REMARQ API

Accessing the RemarQ 1.X .NET API

Abstract

This document will describe the RemarQ 1.X enterprise API and how you can use that API for integrating your applications with RemarQ

Robert Ginsburg

support@balsamicsolutions.com

Prerequisites

The RemarQ API (application programming interface) is available to any customer who is licensed for the Enterprise edition of RemarQ. It will function only on a fully configured SharePoint server that is a member of an Enterprise licensed RemarQ farm. It will work with the evaluation edition as long as the evaluation edition is not expired. RemarQ is compiled in C# and targets the .NET Framework 4.5.

API References

Any .NET supported project type can make use of the RemarQ API. You must have a reference to the currently active *BalsamicSolutions.ReadUnreadSiteColumn* assembly. This assembly will be in the Global Assembly Cache of all of your SharePoint servers. RemarQ uses assembly versioning, this means that your app must also track the current RemarQ version. If you are writing web parts or SharePoint timer jobs this is taken care of for you, if not you should copy the appropriate dependent assembly section from one of your SharePoint site's web.config files to your application's app.config file.

Security

RemarQ content is stored in an independent SQL database. Access to it is managed by your configuration. If you configured your SQL access to use integrated security, then the Windows process that hosts your API application must have DBO permissions to the database. This is described in more detail in the FAQs section of http://www.remarq.info.

API Access

With your application configured the class you need to get access to the RemarQ API is *BalsamicSolutions.ReadUnreadSiteColumn.ApiAccess*. It is a public static class and all of its members are static. The individual methods are described below:

```
public static string LicenseMode()
```
This method returns a string that describes the current licensing mode of the system. It is the only method that can be called without an Enterprise license. The string includes the text name of license type and the number of servers that are licensed

```
public static bool IsRead(SPUser userId, SPListItem listItem)
```
This method returns a true if the item passed in has been read by the user passed in. Otherwise it returns false

```
public static bool MarkRead(SPUser userId, SPListItem listItem)
```
This method returns true if it successfully marks the item identified in listItem as read for the user identified by userId.

```
public static bool MarkUnread(SPUser userId, SPListItem listItem)
```
This method returns true if it successfully clears the read marks for the item identified in listItem as read for the user identified by userId.

```
public static IEnumerable<int> ReadMarks(SPUser userId, SPList sharePointList)
```
This method returns a list of item Ids contained by the sharePointList that have been read by the user identified by userId

```
public static IEnumerable<ReportData> GetReport(SPListItem listItem)
```

This method returns a specialized collection of ReportData items that describe the read history of the listItem parameter. The ReportData Class source is:

```
public class ReportData
{
        public string Id { get; set; }

        public int Version { get; set; }

        public DateTime ReadOn { get; set; }

        public int UserId { get; set; }

        public string ReadBy { get; set; }

        public string ReadByEmail { get; set; }

        public ReportData()
        {
                this.Id = Guid.NewGuid().ToString("B");
        }
}
```