

## **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 header with columns: NAME ↑, STATUS ↑, LOCATION ↑, and SUBSCRIPTION ↑. In the center of the page is a large, stylized planet icon with a ring. Below the icon, the text 'No Azure Cosmos DB accounts to display' is centered. At the bottom of the page, there is a promotional message: 'Create a globally distributed, multi-model, fully managed database using API of your choice. Or try it for free, up to 20k RU/s, for 30 days with unlimited renewal. Learn more' with a link icon. A prominent blue button at the bottom right says '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 purple 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. The page contains several input fields:

- Project Details:** 'Subscription' dropdown set to 'Visual Studio Enterprise - MPN', 'Resource Group' dropdown set to 'patsam-rg'.
- Instance Details:**
  - Account Name:** 'patsam-cosmosaccount'
  - API:** 'Core (SQL)' dropdown.
  - Location:** 'Central US' dropdown.
  - Geo-Redundancy:** 'Enable' button.
  - Multi-region Writes:** 'Enable' button.
- A note at the bottom states: 'Up to 33% off multi-region writes is available to qualifying new accounts only. Accounts must be created between August 16, 2019 and November 30, 2019. Offer limited to accounts with both account locations and geo-redundancy only in North Europe, West Europe, South Central US, or North Central US regions, and applies only to multi-region writes in those same regions. Both Geo-Redundancy and Multi-region Writes must be enabled in account settings. Actual discount will vary based on number of qualifying regions selected.'

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

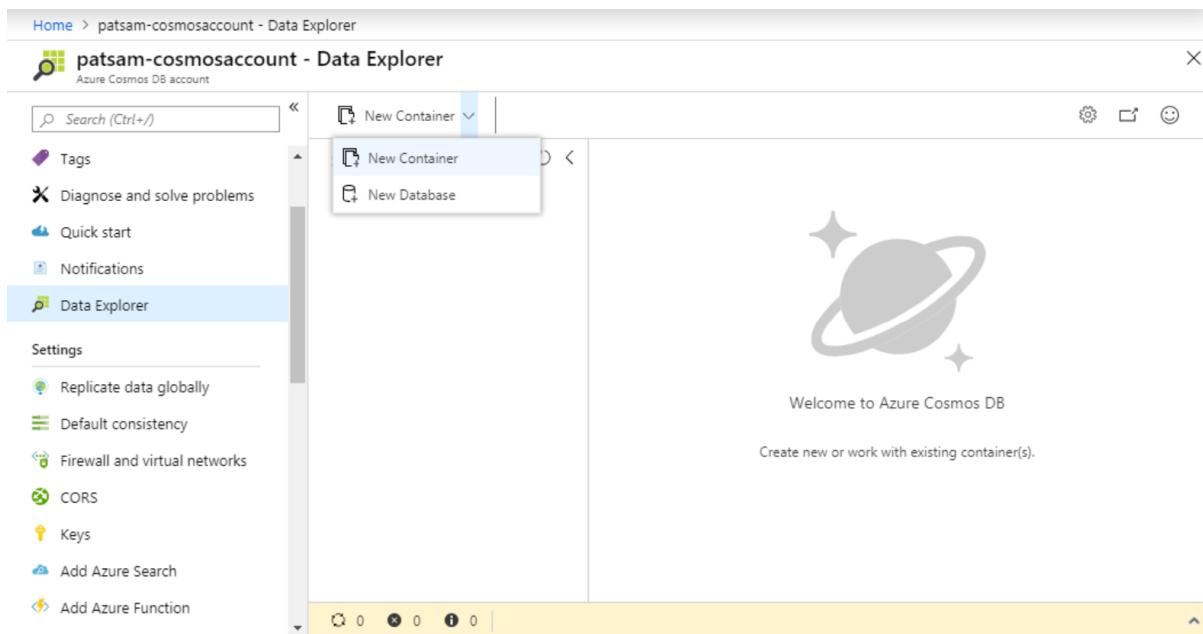
Once the deployment finishes, we can see the newly created Cosmos DB Account.

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. The left sidebar has 'Data Explorer' selected. A dropdown menu is open under the 'New Container' button, showing 'New Container' and 'New Database'. The main area features a large planet icon with stars and the text 'Welcome to Azure Cosmos DB' and 'Create new or work with existing container(s.)'. The bottom status bar shows metrics: 0 documents, 0 partitions, and 0 triggers.

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 —  
+

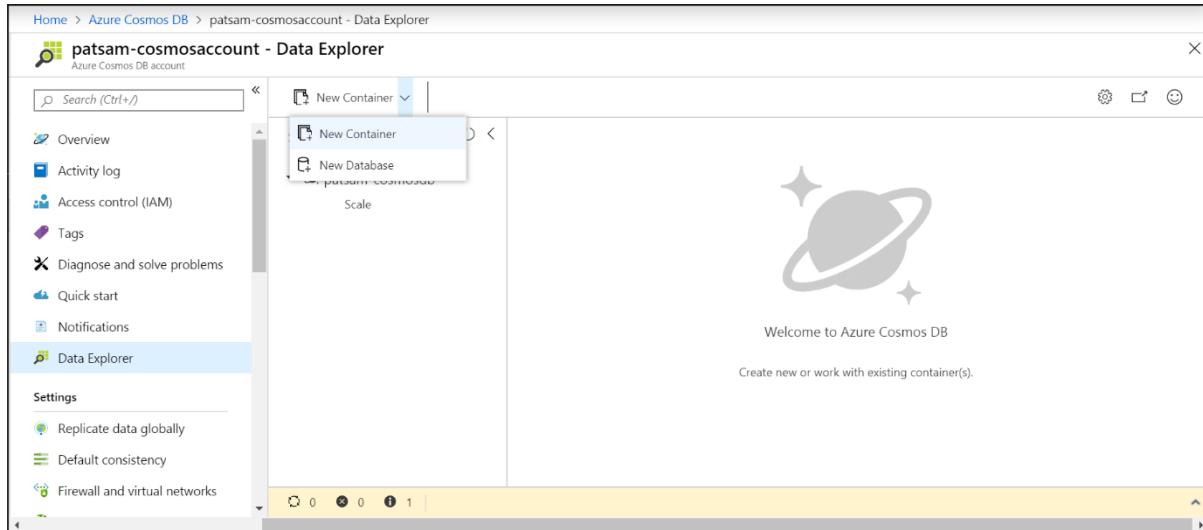
Estimated spend (USD): **\$0.032 hourly / \$0.77 daily** (1 region, 100RU/s, 100000RU/s)

