

# Adding the First EF Core DbContext



**Julie Lerman**

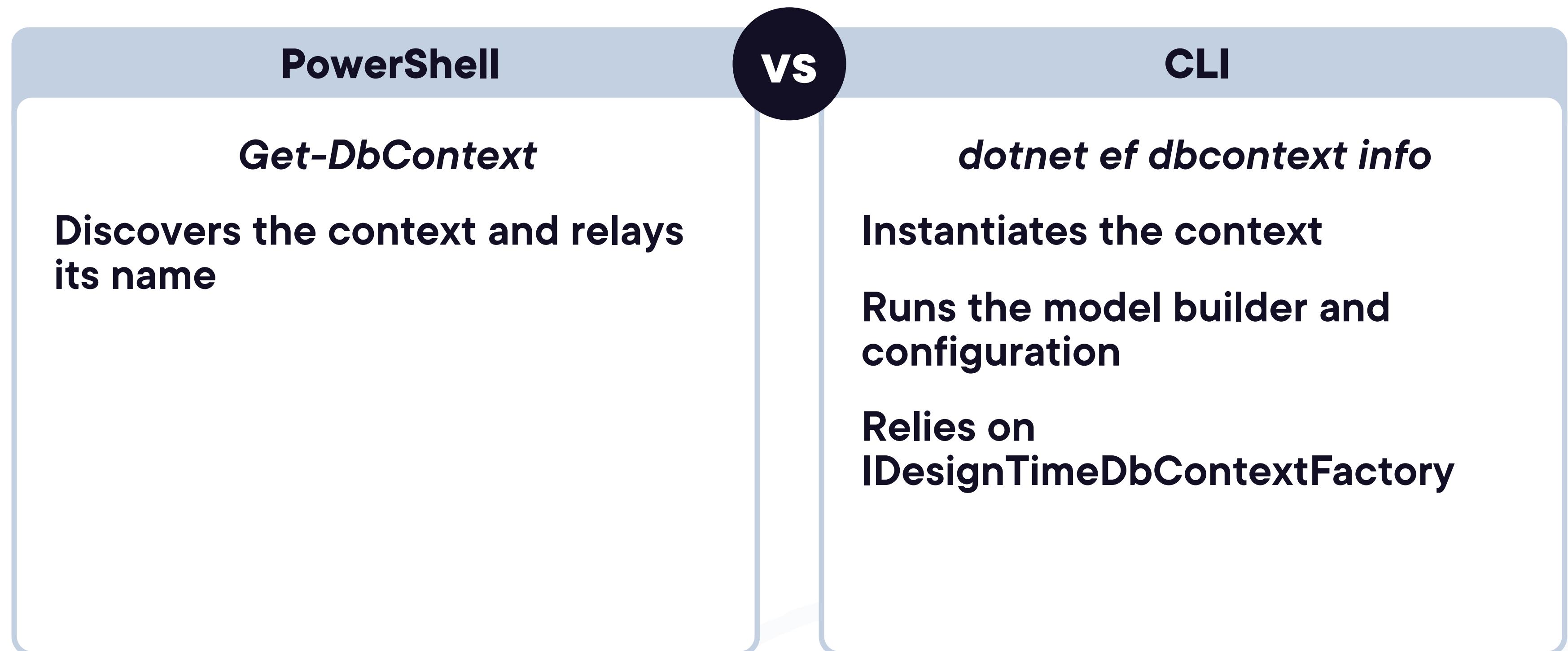
EF Core Expert and Serial DDD Advocate

@julielerman | thedatafarm.com

# | Validating the DbContext via Migrations



# Migration Command to Verify DbContext



```
dotnet tool install --global dotnet-ef
```

```
dotnet tool install --global dotnet-ef --version 6.0.13
```

```
dotnet tool update --global dotnet-ef
```

```
dotnet tool update --global dotnet-ef --version 6.0.13
```

**Be Sure EF Core CLI Tools are Installed**

**And that the version matches the version of EF Core you are using**



# Even Though ContractContext Has Only One DbSet

You can still technically query and add/update/delete non-root entities

```
var versions = _context.Set<ContractVersion>().ToList();
```

```
_context.Add(someContractVersionObject); ← EF Core will detect the entity type  
_context.SaveChanges();
```



# Fixing the DbContext Complaints About Constructors



No suitable constructor was found for entity type 'Contract'.  
The following constructors had parameters  
that could not be bound to properties of the entity type:  
cannot bind 'initDate', 'authors', 'workingTitle'  
in 'Contract(DateTime initDate, List<Author> authors, string workingTitle)'.

## Parsing the Error Message

EF Core uses reflection to materialize objects.

