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

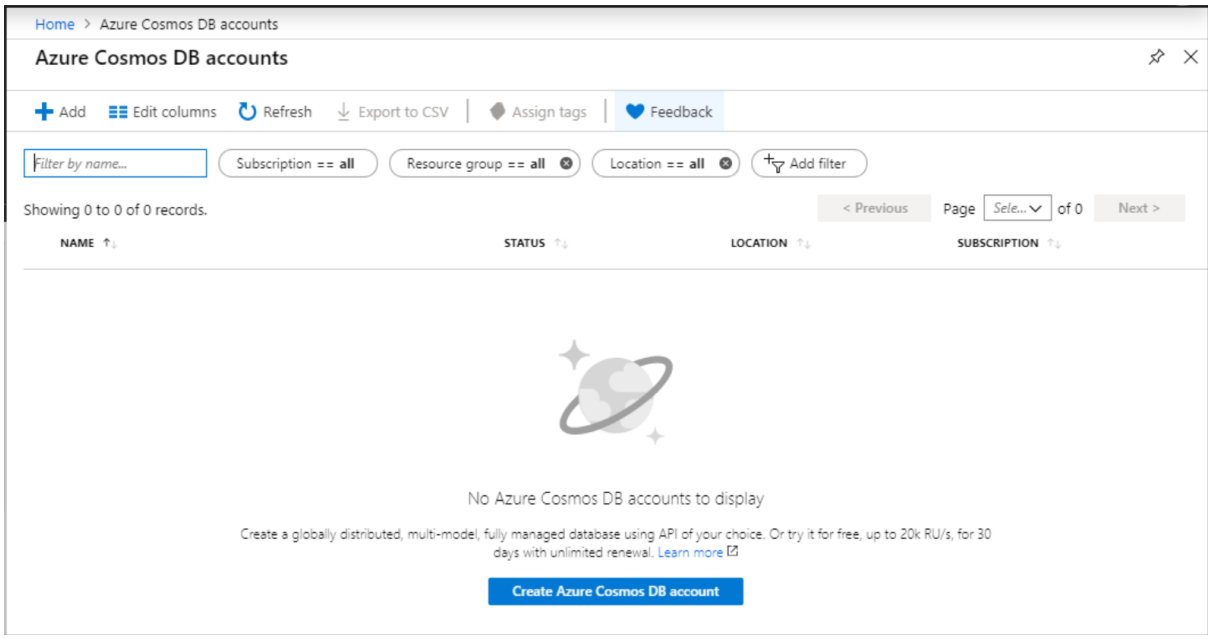
Source code for this Project can be found here:

https://github.com/patricksameerajayalath/CosmosDBTutorial

Patrick Sameera Jayalath

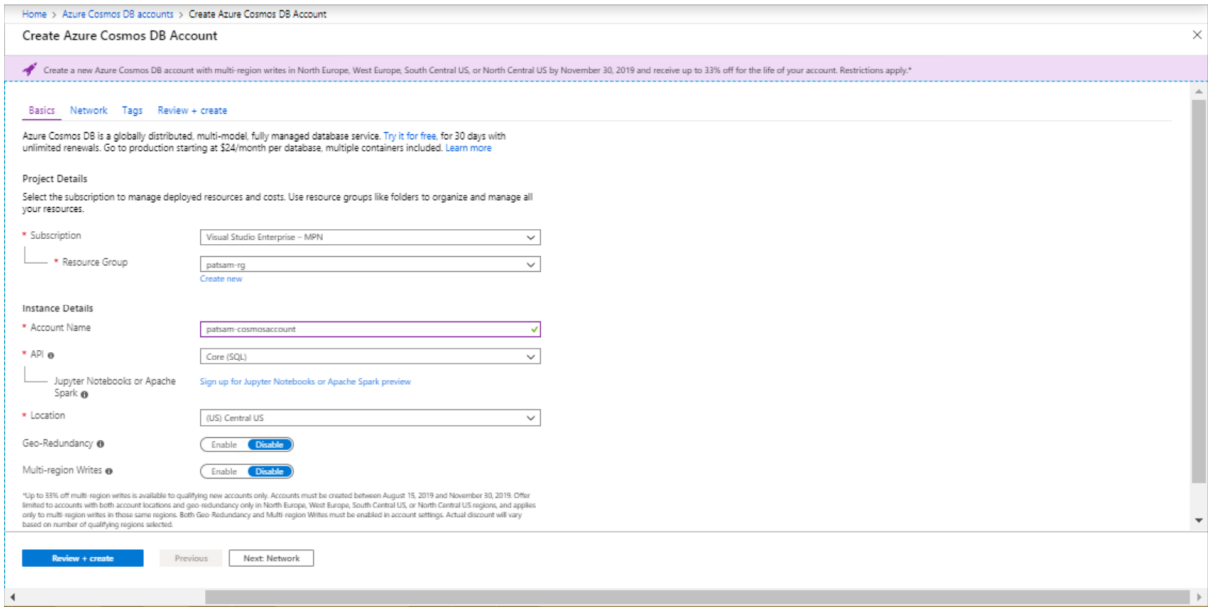
Step 01: Create an Azure Cosmos DB Account

Click + Add to add new Cosmos DB Account.



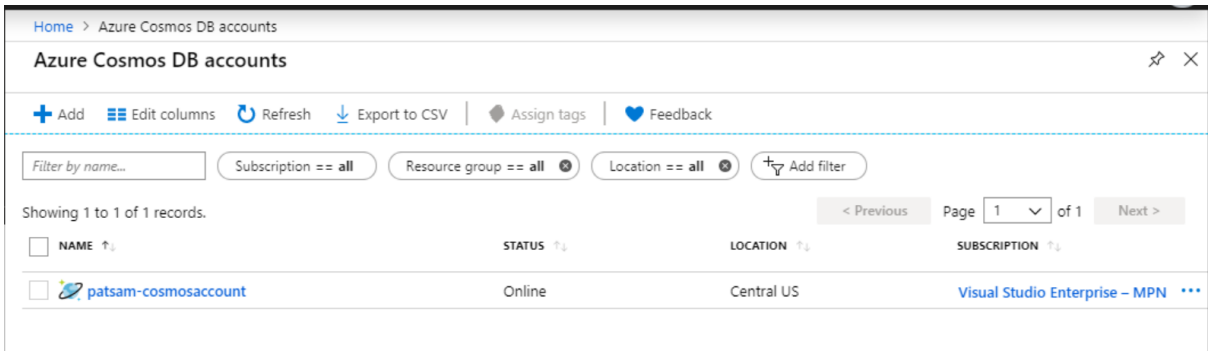
Provide relevant details and click Review + Create.

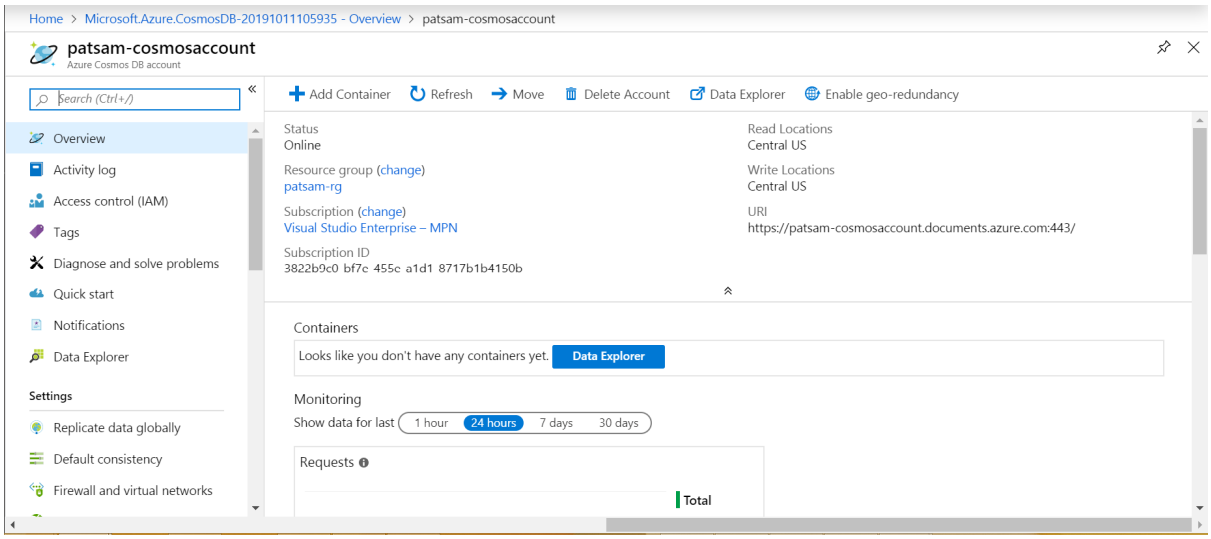
- Resource group: patsam-rg
- Account name: patsam-cosmosaccount
- API: Core (SQL)
- Location: Central US



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.

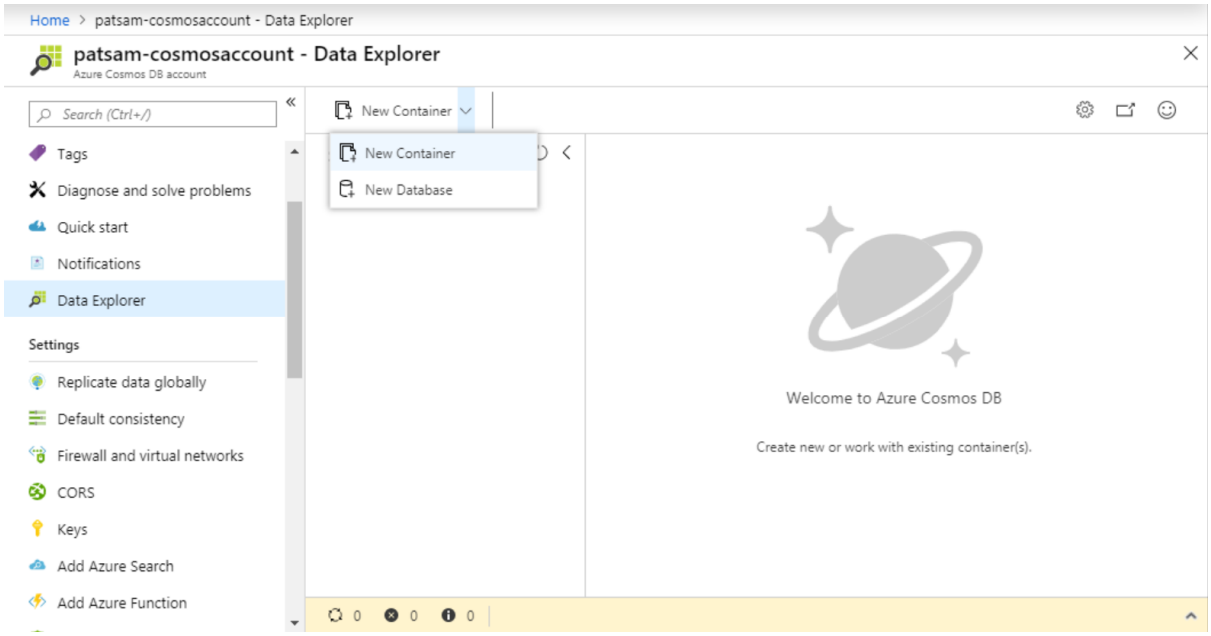




Step 02: Create an Azure Cosmos DB

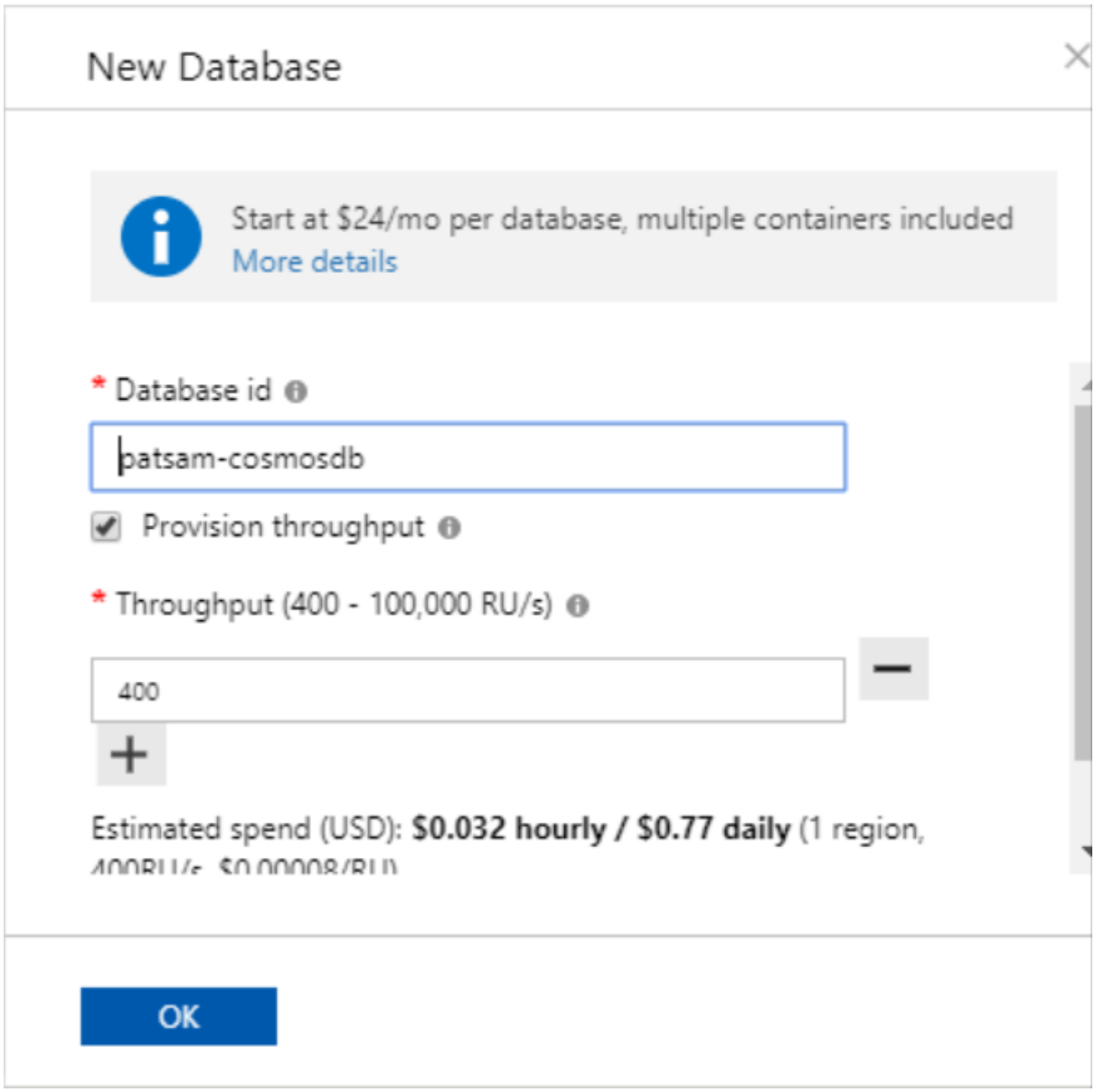
Next, we need to add a Cosmos DB. Click add New Database.

Data Explorer → New Database



Provide relevant details and Click OK.

