



# What's New in Entity Framework 4.0

Anthony Sneed

Email: [tony@tonysneed.com](mailto:tony@tonysneed.com)

Blog: <http://blog.tonysneed.com>

DevelopMentor: <http://develop.com>

# Objectives

- Entity Framework Overview
- POCO Classes
- FK Associations
- T4 Templates
- Lazy Loading
- Change Tracking
- Model-First
- Code-Only
- N-Tier Support

# Q & A

*There will be Q&A sessions both in the **middle** and at the **end** of the presentation.*

*Please ask your questions in **writing**:*

- Twitter
  - #dmscreencast
- Live Meeting
  - Q & A

# What is the Entity Framework?

- **Layer of abstraction**
  - provides a **conceptual model** of the database
  - helps **insulate** your app from db schema changes



# EF 1.0 Limitations

- Lack of POCO support
- No lazy loading
- N-tier development *really* hard
- Other issues
  - unsupported query operators
  - lost sorting instructions
  - concurrency management
  - various glitches and bugs

# Generated entity classes

- Tightly coupled to the Entity Framework
  - entities inherit from **EntityObject**
  - inappropriate for use in a data access layer (DAL)

```
public abstract class EntityObject : StructuralObject,
    IEntityWithKey, IEntityWithChangeTracker,
    IEntityWithRelationships
{
    public EntityKey EntityKey { get; set; }
    public EntityState EntityState { get; }

    // Other members elided for clarity ...
}
```

# POCO classes

- Plain Old CLR Objects
  - classes have **persistence ignorance**

```
public class Category
{
    public int CategoryID { get; set; }
    public string CategoryName { get; set; }
    public List<Product> Products { get; set; }
}
```

**No reference to Entity Framework**



# Object Context

- Container derives from **ObjectContext**
  - properties are of type **ObjectSet<T>**
  - **CreateObjectSet<T>** used to initialize properties

```
public class NorthwindContext : ObjectContext
{
    public ObjectSet<Product> Products { get; set; }

    public NorthwindContext(string conn) :
        base(conn, "NorthwindEntities")
    {
        this.Products = CreateObjectSet<Product>();
    } ...
}
```

# FK Associations

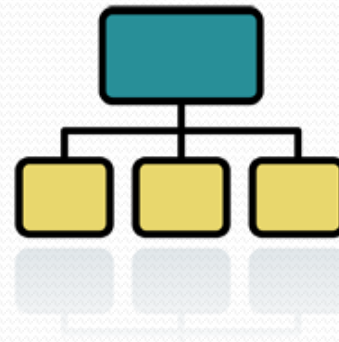
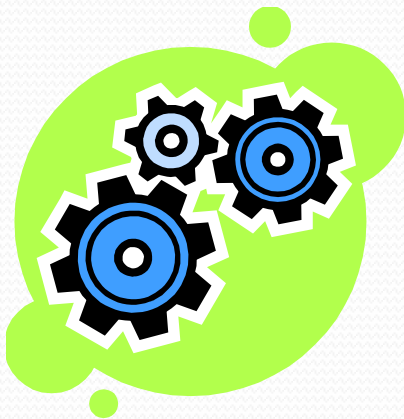
- EF treats Associations are first-class citizens
- But it's often more convenient to assign a related entity based on a **foreign key value**
  - **data binding** scenarios
  - **n-tier** scenarios

```
public class Product
{
    public int ProductID { get; set; }
    public int CategoryID { get; set; }
    public Category Category { get; set; }
}
```



# T4 Templates

- Code-generation technology
  - built into Visual Studio 2008 or later
  - POCO template included with EF 4.0
  - customize or create your own



# Lazy loading

- Enabled using **dynamic proxies**
  - mark navigation properties as **virtual**
  - **ObjectContext.LazyLoadingEnabled** must be set to **true**

```
public class Category
{
    public int CategoryID {get; set;}
    public string CategoryName {get; set;}
    public virtual List<Product> Products {get; set;}
}
```

# Explicit loading

- You may wish to turn off lazy loading
  - you can exercise more control
  - call **LoadProperty** on **ObjectContext**

```
using (var ctx = new NorthwindEntities())
{
    Product product =
        (from p in ctx.Products
         select p).Single();
    ctx.LoadProperty(product, p => p.Category);
    Debug.Assert(p.Category != null);
}
```

loads Product.  
Category





# Eager loading

- Use the **Include** operator to load entities up front
  - supports **chaining** and **nesting**
  - not new to EF 4.0

