EF-Code First

本节以新数据库为例—博客帖子系统

步骤1 创建一个新控制台项目**CodeFirstNewDatabaseSample**

# 创建模型Code First、EF 设计器和 EF 工作流

EF 模型可存储有关应用程序类和属性如何映射到数据库表和列的详细信息。

EF工作流

|  |
| --- |
|  |

**使用CodeFirst创建模型**

**步骤2**

创建一个Models文件夹，并在其下创建两个类(博客类Blog和博文类Post)

二者在数据库中是一对多关系

|  |
| --- |
|  |
| public class Blog  {  public int BlogId { get; set; }  public string Name { get; set; }  **public virtual List<Post> Posts { get; set; }**  }  //一方加入多方列表 |
| public class Post  {  public int PostId { get; set; }  public string Title { get; set; }  public string Content { get; set; }  **public int BlogId { get; set; }**  **public virtual Blog Blog { get; set; }**  }  //多方加入一方键和实体 |

两个导航属性 （Blog.Posts 和 Post.Blog） 虚拟。 这使实体框架的延迟加载功能。 延迟加载意味着，这些属性的内容将自动从数据库加载时尝试对其进行访问。

# 添加EF引用和创建派生上下文

与应用程序进行交互的主类是System.Data.Entity.DbContext（通常称为上下文类）。 可以使用与模型关联的 DbContext实现

*  编写和执行查询
*  将查询结果具体化为实体对象
*  跟踪对这些对象所做的更改
*  保存在数据库上返回的对象更改
*  在内存中对象绑定到 UI 控件

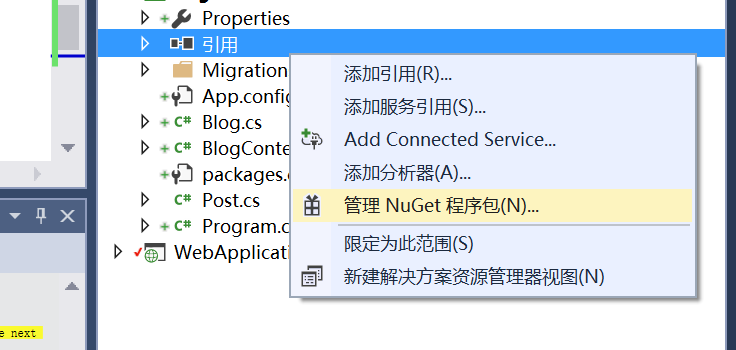
我们定义的上下文，它派生自 System.Data.Entity.DbContext 并公开的类型化的 DbSet<TEntity>我们的模型中的一个类。

