EnterpriseLibrary Guideline

## Overview

EnterpriseLibrary là một thư viện tiện ích được xây dựng trên chuẩn MS Sql giúp cho việc thao tác với cơ sở dữ liệu được tiện lợi, nhanh chóng và tránh mắc lỗi trong quá trình phát triển.

**Ưu điểm:**

* Dễ dàng insert, update vào database mà không cần quan tâm đến cơ sở dữ liệu.
* Một hàm update hoạt động cho nhiều ngữ cảnh nghiệp vụ.
* Không cần change code insert, update khi thay đổi cấu trúc database.
* Dễ dàng gen **entity**, **dao**, **biz** vì có tool đi kèm.
* Tự động checking version, delete flag.
* Hỗ trợ rất nhiều hàm tiện ích để truy vấn như select, select top, paging…

Phần dưới đây sẽ mô tả chi tiết các class, method của EnterpriseLibrary.

## Chú ý trước khi dùng

Thư viện có rất nhiều hàm thao tác với 1 bảng như Update method, select method, delete method. Các hàm này đều đã loại trừ những bản ghi được đánh dấu xóa = 1 cho nên khi sử dụng không cần quan tâm đến các bản ghi này.

Chỉ trong trường hợp application viết câu SqlNative và execute câu truy vấn đó mới cần check đến cột Deleted = 0.

## Định nghĩa Entity Class

Mọi Entity sử dụng EnterpriseLibrary phải được kế thừa từ BaseEntity. Các Properties có attribute để mapping với cơ sở dữ liệu theo cú pháp sau.

Table test thử

|  |
| --- |
| CREATE TABLE [dbo].[Employee](  [EmployeeID] [int] IDENTITY(1,1) NOT NULL,  [Username] [varchar](320) NULL,  [Name] [nvarchar](640) NULL,  [Deleted] [int] NULL,  [Version] [int] NULL,  CONSTRAINT [PK\_Employee] PRIMARY KEY CLUSTERED  (  [EmployeeID] ASC  )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]  ) ON [PRIMARY]  GO |

|  |
| --- |
| public class Employee : BaseEntity  {  public const string C\_EmployeeID = "EmployeeID";  private int? \_EmployeeID;  [PropertyEntity(C\_EmployeeID, true)]  public int? EmployeeID  {  get { return \_EmployeeID; }  set { \_EmployeeID = value; NotifyPropertyChanged(C\_EmployeeID); }  }  public const string C\_Username = "Username";  private string \_Username;  [PropertyEntity(C\_Username, false)]  public string Username  {  get { return \_Username; }  set { \_Username = value; NotifyPropertyChanged(C\_Username); }  }  public const string C\_Name = "Name";  private string \_Name;  [PropertyEntity(C\_Name, false)]  public string Name  {  get { return \_Name; }  set { \_Name = value; NotifyPropertyChanged(C\_Name); }  }  public const string C\_Deleted = "Deleted";  private int? \_Deleted;  [PropertyEntity(C\_Deleted, false)]  public int? Deleted  {  get { return \_Deleted; }  set { \_Deleted = value; NotifyPropertyChanged(C\_Deleted); }  }  public const string C\_Version = "Version";  private int? \_Version;  [PropertyEntity(C\_Version, false)]  public int? Version  {  get { return \_Version; }  set { \_Version = value; NotifyPropertyChanged(C\_Version); }  }  public Employee() : base("Employee", "EmployeeID", "Deleted", "Version") { }  } |

* Attribute PropertyEntity của mỗi Property được dùng để mapping với cột trong cơ sở dữ liệu. Attribute này có 2 tham số.
  + Tham số thứ nhất là tên cột trong cơ sở dữ liệu.
  + Tham số thứ 2 chỉ ra rằng cột đó có phải cột tự tăng hay không.
* Constructor kế thừa từ base constructor có 4 tham số:
  + Tham số tableName: Chỉ ra tên của table tương ứng trong database.
  + Tham số primaryKey: Chỉ ra tên cột khóa chính.
  + Tham số deleteColumn: Chỉ ra tên cột chứa cờ đã xóa của table.
  + Tham số versionColumn: Chỉ ra tên cột chứa version của table.

## Định nghĩa DataContext

DataContext là một class kế thừa từ EnterpriseLibrary với mục đính gán ConnectionString cho EnterpriseLibrary. Việc này giúp cho quá trình develop không phải mất công set connectionString cho mỗi lần sử dụng.

|  |
| --- |
| public class DataContext : EnterpriseService  {  public DataContext()  : base("Data Source=.;Initial Catalog=DBName;User ID=User;Password=MyPass")  {  }  public DataContext(string connectionString)  : base(connectionString)  {  }  } |

* Constructor thứ nhất không có tham số để gán connection mặc định cho EnterpriseLibrary.
* Constructor thứ 2 để sử dụng connectionString khác khi cần nếu ứng dụng có nhiều connectionString.

