Entity Framework 4.1 Introduction

First look at Code First and DbContext in EF 4.1

Objectives

- What's in EF 4.1?
- How Code First works
- Code First Feature Essentials
- DbContext Feature Essentials
- Resources

EF4.1 Adds to EF4

Code First

Model-less EF

DbContext

Simpler Access

Entity Framework 4.0

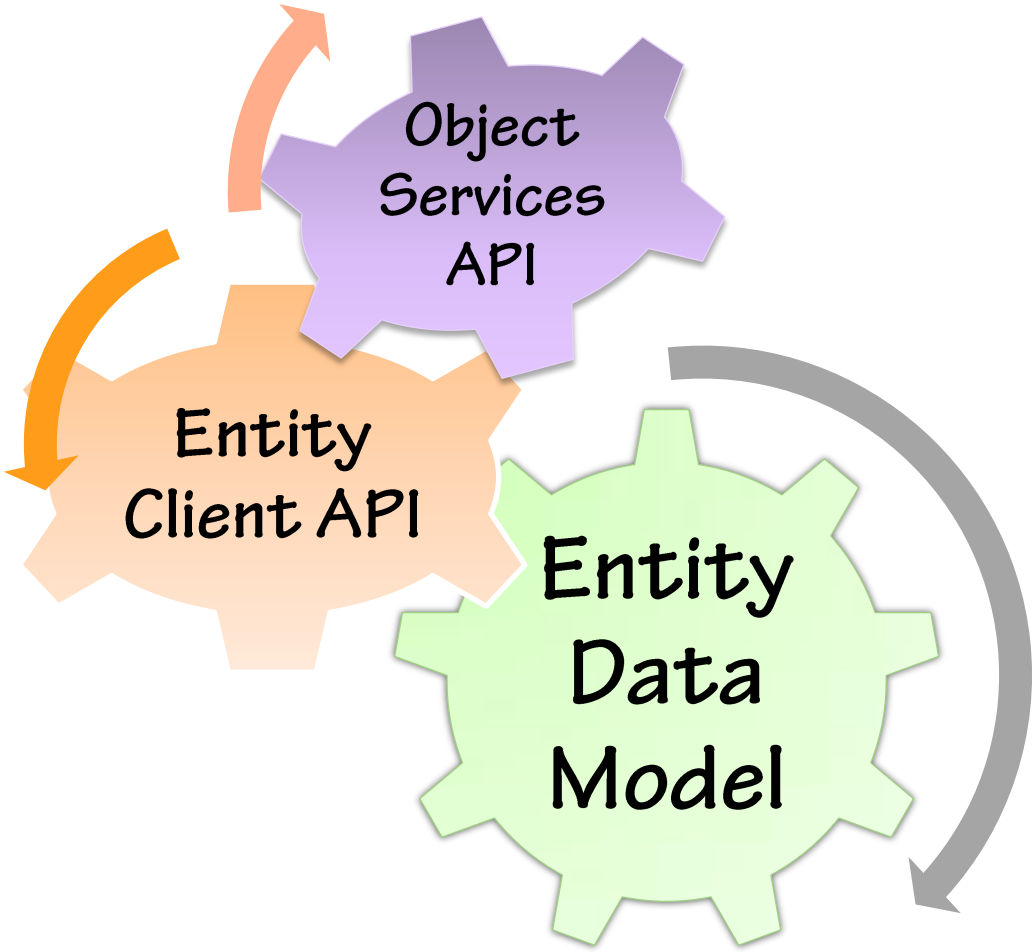
(in .NET 4.0/VS2010 Release)

Getting/Installing Entity Framework 4.1

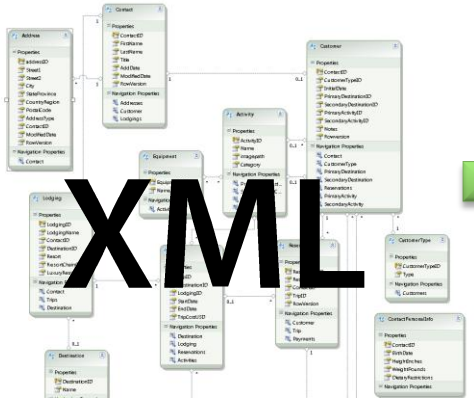
Option 1:
http://msdn.com/data/ef

Option 2:
via Library Package Manager,
aka "Nuget", installed with MVC 3

EF Components



EDMX at Run-Time



System.Data.Metadata.Edm:
EntityType
.EdmProperty
.AssociationType
.EdmType
.Facet

