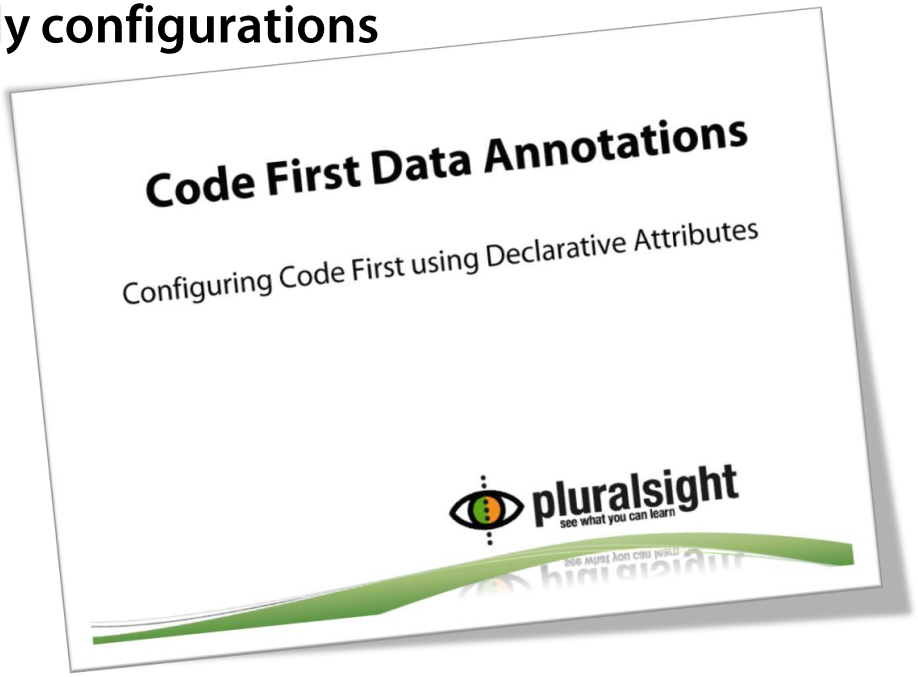
Code First Fluent API

Configuring Entity Framework Code First through the Model Builder

Objectives

- Fluent API Overview
- Review the Various Fluent Configurations
- Quick look at configurations already covered in Data Annotations
- More in-depth look at Fluent-only configurations



Configure Code First Classes for EF EDM

Data Annotations

```
[Table("InternalBlogs")]
   public class Blog{
       [Key, Column("IBKey")]
       public int IdBlog { get; set; }
       [Required]
       public string Title { get; set; }
       [MaxLength(10)]
       public string BloggerName { get; set; }
       [Timestamp]
       public Byte[] TimeStamp { get; set; }
       public BlogDetails BlogDetail { get; set; }
       public virtual ICollection<Post> Posts { get; set; }
       [NotMapped]
       public string BlogCode {
           get {
               return Title.Substring(0, 1) + ":" + BloggerName.Substring(0, 1);
           }
       }
   }
```

Fluent API

```
modelBuilder.Entity<Blog>().HasKey(b => b.IdBlog).ToTable("InternalBlogs");
modelBuilder.Entity<Blog>().Property(p => p.IdBlog).HasColumnName("IBKey");
```

Fluent Configurations

Entity

- Map: Table Name, Schema
- Inheritance Hierarchies
- Complex Types
- Entity->Multiple Tables
- Table->Multiple Entities
- Specify Key (incl. Composite Keys)

Property

- Attributes [w/Validation]
- Map: Column Name, Type, Order
- Relationships
- Concurrency

Configuring with Model Builder

```
public class TwitterContext:DbContext
{
  protected override void OnModelCreating
   (DbModelBuilder modelBuilder)
  {
    //configure Entity
    modelBuilder.Entity<Person>.[configurations…]

    //configure Property
    modelBuilder.Entity<Person>()
                .Property(p => p.Name) .[configurations…]
  . . .
```

Configuration Classes

```
public class PersonConfiguration :
  EntityTypeConfiguration<Person>
   {
       public PersonConfiguration ()
       {
           ToTable(“Personnes",);
           HasKey(e => e.PersonKey);
        }
   }
-----
modelBuilder.Configurations.Add(new PersonConfiguration ());
```

