Mogreet SDK
Tutorials in C# .Net

proposed by Anais Brossas
November 02, 2011

Abstract

This document provides several tutorials explaining
how to install, use and modify the SKD.

Contents

1 . Introduction

The Mercury Software Development Kit (SDK) provides the functions to implement the Mogreet API Mogreet Messaging System (MoMS).

This documentation will help you to understand the SDK (developed in .Net) via tutorials and schemas explaining the global operation of the application and how to install and use it in your computer.

2 . Prerequisites

Before trying to test the SDK, you have to install Visual Studio on your pc (if it is not already installed).

You can download the trial version here, which is free:
http://www.microsoft.com/visualstudio/en-us/try

We can now launch the application.

3 . Tutorials

3 . 1 . How to start the application?

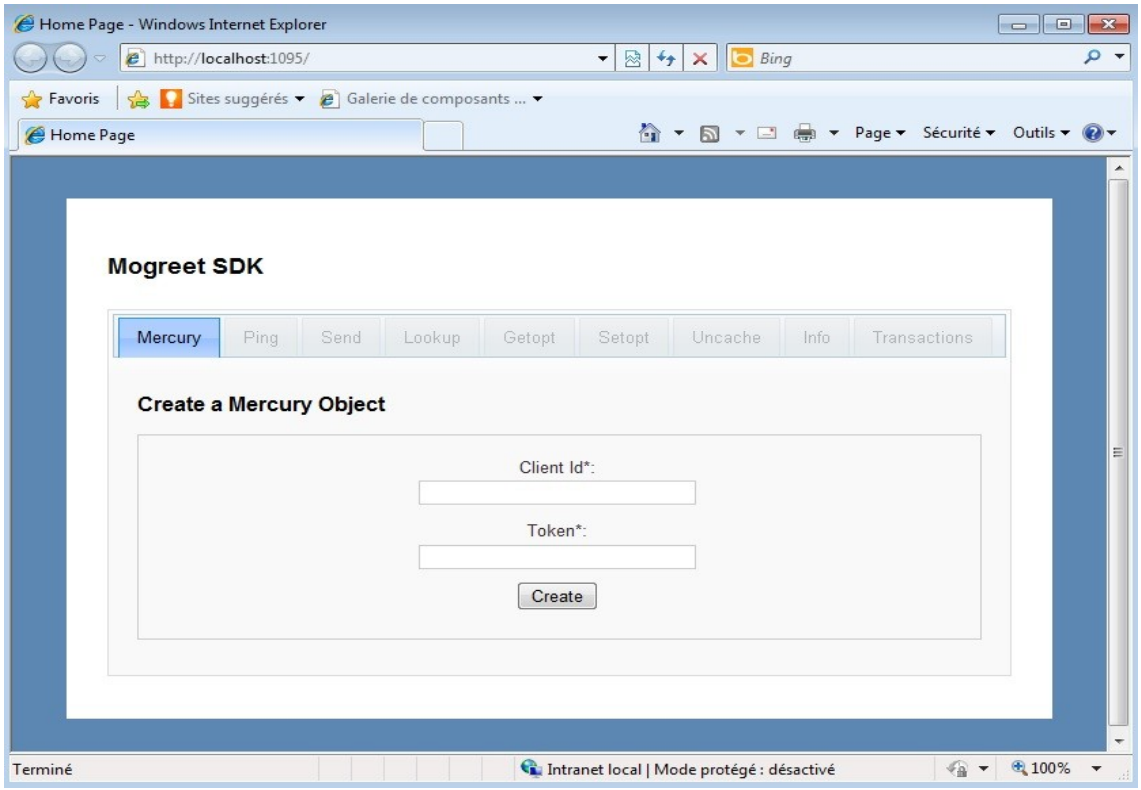
After launching Visual Studio, click on **File** **Open Project**, find the folder SDKApplication and double click on SDKApplication.sln.

Now click on **Debug** **Execute without debugging**. It will open Internet Explorer automatically and launch the application.

You have now launched the application, you can use it directly, or follow the next tutorial to know how it works.

3 . 2 . Create a Mercury Object and Ping

After launching the application you will have to enter a Client Id and a Token in order to create a Mercury object. You cannot access to the other tabs before creating it This object will allow you to execute any request to the Moms API. You will keep the same Client id and the Token during the application, but if you want to change them, you can, at any moment, go back to the Mercury Tab and create another Mercury Object.



