

dotnet new console -o NetbcCosmos

cd NetbcCosmos

01 Cosmos\_package.txt

dotnet add package Microsoft.EntityFrameworkCore.Cosmos

Models/Course.cs

public class Course {

02\_Course.cs.txt

public Guid Id { get; set; }

public string Number { get; set; }

public string Description { get; set; }

public Room Room { get; set; }

}

public class Room {

public string Building { get; set; }

public int RoomNumber { get; set; }

}

03 SchoolContext.txt

Data/SchoolContext.cs

public class SchoolContext : DbContext {

public DbSet<Course> Courses { get; set; }

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) {

optionsBuilder.UseCosmos(

"\*YOUR-COSMOSDB-ENPOINT\*",

"\*YOUR-KEY\*",

"EFCoreTest"

);

}

protected override void OnModelCreating(ModelBuilder modelBuilder) {

modelBuilder.Entity<Course>().OwnsOne(j => j.Room);

}

}

Add the following helper methods to *Program.cs*:

04 Program.cs.txt

static void InsertSimpleCourse() {

var course = new Course {

Id = Guid.NewGuid(),

Room = new Room {

Building = "SE6",

RoomNumber = 127

}

};

using (var context = new SchoolContext()) {

context.Database.EnsureCreated();

context.Add(course);

context.SaveChanges();

}

}

static void dispCourses() {

using (var context = new SchoolContext()) {

var courses = context.Courses.ToList();

foreach (var course in courses) {

displayCourseInfo(course);

Console.WriteLine("===========================================");

}

}

}

static void displayCourseInfo(Course course) {

Console.WriteLine($"{course.Id}");

Console.WriteLine($"Room: {course.Room.Building}, {course.Room.RoomNumber}");

}

Let this be the only content in the *Main()* method in *Program.cs*:

InsertSimpleCourse();

dispCourses();

Run the app. The output should look like this:

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)Full Stack Web Development

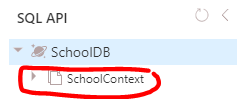
ID: 22619fdf-1132-446b-b290-2fac1e2a09dc

NUMBER: COMP4870

ROOM: SE12, 320

===========================================

The container name defaults to the context name.

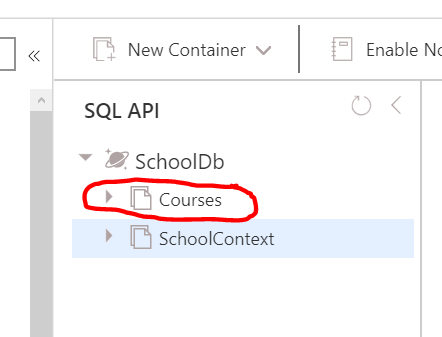


Add the following code in OnMedelCreating method of the context class so that *Courses* is the default container name:

05 OnModelCreating.txt

modelBuilder.HasDefaultContainer("Courses");

If you run the application again, you will notice that a new container named *Courses* gets created.



Data on Azure CosmosDB looks like this:

{

"Id": "22619fdf-1132-446b-b290-2fac1e2a09dc",

"Description": "Full Stack Web Development",

"Discriminator": "Course",

"Number": "COMP4870",

"id": "Course|22619fdf-1132-446b-b290-2fac1e2a09dc",

"Room": {

"Building": "SE12",

"RoomNumber": 320

},

"Students": null,

"\_rid": "ZYA1ALJGgJkBAAAAAAAAAA==",

"\_self": "dbs/ZYA1AA==/colls/ZYA1ALJGgJk=/docs/ZYA1ALJGgJkBAAAAAAAAAA==/",

"\_etag": "\"00009100-0000-0800-0000-5e3a46220000\"",

"\_attachments": "attachments/",

"\_ts": 1580877346

}

## Update

static void updateOneCourse(string id) {

Guid g = new Guid(id);

06 updateOneCourse.txt

using (var context = new SchoolContext()) {

var course = context.Courses.Find(g);

if (course != null) {

course.Room.Building = "NE1";

course.Room.RoomNumber = 123;

context.SaveChanges();

Console.WriteLine($"Course with ID={id} has been UPDATED");

}

}

}

07 deleteOneCourse.txt

## Delete

static void deleteOneCourse(string id) {

Guid g = new Guid(id);

using (var context = new SchoolContext()) {

var course = context.Courses.Find(g);

if (course != null) {

context.Courses.Remove(course);

context.SaveChanges();

Console.WriteLine($"Course with ID={id} has been DELETED");

}

}

}

## Embedded collections

Let us add a list of *Student* objects as part of the *Course* class. Add the following *Student* class to *Course.cs*:

08 Student.txt

public class Student {

public string StudentId { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Gender { get; set; }

}

Add this property to the *Course* class:

09 to\_Course.txt

public List<Student> Students { get; set; }

Update the *OnModelCreating* method in the context class to configure the *Students* on *Course* as an owned collection

10 OnModelCreating.txt

modelBuilder.Entity<Course>().OwnsMany(j => j.Students);

Add another method named *InsertCourseWithStudents()* in *Program.cs* that inserts a course + students as shown below:

11 InsertCourseWithStudents.txt

static Guid InsertCourseWithStudents() {

var course = new Course {

Id = Guid.NewGuid(),

Number = "COMP3717",

Description = "Android Development",

Room = new Room {

Building = "SE12",

RoomNumber = 322

},

Students = new List<Student>() {

new Student {

StudentId = "A00111111",

FirstName = "Bob",

LastName = "Fox",

Gender = "Male"

},

new Student {

StudentId = "A00222222",

FirstName = "Ann",

LastName = "Fay",

Gender = "Female"

}

}

};

using (var context = new SchoolContext()) {

context.Database.EnsureCreated();

context.Add(course);

context.SaveChanges();

}

return course.Id;

}

12 to\_displayCourseInfo.txt

Update *displayCourseInfo()* in *Program.cs*. Append this code to the method:

if (course.Students == null || course.Students.Count == 0) {

Console.WriteLine(" NO STUDENTS");

} else {

Console.WriteLine($" Students: {course.Students.Count()}");

course.Students.ForEach(x => {

Console.WriteLine($" ID: {x.StudentId} Name: {x.FirstName} {x.LastName}, {x.Gender}");

});

}

In the *Main()* method, comment out the following:

deleteOneCourse();

Underneath the above commented code, add this statement:

Guid g = InsertCourseWithStudents();

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)Run the application. The output should look like this:

Android Development

ID: b8d9a7a7-a150-4172-bc55-63d5fcb94640

NUMBER: COMP3717

ROOM: SE12, 322

Students: 2

ID: A00111111 Name: Bob Fox, Male

ID: A00222222 Name: Ann Fay, Female

===========================================

13 dispOneCourse.txt

Add a new method named *dispOneCourse()* that displays Course from Guid:

static void dispOneCourse(string id) {

Guid g = new Guid(id);

using (var context = new SchoolContext()) {

var course = context.Courses.Find(g);

if (course != null) {

displayCourseInfo(course);

Console.WriteLine("===========================================");

}

}

}

Also, in the *Main()* method, comment out this code:

dispCourses();

Underneath the above code that was just commented out, add this statement to display only one course:

dispOneCourse(g.ToString());

The output will look like the same as before:

Data in Azure CosmosDB looks like this:

{

"Id": "6f097e43-5fd8-411f-afa2-1cc2591cb26b",

"Description": "Android Development",

"Discriminator": "Course",

"Number": "COMP3717",

"id": "Course|6f097e43-5fd8-411f-afa2-1cc2591cb26b",

"Room": {

"Building": "SE12",

"RoomNumber": 322

},

"Students": [

{

"StudentId": "A00111111",

"FirstName": "Bob",

"Gender": "Male",

"LastName": "Fox"

},

{

"StudentId": "A00222222",

"FirstName": "Ann",

"Gender": "Female",

"LastName": "Fay"

}

],

"\_rid": "ZYA1AP3uFBcDAAAAAAAAAA==",

"\_self": "dbs/ZYA1AA==/colls/ZYA1AP3uFBc=/docs/ZYA1AP3uFBcDAAAAAAAAAA==/",

"\_etag": "\"0000eb00-0000-0800-0000-5e3a49540000\"",

"\_attachments": "attachments/",

"\_ts": 1580878164

}

Notice the discriminator property represents the C# class. The id property with a lowercase I is a combination of the class name and the Id of the document.

## Linked Entities / Documents

Let us create a new entity called *Instructor* and add it to the *Course* class. Append the following *Instructor* class definition to *Course.cs*:

14 Instructor.txt

public class Instructor {

public Guid Id { get; set; }

public string Title { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

}

Add these properties to the *Course* class definition in *Course.cs*:

15 to\_Course.txt

public Guid AssignedInstructorId { get; set; }

public Instructor AssignedInstructor { get; set; }

Update *SchoolContext* to configure the *Instructor* as a linked entity by appending this code to the *OnModelCreating()* method:

16 OnModelCreating.txt

modelBuilder.Entity<Course>().HasOne(j => j.AssignedInstructor);

Add this *InsertCourseWithStudentsAndLinkedInstructor()* method to *Program.cs* that adds a Course with an instructor:

static Guid InsertCourseWithStudentsAndLinkedInstructor() {

var course = new Course {

Id = Guid.NewGuid(),

Number = "COMP2914",

Description = "Database Design",

Room = new Room {

Building = "SE12",

RoomNumber = 308

},

17 InsertCourseWithStudentsAndLinkedInstructor.txt

Students = new List<Student>() {

new Student {

StudentId = "A00333333",

FirstName = "Tom",

LastName = "Dow",

Gender = "Male"

},

new Student {

StudentId = "A00444444",

FirstName = "Sue",

LastName = "Lee",

Gender = "Female"

}

},

AssignedInstructor = new Instructor {

Id = Guid.NewGuid(),

Title = "Mr",

FirstName = "Tim",

LastName = "Coe"

}

};

using (var context = new SchoolContext()) {

context.Database.EnsureCreated();

context.Add(course);

context.SaveChanges();

}

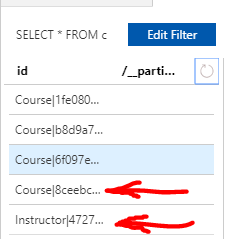
return course.Id;

}

Comment out all the statements in the *Main()* method and only call the following method:

Guid g2 = InsertCourseWithStudentsAndLinkedInstructor();

Run the application then go to Azure Cosmos DB. You will notice that two documents have been created. One for *Course* and the other for *Instructor*.