## InsertItem method.

Dùng để insert một object vào database.

|  |
| --- |
| public void InsertEmployee(DataContext dataContext, EmployeeParam param)  {  dataContext.InsertItem<Employee>(param.Employee);  } |

**Lưu ý:** Dynamic type chỉ ra kiểu object cần insert.

## UpdateItem methods

### UpdateItem

Dùng để update một table theo khóa chính của bản ghi truyền vào và trả ra số lượng bản ghi bị thay đổi.

**Khai báo:**

|  |
| --- |
| public virtual int UpdateItem<T>(BaseEntity entity, bool alwaysIncreaseVersion = false) |

* Hàm này tự động check version của đối tượng truyền vào. Nếu version không được set sẽ không check version. Nếu version được set sẽ check version đồng thời tăng version của bản ghi trong database sau khi update.
* Param entity: Đối tượng cần update. Chỉ những cột được set value từ khi khởi tạo mới được update vào database.
* Option param alwaysIncreaseVersion: Luôn tăng version cho dù không set version vào object.

**Sử dụng:**

|  |
| --- |
| public void UpdateEmployee(DataContext dataContext, EmployeeParam param)  {  int affectedRows = dataContext.UpdateItem<Employee>(param.Employee);  if (affectedRows == 0)  {  throw new SMXException(Messages.Common.ItemChanged);  }  } |

### UpdateItemByColumnName – Một điều kiện

Dùng để update một hoặc nhiều record theo một cột bất kỳ.

**Khai báo:**

|  |
| --- |
| public virtual int UpdateItemByColumnName<T>(BaseEntity entity, string columnName, object value, bool alwaysIncreaseVersion = false) |

* Hàm này không tự động check version của đối tượng cần update.
* Param entity: Đối tượng cần update. Chỉ những cột được set value từ khi khởi tạo mới được update vào database.
* Param columnName: Tên cột cần so sánh trong điều kiện update
* Param value: Giá trị cột cần so sánh trong điều kiện update
* Option param alwayIncreaseVersion: Tự động tăng version khi update

### UpdateItemByColumnName – Nhiều điều kiện

Dùng để update một hoặc nhiều record theo nhiều điều kiện truyền vào

**Khai báo:**

|  |
| --- |
| public virtual int UpdateItemByColumnName<T>(BaseEntity entity, ConditionList condition, bool alwaysIncreaseVersion = false) |

* Hàm này không tự động check version của đối tượng cần update.
* Param entity: Đối tượng cần update. Chỉ những cột được set value từ khi khởi tạo mới được update vào database.
* Option param alwayIncreaseVersion: Tự động tăng version khi update

**Sử dụng:**

|  |
| --- |
| Employee enUpdating = new Employee();  enUpdating.Deleted = 1;  int affectedRows = dataContext.UpdateItemByColumnName<Employee>(param.Employee,  new ConditionList()  {  {Employee.C\_Status, 1},  {Employee.C\_Gender, 1},  }, true);  if (affectedRows == 0)  {  throw new SMXException(Messages.Common.ItemChanged);  } |

* Tập lệnh trên update cột Deleted = 1 cho tất cả Employee có Status = 1 và Gender = 1
* Version của các bản ghi sẽ tự động tăng sau update vì Option param alwaysIncreaseVersion = true

## Delete methods

### DeleteItem

Dùng để xóa một item theo khóa chính.

**Khai báo:**

|  |
| --- |
| public virtual int DeleteItem<T>(object id) |

* Hàm này thực hiện xóa thật bản ghi trong database.
* Trả ra số lượng bản ghi bị xóa

**Sử dụng:**

|  |
| --- |
| int affectedRows = dataContext.DeleteItem<Employee>(1);  if (affectedRows == 0)  {  throw new SMXException(Messages.Common.ItemChanged);  } |

* Tập lệnh trên sẽ xóa Employee có ID = 1

### DeleteItemByColumnName

Dùng để xóa một hoặc nhiều Item theo một cột bất kỳ trong bảng và trả ra số lượng bản ghi đã xóa.

**Khai báo:**

|  |
| --- |
| public virtual int DeleteItemByColumn<T>(string columnName, object value) |

* Hàm này thực hiện xóa thật bản ghi trong database.
* Param columnName: Chỉ ra cột cần so sánh trong điều kiện xóa.
* Param value: Chỉ ra giá trị cột cần so sánh trong điều kiện xóa.