**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 ⓘ

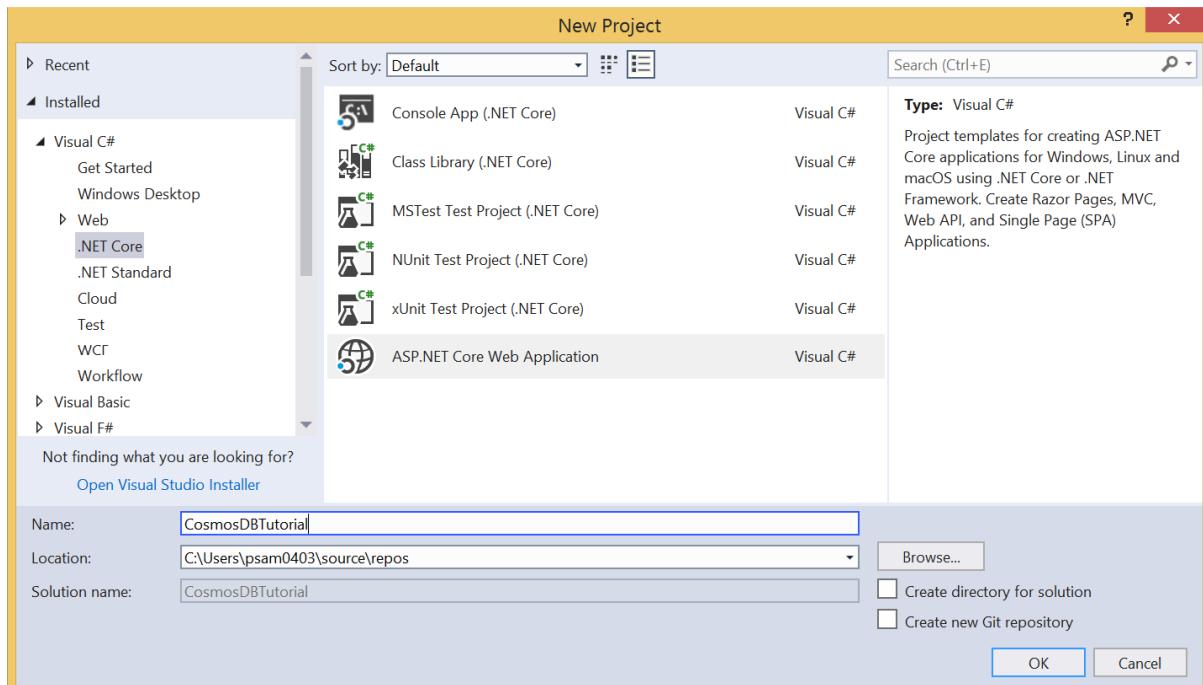
Unique keys ⓘ

+ Add unique key

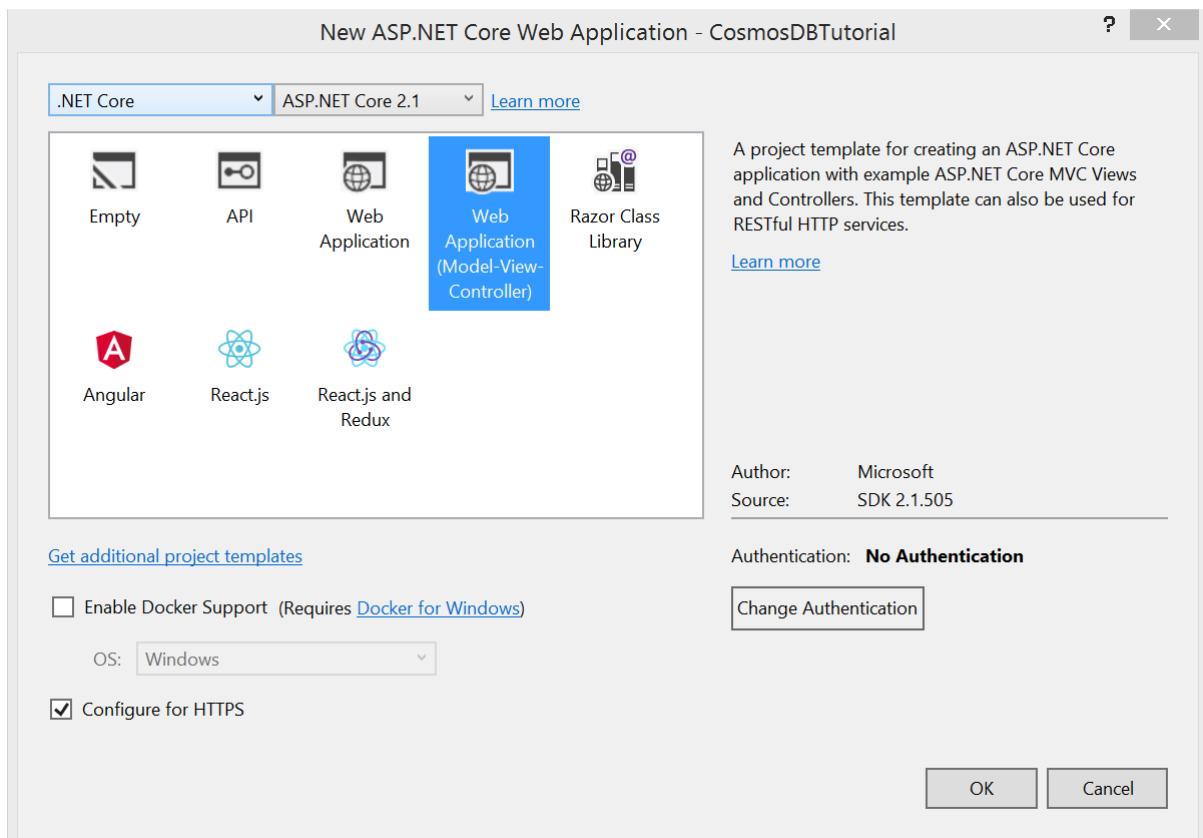
OK

## **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.

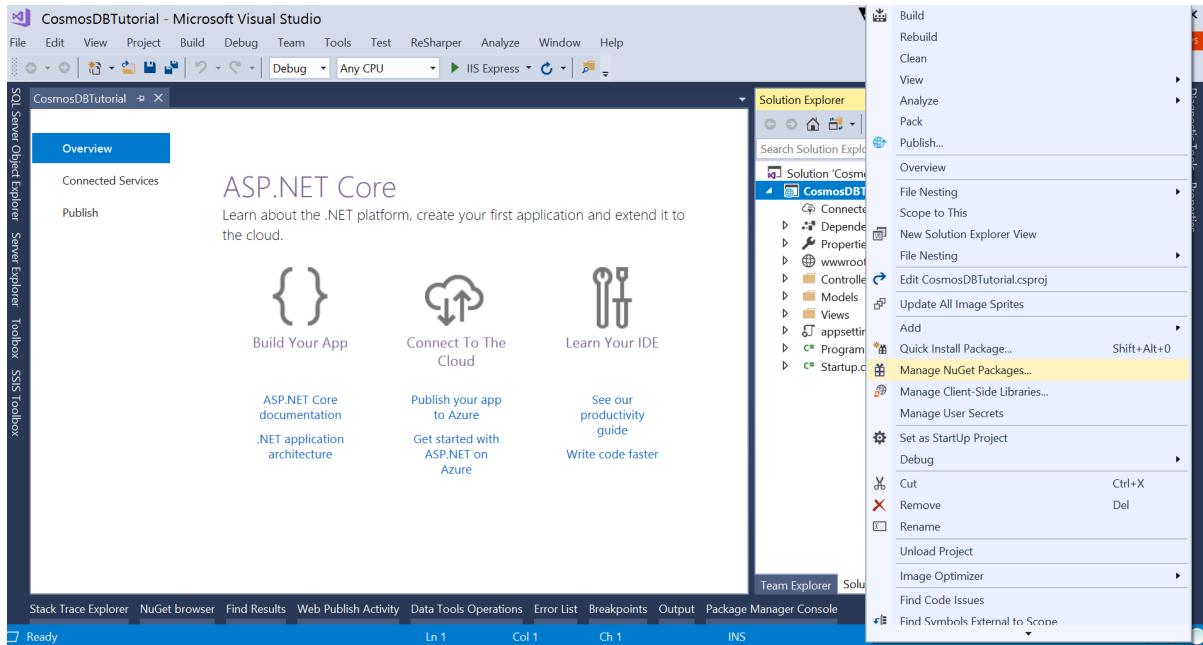
The screenshot shows the Microsoft Visual Studio interface for a project named "CosmosDBTutorial". The left pane displays the "Overview" of the ASP.NET Core project, featuring sections for "Build Your App", "Connect To The Cloud", and "Learn Your IDE", along with links to documentation and productivity guides. The right pane shows the "Solution Explorer" with the project structure, including files like Program.cs and Startup.cs. Below the main window, the taskbar shows various open tools and the current file path: "Home Page - CosmosDBTutorial".

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. Below the header, text promotes new features for building modern web apps, with a "Learn More" button. The browser's address bar and toolbars are visible at the top.

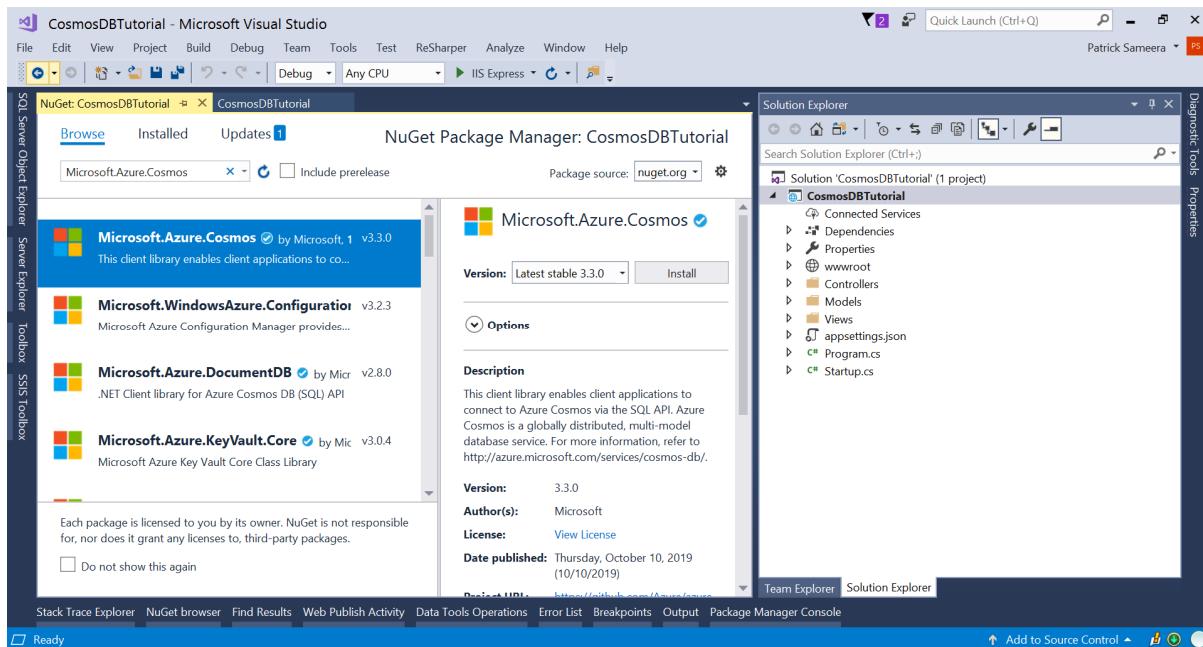
Application uses	How to	Overview	Run & Deploy
<ul style="list-style-type: none"><li>Sample pages using ASP.NET Core MVC</li><li>Theming using Bootstrap</li></ul>	<ul style="list-style-type: none"><li>Add a Controller and View</li><li>Manage User Secrets using Secret Manager.</li><li>Use logging to log 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

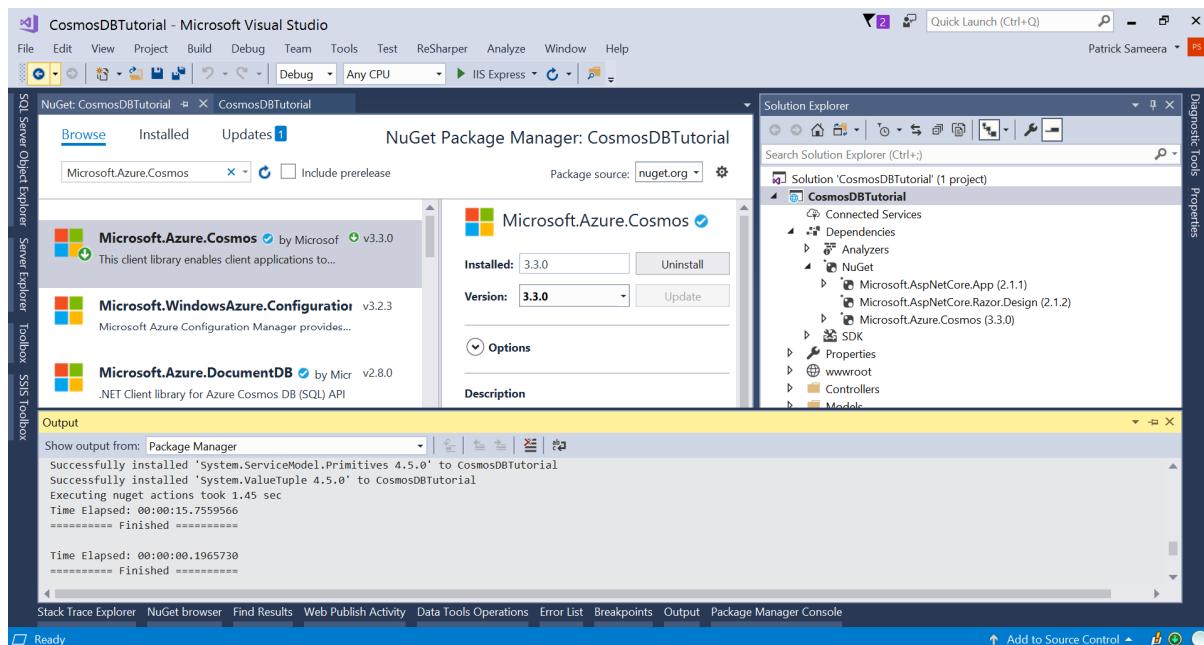
In the Solution Explorer, right click on the Project and select Manage NuGet Packages.



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:

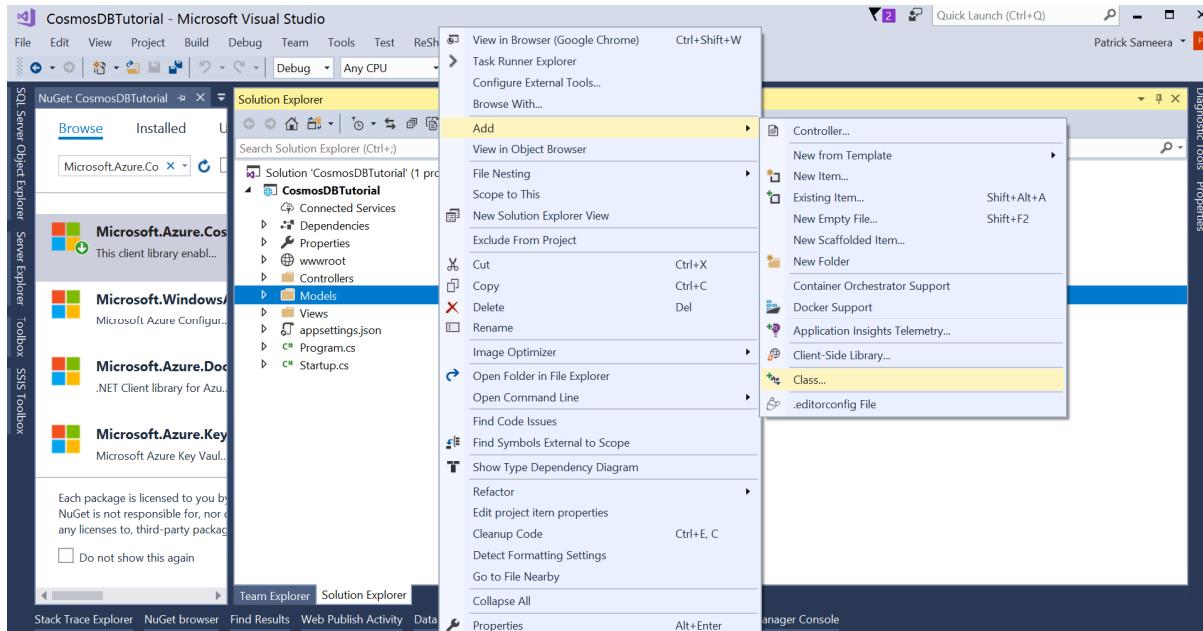
Id	Name	Age	DOB	Manager
211f027a-319a-4ae8-b49e-a65635ea73ce	Patrick Sameera	30	5/05/2000	<input checked="" type="checkbox"/>

