What’s New In Entity Framework 8

Barret Blake

Titanium Sponsors

Platinum Sponsors


Temporal


Pulumi

Gold Sponsors

Speaker Dinner

Friends of KCDC

About Me

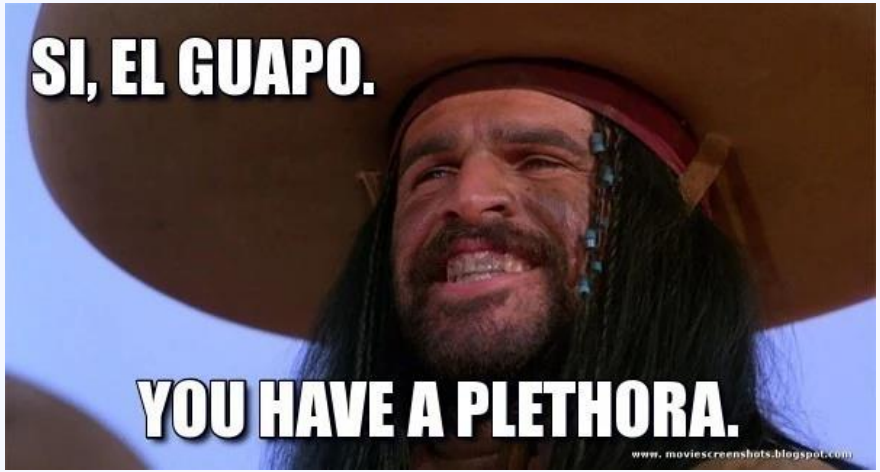
Microsoft MVP

Business Apps
Power Automate

Husband & Father
Azure Solutions Architect
Gamer
Speaker & Blogger
Mentor & Teacher
Model Railroader

barretblake.dev
youtube.com/@barretcodes
twitter.com/barretblake
linkedin.com/in/barretblake

“Would you say that I have a plethora of new features in Entity Framework Core 8?”



--Three Amigos, sort of

Agenda

01

Complex Types

Compound classes made simple

02

Primitive Collections

Storing simple lists as JSON

03

JSON

Vastly easier to work with and query

04

Other Updates

So, so, so many other updates

01

Complex Types

Complex Types

```
public class Customer {
    public Guid Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public Address Address { get; set; }
}

public Class Address {
    public string Street { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string Zip { get; set; }
}
```

Complex Types

```
public class Customer
{
    public Guid Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public Address Address { get; set; }
}

[ComplexType]
public class Address
{
    public string Street { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string Zip { get; set; }
}
```

Complex Types

Column Name	Data Type	Allow Nulls
Id	uniqueidentifier	☐
FirstName	nvarchar(MAX)	☐
LastName	nvarchar(MAX)	☐
Address_City	nvarchar(MAX)	☐
Address_State	nvarchar(MAX)	☐
Address_Street	nvarchar(MAX)	☐
Address_Zip	nvarchar(MAX)	☐
		☐

Complex Types

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Customer>()
        .ComplexProperty(e => e.Address);
}
```

Complex Types

```
[ComplexType]
public class Address
{
    public string Street { get; set; }
    public string City { get; set; }
    public string State { get; set; }
    public string Zip { get; set; }
    public GeoCoordinates Coordinates { get; set; }
}

[ComplexType]
public class GeoCoordinates
{
    public double Latitude { get; set; }
    public double Longitude { get; set; }
}
```

Complex Types

	Column Name	Data Type	Allow Nulls
	Id	uniqueidentifier	☐
	FirstName	nvarchar(MAX)	☐
	LastName	nvarchar(MAX)	☐
	Address_City	nvarchar(MAX)	☐
	Address_State	nvarchar(MAX)	☐
	Address_Street	nvarchar(MAX)	☐
	Address_Zip	nvarchar(MAX)	☐
	Address_Coordinates_Latitude	float	☐
	Address_Coordinates_Longitude	float	☐

Complex Types - Querying

```
var nw = new GeoCoordinates(48.19190, -83.20889);
var sw = new GeoCoordinates(39.83429, -82.83330);

var results = await _context.Customers
    .Where(c => c.Address.Coordinates.Latitude < nw.Latitude
            && c.Address.Coordinates.Latitude > sw.Latitude
            && c.Address.Coordinates.Longitude > nw.Longitude
            && c.Address.Coordinates.Longitude < sw.Longitude)
    .ToListAsync();
```

Complex Types - Mutability

```
var address = new Address() { City = "Columbus", State = "OH", Street = "123 Main St", Zip =
"43215" };
var cust1 = new Customer() { FirstName = "John", LastName = "Doe", Address = address };
var cust2 = new Customer() { FirstName = "Jane", LastName = "Smith", Address = address };
var cust3 = new Customer() { FirstName = "Bob", LastName = "Jones", Address = address };

_context.Customers.Add(cust1);
_context.Customers.Add(cust2);
_context.Customers.Add(cust3);
_context.SaveChanges();

cust1.Address.City = "Dublin";
_context.SaveChanges();
```

Complex Types – Projections & Predicates