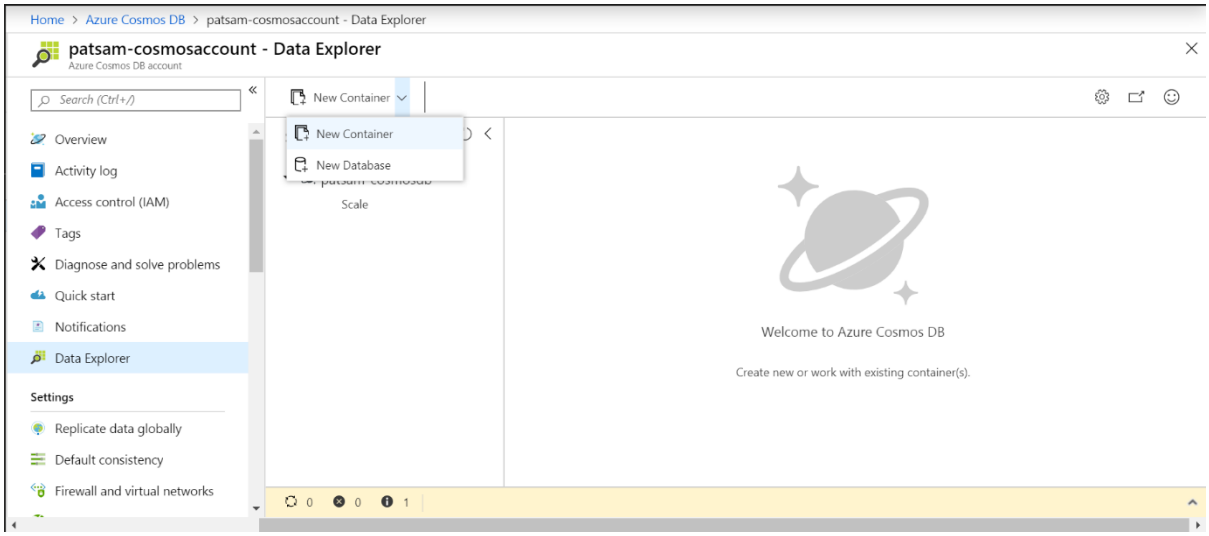
- Database id: patsam-cosmosdb



Step 03: Add a Container to the Cosmos DB

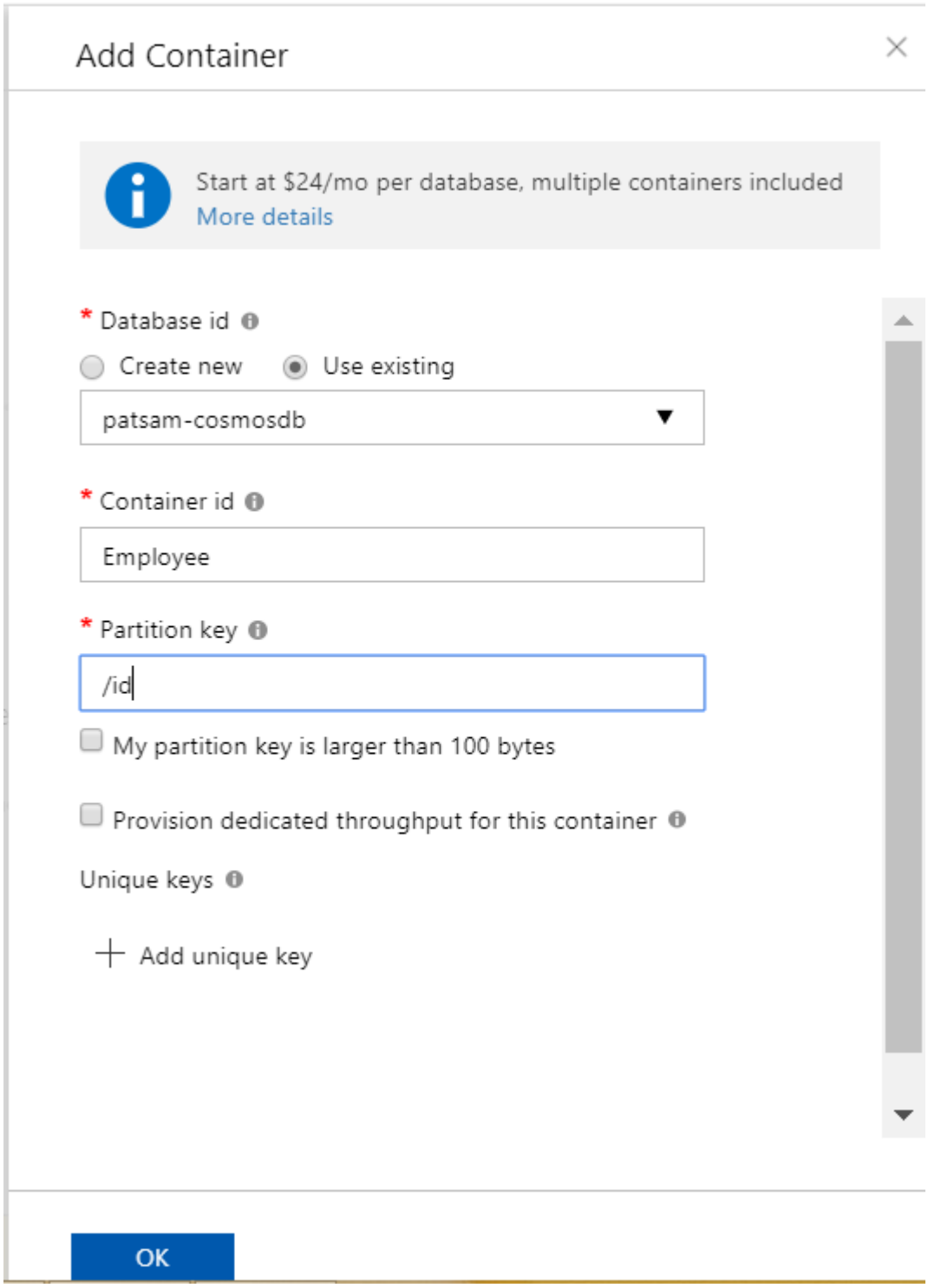
Next, we need to add a Container. Click on New Container.

Data Explorer → New Container



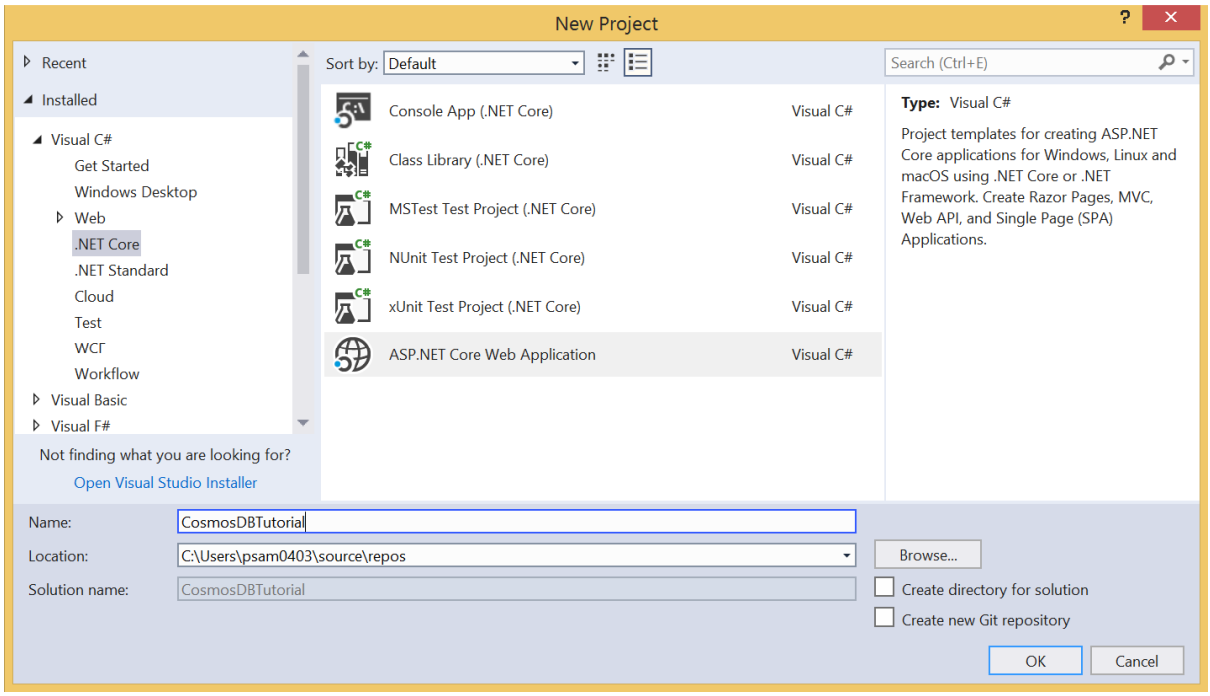
Provide relevant data and Click OK.

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

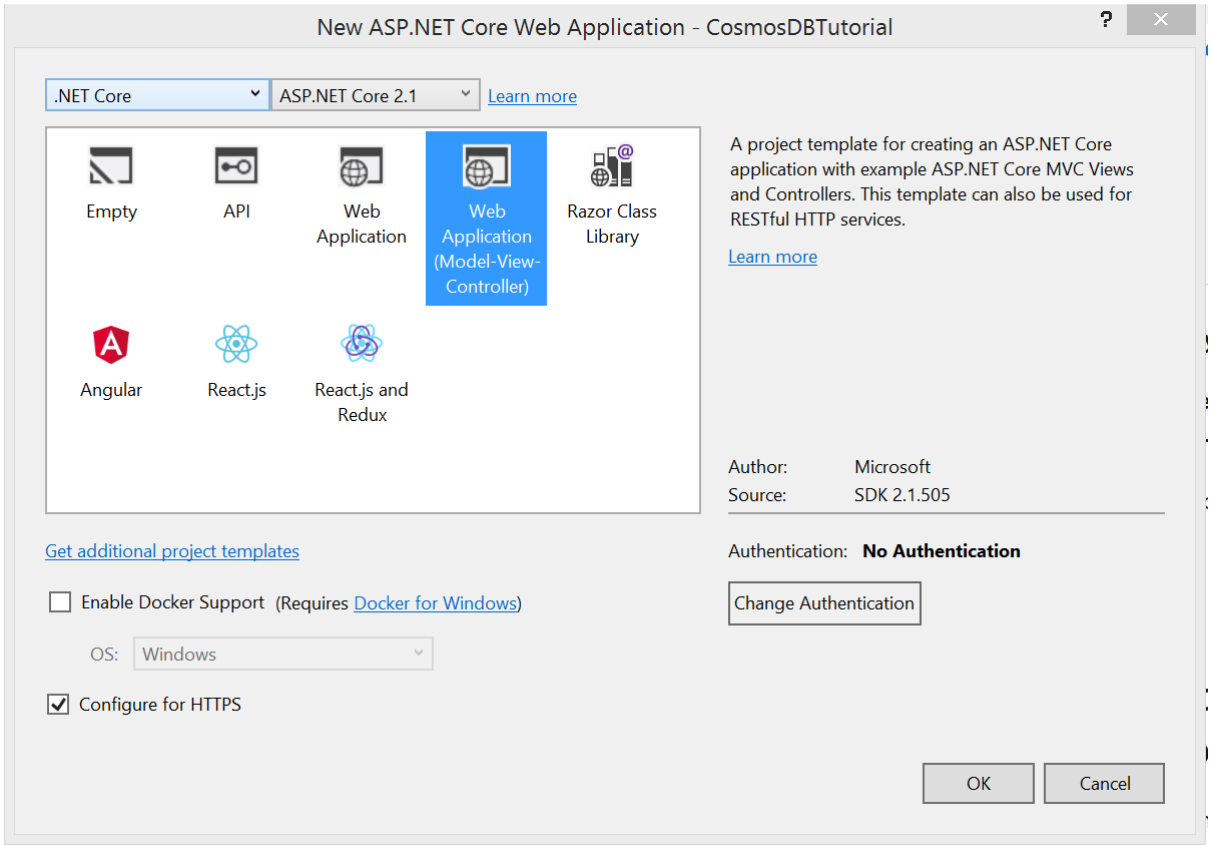

Add Container
Start at $24/mo per database, multiple containers included
More details
* Database id
Create new
Use existing
patsam-cosmosdb
* Container id
Employee
* Partition key
/id
My partition key is larger than 100 bytes
Provision dedicated throughput for this container
Unique keys
Add unique key
OK

Step 04: Create an ASP.NET Core MVC Application

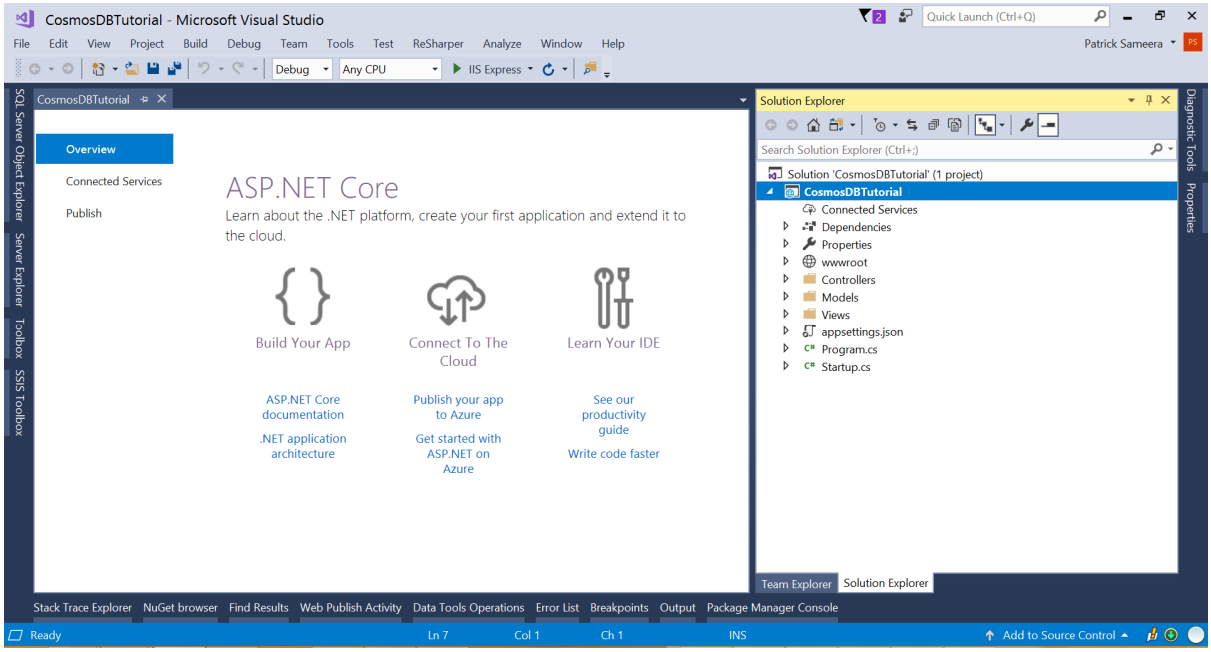
Create a new .Net Core Web Project.

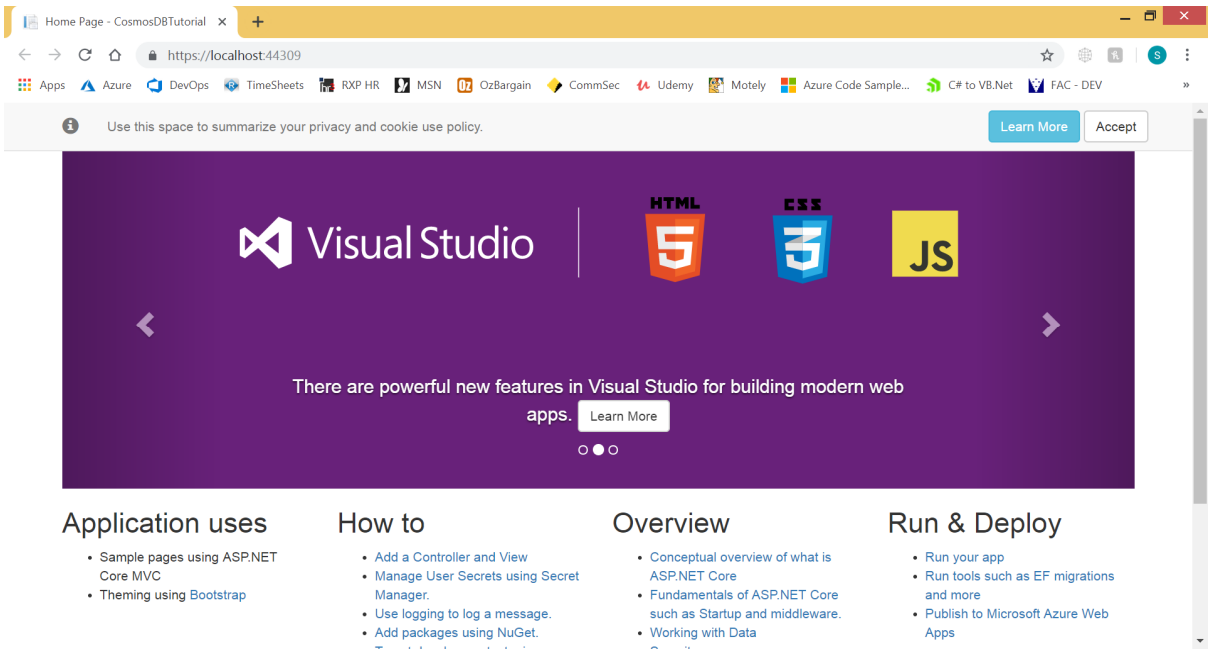


Select MVC.



