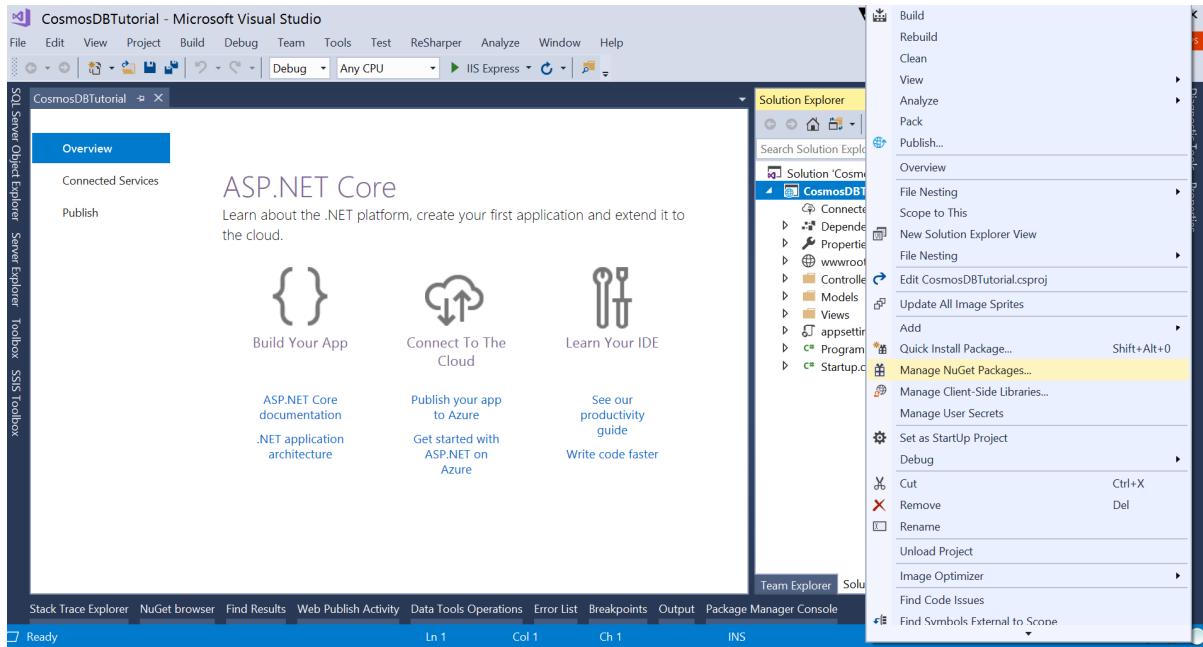
  

The screenshot shows a web browser window titled "Home Page - CosmosDBTutorial" at the URL <https://localhost:44309>. The page features a purple header with the Visual Studio logo and icons for HTML, CSS, and JS. The main content area includes a message about new features for building modern web apps, a "Learn More" button, and a navigation bar with links like Apps, Azure, DevOps, TimeSheets, RXP HR, MSN, OzBargain, CommSec, Udemy, Motley, Azure Code Sample..., C# to VB.Net, and FAC - DEV.

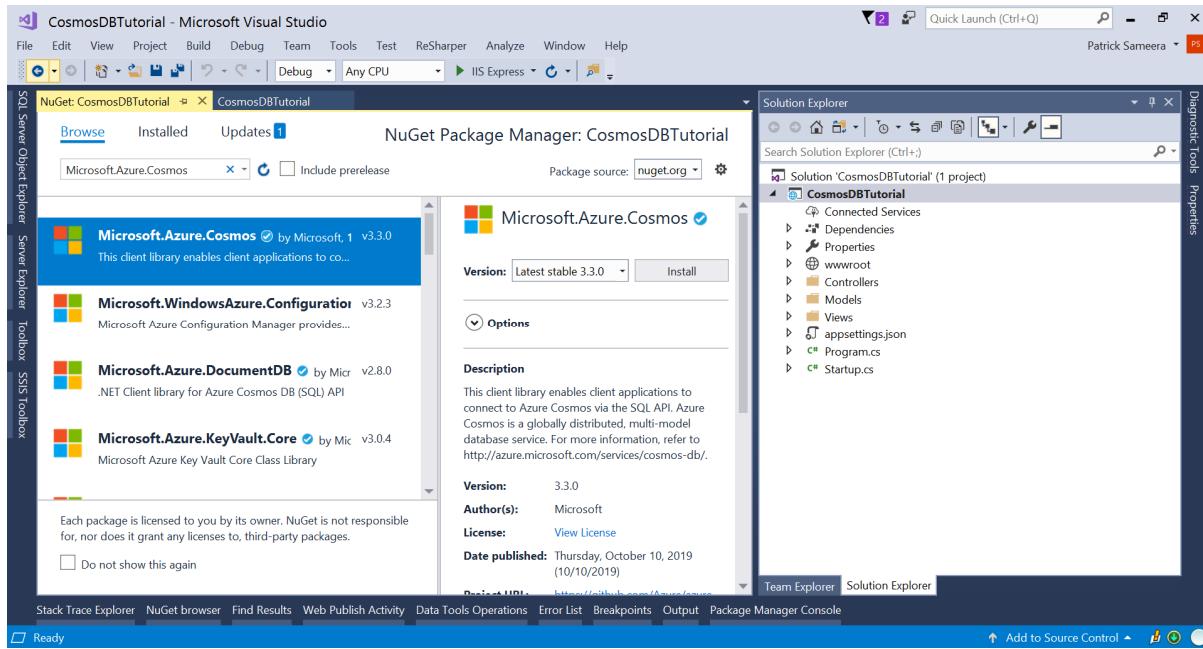
Application uses	How to	Overview	Run & Deploy
<ul style="list-style-type: none"><li>Sample pages using ASP.NET Core MVC</li><li>Theming using Bootstrap</li></ul>	<ul style="list-style-type: none"><li>Add a Controller and View</li><li>Manage User Secrets using Secret Manager.</li><li>Use logging to log a message.</li><li>Add packages using NuGet.</li></ul>	<ul style="list-style-type: none"><li>Conceptual overview of what is ASP.NET Core</li><li>Fundamentals of ASP.NET Core such as Startup and middleware.</li><li>Working with Data</li></ul>	<ul style="list-style-type: none"><li>Run your app</li><li>Run tools such as EF migrations and more</li><li>Publish to Microsoft Azure Web Apps</li></ul>

## Step 05: Add Azure Cosmos DB NuGet package to the Project

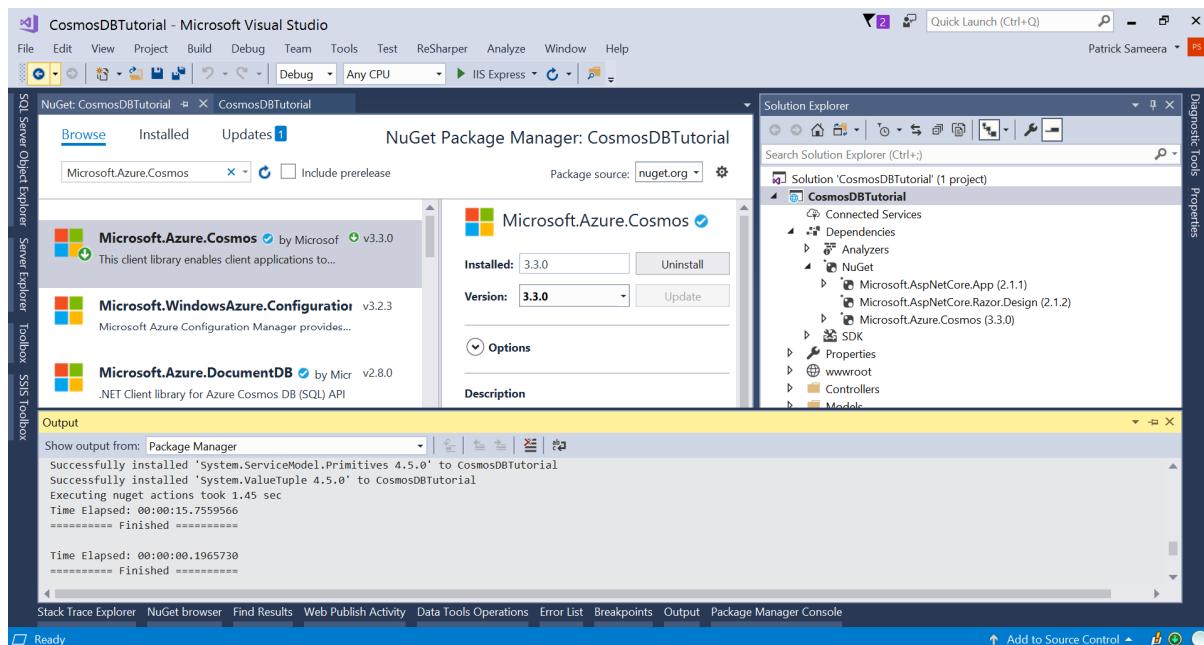
In the Solution Explorer, right click on the Project and select Manage NuGet Packages.



Under Browse tab Search for Microsoft.Azure.Cosmos and on the results Select it and Click Install.



Make sure it gets installed successfully.



## Step 06: Set up the code for ASP.NET Core MVC application for CRUD Operations

We are going to create Web Application to add new Employees. We will be having bellow structure for an employee.

Column Name	Type	.Net Control
Name	String	Text box
Age	Int	Text box
DOB	Date	Text box
Manager	Bool	Check box

We are trying to achieve something like bellow:

The screenshot shows a web browser window with the title "Index - CosmosDBTutorial". The URL in the address bar is "https://localhost:44309/employee". The page content is the "Index" view of an ASP.NET Core MVC application. It features a table with the following data:

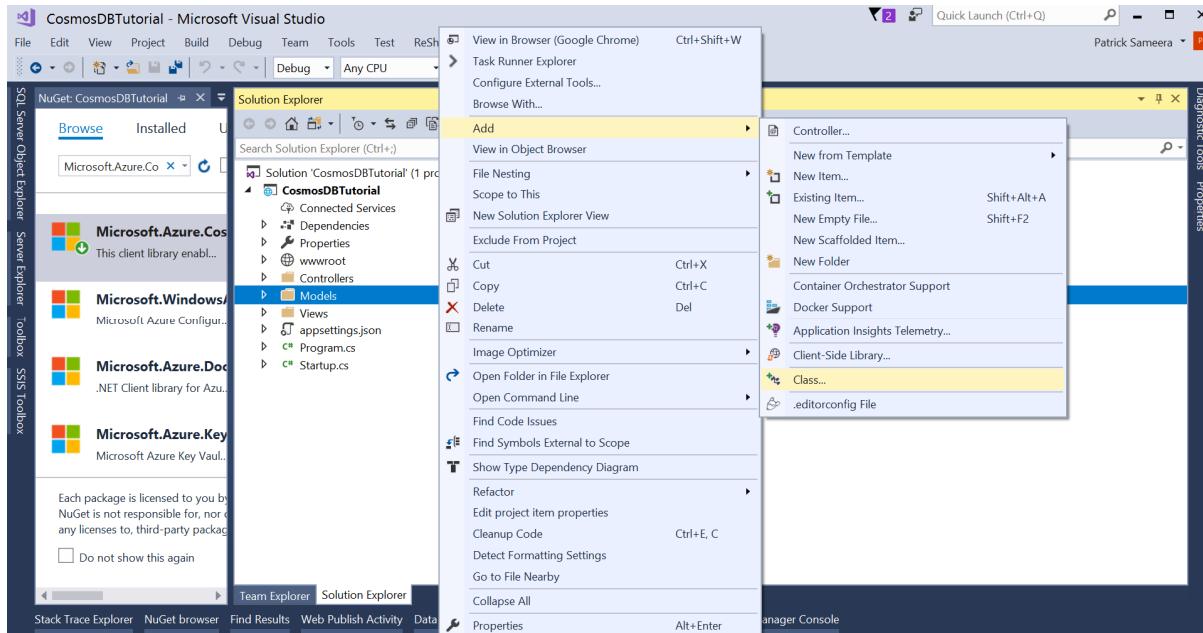
Id	Name	Age	DOB	Manager
211f027a-319a-4ae8-b49e-a65635ea73ce	Patrick Sameera	30	5/05/2000	<input checked="" type="checkbox"/>

Below the table, there is a link "Edit | Details | Delete". At the bottom of the page, a copyright notice reads "© 2019 - CosmosDBTutorial".

