Custom Forms Scripting

Since web version of form designer, custom forms bring custom scripting, which is pretty powerfull feature. It is evolving part of application, based on needs of our clients, so there is no definite state of this feature. Scripting is pretty straitforward and selfexplanatory (you just write an ActionScript to it), but you need to know some basic principles, like accessing the form elements, accessing the selected context value or interaction with the application itself. Those features are providet by scripting API - set of functions which enables these features.

Accessing component in form

If you want to acces some component in form, you ahve to set the identifier for this component. Then you can access it by method **getComp**. Then you can work this the component as you wish, access its methods and properties.

The following example will sum values from components textInputA and textinputB into component textfieldC:

```
var valueA: Number = Number(getComp("textInputA").value);
var valueB: Number = Number(getComp("textInputB").value);


getComp("textfieldC").text = valueA + valueB;
```

You can find component properties on Adobe documentation, for example Textfield:

http://help.adobe.com/cs_CZ/FlashPlatform/reference/actionscript/3/flash/text/TextField.html

Accesing information from application

The script is running in isolated enviroment, but there is a way how to recieve some information from the system. It is exspecially usefull in cooperation with workflow scripts or other web applications, where you can call them and add contextId, session, server and other information as a parameter.

Method **getKeyValue** has list of predefined values, which can be used in scripting. Here is the list of keys:

```
keyValuesDictionary["contextId"] = context;
keyValuesDictionary["formId"] = formId;
keyValuesDictionary["formValuesId"] = formValuesId;
keyValuesDictionary["instanceName"] = SessionApplication.instance.instanceName; //not working
keyValuesDictionary["serverName"] = SessionApplication.instance.serverName; //not working
keyValuesDictionary["baseUrl"] = SessionApplication.instance.baseUrl;
keyValuesDictionary["endpointUrl"] = SessionApplication.instance.endpointUrl;
keyValuesDictionary["rootDomain"] = SessionApplication.instance.getFullRootDomain();
keyValuesDictionary["wsdlFolderUrl"] = SessionApplication.instance.wsdlFolderUrl;
keyValuesDictionary["username"] = SessionApplication.instance.serviceManager.session.userName;
keyValuesDictionary["userId"] = SessionApplication.instance.serviceManager.session.userId;
keyValuesDictionary["organizationId"] =
SessionApplication.instance.serviceManager.session.organizationId;
keyValuesDictionary["session"] = SessionApplication.instance.serviceManager.session.session;
keyValuesDictionary["server"] = SessionApplication.instance.serviceManager.session.server;
```

Here you can see usage of such a key:

```
trace(getKeyValue("server"));      //traces pioneer
trace(getKeyValue("username"));    //traves admin@pioneer

var text:String = "";
text += "\n" + "contextId " + getKeyValue("contextId");
text += "\n" + "formId " + getKeyValue("formId");
text += "\n" + "formValuesId " + getKeyValue("formValuesId");
text += "\n" + "instanceName " + getKeyValue("instanceName");
text += "\n" + "serverName " + getKeyValue("serverName");
text += "\n" + "baseUrl " + getKeyValue("baseUrl");
text += "\n" + "endpointUrl " + getKeyValue("endpointUrl");
text += "\n" + "rootDomain " + getKeyValue("rootDomain");
text += "\n" + "wsdlFolderUrl " + getKeyValue("wsdlFolderUrl");
text += "\n" + "loggedUser ID " + getKeyValue("loggedUser");
text += "\n" + "username " + getKeyValue("username");
text += "\n" + "userId " + getKeyValue("userId");
text += "\n" + "organizationId " + getKeyValue("organizationId");
text += "\n" + "session " + getKeyValue("session");
text += "\n" + "server " + getKeyValue("server");
getComp("textArea").text = text;
```

Triggering events

When you know how to work with components, next step is define, when to run the event. The events are solved method **createListener** which takes 3 parameters - name of object where we listen, name of event and function to be triggered.

Following function will show you how to trigger the functionality from previous example on change of bot inputs - textInputA and textInputB.

```
this.createListener("textInputA", "change", sumInputs);
this.createListener("textInputB", "change", sumInputs);

function sumInputs(evt:Object):void
{
    var valueA:Number = Number(getComp("textInputA").value);
    var valueB:Number = Number(getComp("textInputB").value);
    getComp("textfieldC").text = valueA + valueB;
}
```

You can find events on objects using classical ActionsScript reference, for example button events like click, mouseDown, mouseUp, rollOver, mouseOut and so can be found on link:

http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/fl/controls/Button.html

Alert window

Sometimes we need to tell user that some other action is needed or so. Alert is typical way how to do so, so we can use prepared method showAlert. It encapsulates classical MessagePopUp with the same

