1. Create Database objects
   1. Create schema (optional)

If there is no schema for business component in database, create the schema.

Ex. CREATE SCHEMA Configuration

* 1. Create table
     + 1. Create related tables in database/related component folder/Tables in visual studio and publish.
       2. Naming convention: <Businessunit>.<component> (For Database, in VS only TableName.sql)
  2. Create view (optional)
     1. Create related views in database/related component folder/Views in visual studio and publish.
     2. Naming convention: <Businessunit>.<component> (For Database, in VS only ViewName.sql)
  3. Create stored procedure
     1. Create basic 5 “CRRUD” procedures (Create, Read, ReadAll, Update and Delete) and publish
     2. Path : database/related component folder/Stored Procedures in VS
     3. Naming Convention : <Businessunit>.<component>Insert/Read/ReadAll/Update/Delete

Ex: “CREATE PROCEDURE [Configuration].[ReligionInsert]

(

@Name Varchar(50),

@Id Numeric(10,0) OUTPUT

)

AS

BEGIN

INSERT INTO Configuration.[Religion](Name)

VALUES(@Name)

SET @Id = @@IDENTITY

END”

1. Create components
   * 1. Add Data.cs, Dao.cs, Validator.cs and Server.cs files under component (For Configuration Components create a folder for component and add cs files under the folder)

#### Data.cs :

Use System namespace.

Define the class as Public and Inherit from BinAff.Core.Data.

Declare the Entities and add get and set accessors to them.

#### Dao.cs :

Use System and System.Data namespace.

Define the class as Public and inherit from BinAff.Core.Dao

Add Constructor for Dao and base with “data” as parameter.

Add protected override method Compose and add CRRUD procedures.

Ex. base.CreateStoredProcedure = Configuration.ReligionInsert";

Add method to convert data to object as example,

ex. protected override BinAff.Core.Data CreateDataObject(DataRow dr, BinAff.Core.Data data)

{

Data dt = data as Data;

dt.Id = Convert.IsDBNull(dr["Id"]) ? 0 : Convert.ToInt64(dr["Id"]);

dt.Name = Convert.IsDBNull(dr["Name"]) ? String.Empty : Convert.ToString(dr["Name"])

return dt;

}

Add method to assign parameter as given in example

Ex. protected override void AssignParameter(String procedureName)

{

base.AddInParameter("@Name", DbType.String, (this.Data as Data).Name);

}

#### Validator.cs :

Use System.Collections.Generic namespace.

Use BinAff.Core and BinAff.Utility namespaces.

Define the class as public and inherit from BinAff.Core.Validator

Add constructor for Validator with parameter “Data data” and base with “data”

Add Protected override for List<BinAff.Core.Message> Validate() as given in example

Ex. protected override List<BinAff.Core.Message> Validate()

{

List<Message> msg = new List<Message>();

Data data = base.Data as Data;

if (ValidationRule.IsNullOrEmpty(data.Name))

{

msg.Add(new Message("Religion name cannot be empty.", Message.Type.Error));

}

return msg;

}

#### Server.cs :

No need to use any other namespace.

Define the class as public and inherit from BinAff.Core.Crud

Add constructor for Server with parameter “Data data” and base with “data”

Add method compose and define component, Validator and Data access

Ex. protected override void Compose()

{

base.Name = "Religion";

base.Validator = new Validator(this.Data as Data);

base.DataAccess = new Dao(this.Data as Data);

}

Override methods BinAff.Core.Data CreateDataObject() and BinAff.Core.Crud CreateInstance(BinAff.Core.Data data) as given in example.

Ex. public override BinAff.Core.Data CreateDataObject()

{

return new Data();

}

public override BinAff.Core.Crud CreateInstance(BinAff.Core.Data data)

{

return new Server(data as Data);

}

1. Create user interface
2. Create façade