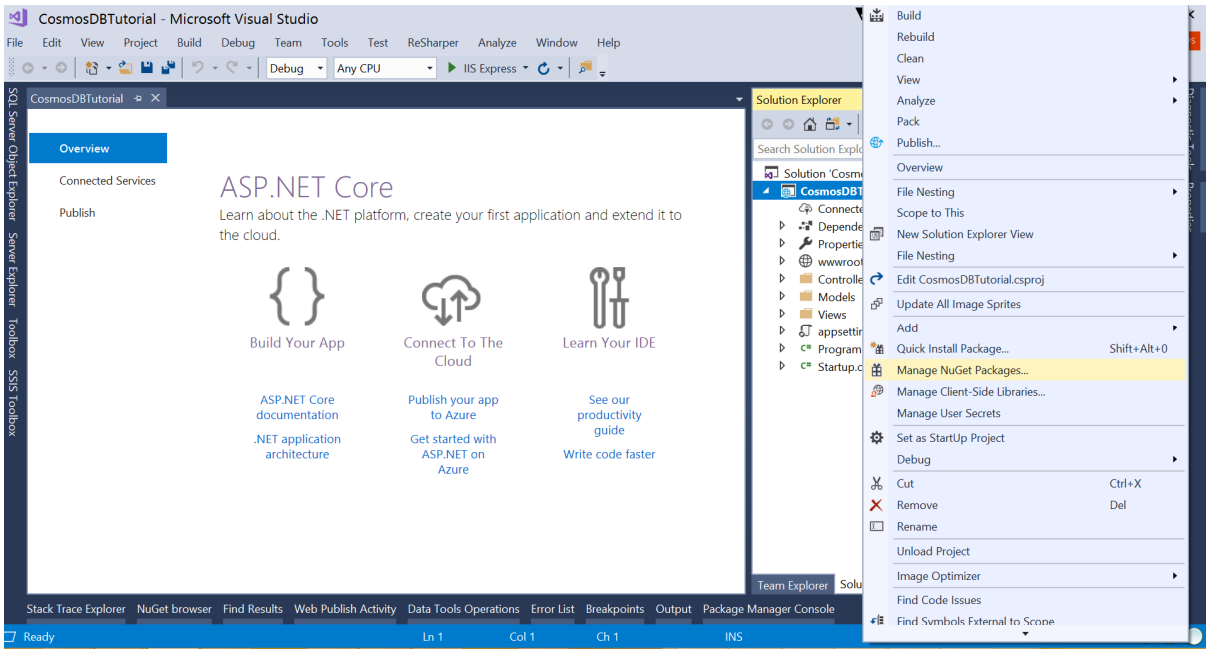
Once Project gets created Click F5 to run the Project, just to make sure it compiles properly.



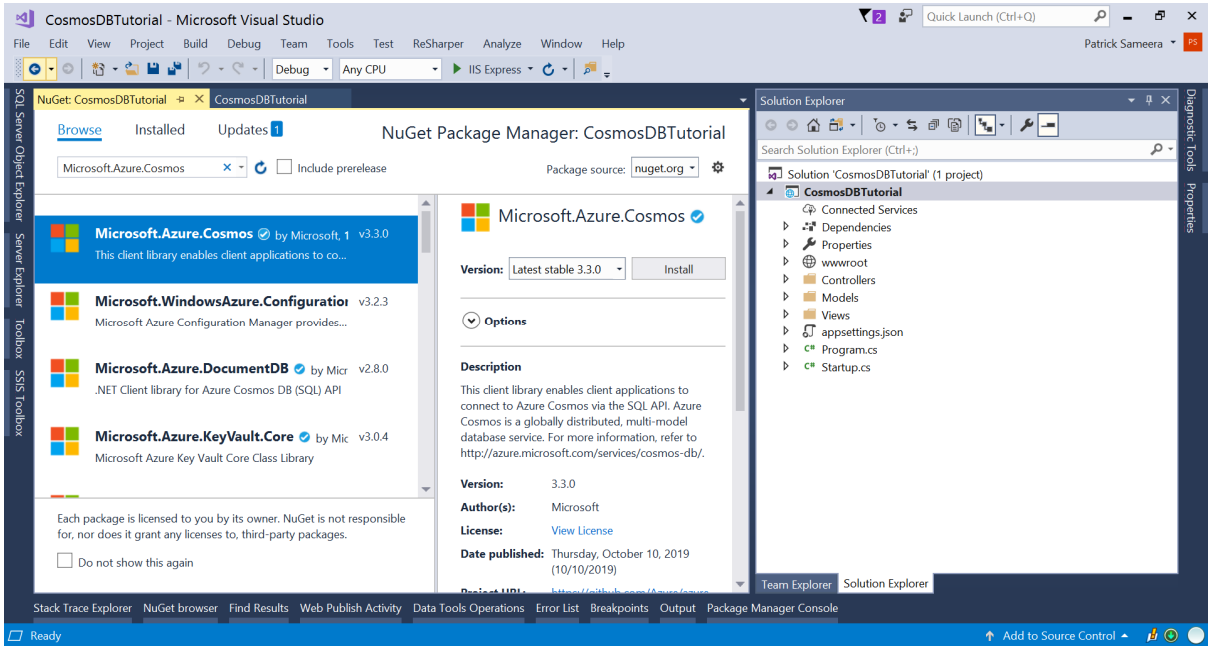


Step 05: Add Azure Cosmos DB NuGet package to the Project

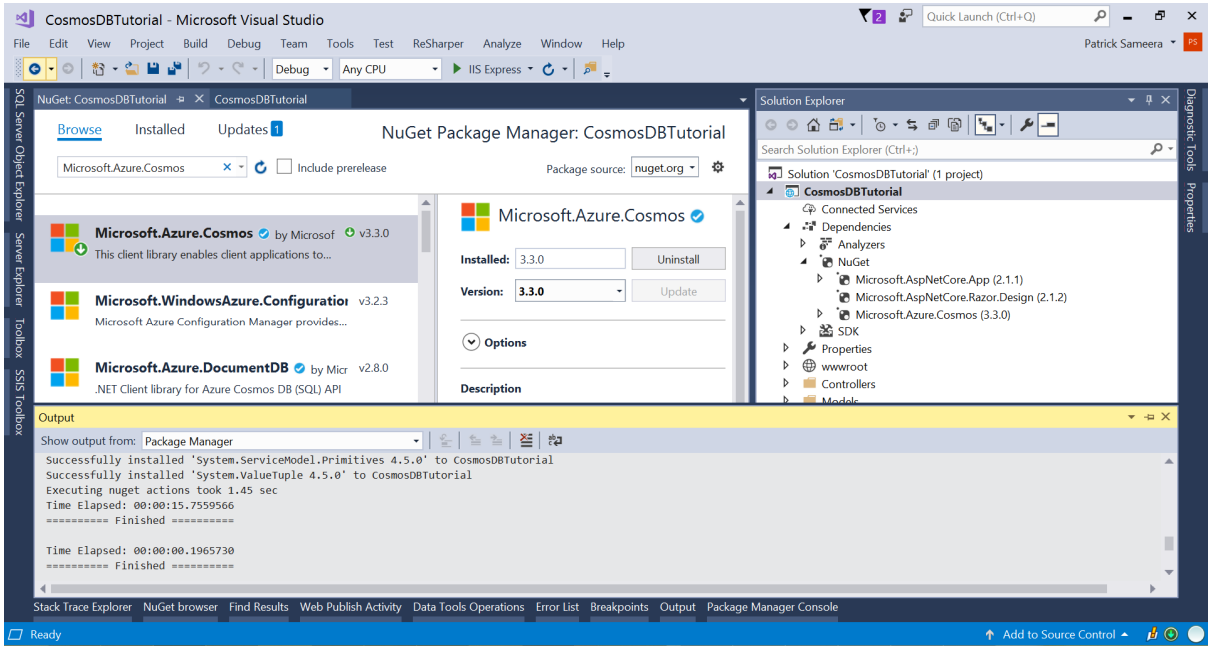
In the Solution Explorer, right click on the Project and select Manage NuGet Packages.



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



Make sure it gets installed successfully.



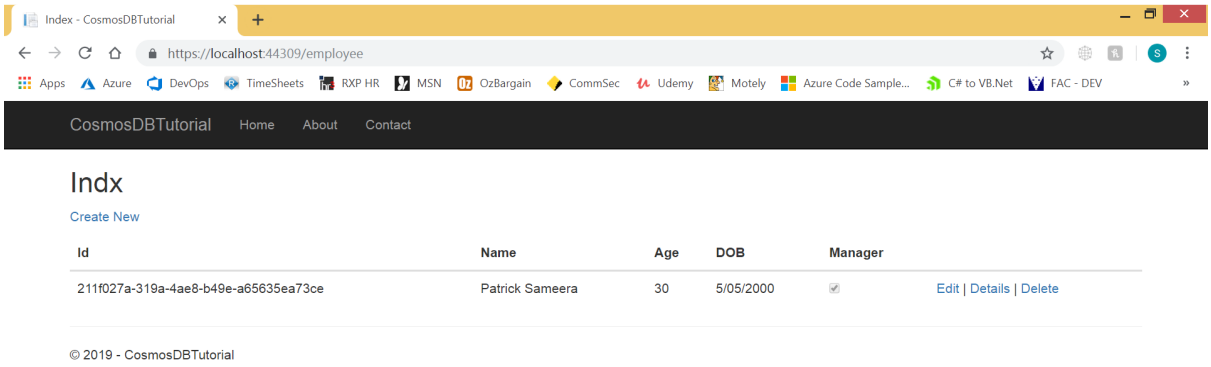
Patrick Sameera Jayalath

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

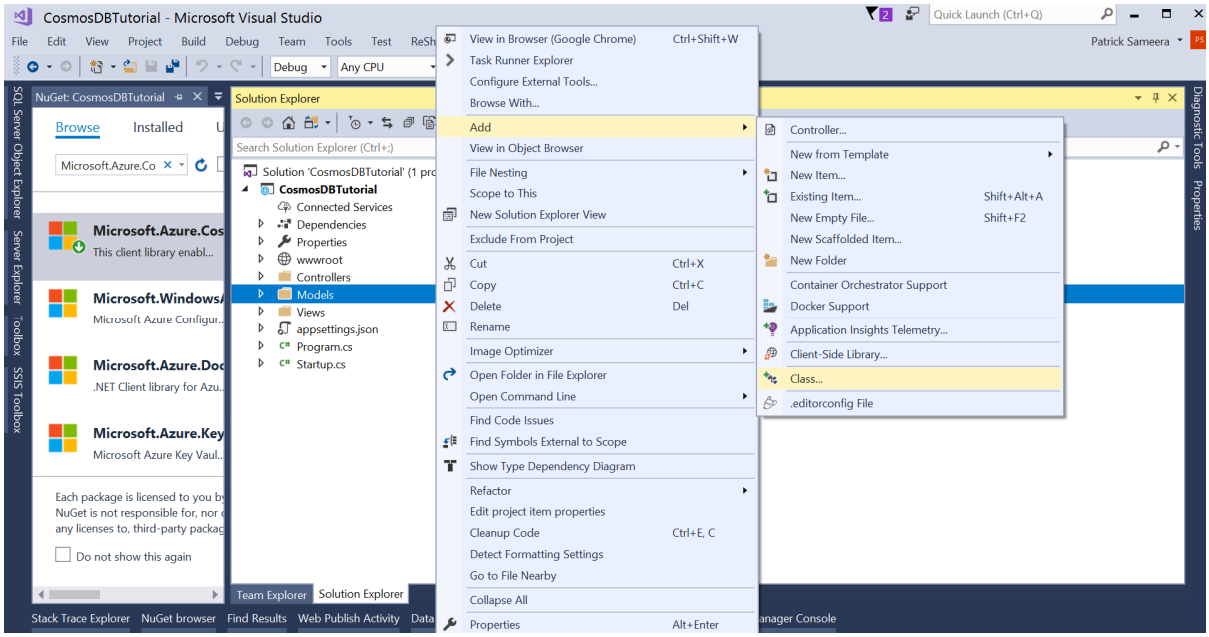
We are trying to achieve something like bellow:



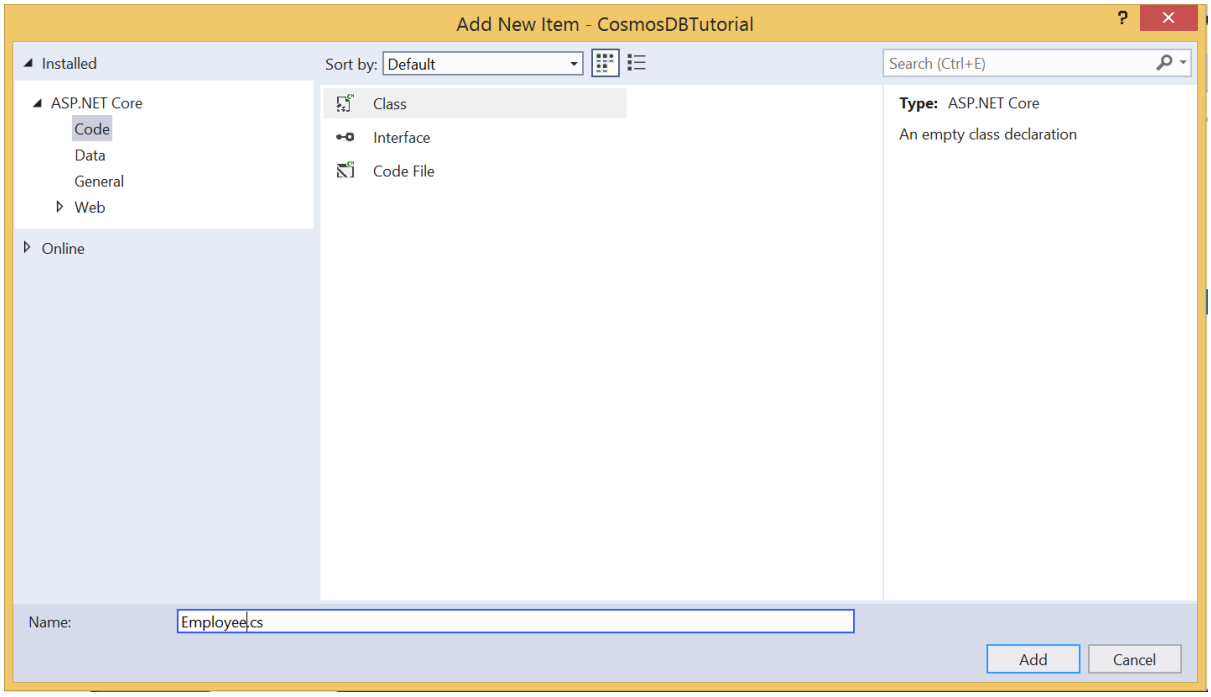
Adding Model

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

- Models → Add → Class

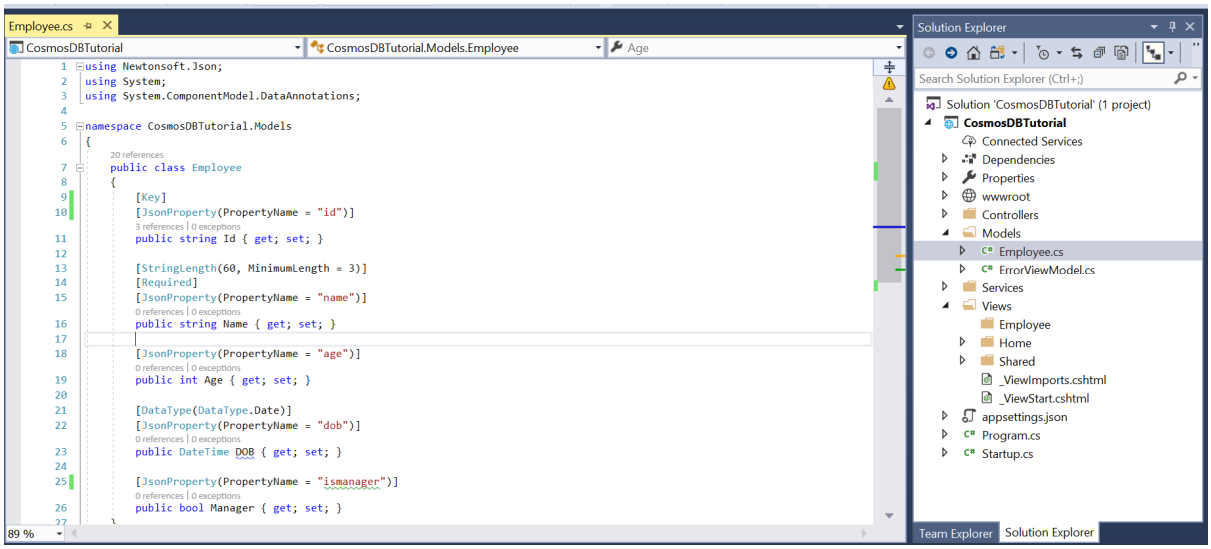


Name it Employee.cs



Employee.cs

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



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.