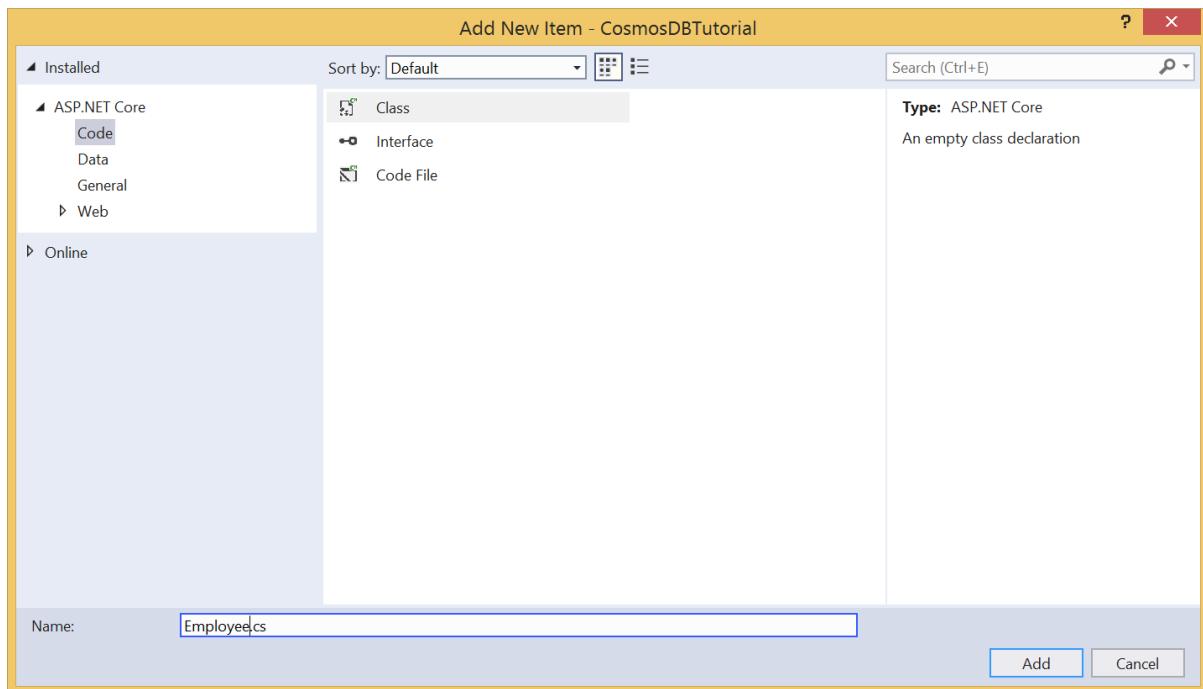
## Adding Model

We need to add a Class. Right Click on Models Folder.

- Models → Add → Class

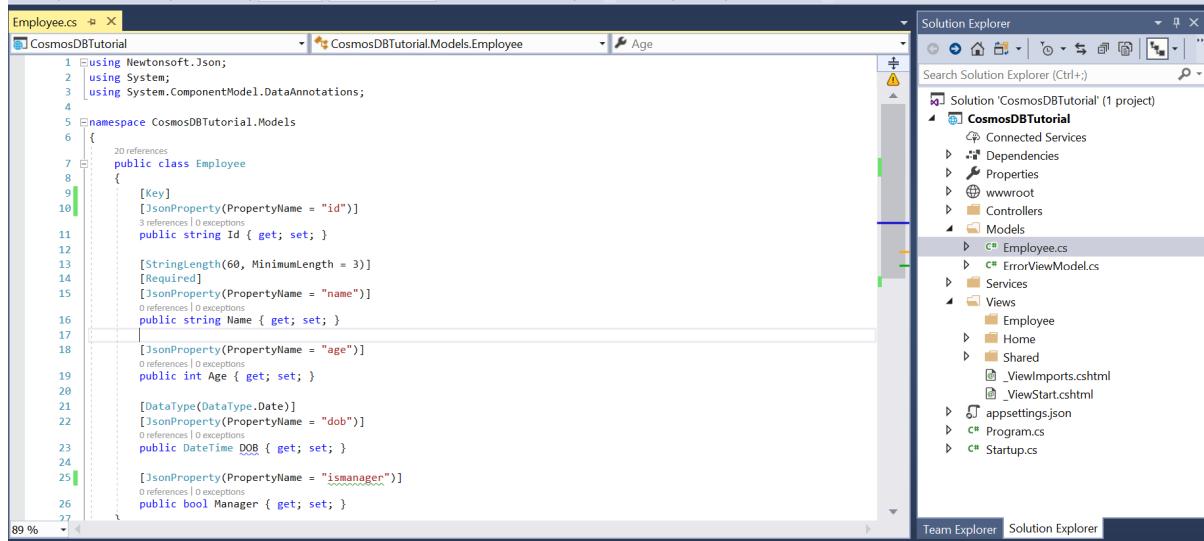


Name it Employee.cs



## Employee.cs

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Models/Employee.cs>



The screenshot shows the Visual Studio IDE interface. On the left is the code editor window titled "Employee.cs" under the "CosmosDBTutorial" project. The code defines a class "Employee" with properties: Id, Name, Age, DOB, and Manager, each annotated with JSON serialization attributes like [JsonProperty(PropertyName = "id")]. On the right is the "Solution Explorer" window, which displays the project structure for "CosmosDBTutorial". The "Models" folder contains the "Employee.cs" file. Other files visible include "ErrorViewModel.cs", "Services", "Views" (containing "Employee", "Home", "Shared", "\_ViewImports.cshtml", and "\_ViewStart.cshtml"), and "Controllers", "Properties", and "Startup.cs".

```
Employee.cs
1 using Newtonsoft.Json;
2 using System;
3 using System.ComponentModel.DataAnnotations;
4
5 namespace CosmosDBTutorial.Models
6 {
7     public class Employee
8     {
9         [Key]
10        [JsonProperty(PropertyName = "id")]
11        public string Id { get; set; }
12
13        [StringLength(60, MinimumLength = 3)]
14        [Required]
15        [JsonProperty(PropertyName = "name")]
16        public string Name { get; set; }
17
18        [JsonProperty(PropertyName = "age")]
19        public int Age { get; set; }
20
21        [DataType(DataType.Date)]
22        [JsonProperty(PropertyName = "dob")]
23        public DateTime DOB { get; set; }
24
25        [JsonProperty(PropertyName = "ismanager")]
26        public bool Manager { get; set; }
27    }
}
```

*Note: Azure Cosmos DB uses JSON to move and store data. You can use the `JsonProperty` attribute to control how JSON serializes and deserializes objects.*

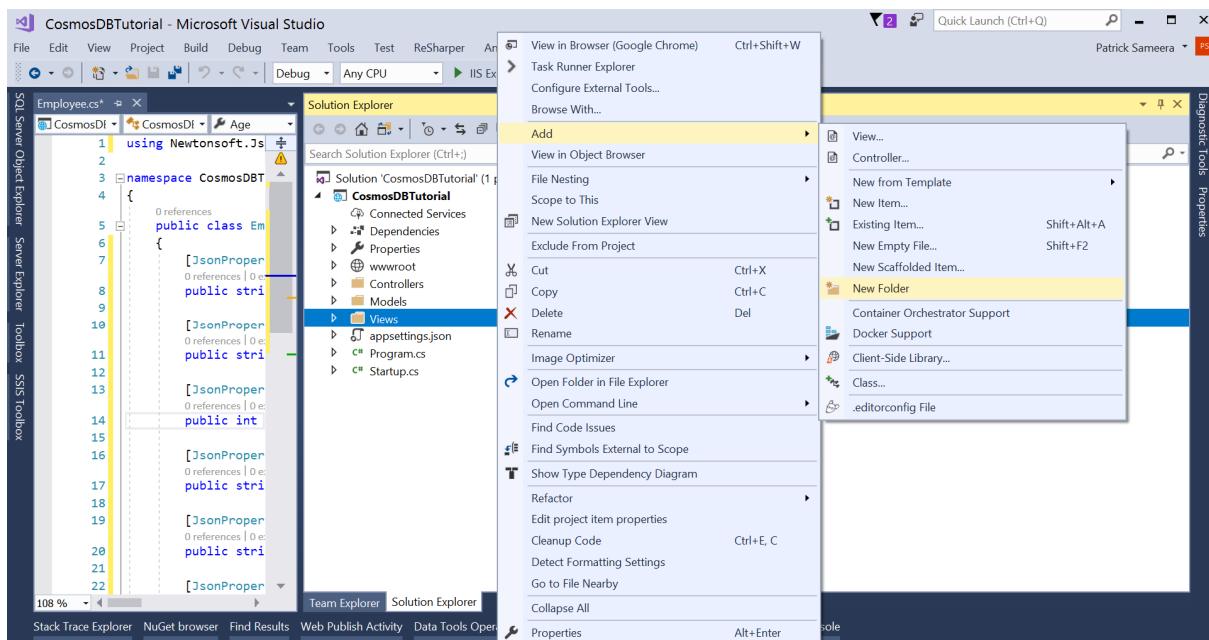
## Adding Views

Next, we need to add 5 Views.

