Resolving a Template with Merge Fields

This article explains how to render a template with merge fields, using Apex.

Prerequisites

The developer will need proficiency in:

- Salesforce.com Object model
- Apex programming

What are templates?

SMS-Magic is a native application that integrates with Salesforce.com, which is primarily used to store lead and contact information. SMS communication is all about 1-to-1 communication, and many customers will give a prompt response if the message is personalized.

Merge-field templates can be used by sales or marketing teams to send personalized messages. This article explains how to do this with Salesforce.com Apex. It is very simple to implement, so that you can maintain focus on your business logic while SMS-Magic handles the SMS messaging.

Class & Method Details

Class: TemplateResolver

This class renders a template by resolving merge fields in the template text. There is only one global method definition in this class (see below).

Method: resolveTemplate

resolveTemplate(String originalTemplate, SObjectType objectType, List recordIds, Set extraFields)
Return Type : Map

The returned **Map** object key will reference the record in the object. The corresponding value will contain the template text.

Parameters:

- originalTemplate – the text of the template. If this value is passed as null or blank, an exception error will be raised.
- objectType – the sObjectType of the object instance that corresponds to the template. An exception error will be raised if this is null.
- recordIds – IDs of the records in the object instance. The field values will be used to populate the dynamic fields in originalTemplate. An exception error will be raised if this is null.
- extraFields – fields in addition to those in the template text.

Suppose mobile number of the record is also needed, then the user will not have to write separate query to fetch mobile number. Just mention that field in the set of extraFields, returned map will fetch the mobile number also.

How to render the template text

Now, let's see how to use the **TemplateResolver** class and **resolveTemplate** method in Apex.

First, a template containing merge fields must be created from the **Template** tab in SMS-Magic. It will be stored in the **SMS Template** custom object, which is accessible with API name

smagicinteract__SMS_Template__c.

Any template that corresponds to an object instance (and template text) will contain the merge fields of that object. Resolving a template consists of substituting actual field

values in the merge fields of the template text.

Template text is resolved with the **TemplateResolver** class using its **resolveTemplate** method. Follow these steps:

1. Select a predefined template record that corresponds to a Salesforce.com object.
2. Using **Schema.getGlobalDescribe()**, set **sObjectType** to the object that correspond to the template.
3. Create a list of IDs for the records that correspond to the template.
4. Optionally, create a set of **extraFields**.
5. Create an instance of the object **smagicinteract.TemplateResolver** class.
6. Call the **resolveTemplate** method, using that object instance.

Code Sample

Feel free to copy and modify this code example:

```
String templateText ='Hello  {!Contact.name}, Welcome to SMS-Magic . Regards, {!Contact.Account.name}';
List contactlist=[select id from Contact limit 10];
List IdList=new List();
for(Contact key:contactlist){
IdList.add(key.id);


Set extraFieldSet=new Set{'name' , 'department', 'MobilePhone'};
smagicinteract.TemplateResolver temp = new smagicinteract.TemplateResolver();
Map objectTextmap =temp.resolveTemplate(templateText, contactlist.getSObjectType(), IdList, extraFieldSet);
```

Double checking

1. Length of List of Objects (Number of record IDs) should not be greater than **200** else it will violate SF Limits.

2. Don’t send the TemplateText, ObjectType, or the record list as null or blank.
3. Don’t pass the Object ID field in **extraFields**.
4. If there are more than 20 merge fields in the template, the result will be null.
5. If the user does not have field-level permissions for each of the fields in the template text, then returned result will be null.
6. It is also necessary for the user to have read permission for the object.