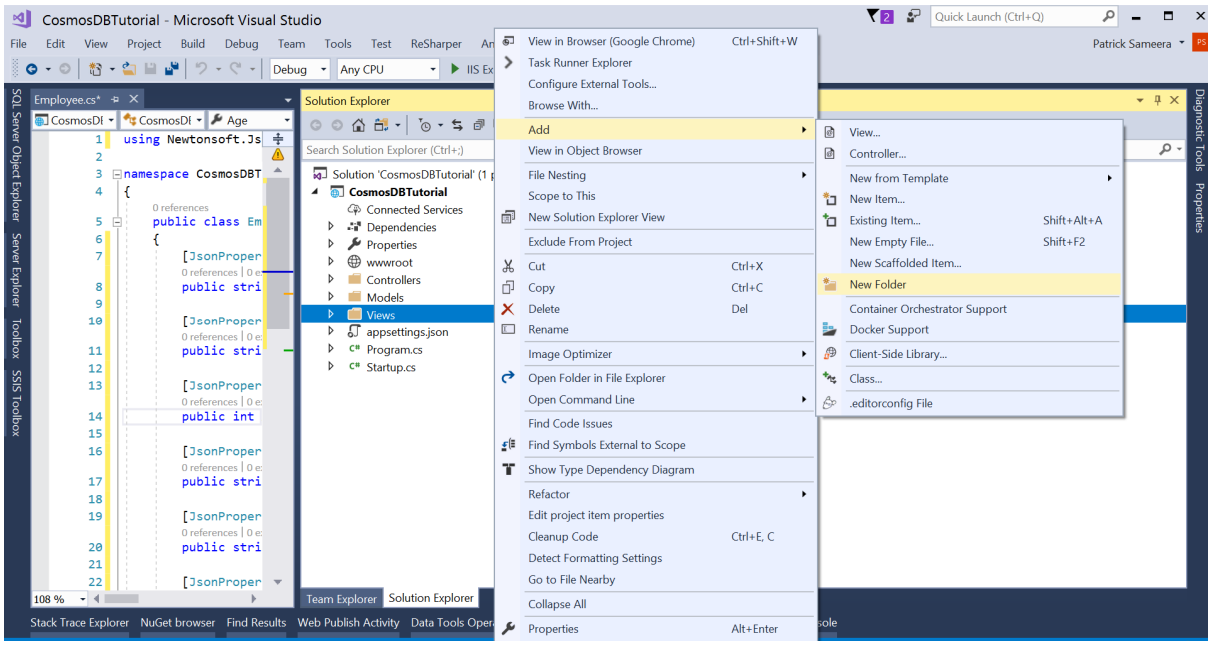
Adding Views

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

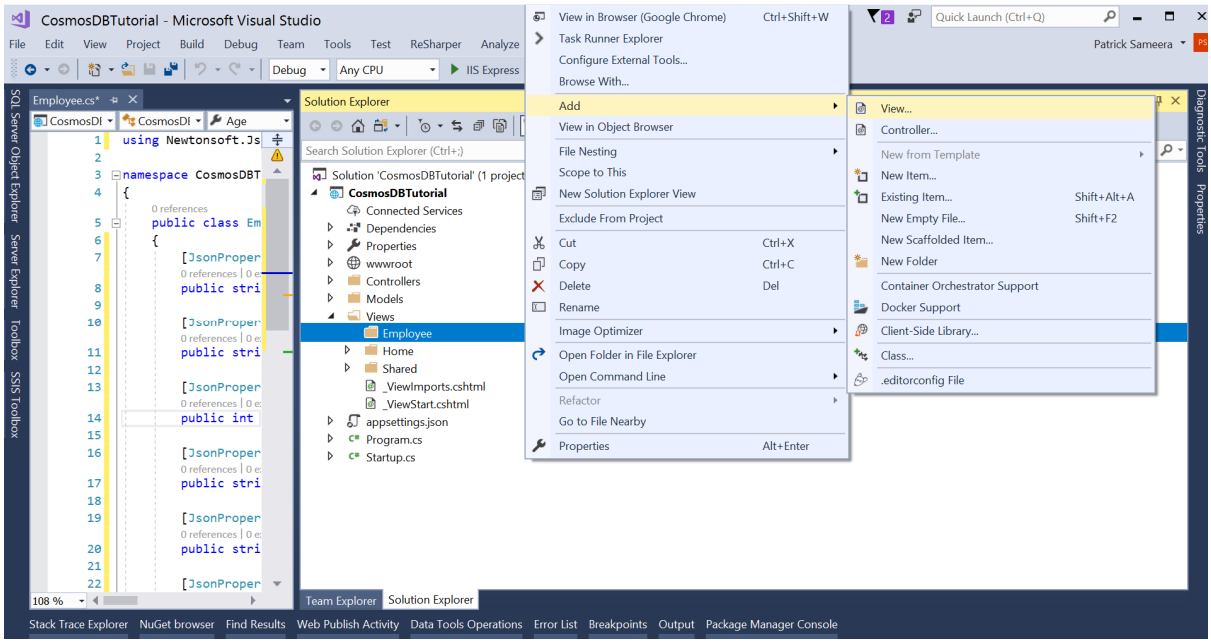
First create a new Folder under Views by the name Employee. Right Click on Views.

- Views → Add → New Folder → Employee



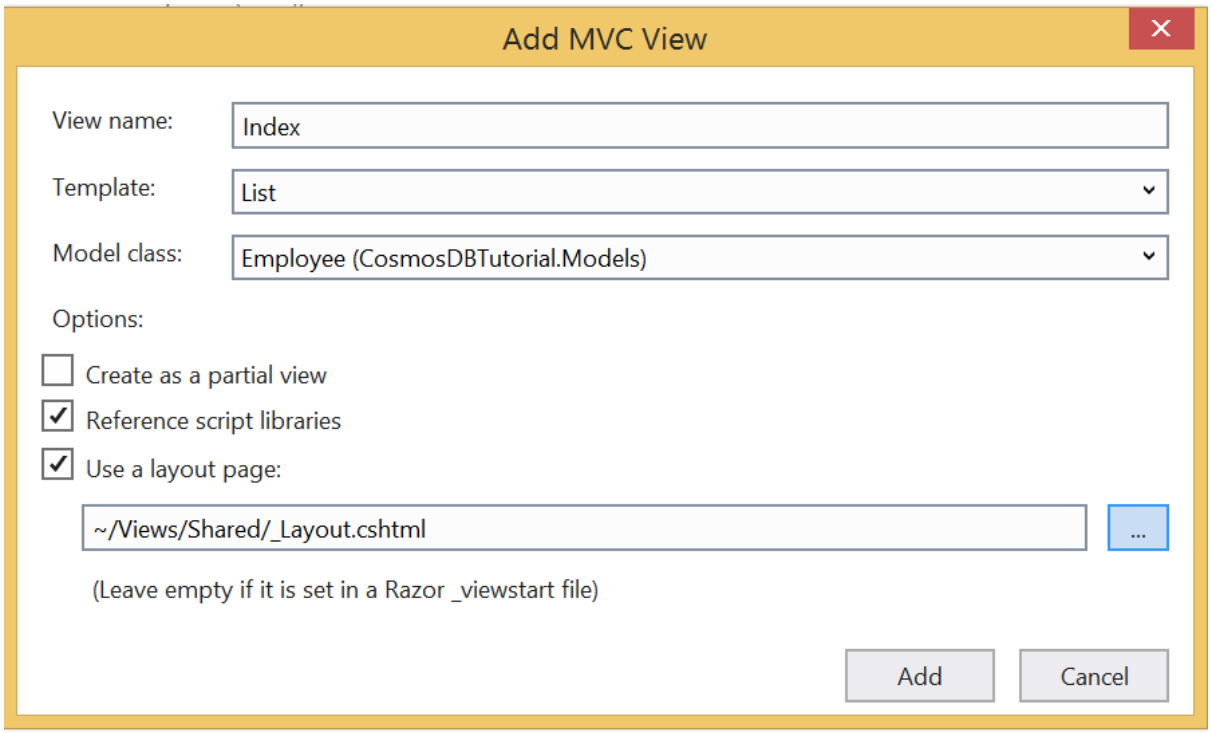
Next, we need to add the 5 Views. Right Click on Employee Folder under Views to add Views.

