CandySql

CandySql is an ORM based on the ORM library Candy. It's an official Candy implementation and the implementation of Candy within **CandySql** should be the way Candy is implemented.

The basic setup for an implementation of **CandySql** is the following:

Dal*.cs

The Dal folder should contain dal files to manage all sql queries. Dal classes should always be declared as static. There is no specific design for how methods in the Dal classes should be implemented, however it's recommended to only declare methods that just executes sql queries
and returns the result. There is one requirements for the implementation however; all methods must use the respective methods declared in **SqlDalHelper** to get either a single result or many results. However for inserts, updates and deletes it's recommended to use the model's implementations (***See CandySql – Models : Create, Update, Delete***.)

Models*.cs

The models folder should contain model files which are based on **Candy.SqlModel**.

All code found in this pdf can be found in the CandyTest project.

CandySql – Models

Model classes are required to have a **DataEntryAttribute** which takes two properties **Name** and **EntryPoint**. It's usually essential for all models used by Candy to be declared with the attribute.

For SqlModel the properties have the following meaning.

Name

The name of the table associated with the model.

EntryPoint

The connection string associated with the table's database.

Note: *It's recommended to store a constant with the connection string, so you wouldn't need to declare it for every model.*

```
[DataEntry(Name = "Employees", EntryPoint = Helpers.SqlTests.ConnectionString)]
public class Employee : SqlModel<Employee>
{
}
```

Next after creating the entry point of the model then you have to declare the associated fields with the model.

There is only one property that's required and it's the id property. The id property must be declared with a **DataSpecialTypeAttribute** and it's usually recommended that it's declared with a **DataIgnoreAttribute** as well, where the **IgnoreType** is **Write**.

```
[DataSpecialType(DataType = SpecialDataType.Id)]
[DataIgnore(IgnoreType = DataIgnoreType.Write)]
public int Id { get; set; }
```

Fields that are common during reads and writes does not require any attributes and can just be declared as plain properties with get; set;

```
public int OrgId { get; set; }
public string FirstName { get; set; }
public string MiddleName { get; set; }
public string LastName { get; set; }
```

If a field name does not match a table name (Even if the table name isn't case-sensetive.) then you can use the **DataMemberInfoAttribute** to declare the table name, where the **Name** property is the table name.

```
[DataMemberInfo(Name = "SSN")]
public string SocialSecurityNumber { get; set; }
```

A model can contain fields that does not exist in the table too, but these must be declared with the **DataIgnoreAttribute**. You can choose to set the IgnoreType, if you want it to be ignored during reads only or writes only. If the IgnoreType isn't set then it defaults to both reads and writes.

```
[DataIgnore()]
public int IgnoreProperty { get; set; }
```

A property that is considered a timestamp can be declared with the **DataSpecialTypeAttribute** where the **DataType** property is set to **Timestamp**. This will make sure it's always updated to DateTime.UtcNow and will always be set as an update parameter during updates.

```
[DataSpecialType(DataType = SpecialDataType.Timestamp)]
public DateTime? Timestamp { get; set; }
```

Note: *You're not required to have all columns existing in the table declared. You may only declare the rows you're using.*

All models come with a **Create**, **Update** and **Delete** method.

Create

Inserts the model into a row in the database and retrieves its id.

Update

Updates the row associated with the model.

Delete

Deletes the row associated with the model.

Important: If you update the id of a model, make sure to call the **UpdateIdProperty** method. Otherwise reads and writes may be invalid due to the id property being cached.