d

Development Standards Document

Contents

[Document Purpose 2](#_Toc410383363)

[General Architecture 3](#_Toc410383364)

[System Layers 3](#_Toc410383365)

[Projects Structure 4](#_Toc410383366)

[Data Access Layer (DAL) 5](#_Toc410383367)

[DAL General Architecture 5](#_Toc410383368)

[DAL Definitions Project 5](#_Toc410383369)

[DAL Implementations Project 6](#_Toc410383370)

[DAL and SQL Server 7](#_Toc410383371)

[Business Layer 8](#_Toc410383372)

[Service Layer 9](#_Toc410383373)

[Runtime Services 10](#_Toc410383374)

[Caching 11](#_Toc410383375)

[Logging and Tracing 12](#_Toc410383376)

[Security 13](#_Toc410383377)

[Client UI 14](#_Toc410383378)

# Document Purpose

The purpose of this document is to ensure that the development team has one standard that they follow during the development process

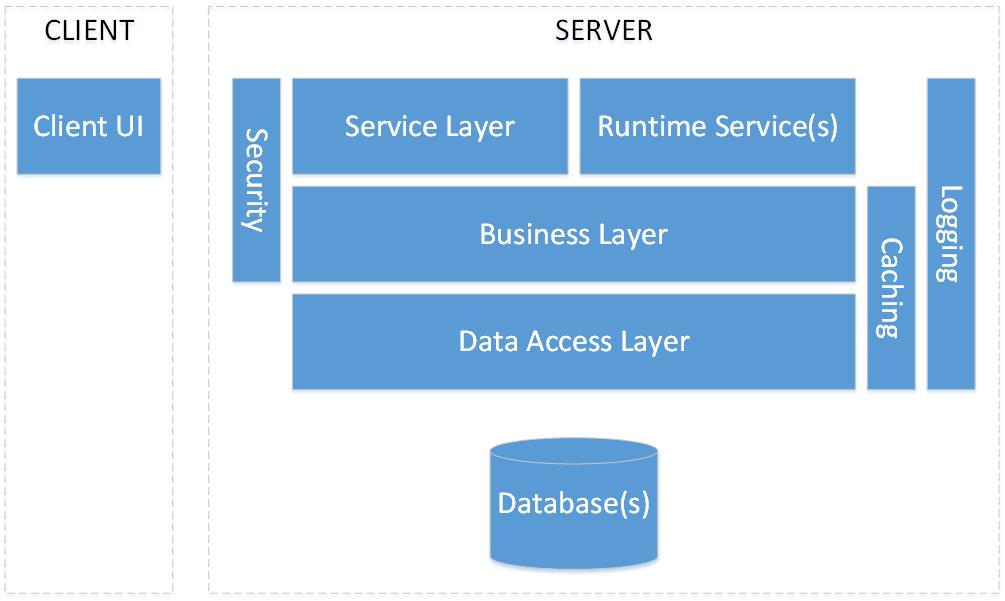
# General Architecture

## System Layers

We use the following methodologies in our development process:

* Two Tiers design: Client-Server
* 3 layers architecture: Data, Business, Service

The following diagram shows the different components that would be involved in the development process of any system



This table gives a brief description about each component:

|  |  |
| --- | --- |
| Component | Description |
| Database(s) | The data store of the system data (e.g. SQL, Oracle…) |
| Data Access Layer | This component is responsible for all kind of data access. It interacts with the Database(s) and provide needed operations to upper layer |
| Business Layer | All business logic in the system should be implemented in this layer |
| Service Layer | This layer provides a set of services (e.g. web services) that allows other components (e.g. Client UI) and other systems to interact with our system |
| Runtime Service(s) | This component consists of hosting the system services that need to be always running or run based on schedules. Those services would perform system jobs and process queued requests |
| Caching | This is a crosscutting component gives caching services to the Data and Business layers |
| Logging | This is a crosscutting component gives logging/tracing services to all other layers |
| Security | The security layer applies required security mechanism to the Service and Business layers. This includes Authentication, Authorization, Encryption/Decryption |
| Client UI | This consist of the User Interface component(s). this should interact with the system through the service layer only (after applying the security mechanism) |

The sections in this document describes the responsibilities and base implementations of the above components in further details

## Projects Structure

Each module in the system should have at least the following visual studio projects:

|  |  |
| --- | --- |
| Project | Description |
| Entities | * Example Name: TOne.LCR.Entities * It should include C# classes of all entities of the module * The classes should only have properties (no methods, no business logic) |
| Data Projects | * Example Names: TOne.LCR.Data and TOne.LCR.Data.SQL * It should include data managers that retrieve data from the data store and convert them to business entities; and that manipulate data in the data store |
| Business Project | * Example Name: TOne.LCR.Business * It should include business managers that manipulate the business entities of the module and apply business logic |

# Data Access Layer (DAL)

The Data Access Layer, from here on will be known as DAL, shall be responsible for all kind of accessing data in the data store. This includes retrieving and manipulating data. The DAL is the only component in the entire system who can deal with the data store and can be dependent on the format and structure of the data.

