# Introduction

Presented by: Arash Rahimian

DITeN

www.Diten.net



DITeN Framework

technical document

Version 1.0

October 22, 2016

Diten framework is a solution for enterprises in office automation. The solution is based on Ext.Net components and Microsoft ASPX. The database engine is used in this solution is Microsoft SQL Server and the data tire of the framework is based on ADO.Net. There are some futures in the framework that you must learn at the beginning. There are some important futures and laws that you must learn before beginning to programming with this platform. You can find all the technical information about this framework in this document. This is a technical document for who wants to program with this framework. You can find detailed information about classes, methods, and objects that are designed for use in this platform. There are some laws that a programmer who wants to develop applications based on this framework must obey them. For example, in the framework all entities must be introduced in entities table at first then you can deploy details of entity in other tables. In the first chapter, you can find some information about the design structure of the framework. In the second chapter, you can find some information about database structure and laws that you must obey in the development structure. In the third chapter, you can find some information about data layer design and you can find detailed information about data sets and data helpers. In the fourth chapter, you can find some information about registration and login processes. Chapter 4th is a very important chapter in analyzing application structure and system of authorization and authentication. In this chapter, you can find information about installing applications and structure of application setup. In chapter 5th, you can find detailed information about authentication and authorization of the system. In chapter 6th, you can find some information about classes and methods that are used in the framework. For example, you can find some information about security class and cryptography. In addition, you can find some information about text tools and so on. In chapter 7th, you can find information about desktop page. Desktop page is the most important page of the framework. All applications will be executed on the desktop page. In chapter 8th, you can read details about web controls namespace. In chapter 9th, you can read about some web controls like resource handler and embedded resources. All of the web controls and themes and resources are embedded. You must read all resources from embedded resources by using ***resource handler*** object.

**Arash Rahimian**

Saturday, January 7, 2017

Table of Contents

[Introduction 0](#_Toc471754393)

[Chapter 01 3](#_Toc471754394)

[Introduction to designe structure 4](#_Toc471754395)

[Dictionary 5](#_Toc471754396)

[multilingual 5](#_Toc471754397)

[translations 5](#_Toc471754398)

[domains 5](#_Toc471754399)

[Database Views 6](#_Toc471754400)

[Types 6](#_Toc471754401)

[Embedded resources 6](#_Toc471754402)

[important classes and futures 7](#_Toc471754403)

[UserControl, page and masterpage 7](#_Toc471754404)

[Application dictionary 7](#_Toc471754405)

[Encrypted URL 8](#_Toc471754406)

[Translate method 8](#_Toc471754407)

[Control panel 8](#_Toc471754408)

[Setup applications 8](#_Toc471754409)

[Installing applications 8](#_Toc471754410)

[Uninstalling application 9](#_Toc471754411)

[backup data 9](#_Toc471754412)

[dictionary 9](#_Toc471754413)

[Cultures 9](#_Toc471754414)

[Properties window 9](#_Toc471754415)

[general tab 10](#_Toc471754416)

[Security tab 10](#_Toc471754417)

[Previous versions 10](#_Toc471754418)

[Recycle bin 10](#_Toc471754419)

# Chapter 01

Design structure of DITeN Framework

# Introduction to design structure

In the DITeN framework, you have an object called an entity. All objects are entities in this framework. For example, when you want to introduce a commodity you must create an entity with the type of commodity at first.

An entity object, you must record some general information of entity like Title, Creation Date, and Last Modify Date, etc. The most important field of entities table is ***DomainID***. Domain ID will separate data from each other. All of the data from a domain will be stored under Domain ID. This ability is included because we need to separate the corporation’s data from each other. All information about a corporation will be stored under a domain id. Domain ID will be generated during registration and owner of domain will be administrator of the domain. In the future when a user of the domain creates an entity, it will be stored under the user domain. We have a property window that is accessible from everywhere that you have an entity. In this window, you can find detailed information about the entity. You have three tabs on this window. At first tab, you will find some information about the entity, like the type of entity, last modified date, creation date, description, and name of the entity. In the second tab, you will find security information about the entity. At this tab, you can set the access levels of users and groups or add and remove users or groups. By this ability, you can set the authorization level of the entity and you can set who can read, delete or update an entity. In the third tab, you can find the history of the entity. In the DITeN platform, each entity will have a history of changes that are made by users on the entity and you can restore changes if you have sufficient privilege. By restoring an entity, you will have entity data at date of *last modify the date* of the entity that you restored and the latest entity will be stored as history. History of the entities will remain in another database that is a copy of the main database. By this methodology, we can store the history of the entity forever. A table of entities will store much information. We talked about that each object in this framework will be stored as an entity in the entities table and details of the object will be stored in object-related tables. This structure is designed because we need to save history at first. In addition, we need to have the ability to recover deleted information.

