NoticeMgr 1.0 Beta

Posted At : June 21, 2007 5:52 PM | Posted By : Steve
Related Categories: Mailer.cfc, com.sebtools, DataMgr

On most web sites, I typically allow my clients to edit the content on the pages of the site. I also generally have email messages sent out in response to certain events.

For example, I might have an email that tells a user that we have received their application. This is generally sent out by the component that handles processing the application (in my case, usually using **Mailer.cfc**).

Much like in the case of content on their site, the client will often want to change the content of the email. It isn't as easy to have them do this, however, as the component needs to be aware of the email message and having the component reference a primary key of data in a database seems like bad form (violating encapsulation - after all the key may be different on the dev database than the live one).

Still, changing the content of the email myself isn't challenging work and isn't the best use of my clients money. As such, I needed a better solution.

Enter NoticeMgr

NoticeMgr is really the child of two other components I created: DataMgr and Mailer. It requires both in order to work.

NoticeMgr allows a component to create an email notice that it will send in response to an event and allow the client to edit the contents of that notice without the component losing reference to it.

Using NoticeMgr

In order to use NoticeMgr, it must be passed in to the component that will use it.

```
<cffunction name="init" access="public" returntype="any" output="no">
  <cfargument name="NoticeMgr" type="any" required="yes">

  <cfset variables.NoticeMgr = arguments.NoticeMgr>

  <cfreturn this>
</cffunction>
```

Now the component has a reference to NoticeMgr. Next you must tell NoticeMgr about the email that you want to send using the addNotice() method:

- **Component** (required): The path to your component. Used as a unique identifier for your component.
- **Name** (required): The unique name for this notice.
- **Subject** (required): The subject of the email.
- **Text** (optional): The text contents of the email.
- **HTML** (optional): The HTML contents of the email.
- **DataKeys** (optional): A list of data keys (more below).

It is safe to add the same notice multiple times, as NoticeMgr will ignore subsequent requests to add the same notice from the same component. If another component tries to use a name that is already in use, however, NoticeMgr will throw an error.

Neither Text nor HTML arguments are required, but one of the two must be passed in.

Internally, NoticeMgr saves the notice as a row in a table that it creates via DataMgr called "emlNotices". So long as you call addNotice from the init method of your component, it is guaranteed to exist.

Then you can send the message by calling the sendNotice method and passing in the name of the

notice that you want to send.

Personalization Through Data Keys

None of this is much good, of course, if you can't personalize the email that you send. In order to personalize the message, you need to use DataKeys.

Here is an example of creating a notice that will be personalized using first and last names.

```
<cfinvoke component="#variables.NoticeMgr#" method="addNotice">
  <cfinvokeargument name="Component" value="cfcs.mycfc">
  <cfinvokeargument name="name" value="application received">
  <cfinvokeargument name="Subject" value="Application Received from [FirstName] [LastName]">
  <cfinvokeargument name="Text" value="[FirstName] [LastName], Thank you for your application.">
  <cfinvokeargument name="datakeys" value="Firstname,LastName">
</cfinvoke>
```

The "FirstName" and"LastName" keys are included in the comma delimited list of DataKeys and they are included in brackets in the text and subject of the email.

To send this notice, call the sendNotice() method and pass in a structure of data keys:

```
<cfset sData = StructNew()>
<cfset sData["FirstName"] = "Sally">
<cfset sData["LastName"] = "Jones">
<cfset sData["To"] = "sally.jones@example.com">
<cfset variables.NoticeMgr.sendNotice("application received",sData)>
```

This will replace "[FirstName]" with "Sally" and "[LastName]" with "Jones". You may notice, however, that there isn't a "To" data key. You can also use the data keys structure to replace any argument of the **send() method** of Mailer.cfc.

Editing Notices

Of course, I also advertised that NoticeMgr would allow your clients to edit the email messages.

NoticeMgr doesn't provide the user interface for editing the messages, but does provide to methods for this purpose.

First the getNotices() method returns a query of all of the notices that NoticeMgr is aware of. The saveNotice method can be used to update a notice. It has the same arguments as the addNotice method. instead of ignoring already existing notices, however, it updates them.

I have a demonstration of this functionality on my **demonstration site**. Note that if you see "Database" Sim" near the top of the page then the demonstration won't make much since as it is running on DataMgr's simulated database (which doesn't save any real data). The site switch database types every few hours.

Conclusion

The NoticeMgr component provides a good generic way to unify the handling of event-driven email messages. It allows your client to edit content that could otherwise be hidden away in programming logic.

NoticeMgr is open source and free for any use. It is also the newest addition to the **DataMgr Utility Set**.