- Employee list view: Index
- Add Employee view: Create
- Edit Employee view: Edit
- Detail Employee view: Detail
- Delete Employee view: Delete

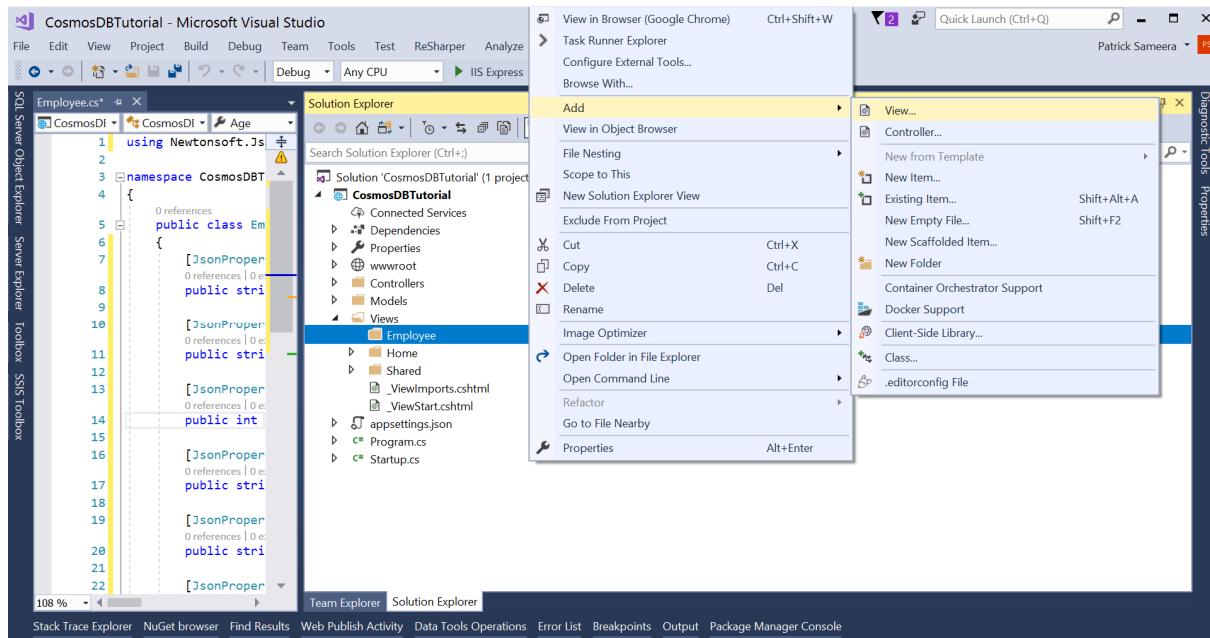
First create a new Folder under Views by the name Employee. Right Click on Views.

- Views → Add → New Folder → Employee



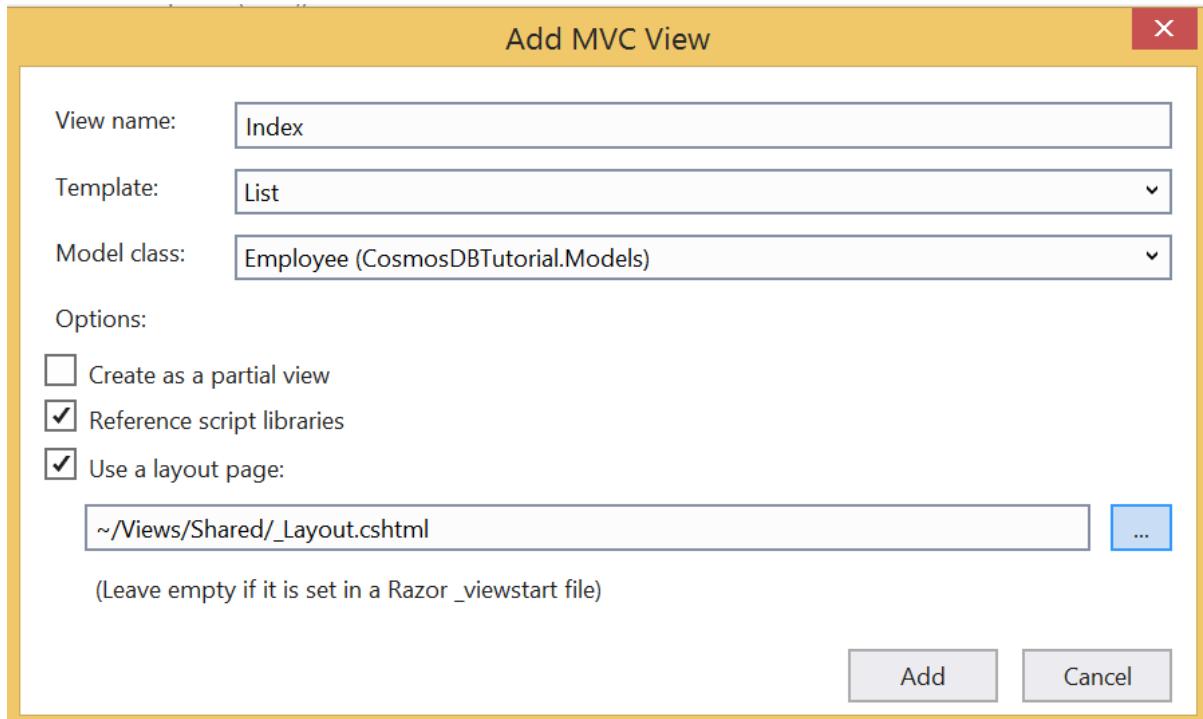
Next, we need to add the 5 Views. Right Click on Employee Folder under Views to add Views.