Uses a parameterless constructor or one that matches the scalar properties

Solution 1: Add a constructor that matches the property list

Solution 2: Add a private parameterless constructor without impacting the aggregate!

**Various tools rely on  
parameterless constructors,  
not just EF Core,  
e.g., JSON serialization.**



# Custom Constructors In:

**Contract**

**ContractVersion**

**Author**

**PersonName**

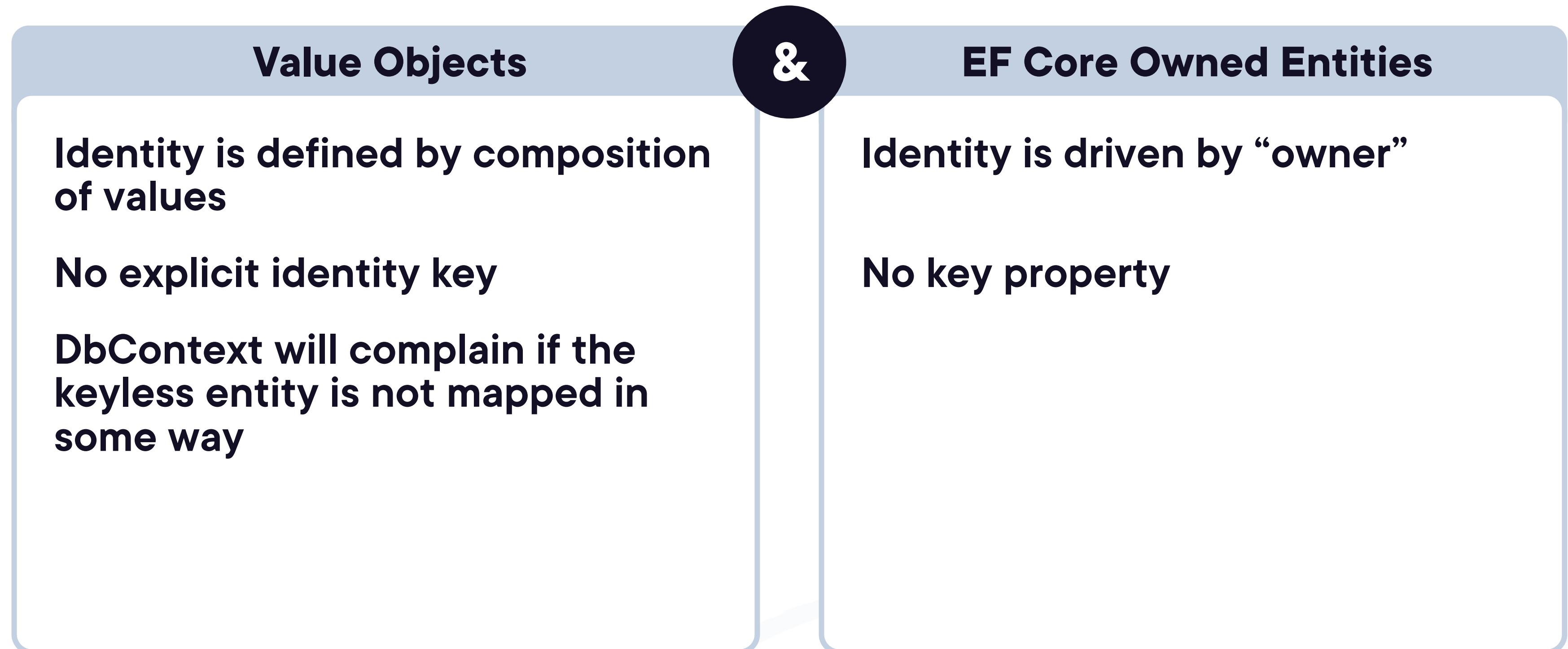
**SpecificationSet**



# Mapping the Value Objects



# Use Owned Entities to Map Value Objects



## Summary



**Added the most basic EF Core DbContext**

**Ensure that EF Core can process the model minimally**

**PowerShell `get-dbcontext` is not very helpful**

**CLI `dbcontext info` does try to build the model**

**Adding private ctors and mapping value objects enables migrations to create a data model – even if it's not perfect yet!**



# Resources Mentioned in this Module

**EF Core CLI Tools Installation Docs**

[learn.microsoft.com/en-us/ef/core/cli/dotnet](https://learn.microsoft.com/en-us/ef/core/cli/dotnet)

**Owned Entity Docs**

[learn.microsoft.com/en-us/ef/core/modeling/owned-entities](https://learn.microsoft.com/en-us/ef/core/modeling/owned-entities)



**Up Next:**

# **Exploring EF Core's Default Mapping of the Data Model**

---

