

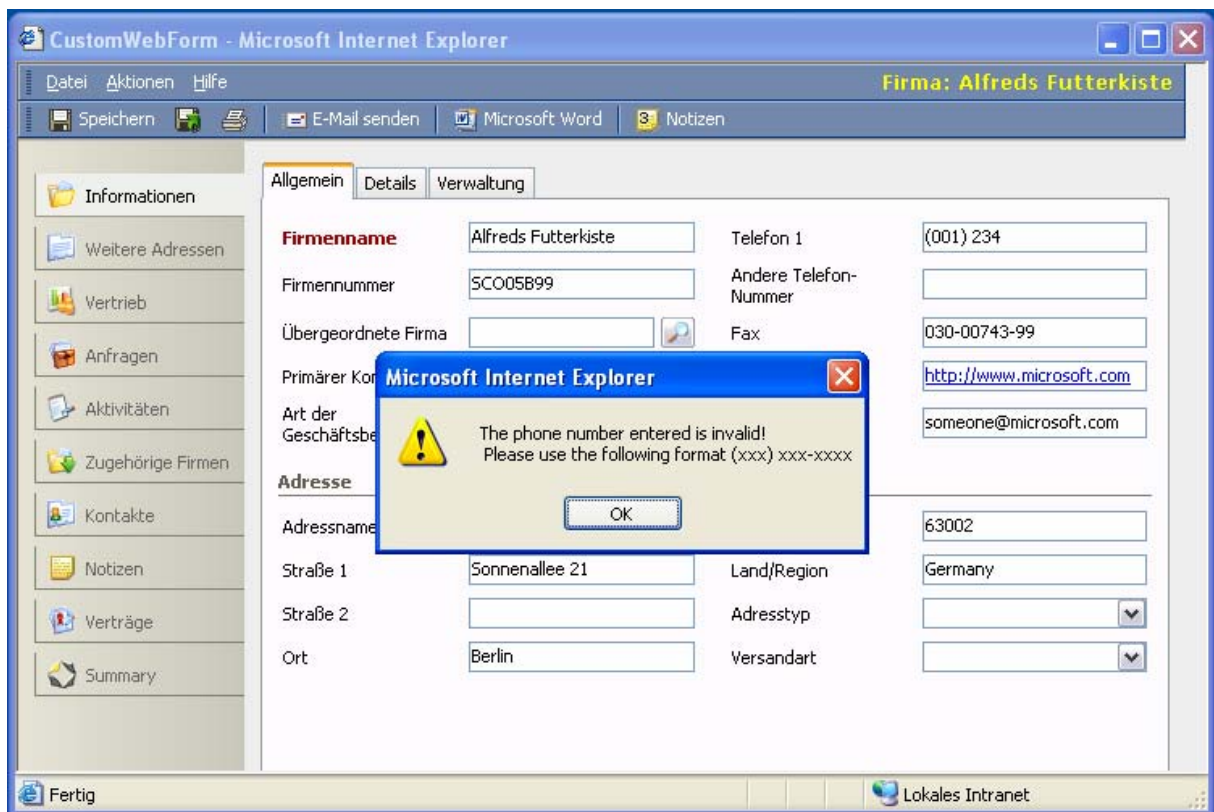
Advanced UI Customization for Microsoft CRM

Hello Microsoft CRM Gurus! Today I would like to show you some really cute tricks how to extend the User Interface (UI) of Microsoft CRM. There are great tools from Microsoft to customize Microsoft CRM to your needs, but sometimes the requirements ask for more flexibility and there are several ways to achieve that. As you know, all roads lead to Rome, but some of them may be very long. So you may find this shortcut valuable:

I have seen a lot of amazing customizations for the existing UI, but most of them are not supported by Microsoft, which means, that your modifications to the Microsoft files could break any upgrade (hotfixes, service packs or a version upgrade). Even if you could modify existing files, which are provided by Microsoft, you should not do so and there are better ways to get to the same result, without getting to an unsupported state.