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

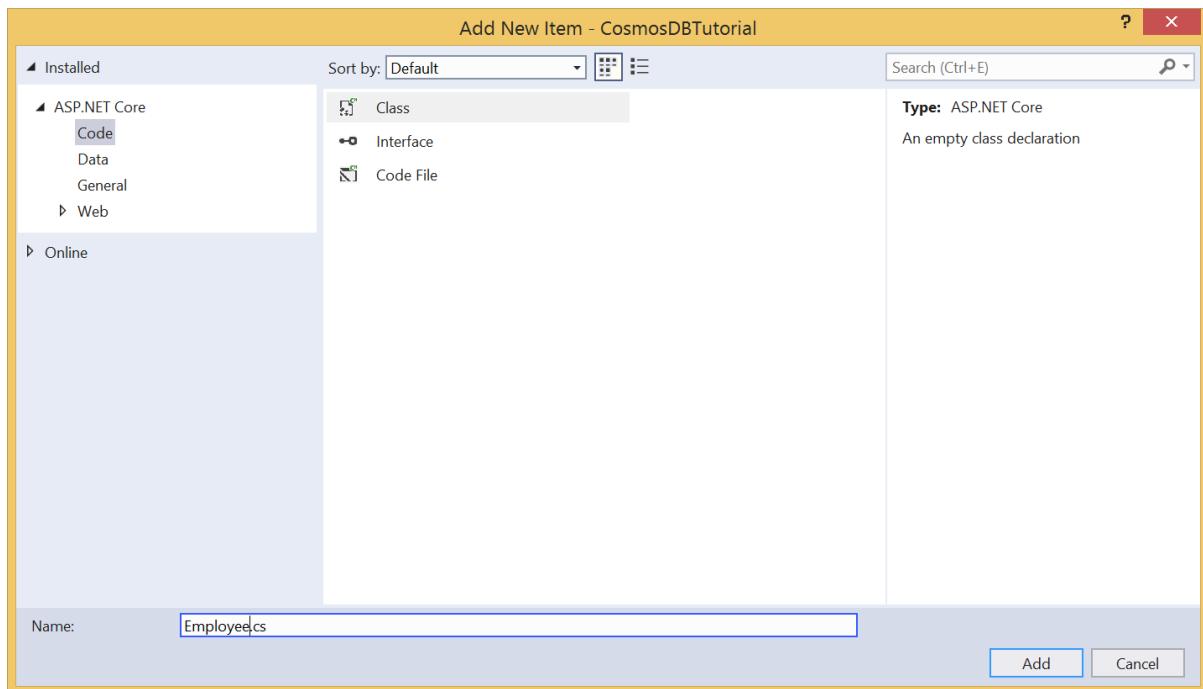
## Adding Model

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 "Employee.cs", "ErrorViewModel.cs", and "Program.cs". The "Views" folder contains "Employee", "Home", "Shared", "\_ViewImports.cshtml", and "\_ViewStart.cshtml". Other files like "Startup.cs" and "appsettings.json" are also listed.

```
Employee.cs
1 using Newtonsoft.Json;
2 using System;
3 using System.ComponentModel.DataAnnotations;
4
5 namespace CosmosDBTutorial.Models
6 {
7     [20 references]
8     public class Employee
9     {
10         [Key]
11         [JsonProperty(PropertyName = "id")]
12         [3 references | 0 exceptions]
13         public string Id { get; set; }
14
15         [StringLength(60, MinimumLength = 3)]
16         [Required]
17         [JsonProperty(PropertyName = "name")]
18         [0 references | 0 exceptions]
19         public string Name { get; set; }
20
21         [JsonProperty(PropertyName = "age")]
22         [0 references | 0 exceptions]
23         public int Age { get; set; }
24
25         [DataType(DataType.Date)]
26         [JsonProperty(PropertyName = "dob")]
27         [0 references | 0 exceptions]
28         public DateTime DOB { get; set; }
29
30         [JsonProperty(PropertyName = "ismanager")]
31         [0 references | 0 exceptions]
32         public bool Manager { get; set; }
33     }
34 }
```

*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.*

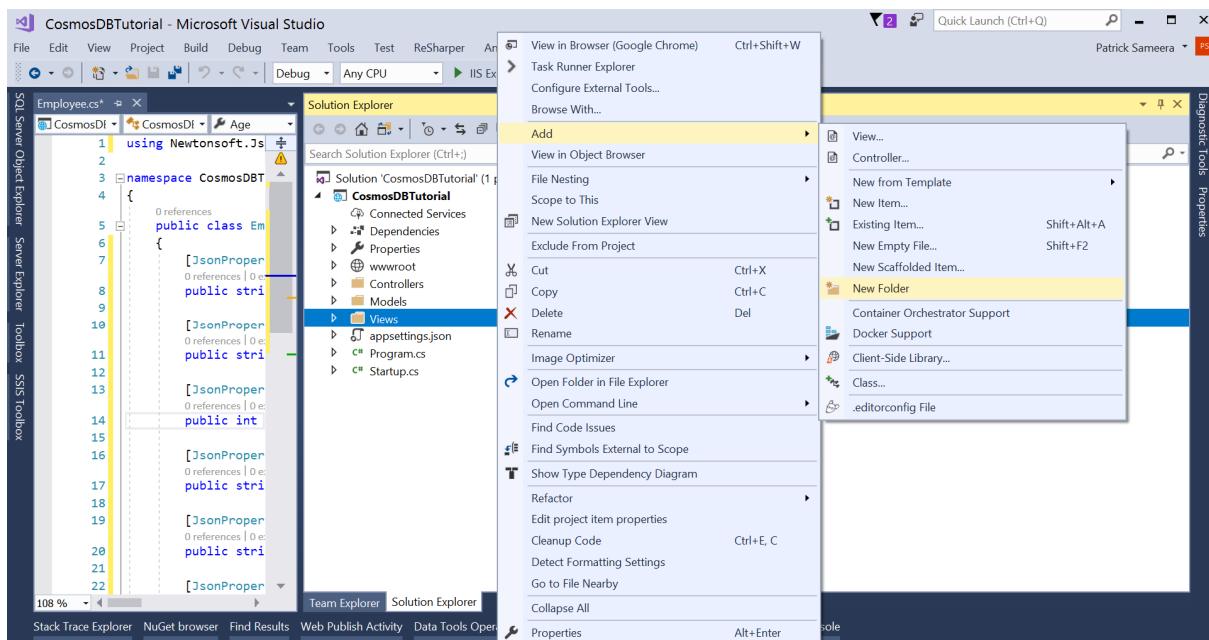
## Adding Views

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

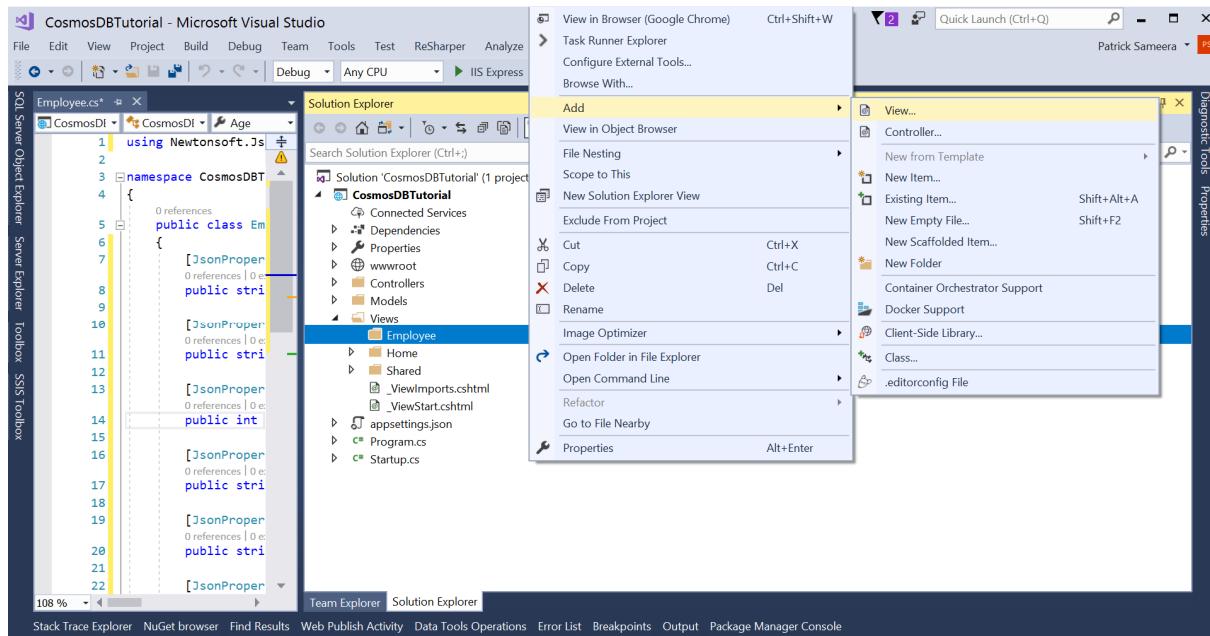
First create a new Folder under Views by the name Employee. Right Click on Views.

- Views → Add → New Folder → Employee



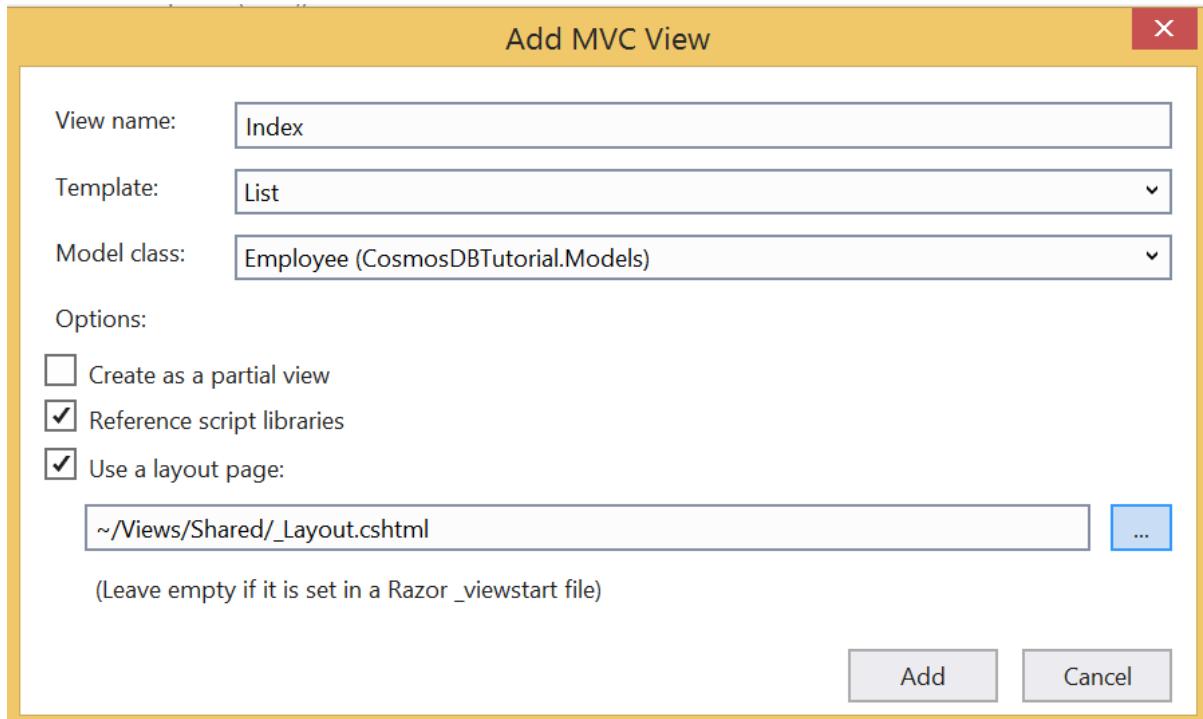
Next, we need to add the 5 Views. Right Click on Employee Folder under Views to add Views.

