Newsletter @ Nybble v1.0

Requirements

- ASP.Net 2.0 Web Site
- ASP.Net Membership and Roles

Functionality Included

The Newsletter @ Nybble allows for custom newsletter preparation.

We may include in the functionality the following points:

- Send automatic newsletters on hourly, daily, weekly, monthly or manual based.
- Allow to generate multiple campaigns.
- Users can subscribe or unsubscribe to campaigns.
- You could define campaigns for all users also.
- Allows sending fixed saved HTML newsletters.
- Allows sending process driven and generated newsletters through extension.
- Can manage multiple Membership applications
- Tracking
- Works as a system service, easy to deploy

Database Creation

You need to execute the script located at P:\Nybble\Newsletter \dbcreation1.0.sql on your database. This will create all necessary tables for the system.

Basic Configuration

You must create at minimum one campaign for the system to work. Let's clarify the fields (bold indicates required):

- **CampaignID:** Automatic
- **Name:** Descriptive Name
- **ApplicationName:** Indicates for which Membership application is valid. It should match the applicationName defined on the web.config file for your web site.
- **Type:** Indicates the type of Campaign
    - 1 indicates that request Subscription.
    - 2 indicates that includes all users.
- **Status:** Indicates the current status of the campaign
    - 1 indicates it is enabled.
    - 0 indicates it is disabled.

- StartDate/EndDate: You may define an schedule for the campaign to be valid.
- **Frequency:** Indicates how often the campaign should be executed
    - 1 indicates Daily.
    - 2 indicates Weekly.
    - 3 indicates Monthly.
    - 4 indicates hour based.
    - 5 indicates manual.
- FixedNewsletterID: Relation to a fixed HTML newsletter to be sent.
- DynamicCode: Indicates a class that creates the HTML piece code to be sent. Newsletter @ Nybble will dynamically try to load this class and determine which interface correspond. In case it is an IPersonalizedNewsletter interface based class, it will request N amount of times (per N subscriptions).
    - This class must implement one of the following interfaces IDynamicNewsletter or IPersonalizedNewsletter (Refer Dynamic Newsletters)

Example of Loading

```
set xact_abort on
go

begin transaction
go

set identity_insert dbo.camCampaigns on
go

INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (1, N'Argentina Daily', N'fppretail', 1, 1, NULL, NULL, 1, N'ARDAILY', NULL,
N'FundProRetail.Newsletter.NewsletterArgentinaDaily, FundProRetail.Newsletter')
INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (2, N'Argentina Weekly', N'fppretail', 1, 1, NULL, NULL, 2, N'ARWEEKLY', NULL,
N'FundProRetail.Newsletter.NewsletterArgentina, FundProRetail.Newsletter')
INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (4, N'Chile Daily', N'fppretail', 1, 1, NULL, NULL, 1, N'CHDAILY', NULL,
N'FundProRetail.Newsletter.NewsletterChileDaily, FundProRetail.Newsletter')
INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (5, N'Chile Weekly', N'fppretail', 1, 1, NULL, NULL, 2, N'CHWEEKLY', NULL,
N'FundProRetail.Newsletter.NewsletterChile, FundProRetail.Newsletter')
INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (7, N'Mexico Daily', N'fppretail', 1, 1, NULL, NULL, 1, N'MXDAILY', NULL,
N'FundProRetail.Newsletter.NewsletterMexicoDaily, FundProRetail.Newsletter')
INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (8, N'Mexico Weekly', N'fppretail', 1, 1, NULL, NULL, 2, N'MXWEEKLY', NULL,
N'FundProRetail.Newsletter.NewsletterMexico, FundProRetail.Newsletter')
INSERT INTO dbo.camCampaigns
  (CampaignID, [Name], ApplicationName, Type, Status, StartDate, EndDate, Frequency, Code, FixedNewsletterID, DynamicCode)
  VALUES (10, N'Portfolio Alerts', N'fppretail', 2, 1, NULL, NULL, 1, N'PORTALERTDAILY', NULL,
N'FundProRetail.Newsletter.PortfolioAlerts, FundProRetail.Newsletter')
go

set identity_insert dbo.camCampaigns off
go

commit transaction
go
```