Database Mappings

Entity->Table

```
modelBuilder.Entity<Alias>()
            .ToTable(“TwitterAccounts”, “dbo”);
```

Property->Column

```
modelBuilder.Entity<Alias>()
    .Property(p => p.CreateDate)
                .HasColumnName("StartDate")
                .HasColumnOrder(0)
                .HasColumnType("date");
```

Attributes

Entity

```
.HasKey(e=>e.AuthorKey);
```

Property

```
IsRequired  /  IsOptional
HasMaxLength(##)
IsMaxLength (use database provider's max)
IsFixedLength  /  IsVariableLength
IsUnicode
```

Note: MinLength DataAnnotation has no Fluent counterpart

Modeling Mappings

```
modelBuilder.ComplexType<Privacy>();

modelBuilder.Ignore<Person>();
```

Entity Splitting

```
modelBuilder.Entity<Alias>()
.Map(mc =>
{
   mc.Properties(a => new {a.AuthorKey, properties..});
   mc.ToTable("TwitterAliases");
})
.Map(mc =>
{
  mc.Properties(p => new{a.AuthorKey,p.Avatar});
  mc.ToTable("TwitterAvatar");
 });
```

Table #1

Table #2

Table Splitting

```
public class Alias{
   public int Id{ get; set; }
   public string Name { get; set; }
   public string UserName { get; set; }
   public string Email { get; set; }
   public virtual TwitterAvatar Avatar { get; set; }
}
public class TwitterAvatar{
   public int Id{ get; set; }
   public byte[] Avatar { get; set; }
}
```

```
modelBuilder.Entity<Alias>().ToTable("TwitterAliases");
modelBuilder.Entity<TwitterAvatar>().ToTable("TwitterAliases");
modelBuilder.Entity<Alias>().HasRequired(a => a.Avatar)
                            .WithRequiredPrincipal();
```

Optimistic Concurrency

```
Property(a => a.Name).IsConcurrencyToken();

Property(a => a.RowVersion).IsRowVersion();
```

DatabaseGenerated

```
.Property.HasDatabaseGeneratedOption(DatabaseGeneratedOption.Computed)
```

System.ComponentModel.DataAnnotations
.DatabaseGeneratedOption.Computed
.DatabaseGeneratedOption.Identity
.DatabaseGeneratedOption.None

ForeignKey

```
public class Alias
{
  [Key]
  public int AuthorKey { get; set; }
  . . .

Public class Tweet
{
 . . .
 public int AuthorId { get; set: }
 [ForeignKey("AuthorId")]
 public Author Author { get; set; }
```

InverseProperty

```
public class Alias
{
  ...
   [InverseProperty("AliasAdministrator")]
   public List<Person> Admins { get; set; }
   [InverseProperty("AliasGuestAuthor")]
   public List<Person> GuestAuthors { get; set; }

----------------------------------------

public class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
    public Alias AliasAdministrator { get; set; }
    public Alias AliasGuestAuthor { get; set; }
}
```

Specify Table/Column Names in Many to Many

```
modelBuilder.Entity<BlogPost>()
                .HasMany(p => p.Tags)
                .WithMany(t => t.BlogPosts)
                .Map(mc =>
                   {
                       mc.ToTable("PostTags");
                       mc.MapLeftKey("PostId");
                       mc.MapRightKey("TagId");
                   });
```

Fluent API Mappings

Key field(s)

Relationships

Attributes

Complex types

Inheritance

Table/Column metadata

Table Splitting

Entity Splitting

Resources

- **Pluralsight training: pluralsight.com**
- **MSDN Developer Center: msdn.com/data/ef**
- **EF 4.1 MSDN Documentation:**
  - msdn.microsoft.com/en-us/library/aa139630.aspx
- **EF Team: blogs.msdn.com/adonet**
- **Rowan Miller Blog: romiller.com**
- **Arthur Vickers: blog.oneunicorn.com**
- **Julie Lerman: thedatafarm.com/blog**
- **LearnEntityFramework.com**