- Views → Employees → Add → View

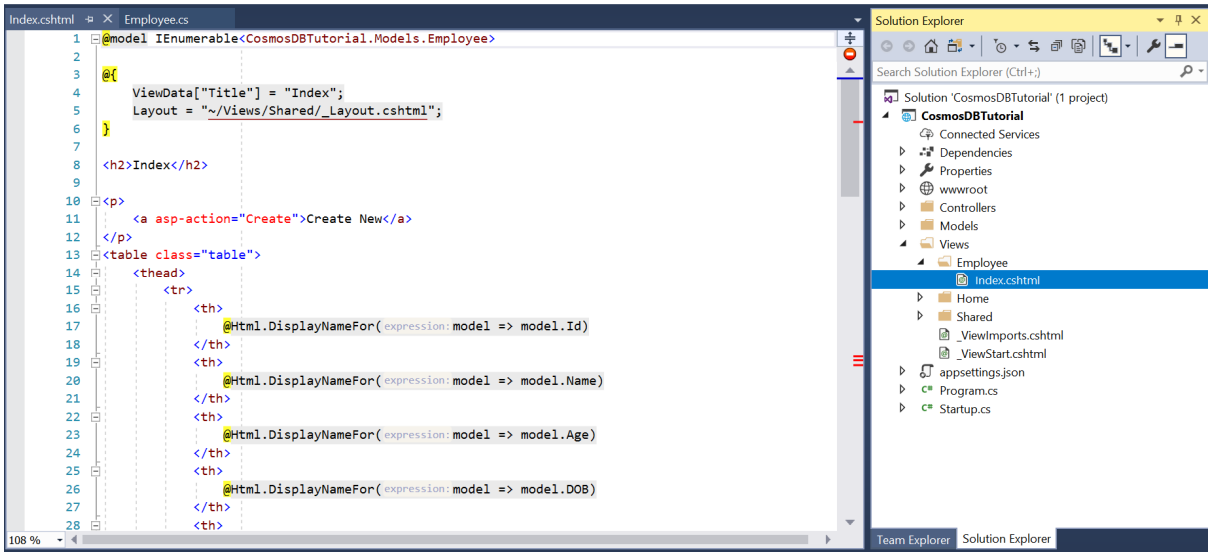


View 01: Index

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

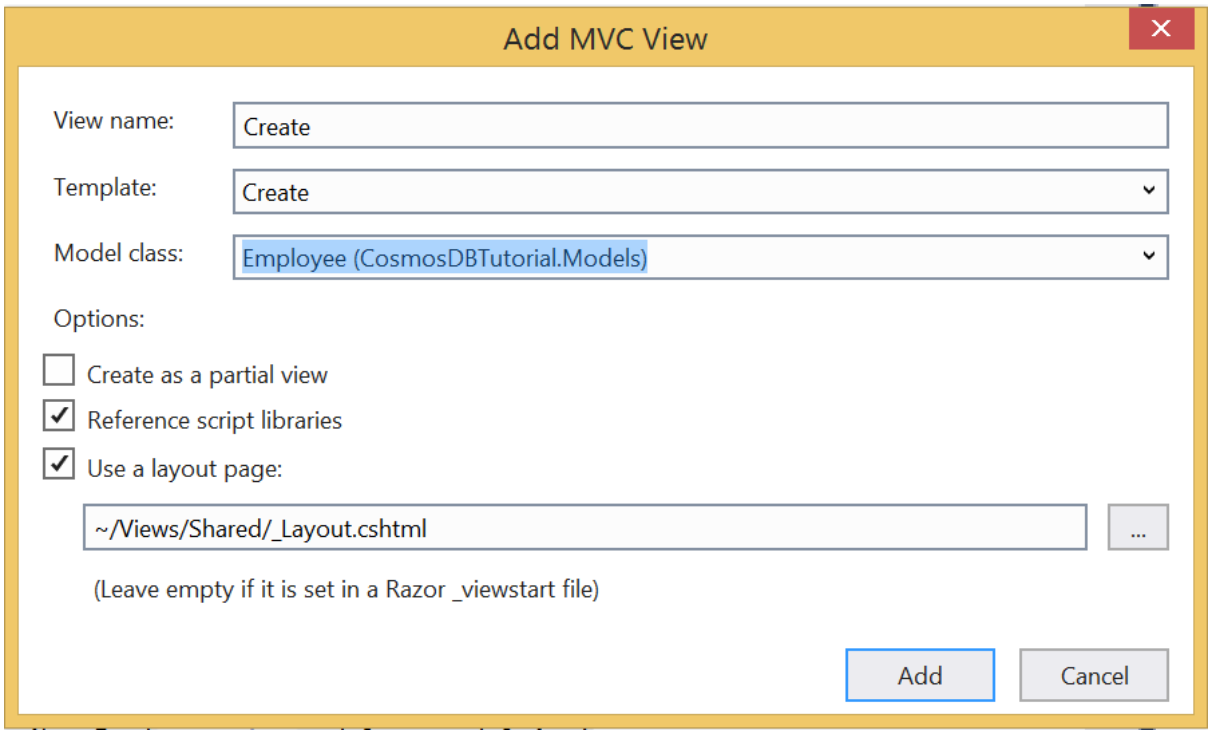


https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Index.cshtml

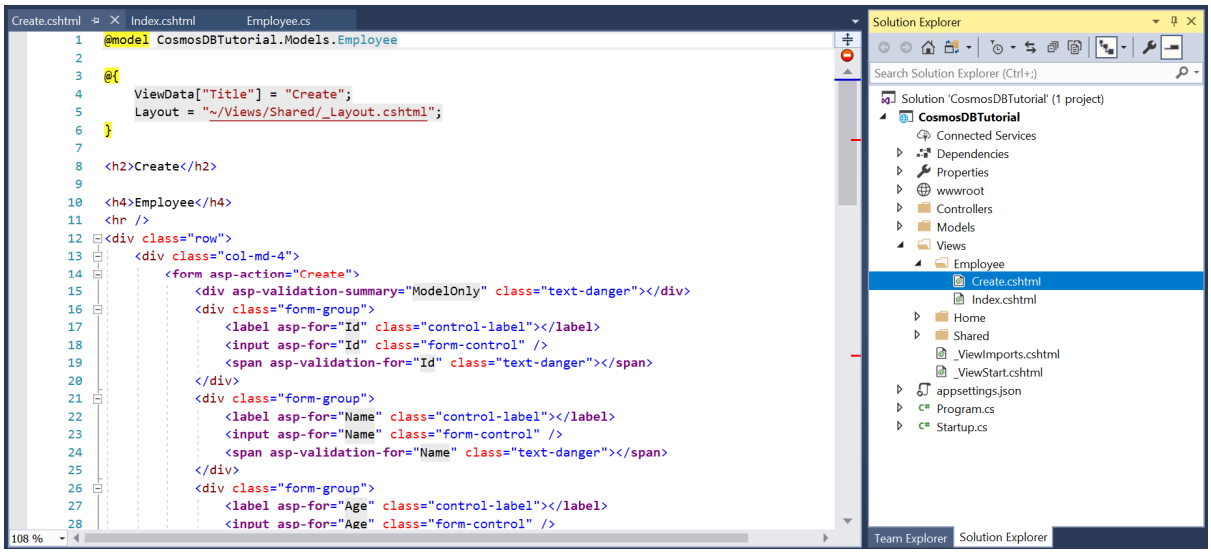


View 02: Create

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

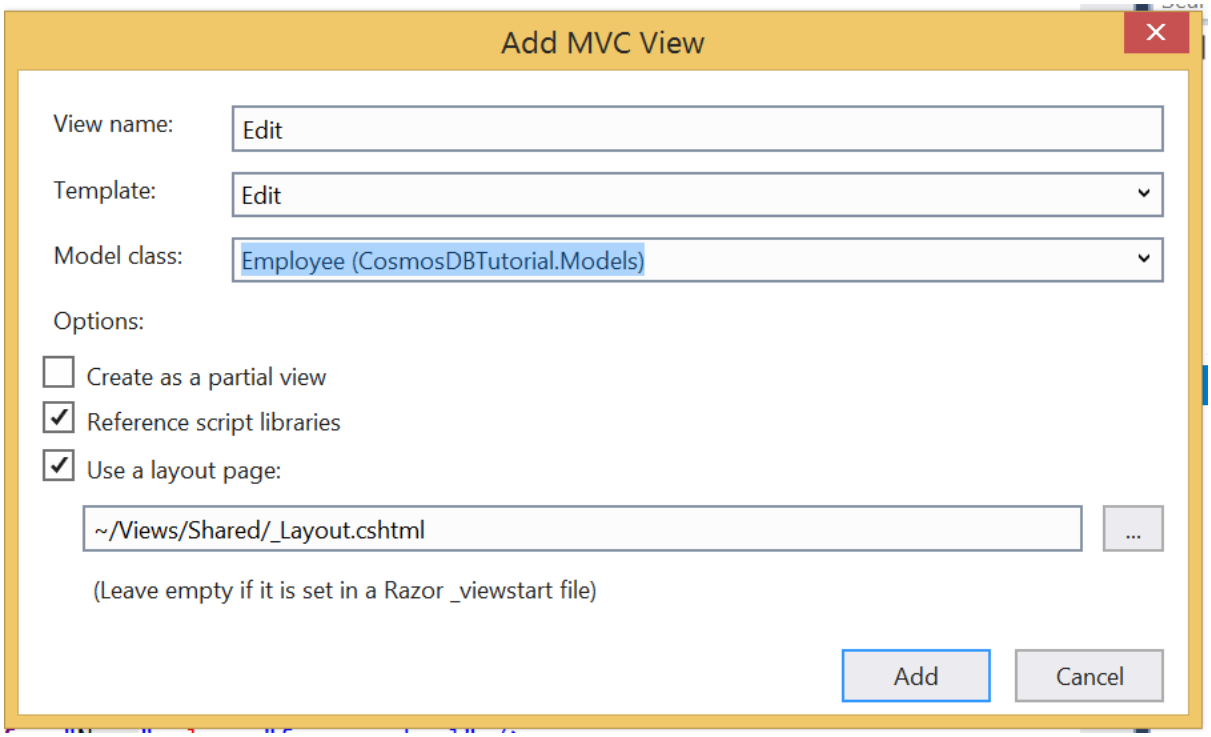


https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Create.cshtml

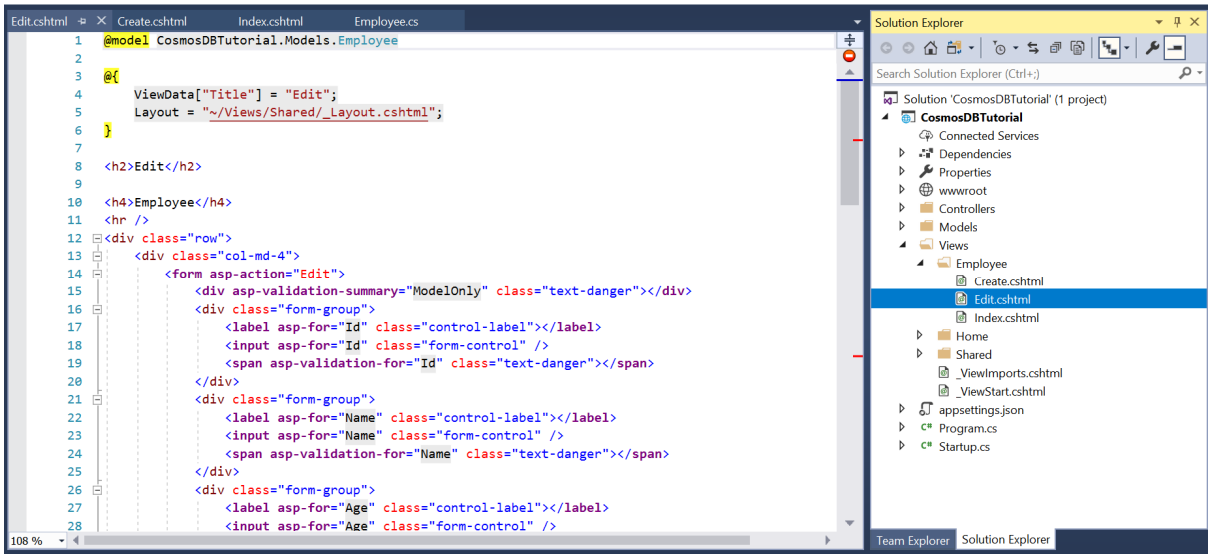


View 03: Edit

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

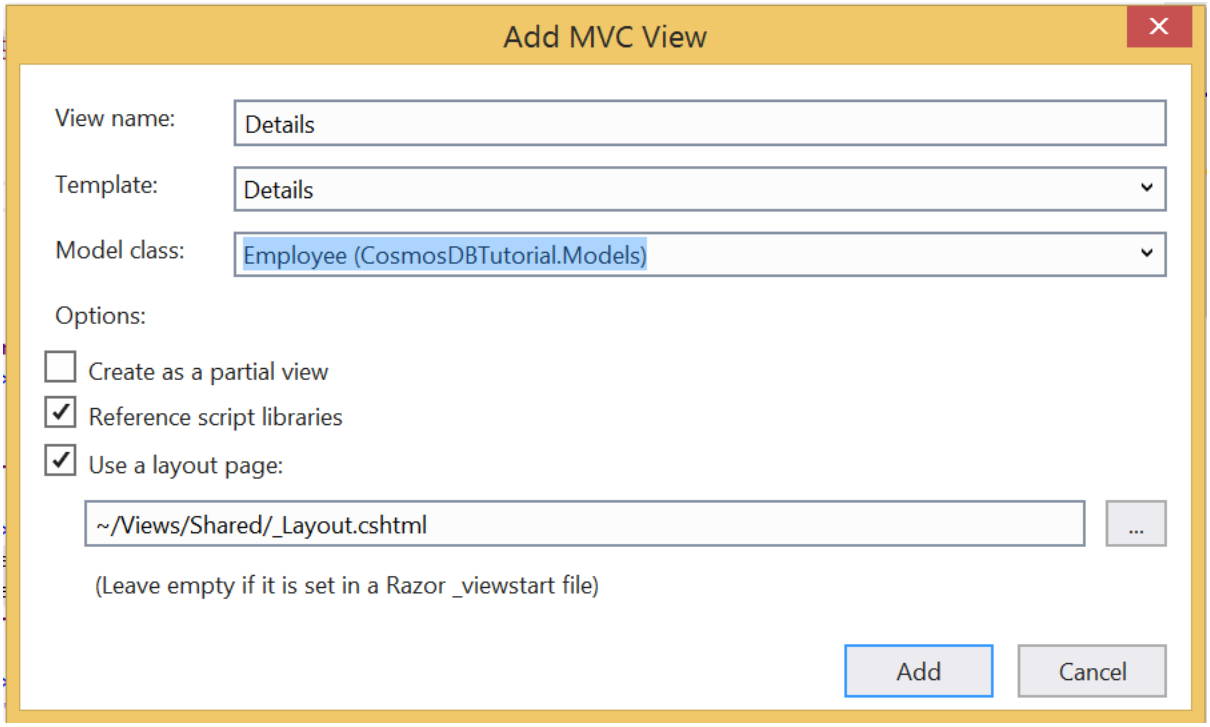


https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Edit.cshtml

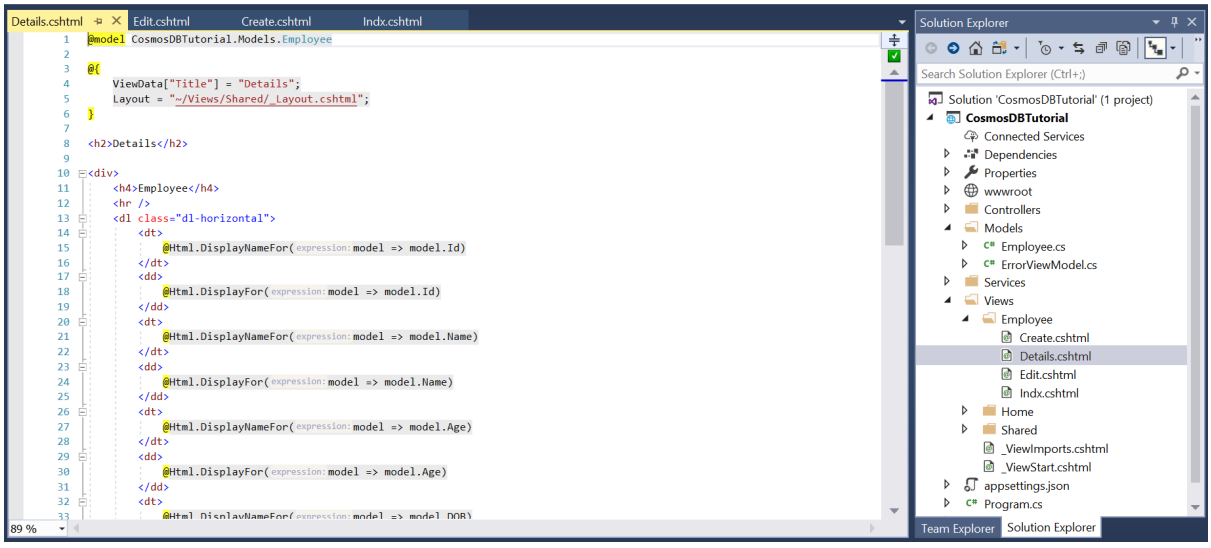


View 04: Details