Now your Mercury is created, check if you entered the right Client id and the Token by doing a Ping request.

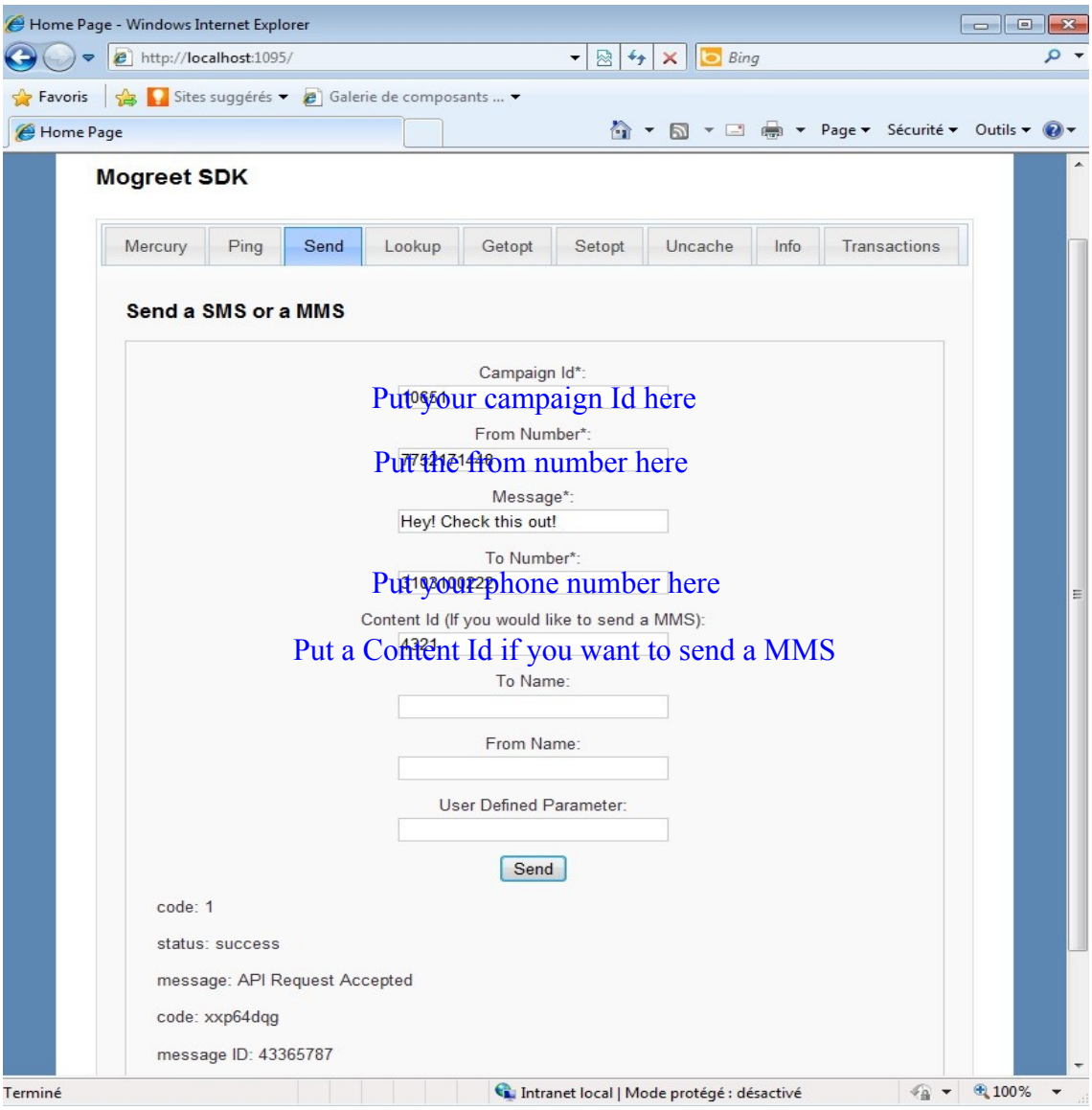
Succes! Your ping worked, and the message sent back is pong. Every time a request works, the code is 1.
If the Ping request doesn't work, an error page will appear with this message “Mogreet: 403 Forbidden”. It means that the Client id or the Token entered aren't good. Go to the previous page and create another mercury object.

