XML-Dialog Verwendung

1 Einleitung

Über die WebKomponente efXMLDialog lassen sich einfach standardisierte Dialoge erstellen und auswerten. Es wird nicht programmiert, um die Elemente anzuordnen, sondern einfach deklariert. Das beschleunigt den Dialog-Entwicklungszyklus und sorgt für einen durchgehenden Standard innerhalb der Webanwendung.

2 Aufbau der XML-Definitionsdatei

2.1 Beispiel (aus /templates/dialogs/simple.xml)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <efDialogPage MULTIROW="0">
        <efDialogRow>
            <efDialogField>
                <DESC>Eingabe 1</DESC>
                <NAME>Name1</NAME>
                <TYPE>INPUT</TYPE>
            </efDialogField>
            <efDialogField>
```

```
            <DESC>Eingabe 2</DESC>
            <NAME>Name 2</NAME>
            <TYPE>INPUT</TYPE>
        </efDialogField>
    </efDialogRow>
</efDialogPage>
<include file="../other.xml"/>
```

3 Elemente

Achtung: da es sich um eine XML-Datei handelt, ist die Groß- /Kleinschreibung relevant.

3.1 <efDialogPage>

Dokumentroot

3.1.1 Attribute

- MULTIROW="1" oder MULTIROW="0"
  Lesen Sie hierzu den Abschnitt "Mehrzeilige Dialoge"
- DIALOGSIZE="xx"
  Nur bei MULTIROW="1" relevant. Legt die Anzahl der Zeilen pro Seite fest.

3.2 <efDialogRow>

Dialogzeile

3.2.1 Attribute

- ROWSPAN

3.3 <efDialogField>

Eingabefeld, Label, Combobox, Link…

3.3.1 Attribute

- COLSPANLABEL, COLSPANFIELD
  Falls sich das colspan auf das Label bezieht, so muss COLSPANLABEL verwendet werden, ansonsten COLSPANFIELD oder beides oder keins
- WIDTHLABEL, WIDTHFIELD
  in HTML-Angabe,z.B. „100px", „10%"; analoge Verwendung zu COLSPANLABEL/COLSPANFIELD

3.3.2 Untertags

- <DESC>
  Enthält eine Beschreibung des Elements, die im Dialog angezeigt wird; wird automatisch übersetzt
  Sie können die Zeichen HOT vor und nach einen Buchstaben setzen. Damit wird ein Label-Element generiert und falls der Benutzer „ALT+Buchstabe" drückt, so wird das zugehörige Control markiert, z.B.:
  HOTuHOTserName führt zur html-Anzeige: username
  Falls der Benutzer „Alt + u" so wird die zugehörige Checkbox markiert
- <NAME>
  eindeutiger Name des Controls
  Wichtig: es sollten keine Sonderzeichen und Leerzeichen verwendet werden, da bei <DIALOGINPUT> dieser Name als XML-Tagname verwendet wird.

- <TYPE>
  Mögliche Werte:
- <COLWIDTH>
  gibt in HTML-Syntax an, wie breit die Spalte ist (Default ist 25%)

INPUT	Standard Eingabebox
LABEL	Anzeigelabel
DISABLED	Gesperrtes Eingabefeld
INPUTFIRST	Nur bei Neuanlage editierbar, ansonsten readonly
PASSWORD	Input-Passwortfeld
HIDDEN	Verstecktes Eingabefeld
TEXTAREA	Mehrzeiliges Eingabefeld
CHECKBOX	Checkbox
LISTCOMBO	Auswahlliste mit festen Werte
DATACOMBO	Datenbasierte Auswahlliste
DATACOMBON	Datacombo mit einem zusätzlichen leeren Eintrag
BUTTON	Button mit [Value] als Beschriftung
BUTTOND	Disabled Button
ENTITY	Name der Entity, die ausgewählt und zugewiesen werden kann (aus tsEntities)
DATE	Datumseingabe