- Views → Employees → Add → View

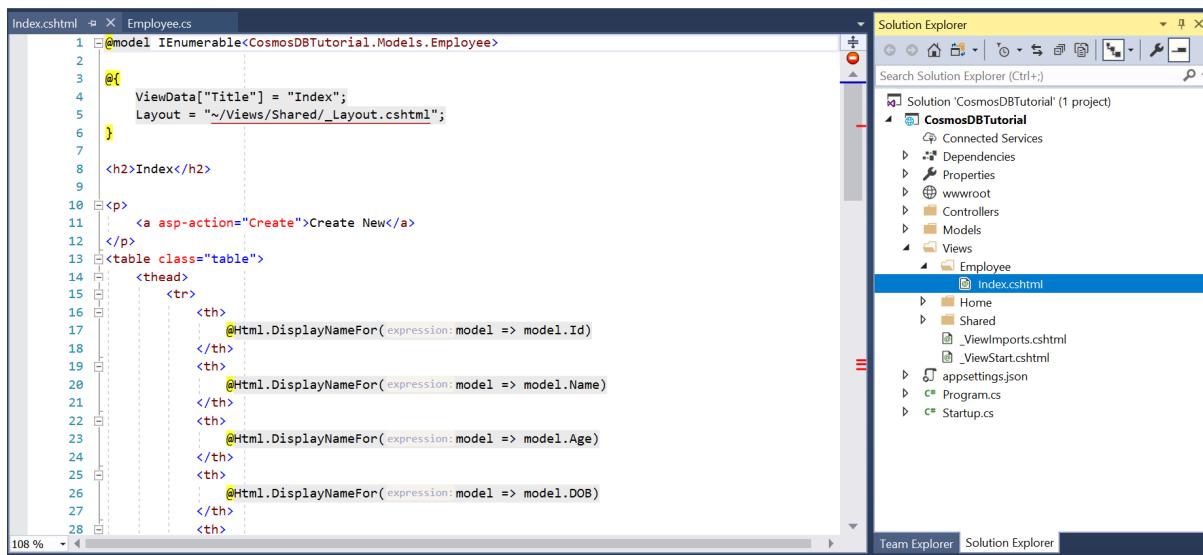


## View 01: Index

- Name: Index
- Template: List
- Model: Employee

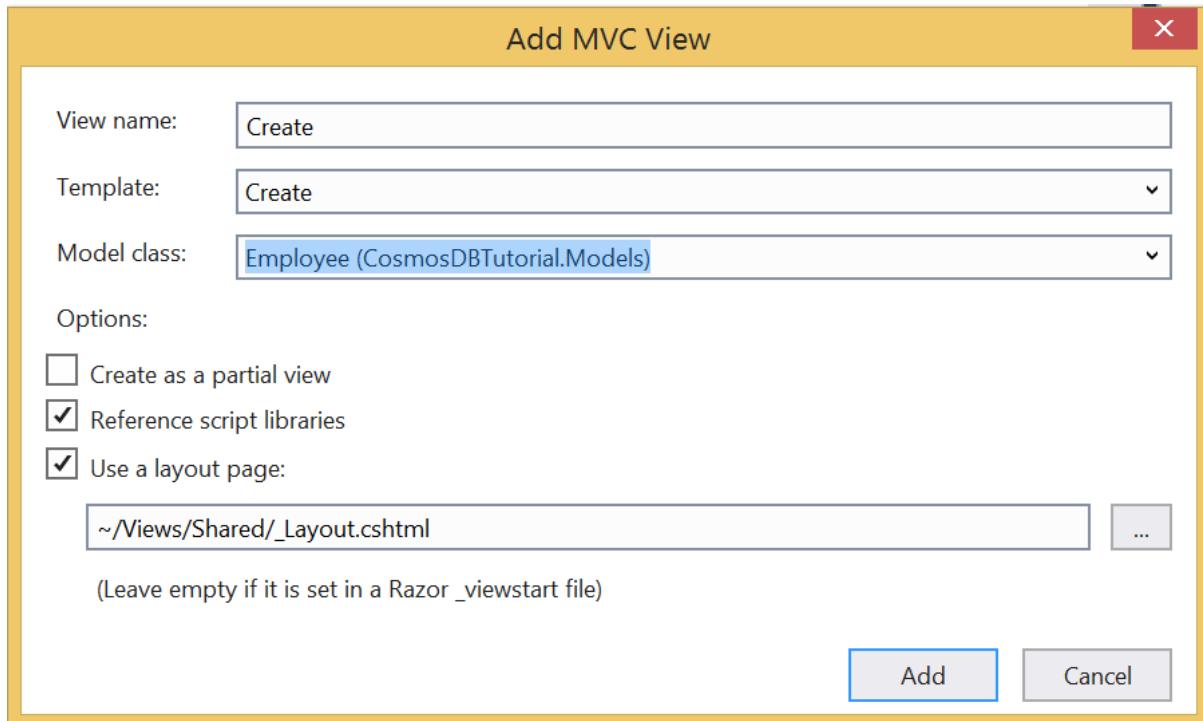


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Index.cshtml>

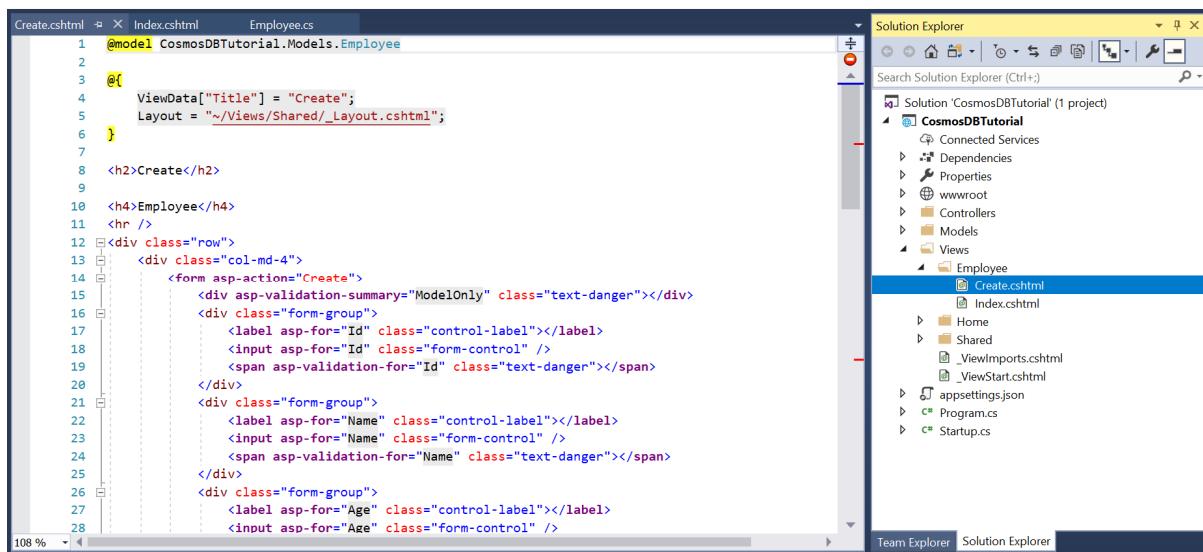


## View 02: Create

- Name: Create
- Template: Create
- Model: Employee

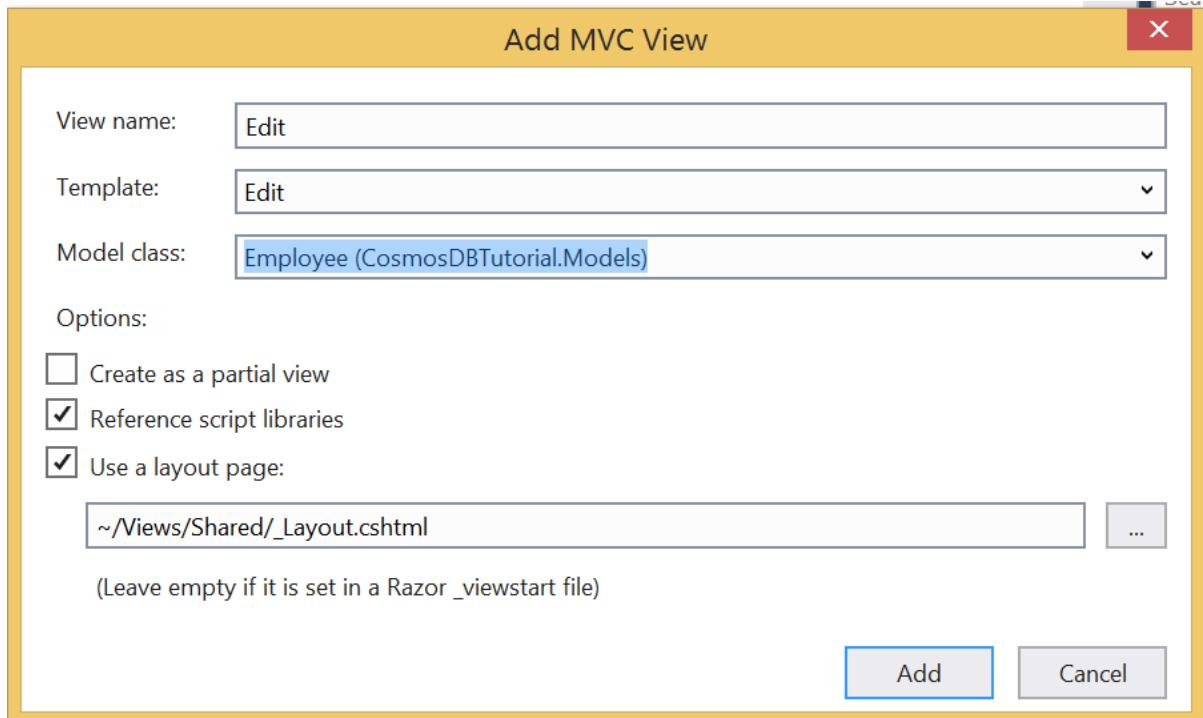


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Create.cshtml>

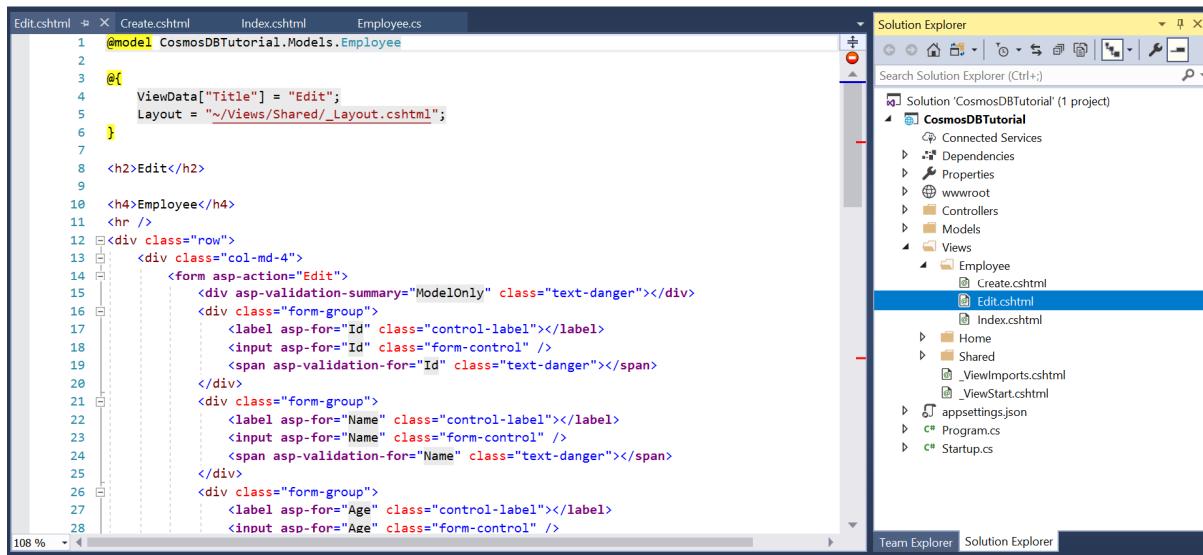


### View 03: Edit

- Name: Edit
- Template: Edit
- Model: Employee

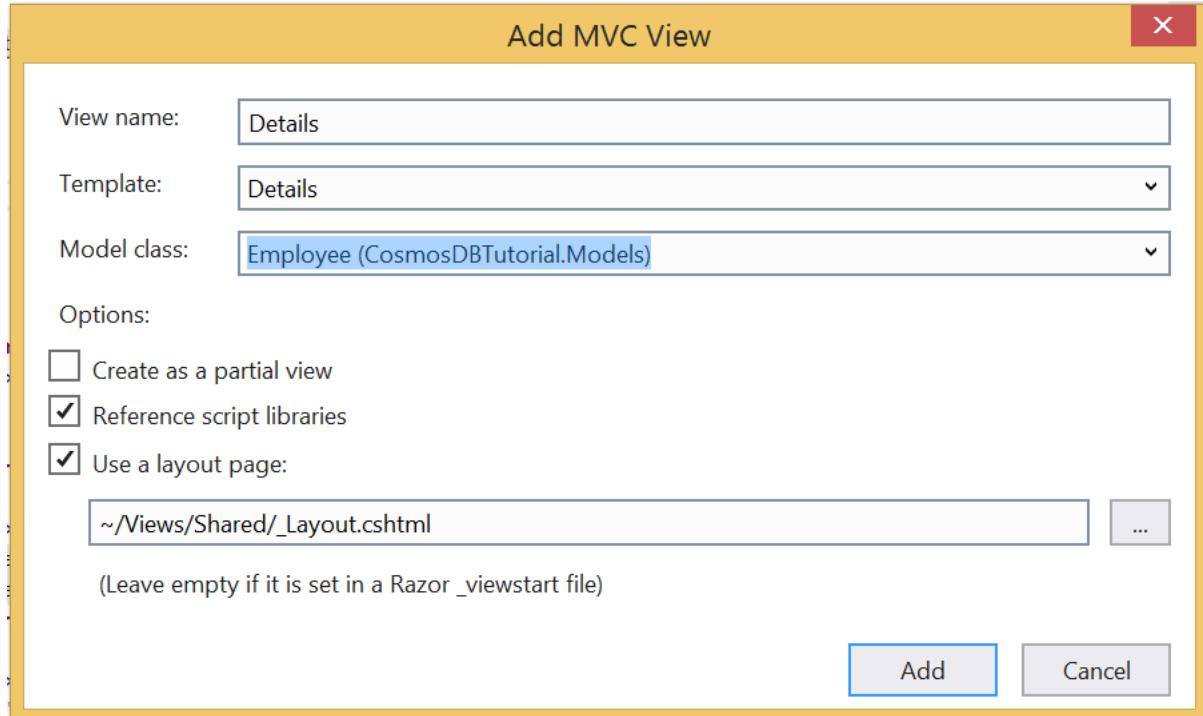


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Edit.cshtml>