Let's try another request now! Except for the Ping, any request need at least one other information that the Client id and the token.

3 . 3 . Do a Send request

Precedent tutorials must be followed before doing this one.
You have now created a Mercury object and check its parameters by doing a Ping request. Let's now send a message via MoMS API.

Go to the Send tab. Then enter the parameters needed:

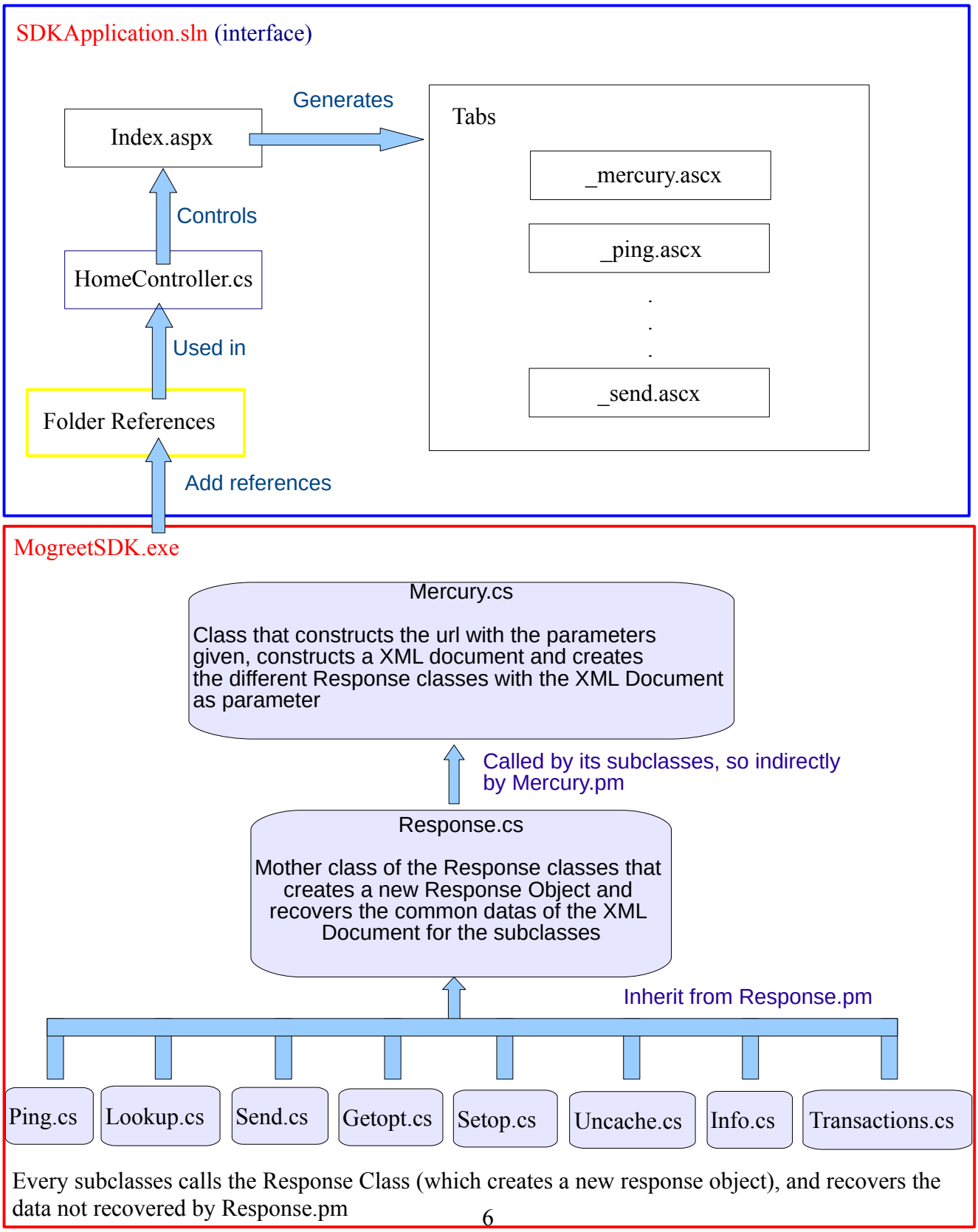


Success! The message has been sent! It is followed by a link to a video (because I entered a Content ID).