We have a section that works like recycle bin of windows. Users can restore deleted data from this section. This is important because maybe users delete some information accidentally and therefore, they can restore data in a very simple way. Deleted data and history are two deferent solutions. In history, you will have a history of entities that are updated but in the recycle bin, you will have deleted data. Also, recycle bin data will be deleted after one month automatically because of data redundancy. History will be stored in a separated database but the deleted entity will be separated by ***Deleted field of entity table.***

In the DITeN framework, we have a table called dictionary. This is a very important table in entire of the platform. All string data will be stored in this table because of redundancy and performance, and its id will be stored in the destination table. This means for example when we create an entity, the *Title of entity* will be stored in the dictionary table and the ID of inserted word will be stored in entities table *TitleID* filed. This is because of the performance. When we want to make a select query over entities table on title field, we can get ID of the word from dictionary table then make queries over the numeric field *TitleID* in entities table. This ability will make our queries very fast because we will make query over the ***GUID*** filed instead of an *NVARCHAR (MAX)* field.

|  |
| --- |
| **IMPORTANT**: Because of using a dictionary structure when we want to update a field, we must insert a new record in the dictionary table. This is important because maybe there are some relations between a word and another object and because of that, we could not delete or update the word in the dictionary table. |

# Dictionary

Dictionary table is the most important table in this framework. *All* *Characteristic data will be stored in a dictionary table.* This means when we want to create an object (entity) details table, we must set characteristic fields as ***GUID*** and storing characteristic data into the dictionary table. This ability is designed because of data redundancy. DITeN framework is designed for serving on many enterprises and because of that, we will have a big data problem. In big data when we want to make a query over a huge number of records, we will need to make queries over a big table and we know making queries over the numeric records will be faster than characteristic records. Also at first, by this methodology, we can make very fast indexes over numeric fields and we need to make a characteristic index over one field in entire of the databases. This design structure will give us an ability to have separated databases and this means we can create a farm and cloud base system by serving from many database servers. In addition, we must update the characteristic index over one table and this will make our index updating progress very fast. In addition, in the dictionary table ***Text*** field is unique and because of that, we will not have a data redundancy problem over this table. Text field data is separate from the culture field and therefore, we can have repetitious of data in different cultures.

## multilingual

Diten framework is a multilingual structure. We have this ability for separating data in all languages. We have a table called ***Cultures*** and, in this table, we will be introducing new cultures. In the beginning, all users must select their language and the entire interface will be translated in the language that they selected at registration. In addition, they can change the language in the control panel. With this ability, we can serve services to other cultures. We will separate data in the dictionary table by using the ***CultureID*** field and this will make queries much faster over the dictionary table. This ability will let us have repetitious data in the dictionary table. This future is designed because there are many shared words in the language but the meanings may be deferent form each other. By separating words by culture, we can support the repetitious of the words in our database system. In the DITeN framework, we have a culture named ***SYSTEM CULTURE***. This culture will let us have a basic culture and then we can translate this basic culture into other languages. This basic culture will be translated to the destination culture and all control’s textbase properties must be set by this culture structure. This culture is based on ***PINGLISH*** (Persian language by English letters handwriting) and when we want to set text of *FiledLabel* property of a *TextField* object, we must type its information text in this language than by using ***Translate method*** of the web controls, pages or master pages we can translate the word into the user-selected culture.

## translations

This table is designed for translating cultures into each other. By this ability, we can translate system words[[1]](#footnote-1) into destination cultures. In addition, we can translate culture into another by referencing the same system words. The translate method will be using this table for translating system words into the user-selected culture.

# domains

Because of multi corporation's support of the DITeN Framework we have to select a domain at the beginning. We will separate each enterprise data by using domain ID. Each enterprise could have a domain name that is subdomain of ***Diten.Net*** basic domain or they can have an individual domain. This domain name must be unique and by this ability, we can register repetitious of user names in our database. User names will be unique in each domain but their registered email address must be unique in entire of the platform. This is because of the login problem. Users can log in by their registered email or by their registered user ID. For example, if an enterprise registered ***SampleDomain.Diten.Net*** in our database, all users under this subdomain will have this structure of user names [***UserName@SampleDomain.Diten.Net***](mailto:UserName@SampleDomain.Diten.Net) or they cloud have user under the registered domain of the corporation like this [***UserName@SampleDomain.com***](mailto:UserName@SampleDomain.com). This ability will let users that have single user name by deferent domains but ***email address must be unique in entire of the platform***. In addition, we will let users have an email address under entire of the platform. All users can make login by using their user name that is registered under the domain of corporation or they can use their registered email address for the login process. We will generate a portal for each enterprise under a subdomain of the diten.net domain or the registered domain of the corporation, and they can use this portal as their web site too. In the future, we will create an email address for users under their domain address.