Query Compilation

Table Based Results

Entity Based Results

Configure Code First Classes for EF EDM

Data Annotations

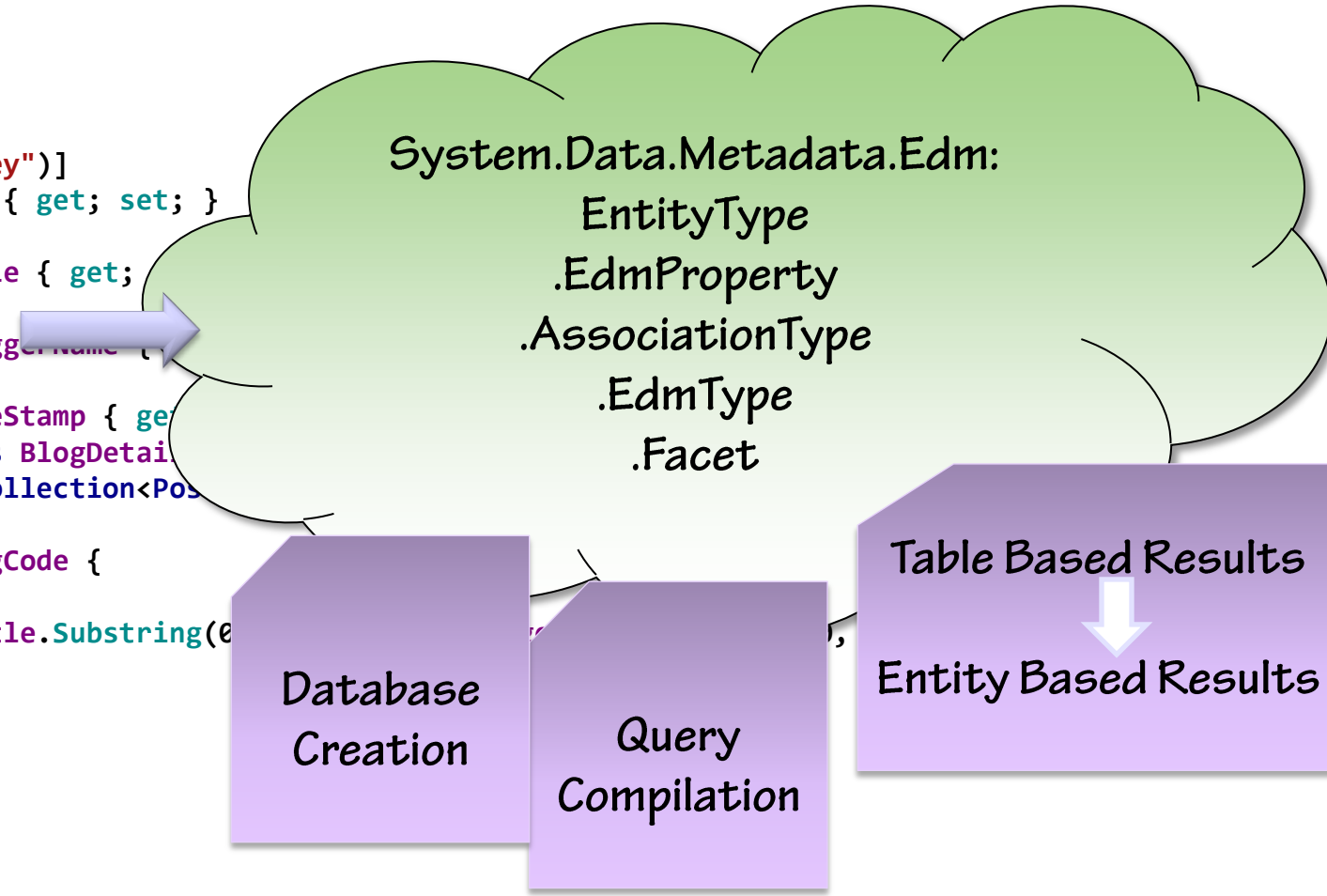
```
[Table("InternalBlogs")]
   public class Blog{
       [Key, Column("IBKey")]
       public int IdBlog { get; set; }
       [Required]
       public string Title { get; set; }
       [MaxLength(10)]
       public string BloggerName { get; set; }
       [Timestamp]
       public Byte[] TimeStamp { 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");
```