You know now everything about how to use the application, let's see how it works, and how you can modify it !

4 . Mogreet SDK architecture

4 . 1 . Global operation



4 . 2 . MogreetSDK.exe

MogreetSDK.exe is the executable of Mogreet SDK in C#.
It contains all the classes needed to do requests to MoMs API.

If you want to modify a class, open MogreetSDK.sln in Virtual Studio.
Don't forget after your modifications to generate the executable of the application. You can do that by clicking on **Generate** **Generate MogreetSDK**.
Then, import it in SDKApplication.sln to use your new classes or methods, by clicking on **References** **Add references** and find it in your folders.

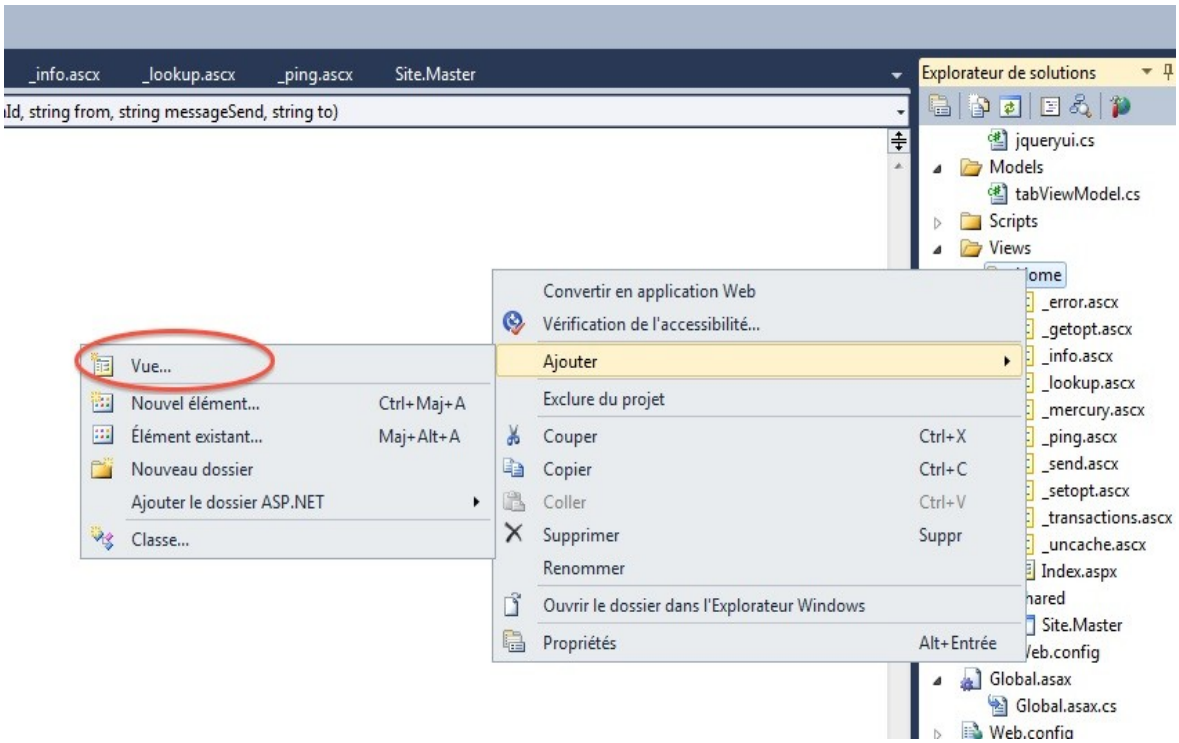
If you just need to access an attribute of a class, methods are available.
Refer to the documentation to know the name of the methods.