NUMBER	Nummerneingabe

- <LABELWIDTH>
  gibt in HTML-Syntax an, wie breit die Spalte ist (Default ist 25%)
- <EDITION>
  not yet implemented
- <PERMISSION>
  not yet implemented
- <READONLY>
  kann immer ein Feld sperren (Werte: 0 und 1)
- <ONCLICK>, <ONDBLCLICK>, <ONMOUSEDOWN>, <ONMOUSEUP>, <ONMOUSEOVER>, <ONMOUSERMOVE>, <ONMOUSEOUT>, <ONKEYPRESS>, <ONKEYDOWN>, <ONKEYUP>, <ONBLUR>, <ONCHANGE>, <ONSELECT>
  enthält den Rumpf einer Javascript-Funktion;
  <ONCLICK>
  alert(„good-bye“);
  window.close();
  </ONCLICK>
- <VALUE>
  Defaultwert beim Öffnen des Dialogs(wenn kein Wert über xXmlValues angegeben ist)
  Die Angabe von $query$xxxx z.B. „$query$select top 1 name from tdCustomers“ ist hier möglich
  Bei Button: die Beschriftung des Buttons
- <SRC> nur für DATACOMBO, LISTCOMBO
  bei DATACOMBO – Bezeichnung der Datenquelle aus Tabelle tsDataCombo
  bei LISTCOMBO – Semikolon separierte Liste aus Werten z.B.:

<INPUT>LISTCOMBO</INPUT>
<SRC>Kunde;Lieferant</SRC>
<DATA>C;S</ DATA >

- <DATA> nur für LISTCOMBO
  semikolonseparierter String, der die Datenbankwerte der Werte aus <SRC> enthält, z.B.:
  <INPUT>LISTCOMBO</INPUT>
  <SRC>Kunde;Lieferant</SRC>
  <DATA>C;S</ DATA >
- <BUTTONTYPE> nur für Buttons
  mögliche Werte:
  User <default>
  Submit – submits the form
  Reset – resets the form
- <BUTTONSTYLE> nur für Buttons
  mögliche Werte:
  cmdButton <default>
  cmdButtonDouble
  cmdButtonTriple
  cmdButtonSmall
- <COLS>, <ROWS> nur für Input = TEXTAREA
  Anzahl der Zeilen und Spalten eines mehrzeiligen Eingabefeldes

Bei der Erzeugung des Controls werden automatisch Präfixe vorangestellt.
Folgende werden verwendet:

-lbl	LABEL
-txt	INPUT, DISABLED, DISABLEDL, INPUTFIRST, PASSWORD, TEXTAREA
-chk	CHECKBOX
-cmd	BUTTON
-cbo	LISTCOMBO, DATACOMBO, DATACOMBON

4 Datenstruktur XMLDialog (über Eigenschaft sXmlValues)

Der Dialog enthält eine XML-Struktur als Grundlage. Diese ist folgendermaßen aufgebaut:

Single-Row Dialog	Multi-Row Dialog
<DIALOGINPUT> <xxxxxx></xxxxx> <xxxxxx></xxxxx> <xxxxxx></xxxxx>	<DIALOGINPUT> <ROW NUMBER="1"> <xxxxxx></xxxxx> <xxxxxx></xxxxx>

<xxxxxx></xxxxx> <xxxxxx></xxxxx> </DIALOGINPUT>	<xxxxxx></xxxxx> <xxxxxx></xxxxx> <xxxxxx></xxxxx> </ROW> </DIALOGINPUT>

xxxx enthält die Namen, die auch tatsächlich in der Dialog-XML-Datei definiert sind (s. Beispiel (aus /templates/dialogs/simple.xml))

Benutzen Sie die Methode `easyFramework.Frontend.ASP.AspTools.Tools.gsXmlsRecordset2DialogInput`, um komfortabel aus einem Recordset das entsprechende XML für einen Dialog zu erzeugen.

Anmerkung: falls der Dialog die Daten dynamisch im geladenen Browserfenster nachladen soll, so darf nicht die Eigenschaft sXmlValues, sondern sDataPage verwendet werden. Clientseitig werden die Dialogdaten zugewiesen. Details lesen Sie bitte unter Benutzung einer Datapage als Dialogdatenquelle.

Nach Verarbeitung des Dialogs erhält man XML nach folgender Struktur:

Single-Row Dialog	Multi-Row Dialog