You must also define the available frequencies per application name:

```
set xact_abort on
go
```

```
begin transaction
go

set identity_insert dbo.camFrequencies on
go

INSERT INTO dbo.camFrequencies
  (FrequencyID, FrequencyCode, ApplicationName)
  VALUES (1, 1, N'fppretail')
INSERT INTO dbo.camFrequencies
  (FrequencyID, FrequencyCode, ApplicationName)
  VALUES (2, 2, N'fppretail')
INSERT INTO dbo.camFrequencies
  (FrequencyID, FrequencyCode, ApplicationName)
  VALUES (3, 4, N'fppretail')
go

set identity_insert dbo.camFrequencies off
go

commit transaction
go
```

This should let the system ready to work.

Event Logging

You need to create a new source for the event logging to work correctly in the server and testing machines.

```
// Create the source, if it does not already exist.
if (!EventLog.SourceExists("StockForecast"))
{
    EventLog.CreateEventSource("StockForecast", "StockForecast");
}
```

Dynamic Newsletters

IDynamicNewsletter Interface

We will use this interface when we need to create one unique newsletter for all subscribed users to a campaign.

Example of Usage

```
public class NewsletterArgentina : BaseNewsletter, IDynamicNewsletter
{
        private readonly string title = HttpUtility.HtmlEncode( "Newsletter Semanal de Fund Pro Performance" );
        private readonly string bannerSource = "cou1";

        public MailMessage Get(string templatePath, DateTime? lastRunDate)
        {
  IList<News> nlist = controllerFactory.News().GetForNewsletterWeekly((int)enmCountries.Argentina, 0,
enmNewsCategory.News);
                if( nlist.Count == 0 )
                        return null;

                // Determine if is it a full path
                if (Path.IsPathRooted(templatePath))
                {
```

```
                                        // Determine if we are running in a web context
                                        if( HttpContext.Current != null )
                                                templatePath = HttpContext.Current.Server.MapPath( templatePath );
                                        else
                                                templatePath =
Path.Combine(Path.GetDirectoryName(Assembly.GetCallingAssembly().Location), templatePath);
                                }

                                string fullMail = File.ReadAllText( Path.Combine( templatePath , "newsletter.htm" ) );
                                string newsTemplate = File.ReadAllText(Path.Combine(templatePath, "noticia.htm"));

                                string newsInfo = "";
                                string body = "";
                                string subject = HttpUtility.HtmlDecode( title );

                                foreach (News n in nlist)
                                {
                                        newsInfo = newsTemplate;
                                        newsInfo = newsInfo.Replace("[TITLE]", n.Title);
                                        newsInfo = newsInfo.Replace( "[BRIEF]" , n.Brief);
                                        newsInfo = newsInfo.Replace("[LINK]", SiteUrl +  n.Url);
                                        body += newsInfo;
                                }

                                body += File.ReadAllText(Path.Combine(templatePath,
"noticia_banner.htm")).Replace("BANNER_SOURCE", bannerSource);

                                fullMail = fullMail.Replace("[BODY]", body);
                                fullMail = fullMail.Replace( "[TITLE]" , title );

                                MailMessage m = new MailMessage();
                                m.Body = fullMail;
                                m.Subject = subject;
                                return m;
                        }
                }
```

IPersonalizedNewsletter Interface

We will use this interface when we need to create a customized newsletter with member related information.

