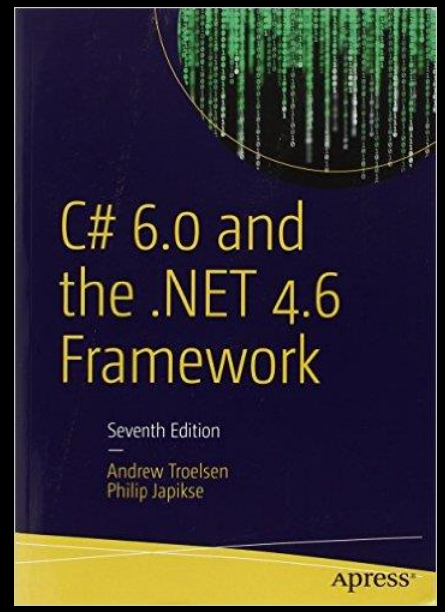
~~10~~ 12 ENTITY FRAMEWORK FEATURES YOU NEED TO KNOW (6.X & CORE 1)

Philip Japikse (@skimedic)
skimedic@outlook.com
www.skimedic.com/blog
Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
Principal Consultant/Architect, Strategic Data Systems

Phil.About()

- Principal Consultant/Architect, Strategic Data Systems
  - http://www.sds-consulting.com
- Developer, Coach, Author, Teacher
  - http://bit.ly/pro_csharp
- Microsoft MVP, ASPInsider, MCSD, MCDBA, CSM, CSP
- Founder, Agile Conferences, Inc.
  - http://www.dayofagile.org
- President, Cincinnati .NET User's Group



#1 - IT'S NOT THE RING OF POWER!

#2 WHEN CHOOSING EF 6.X VS EF CORE 1, CHOSE WISELY

- EF 6.x
  - Windows only
  - Full featured
  - Widely used
    - Lots of resources
- EF Core 1
  - Cross Platform
  - Just released as “RTM”
  - Missing a lot of features still

FEATURE COMPARISON

Creating a Model	EF6.x	EF Core 1.0.0
Custom conventions	Yes	Partial
Inheritance: Table per type (TPT)	Yes	
Inheritance: Table per concrete class (TPC)	Yes	
Shadow state properties		Yes
Alternate keys		Yes
Many-to-many: Without join entity	Yes	
Key generation: Client		Yes
Complex/value types	Yes	
Spatial data	Yes	
Graphical visualization of model	Yes	
Graphical drag/drop editor	Yes	
Model format: EDMX (XML)	Yes	
Reverse engineer model from database: Command line		Yes
Reverse engineer model from database: VS wizard	Yes	
Incremental update model from database	Yes	

Querying Data	EF6.x	EF Core 1.0.0
LINQ: Moderate queries	Stable	Stabilizing
LINQ: Complex queries	Stable	In-Progress
LINQ: Queries using navigation properties	Stable	In-Progress
“Pretty” SQL generation	Poor	Yes
Mixed client/server evaluation		Yes
Loading related data: Lazy	Yes	
Loading related data: Explicit	Yes	
Raw SQL queries: Un-mapped types	Yes	
Saving Data	EF6.x	EF Core 1.0.0
Accessing tracked state	Yes	Partial
Batching of statements		Yes
Stored procedure	Yes	
Detached graph support (N-Tier): Low level APIs	Poor	Yes
Detached graph support (N-Tier): End-to-end		Poor

FEATURE COMPARISON - CONTINUED

Other Features	EF6.x	EF Core 1.0.0
Seed data	Yes	
Connection resiliency	Yes	
Lifecycle hooks (events, command interception, ...)	Yes	
Database Providers	EF6.x	EF Core 1.0.0
MySQL	Yes	Paid only, unpaid coming soon [1]
Oracle	Yes	Paid only, unpaid coming soon [1]
InMemory (for testing)		Yes
Azure Table Storage		Prototype
Redis		Prototype

TOP 10 FEATURES YOU NEED TO KNOW (IN NO PARTICULAR ORDER)

- Multi-Context Migrations
- Find()
- Data Annotations for Generated Classes
- Database Initialization/Seeding
- Concurrency
- EF 6->Logging and Interceptors
- EF Core 1 -> Mixed Evaluation
- Transactions
- Bulk Copy
- Lazy, Eager, Explicit Loading
- Connection Resiliency

