TOP 40 11 ENTITY FRAMEWORK 6.1.X FEATURES YOU NEED TO KNOW

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

Microsoft MVP, ASPI Insider, 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!
TOP 10 FEATURES YOU NEED TO KNOW (IN NO PARTICULAR ORDER)

- Migrations/Multi-Migrations
- Conventions/Annotations
- Database Initialization/Seeding
- Concurrency
- Interceptors/Logging
- Transactions
- Bulk Copy
- Lazy, Eager, Explicit Loading
- Stored Procedure Support
- Connection Resiliency
MIGRATIONS
EF NOW SUPPORTS MULTIPLE CONTEXTS IN ONE PROJECT/SOLUTION

Enable-Migrations

Specify Context Type Name, Project and Directory

-ContextTypeName -ProjectName -MigrationDirectory

[Optional] Specify connection string

-ConnectionStringName

Add-Migration && Update-Database

Specify Configuration Type Name and Project

-ConfigurationTypeName -ProjectName
SHARING TABLES

First Context
- Add DbSet<tableName>
- Create migration
- Update database

Second Context
- Add DbSet<tableName>
- Create migration with --ignorechanges
- Update Database
CODE FIRST CONVENTIONS

- Type Discovery
  - Referenced types are pulled in
- Primary Keys –
  - “Id” or “[ClassName]Id”
  - Numeric/Integer becomes Identity
- Navigation Properties
  - ICollection<type>/*<type>*

- Foreign Keys
  - Not Nullable == cascade delete
- Complex Types
  - Derived from Entity Types
  - Do not contain primary key
DATA ANNOTATIONS
FREQUENTLY USED ATTRIBUTES

- Key – sets the primary key
- Composite Keys
- Requires [Column(Order=x)]
- DatabaseGenerated(DatabaseGeneratedOption)
- Identity, Computed, or None
- Index
- Multi Column requires Order
- Name, Unique, Clustered,
- Timestamp – creates RowVersion
- ConcurrencyCheck
- StringLength
- Table(Name,Schema)
- Column(Name,TypeName)
- Required
- NotMapped
- ForeignKey
Adding Data Annotations to Designer Generated Classes

- Create partial class
- Add MetadataType(
  typeof(<targetclass>))
- In metadata class, add annotations to public fields
DATA INITIALIZERS/SEEDING DATA
DATA INITIALIZERS/SEEDING

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

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

- Typically used with Timestamp (rowversion) properties
- Error throws DbUpdateConcurrencyException
- Access entities in error through IEnumerable<DbEntityEntry>

Contains

- CurrentValues
- OriginalValues (at time of object materialized)
- GetDatabaseValues (calls database to get server values)
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
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!
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

```csharp
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);
```
LAZY, EAGER, EXPLICIT LOADING
LAZY AND EAGER

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

- 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 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.

```csharp
public List<Category> GetAllWithProducts()
{
    return Context.
        Categories
        .Include(x => x.Products)
        .ToList();
}
```
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
  ```csharp
  modelBuilder.Entity<Foo>().MapToStoredProcedures
  ```
- Add migration to build the stored procedures
- Names and parameters are configurable
  ```csharp
  MapToStoredProcedures(s=>s.Update(u=>u.HasName("foo_update")))
  ```
- Allow for using existing (or DBA generated) stored procedures
- More information available here:
CONNECTION RESILIENCY
CONNECTION RESILIENCY

- 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 RetryLimitExceeded Exception
- Actual exception is inner exception
Entity Framework
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