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

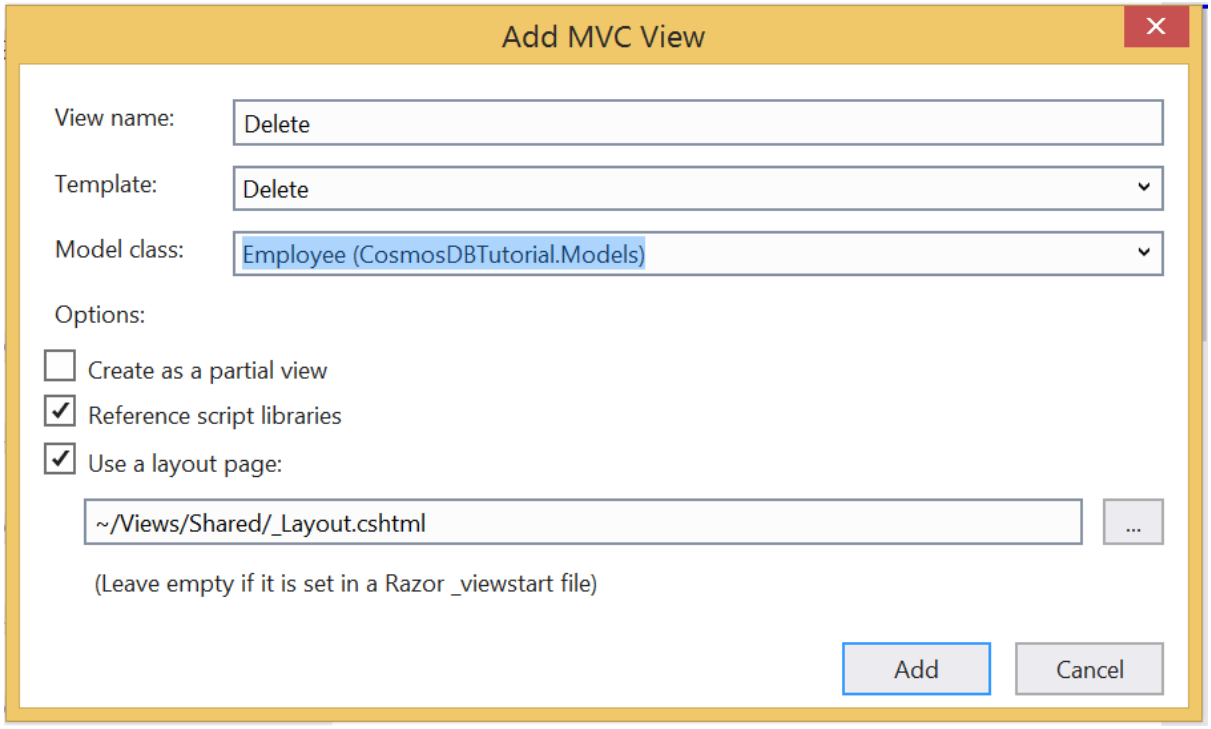


https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Details.cshtml

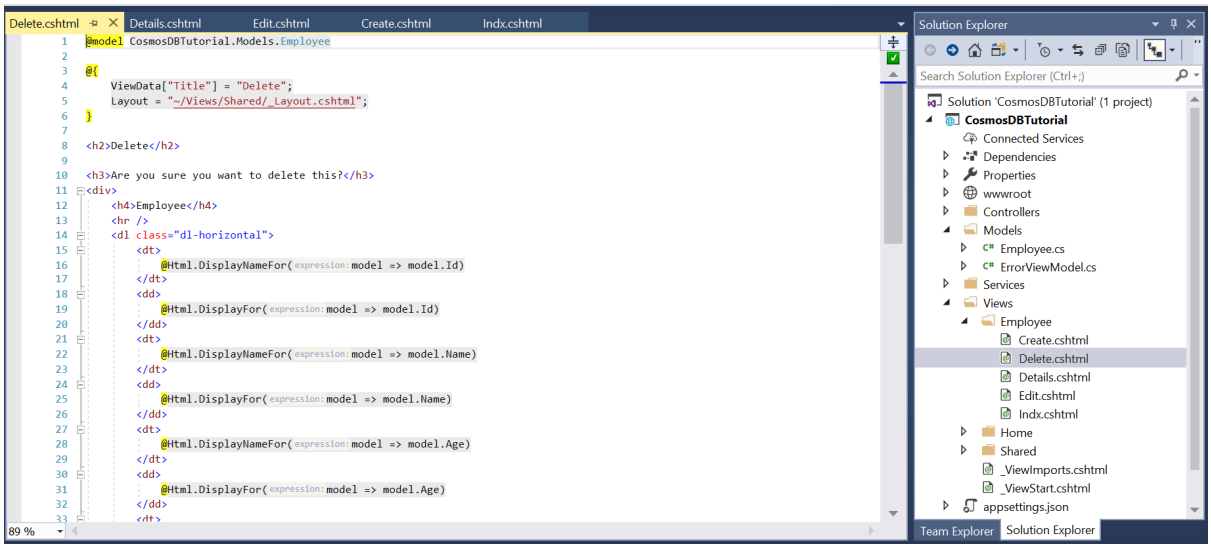


View 04: Delete

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



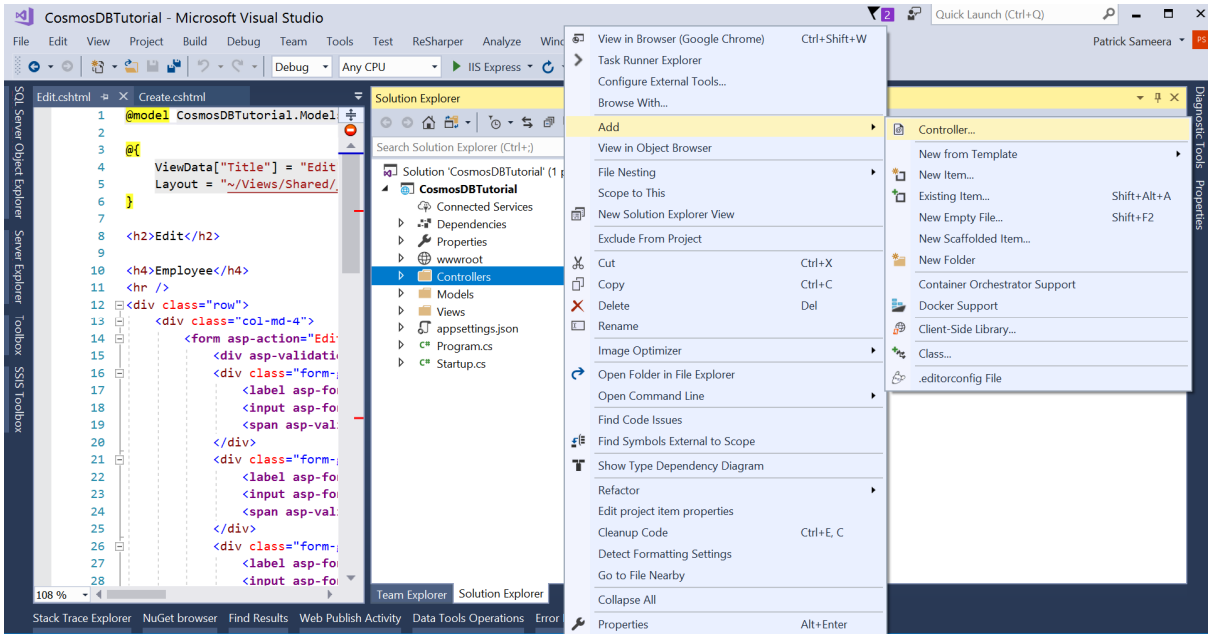
https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/Views/Employee/Delete.cshtml



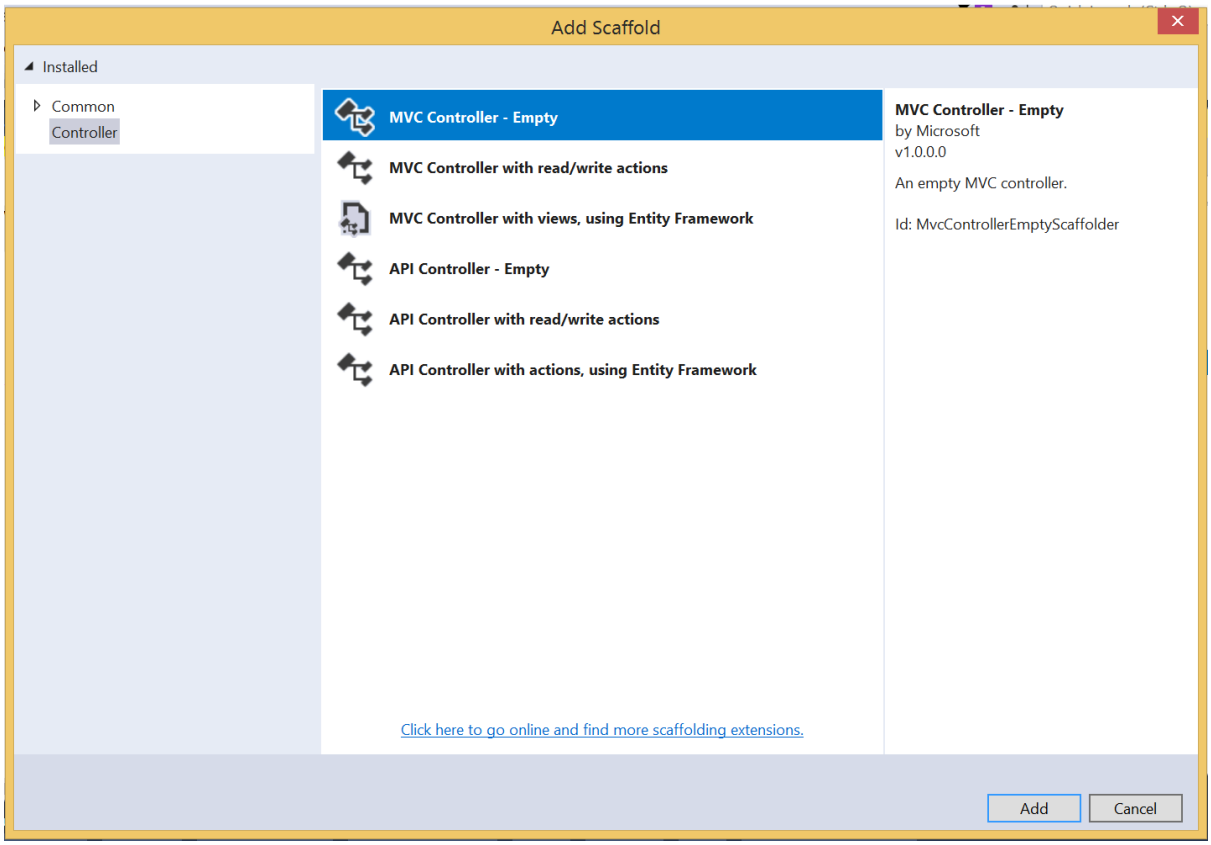
Adding Controllers

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

Controllers → Add → Controller

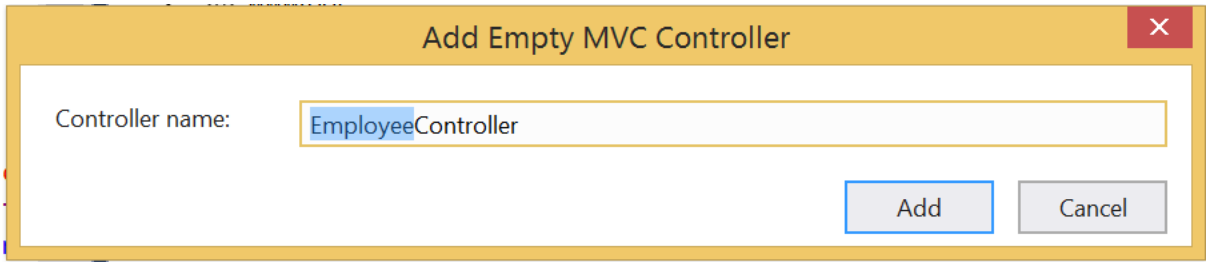


Select MVC Controller – Empty and Click Add.



Patrick Sameera Jayalath

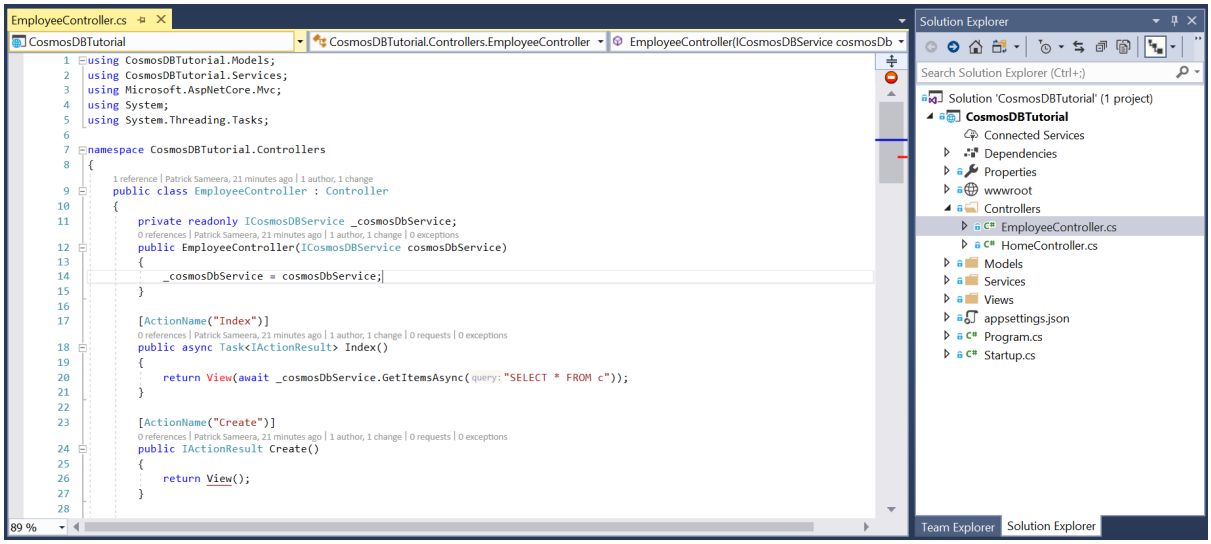
Name it EmployeeController.



EmployeeController.cs

Copy paste the code to the file.