In the *Course* document you will find a property *AssignedInstructorId* that points to the *Id* of the *Instructor*.



How do we display instructors for a *Course*. First we need to add the following to *SchoolContext.cs*:

18 to\_SchoolContext.txt

public DbSet<Instructor> Instructors { get; set; }

19 to\_displayCourseInfo.txt

Add the following to the bottom of the *displayCourseInfo()* method:

using (var context = new SchoolContext()) {

var instructor = context.Instructors.First(x => x.Id == course.AssignedInstructorId);

Console.WriteLine($"Assigned Instructor: {instructor?.FirstName} {instructor?.LastName}");

}

Now, grab the Id of the last course that was created with an instructor then make a request to display the data. In the *Main()* method, comment out all the statements then add a statement as follows:

dispOneCourse("1ac377e8-e09f-4c42-be28-3126d3b41998");

When you run the application the output will look like this:

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)Database Design

ID: 8ceebc54-af20-4bb6-82bc-ba9844ee5dcc

NUMBER: COMP2914

ROOM: SE12, 308

Students: 2

ID: A00333333 Name: Tom Dow, Male

ID: A00444444 Name: Sue Lee, Female

Assigned Instructor: Tim Coe

===========================================

## Multiple courses linked to the same instructor

How about if there are multiple courses linked to the same instructor? Let us:

* Create an Instructor entity
* Create 2 Course entities
* Assign the Instructor to the 2 courses
* Save to the database
* Create a new DbContext instance (this will ensure we have a clean DbContext with no tracked entities)
* Load the Instructor
* Load the 2 Course entities
* Check to see if the AssignedInstructor has automatically been set by EF Core using the pre-fetched Instructor

Add a new method *InsertInstructorWithLinkedCourses()* to *Program.cs*:

static Guid InsertInstructorWithLinkedCourses() {

var instructorId = Guid.NewGuid();

var instructor = new Instructor {

Id = instructorId,

Title = "Mr",

FirstName = "Joe",

20 InsertInstructorWithLinkedCourses.txt

LastName = "Doe"

};

var course1 = new Course {

Id = Guid.NewGuid(),

Number = "COMP3974",

Description = "Data Communications",

Room = new Room {

Building = "SE12",

RoomNumber = 320

},

AssignedInstructor = instructor

};

var course2 = new Course {

Id = Guid.NewGuid(),

Number = "COMP2954",

Description = "HTML",

Room = new Room {

Building = "SE12",

RoomNumber = 319

},

AssignedInstructor = instructor

};

using (var context = new SchoolContext()) {

context.Database.EnsureCreated();

context.Add(course1);

context.Add(course2);

context.SaveChanges();

}

return instructorId;

}

Comment out all the code in the Main() method and add this statement:

Guid g3 = InsertInstructorWithLinkedCourses();

Run the app to create an instructor and two courses associated to that instructor:

21 dispOneInstructor.txt

Let us now add a method that displays the instructor and two courses. Add method *dispOneInstructor()* to *Program.cs*:

static void dispOneInstructor(string id) {

Guid g = new Guid(id);

using (var context = new SchoolContext()) {

var instructor = context.Instructors.Find(g);

if (instructor != null) {

Console.WriteLine($"Instructor ID: {instructor.Id}");

Console.WriteLine("===========================================");

// Load all courses with the same assigned instructor id

var jobs = context.Courses.Where(x => x.AssignedInstructorId == g).ToList();

foreach (var job in jobs) {

displayCourseInfo(job);

Console.WriteLine("===========================================");

}

}

}

}

Grab the last *InstructorId* from Azure Cosmos DB. Use it to display the last instructor and courses by commenting everything in *Main()* and adding only this statement:

dispOneInstructor("fd4ad351-be2a-401a-9a1e-4ee7b03d79f4");

Run your app and you should experience output similar to this:

Instructor ID: 5dbe456e-dca3-4067-90c2-0a8b57aaf074

===========================================

[](https://www.bing.com/images/search?view=detailV2&ccid=g6nYHgjm&id=9DCD800E35FF6162FE4C44ABC6C70A5783B46357&thid=OIP.g6nYHgjml5I7LyjheiVEGwElEs&q=run+logo&simid=608050058236200411&selectedIndex=67)Data Communications

ID: d314d852-897c-4c18-a62e-0631658ce1d7

NUMBER: COMP3974

ROOM: SE12, 320

NO STUDENTS

Assigned Instructor: Joe Doe

===========================================

HTML

ID: 357778f8-0f80-42b0-92a5-82156d22fd3d

NUMBER: COMP2954

ROOM: SE12, 319

NO STUDENTS

Assigned Instructor: Joe Doe

===========================================