- Honorable Mention:
  - Stored Procedure Support
  - Conventions

3) MULTI-CONTEXT MIGRATIONS

EF 6.X MULTI-CONTEXT MIGRATIONS

- EF 6.x now supports more than one DbContext
  - E.g. ApplicationDbContext (ASP.NET Identity) and StoreContext
- Specify Directory and type

```
Enable-Migrations
    -ContextTypeName DAL.EF.FirstContext
    -MigrationsDirectory
EF\Migrations\First
    -ProjectName DAL

add-migration Initial
    -ConfigurationTypeName

DAL.EF.Migrations.First.Configuration
    -ProjectName DAL
    [-Force] [-IgnoreChanges]


update-database
    -targetmigration Initial
    -configurationtypename

DAL.EF.Migrations.First.Configuration
    -projectname DAL –verbose
```

EF CORE 1 MULTI-CONTEXT MIGRATIONS

- Specify:
  - Context (fully qualified)
  - Output Directory fixed in RC2
- Ignore changes isn't supported
  - Must edit the Up method

```
dotnet ef migrations add initial
  -o EF\Migrations
  -c SpyStore.DAL.EF.SpyStoreContext

dotnet ef database update initial
  -c SpyStore.DAL.EF.SpyStoreContext
```

4) FIND()

DBSET<T> FIND METHOD

- EF 6.x
  - Finds entity in context by primary key
  - Else calls to database

```
public T Find(int? id)
{
    var entity =
        Context.ChangeTracker.Entries<T>()
            .Select((EntityEntry e) => (T) e.Entity)
            .FirstOrDefault(x => x.Id == id);
    return entity ?? GetOne(id);
}
```

- EF Core
  - Find doesn't exist
  - Has been added back into the backlog

5) DATA ANNOTATIONS FOR GENERATED CLASSES

ADDING DATA ANNOTATIONS TO DESIGNER GENERATED CLASSES

- Create partial class
- Add MetadataType(typeof(<targetclass>))
- In metadata class, add annotations to public fields
- Use ModelMetadataType for EF Core

```
public class AddressMetadata
{
  [Display(Name="Address Line 1")]
  public string AddressLine1;
}

[MetadataType(typeof(AddressMetaData))]
public partial class Address
{
}
```

6) DATA INITIALIZERS/SEEDING DATA

SEEDING DATA / DATABASE INITIALIZERS EF 6.X

- Seeding With Migrations
  - Typically used to seed initial data
  - Runs with *update-database*

- Data Initializers
  - Used to add data to the database for testing
  - Can be set to reload data every run or on model changes
  - Can be called from code or set in config file

INITIALIZING DATABASE AND DATA

```
 protected override void Seed(
   DAL.EF.FirstContext context)
 {
     //  This method will be called after migrating to the latest
version.
     //  You can use the DbSet<T>.AddOrUpdate() helper
extension method
     //  to avoid creating duplicate seed data. E.g.
     //
     //    context.People.AddOrUpdate(
     //      p => p.FullName,
     //      new Person { FullName = "Andrew Peters" },
     //      new Person { FullName = "Brice Lambson" },
     //      new Person { FullName = "Rowan Miller" }
     //    );
     //
}
```

```
public class StoreInitializer
:DropCreateDatabaseAlways<StoreContext>
//DropCreateDatabaseIfModelChanges<T>
{
    protected override void Seed(StoreContext context)
    {
      StoreData.GetAllStoreRecords()
          .ForEach(x => context.Categories.Add(x));
      context.SaveChanges();
    }
}
Database.SetInitializer(new StoreInitializer());
```

DATA INITIALIZATION - EF CORE

- Largely a manual process
  - Drop/Create base classes from EF 6.x don't exist
- EnsureDeleted
  - Drops database
- EnsureCreated
  - Creates database *based on model*
- EnsureMigrations
  - Created database and runs all migrations as well
  - Usually placed in ctor for DbContext classes
- No way to set initializer – must call from code (Startup.cs?)

7) CONCURRENCY

CONCURRENCY

- Typically used with Timestamp (rowversion) properties
- Error throws DbUpdateConcurrencyException
  - Access entities in error through IEnumerable<DbEntityEntry>
    - CurrentValues
    - OriginalValues (at time of object materialized)
    - GetDatabaseValues (calls database to get server values)
      - 6.x only

8A) LOGGING AND INTERCEPTION

LOGGING INTERCEPTION (BUILT-IN)

- EF 6.1+ has built in logging interceptor
  - Add Interceptor into config file or through code
    - Two parameters
      - File location and name
      - Append to file (default is overwrite)

CUSTOM INTERCEPTORS

- Inherit from IDbCommandInterceptor
  - Add Interceptor into config file or through code
- Largely replaced with DbContext Events
  - ObjectMaterialized
  - SavingChanges

HANDLING THE DBCONTEXT EVENTS

- Get the ObjectContext
  - Must cast context to IObjectContextAdapter
- ObjectMaterialized
  - Occurs when object is reconstituted from data store
  - Exposes entity through the event args
- SavingChanges
  - Fires just before persisting object to data store
  - Must use the ObjectStateManager to get entities involved