# Database Views

Database views will be used for getting all readable information about entities. Also, the make relations between entity table and entity details table. Developers can use these views for making quick queries over the database. Because we need to get word ids before making queries over the database, it makes the development of stored procedures boring and therefore, we designed these views for quick queries, but developers must not use these views at all. They must get word id from the dictionary table then make queries over the tables. Using these views for making queries is to slow and therefore, developers must not use them for making queries in the finalized system. They must convert queries into **Normal Queries[[2]](#footnote-2)** at the publishing time.

# Types

We will store types of entities into the ***Types*** table. Indeed, we have an enumerator in ***Diten.EntityTypes*** that we will create a type for each entity type that we need in the framework in this enumerator and [Diten.Data.Helpers.Types.GetTypeId(EntityTypes type)] method will save a new type into the database automatically on the calling method. Types of entities is so important because we will separate entities in queries by types. By using this methodology, when we want to get an entity by Title field, we will send type of entity into query and the search result will be faster at first and at second we will get minimum result of the entities.

|  |
| --- |
| **REMINDED**: We will separate data by domain ID for each enterprise system and therefore the result of this methodology for repetitious of the entities will be at a minimum. In time, we will check the singularity of the entity manually. This methodology is designed because in some applications we can have the same title for same type entities and therefore, we cannot make single row selecting result queries on the Title property of the entities. |

# Embedded resources

All resources in the DITeN framework are stored as embedded resources. All web controls, web resources, java scripts, CSS classes; Icons, etc. are stored as embedded resources. There is an object called ***Resource Handler*** that handles all embedded resources. When you need an embedded resource, you must use this object for getting a resource URL. For example, if you want to use *Main.js* you must get its URL by resource handler control. Resource handler control has a property called *DoRender.* It is a Boolean property and when developer set it *True* the control will return *Theme* resources like scripts and CSS classes and render these resources into the page or control that use this control. You must set theme property of the Resource Handler control because it will use the name of them for getting embedded resources.

|  |
| --- |
| **ATTENTION:** You do not need to render resources by this control in web controls because, on the ***Desktop*** page, you have this control by calling render method and therefore you do not need to call them again. You will have access to the general and theme specialized resources like java scripts and CSS classes on time. You must set do render (on web users controls that need to use this control for getting URL of the embedded resources) to false. This is important because when you set this property to false, control will not render resources again. Only on standalone pages,[[3]](#footnote-3) you need to set ***DoRender*** property into the true. |

For loading embedded web controls, you must use the load method of the page or web user control. All pages will inherit ***Diten.Web.UI.Page*** class and all web controls will inherit ***Diten.Web.UI.UserControl*** class. These base classes have many methods that are useful for development. *Translate* and *LoadControl* methods are the two most important methods of these base classes. By the *Translate* method, you can translate a system word into the cultural word and by *LoadControl* method, you can load embedded web controls. For using embedded web controls, you must use the *LoadControl* control method because this method knows the path of embedded web controls. All resources are embedded in ***Diten.Web.UI*** DLL library. When you call an embedded resource, application will get that from this library.

# important classes and futures

In this section, you will read about the most important classes and you will find some information about these classes. These are base classes of web controls, pages, and master pages and they have some futures that improve .Net base classes. All of these classes are in ***BaseClasses.cs*** file and you can find the code of them in this file.

## User Control, page, and masterpage

These classes inherit ***System.Web.UI.UserControl, System.Web.UI.Page, System.Web.UI.MasterPage*** classes and improve their futures. You can find encryption and decryption methods, application dictionary, theme, load control and many more methods and properties in this class that we will survey their detailed information in chapter 6th. These classes are base on Web Controls, Pages and master Pages of DITeN Framework and all of these objects must be inherited from them. They know about the encryption key, the basic culture of the system, translating system words, and many more futures that we will use in the entire application. Also, they will control the authorization of users over web controls, pages, and master pages.