- Views → Employees → Add → View

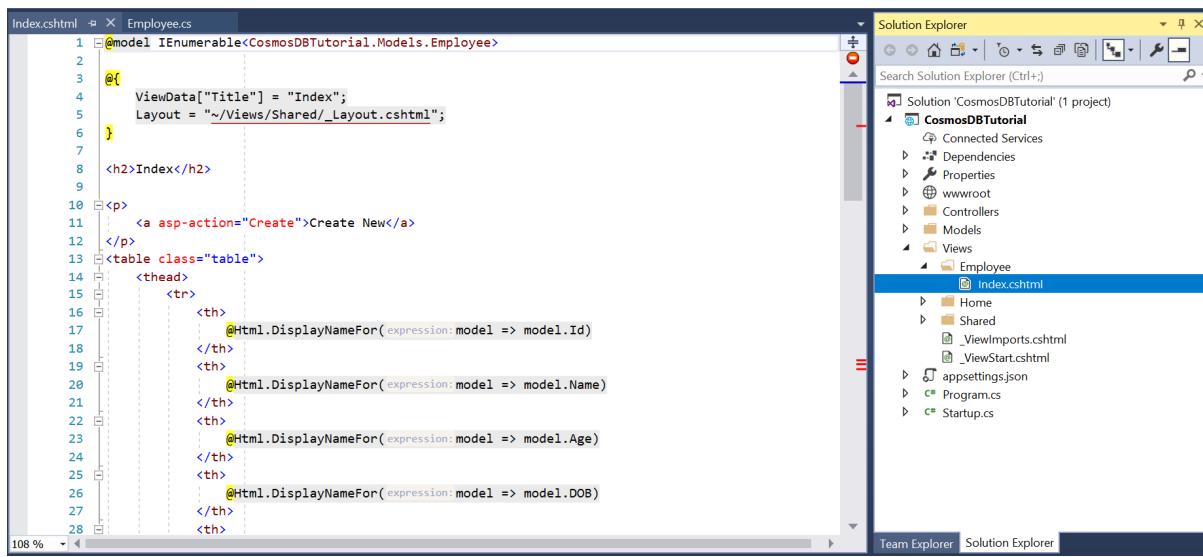


## View 01: Index

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

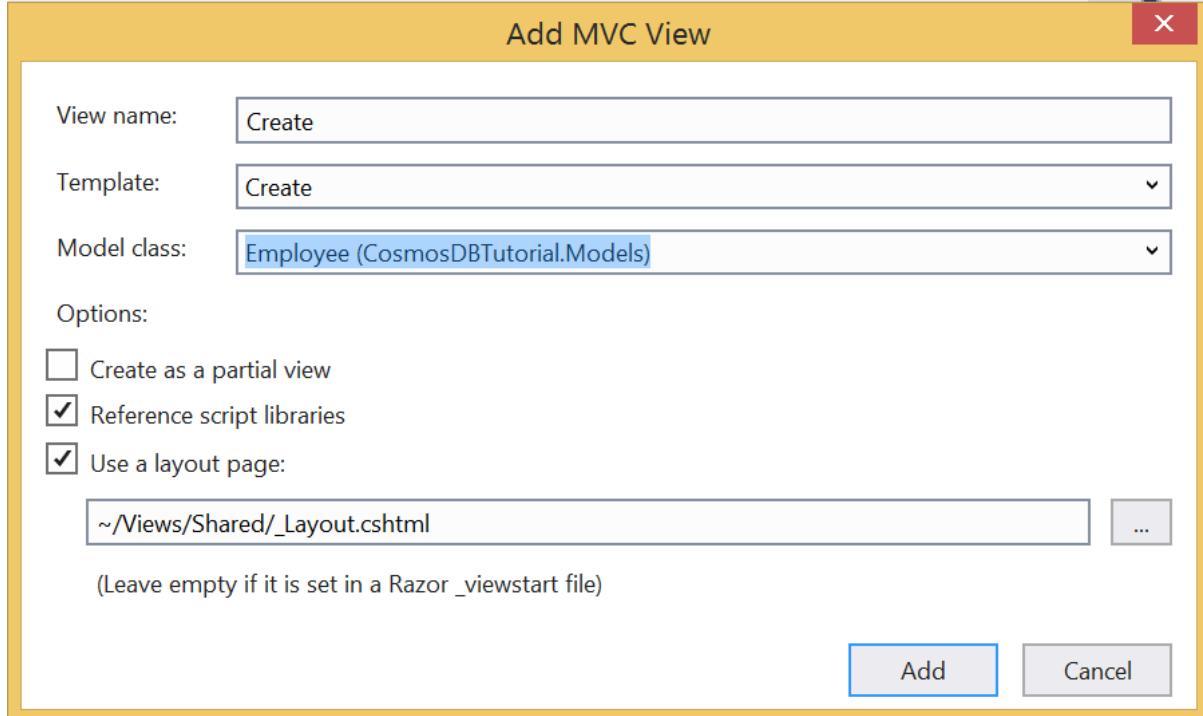


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Index.cshtml>

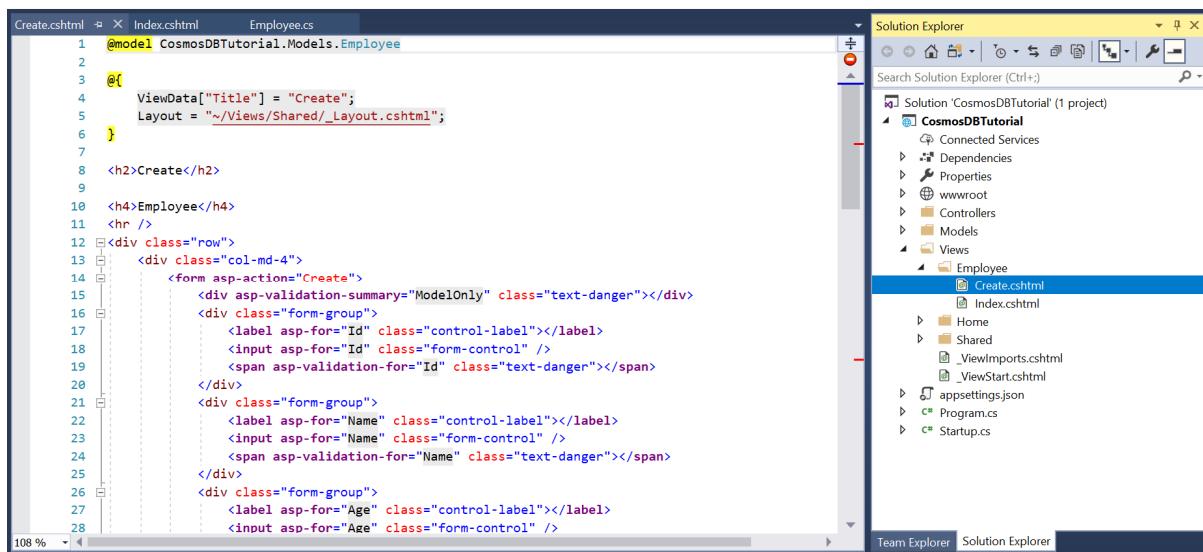


## View 02: Create

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

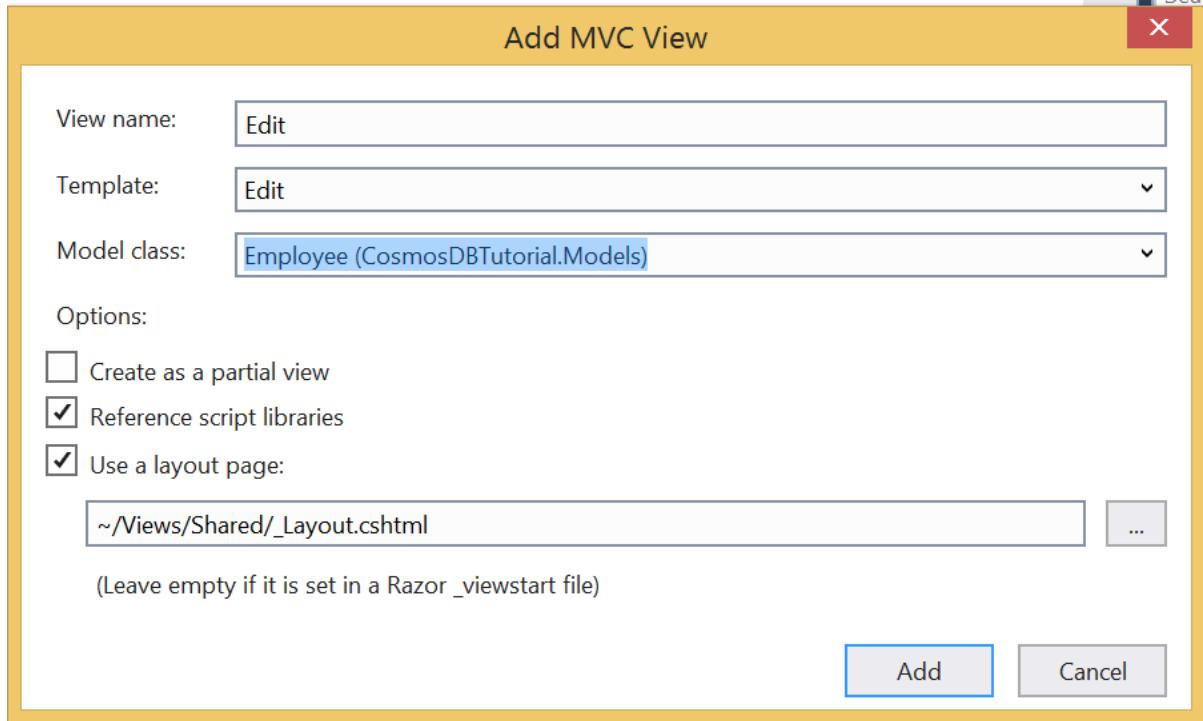


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Create.cshtml>

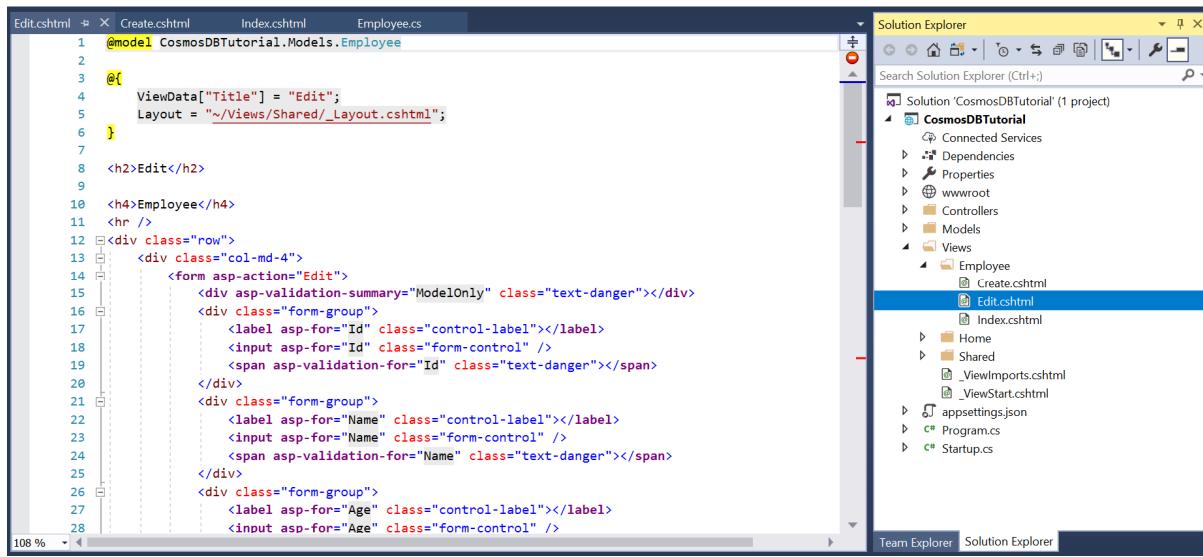


### View 03: Edit

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

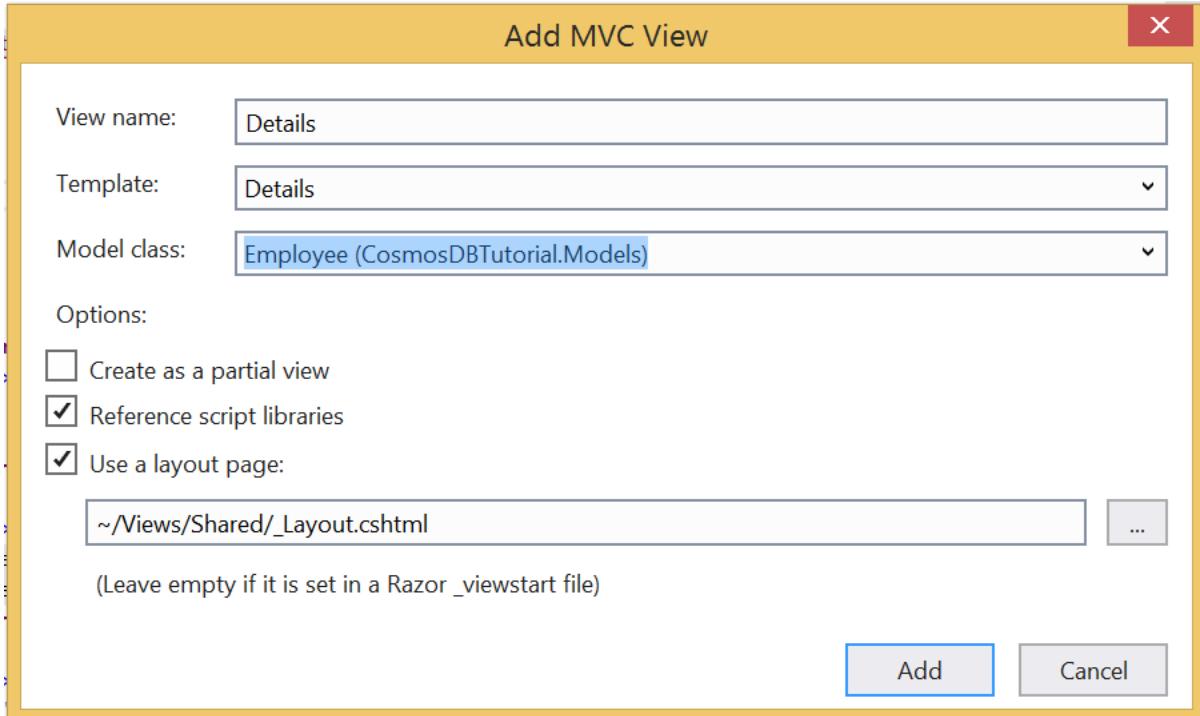


<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Edit.cshtml>



## View 04: Details

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



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

The screenshot shows the Visual Studio IDE. The code editor displays the 'Details.cshtml' file with the following content:

```

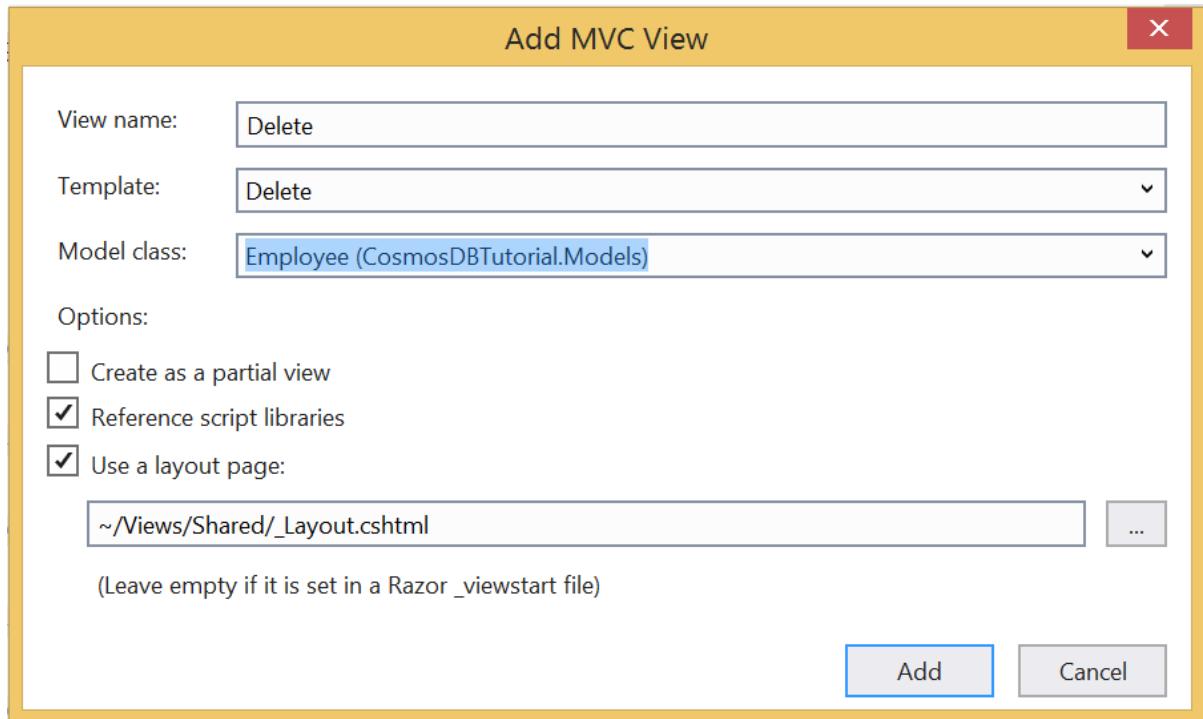
1 <model> CosmosDBTutorial.Models.Employee
2 
3 <
4   ViewData["Title"] = "Details";