4 . 3 . SDKApplication.sln

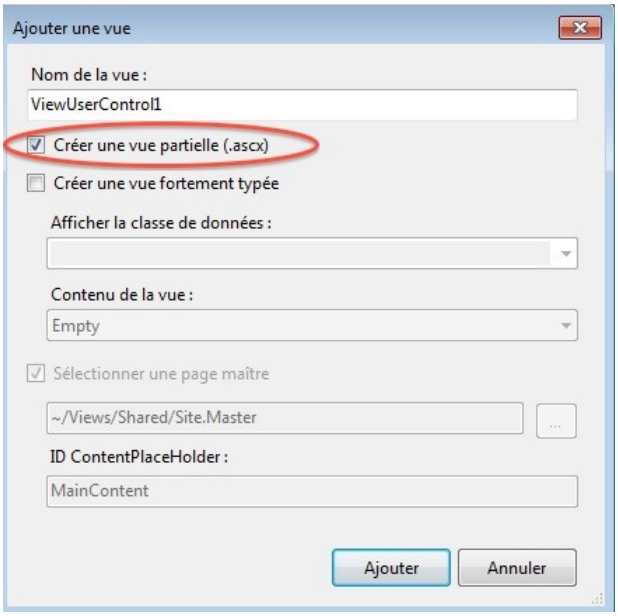
SDKApplication.sln is the graphical interface, in ASP.Net, of Mogreet Sdk in C#.
We will see how it works, how you can create a tab, add a field to a tab, recover data entered in a tab, use it for your mercury object...

4 . 3 . 1. Create a tab

First of all, you have to know that a tab is represented by a partial control view.
That is what you will create first: Click on the folder **Home** **Add** **Vue**



And then click on **Create partial view (.ascx)** and enter the name of the new view.



Let's see of what _setopt.ascx is composed to know what you should put in your view:

```
1  <%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl<SDKApplication.Models.tabViewModel>" %>
2
3  <h3>Do a Getopt request</h3>
4
5  <% using (Html.BeginForm("Index","Home"))%>
6   <% { %>
7          <fieldset>
8              <center><p>
9                  <label for="numberGetopt">Number*:</label>
10                 <%= Html.TextBox("numberGetopt") %>
11                 <%= Html.ValidationMessage("numberGetopt", "*") %>
12             </p>
13             <p>
14                 <label for="campaignIdGetopt">Campaign ID:</label>
15                 <%= Html.TextBox("campaignIdGetopt") %>
16             </p>
17             <p>
18                 <input type="submit" value="Getopt" />
19             </p></center>
20             <p> <%= ViewData["codeGetopt"] %> </p>
21             <p> <%= ViewData["statusGetopt"] %> </p>
22             <p> <%= ViewData["messageGetopt"] %> </p>
23             <ul><% if(ViewData["campaignsGetopt"] != null) %>
24                 <% foreach (var item in (List<string>)ViewData["campaignsGetopt"]){ %>
25                 <li>
26                     <%= item %>
27                 </li>
28             <% } %>
29             </ul>
30         </fieldset>
31 <%= Html.Hidden("getoptID", Model.ID) %>
32
33  <% } %>
```

Title of the page

Inherits of the same model as the other views, don't forget to put it!

Will use the Index function in HomeController

*Label of the TextBox: Number**
Id of the TextBox: numberGetopt
(will be used in Index function of HomeController)

The Number is a required data which will be checked in Validate_getopt function of HomeController

Submit Button

Display data from HomeController
Thanks to the id

As we don't know how much campaigns there is before the request, this will display all the campaigns

Gives to the controller the Id of the tab: *getoptID*

So I will recommend you to copy and paste this code, and to add or delete what you don't need, I put what you should change in Italic.

We are now going to see what you have to change in the **HomeController**.
In the Index function, add “string tabID” as parameter, and replace tabID by the ID of your new view (in the previous case getoptID).
Then add the Ids of the TextBox as parameters of the function. In our example “string numberGetopt, string campaignIdGetopt”.

Then add a new part of Index function code that will treat the data in the new view. Let's take the part of the function which concerns Getopt Tab as an example:

```
//we are in getopt tab
if (getoptID != null)
{
    selectTab = 4;
    if (!Validate_getopt(numberGetopt))
    {
        ViewData["tabID"] = selectTab;
        return View(tabView);
    }
    //Creates Dictionary (request parameter)
    Dictionary<String, String> parameters = new Dictionary<String, String>();
    parameters.Add("number", numberGetopt);
    if (campaignIdGetopt != null)
        parameters.Add("campaign_id", campaignIdGetopt);

    //Getopt request
    Getopt getopt = myM.getopt(parameters);

    //Prints elements
    ViewData["codeGetopt"] = "code: " + getopt.getResponseCode();
    ViewData["statusGetopt"] = "status: " + getopt.getResponseStatus();
    ViewData["messageGetopt"] = "message: " + getopt.getResponseMessage();

    List<int> list = getopt.getCampaignIdList();
    List<string> campaigns = new List<string>();
    for (int i = 0; i < list.Count; i++)
    {
        campaigns.Add("campaign id:" + list[i] + " status code: " +
        getopt.getCampaignStatusCode(list[i]) + " -> " + getopt.getCampaignStatus(list[i]));
    }

    ViewData["campaignsGetopt"] = campaigns;
    ViewData["tabID"] = selectTab;
    return View(tabView);

}
```

