LogixHealth Coding Standards and Guidelines

Terminology and Definitions

The following terminology is referenced throughout this document	
Coloring & Emphasis	
Blue	Text colored blue indicates a C# keyword or .NET type
Bold	Text with additional emphasis to make it stand-out
Keywords	
Always	Emphasizes this rule must be enforced
Never	Emphasizes this action must not happen
Do Not	Emphasizes this action must not happen
Avoid	Emphasizes that the action should be prevented, but some exceptions may exist
Try	Emphasizes that the rule should be attempted whenever possible and appropriate
Example	Precedes text used to illustrate a rule or recommendation
Reason	Explains the thoughts and purpose behind a rule or recommendation

How To – Consume LH Enterprise Application Blocks

Overview – LH Enterprise Application Blocks

Introduction

LH Enterprise Application Blocks is a collection of reusable components, which are programming libraries developed using .NET Framework and .NET Core 2.1 Framework. They are designed to help developer's deal with cross cutting concerns such as data access, validation, logging and exception handling. Following are the LogixHealth Enterprise Components –

- Enterprise Logger
- Enterprise Data Access Library
- Enterprise Email Gateway
- Enterprise App Services Gateway
- Enterprise Cryptography
- .NET Core Extensions
- Authentication Extension Services

How To – Consume LH Enterprise Application Blocks "Enterprise Data Access Library"

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – General Guidelines

LogixHealth Enterprise Library for Data Access – This component provides an interfaces for accessing the data from the SQL Server Database that reduce the amount of code that is required

This component has two versions,

- One – Developed using .NET Framework 4.5 (can be consumed by applications developed using .NET Framework(s))
- Two – Developed using .NET Core 2.1 (can be consumed by applications developed using .NET Core)

Go to TFS - http://bedtfsapp001:8080/tfs/main > EnterpriseLibraries

Map this TFS folder $/EnterpriseLibraries/StableVersion to [C: or D: drive]:\TFS\EnterpriseLibraries\StableVersion and Get the latest code base

Go to your application folder and open your application solution file in Visual Studio 2017

- Right click on Solution file > Add > click New Solution Folder > name this folder as "LogixHealth.EnterpriseLibraries"
- Right click on this Solution Folder "LogixHealth.EnterpriseLibraries" and add the Enterprise Data Access Library project from the location mentioned where you have mapped.
  - LogixHealth.EnterpriseLibrary.DataAccess.Core – If you`re working on .NET Core Applications
  - LogixHealth.EnterpriseLibrary.DataAccess – If you`re working on .NET Framework Applications
- Right click on your Data Access Application/Project > click Add > click Reference > Projects > select Enterprise Library Data Access project (If NOT referenced)

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library

Go to your .NET Core Data Access Application > Add References > LogixHealth.EnterpriseLibrary.DataAccess.Core as project reference. Initialize and Call the respective Enterprise Data Access methods accordingly

Go to your .NET Data Access Application > Add References > LogixHealth.EnterpriseLibrary.DataAccess as project reference. Initialize and Call the respective Enterprise Data Access methods accordingly

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – Interface Definitions

IUnitOfWork – Unit of Work is the concept related to the effective implementation of the repository pattern.

IRepository<T> – Repository pattern, the domain entities, the data access logic and the business logic talk to each other using interfaces.

IQueryRepository<T> – Query Repository pattern, the data access logic in stored procedures and the business logic talk to each other using interfaces

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IRepository<T>

In your Data Access Classes, declare unit of work and repository Interfaces.
Example –

```
IUnitOfWork unitOfWork;
IRepository<ApplicationArea> repository;
```

Note – In the above example ApplicationArea is a business entity or model

In the constructor or method instantiate these interfaces.
Example –

```
unitOfWork = UnitOfWorkFactory.CreateUnitOfWork(_connectionString);
repository = unitOfWork.CreateRepository<ApplicationArea>("Connect", "ApplicationArea");
```

Note – In the above example “Connect” is the schema and “ApplicationArea” is the table name

Using the repository to call the respective methods to fetch the records.

IRepository – Can be used if you want to fetch any rows from a table directly without using SPs

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IRepository<T>

Methods Definitions in IRepository<T> interface

Signature

```
/// <summary>
/// This method fetches all the rows from the respective Table
/// </summary>
/// <returns>collection of entities</returns>
System.Collections.Generic.ICollection<T> AllRecords();
```

Example

```
ICollection<ApplicationArea> applicationAreas = repository.AllRecords();
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IRepository<T>

Methods Definitions in IRepository<T> interface

Signature

```
/// <summary>
/// This method fetches all the rows from the respective Table based on the where clause
/// </summary>
/// <param name="where">filters</param>
/// <returns>collection of entities</returns>
ICollection<T> FindRecords(System.Linq.Expressions.Expression<System.Func<T, bool>> where);
```

Example