5   Layout = "~/Views/Shared/_Layout.cshtml";
6 >
7 
8 <h2>Details</h2>
9 
10 <div>
11   <h4>Employee</h4>
12   <hr />
13   <dl class="dl-horizontal">
14     <dt>
15       @Html.DisplayNameFor(expression: model => model.Id)
16     </dt>
17     <dd>
18       @Html.DisplayFor(expression: model => model.Id)
19     </dd>
20     <dt>
21       @Html.DisplayNameFor(expression: model => model.Name)
22     </dt>
23     <dd>
24       @Html.DisplayFor(expression: model => model.Name)
25     </dd>
26     <dt>
27       @Html.DisplayNameFor(expression: model => model.Age)
28     </dt>
29     <dd>
30       @Html.DisplayFor(expression: model => model.Age)
31     </dd>
32     <dt>
33       @Html.DisplayNameFor(expression: model => model.DOB)
34     </dt>

```

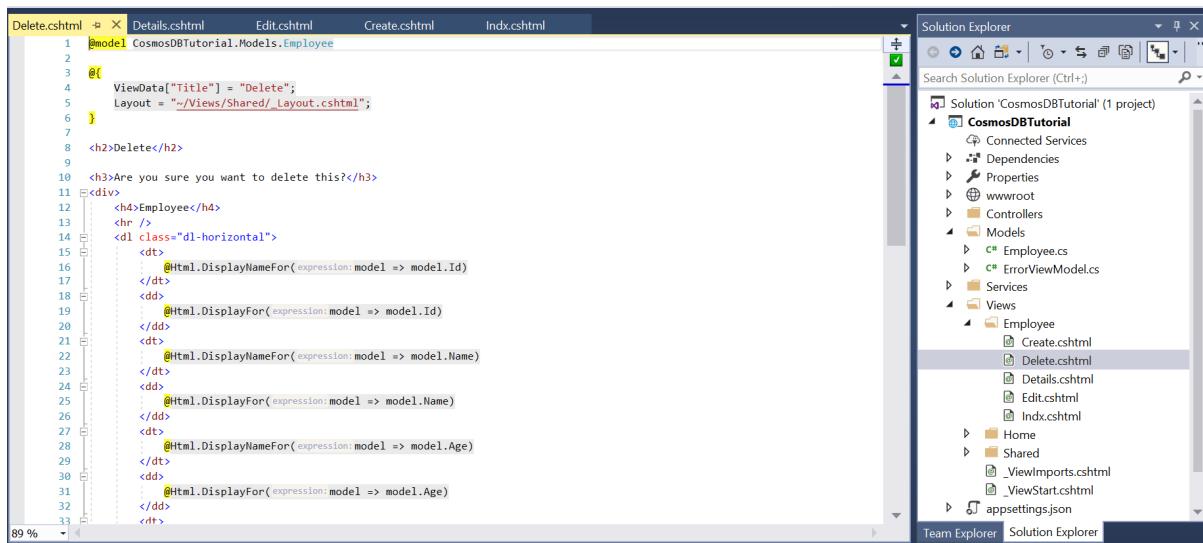
The Solution Explorer on the right shows the project structure with the 'Details.cshtml' file selected under the 'Views\Employee' folder.

## View 04: Delete

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



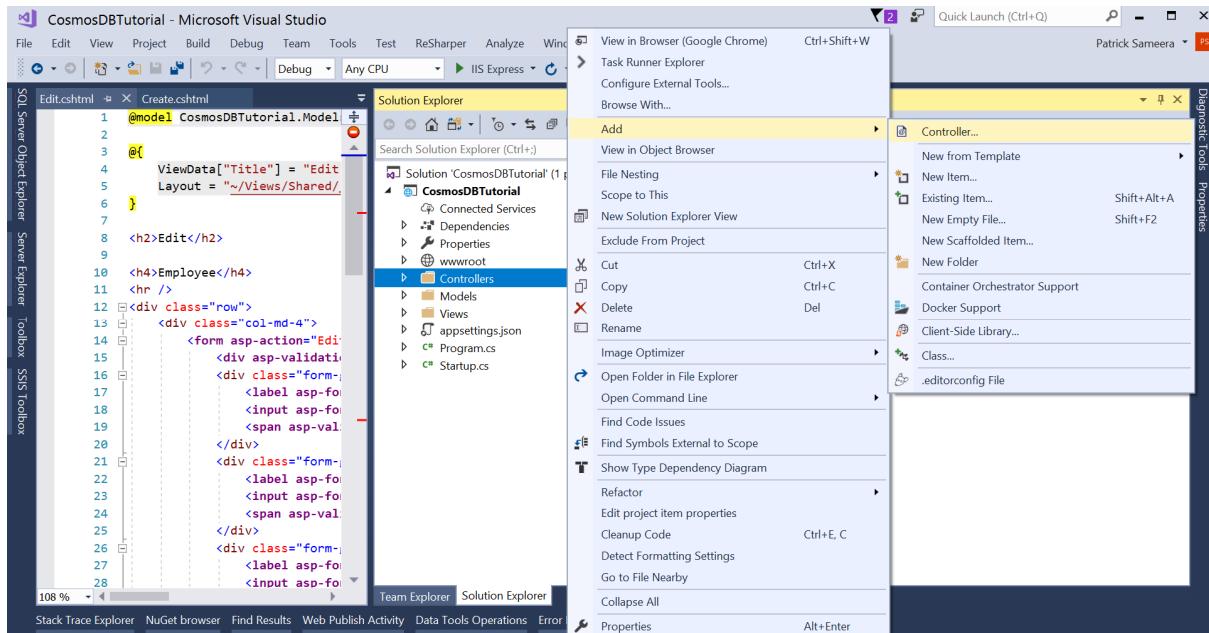
<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Delete.cshtml>