Replace getoptID by your view ID

Getopt is the 5th tab (but we begin to count at 0 so it has the number 4), put the place of your new tab here

Check if the required parameter is not null by calling Validate_getopt function

Display the elements in the view page where we put ViewData[“*id*”]

Returns tab with data displayed

Don't forget to add ValidateNameOfView at the end of the HomeController class:

```
private bool Validate_getopt(string numberGetopt)
{
    if (String.IsNullOrEmpty(numberGetopt))
    {
        ModelState.AddModelError("numberGetopt", "You must specify a number.");
    }

    return ModelState.IsValid;
}
```

And finally, in the **Index** page, you will have to add a link to the new view, let's see how to do that with our example:

```
9     <script type="text/javascript">
10        $(document).ready(function () {
11            $("#tabs").tabs();
12            if(<%=ViewData["init"]%> != 1){
13               $("#tabs").tabs("option", "disabled", [1,2,3,4,5,6,7,8]);
14            };
15            if(<%=ViewData["init"]%> == 1){
16               $("#tabs").tabs("option", "disabled", []);
17            };
18            $("#tabs").tabs("select", <%=ViewData["tabID"]%>);
19        });
20    </script>
21
22 </asp:Content>
23
24
25 <asp:Content ID="indexContent" ContentPlaceHolderID="MainContent" runat="server">
26      <h2>Mogreet SDK</h2>
27
28    <div id="tabs">
29        <ul>
30            <li><a href="#mercury">Mercury</a></li>
31            <li><a href="#ping">Ping</a></li>
32            <li><a href="#send">Send</a></li>
33            <li><a href="#lookup">Lookup</a></li>
34            <li><a href="#getopt">Getopt</a></li>
35            <li><a href="#setopt">Setopt</a></li>
36            <li><a href="#uncache">Uncache</a></li>
37            <li><a href="#info">Info</a></li>
38            <li><a href="#transactions">Transactions</a></li>
39        </ul>
40        <div id="mercury">
41            <% Html.RenderPartial("_mercury", new SDKApplication.Models.tabViewModel { ParentModel = Model, ID = "mercury" });  %>
42        </div>
43        <div id="ping">
44            <% Html.RenderPartial("_ping", new SDKApplication.Models.tabViewModel { ParentModel = Model, ID = "ping" });  %>
45        </div>
46        <div id="send">
47            <% Html.RenderPartial("_send", new SDKApplication.Models.tabViewModel { ParentModel = Model, ID = "send" });  %>
48        </div>
49        <div id="lookup">
50            <% Html.RenderPartial("_lookup", new SDKApplication.Models.tabViewModel { ParentModel = Model, ID = "lookup" });  %>
51        </div>
52        <div id="getopt">
53            <% Html.RenderPartial("_getopt", new SDKApplication.Models.tabViewModel { ParentModel = Model, ID = "getopt" });  %>
54        </div>
55        <div id="setopt">
```

Ids of tabs disabled before the creation of a Mercury object.
If you add a 10th tab, add “9” here.

Enable the tabs

Create tabs

Add a new tab to Index page

You will have to add a new div with, as Id, the same you put to add a new tab but without “#”.
Then, instead of “_getopt”, add the name of your new view, and instead of ID=”getopt”, add ID=”Id of the div”

4 . 3 . 3. Delete a tab

By following the previous tutorial, you know how the application works, how you can add a tab, and finally how to delete it!
Because you just have to delete what you just added in the previous part of this document:

- the partial view
- what you added in Index.aspx
- what you added in HomeController (in the Index function and the ValidateNameOfView corresponding).