<DIALOGOUTPUT> <txtXXXXX>wert 1</txtXXXXX> <chkYYYYY>1</YYYYY> </DIALOGOUTPUT>	<DIALOGOUTPUT> <ROW NUMBER="1"> <txtXXXXX>wert 1</txtXXXXX> <chkYYYYY>1</YYYYY> </ROW> </DIALOGOUTPUT>

Das Voranstellen der Dialogpräfixe verbessert die Lesbarkeit des Codes. Zudem ist eine Fehlerkontrolle möglich, wenn Steuerelemttypen z.B. von Combobox zu Checkbox verändert werden. Der ursprünglich erwartet Name wäre dann nicht mehr vorhanden, und ein Fehler wird im Benutzercode ausgeführt, wenn versucht wird, auf den alten Namen zuzugreifen.

4.1 Erzeugtes HTML des Dialogs

Das Ergebnis XML <DIALOGOUTPUT/> wird von der Client-seitigen Javascript Funktion `gsXMLDlgOutput` in efDlgParams.js erstellt. Das hidden-field „__dialogtype“ enthält die Werte „multirow“ bzw. „singlerow“ und nimmt daher Einfluss auf das Ergebnis XML.

4.2 Präfix bei Multi-Row Dialogen

Bei Multirow-Dialogen wird beim Namen jedes Elementes das Präfix „ROWxxx_“ vorangestellt, z.B. „ROW0_txtPassword“, „ROW424_chkSave“, wobei „xxx“ mit der gegenwärtigen Row-Number ersetzt wird.

5 Aufruf des Dialoges und Auswerten der Daten

5.1 Aufruf der Dialogkomponente

Die Dialogkomponente befindet sich standardmäßig auf einer ASP-Seite. Im Regelfall wird im Page_Load-Event der ASP-Seite der Seiteninhalt initialisiert.

Das Objekt DataTools enthält die Methode gsXmlRsToDialogInput, welche ein Recordset zu Dialogdaten umwandelt. Das gewonnene XML kann in der Eigenschaft **sXmlValues** der Dialogkomponente gesetzt werden. Natürlich kann dieses Xml auch auf andere Weise generiert werden - man ist nicht auf das Recordset beschränkt.

5.2 XML-Definitionsdatei

Sie können in der Eigenschaft sDefinitionFile entweder einen Dateinamen, oder aber direkt das XML einer Definitionsdatei hinterlegen.

5.3 Benutzung einer Datapage als Dialogdatenquelle

Ebenso kann als Datenquelle des Dialogs eine Daten-ASPX-Datei angegeben werden. Hierzu muss die Eigenschaft sDataPage auf den Pfad der Daten-ASPX-Datei gesetzt werden. Der Abfruf der Daten aus sDataPage erfolgt NICHT automatisch, sondern erst per JavaScript-Befehl.

Der Vorteil, falls eine DataPage verwendet wird, ist der, dass zur Laufzeit des Formular verschiedene Dateninhalte nachgeladen werden können, z.B. nach Auswahl aus einer Listbox im selben Formular. Dafür müssen dann jedoch mindestens zwei Serveraufrufe stattfinden, einmal beim Erzeugen des HTMLs des Dokuments und dann beim Abruf der Daten.
Falls in dem Formular von Anfang nur ein fixes Datensatz bearbeitet werden muss, ist es einfacher, im page-loadevent die sXmlValues-Eigenschaft des Dialogs zu setzen. Es ist hierfür auch nur eine Serveraufruf erforderlich, da die Dialogdaten samt dem HTML des Dialogs beim Client ankommen. Es muss nichts nachgeladen werden. Nachteil ist jedoch, dass dynamisch keine Inhalt nachgeladen werden können.

Der Dialog erzeugt folgende Javascript-Befehle, falls eine sDatapage verwendet wird:

5.3.1 Dialogdaten per Javascript laden

Aufruf von g<DialogID>fetchData(sAddParams) im Javascript, z.B.