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



Patrick Sameera Jayalath

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

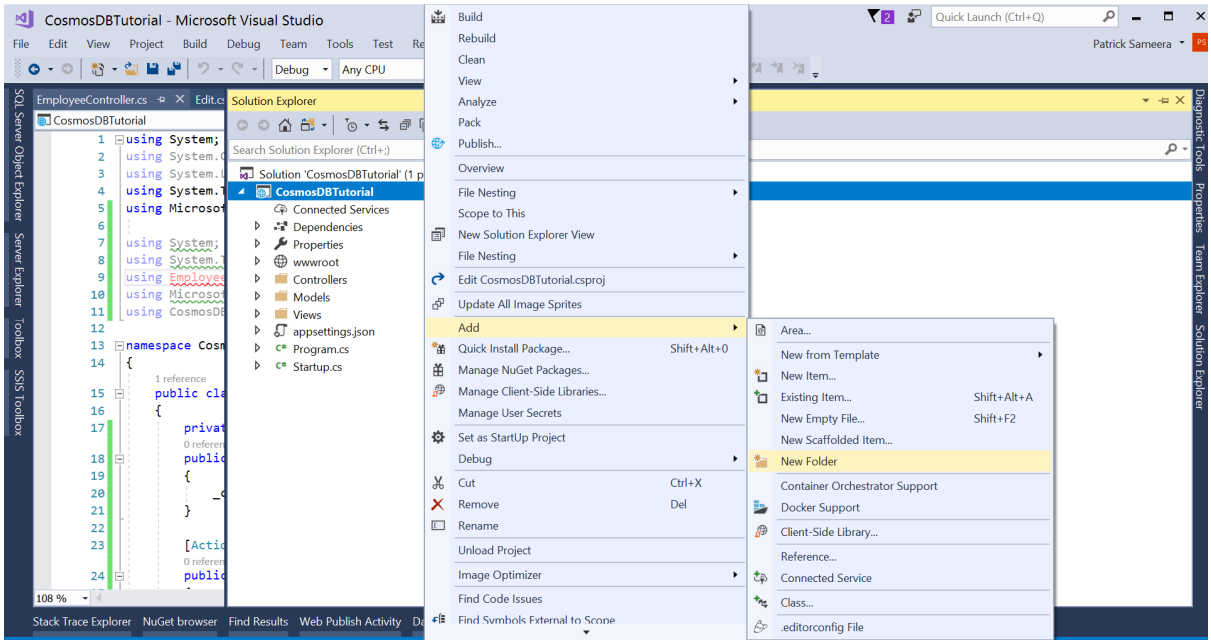
```
75          [ActionName("Delete")]
76          public async Task<ActionResult> DeleteAsync(string id)
77          {
78              if (id == null)
79              {
80                  return BadRequest();
81              }
82
83              Employee employee = await _cosmosDbService.GetItemAsync(id);
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.

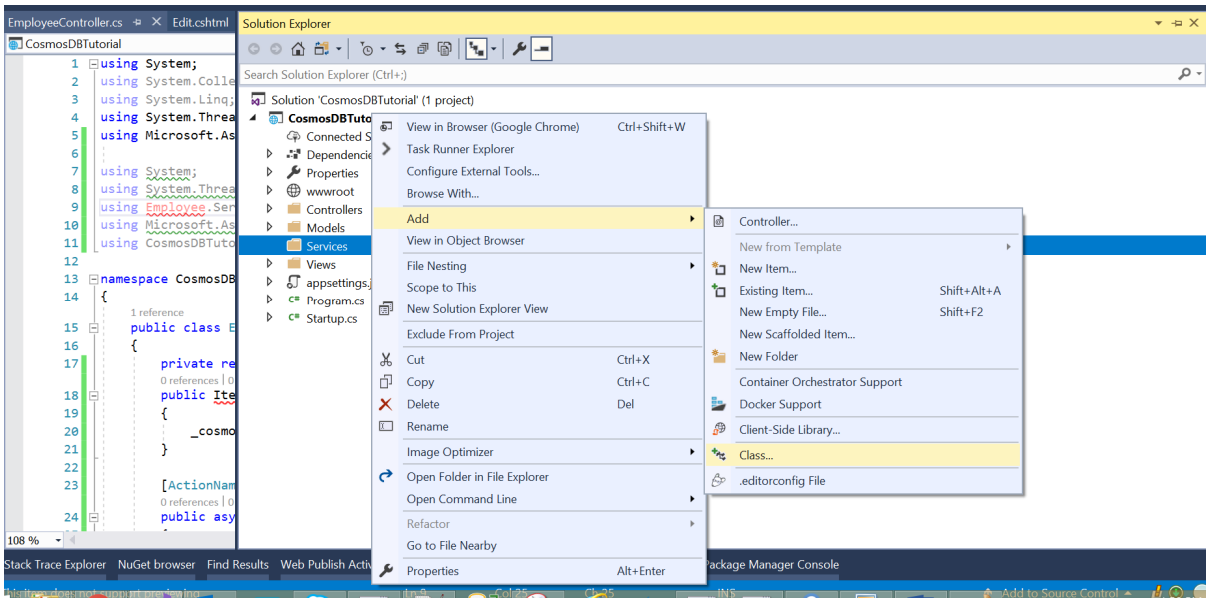
First Add a new Folder called Services. Right Click on the Solution.

- Add → New Folder



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

- Services → Add → Class

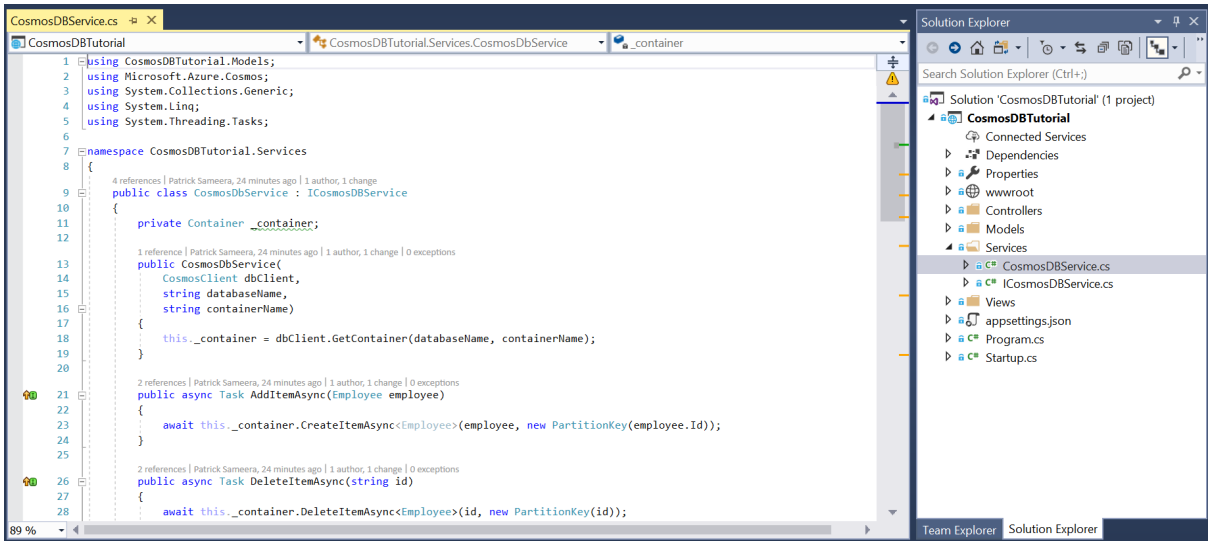


Patrick Sameera Jayalath

CosmosDBService.cs

Copy paste the code to the file.

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



Patrick Sameera Jayalath

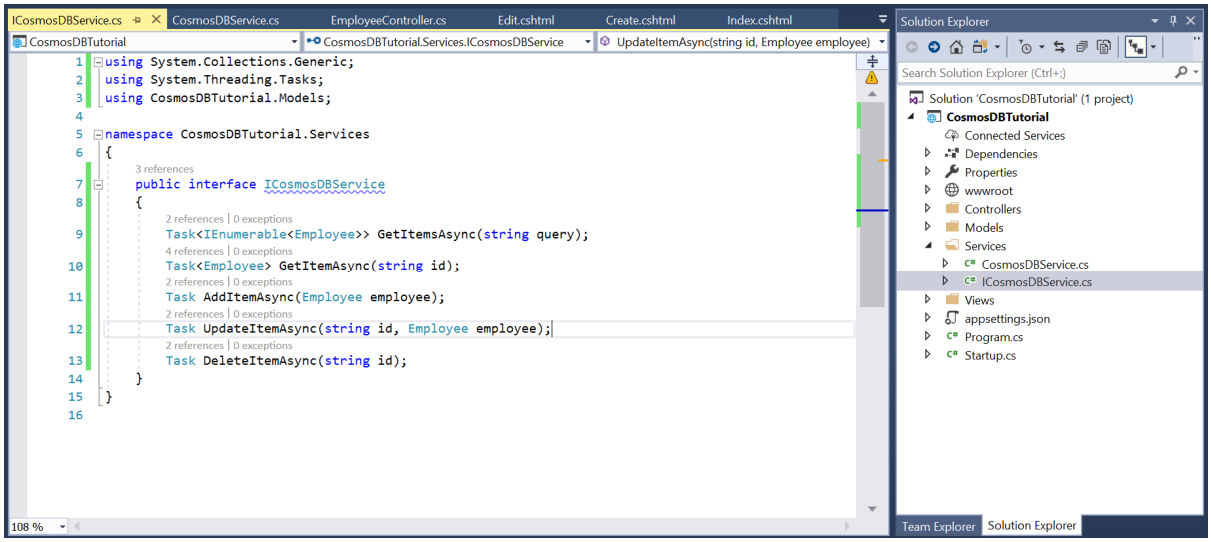
Next under Services Folder add new Class by the name ICosmosDBService.cs

- Services → Add → Class

ICosmosDBService.cs

Copy paste the code to the file.

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



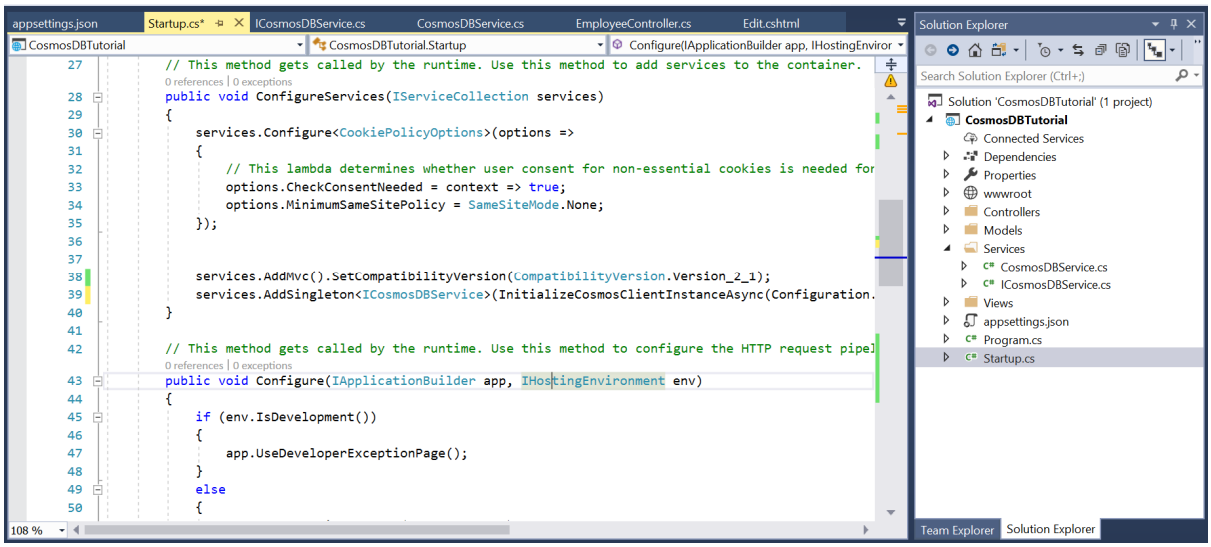
Startup.cs

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

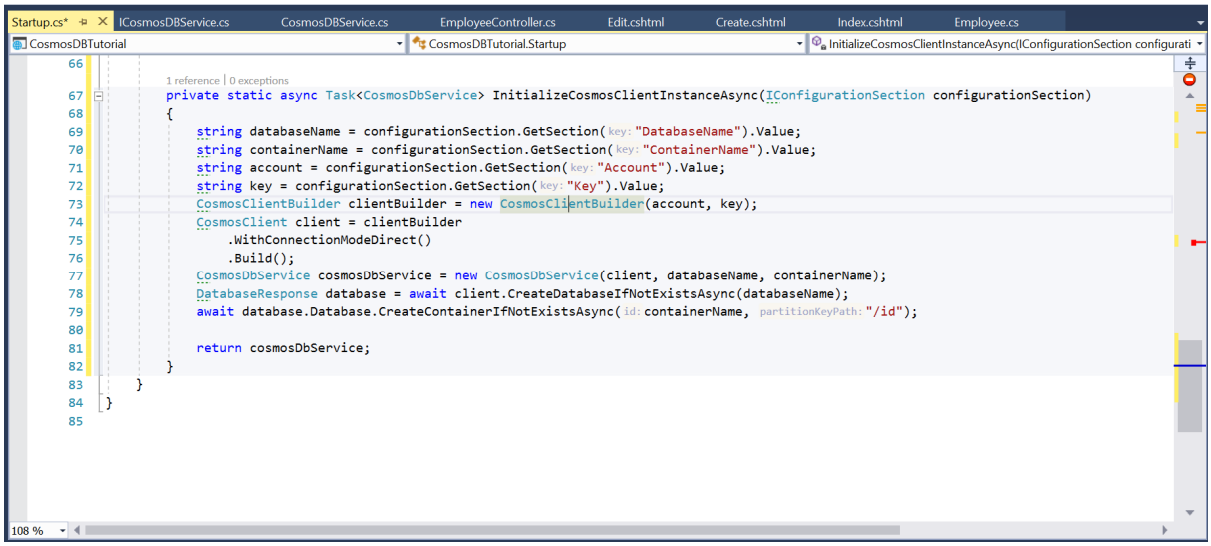