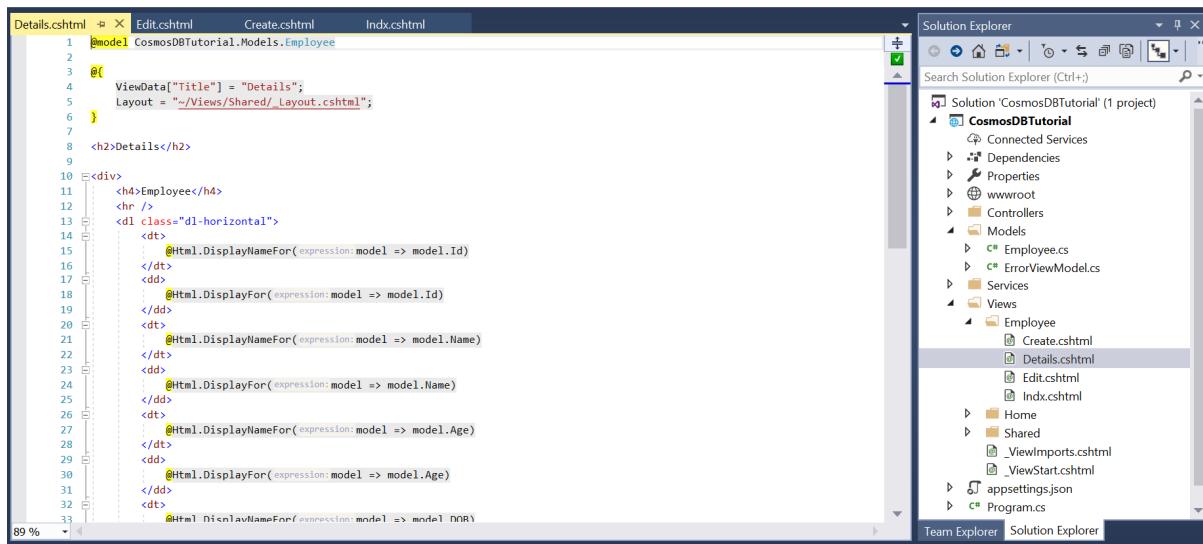


#### View 04: Details

- Name: Details
- Template: Details
- Model: Employee

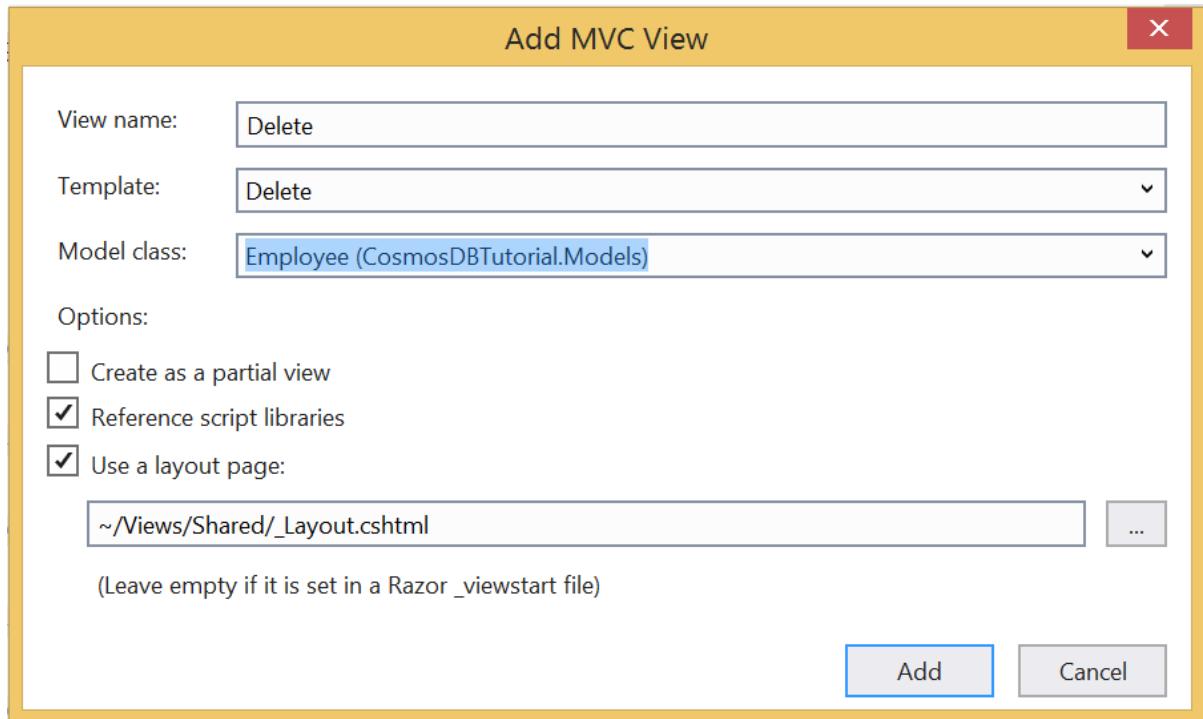


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Details.cshtml>

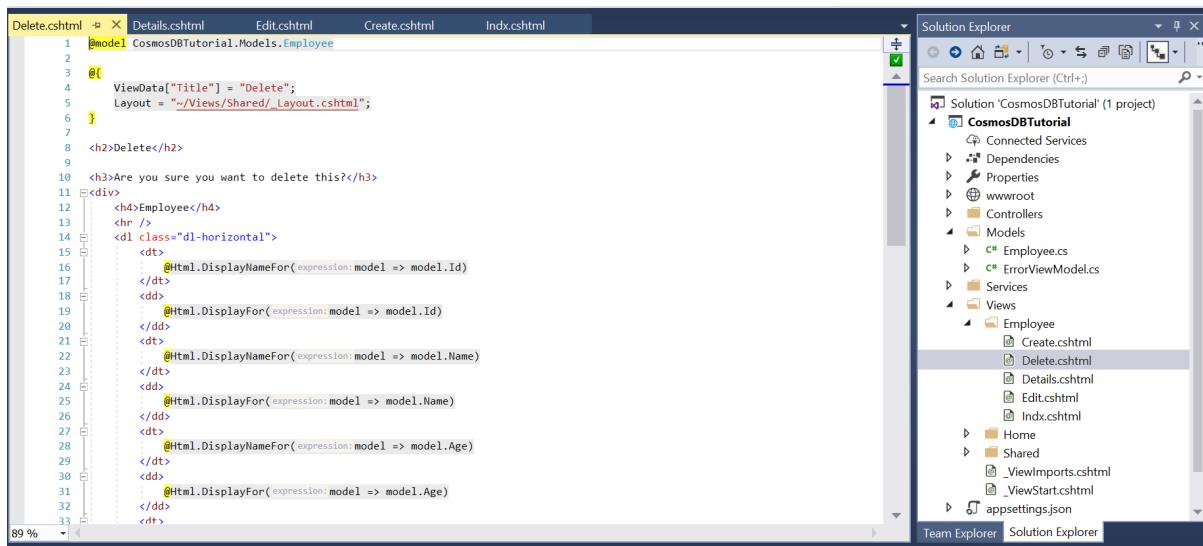


## View 04: Delete

- Name: Delete
- Template: Delete
- Model: Employee



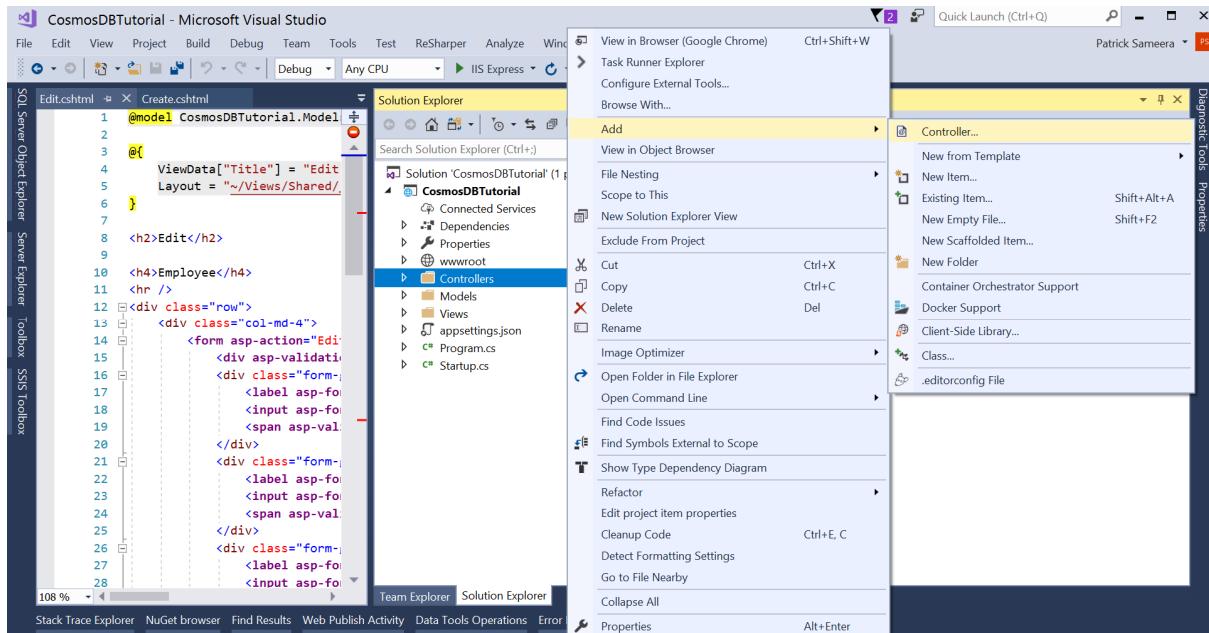
<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Delete.cshtml>



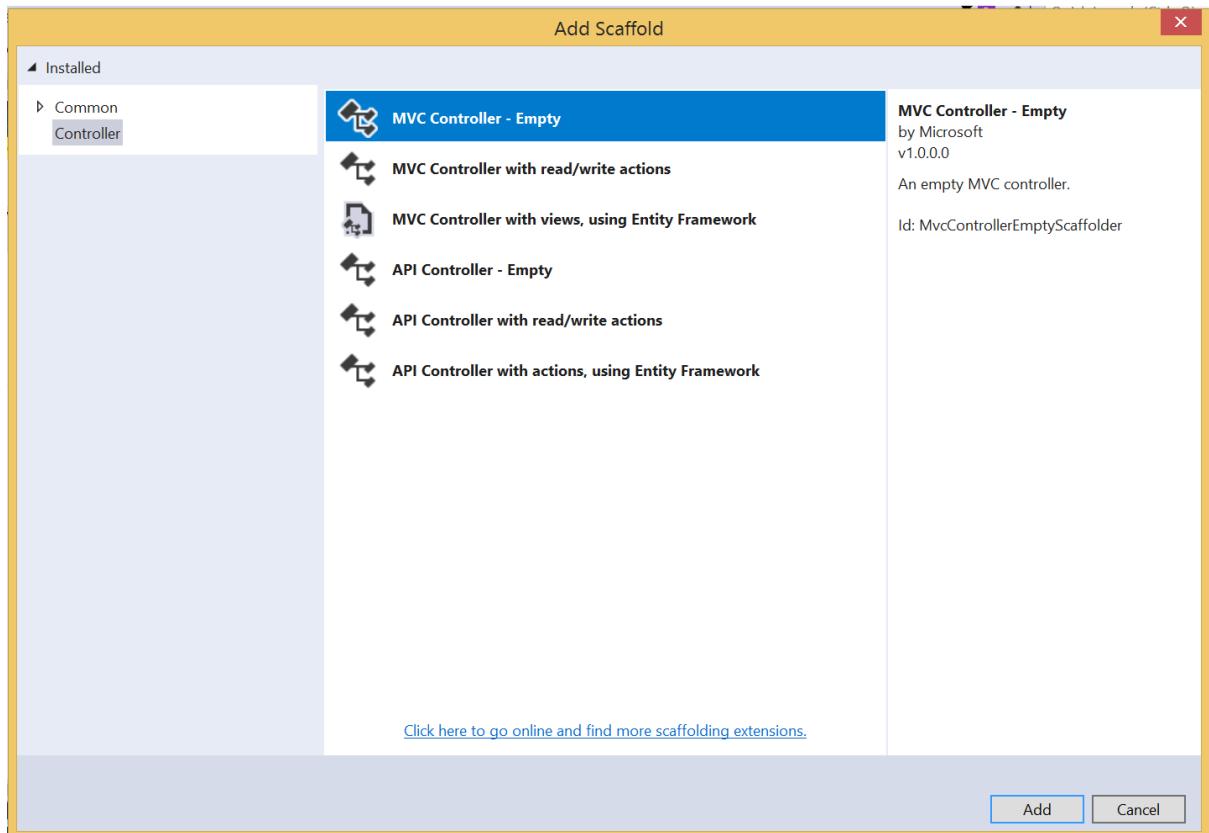
## Adding Controllers

Next, we need to add a Controller by the name Employee. Right Click on Controllers.

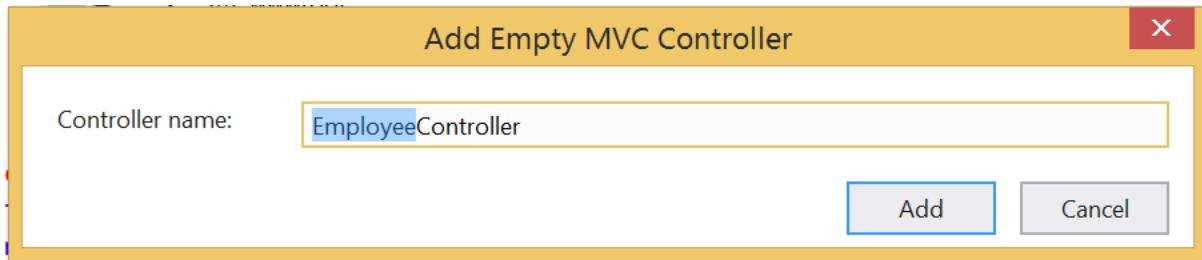
Controllers → Add → Controller



Select MVC Controller – Empty and Click Add.