```
var address = await _context.Customers
    .Where(e => e.Address.City == "Kansas City")
    .Select(e => e.Address)
    .ToListAsync();
```

Complex Types - Limitations

- Collections of complex types
- Allow complex type properties to be null
- Mapping to JSON columns
- Constructor injection
- Seed data support
- Cosmos DB support
- In-memory DB support

02

Primitive Collections

Primitive Collections

```
public class Customer
{
    public Guid Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    public Address Address { get; set; }
    public List<Guid> WishListProducts { get; set; }
}
```

Primitive Collections

	Column Name	Data Type	Allow Nulls
	Id	uniqueidentifier	☐
	FirstName	nvarchar(MAX)	☐
	LastName	nvarchar(MAX)	☐
▶	WishListProducts	nvarchar(MAX)	☐
	Address_City	nvarchar(MAX)	☐
	Address_State	nvarchar(MAX)	☐
	Address_Street	nvarchar(MAX)	☐
	Address_Zip	nvarchar(MAX)	☐
	Address_Coordinates_Latitude	float	☐
	Address_Coordinates_Longitude	float	☐

```
[
 "ad39b7c2-9011-41b7-b2ca-716e627527d1",
 "97137edf-b7f4-4db3-b9ae-76a51c4a3b48",
 "78ab4609-b58c-4107-8b5c-c2b0fecc39df",
 "00b3c576-dd34-4905-ba99-f37cef1f327d"
]
```

Primitive Collections

```
Guid productId = new Guid("f5b3f4b3-3b4b-4b4b-8b4b-4b4b4b4b4b4b");
var customersWhoWantItem = await _context.Customers
    .Where(x => x.WishListProducts.Contains(productId))
    .ToListAsync();
```

Primitive Collections – Array Parameters

```
var firstNames = new[] { "John", "Jane", "Jim", "Jill" };
var customers = await _context.Customers
    .Where(c => firstNames.Contains(c.FirstName))
    .ToListAsync();
```

Primitive Collections – Array Parameters

```sql
SELECT [c].[ID], [c].[FirstName], [c].[LastName]
FROM [Customers] AS [c]
WHERE [c].[FirstName] IN ('John', 'Jane', 'Jim', 'Jill')
```

```sql
SELECT [c].[ID], [c].[FirstName], [c].[LastName]
FROM [Customers] AS [c]
WHERE EXISTS (
    SELECT 1
    FROM OpenJson(@__names_0) AS [n]
    WHERE [n].[value] = [c].[FirstName])
)
```

03

JSON

JSON

```
public class CustomerWithJson
{
    public Guid Id { get; set; }
    public string FirstName { get; set; }
    public string LastName { get; set; }
    [Required]
    public List<Address> Addresses { get; set; }
    public List<Guid> WishListProducts { get; set; }
}
```

JSON

```
var custWithAddresses = await _context.CustomerWithJson
    .Where(c⇒ c.Addresses.Any(a ⇒ a.City == "Kansas City"))
    .Include(c ⇒ c.Addresses)
    .ToListAsync();
```

JSON

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<CustomerWithJson>()
        .OwnsMany(cj => cj.Addresses)
        .ToJson();
}
```

JSON

```
var custWithAddresses2 = await _context.CustomerWithJson
    .Where(c => c.Addresses.Any(a => a.City == "Kansas City"))
    .ToListAsync();
```

Other JSON Enhancements

JSON Columns

Improved support for querying into SQL Server JSON column metadata

Embedded Collections

Support for querying into collections (primitive and non-primitive) embedded in JSON Columns

SQLite

Full support for JSON columns in SQLite databases

04

Other Updates

Database Defaults & Sentinel Values

```
modelBuilder.Entity<Customer>()
    .Property(e⇒e.FirstName)
    .HasDefaultValue("John");
```

```
modelBuilder.Entity<Customer>()
    .Property(e⇒e.FirstName)
    .HasDefaultValue("John")
    .HasSentinel("N/A");
```

```
modelBuilder.Entity<Customer>()
    .Property(e⇒e.CreatedOn)
    .HasDefaultValueSql("GETUTCDATE()");
```

Other Enhancements

HierarchyId

Unmapped Types

Lazy Loading

Query Tracked Entities

DateOnly & TimeOnly

SQLite

Breaking Changes

Contains

May no longer work in older versions of SQL Server

Enums

Enums in JSON data are now stored as ints, not strings

SQL Server DATE & TIME

DATE & TIME now map to DateOnly & TimeOnly in .NET

Bool

Non-nullable bool columns in DB that have a default value now scaffold to non null bool in .NET

Preview of EF Core 9

- Cosmos DB
- More updates to primitive collections and complex types
- AOT and pre-compiled queries
- Improvements to Linq translations
- ExecuteUpdate/ExecuteDelete
- Temporal table migrations improvements
- Automatically build and use pre-compiled models
- Read-only primitive collections
- Setting caching options for db sequences
- Specify fill factor for indexes and keys
- Improving extensibility of modelbuilder

Thanks!

Do you have any questions?

barretblake@live.com
https://barretblake.dev
https://linkedin.com/in/barretblake
https://youtube.com/@barretcodes
@barretblake

Speaker Feedback

Barret Blake
What's New in Entity Framework 8
KCDC 2024