### Application dictionary

This future will give us a system dictionary. For improving performance and decreasing database requests, we are loading system dictionary of all supported cultures in an application variable and developers can access this information form ***ApplicationDictionary*** property of every page or web controls. By this ability, we do not need to make queries on database for each word that we want to translate it. This future will improve the performance of the database. Each time that a new word added to the database it will be refreshed and will load a new dictionary and therefore, we do not need to make a query on the database for new words. ***Translate*** method will check this dictionary at first then if it could not find the translation it will make queries over the database and update the application dictionary. This is stored in the application layer because in entire of the application we need to access this dictionary and because the information stored in this variable is general, we put it in an application layer variable. Indeed, we put a data table into this variable and we will make queries over this data table.

### Encrypted URL

All redirections into the pages must be handled by the ***Redirect*** method and all parameters will be encrypted in the method then they will be passed over the URL. This future is for the security of the application. By this future, the user will not be informed about the parameters that developer passed them in the URL. You can find this method in every page, web control and master page classes. Developers must use the redirect method of the control or page and they must not use the standard redirection method of .Net Framework. In addition, for getting a parameter form encrypted URL; developers must use ***QueryString*** property of the page or web control. This property will decrypt the URL automatically and return a *Dictionary<string, string>* and developer can get the parameter value by using the name of the parameter as the key of the dictionary that is returned by *QueryString* property.

### Translate method

This method is designed for getting translation of the system works automatically. Developer must use this method of the pages or web controls for translating FieldLabel of TextFields, Title of the grid columns, Text of the buttons, etc. At the *InitializeComponets* method of the all pages or web controls, developers must translate text labels or title of the objects. This method will translate automatically the system word into the user culture. System will translate all text base properties of the controls automatically and we do not need to translate each control properties in *InitializeComponets* component but when we need to make some specialized translation over the controls or texts, we need to use this method individually.

# application

Applications are those controls that used for opening entities. Each application will open a specified entity type. Indeed, entity type works like file extension in windows. In windows, each file extension will be related to an application and that application will open that file. In DITeN Framework we have entity types instead of file extension. Each application will be associated with a specified entity type and will open that type of entity.

## Setup Applications

When the development of the application is finished, developers must setting-up applications over the platform. This means they must add loading information of the applications into the desktop page and system will automatically setup applications over the entire platform.

|  |
| --- |
| **ATTENTION:** All applications must be set up before use. After setting-up application over the platform, the application will be available for all of the domains. |

# DTI Entities

DTI file (shortens of DITeN Installer file) is an XML file that contains application manifest. When the administrator executes this entity with explorer object application will be installed over the platform and will be accessible for all domains. Explorer object can open and read this file and by using information that is included in this manifest file, the explorer can setting-up application over the platform. These manifests will be stored in a DLL and explorer can read this information from Diten.Manifests.DLL.

# Control panel

When you are clicking on start menu, you can find a button named Settings. By clicking on this button, you can open the control panel. In this window, users can manage settings of the. This is the control center of the entire the platform. You can set the language, application settings, language settings and many more options in this window.

## Backup Data

Domain admins and backup operators can make a backup of data and download this backup from the server into the local client. The backup process will copy domain data into a temporary database then make backup from that database by SQL server backup ability then prepare the link of download and email it for the backup operators and domain admins. This link will be available about 24 hours and after that, these data will be deleted automatically from the server. The backup process will be scheduled on the server and will be put in the queue of the backup process. This backup data will be protected by a password and the metadata of the domain will be added to the backup as an XML file.

|  |
| --- |
| **ATTENTION:** This backup can be restored only on the DITeN Platform because only the platform engine will know the password of the compressed data. |

## Dictionary

***Platform Administrators*** and ***Translators’*** groups have full access on this page. On this page, they can translate system words into the cultures. This system dictionary will be loaded into the ***Application Dictionary*** general variable. All interface of the platform will be translated by using the entry of this page. You can delete or edit the translation in this page.

|  |
| --- |
| **ATTENTION:** Deleting words in the dictionary window will delete words in the database and because system words are, embedded texts the system words will be added automatically after visiting the container of the word. This means by deleting a word from the dictionary page, translation of that word in selected culture will be deleted from the database, not the system word. |

## Cultures

In this window, you can add a culture into the database. Only platform administrators can add a new culture into the platform. For example, if you want to add a new dictionary into the system, the platform administrator must add the culture of the new dictionary by using this page then you will have the dictionary of culture in the dictionary window.