services.AddSingleton<ICosmosDbService>(InitializeCosmosClientInstanceAsync(Configuration.GetSection("CosmosDb")).GetAwaiter().GetResult());

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.

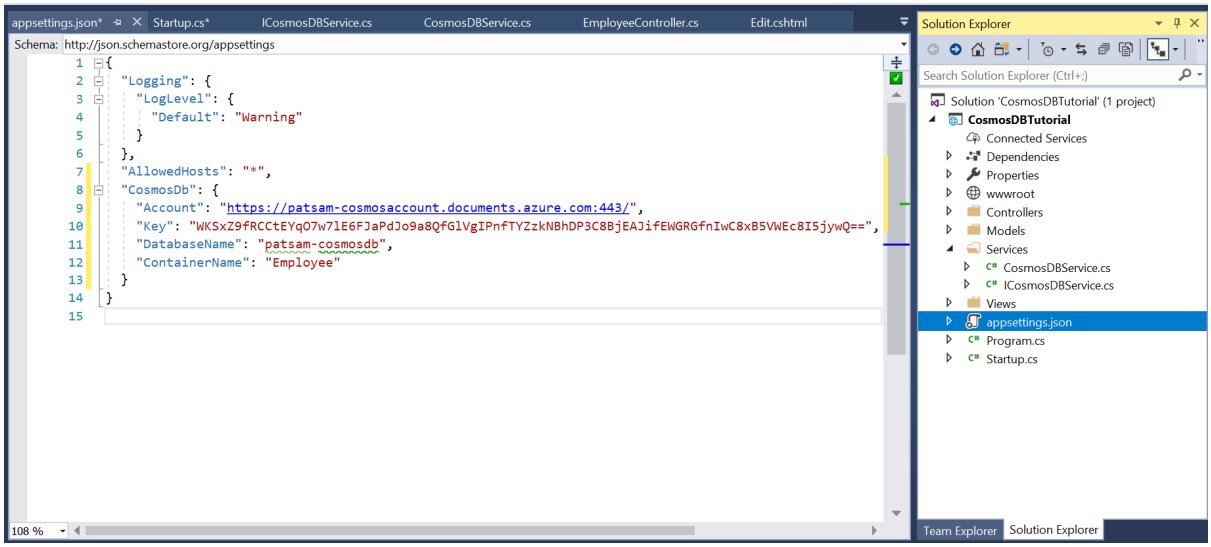


appsettings.json

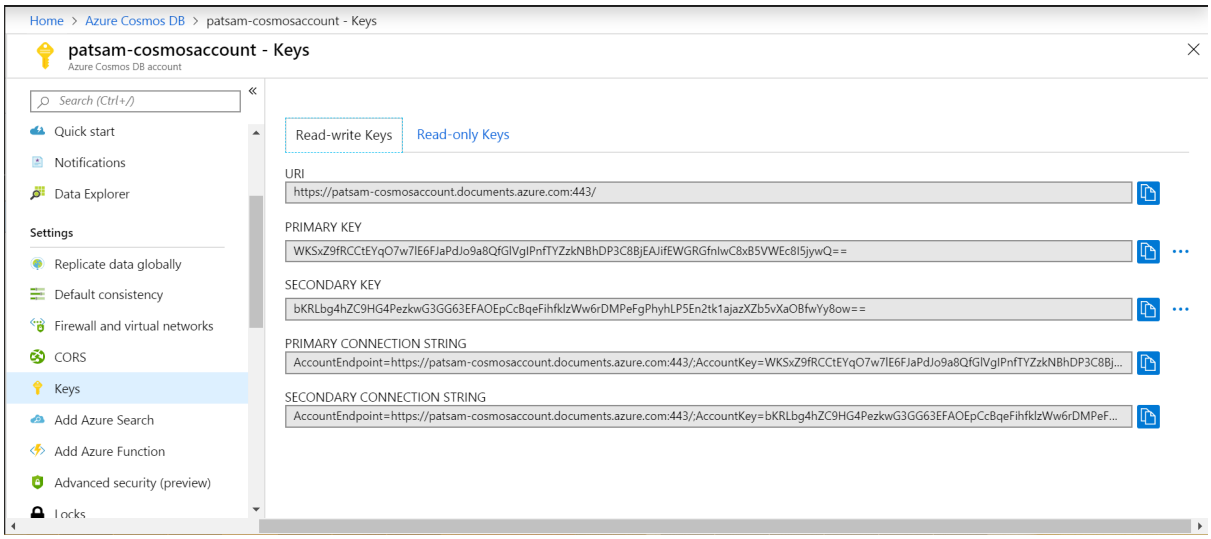
Define the configuration in the project's appsettings.json file. Open the file and add a section called CosmosDb.

- Account: https://patsam-cosmosaccount.documents.azure.com:443/
- Key: WKSxZ9fRCCtEYqO7w7lE6FJaPdJo9a8QfGlVgIPnfTYZzkNBhDP3C8BjEAJifEWGRGfnIwC8xB5VWEc8I5jywQ==
- Database Name: patsam-cosmosdb
- Container Name: Employee

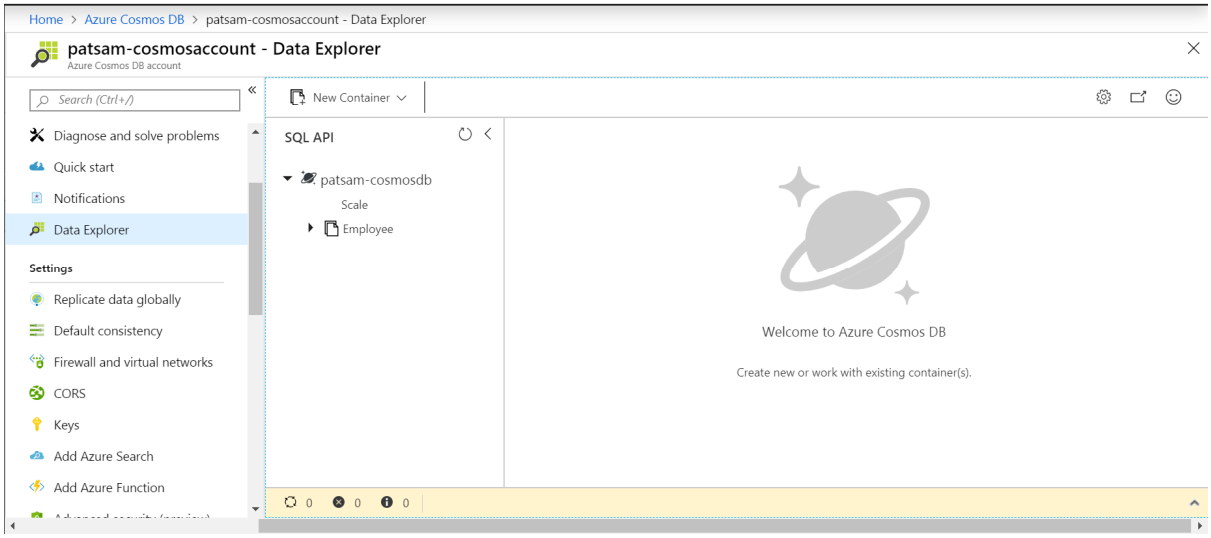
https://github.com/patricksameerajayalath/CosmosDBTutorial/blob/master/appsettings.json



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



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



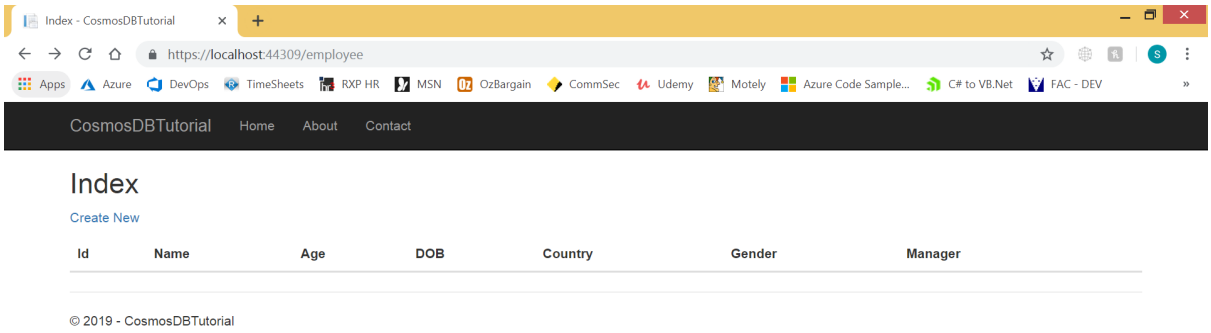
Patrick Sameera Jayalath

Step 07: CRUD Operations

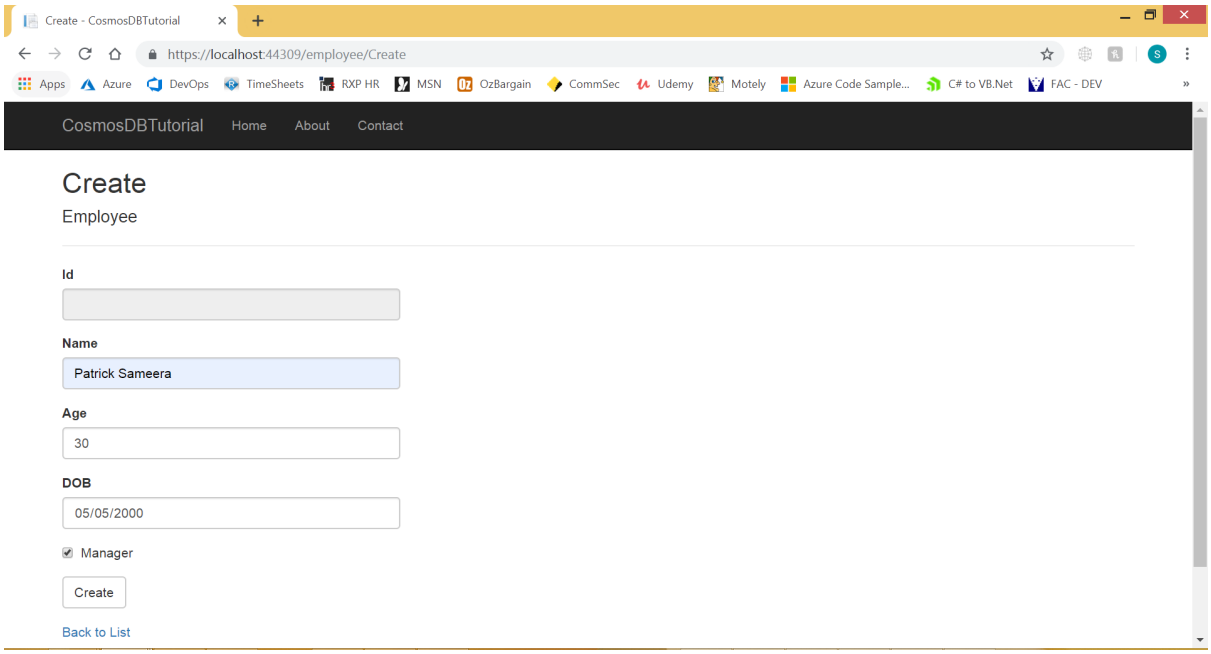
Now we have finished putting up the code. Run the Project F5.

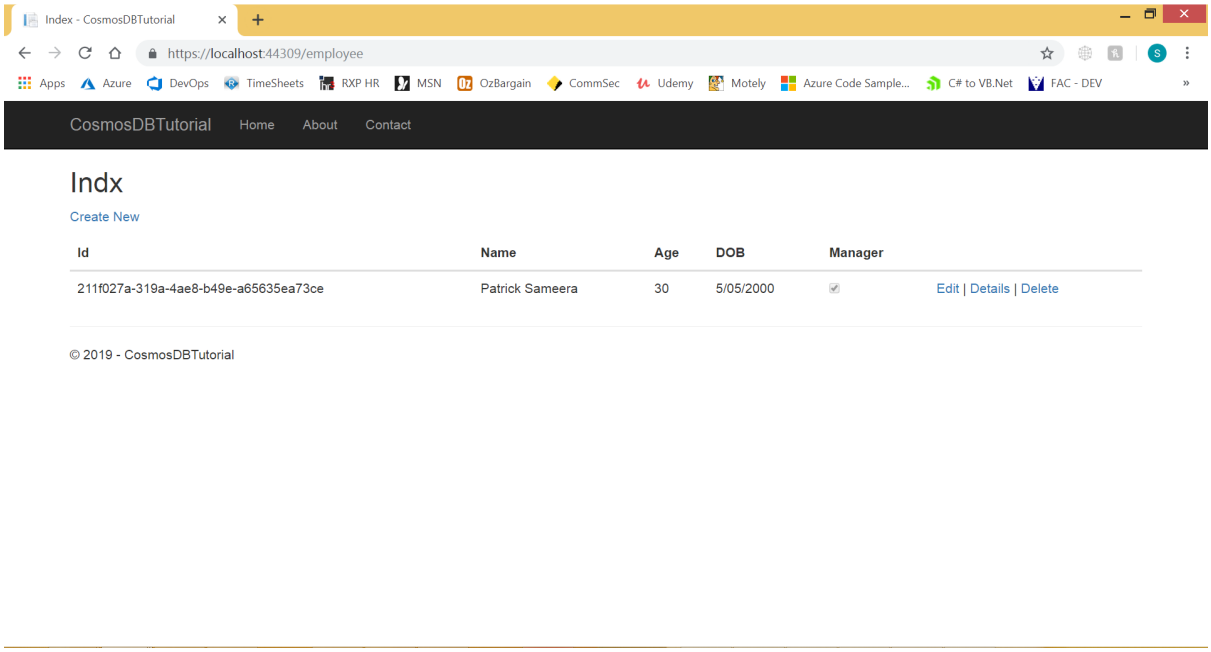
https://localhost:44309/employee

Employee list functionality:

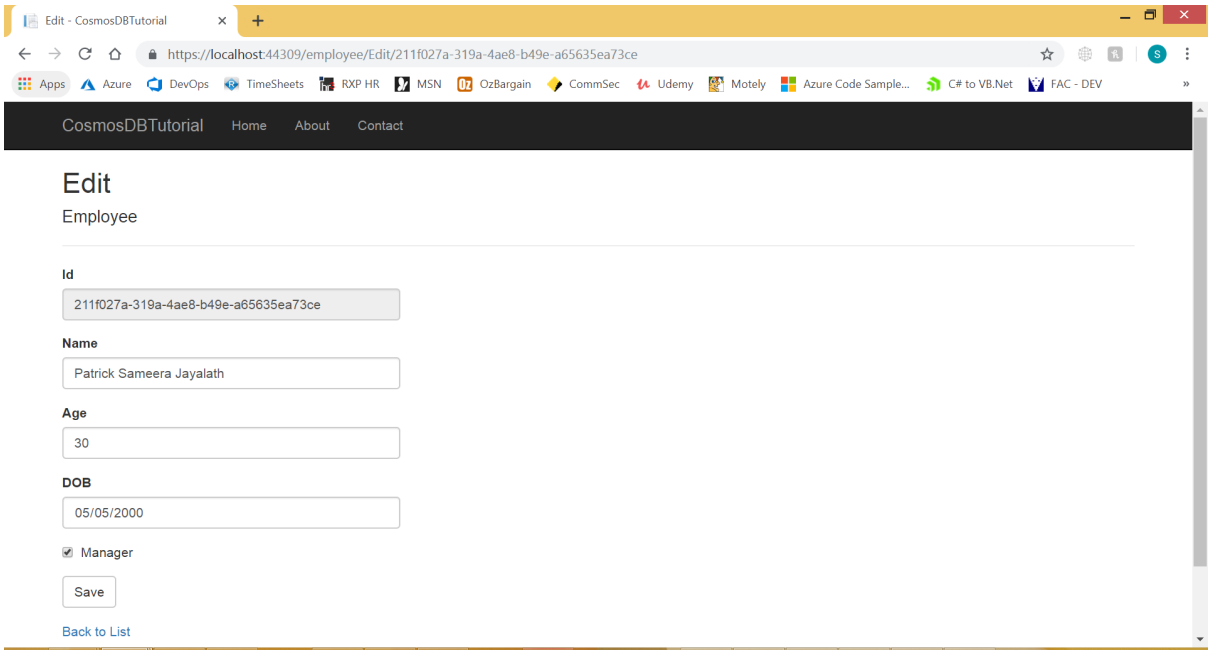


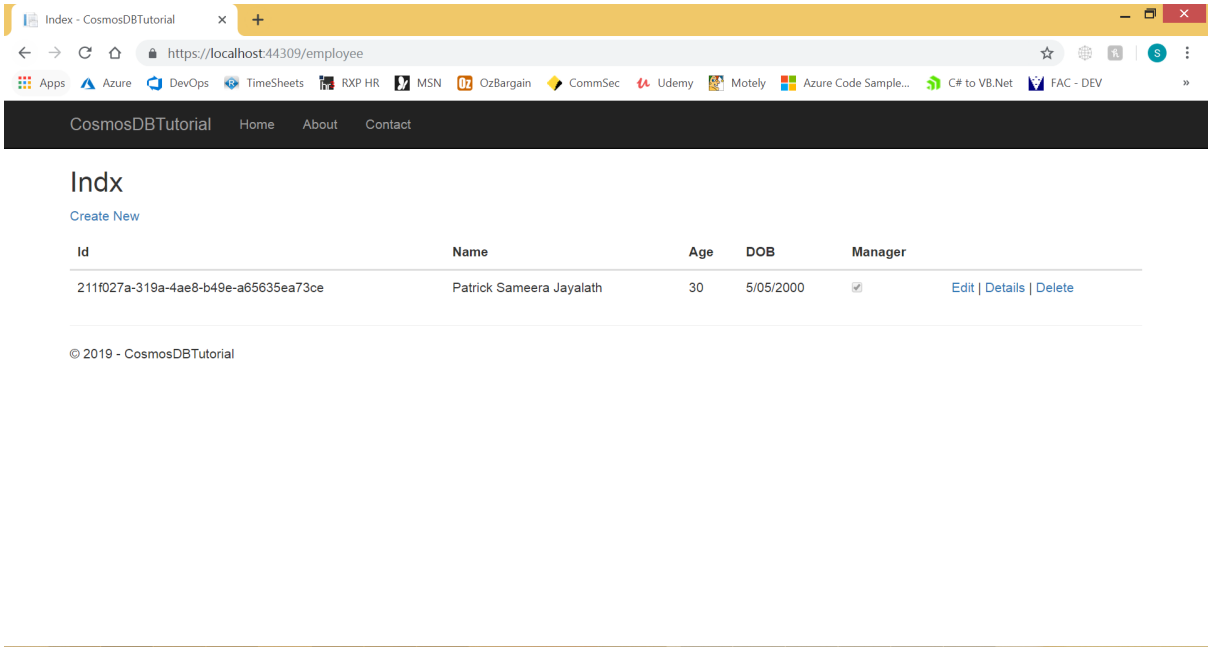
Create functionality:



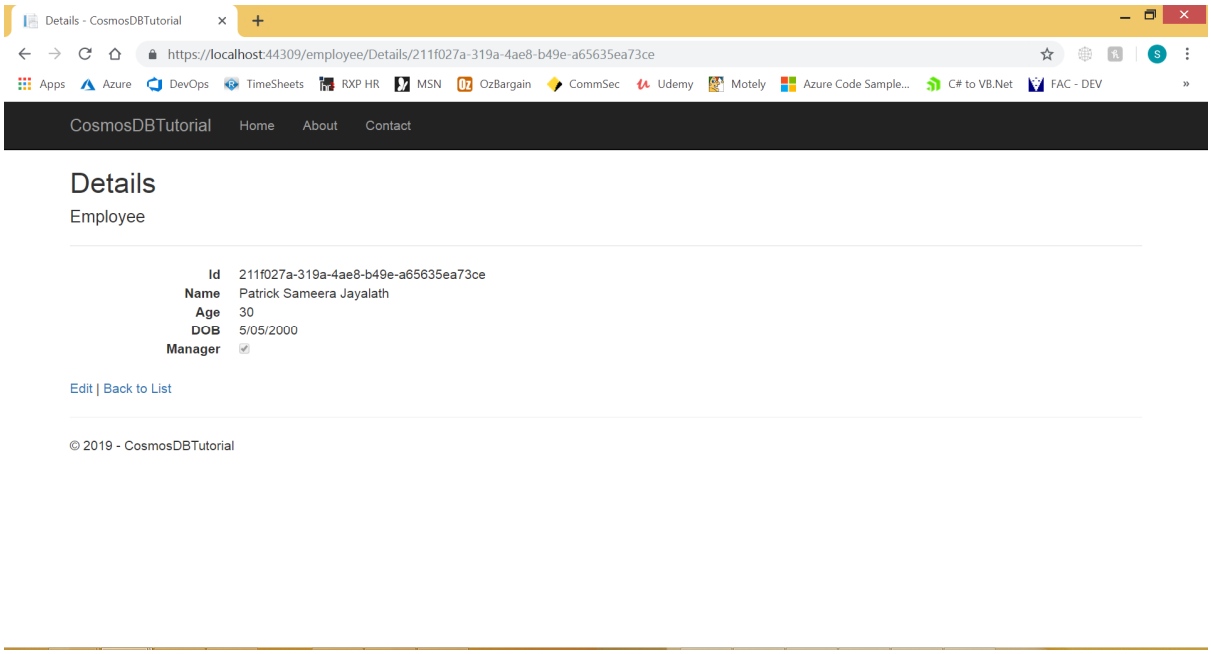


Edit functionality:

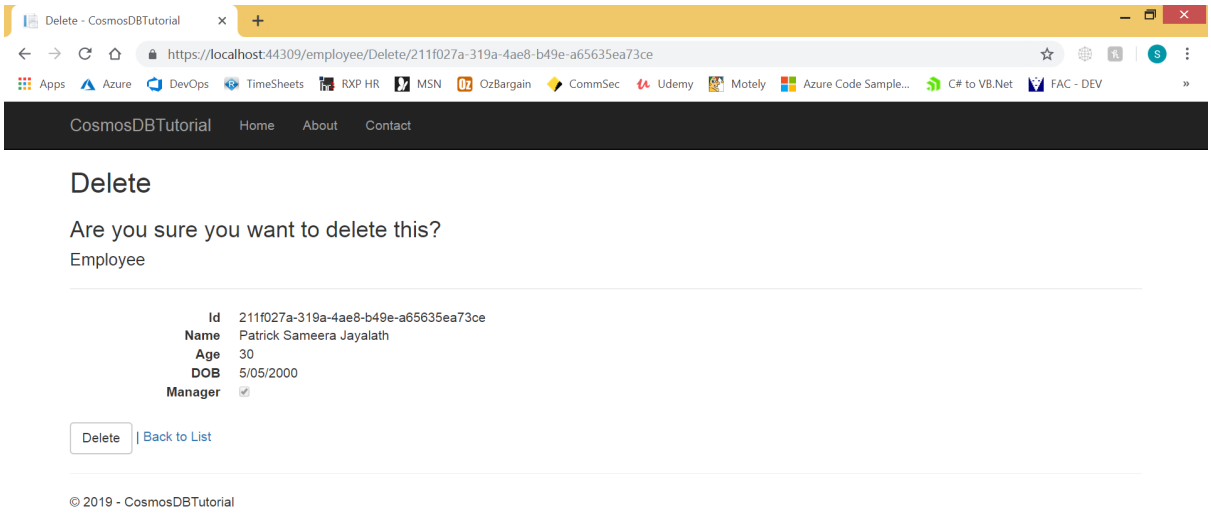




Details functionality:



Delete functionality:



We can inspect the data through Data Explorer tab.