```
function OnLoad() {
      gefXmlDialog1fetchData("<userid>12</userid>")
}
```

5.3.2 Dialogdaten leeren

Aufruf von g<DialogID>flush() im Javascript, z.B.:

```
function onCmdNewClick() {
      gefXmlDialog1flush();
}
```

5.3.3 Format des Return-Strings der DialogDataPage

Tipp: verwenden Sie die Methode `easyFramework.Frontend.ASP.AspTools.Tools.gsXmlsRecordset2DialogInput` um bequem aus einem bestehenden Recordset Dialoginputdaten für die JavaScript-Routine zu erzeugen. Geben Sie den Parameter `enFormat` mit `efEnumDialogInputType.efJavaScriptString` an.

Die einzelnen Felder werden mit dem Pipe-Zeichen getrennt. Am Ende jeder Zeile steht immer „-||-“. Jeder neue Datenblock beginnt mit einer Zeile „__ROW|0|-||-“ bzw. „__ROW|1|-||-“ ….
Der Zeilenindex gebinnt bei 0 (in Worten: Null).

Jede Zeile enthält einen Feldwert des Dialogs.
1.Feld: Zeilennummer (für mehrzeilige Dialoge relevant; bei einzeiligen immer „1“)
2.Feld: Feldname
3.Feld: Feldwert

z.B.
__ROW|0|-||-
Vorname|Marc-Christian|-||-
Name|Wimmer|-||-
__ROW|1|-||-
Vorname|Andreas|-||-
Name|Tropschug|-||-

5.4 Extrakt der Dialogdaten und Weitergabe an Processpage

Wenn der Benutzer auf einen OK-Button klickt, soll der Dialog seine Daten als XML ausgeben. Dieses XML soll dann an eine Process-Seite zur Auswertung übergeben werden.

In der JavaScript-Datei efDlgParams.js steht die Funktion `gsXMLDlgOutput(frm, sAdd, bAddEmptyValues)` zur Verfügung, welche aus dem angegebenen Formular das DIALOGOUTPUT-XML erstellt, welches an die Standard-Javascript `gsCallServerMethod` Funktion aus efServerProcess.js zum weiterverarbeiten übergeben werden kann.

5.5 Gegencheck zu Originalwerten

Zu jedem Element wird der ursprüngliche Wert hinterlegt, z.B.:

```
<DIALOGOUTPUT>
      <ROW NUMBER="1">
            <txtUsername>Marc</txtUsername>
            <txtUserName_old>Bianca</txtUserName_old>
      </ROW>
```

</DIALOGOUTPUT>

An den Namen des Elements wird das Suffix „_old“ angehängt.
Mit Hilfe der Original-Wertfelder kann leicht abgeprüft werden, ob sich Feldinhalte verändert haben.

5.6 Standardverhalten eines mehrzeiligen Dialogs (multirow)

Mehrzeilige Dialoge verhalten sich im easyFramework stets nach demselben Prinzip. Im Attribut „DIALOGSIZE“ wird die Anzahl der Reihen auf einer Seite festgelegt. Automatisch werden die Navigationsbuttons für „Erste Seite – Zurück – Vor – Letzte Seite“ angezeigt, mit denen der Benutzer durch die Seiten navigieren kann. Falls DIALOGSIZE beispielsweise den Wert 10 hat, werden 10 Reihen zur Eingabe bereitgestellt. Möchte der Benutzer die nächsten 10 Reihen bearbeiten oder Werte in weiteren leeren Zeilen eingeben, so klickt er auf den Navigationsbutton „Weiter“. In diesem Augenblick wird die aktuelle Seite neu geladen. Eventuell erscheint vorher eine Meldung „Änderungen speichern?“, falls die Daten verändert wurden. Die Seite wird mit neuen URL-Parametern geladen. Es werden sämtliche bestehenden Parameter beibehalten, zusätzlich wird der Parameter „__DialogPage“ hinzugefügt, um die aktuelle Seite zu kennzeichnen (Standardwert von __DialogPage ist 1).
Der Parameter __DialogPage kann im Page-Load-Event verwendet werden, um die Daten für die Seite x zu laden (z.B. über die Methode DataMethods.gRsGetDirectPaged)

6 Javascript-Elemente

6.1 Aufrufbare Methoden im Zusammenhang mit XML-Dialogen

- `gFocusFirstElement(sHtmlFormName)`
  Fokusiert das erste sichtbare Element eines Formulars
- `gXmlDialogFetchData (sXmlDialogId)`
  Füllt einen Dialog durch Abruf einer DataPage
- `gXmlDialogFlush (sXmlDialogId)`
  leert einen Dialog