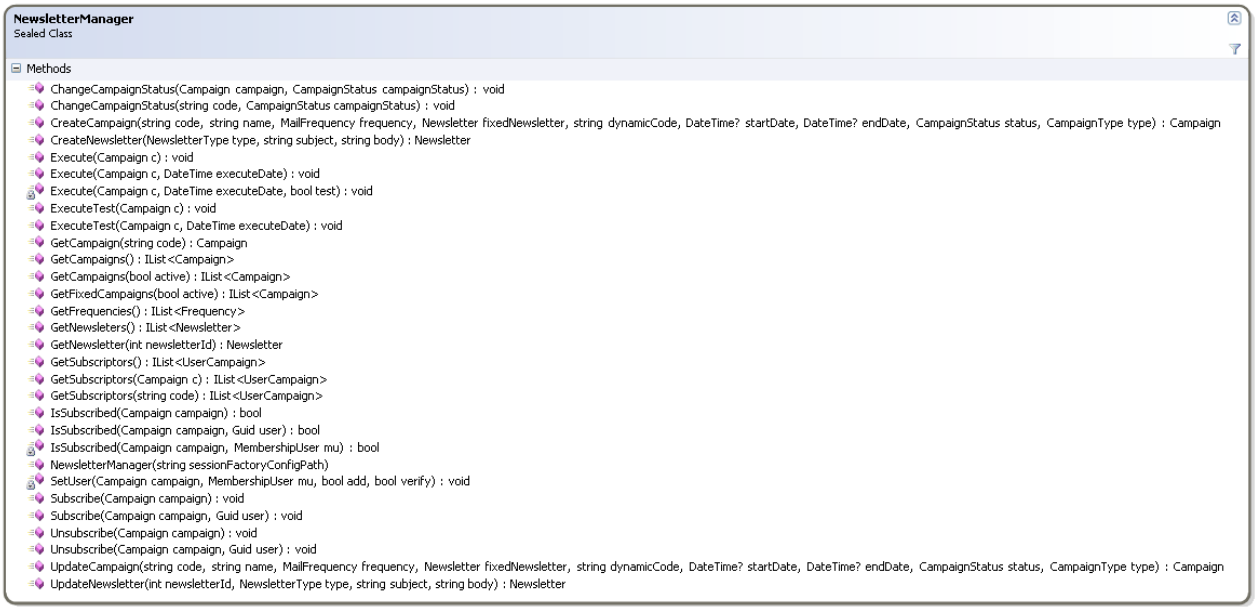
Example of Usage

```
        public class PortfolioAlerts : BaseNewsletter, IPersonalizedNewsletter
        {
                #region IPersonalizedNewsletter Members

                public System.Net.Mail.MailMessage Get(System.Web.Security.MembershipUser mu, string templatePath,
DateTime? lastRunDate)
                {
          IList<News> nlist = controllerFactory.News().GetForAlert((Guid)mu.ProviderUserKey, enmNewsCategory.News);
                        if( nlist.Count == 0 )
                                return null;

                        if (Path.IsPathRooted(templatePath))
                        {
                                // Determine if we are running in a web context
                                if (HttpContext.Current != null)
                                        templatePath = HttpContext.Current.Server.MapPath(templatePath);
                                else
                                        templatePath =
Path.Combine(Path.GetDirectoryName(Assembly.GetCallingAssembly().Location), templatePath);
                        }

                        string fullMail = File.ReadAllText(Path.Combine(templatePath, "alert.htm"));
                        string newsTemplate = File.ReadAllText(Path.Combine(templatePath, "noticia.htm"));

                        string newsInfo = "";
                        string body = "";
                        string subject = "Alerta de noticias en su portafolio de inversión";
```

```
                    foreach (News n in nlist)
                    {
                              newsInfo = newsTemplate;
                              newsInfo = newsInfo.Replace("[TITLE]", n.Title);
                              newsInfo = newsInfo.Replace( "[BRIEF]" , n.Brief );
                              newsInfo = newsInfo.Replace( "[LINK]" , SiteUrl + n.Url );
                              body += newsInfo;
                    }

                    body += File.ReadAllText(Path.Combine(templatePath, "noticia_banner_alert.htm"));
                    fullMail = fullMail.Replace("[BODY]", body);
                    fullMail = fullMail.Replace("[TITLE]", subject);

                    MailMessage m = new MailMessage();
                    m.Body = fullMail;
                    m.Subject = subject;
                    return m;
          }

          #endregion
}
```

Newsletter API

The Newsletter @ Nybble exposes a centralized component with all necessary methods:



Check the most common used methods once the campaigns are defined:

- Subscribe(Campaign) : Subscribes current user
- Unsubscribe(Campaign): Unsubscribe current user
- GetCampaign(Code): Obtain the campaign by code.