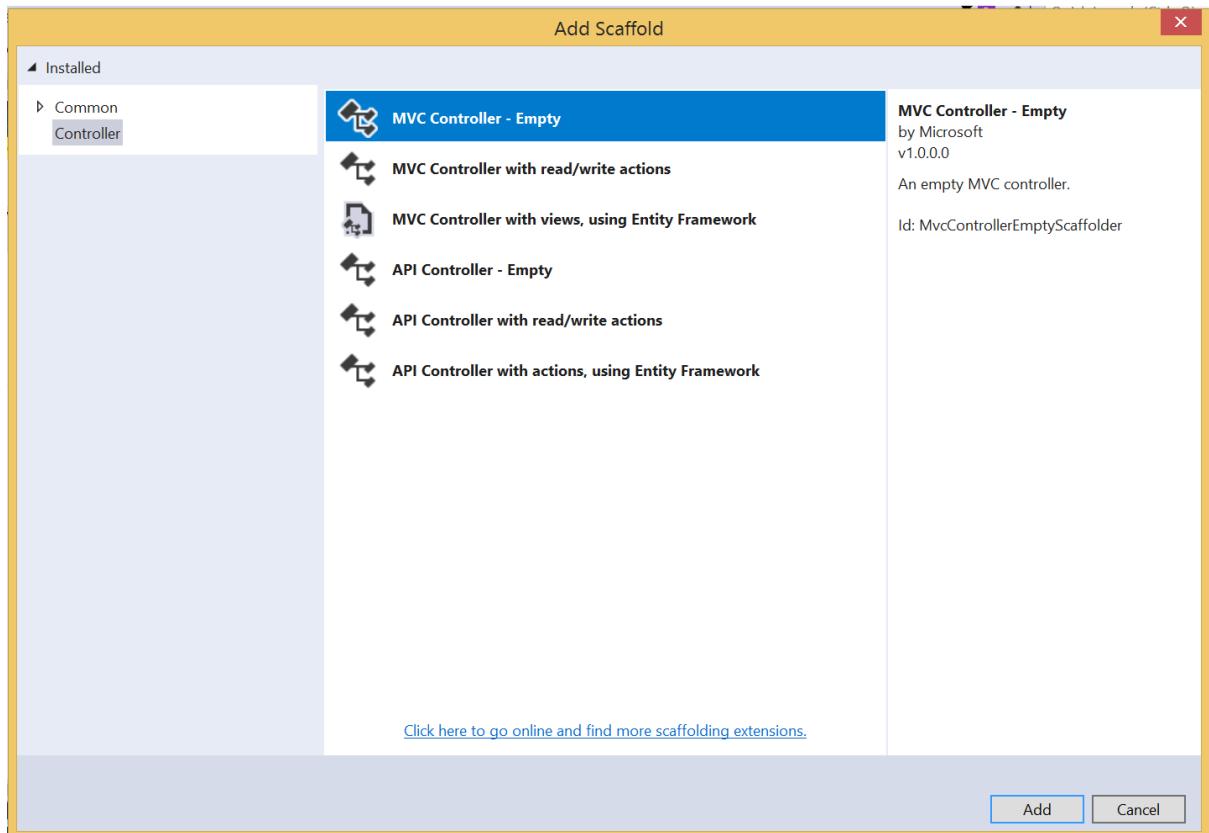
## Adding Controllers

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

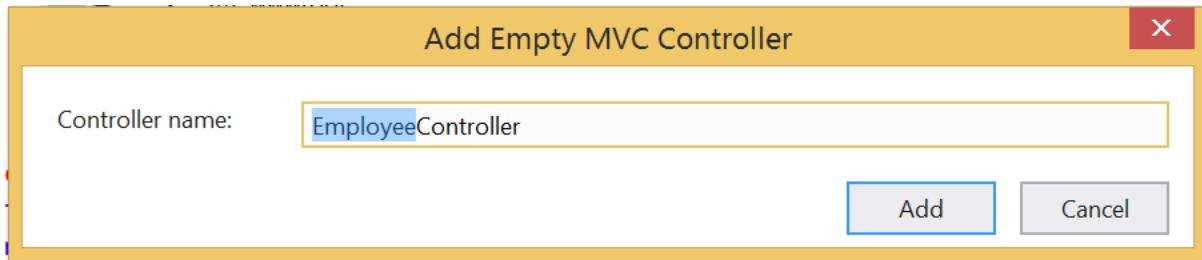
Controllers → Add → Controller



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>

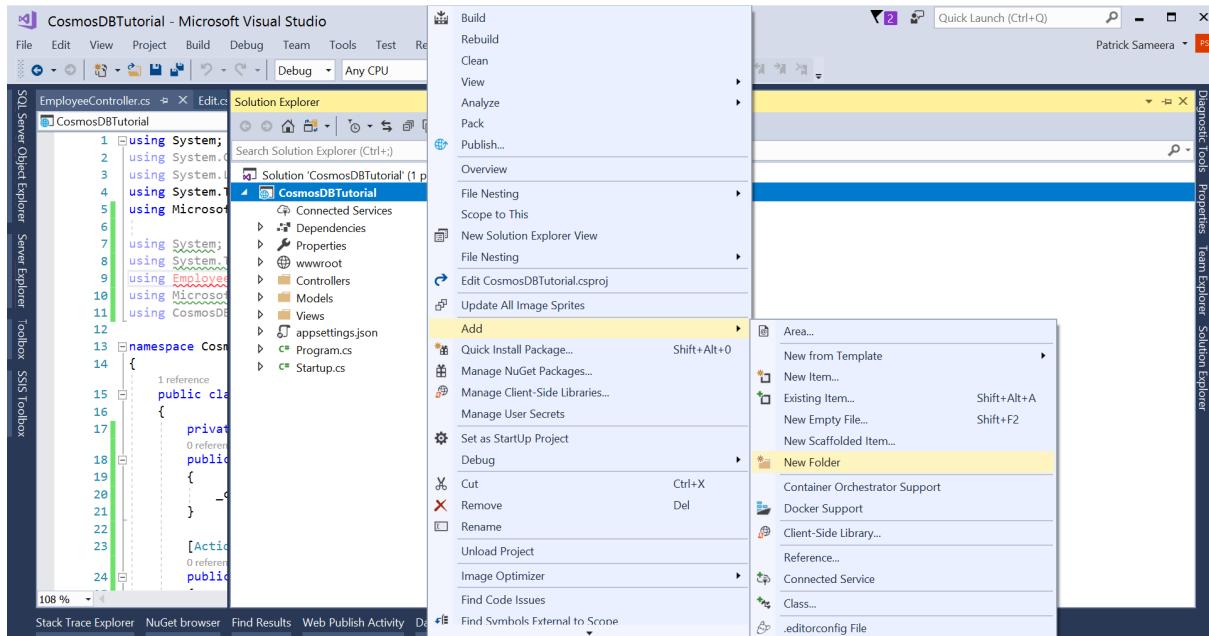
A screenshot of the Visual Studio IDE. On the left, the code editor shows the "EmployeeController.cs" file with C# code. On the right, the "Solution Explorer" pane shows the project structure for "CosmosDBTutorial". The "Controllers" folder contains "EmployeeController.cs" and "HomeController.cs". Other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

## 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.

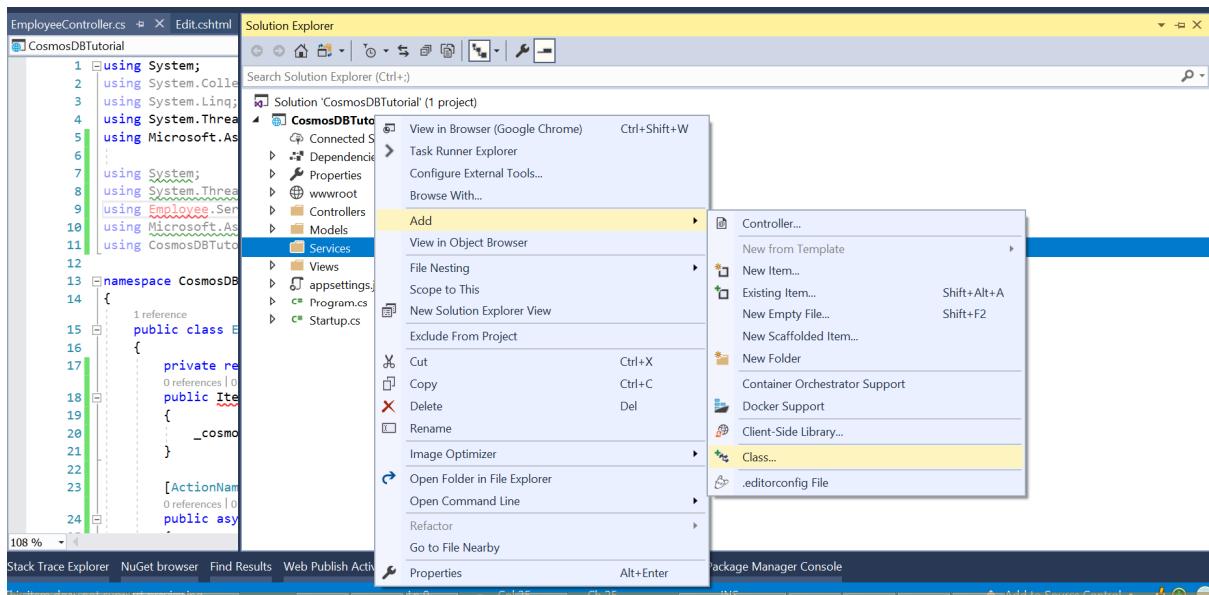
First Add a new Folder called Services. Right Click on the Solution.