**Sử dụng:**

|  |
| --- |
| int affectedRows = dataContext.DeleteItemByColumn<Employee>(Employee.C\_Status, 1)  if (affectedRows == 0)  {  throw new SMXException(Messages.Common.ItemChanged);  } |

* Tập lệnh trên sẽ xóa tất cả Employee có cột Status = 1

## Select methods

### SelectItemByID

Select 1 bản ghi trong database với điều kiện là khóa chính truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual T SelectItemByID<T>(object id) |

**Sử dụng:**

|  |
| --- |
| param.Employee = dataContext.SelectItemByID<Employee>(param.EmployeeID); |

### SelectItemByColumnName – Một điều kiện

Select nhiều bản ghi trong database thỏa mãn một điều kiện truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual List<T> SelectItemByColumnName<T>(string columnName, object value) |

* Param columnName: Tên cột cần so sánh.
* Param value: Giá trị cột cần so sánh

**Sử dụng:**

|  |
| --- |
| param.Employees = dataContext.SelectItemByColumnName<Employee>(Employee.C\_Status, 1); |

* Lệnh trên sẽ select toàn bộ employee có Status = 1

### SelectItemByColumnName – Nhiều điều kiện

Select nhiều bản ghi trong database thỏa mãn nhiều điều kiện truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual List<T> SelectItemByColumnName<T>(ConditionList conditions) |

* Param conditions: Tập điều kiện cần so sánh.

**Sử dụng:**

|  |
| --- |
| param.Employees = dataContext.SelectItemByColumnName<Employee>(  new ConditionList()  {  {Employee.C\_Status, 1},  {Employee.C\_Gender, 1}  }); |

* Lệnh trên sẽ select toàn bộ employee có Status = 1 và Gender = 1

### SelectFieldsByID

Select một vài trường của 1 bản ghi trong database với điều kiện khóa chính truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual T SelectFieldsByID<T>(string[] fieldNames, object key) |

* Param fieldNames: Tập cột cần select
* Param key: Giá trị khóa chính

**Sử dụng:**

|  |
| --- |
| param.Employee = dataContext.SelectFieldsByID<Employee>(new string[] { Employee.C\_EmployeeID, Employee.C\_DisplayName }, 1); |

* Lệnh trên sẽ select cột EmployeeID, DisplayName của bảng Employee có khóa chính = 1

### SelectFieldsByColumnName – Một điều kiện

Select một vài trường của các bản ghi trong database thỏa mãn 1 điều kiện truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual List<T> SelectFieldsByColumnName<T>(string[] fieldNames, string columnName, object value) |

* Param fieldNames: Tập cột cần select
* Param columnName: Cột cần so sánh trong điều kiện
* Param: value: Giá trị cột cần so sánh trong điều kiện

**Sử dụng:**

|  |
| --- |
| param.Employees = dataContext.SelectFieldsByColumnName<Employee>(new string[] { Employee.C\_EmployeeID, Employee.C\_DisplayName }, Employee.C\_Status, 1); |

* Lệnh trên sẽ select cột EmployeeID, DisplayName của bảng Employee theo điều kiện cột Status=1

### SelectFieldsByColumnName – Nhiều điều kiện

Select một vài trường của các bản ghi trong database thỏa mãn nhiều điều kiện truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual List<T> SelectFieldsByColumnName<T>(string[] fieldNames, ConditionList conditions) |

* Param fieldNames: Tập cột cần select
* Param conditions: Danh sách điều kiện so sánh

**Sử dụng:**

|  |
| --- |
| param.Employees = dataContext.SelectFieldsByColumnName<Employee>(new string[] { Employee.C\_EmployeeID, Employee.C\_DisplayName },  new ConditionList()  {  {Employee.C\_Status, 1},  {Employee.C\_Gender, 1}  }); |

* Lệnh trên sẽ select cột EmployeeID, DisplayName của bảng Employee theo điều kiện cột Status=1 và cột Gender = 1

## Select top methods

### SelectTopItemByColumnName – Một điều kiện

Sử dụng select top bản ghi trong table theo 1 điều kiện truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual List<T> SelectTopItemByColumnName<T>(string columnName, object value, int top, string orderingStatement = "") |

* Param columnName: Tên cột cần so sánh
* Param value: Giá trị cột cần so sánh
* Param top: Số lượng bản ghi cần select
* Param orderingStatement: Biểu thức order, mặc định là tăng dần của khóa chính

**Sử dụng:**

|  |
| --- |
| param.Employees = dataContext.SelectTopItemByColumnName<Employee>(Employee.C\_Status, 1, 10); |

* Lệnh trên sẽ select 10 Employee có cột Status = 1

### SelectTopItemByColumnName – Nhiều điều kiện

Sử dụng select top bản ghi trong table theo nhiều điều kiện truyền vào.