Configure Newsletter Manager on WebCastle & NHibernate

You need to add a new component to be available:

```
                <component  id="newsletterManager"
                                        type="ProjectBase.Newsletter.NewsletterManager, ProjectBase.Newsletter"
                                        service="ProjectBase.Newsletter.NewsletterManager,
ProjectBase.Newsletter">
                        <parameters>
                                <sessionFactoryConfigPath>~/WebNHibernate.config</sessionFactoryConfigPath>
                        </parameters>
                </component>
```

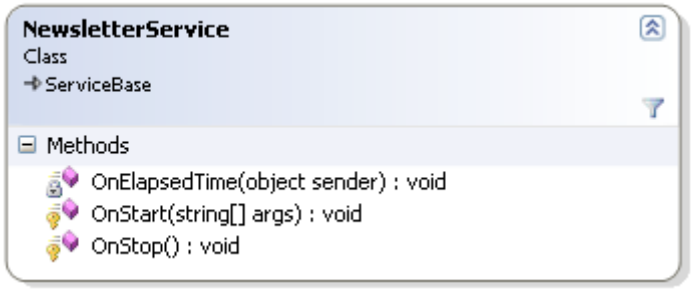
You also need to add the assembly to the NHibernate configuration file:

```
                <mapping assembly="ProjectBase.Newsletter" />
```

Service Creation

Because we may want to have multiple Newsletter @ Nybbles on the same server, you need to create your own service to execute it on the server.

You should create a normal Windows Service but it should inherit from NewsletterService instead of ServiceBase.



Review the included Service SourceCode on P:\Nybble\ Newsletter\Service SourceCode\ for further information.

Service Configuration

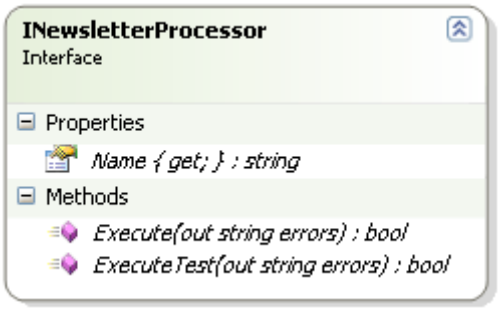
Above all common .NET configuration files and NHibernate, you need to configure the correct NewsletterProcessor on Castle configuration.

```
<component id="fundproProcessor"
   type="ProjectBase.Newsletter.NewsletterProcessor, ProjectBase.Newsletter"
   service="ProjectBase.Newsletter.INewsletterProcessor, ProjectBase.Newsletter">
          <parameters>
                    <SessionFactoryConfigPath>NHibernate.config</SessionFactoryConfigPath>

<TemplatePath>C:\Projects\LatinAssetManagement\FundProRetail\FundProRetail.WebSite\res\mail\</TemplatePath>
                    <TimesToExecute>
                              <array>
                                        <item>13:40</item>
                              </array>
                    </TimesToExecute>
                    <DaysToExecute>
                              <array>
                                        <item>1</item>
                                        <item>2</item>
                                        <item>3</item>
                                        <item>4</item>
                                        <item>5</item>
                              </array>
                    </DaysToExecute>
                    <WeekDay>1</WeekDay>
                    <MonthDay>1</MonthDay>
          </parameters>
</component>
```

Let ‘s review each parameter:

- SessionFactoryConfigPath: NHibernate configuration file.
- TemplatePath: Location for template files to use for mailing purposes.
- TimesToExecute (Array): Indicates in which hours the Newsletter @ Nybble should process all campaigns available.
- DaysToExecute: Indicates in which days the Newsletter @ Nybble should run.
- WeekDay: Indicates which the Weekly Sending Day is.
- MonthDay: Indicates which the Monthly Sending Day is.

As you can see, you can extend and use your own processor in case the one provided is not enough because is based on the INewsletterProcessor interface as follows:



Latest File Versions

You can always find the latest file version at P:\Nybble\Redistribution Files location.