HeaderData

# Introduction

The HeaderData class is Server-side code. It keeps track of variables required for sorting, paging, filtering and other information if required. The idea is to persist this kind of data across web pages for each web user.

# How it works

When a HeaderData class is created a new Guid is created called “HeaderCacheKey”.

The HeaderData Class is automatically cached into memory using this HeaderCacheKey.

The cache groupid of “headerdata” is used.

### Saving the HeaderData to cache

Saving to cache is automatically done when the class is initiated.

### Retrieving the HeaderData

To get the HeaderData from Cache we use the “HeaderCacheKey” which was created when first initialized.

The must be passed to the HeaderData class using the parameter Info class, which was passed from the webpage.

Therefore, to get the HeaderData out of cache, we must have the HeaderCacheKey rendered on the html page and placed into a simplisity paramField.

Example:

// send headerchaekey use, so server side can get data.

simplisity\_setParamField('headercachekey', '@(articleDataList.Header.HeaderCacheKey)');

*This could have been done with a cookie, but cross site calls can be awkward with cookies, so we have decided to make the method a field on the html page and reply with simplisity params.*

In the above example we place the “HeaderCacheKey” into a “headercachekey” field. All param fields are passed back to the server from the client and used to populate the SimplisityInfo class paramInfo. We can then use this class to populate the HeaderData.

### What happens when HeaderData is initiated

On initialization of the HeaderData we pass the paramInfo which came from the client on the API.

Header = new HeaderData(paramInfo);

The paramInfo will contain the HeaderCacheKey that we rendered on the page, the HeaderData class will use that field (“genxml/hidden/headercachekey”) to get the HeaderData from cache.

But this is only the first step, to be useful we must change the values storage and again save to memory.

The HeaderData class reads all the data fields in the paramInfo class, it uses this to overwrite the existing fields. Any field name/id can be used, but we have static defined ones, to deal with standard operations.

### Standard HeaderData Fields

string HeaderCacheKey = Info.GUIDKey

//Order by

string OrderBySQL = Info.GetXmlProperty("genxml/hidden/orderbysql")

int SortActivate = Info.GetXmlPropertyInt("genxml/hidden/sortorderactivate")

// Paging

int PageSize = Info.GetXmlPropertyInt("genxml/hidden/pagesize")

int Page = Info.GetXmlPropertyInt("genxml/hidden/page")

int RowCount = Info.GetXmlPropertyInt("genxml/hidden/rowcount")

//Filter SQL

string FilterSQLref = Info.GetXmlProperty("genxml/hidden/filtersqlref")

After the values in the header have been updated by the paramInfo, the HeaderData will save itself to cache.

# Filtering and Sorting Data Lists

The HeaderData is also used to keep track of filtering and sorting data. Each DNN Rocket module can have settings to define the filtering and sorting.