# Properties window

In the entire of the platform when you have an entity, you can get properties of that entity by right click on the entity. For example, if you are in add/remove programs window you can get property of the program that is installed on the domain by right-clicking on the installed program or if you are in Accounting program you can access this window by right-clicking on the entity and then choosing Properties from the pop-up menu.

## general tab

In this tab, you will find some information about the entity. For example, you can find information about entity type, creation date, last modified date and description of the entity. This information is useful when you want to understand the details of the entity.

## Security tab

In this tab, you can set the authorization level of the entity. By this ability, you can design access of the domain groups or users to the entity. For example, when you are in an accounting application, you can set the authorization of the users to a commodity. You can set some users can edit it and some groups can delete or modify it or you can deny some users or groups access to the entity.

## Previous versions

In this tab, you can find previous versions of the entity. This ability is useful when you want to restore some version of the entity. For example, a user changed entity information and you want to undo this job. You can simply choose the version that you want by clicking on the restore button you can restore that version. This job will make a current version as history version and restore selected version.

|  |
| --- |
| **WARNING:** The history of an entity will remain about one month and the system will delete this information after that. This is because of data redundancy. |

There are differences between the Recycle bin and the previous version of the entity. Recycle bin will store deleted entities not history of entities. However, in previous version ability, you can find different versions of an entity. We will handle this future by adding the ***PID*** field into the Entities data table in the database. We can detect entities that are marked as history by this field. When you update an entity PID field of current entity will be set as ID of the new entity, the current entity will be marked as history entity, and Last Modify Date of the current entity will be set to now time. After that, a new entity will be saved into the database (You will find history-marked entities in previous versions tab, not the selected entity.)

# Recycle bin

This future is included because sometimes users maybe deleting data accidentally. By this ability, they can restore deleted data. In addition, this information will be deleted after a month because of data redundancy. We will handle this future by adding ***Deleted*** filed into the Entities data table. When the user deletes an entity, it will have marked as deleted and the Last Modify Date of the entity will be set as current date. Users can find deleted entities in the ***Recycle Bin*** folder and they can restore the entity from this menu by right-clicking over the selected entity and clicking on restore menu item. The entities that are stored in the recycle bin (marked as deleted) will not show in the entire the domain and you will find it only in recycle bin folder.

|  |
| --- |
| **ATTENTION:** Only those users that have permission to the deleted entity can restore the entity and restoring entity will not overwrite destination folder entities.  After a month, the entire data of the entity will be deleted from the database, even detail information, and information that is stored in related tables.  Some entities will not be restorable like applications and domains. |

# System Folders

There are some folders that are must important folder in Diten Platform that we describe them below.

## Cultures

This folder will contain cultures that introduce to the system. System will store words in these cultures. Folder of cultures is **[.\Doors\System\Cultures]**. All cultures will be stored in this folder. Extension (or type) of a culture is **[.cult]**. A culture must be unique in this folder. Each culture has two properties, **Name,** and **DisplayName**. For example, for Iranian Persian culture, we will have **Name=Fa-ir** and **DisplayName=Persian (Iran)**.

### System Culture

There is a culture in this system that is based culture of the system. All textbase properties of the controls in this framework will be stored as system culture. For example, in login dialog system will store label text of User Name in the database under System Culture. Then we will translate these words that are stored under this culture to other cultures. **(Name=sys-sys, DisplayName=System Culture)**

## Types

This folder will contain types that are registered in the system. The folder that will store these types is **[.\Doors\System\Types]**. Each type must be unique in the system. In another view, we can call these types as ***Extensions***. For example, a folder will be stored in The database as **[Folder Name].fldr**. We will introduce system registered extensions (Types) in the table below.

|  |  |  |
| --- | --- | --- |
| Name | Display Name | Description |
| .fldr | Folder | Will introduce folder type. |
| .usr | User | Will introduce user type. |
| .grp | Group | Will introduce group type. |
| .dom | Domain | Will introduce domain type. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

### Attention

Because types are base of entities, type of types is empty GUID.

1. System word is an idiom for PINGLISH words that used in structure of the system. A system word structure is like this ***[WORD]*** but for simplify we will convert simple words into system word in Translate method then we will make query over the database by using system word. You can find *ToSystemWord* function in *Diten.Conver* class. [↑](#footnote-ref-1)
2. Normal Queries is an idiom for those queries that get system words IDs from dictionary table and it will be used for DITeN roles designed stored procedures or queries. [↑](#footnote-ref-2)
3. Standalone pages are those pages that are under ***Diten.Web.UI.Pages*** namespace. [↑](#footnote-ref-3)