AppGene Help

# What is AppGene?

AppGene is an application that helps developers or users to build their applications quickly.

# Getting Start

AppGene can build a database application; the minimum inputs are database entities.

AppGene is a data model driving system

## Prerequisites

## First Entity

## Run It!

# For Beginners

## Entities

## Connection String

## Basic Entity Attributes

# For Middle Level Users

## Change Product Basic Information

## Entity / Model / Entity Model

An entity is a class that is used in the data layer. And a model is a class that is used in the presentation layer.

For the presentation layer functions, you have 3 choices: entity-entity, model-model, and mode-entity.

* Entity-entity mode.

It is the simplest mode; you only need to provide the entity to get the presentation work.

* model-entity mode

It is a strong mode, the limitation is that you must implement AppGene.Common.Core.IEntityModel and your business layer is using the entity class.

AppGene.Ui.Core.AbstractEditableModel<TModel, TEntity>

AppGene.Ui.Core.ModelProxy<TModel, TEntity>

AppGene.Ui.Core.BaseEditableEntityModel<TEntity>

In the model class, each property will be mapped to an UI element; you need to control the data relationship in your model class.

For example: In the birthday-age case, birthday is a date property in your entity class. In the relate presentation view, you want to display age information that is a computed/read only property.

class SampleEntity

{

...

datetime Birthday { get; set; }

...

}

SampleModel : SampleEntity

{

}

SampleModel: BaseEditableEntityModel<SampleEntity>

{

datetime Birthday

{

get

{

return GetEntity().Birthday;

}

{

if (SetPropertyValue("", value))

{

OnPropertyChanged("Age");

}

}

}

int Age

{

get

{

return Date.Now- base.Entity.Birthday;

}

}

}

* model-model mode

Similar as model-model mode, the only different is your business service is using the model class.

## Conventions

## Entity Configurations

## Key Interfaces

### System.ComponentModel.INotifyPropertyChanged

The interface supports data binding

### System.ComponentModel.IDataErrorInfo

### System.ComponentModel.IEditableObject

### AppGene.Common.Core.IEntityMode

## Key Attributes

# For Advance Users

## User Interface Styles

# Table Categories in AppGene

* Simple Table
  + 1 pk and 0..\* index
  + pk has 1 columns
  + Id + Code + Name
  + Id + Name
  + Id + \* (PK, editable)
* Simple Table with reference columns
  + reference key s only include 1 column
* Table: PK has multiple columns
  + some PK columns are reference another table.
* Tree Table
  + PK has multiple columns
* One record table
* PK is editable table
* History Table

Same as the original table, but has an additional column "HistoryId"

* CheckOut Table

Same as the original table, but has some checkout information, checkout user, checkout time.

We focus on 1 primary key + 1 unique key (optional) only.

* Has Primary Key
* Does Display Id
* Is Tree Table
* Special columns
  + timestamp
  + Is Deleted
    - Case 1: The column's data type is same as PK, the value is 0 for non-deleted records, and is same as Id for deleted records.
    - Case 2: The code column is indexed but not unique
    - Case 3: No code columns.
  + Create User
  + Create Time
  + Modify User
  + Modify Time
  + Modify Times
  + Disabled

The column indicates if the data can be referenced now.

* + Order

# Design Patterns in AppGene

# Terminology

* AppGene

AppGene is this product.

* Three layers architecture

AppGene is using three layers architecture; they are presentation layer, business layer and data layer.

* Presentation layer

The presentation layer is user interface layer; the layer provides interactive functions with end users, and works with the business layer.

* Business layer

The business layer is business logical layer; the layer provides functions to the presentation layer, and invokes the data layer

* Data layer

The data layer is data access layer, and provides data access functions to the business layer. The layer controls data connection, transactions.

* Entity

An entity is a class that is used in your data model, in most case the entity represents a database table, and entity classes may be shared in all layers as data transition.

* Column Property
* Field Property
* Entity Model / Business Model / Business Entity

Model is a concept on the presentation layer, which can help the presentation objects to understand the function better

There are 2 situations your entity and model are same one.

Situation 1: Your business service uses instances of model.

Situation2: Your business service uses instances of entity, and the entity class provides enough information to get the presentation layer to use it.

* Model
* MVVM
* View
* View Model
* View Controller
* View Constructor

# Licenses