In this example I will show you some advanced UI customizations without modifying existing Microsoft files. The only thing you have to do is to add one line of code to the *web.config* of the Microsoft CRM web application. The rest is done by injecting/redirecting code at runtime! So no modifications to the Microsoft files have to be done and a possible upgrade will not fail.

To demonstrate it, I wrote a few lines of code to add the possibility to validate the input of specific text fields. E.g. check for valid postcodes, telephone numbers, make dupe detection before saving the form, etc.



So let's start! Thanks to .NET and the architecture of Microsoft CRM, we could inject code to existing Microsoft CRM forms at runtime or redirect specific request to custom applications. This will be done by using [httpModules](#). httpModules makes it easy to extend the functionality of a web application with additional functionality, without having to touch the existing code.

Therefore we will create a new module to latch into the execution flow of the web application of Microsoft CRM. This allows us to check every single request and if necessary to redirect to a custom web application.

Three steps are required:

- Step 1, create the HttpModule
- Step 2, create a custom web application
- Step 3, add the HttpModule to the web.config of Microsoft CRM

But first things first:

Step 1, create the HttpModule:

In Visual Studio you should create a new class library; you may call it "ExtensionModul" and the class "ExtensionClass"

Add a new class which implements from the IHttpModule interface:

```
using System;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace ExtensionModul
{
    public class ExtensionClass: IHttpModule
    {
    }
    ...
}
```

This class allows us to add functionality to the web application workflow. The tricky thing is to hook up into "OnPreRequestHandlerExecute" event by using the Init method which makes it possible to get a reference to the page object:

```
public void Init(HttpApplication app)
{
    //register events
    app.PreRequestHandlerExecute += new
        EventHandler(this.OnPreRequestHandlerExecute);
}
```

So let's define the method which is used by the "PreRequestHandlerExecute" event:

```
public void OnPreRequestHandlerExecute(object o, EventArgs args)
{
    HttpContext context = ((HttpApplication)o).Context;
    IHttpHandler handler = context.Handler;

    //Hookup into PreRender event of the Page
    ((Page)handler).PreRender += new
        EventHandler(this.OnPagePreRender);
}
```

This method will be used to get a reference of the page object which makes it possible to latch into the PreRender event of the page object:

```
public void OnPagePreRender(object sender, EventArgs eventArgs)
{
    Page page = (Page)sender;

    //Check, if the customer card has been requested
    if (page.Request.FilePath.StartsWith("/SFA/accts/edit.aspx"))
    {
        //This line will avoid an endless redirection loop
        if (page.Request.QueryString["redirect"] == null)
        {
            //Redirect to a custom web application
            page.Response.Redirect
            ("/Custom/ExtensionHandler/CustomWebForm.aspx?" +
            page.Request.QueryString.ToString());
        }
    }
}
```

You could now use this method to inject any code you want into the web application. But we will not do that, because for performance reasons Microsoft CRM does not use ASP.NET controls. So instead of injecting code, we will redirect the web request to a custom web application. In this example we will modify the behaviour of the customer card and you may have noticed that we check the request file path and if that is equal to the customer card, we will redirect to our custom web application. It is very important to add the QueryString to the redirection. Else you would not be able to get the customer id in your custom application.

Before going forward one last task: compile the class library as a dll and copy it to the bin folder of the Microsoft CRM web application. This allows us to refer to our new extension module from the *web.config*. Congratulations, step 1 of your mission has been accomplished!