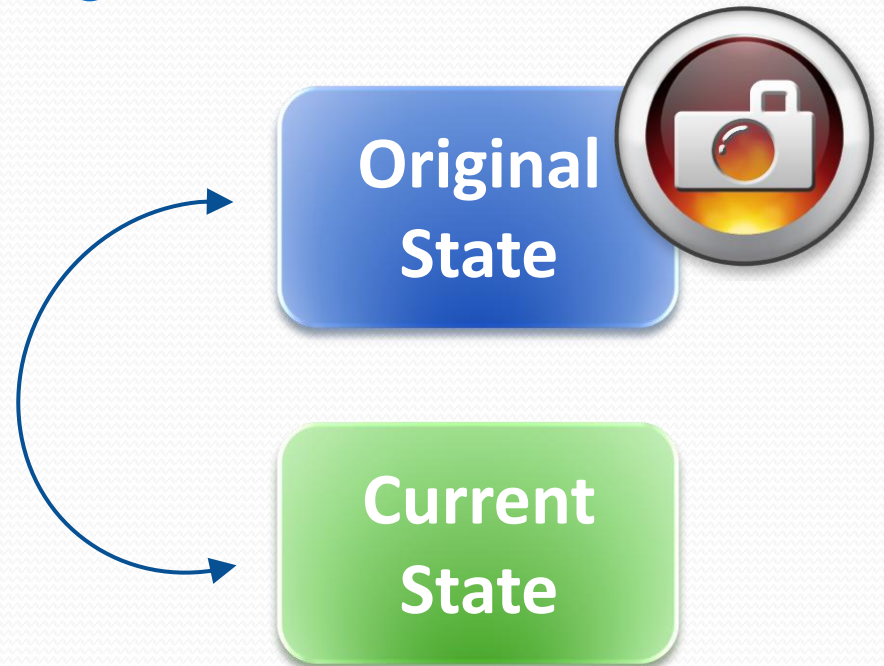
```
using (var ctx = new NorthwindEntities())
{
    var orders =
        from o in ctx.Orders
            .Include("Customer")
            .Include("OrderDetail.Product")
        select o;
}
```

SQL includes  
an **outer join**

# Snapshot change tracking

- EF compares *current* state to **snapshot** of *original* state
  - Performed on **SaveChanges** and **DetectChanges**
  - Sync's up **ObjectStateManager** with the current state

```
product.UnitPrice++;  
ctx.SaveChanges();  
  
product.UnitPrice--;  
ctx.DetectChanges();
```





# Real-time change tracking

- EF uses **dynamic proxies** to track changes
  - proxies subclass POCOs to implement **IEntityWithChangeTracker**
  - make class public with all **virtual** properties
  - navigation properties must be **ICollection<T>**

Also supports  
lazy loading

```
public class Category
{
    public virtual int CategoryID {get; set;}
    public virtual string CategoryName {get; set;}
    public virtual ICollection<Product> Products ...
}
```

# Q & A

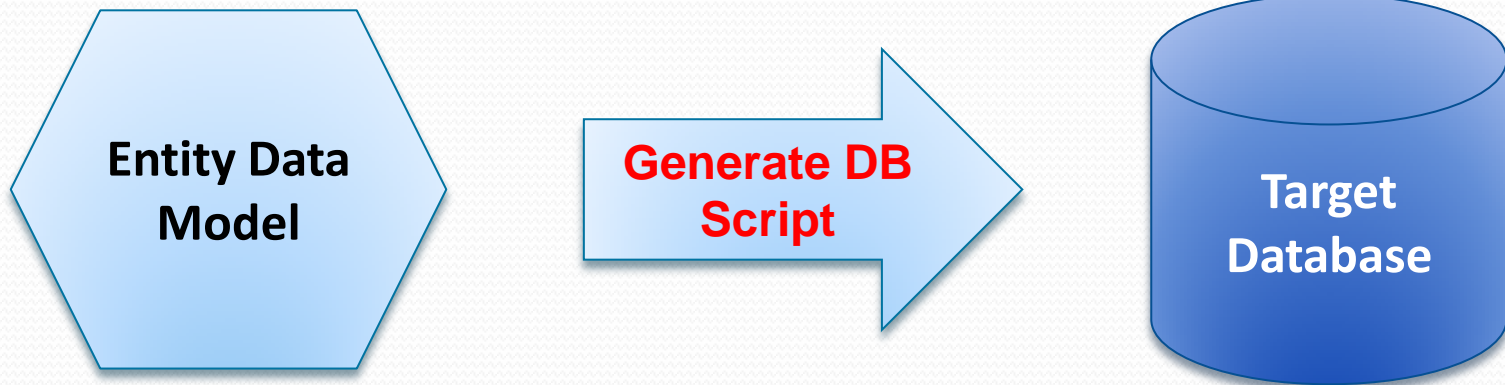
*Please ask your questions in **writing**:*