**Khai báo:**

|  |
| --- |
| public virtual List<T> SelectTopItemByColumnName<T>(ConditionList conditions, int top, string orderingStatement = "") |

* Param columnName: Tên cột cần so sánh
* Param conditions: Tập điều kiện so sánh
* Param top: Số lượng bản ghi cần select
* Param orderingStatement: Biểu thức order, mặc định là tăng dần của khóa chính

**Sử dụng:**

|  |
| --- |
| param.Employees = dataContext.SelectTopItemByColumnName<Employee>(  new ConditionList()  {  {Employee.C\_Status, 1},  {Employee.C\_Gender, 1}  },10, "EmployeeID desc"); |

* Lệnh trên sẽ select 10 Employee có cột Status = 1 và Gender = 1 sắp xếp theo thứ tự đảo ngược của khóa chính

### SelectTopFieldsByColumnName

Tương tự như select topItem nhưng cần truyền vào một mảng string để chỉ ra danh sách các cột cần select.

## Execute select

Thực hiện execute một câu select hoặc một SqlCommand và cast giá trị trả về theo Dynamic type truyền vào.

**Sử dụng:**

|  |
| --- |
| string sql = @"select EmployeeID , UserName , DisplayName from Employee  where Deleted = 0";  SqlCommand command = new SqlCommand(sql);  param.Employees = dataContext.ExecuteSelect<Employee>(command); |

## Execute select top

Thực hiện select top bản ghi từ một câu truy vấn hoặc một SqlCommand sau đó cast giá trị trả về theo Dynamic type truyền vào.

**Sử dụng:**

|  |
| --- |
| string sql = @"select \* from Employee  where Deleted = 0 and Email = @Email and EmployeeID <> @EmployeeID";  SqlCommand command = new SqlCommand(sql);  command.Parameters.AddWithValue("@Email", param.Email);  command.Parameters.AddWithValue("@EmployeeID", param.EmployeeID);  param.Employee = dataContext.ExecuteSelectTop<Employee>(command, 1, "EmployeeID desc").FirstOrDefault(); |

* Tập lệnh trên trả ra 1 bản ghi cuối cùng của câu truy vấn.

## Execute paging

Thực hiện paging record của một câu truy vấn sau đó cast giá trị trả về theo Dynamic type truyền vào. Đồng thời out số lượng record của câu truy vấn đó.

**Khai báo:**

|  |
| --- |
| public virtual List<T> ExecutePaging<T>(SqlCommand command, int pageIndex, int pageSize, string orderStatement, out int recordCount, bool countingRecords = true) |

* Param command: Câu truy vấn select all dữ liệu chưa bao gồm paging
* Param pageIndex: Chỉ số của trang cần lấy dữ liệu
* Param pageSize: Độ lớn của 1 trang
* Param orderStatement: Biểu thức order trên tập dữ liệu để phân trang
* Param recordCount: Số lượng bản ghi sẽ trả ra
* Param countingRecord:
  + True: Có thực hiện đếm tổng số bản ghi
  + False: Không thực hiện đếm tổng số bản ghi: Sử dụng trong trường hợp báo cáo cần lấy truy vấn nhiều lần, chỉ cần lấy dữ liệu mà không cần lấy số lượng bản ghi

**Sử dụng:**

|  |
| --- |
| EmployeeFilter enSearch = param.EmployeeFilter;  PagingInfo pInfo = param.PagingInfo;  string sql = @"select \* from Employee  where Deleted = 0  AND (Username like @Username)  AND (DisplayName like @DisplayName)  AND (Mobile like @Mobile)  AND (Email like @Email)  AND (@Status is null OR Status = @Status)";  SqlCommand command = new SqlCommand(sql);  command.Parameters.AddWithValue("@UserName", BuildLikeFilter(enSearch.UserName));  command.Parameters.AddWithValue("@DisplayName", BuildLikeFilter(enSearch.DisplayName));  command.Parameters.AddWithValue("@Mobile", BuildLikeFilter(enSearch.Mobile));  command.Parameters.AddWithValue("@Email", BuildLikeFilter(enSearch.Email));  command.Parameters.AddWithValue("@Status", enSearch.Status);  int recordCount = 0;  param.Employees = dataContext.ExecutePaging<Employee>(command, pInfo.PageIndex, pInfo.PageSize, "UserName", out recordCount);  pInfo.RecordCount = recordCount; |

## Các hàm execute nguyên thủy của Sql

* **ExecuteDataTable**: Thực thi truy vấn và trả ra 1 DataTable
* **ExecuteNoneQuery**: Thực thi truy vấn và trả về số lượng bản ghi bị ảnh hưởng.
* **ExecuteScalar**: Thực thi truy vấn và trả về giá trị ở cell đầu tiên.

***Hết***

***Mario*** *05-2014*