- Add → New Folder



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

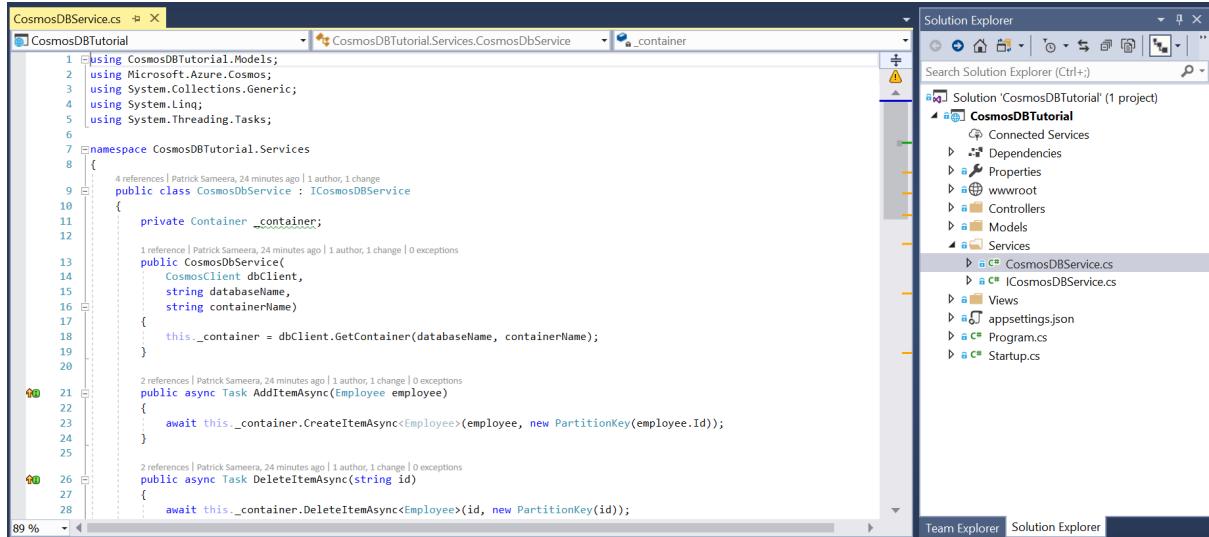
- Services → Add → Class



## 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 class has two methods: `AddItemAsync` and `DeleteItemAsync`, both of which use the `CreateItemAsync` and `DeleteItemAsync` methods of the `CosmosClient` to interact with a database and container. 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      public class CosmosDbService : ICosmosDBService
10     {
11         private Container _container;
12
13         public CosmosDbService(
14             CosmosClient dbClient,
15             string databaseName,
16             string containerName)
17         {
18             this._container = dbClient.GetContainer(databaseName, containerName);
19         }
20
21         public async Task AddItemAsync(Employee employee)
22         {
23             await this._container.CreateItemAsync<Employee>(employee, new PartitionKey(employee.Id));
24         }
25
26         public async Task DeleteItemAsync(string id)
27         {
28             await this._container.DeleteItemAsync<Employee>(id, new PartitionKey(id));
29         }
30     }
31 }
```

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 containing 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 displays the project structure for `CosmosDBTutorial`. The `Services` folder contains both `CosmosDBService.cs` and `ICosmosDBService.cs`, indicating they are part of the same service layer.

```
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
```

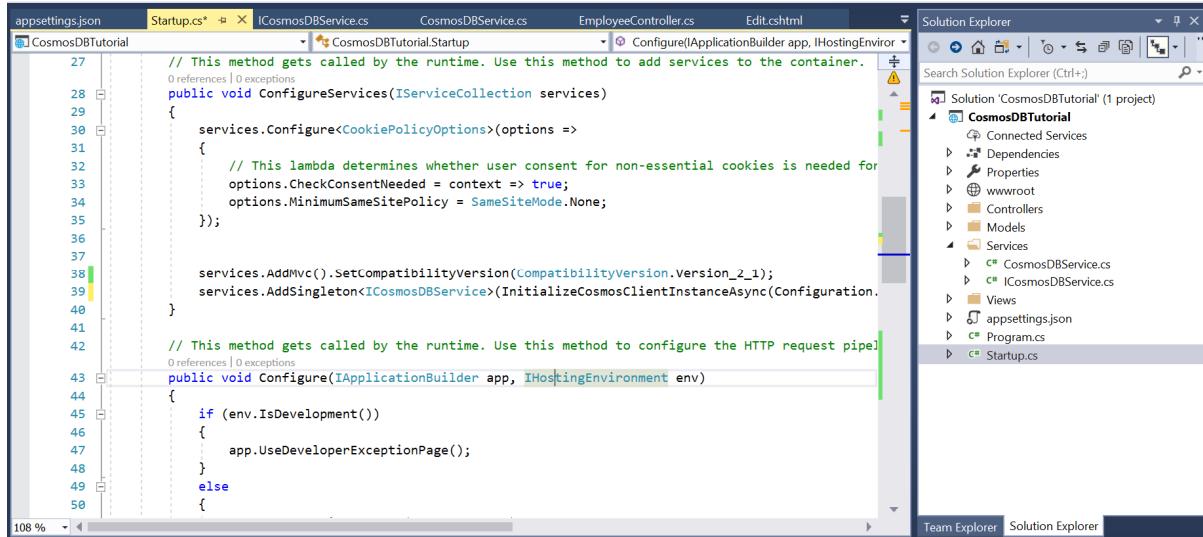
## Startup.cs

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.

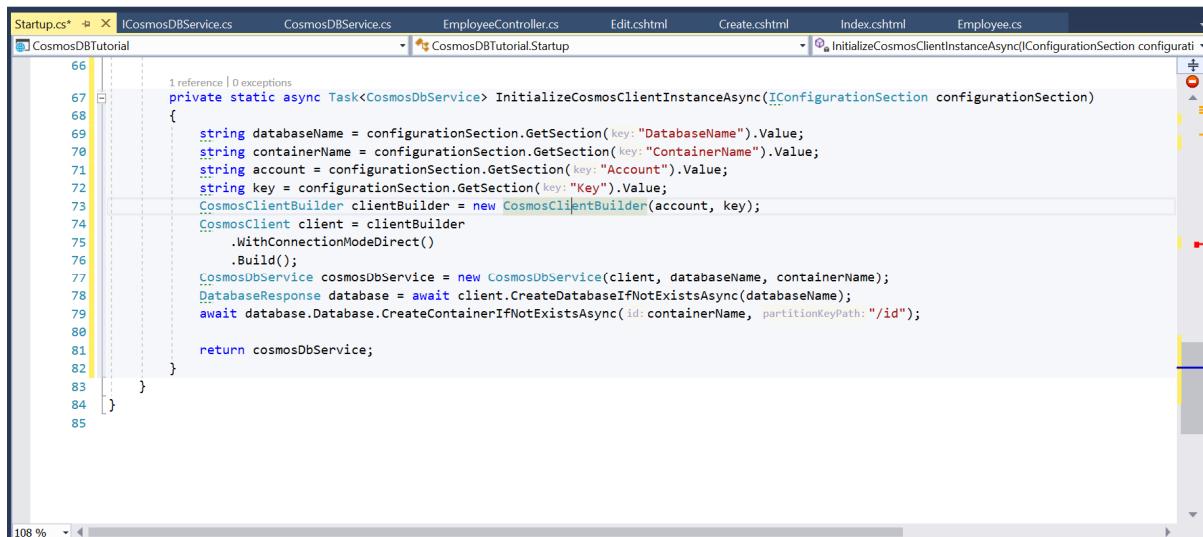
<https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Startup.cs>



A screenshot of the Visual Studio IDE. The left side shows the code editor with the file 'Startup.cs' open. The code contains the ConfigureServices and Configure methods with the added line of code. The right side shows the Solution Explorer with the project 'CosmosDBTutorial' selected, displaying its files and structure.

```
27 // This method gets called by the runtime. Use this method to add services to the container.
28 public void ConfigureServices(IServiceCollection services)
29 {
30     services.Configure<CookiePolicyOptions>(options =>
31     {
32         // This lambda determines whether user consent for non-essential cookies is needed for
33         options.CheckConsentNeeded = context => true;
34         options.MinimumSameSitePolicy = SameSiteMode.None;
35     });
36
37     services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_1);
38     services.AddSingleton<ICosmosDbService>(InitializeCosmosClientInstanceAsync(Configuration.
39
40
41     // This method gets called by the runtime. Use this method to configure the HTTP request pipe
42     public void Configure(IApplicationBuilder app, IHostingEnvironment env)
43     {
44         if (env.IsDevelopment())
45         {
46             app.UseDeveloperExceptionPage();
47         }
48         else
49         {
50
51
52
53
54
55
56
57
58
59
5
60
61
62
63
64
65
66
67     private static async Task<CosmosDbService> InitializeCosmosClientInstanceAsync(IConfigurationSection configurationSection)
68     {
69         string databaseName = configurationSection.GetSection(key: "DatabaseName").Value;
70         string containerName = configurationSection.GetSection(key: "ContainerName").Value;
71         string account = configurationSection.GetSection(key: "Account").Value;
72         string key = configurationSection.GetSection(key: "Key").Value;
73         CosmosClientBuilder clientBuilder = new CosmosClientBuilder(account, key);
74         CosmosClient client = clientBuilder
75             .WithConnectionModeDirect()
76             .Build();
77         CosmosDbService cosmosDbService = new CosmosDbService(client, databaseName, containerName);
78         DatabaseResponse database = await client.CreateDatabaseIfNotExistsAsync(databaseName);
79         await database.Database.CreateContainerIfNotExistsAsync(id: containerName, partitionKeyPath: "/id");
80
81         return cosmosDbService;
82     }
83
84
85 }
```

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



A screenshot of the Visual Studio IDE showing the 'InitializeCosmosClientInstanceAsync' method in the 'Startup.cs' file. The code uses the ConfigurationSection to read values for database name, container name, account, and key, then creates a CosmosClientBuilder and a CosmosDbService.

```
66
67     private static async Task<CosmosDbService> InitializeCosmosClientInstanceAsync(IConfigurationSection configurationSection)
68     {
69         string databaseName = configurationSection.GetSection(key: "DatabaseName").Value;
70         string containerName = configurationSection.GetSection(key: "ContainerName").Value;
71         string account = configurationSection.GetSection(key: "Account").Value;
72         string key = configurationSection.GetSection(key: "Key").Value;
73         CosmosClientBuilder clientBuilder = new CosmosClientBuilder(account, key);
74         CosmosClient client = clientBuilder
75             .WithConnectionModeDirect()
76             .Build();
77         CosmosDbService cosmosDbService = new CosmosDbService(client, databaseName, containerName);
78         DatabaseResponse database = await client.CreateDatabaseIfNotExistsAsync(databaseName);
79         await database.Database.CreateContainerIfNotExistsAsync(id: containerName, partitionKeyPath: "/id");
80
81         return cosmosDbService;
82     }
83
84
85 }
```

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: WKSxZ9fRCCtEYqO7w7lE6FJaPdJo9a8QfGIVgIPnfTYZzkNBhDP3C8BjEAJifEWGRGfnIwC8xB5VWEc8I5jywQ==
- Database Name: patsam-cosmosdb
- Container Name: Employee

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": "WKSxZ9fRCCtEYqO7w7lE6FJaPdJo9a8QfGIVgIPnfTYZzkNBhDP3C8BjEAJifEWGRGfnIwC8xB5VWEc8I5jywQ==",
11        "DatabaseName": "patsam-cosmosdb",
12        "ContainerName": "Employee"
13    }
14 }
15 
```

The Schema reference <http://json.schemastore.org/appsettings> is shown above the code editor. On the right, the Solution Explorer pane shows the project structure for 'CosmosDBTutorial' with files like `CosmosDBService.cs`, `EmployeeController.cs`, `Edit.cshtml`, `Startup.cs`, `Program.cs`, and `appsettings.json`.

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

The screenshot shows the 'Keys' tab of the Azure Cosmos DB account 'patsam-cosmosaccount'. The 'Read-only Keys' tab is selected. It displays the following information:

- URI:** https://patsam-cosmosaccount.documents.azure.com:443/
- PRIMARY KEY:** WKSxZ9fRCCtEYqO7w7IE6FJaPdJo9a8QfGIVglPnfTYZzkNBhDP3C8BjEAJifEWGRGfnlwC8xB5VWEc8l5jywQ==
- SECONDARY KEY:** bKRLbg4hZC9HG4PezkwG3GG63EFAOEpcBqeFihfkIzWw6rDMPeFgPhyhLP5En2tk1ajazXzb5vXaOBfwYy8ow==
- PRIMARY CONNECTION STRING:** AccountEndpoint=https://patsam-cosmosaccount.documents.azure.com:443/;AccountKey=WKSxZ9fRCCtEYqO7w7IE6FJaPdJo9a8QfGIVglPnfTYZzkNBhDP3C8Bj...
- SECONDARY CONNECTION STRING:** AccountEndpoint=https://patsam-cosmosaccount.documents.azure.com:443/;AccountKey=bKRLbg4hZC9HG4PezkwG3GG63EFAOEpcBqeFihfkIzWw6rDMPeF...