- Twitter
  - #dmscreencast
- Live Meeting
  - Q & A



# Model-First approach

- Create a **model** from scratch
  - then **generate a database** from the model
  - supports inheritance mapping
  - customizable and extensible





# Code-Only approach

- Create classes and have EF **infer the model**
  - you can also **create the database** dynamically

```
SqlConnection connection = new SqlConnection(...);
using (ProductDataContext cxt =
    ContextBuilder.Create<ProductDataContext>(cn)
{
    if (!context.DatabaseExists())
        context.CreateDatabase();

    // Update and query database
}
```

# N-tier change tracking

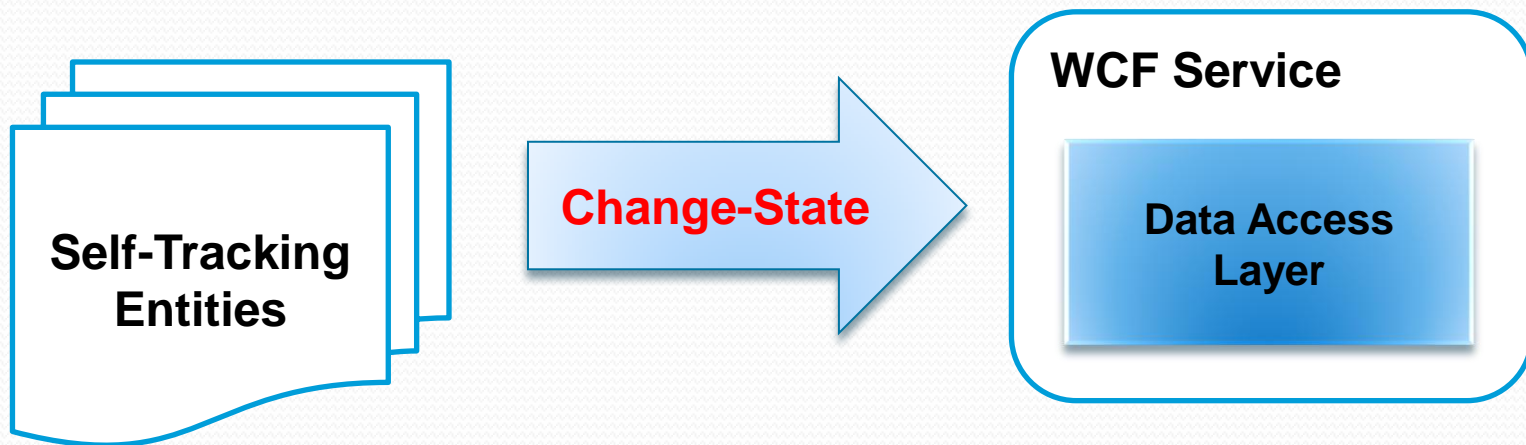
- EF ships with a T4 template for **self-tracking** entities
  - entities contain change-tracking code
  - **ObjectChangeTracker** does the heavy lifting





# N-tier batch updating

- Change state is **serialized** and passed to service
  - client and service reference assembly with STEs
  - client is completely **decoupled** from EF



# What else is new?

- better entity naming
- improved model designer
- foreign key associations
- mapping sprocs to complex types
- support for test-driven development
- model-defined functions
- more LINQ operators
- more efficient SQL generation
- numerous bug fixes, and *much much more!*

# Conclusion

- *Entity Framework 4.0 is ready for prime time!*
- POCO and lazy loading
- End-to-end n-tier support
- Better developer experience
- Lots of new features

# Q & A

*Please ask your questions in **writing**:*

- Twitter
  - #dmscreencast
- Live Meeting
  - Q & A

# EF 4.0 Resources

- Tony Sneed – *blog has slides and code for this talk*
  - blog: <http://blog.tonysneed.com>
  - email: [tony@tonysneed.com](mailto:tony@tonysneed.com)
- DM Course – Essential LINQ with EF 4.0
  - <http://develop.com/course/linq-entity-framework4>
- EF Team Blog
  - <http://blogs.msdn.com/adonet>
- EF 4.0 Online Forum
  - <http://social.msdn.microsoft.com/forums/en-us/adonetefx>