Code First at Run-Time

```
[Table("InternalBlogs")]
   public class Blog{
       [Key, Column("IBKey")]
       public int IdBlog { get; set; }
       [Required]
       public string Title { get;
       [MaxLength(10)]
       public string Blogge
       [Timestamp]
       public Byte[] TimeStamp { ge
       public BlogDetails BlogDetai
       public virtual ICollection<Pos
       [NotMapped]
       public string BlogCode {
           get {
               return Title.Substring(0
           }
       }
   }
```



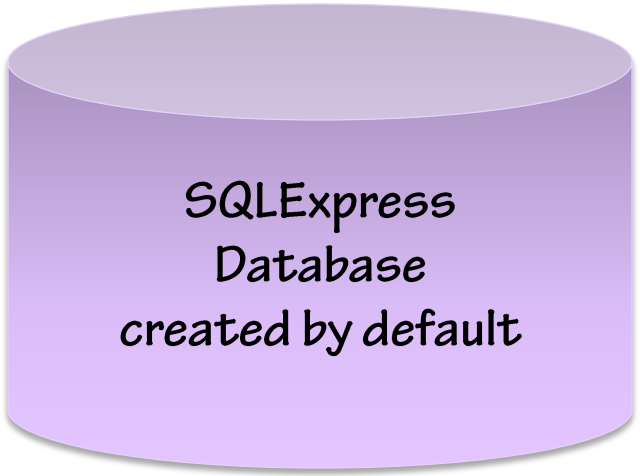
Defaults

Key->
Id or "class"Id

Relationship
inferred

```
public class Blog{
  public int Id { get; set; }
  public string Title { get; set; }
  public string BloggerName { get; set; }
  public List<Post> Posts { get; set; }
  public string BlogCode {
    get {return Title.Substring(0, 1) + ":" + BloggerName.Substring(0, 1);}
  }
}

public class Post{
  public int Id { get; set; }
  public string Title { get; set; }
  public string Content{ get; set; }
  public virtual Blog Blog{ get; set; }
}
```



Configuration: Attributes

Data Annotations:

```
[MaxLength(20), MinLength(5)]
public string Name
```

Fluent API:

```
 modelBuilder.Entity<Post>()
  .Property(p => p.Title).HasMaxLength(10);
```

Configuration: Mappings

Data Annotations:

```
[Column("RandomKey")]
 public int Id

[Timestamp]
public byte[] RowVersion

[NotMapped]
public decimal CalculatedProperty
```

Fluent API:

```
modelBuilder.Entity<Blog>()
  .ToTable("InternalBlogs")

modelBuilder.Entity<Post>()
  .Property(p => p.Content).HasColumnName("Body");
```

Configuration: Relationships

Fix up:

- Many to Many Join Table details
- FK with no Reference Entity
- Non-conventional FK property names
- Etc…

Data Annotations:

```
[ForeignKey("FKBlogId")]
```

Fluent API:

```
modelBuilder.Entity<Post>().HasRequired(p =>
   p.Blog)
.WithMany(b => b.Posts)
.HasForeignKey(p => p.FKBlogId);
```

Database

Initialization Strategies

```
Database.SetInitializer
     (new DropCreateDatabaseIfModelChanges<BlogContext>());
```

or DropCreateDatabaseAlways

or CreateDatabaseIfNotExists

Seeding Data

```
Database.SetInitializer(new MyInitializer());


public class MyInitializer:
   DropCreateDatabaseIfModelChanges<BlogContext>{
      protected override void Seed(BlogContext context){
         ...
      }
}
```

DbContext

- **Simplified Access to:**
  - Cached Data (Local)
  - Locating entities with key value
  - Add/Remove/Attach
  - Modify State
  - Change Tracker Info (Current/Original values, state, etc.)
- **Extensibility Points for Complex Validations**
- **Use with Code First, Database First or Model First**
  - T4 Templates for use with EDMX

Summary

- EF 4.1 adds Code First and DbContext
- Code First for model-less Entity Framework
- Convention over configuration
- Auto-create/seed database or use existing
- DbContext for simpler access to common EF features
- No changes to core API

Resources

- Pluralsight training: pluralsight.com
- MSDN Developer Center: msdn.com/data/ef
- EF Team: blogs.msdn.com/adonet
- Rowan Miller Blog: romiller.com
- Arthur Vickers: blog.oneunicorn.com
- Julie Lerman: thedatafarm.com/blog
- LearnEntityFramework.com