8B) CLIENT VS SERVER SIDE EVALUATION

EF CORE SUPPORTS MIXED EVALUATION

- This can help or hurt
  - Especially if you don't understand the intricacies
  - Best to start with it "disabled", then "enable" if needed

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    if (!optionsBuilder.IsConfigured)
    {
        optionsBuilder
            .UseSqlServer(@"Server=(localdb)\mssqllocaldb;
                Database=EFCore_Top_Ten;Trusted_Connection=True;
                MultipleActiveResultSets=true;")
            .ConfigureWarnings(warnings =>
                warnings.Throw(RelationalEventId.QueryClientEvaluationWarning));
    }
}
```

9) TRANSACTIONS

TRANSACTIONS AND ENTITY FRAMEWORK

- Automatic, implicit transactions:
  - SaveChanges
  - Database.ExecuteSQLCommand
  - Both default to server's isolation level
- New in EF 6+
  - Database.BeginTransaction – starts a new transaction
  - Database.UserTransaction – leverages an existing transaction
- Works with async – but use critical thought!

10) BULK COPY

EXECUTING BULK COPY FUNCTIONS

- Create a data reader that implements IDataReader
  - FieldCount, GetName, GetOrdinal, GetValue, Read
- Create new SQLBulkCopy instance
  - Get ConnectionString and DestinationTableName from DbContext
  - Call WriteToServer, passing in data reader

```
var db = new StoreContext();
var connString =
   db.Database.Connection.ConnectionString;

SqlBulkCopy bc = null;
bc = new SqlBulkCopy(connString)
{
  DestinationTableName =
     new AccountTransactionRepo().GetTableName()
};

var dataReader =
    new AccountTransactionDataReader(records);

bc.WriteToServer(dataReader);
```

11) LAZY, EAGER, EXPLICIT LOADING

LAZY (6.X) AND EAGER

- Lazy and Eager are both deferred -
  - Virtual properties are proxied
- Navigation properties not marked virtual are not lazy loaded
- Turn off lazy loading if serializing results (6.x)
  - context.Configuration.LazyLoadingEnabled=false
- Eager combines all objects in one query
  - Categories.Include(x=>x.Products)

  - EF Core adds .ThenInclude()

EXPLICIT LOADING (6.X)

- Used when lazy loading is disabled
- Get Object State Manager entry for entity using
  - context.Entry(foo)
- Load properties through code
  - Collection(x=>x.<ICollection>).Load
  - Reference(x=>x.<Property>).Load
- Can filter what is loaded using Query
  - Collection().Query().Where().Load()

PERFORM EAGER FETCH

- By default, EF 6.1 performs lazy loading of related entities
  - Can force eager fetching per query
- Can change default to be eager (use with caution!)
- To turn off lazy loading for a particular property, do not make it virtual.

```
public List<Category> GetAllWithProducts()
{
    return Context
      .Categories
      .Include(x => x.Products)
      .ToList();
}
```

12) CONNECTION RESILIENCY

CONNECTION RESILIENCY (6.X)

- Built in retry mechanism
  - DefaultExecutionStrategy – no retry
  - SqlAzureExecutionStrategy – optimized for SQL Azure
  - DbExecutionStrategy – abstract class for building custom strategies
- Specify retry count and max delay
  - Throws RetryLimitExceededException
  - Actual exception is inner exception

HM1) STORED PROCEDURE SUPPORT

STORED PROCEDURES

- EF 6 added built in support for Create/Update/Delete stored procedures
- Use the fluent API to map entity to stored procedures
  - Add migration to build the stored procedures
- Names and parameters are configurable
  - Allow for using existing (or DBA generated) stored procedures
- More information available here:
  - https://msdn.microsoft.com/en-us/data/dn468673

```
modelBuilder.Entity<Foo>()
  .MapToStoredProcedures

MapToStoredProcedures(s=>s.Update
  (u=>u.HasName("foo_update")))
```

HM2) CONVENTIONS

EF CONVENTIONS

- Primary Keys
  - Id or <classname>Id
  - Numeric types are identity (MS-SS)
- Type Discovery
  - Related types are “pulled in”
- Navigation Properties
  - Create a shadow property if not specified
- Foreign Keys
  - Not nullable == Cascade Delete
- Complex Types
  - Derived from Entity Types
  - Do not have PKs

Questions?

Contact Me

phil@sds-consulting.com
www.sds-consulting.com

skimedic@outlook.com
www.skimedic.com/blog
www.twitter.com/skimedic
www.hallwayconversations.com

www.about.me/skimedic

Thank You!