Name it EmployeeController.



EmployeeController.cs

Copy paste the code to the file.

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Controllers/EmployeeController.cs>

A screenshot of the Visual Studio IDE. On the left, the code editor shows the "EmployeeController.cs" file with C# code. On the right, the "Solution Explorer" pane shows the project structure for "CosmosDBTutorial". The "Controllers" folder contains "EmployeeController.cs" and "HomeController.cs". Other files like "Models", "Services", "Views", "appsettings.json", "Program.cs", and "Startup.cs" are also listed. The status bar at the bottom indicates "89 %".

## Index

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Controllers/EmployeeController.cs#L17>

```
17     [ActionName("Index")]
18     public async Task<IActionResult> Index()
19     {
20         return View(await _cosmosDbService.GetItemsAsync("SELECT * FROM c"));
21     }
```

## Details

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Controllers/EmployeeController.cs#L101>

```
101    [ActionName("Details")]
102    public async Task<ActionResult> DetailsAsync(string id)
103    {
104        return View(await _cosmosDbService.GetItemAsync(id));
105    }
```

## Create

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Controllers/EmployeeController.cs#L23>

```
23     [ActionName("Create")]
24     public IActionResult Create()
25     {
26         return View();
27     }
28
29     [HttpPost]
30     [ActionName("Create")]
31     [ValidateAntiForgeryToken]
32     public async Task<ActionResult> CreateAsync([Bind("Id,Name,Age,DOB,Manager")] Employee employee)
33     {
34         if (ModelState.IsValid)
35         {
36             employee.Id = Guid.NewGuid().ToString();
37             await _cosmosDbService.AddItemAsync(employee);
38             return RedirectToAction("Index");
39         }
40
41         return View(employee);
42     }
```

## Edit

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Controllers/EmployeeController.cs#L45>

```
45     [ActionName("Edit")]
46     [ValidateAntiForgeryToken]
47     public async Task<ActionResult> EditAsync([Bind("Id,Name,Age,DOB,Manager")] Employee employee)
48     {
49         if (ModelState.IsValid)
50         {
51             await _cosmosDbService.UpdateItemAsync(employee.Id, employee);
52             return RedirectToAction("Index");
53         }
54
55         return View(employee);
56     }
57
58     [ActionName("Edit")]
59     public async Task<ActionResult> EditAsync(string id)
60     {
61         if (id == null)
62         {
63             return BadRequest();
64         }
65
66         Employee employee = await _cosmosDbService.GetItemAsync(id);
67         if (employee == null)
68         {
69             return NotFound();
70         }
71
72         return View(employee);
73     }
```

## Delete

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Controllers/EmployeeController.cs#L75>

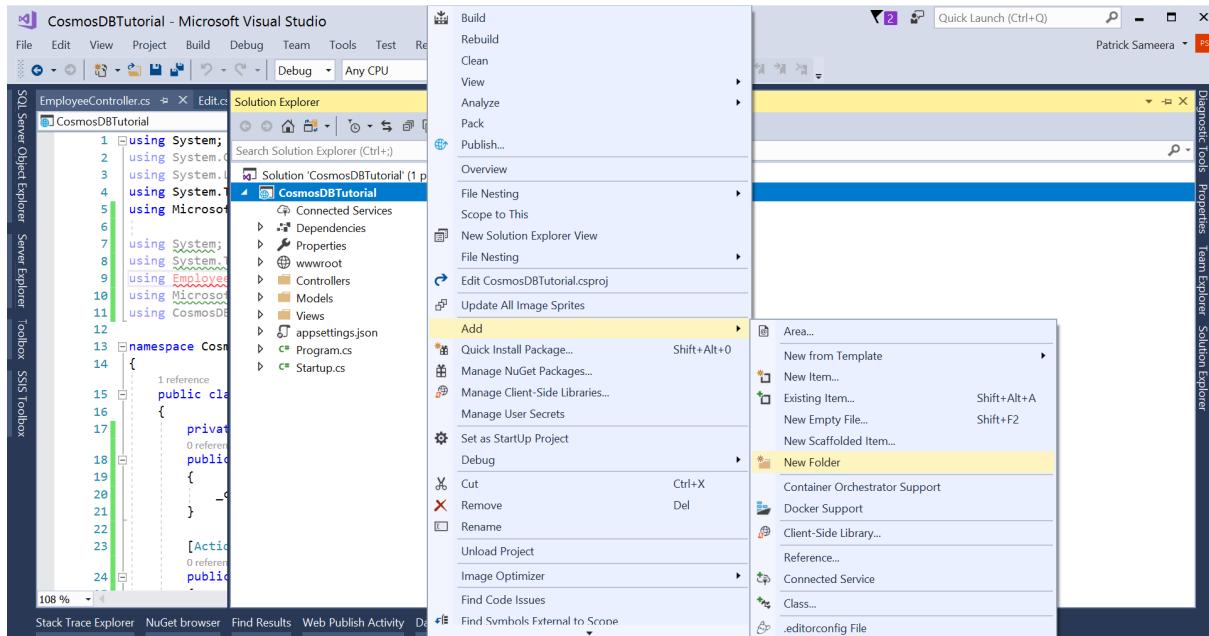
```
75     [ActionName("Delete")]
76     public async Task<ActionResult> DeleteAsync(string id)
77     {
78         if (id == null)
79         {
80             return BadRequest();
81         }
82
83         Employee employee = await _cosmosDbService.GetItemAsync(id);
84         if (employee == null)
85         {
86             return NotFound();
87         }
88
89         return View(employee);
90     }
91
92     [HttpPost]
93     [ActionName("Delete")]
94     [ValidateAntiForgeryToken]
95     public async Task<ActionResult> DeleteConfirmedAsync([Bind("Id")] string id)
96     {
97         await _cosmosDbService.DeleteItemAsync(id);
98         return RedirectToAction("Index");
99     }
```

## Connect to Azure Cosmos DB

Next, we need to add a Class to connect to Azure Cosmos DB. It will be called CosmosDBService and an Interface called ICosmosDBService.

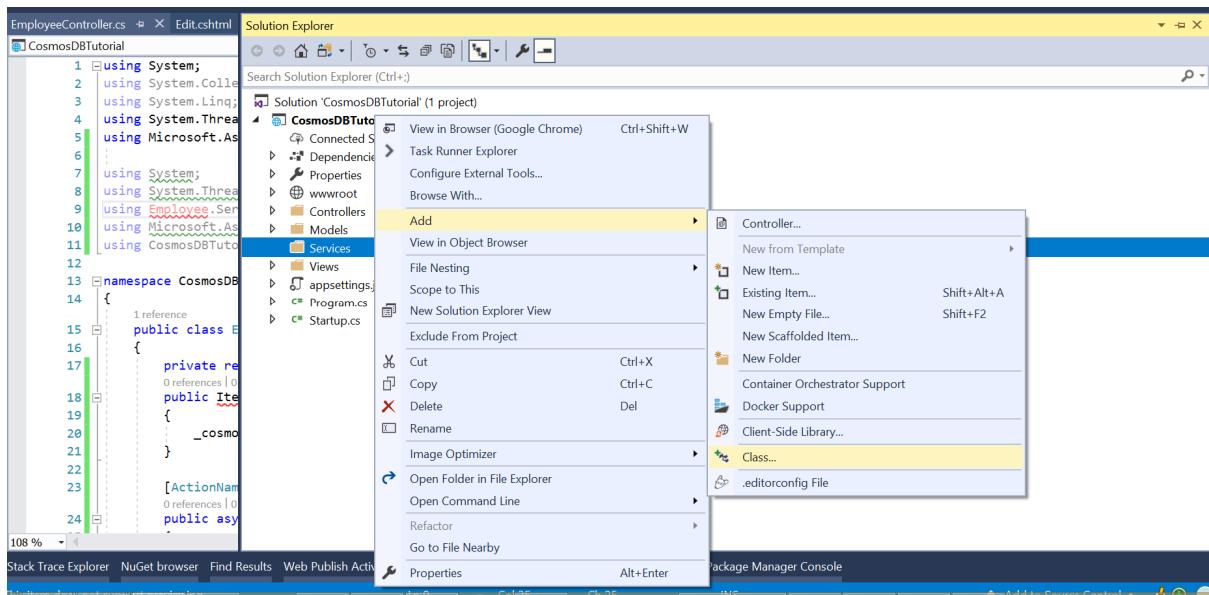
First Add a new Folder called Services. Right Click on the Solution.

- Add → New Folder



Under Services Folder add new Class by the name CosmosDBService.cs

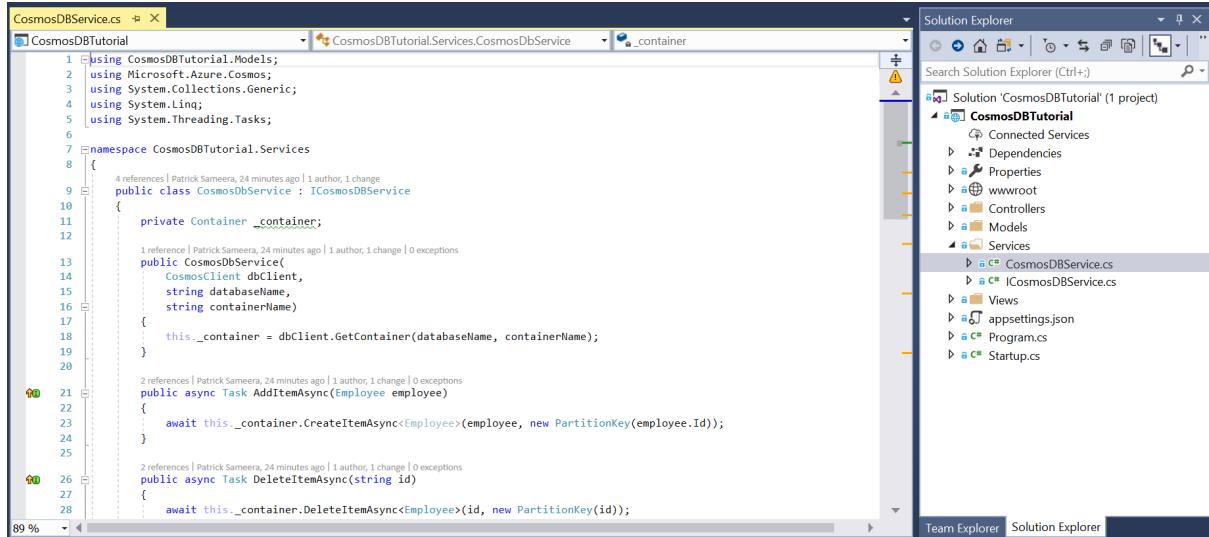
- Services → Add → Class



## CosmosDBService.cs

Copy paste the code to the file.