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

The screenshot shows the 'Data Explorer' tab of the Azure Cosmos DB account 'patsam-cosmosaccount'. The 'Data Explorer' tab is selected. It displays the following structure:

- SQL API**
- patsam-cosmosdb** (selected)
  - Scale**
  - Employee**

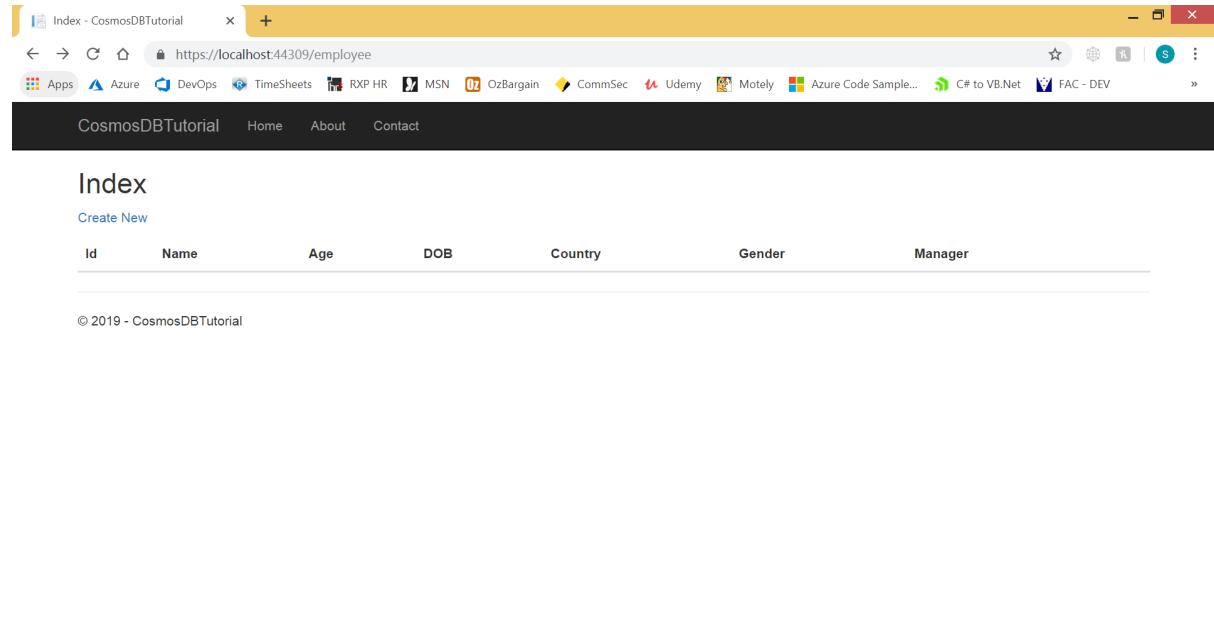
The interface includes a large planet icon with stars, a welcome message 'Welcome to Azure Cosmos DB', and a note 'Create new or work with existing container(s.)'.

## **Step 07: CRUD Operations**

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

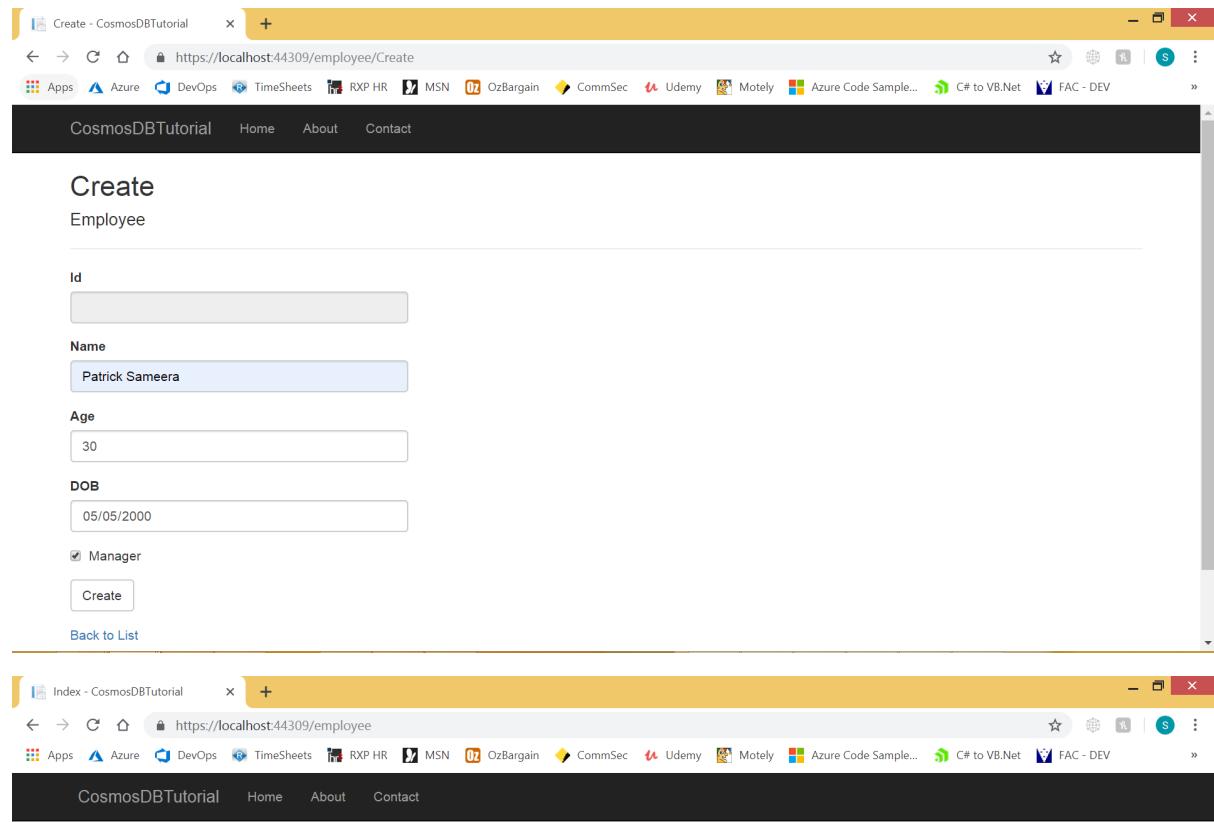
<https://localhost:44309/employee>

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:



**Create**  
Employee

**Id**

**Name**

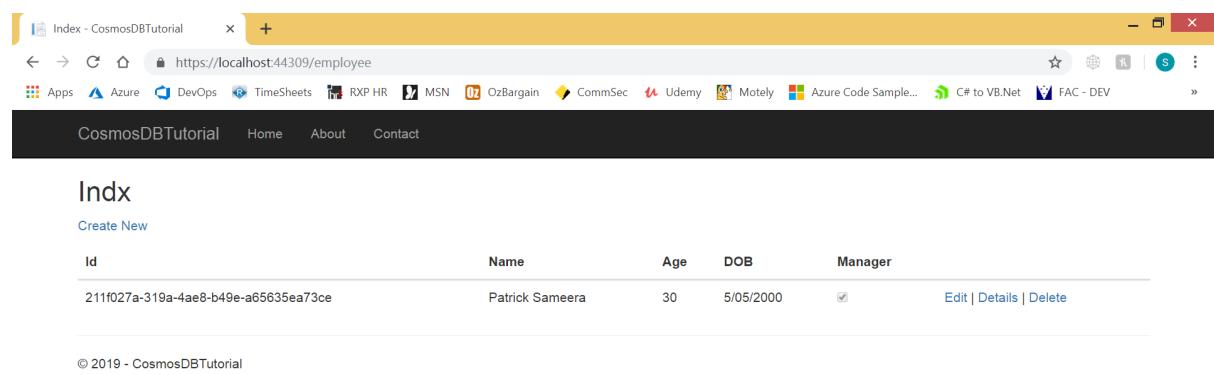
**Age**

**DOB**

Manager

**Create**

[Back to List](#)



**Index**

[Create New](#)

<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>DOB</b>	<b>Manager</b>	
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' and navigation links for 'Home', 'About', and 'Contact'. Below the header is a form titled 'Edit Employee'. The form contains fields for 'Id' (with value 211f027a-319a-4ae8-b49e-a65635ea73ce), 'Name' (with value Patrick Sameera Jayalath), 'Age' (with value 30), 'DOB' (with value 05/05/2000), and a checked checkbox for 'Manager'. There is also a 'Save' button and a link to 'Back to List'.

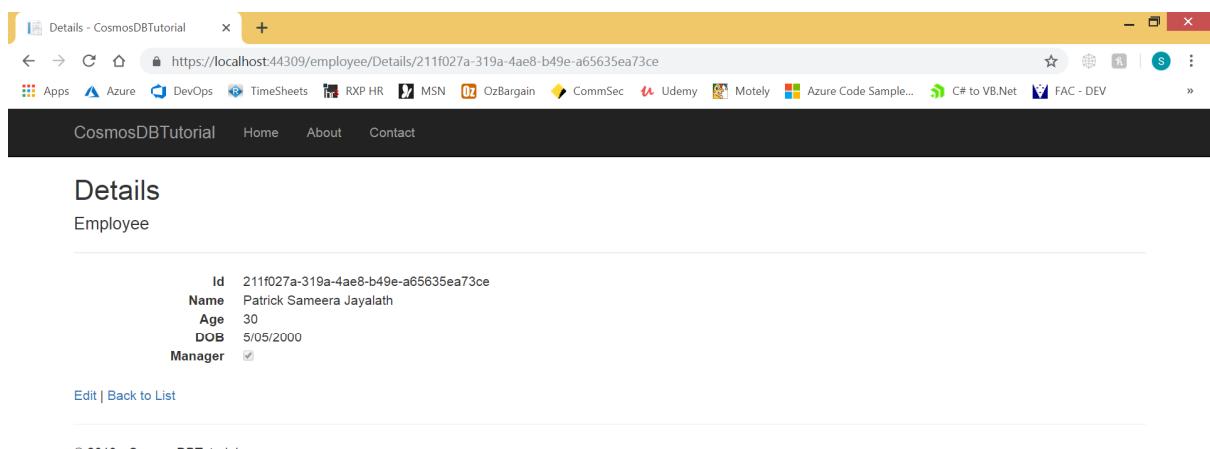
The screenshot shows a web browser window titled 'Index - CosmosDBTutorial'. The URL is https://localhost:44309/employee. The page has a header with 'CosmosDBTutorial' and navigation links for 'Home', 'About', and 'Contact'. Below the header is a table titled 'Idx' with the following data:

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

At the bottom of the page, there is a 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 page content displays the details of an employee named Patrick Sameera Jayalath, with the following information:

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

Below the table, there are links for "Edit" and "Back to List". The footer of the page includes the 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 asks "Are you sure you want to delete this? Employee". Below the question, there is a table with the following data:

<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"/>

At the bottom left is a "Delete" button, and at the bottom right is a "Back to List" link. The footer of the page says "© 2019 - CosmosDBTutorial".

We can inspect the data through Data Explorer tab.

The screenshot shows the "Data Explorer" tab in the Azure Cosmos DB portal for the database "patsam-cosmosaccount". The left sidebar shows various management options like "Diagnose and solve problems", "Quick start", "Notifications", and "Data Explorer" (which is selected). The main area shows the "Employee" collection under the "patsam-cosmosdb" database. A SQL query "SELECT \* FROM c" is run against the "Employee" collection, and the results are displayed in a table and a JSON pane. The JSON pane shows the following document:

```
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": "'4c00ed17-0000-0300-0000-5da0fe000000'",
9   "_attachments": "attachments/",
10  "_ts": 1570770912
```