The following sections discuss this layer in details

## DAL General Architecture

To have a DAL component for a system module (e.g. LCR module), we shall create these visual studio projects:

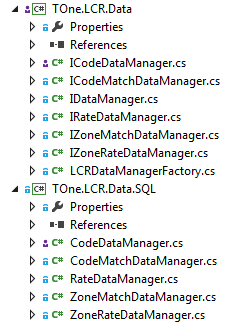
1. A Data Operations Definitions project
2. An implementation project for each database platform that we need to support (e.g. SQL, Oracle, MySQL…)

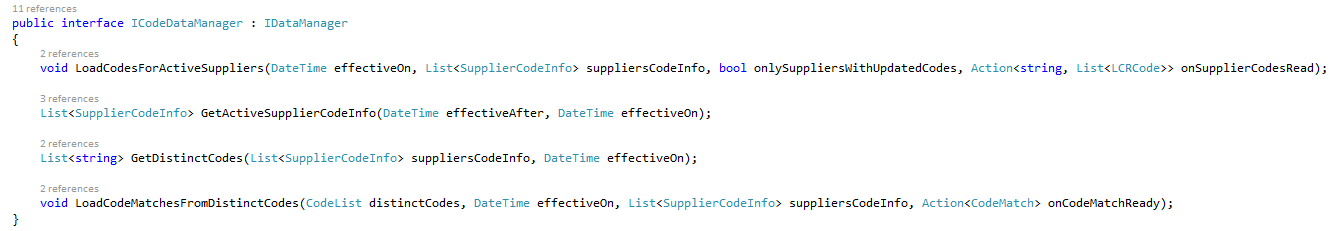


### DAL Definitions Project

The definitions project should have the name [*ModuleName*].Data (e.g. TOne.LCR.Data). In this project, we should create:

* A C# interface for each entity where the needed operations are defined
* A factory class with a Generic Static method that returns an instance of the request Data interface type. This instance would be a concrete implementation in one of the Implementations projects





namespace TOne.LCR.Data

{

public static class LCRDataManagerFactory

{

static ObjectFactory s\_objectFactory;

static LCRDataManagerFactory()

{

s\_objectFactory = new ObjectFactory(Assembly.Load("TOne.LCR.Data.SQL"));

}

public static T GetDataManager<T>() where T : class, IDataManager

{

return s\_objectFactory.CreateObjectFromType<T>();

}

}

}

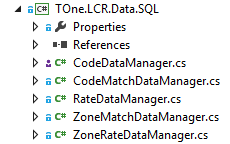
We should add references to the following framework libraries:

* C:\TFS\Vanrise\Code\Output\Vanrise.Common.dll

### DAL Implementations Project

For each database platform that we need to support in the system, we shall create an implementation project of the DAL. In this project, we should create the classes that implement the interfaces defined in the DAL Definitions project. The implementation should consists of access the target database in the proper way.

The name of this project should be: [*ModuleName*].Data.[*PlaformName*] (e.g. TOne.LCR.Data.SQL).



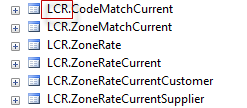


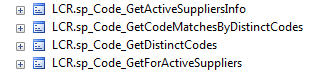
## DAL and SQL Server

### Database Design Convention

#### SQL Schema

1. We shall create an SQL schema for each module. This schema would group all database objects that belong to that module



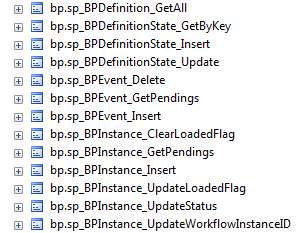


#### Table Conventions

1. The table name should be in singular (for example Route and NOT Routes)
2. Each table should have a Primary key

#### Queries Conventions

1. All queries (select, insert, update, delete) shall be defined as stored procedures and only stored procedures. Queries in code are not allowed. Exceptions to this point should be discussed and agreed on case by case
2. The stored procedures should not include any business logic. Exceptions to this point should be discussed and agreed on case by case
3. The stored procedure should have the following naming convention: [*Prefix*]\_[*EntityName*]\_[*Operation*][*AdditionalInfoIfNeeded*]
   1. [*Prefix*]: sp for general procedures, rp for reports procedures, bp for business process procedures
   2. [*EntityName*]: usually main table name against which the query is built
   3. [*Operation*]: Insert, Update, Delete, Get
   4. [*AdditionalInfoIfNeeded*]: Added to the operation in case it perform partial manipulation or it retrieves rows based on filter. For example:
      1. GetByID
      2. GetByFilter
      3. UpdateStatus

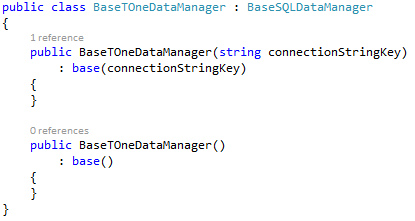


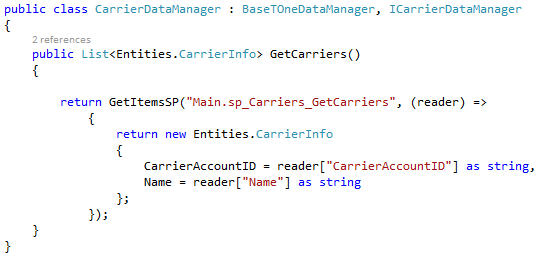
1. DO NOT use SELECT \*. We should always specify the column names
2. Whenever we have JOIN in the query, we need to use Table Alias and specify the alias with each column used in the query

### DAL SQL Implementation Project

In the DAL Implementation project against SQL server database, we shall:

1. Name it [*ModuleName*].Data.SQL
2. Add references to the framework libraries:
   1. C:\TFS\Vanrise\Code\Output\Vanrise.Data.dll
   2. C:\TFS\Vanrise\Code\Output\ Vanrise.Data.SQL.dll
3. Inherit each Data Manager class from the Vanrise.Data.SQL.BaseSQLDataManager





# Business Layer

# Service Layer

# Runtime Services

# Caching

# Logging and Tracing

# Security

# Client UI