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Services/CosmosDBService.cs>



The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the `CosmosDBService.cs` file. The code defines a class `CosmosDbService` that implements the `ICosmosDBService` interface. It uses the `Container` class from the `CosmosDBTutorial.Models` namespace and the `CosmosClient` class from the `Microsoft.Azure.Cosmos` namespace. The `AddItemAsync` method creates a new item in the database, and the `DeleteItemAsync` method removes an item by its ID. On the right, the Solution Explorer pane shows the project structure for 'CosmosDBTutorial'. It includes a `Solution` node, a `CosmosDBTutorial` node containing `Connected Services`, `Dependencies`, `Properties`, `wwwroot`, `Controllers`, `Models`, and a `Services` node which contains the `CosmosDBService.cs` file and its interface `ICosmosDBService.cs`. Other files like `Views`, `appsettings.json`, `Program.cs`, and `Startup.cs` are also listed.

```
1 //using CosmosDBTutorial.Models;
2 //using Microsoft.Azure.Cosmos;
3 //using System.Collections.Generic;
4 //using System.Linq;
5 //using System.Threading.Tasks;
6
7 namespace CosmosDBTutorial.Services
8 {
9     4 references | Patrick Sameera, 24 minutes ago | 1 author, 1 change
10    public class CosmosDbService : ICosmosDBService
11    {
12        private Container _container;
13
14        1 reference | Patrick Sameera, 24 minutes ago | 1 author, 1 change | 0 exceptions
15        public CosmosDbService(
16            CosmosClient dbClient,
17            string databaseName,
18            string containerName)
19        {
20            this._container = dbClient.GetContainer(databaseName, containerName);
21
22            2 references | Patrick Sameera, 24 minutes ago | 1 author, 1 change | 0 exceptions
23            public async Task AddItemAsync(Employee employee)
24            {
25                await this._container.CreateItemAsync<Employee>(employee, new PartitionKey(employee.Id));
26
27            2 references | Patrick Sameera, 24 minutes ago | 1 author, 1 change | 0 exceptions
28            public async Task DeleteItemAsync(string id)
29            {
30                await this._container.DeleteItemAsync<Employee>(id, new PartitionKey(id));
31            }
32        }
33    }
34}
```

Next under Services Folder add new Class by the name ICosmosDBService.cs

- Services → Add → Class

### ICosmosDBService.cs

Copy paste the code to the file.

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Services/ICosmosDBService.cs>

The screenshot shows the Visual Studio IDE interface. On the left is the code editor window displaying the `ICosmosDBService.cs` file. The code defines a public interface `ICosmosDBService` with methods for getting items, adding items, updating items, and deleting items. On the right is the Solution Explorer window, which shows the project structure for `CosmosDBTutorial`. The `Services` folder contains the `CosmosDBService.cs` and `ICosmosDBService.cs` files. Other folders like `Controllers`, `Models`, `Views`, and configuration files are also visible.

```
1 using System.Collections.Generic;
2 using System.Threading.Tasks;
3 using CosmosDBTutorial.Models;
4
5 namespace CosmosDBTutorial.Services
6 {
7     public interface ICosmosDBService
8     {
9         Task<IEnumerable<Employee>> GetItemsAsync(string query);
10        Task<Employee> GetItemAsync(string id);
11        Task AddItemAsync(Employee employee);
12        Task UpdateItemAsync(string id, Employee employee);
13        Task DeleteItemAsync(string id);
14    }
15}
16
```

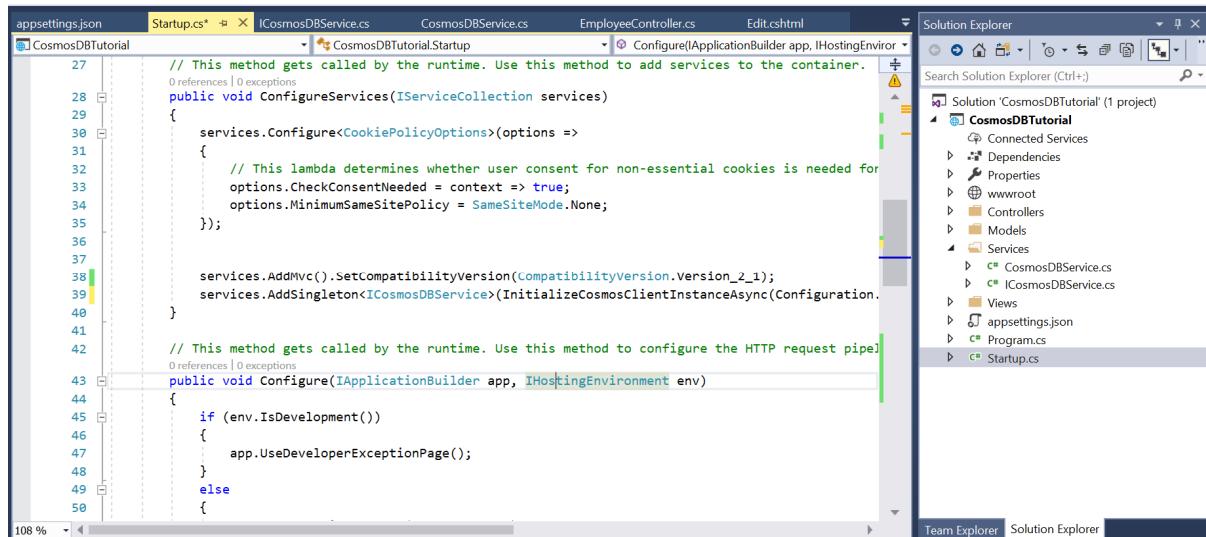
## Startup.cs

Inside Startup.cs file in the ConfigureServices handler, add the following line:

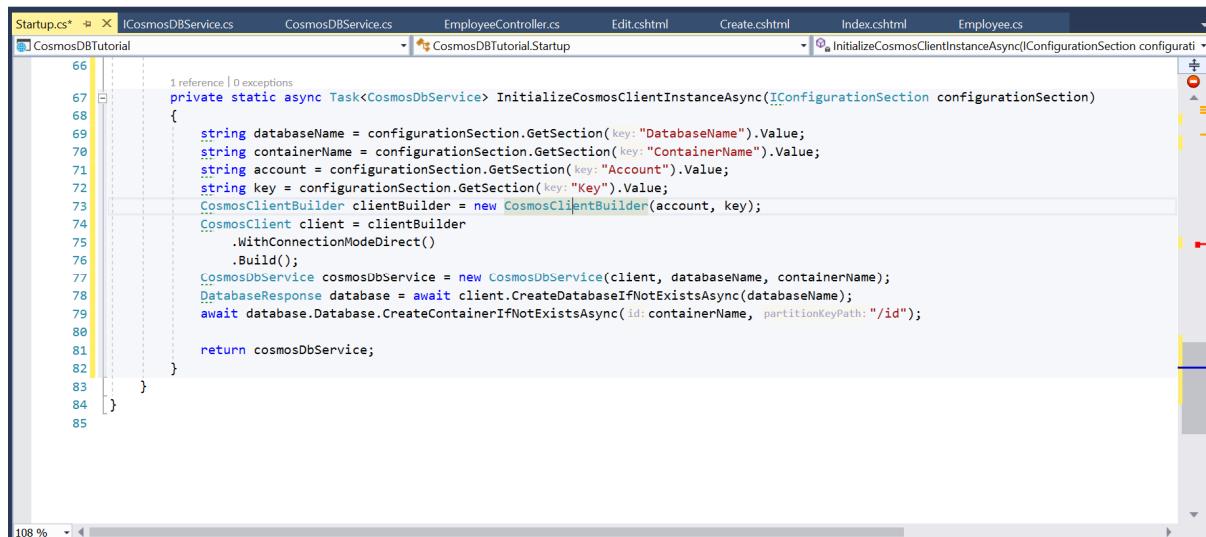
```
services.AddSingleton<ICosmosDbService>(InitializeCosmosClientInstanceAsync(Configuration.GetSection("CosmosDb")).GetAwaiter().GetResult());
```

The code in this step initializes the client based on the configuration as a singleton instance to be injected through Dependency injection in ASP.NET Core.

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Startup.cs>



Within the same file, add the following method InitializeCosmosClientInstanceAsync, which reads the configuration and initializes the client.

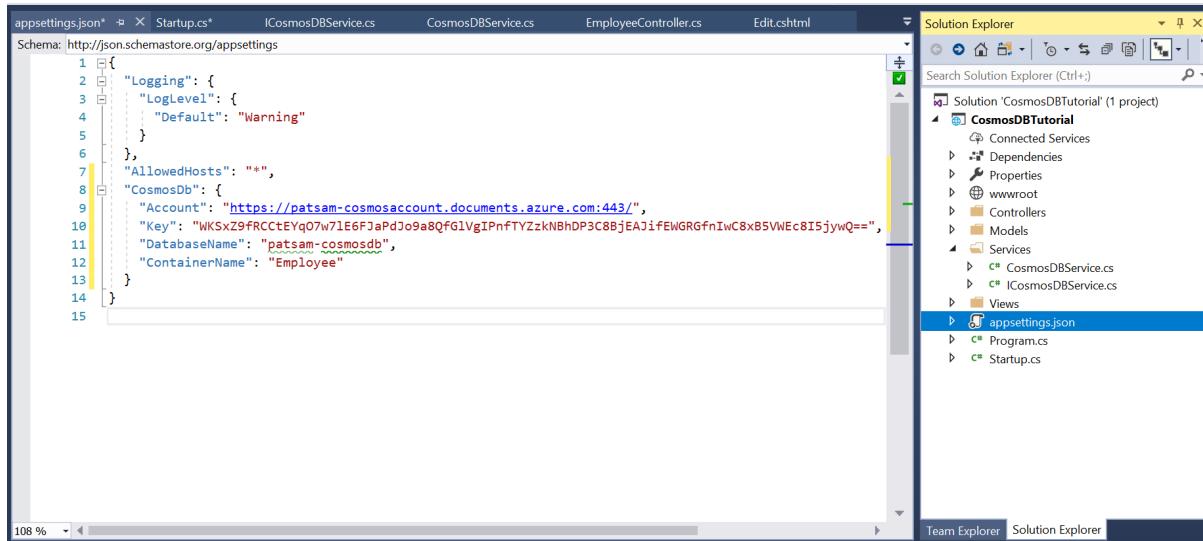


## appsettings.json

Define the configuration in the project's appsettings.json file. Open the file and add a section called CosmosDb.

- Account: <https://patsam-cosmosaccount.documents.azure.com:443/>
- Key: WKSxZ9fRCCtEYqO7w7IE6FJaPdJo9a8QfGIVgIPnfTYZzkNBhDP3C8BjEAJifEWGRGfnIwC8xB5VWEc8I5jywQ==
- Database Name: patsam-cosmosdb
- Container Name: Employee

<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/appsettings.json>



The screenshot shows the Visual Studio IDE interface. On the left, the code editor displays the appsettings.json file with the following content:

```
1  {
2    "Logging": {
3      "LogLevel": {
4        "Default": "Warning"
5      }
6    },
7    "AllowedHosts": "*",
8    "CosmosDb": {
9      "Account": "https://patsam-cosmosaccount.documents.azure.com:443/",
10     "Key": "WKSxZ9fRCCtEYqO7w7IE6FJaPdJo9a8QfGIVgIPnfTYZzkNBhDP3C8BjEAJifEWGRGfnIwC8xB5VWEc8I5jywQ==",
11     "DatabaseName": "patsam-cosmosdb",
12     "ContainerName": "Employee"
13   }
14 }
```

The Solution Explorer on the right shows the project structure for 'CosmosDBTutorial' with files like Startup.cs, ICosmosDBService.cs, CosmosDBService.cs, EmployeeController.cs, Edit.cshtml, Program.cs, and Startup.cs.