```
ICollection<ApplicationArea> applicationAreas = repository.FindRecords
                (
                    x =>
                        x.Id == 17 &&
                        x.AreaName == "R2C1" &&
                        x.UrlAction == "R2C3"
                );
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Fetches all the records/data from database
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>List of objects</returns>
1 reference
IEnumerable<T> All(string storedProcedureName, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    return query.All(_usp_GetAllApplicationArea).ToList();
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Fetches all the records/data from database based on the filter criteria (i.e. parameters)
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>List of objects</returns>
1 reference
IEnumerable<T> GetData(string storedProcedureName, IDictionary<string, object> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userId", userId },
        { "defaultApplicationImage", defaultApplicationImage },
        { "organizationId", 0 }
    };

    return query.GetData(_usp_GetAllApplicationArea, dictionary).ToList();
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Fetches all the records/data from database based on the filter criteria (i.e. parameters)
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="tableTypeName">name of the table type passed as parameter</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>List of objects</returns>
1 reference
IEnumerable<T> GetData(string storedProcedureName, IDictionary<string, object> parameters, string tableTypeName, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userInfo", userInfo }
    };

    return query.GetData(_usp_GetAllApplicationArea, dictionary, Udt_UserInfo);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Finds the specific record from database based on the filter criteria
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>respective entity/object/poco</returns>
1 reference
T SingleOrDefault(string storedProcedureName, IDictionary<string, object> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userId", userId },
        { "defaultApplicationImage", defaultApplicationImage },
        { "organizationId", 0 }
    };

    return query.SingleOrDefault(_usp_GetAllApplicationArea, dictionary);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Finds the specific record from database based on the filter criteria
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="tableTypeName">name of the table type passed as parameter</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>respective entity/object/poco</returns>
1 reference
T SingleOrDefault(string storedProcedureName, IDictionary<string, object> parameters, string tableTypeName, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userInfo", userInfo }
    };

    return query.SingleOrDefault(_usp_GetAllApplicationArea, dictionary, Udt_UserInfo);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Finds the specific record from database based on the filter criteria
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>respective string</returns>
0 references
string GetScalarValue(string storedProcedureName, IDictionary<string, object> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userId", userId },
        { "defaultApplicationImage", defaultApplicationImage },
        { "organizationId", 0 }
    };

    return query.GetScalarValue(_usp_GetAllApplicationArea, dictionary);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Inserts/Updates the changes to the database object (i.e. table) based on the entity state
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>affected rows</returns>
1 reference
bool InstertOrUpdate(string storedProcedureName, IDictionary<string, object> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userId", userId },
        { "firstName", "My First Name" },
        { "lastName", "My Last Name" }
    };

    return query.InstertOrUpdate(_usp_GetAllApplicationArea, dictionary);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Inserts/Updates the changes to the database object and it`s child (i.e. master/detail table) based on the entity state
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair and direction of the parameter</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>returns values for the out parameters</returns>
1 reference
bool InstertOrUpdate(string storedProcedureName, IEnumerable<NameValueType> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    ICollection<NameValueType> dictionary = new Collection<NameValueType>
    {
        new NameValueType { Name="param 1", Value="value 1", IsTableType=false, TableTypeName=null },
        new NameValueType { Name="param 2", Value="value 2", IsTableType=false, TableTypeName=null },
        new NameValueType { Name="param 3", Value="object value 3", IsTableType=true, TableTypeName=Udt_UserInfo }
    };

    return query.InstertOrUpdate(_usp_GetAllApplicationArea, dictionary);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Inserts/Updates the changes to the database object (i.e. table) based on the entity state
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair and direction of the parameter</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>returns values for the out parameters</returns>
1 reference
IDictionary<string, object> InstertOrUpdate(string storedProcedureName, IDictionary<string, Tuple<object, ParameterDirection>> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, Tuple<object, ParameterDirection>> dictionary = new Dictionary<string, Tuple<object, ParameterDirection>>
    {
        { "param 1", new Tuple<object, ParameterDirection>("value 1", ParameterDirection.Input)},
        { "param 2", new Tuple<object, ParameterDirection>("value 2", ParameterDirection.Input)},
        { "param 3", new Tuple<object, ParameterDirection>("value 3", ParameterDirection.InputOutput)},
        { "param 4", new Tuple<object, ParameterDirection>("value 4", ParameterDirection.Output)},
    };

    return query.InstertOrUpdate(_usp_GetAllApplicationArea, dictionary);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Inserts/Updates the changes to the database object (i.e. table) based on the entity state
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="tableTypeName">name of the table type passed as parameter</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>affected rows</returns>
1 reference
bool BulkInstertOrUpdate(string storedProcedureName, IDictionary<string, object> parameters, string tableTypeName, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userInfo", userInfo }
    };

    return query.BulkInstertOrUpdate(_usp_GetAllApplicationArea, dictionary, Udt_UserInfo);
}
```

How To – Consume LH Enterprise Data Access Library

How To – Consume Data Access Enterprise Library – IQueryRepository<T> interface

Signature

```
/// <summary>
/// Deletes the record in the database object(table)
/// </summary>
/// <param name="storedProcedureName">inline query or sp name</param>
/// <param name="parameters">dictionary collection with Key/Value (key - parameter name, value - value for that parameter) pair</param>
/// <param name="commandType">command type - Text or Stored Procedure, default is SP</param>
/// <returns>affected rows</returns>
1 reference
void Delete(string storedProcedureName, IDictionary<string, object> parameters, CommandType commandType = CommandType.StoredProcedure);
```

Example

```
using (IQueryRepository<UserApps> query = new QueryRepository<UserApps>(_connectionString))
{
    IDictionary<string, object> dictionary = new Dictionary<string, object>
    {
        { "userId", userId }
    };

    query.Delete(_usp_GetAllApplicationArea, dictionary);
}
```