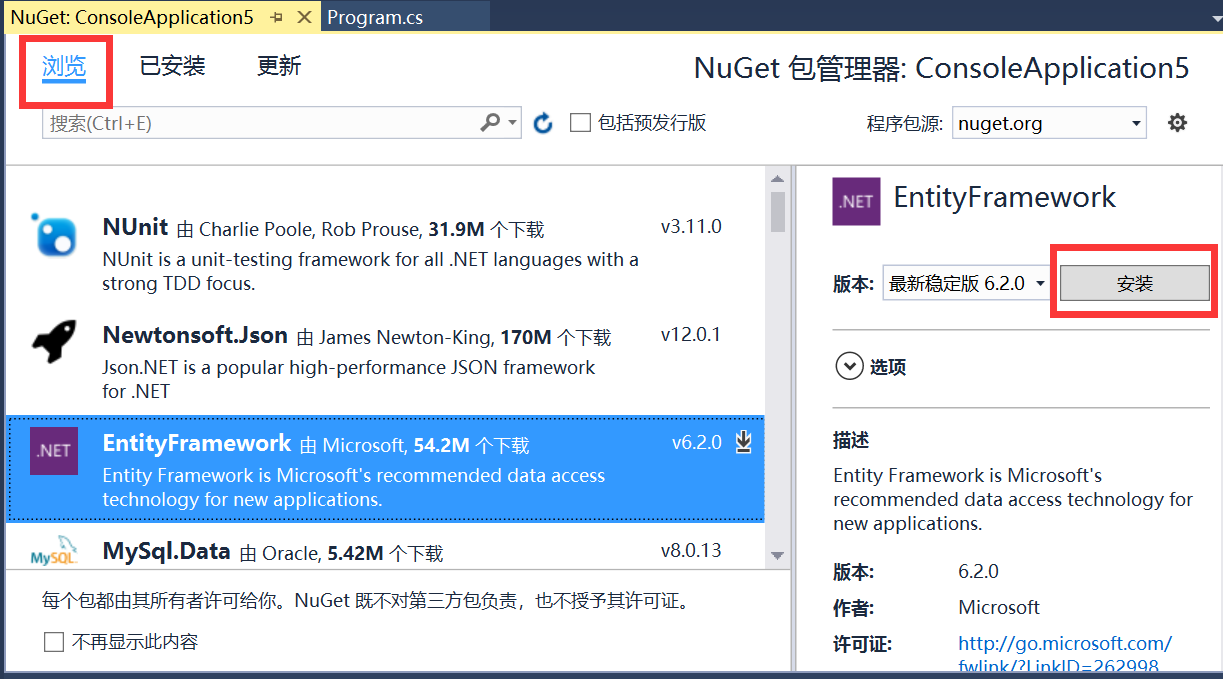
**步骤3：使用以下任意方法获取EF引用**

方法1：使用数据模型工具在添加实体数据模型的同时添加EF应用

方法2：使用NuGet包管理添加引用

**（1）可以在通过右键单击安装 EntityFramework 包引用项目文件夹并选择管理 NuGet 包应用程序**





（2）从包管理控制台安装

工具菜单—》NuGet包管理器—》程序包管理控制台

输入命令

|  |
| --- |
| Install-Package EntityFramework |

无论使用哪种方法，添加EF成功后会在项目引用文件夹中看到EF

|  |
| --- |
|  |

步骤4 创建数据访问层文件夹DataAccessLayer，添加上下文类BloggingContext

|  |
| --- |
|  |
| using CodeFirstNewDatabaseSample.Models;  using System.Data.Entity;  namespace CodeFirstNewDatabaseSample.DataAccessLayer  {  public class **BloggingContext** : DbContext  {  public DbSet<Blog> Blogs { get; set; }  public DbSet<Post> Posts { get; set; }  }  } |

**步骤5 配置文件（app.config）添加链接字符串**

|  |
| --- |
| <connectionStrings>  <add name="**BloggingContext**"  providerName="System.Data.SqlClient"  connectionString="server=.;uid=sa;pwd=123;;Database=Blogging;"/>  </connectionStrings> |

# 生存期

上下文的生存期开始时创建实例，并结束时释放或垃圾回收该实例。

|  |
| --- |
| using (var db = new派生上下文类())  { 数据操作 } |

# 实体状态和 SaveChanges

**扩展阅读**[**https://docs.microsoft.com/zh-cn/ef/ef6/saving/change-tracking/entity-state**](https://docs.microsoft.com/zh-cn/ef/ef6/saving/change-tracking/entity-state)

实体可以是一个五种状态中定义的 EntityState 枚举。

|  |
| --- |
|  |

这些状态包括：

Added： 实体正在由上下文跟踪，但在数据库中尚不存在

Unchanged： 实体正在由上下文跟踪，在数据库中存在并且数据库中的值未更改

Modified： 实体正在由上下文跟踪，在数据库中存在并已修改值

Deleted： 实体正在由上下文跟踪，在数据库中存在但已标记为删除

Detached： 实体未被跟踪的上下文

在以下状态下执行SaveChanges操作后：

Unchanged不涉及通过 SaveChanges。 不会向数据库发送update。

Added添加实体插入到数据库后，状态变为Unchanged。

Modified更新数据库后，装变为Unchanged。

Deleted从数据库中删除，状态变为Detached。

# 将新实体添加到数据库

**步骤6创建业务访问层文件夹，并创建BlogBusinessLayer类**

|  |
| --- |
|  |

BlogBusinessLayer类

|  |
| --- |
|  |

# 查询和查找实体Linq

扩展阅读：<https://docs.microsoft.com/zh-cn/ef/ef6/querying/>

**使用查询查找实体**

DbSet 和 IDbSet 可实现 IQueryable，因此可用作针对数据库编写 LINQ 查询的起点。

**步骤7 BlogBusinessLayer类中添加新方法查询所有博客，并按博客名排序**

|  |
| --- |
|  |

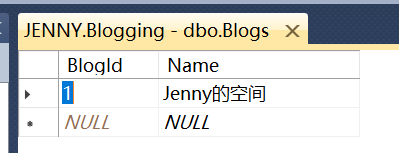
**步骤8在主函数中调用BlogBusinessLayer类添加新博客，并显示所有博客**

|  |
| --- |
|  |

**测试数据**

|  |
| --- |
|  |

**数据库中的状态**



# 更新实体

**步骤9在BlogBusinessLayer类博客业务层类中添加修改Update方法**

|  |
| --- |
|  |

**步骤10 BlogBusinessLayer类博客业务层类中添加根据id查找指定博客的方法**

|  |
| --- |
|  |

**步骤11主函数中添加更新方法调用**

|  |
| --- |
|  |

|  |
| --- |
|  |

# 删除实体

**步骤12在BlogBusinessLayer类博客业务层类中添加修改Delete方法**

|  |
| --- |
|  |

**步骤13在主函数中添加Delete测试模块**

|  |
| --- |
|  |

**测试数据**

|  |
| --- |
|  |