* @param type - type of icon - ERROR, WARNING or INFO
* @param message
* @param description
* @param buttons
* @param closeHandler
function showAlert(type:String, message:String, description:String, buttons:Array = null, closeHandler:Function = null):void;

Here is an example how to show alert and react on the close.

```
this.createListener("btn", "click", btnHandler);

function btnHandler(evt:Object):void
{
    showAlert("ERROR", "ALERT!","Wow cool text", ["OK",  "Cancel", "I do not know"],
closeHandler);
}

function closeHandler(btnIndex:Number):void
{
    getComp("btn").label = btnIndex;
}
```

Opening URL

Since the function navigateToUrl is not working becuase of security reasons, we have to use our own function for url opening. See example bellow:

```
this.createListener("button", "click", btnHandler);

function btnHandler(evt:Object):void
{
    openUrl("https://google.com");
}
```

Calling external URL

Sometimes you need to call external URL and load something (without opening new browser window). That can be solved by method callUrl.

* @param url
* @param resultHandler
* @param failHandler
function callUrl(url:String, resultHandler:Function = null, failHandler:Function = null):void;

```
callUrl("https://fangtooth.atollon.com/external.js", loadComplete, loadFailed);

function loadComplete(data:Object):void
{
```

```
    getComp("textArea").text = data;
}

function loadFailed(evt:Object):void
{
    getComp("textArea").text = evt;
}
```

Escape URL parameters

Since URL does not support symbols like " ", "/", ":", "=" or unsuported ascii characters in URL parametes, it is important to excape all of the parameters, so you prevent problems with some specific strings and other data. This is solved by URLVariables, see method createUrlParams in example bellow:

```
this.createListener("Lic_proc_button", "click", btnHandler);

function btnHandler(evt:Object):void
{
	var urlParams:String = createUrlParams();
	var scriptLocation:String = getKeyValue("rootDomain") + "/fo/imp-exp";
	var fullUrl:String = scriptLocation + "?" + urlParams;

	getComp("Lic_proc_link").text = fullUrl;
	openUrl(fullUrl);
}

function createUrlParams():String
{
        // creation of URLVariables
	var urlVariables:URLVariables = new URLVariables();
	
	// fill URL variables
	urlVariables["company"] = "REAL TRADE PRAHA a.s.";
	urlVariables["ico"] = "25642740";
	urlVariables["street"] = "náměstí 14. října";
	urlVariables["city"] = "Praha 5 - Smíchov";
	urlVariables["street_no"] = "1307/2";
	urlVariables["pemr"] = getComp("Lic_proc_authnumber").text;
```

```
	urlVariables["zip"] = "15000";
	urlVariables["id"] = getKeyValue("formValuesId");
	urlVariables["endpoint"] = getKeyValue("endpointUrl");
	urlVariables["instance"] = getKeyValue("server");
	urlVariables["session"] = getKeyValue("session");

		// toString method of URLVariables returns escaped string
		// so for example Praha 5 - Smíchov looks like:
		// city=Praha%205%20%2D%20Sm%C3%ADchov
	return urlVariables.toString();
}
```

Debugging code

Sometimes it is complicated to find a problem in the script. Usually typo in the name of component, sometimes it is somehting different. To make the debugging as easy as possible, we added function log(message) into the scripting. There are different ways how is hte log shown. The log message will appear in the log view under the script when debugging in formDesigner or as popUp Alert when the form presented when displayng form in the finder.

An example how this works in OSSPO

```
this.createListener("SmlouvaNaDobu", "change", updateSmlouvaNaDobu);
this.createListener("SmlouvaNaDobu2", "change", updateSmlouvaNaDobu);
this.createListener("SmlouvaNaDobu3", "change", updateSmlouvaNaDobu);
this.createListener("DatumUzavreniSmlouvy", "change", updateSmlouvaNaDobu);
this.createListener("DatumUzavreniSmlouvy2", "change", updateSmlouvaNaDobu);
this.createListener("DatumUzavreniSmlouvy3", "change", updateSmlouvaNaDobu);
this.createListener("SmlouvaNaDobuCelkem", "change", updateSmlouvaNaDobu);

function updateSmlouvaNaDobu(evt:Object):void
{
   if(getComp("DatumUzavreniSmlouvy").text != "")
   {
      getComp("SmlouvaNaDobuCelkem").text = getComp("SmlouvaNaDobu").text;
   }

   if(getComp("DatumUzavreniSmlouvy2").text != "")
   {
      getComp("SmlouvaNaDobuCelkem").text = getComp("SmlouvaNaDobu2").text;
   }
```

```
   if(getComp("DatumUzavreniSmlouvy3").text != "")
   {
      getComp("SmlouvaNaDobuCelkem").text = getComp("SmlouvaNaDobu3").text;
   }
}
```

Revision #1
Created 23 January 2020 14:16:29 by Jan Safka
Updated 15 March 2020 08:09:48 by Jan Safka