Cosmos DB Account URL and Key details can be retrieved from Keys tab.

The screenshot shows the 'Keys' tab for the 'patsam-cosmosaccount' database in the Azure portal. The 'Read-only Keys' tab is selected. The 'URI' field contains 'https://patsam-cosmosaccount.documents.azure.com:443/'. Below it are fields for 'PRIMARY KEY' (WKSxZ9fRCCtEYqO7w7IE6FJaPdJo9a8QfGlVglPnfTYZzkNBhDP3C8BjEAJifEWGRGfnlwC8xB5VWEc8l5jywQ==), 'SECONDARY KEY' (bKRLbg4hZC9HG4PezkwG3GG63EFAOEpcBqeFihfkzWw6rDMPeFgPhyhLP5En2tk1ajazXzb5vXaOBfwYy8ow==), 'PRIMARY CONNECTION STRING' (AccountEndpoint=https://patsam-cosmosaccount.documents.azure.com:443/;AccountKey=WKSxZ9fRCCtEYqO7w7IE6FJaPdJo9a8QfGlVglPnfTYZzkNBhDP3C8Bj...), and 'SECONDARY CONNECTION STRING' (AccountEndpoint=https://patsam-cosmosaccount.documents.azure.com:443/;AccountKey=bKRLbg4hZC9HG4PezkwG3GG63EFAOEpcBqeFihfkzWw6rDMPeF...). Each string has a copy icon and a more options icon.

Database Name and Container Name can be retrieved through Data Explorer tab.

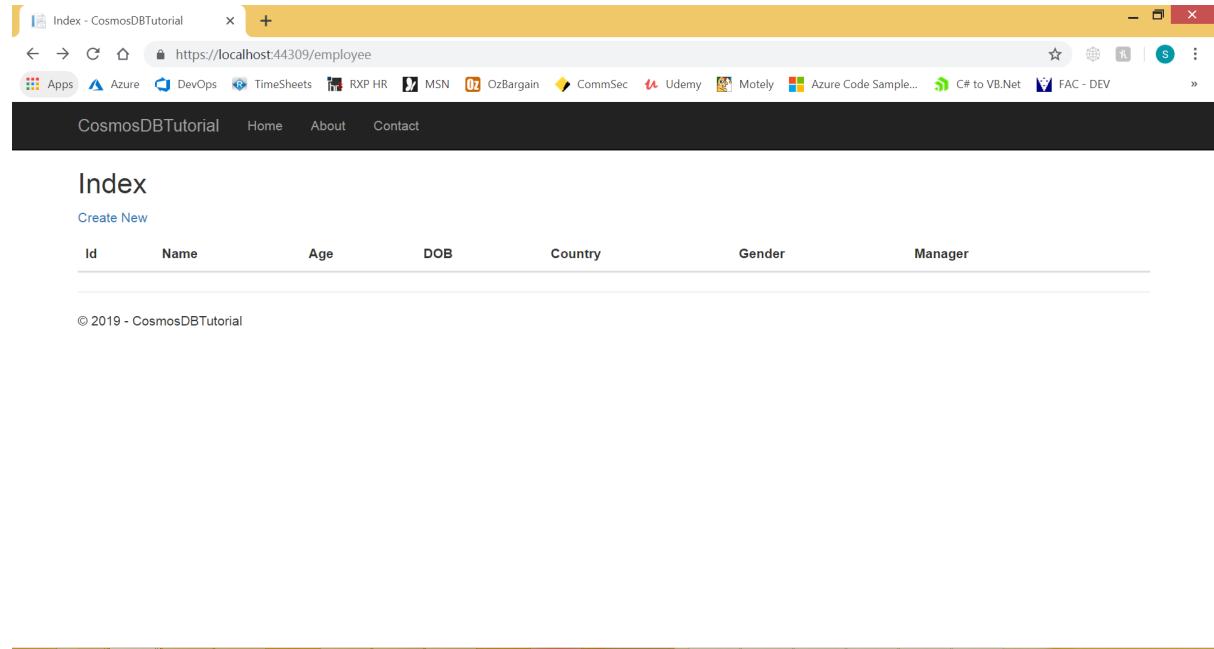
The screenshot shows the 'Data Explorer' tab for the 'patsam-cosmosaccount' database in the Azure portal. The 'Data Explorer' tab is selected. On the left, the navigation menu includes 'Data Explorer' (selected), 'SQL API', 'patsam-cosmosdb' (expanded), 'Employee' (under patsam-cosmosdb), and 'Scale'. The main pane displays a large planet icon with stars and the text 'Welcome to Azure Cosmos DB'. Below it is a message: 'Create new or work with existing container(s.)'. At the bottom, there are status indicators for document count (0), item count (0), and throughput (0).

## **Step 07: CRUD Operations**

Now we have finished putting up the code. Run the Project F5.

<https://localhost:44309/employee>

Employee list functionality:



The screenshot shows a Microsoft Edge browser window with the title "Index - CosmosDBTutorial". The address bar displays the URL "https://localhost:44309/employee". The page content is titled "Index" and includes a "Create New" link. Below it is a table header with columns: Id, Name, Age, DOB, Country, Gender, and Manager. At the bottom of the page, there is a copyright notice: "© 2019 - CosmosDBTutorial".

## Create functionality:

The screenshot shows a browser window with the URL <https://localhost:44309/employee/Create>. The page title is "Create - CosmosDBTutorial". The form fields are as follows:

<b>Id</b>	<input type="text"/>
<b>Name</b>	<input type="text" value="Patrick Sameera"/>
<b>Age</b>	<input type="text" value="30"/>
<b>DOB</b>	<input type="text" value="05/05/2000"/>
<input checked="" type="checkbox"/> Manager	
<input type="button" value="Create"/>	

[Back to List](#)

The screenshot shows a browser window with the URL <https://localhost:44309/employee>. The page title is "Index - CosmosDBTutorial". The table data is as follows:

Id	Name	Age	DOB	Manager	
211f027a-319a-4ae8-b49e-a65635ea73ce	Patrick Sameera	30	5/05/2000	<input checked="" type="checkbox"/>	<a href="#">Edit</a>   <a href="#">Details</a>   <a href="#">Delete</a>

© 2019 - CosmosDBTutorial

## Edit functionality:

The screenshot shows a web browser window titled "Edit - CosmosDBTutorial". The URL is https://localhost:44309/employee/Edit/211f027a-319a-4ae8-b49e-a65635ea73ce. The page has a header with "CosmosDBTutorial", "Home", "About", and "Contact". The main content area is titled "Edit Employee". It contains fields for "Id" (211f027a-319a-4ae8-b49e-a65635ea73ce), "Name" (Patrick Sameera Jayalath), "Age" (30), "DOB" (05/05/2000), and a checked "Manager" checkbox. A "Save" button is present at the bottom left, and a "Back to List" link is at the bottom right.

The screenshot shows a web browser window titled "Index - CosmosDBTutorial". The URL is https://localhost:44309/employee. The page has a header with "CosmosDBTutorial", "Home", "About", and "Contact". The main content area is titled "Idx" and contains a "Create New" link. Below it is a table with one row, showing the details of the employee: Id (211f027a-319a-4ae8-b49e-a65635ea73ce), Name (Patrick Sameera Jayalath), Age (30), DOB (5/05/2000), and Manager (checkbox checked). At the bottom of the table are links for "Edit | Details | Delete". The footer of the page includes the copyright notice "© 2019 - CosmosDBTutorial".

Patrick Sameera Jayalath

## Details functionality:

The screenshot shows a web browser window with the title "Details - CosmosDBTutorial". The URL in the address bar is "https://localhost:44309/employee/Details/211f027a-319a-4ae8-b49e-a65635ea73ce". The browser's toolbar includes standard icons for back, forward, search, and refresh. Below the toolbar, there is a horizontal bar with various links: Apps, Azure, DevOps, TimeSheets, RXP HR, MSN, OzBargain, CommSec, Udemy, Motely, Azure Code Sample..., C# to VB.Net, and FAC - DEV. The main content area has a dark header with the text "CosmosDBTutorial" and navigation links for "Home", "About", and "Contact". The main content is titled "Details" and "Employee". It displays the following data:

	Value
<b>Id</b>	211f027a-319a-4ae8-b49e-a65635ea73ce
<b>Name</b>	Patrick Sameera Jayalath
<b>Age</b>	30
<b>DOB</b>	5/05/2000
<b>Manager</b>	<input checked="" type="checkbox"/>

Below the data, there are links for "Edit" and "Back to List". At the bottom of the page, there is a copyright notice: "© 2019 - CosmosDBTutorial".

Patrick Sameera Jayalath

## Delete functionality:

The screenshot shows a web browser window with the URL <https://localhost:44309/employee/Delete/211f027a-319a-4ae8-b49e-a65635ea73ce>. The page title is "Delete - CosmosDBTutorial". The content area displays a delete confirmation message: "Are you sure you want to delete this? Employee". Below the message is a table showing details of the employee record: Id (211f027a-319a-4ae8-b49e-a65635ea73ce), Name (Patrick Sameera Jayalath), Age (30), DOB (5/05/2000), and Manager (checkbox checked). At the bottom left is a "Delete" button, and at the bottom right is a link to "Back to List".

We can inspect the data through Data Explorer tab.

The screenshot shows the Azure Cosmos DB Data Explorer tab for the "patsam-cosmosaccount" database. The left sidebar shows navigation options like "Diagnose and solve problems", "Quick start", "Notifications", "Data Explorer" (which is selected), "Settings", "Replicate data globally", "Default consistency", "Firewall and virtual networks", "CORS", "Keys", "Add Azure Search", and "Add Azure Function". The main area shows the "SQL API" interface. Under the "patsam-cosmosdb" database, the "Employee" collection is selected. The "Items" section displays a table with columns "id" and "/id". A single item is selected, showing its details: id (211f027a-319a-4ae8-b49e-a65635ea73ce) and /id (211f027a-319a-4...). The JSON representation of the item is shown on the right:

```
1 "id": "211f027a-319a-4ae8-b49e-a65635ea73ce",
2 "name": "Patrick Sameera Jayalath",
3 "age": 30,
4 "dob": "2000-05-05T00:00:00",
5 "ismanager": true,
6 "_rid": "ffc-Aj09mdIGAAAAAAA=",
7 "_self": "dbs/ffc-AA=/colls/ffc-Aj09mdI=/docs/ffc-
8 "_etag": "\"4c00ed17-0000-0300-0000-5da00fe00000\"",
9 "_attachments": "attachments/",
10 "_ts": 1570770912
```