Step 2, create a custom web application

Having defined our extension module, which redirects specific request, we have to build a custom web application, which implements our actual business logic. So roll up your sleeves and let the work begin:

First create a new ASP.NET web application wherever you want, but keep in mind, that you have to point to the right path from the extension module. So keep that in synch. Add a new ASP.NET web form and name it as you defined it in your HttpModule. Inside of the body element add the following:

```
<BODY leftmargin="0" topmargin="0" rightmargin="0" bottommargin="0"
onload="Init()">
<TABLE border="0" cellpadding="0" cellspacing="0" width="100%">
<TR>
<TD>
<IFRAME id="iframe1" width="740" height="550"
src='/SFA/accts/edit.aspx?
<% Response.Write(Request.QueryString.ToString()); %>&redirect=1' />
</TD>
</TR>
</TABLE>
</BODY>
```

Take a look at the HTML code and you will notice that we are using an iframe to include the original website (the customer card) in our custom web application. From our web form we could now add javascript to implement custom business logic, e.g. to achieve custom form validation:

```
<SCRIPT type="text/javascript">
```

```

function Init() {

    // hook up into the onChange event of a specific field
    // Attention the following is one line of code:
    var fld =
        window.frames['iframe1'].
        document.forms['crmForm'].elements['telephone1'];

    fld.onchange = function (e)
    {
        // Here you could implement your custom validation logic
        // I is good idea is to use regex for validation
        // good tutorial:
        // http://www.codeproject.com/dotnet/expresso.asp
        // http://www.codeproject.com/useritems/RegexTutorial.asp

        var phoneRE = /^\\(\\d{3}\\) \\d{3}-\\d{4}$/;
        if (this.value.match(phoneRE))
        {
            alert('The phone number is valid');
            return true;
        }
        else
        {
            alert('The phone number entered is invalid!\\n
                Please use the following format (xxx) xxx-xxxx');
            return false;
        }
    }
}

</SCRIPT>

```

In this example we will validate the field telephone 1. If the format is different to “(xxx) xxx-xxxx”, we will raise an error message to the user. This is only a small example, but I am sure you can now imagine, that we could add any validation or business logic we want: field validation, dupe detection, web service access, etc.

Before going forward, we have to remove a small blemish. Because the original web form resides inside an iframe container, the “save&close” function will not work anymore. We will re-enable it, by overlaying the existing “save&close” button with a new one. So add the following just before the table:

```

<SPAN tabindex=0 style="height: 20px; padding: 1px; padding-left: 7px;
padding-right: 7px; border: 1px solid #7288AC; LEFT: 88px; POSITION:
absolute; TOP: 24px" title="Speichern und schließen"
onclick="SaveAndClose();" ><IMG src="/_imgs/ico/16_saveClose.gif"
class="mnuBtn"></SPAN>

```

And the following javascript will replace the existing “SaveAndClose()” function. So add the following to the javascript section:

```

function SaveAndClose()
{
    window.frames['iframe1'].document.forms['crmForm'].SaveAndClose();
    self.close();
}

```

First this function will call the original “SaveAndClose()” function and after that it will close the whole window. When you are ready, don’t forget to compile your web application.

Can you still follow? Don’t give up, just one final step:

Step 3, add the HttpModule to the web.config of Microsoft CRM

Finally, we just have to add the whole function to Microsoft CRM. This task is very easy, because you just have to add one line of configuration code to the *web.config* of the Microsoft CRM web application.

```
<configuration>
  <system.web>
    <httpModules>
      <add type="ExtensionModul.ExtensionClass,ExtensionModul"
          name="ExtensionModulName" />
    </httpModules>
  ...

```

The type attribute should be defined as follows: "classname,assemblyname", whereas classname is the fully qualified namespace path.

You are done, congratulations! Where is the champagne? Now, open a customer card in Microsoft CRM and enter different values into the phone number field!

Conclusion

By using HttpModules you can easily change the UI behaviour of Microsoft CRM without modifying existing files and this will keep Microsoft CRM in a supported state. This technique makes it possible to implement a lot of different scenarios: field validation, dupe detection before saving, show additional status information, etc.

Just a final warning: by using this technique Microsoft CRM will stay upgradeable, but what is not guaranteed, that your extensions will work in future version, if you do not take care.

If you have any comments, corrections or suggestion please let me know on http://blogs.msdn.com/joris_kalz

Disclaimer:

This posting is provided "AS IS" with no warranties, and confers no rights. Use of included script samples are subject to the terms specified at <http://www.microsoft.com/info/copyright.htm>