# OpenStore Data Access

## Introduction

The OpenStore SQL architecture is very different to many SQL server solutions.

https://github.com/openstore-ecommerce/Developer-Documents/blob/master/SQL-Architecture.pdf

The reason for this unusual architecture in because of the generic way OpenStore deals with data. Very often data based systems start design with the database architecture, and if your using SQL Server the obvious way to build your data is in a Relational Database Structure.

However, OpenStore breaks this general concept. We started by thinking:

**How can we make a generic data structure, that can be easily coded using c#?**

The answer was almost instant, XML. But using XML would make the traditional database design different, and hence we have the SQL database tables we now have.

## The Data Class & NBrightData

In OpenStore we mimic the data structure, like many coding practices. The difference with OpenStore is that we only have 1 base data class, to match the 1 data table. (NBrightBuy).

The data class is called **NBrightInfo** and you'll see it used throughout the OpenStore code and plugin code.

The NBrightInfo class is defined in the NBrightData project, which is the extension used by OpenStore for the templating system. In fact, this dependency project has Interfaces to DNN, Data access, Template rendering and Template filesystem management.

The idea of the NBrightDNN project is to keep OpenStore as independent as possible to DNN, this is a little controversial because many DNN programmer believe that all code should match DNN standards, but because of the complexity of OpenStore it was felt a level of independence should be maintained. This protects OpenStore from changes in DNN, but could also will allow code to be ported to another CMS if required. Although the NBrightDNN interface rule has not been strictly followed, so porting to another CMS is not something anyone would want to undertake unless we have no other option.

## NBrightInfo class

The NBrightInfo class is a quite a heavy structure and includes a XML document. This allows direct manipulation of data using this base class.

The public interfaces are simple and match the data table NBrightBuy.

However, that are a number of interfaces/method that make this class very useful for manipulating data in an XML format.

GetXmlNode : Get data from a single XML node.

RemoveXmlNode : Remove an XML node.

AddSingleNode : Add an XML node.

IsValidXmlString : Validate XML format.

AddXmlNode : Add a XML string structure as a node.

ReplaceXmlNode : Replace an XML string as a node.

GetXmlPropertyInt : Get data as an Integer.

GetXmlPropertyDouble : Get data as a Double.

GetXmlPropertyBool : Get data as a Boolean.

GetXmlProperty : Get data as a String. (Format the output string to match the localization)

GetXmlPropertyRaw : Get data as a string, but in the raw format of the XML.

AppendToXmlProperty : Append to data node.

SetXmlPropertyDouble: Set data value as Double.

SetXmlProperty : Set data as string.

ToXmlItem : Returns the entire data class as an XML string.

FromXmlItem : Populates the class with XML data taken from the "ToXmlItem" method.

Dictionary : Returns all XML nodes as a dictionary list.

AddToDictionary : Add a value to a dictionary list.

UpdateAjax : Takes the Ajax return from the client and populates the class.

AddXref : Add data node under root. (Obsolete)

RemoveXref : Remove data node under root. (Obsolete)

GetXrefList : Get data node as string. (Obsolete)

ValidateXmlFormat : Validate XML data structure.

Clone : Clone the class.

Also, one of the public interfaces is "XMLDoc" which is a XmlDocument and can also be used to extract data. To populate this XmlDocument you only need assign an XML string to "XMLData".

## XML Format

The XML format of data is structure to always have a root node of <genxml>. It the has sub-nodes of "<hidden>","<textbox>","checkbox","checkboxlist","radiobuttonlist","<dropdownlist>".

Apart from the root node, the node structure can use anything you wish. The sub-nodes are only convention to try and make large XML a little more understanding to read.