

Javascriptový vyhodnocovací mechanismus výrazů

Aplikace Form Designer obsahuje vnitřní makrojazyk, který umožňuje vytvářet různě složité formulářové funkce, potažmo ovládání logiky formuláře. Nově je možné výrazy definovat pomocí javascriptového vyhodnocení, voláním metod objektů a příslušných předdefinovaných funkcí.

Pro zpracování zápisu výrazů navržených novým mechanismem je implementován javascriptový interpret V8 společnosti Google.

Tento nový mechanismus slouží k navržení složitějších výrazů a ke zpřístupnění funkčnosti, která nebyla pomocí předchozího mechanismu dostupná. Nově je tedy možné používat například cykly či lépe pracovat s podmínkami. Navíc jsou již zpřístupněny funkce umožňující práci nad více dokumenty, respektive záložkami. Syntaxe se neliší od standardního javascriptového zápisu, navíc je možné používat sadu metod a předdefinovaných funkcí sloužící k práci s formulářem nebo celou aplikací Form Filler.

Zároveň je možné stále používat původní makrofunkce, pokud jsou zapsány samostatně a nekombinují se metodami a funkcemi nového vyhodnocovacího aparátu. Tedy, funkce z nového a staršího mechanismu nelze kombinovat v jednom funkčním formulářovém výrazu.

Nové funkce se syntakticky zapisují výrazem vloženým do složených závorek – { }. Tím se v okně návrháře výrazů navíc automaticky zpřístupní nabídka nových metod a funkcí.

Co přinese?

Umožní využívat možnosti XML formulářů pomocí Javascriptového vyhodnocovacího aparátu. To znamená možnost využívat objektový skriptovací jazyk, namísto vnitřního makrojazyka a jeho sady funkcí.

- Jednoznačný a mnohokrát popsaný jazyk
- Využití cyklů a podmínek
- Jednoznačně definovaná posloupnost funkcí – nemělo by již docházet k asynchronnímu zpracování příkazů
- Možnost definování vlastních vnitřních funkcí
- Možnost využívat vlastní proměnné
- Lepší využívání datových typů
- Možnost využívat globální funkce namísto vložených XSLT transformací

Podpůrné materiály:

- Jednoduchá nápověda k syntaxi Javascriptu - www.jakpsatweb.cz/javascript
- Dokumentace a další materiály jsou dostupné na webu Software602, konkrétně v sekci dokumentace: http://www.602.cz/form_designer/dokumentace

Funkce a metody

Příkazy jsou rozděleny do několika skupin.

- Funkce
- Metody objektu Document
- Metody objektu Element

Funkce

Tyto funkce jsou přímo součástí aplikace, a tedy není nutné definovat kontext, nad kterým jsou volány.

Používají se velmi podobně jako ve starém vyhodnocovacím mechanismu, tedy nejsou volány nad žádným objectem, na rozdíl od metod.

Příklad volání na tlačítku globální funkce MessageBox:

```
{MessageBox("Dokument se změnil.", "titulek okna", MB_OK|MB_ICONINFORMATION);}
```

Metody objektu Document

Metody objektu Document jsou volány nad celým aktivním dokumentem. Pro jejich volání se tedy používá jako kontext celý xml dokument.

Tyto funkce se již volají objektově, respektive se volají metody objektu. Pro jejich deklaraci je nutné nejprve získat kontext dokumentu nad kterým se budou metody volat. Toto se provede pomocí funkce GetDocument(), která získá kontext aktuálně otevřeného formuláře.

Příklad volání metody dokumentu SaveSigned:

```
{var doc = GetDocument(); doc.SaveSigned("c:\\test\\podepsany.zfo", true);}
```

Metody objektu Element

Metody objektu Element se volají nad konkrétním formulářovým objektem, tedy nad fo prvkem či nad elementem datové věty.

Metody se volají stejně jako v případě metod objektu Document. Tedy je nutné nejprve získat kontext elementu nad kterým je metoda volána. Toto se provede pomocí funkce GetElement(), která je metodou dokumentu.

Příklad volání metody SectionNo:

```
{var doc = GetDocument();  
var ele = doc.GetElement();}
```

```
var vysledek = ele.SectionNo();  
MessageBox(""+vysledek,"Číslo sekce",MB_OK);}
```

Všechny tyto funkce a metody mají jasně definované parametry. Tyto parametry jsou popsány v nápovědě k aplikaci Form Designer.

U některých funkcí, jako je například MessageBox, lze použít i předdefinované konstanty, které ovlivňují výsledek výrazu. Tyto konstanty a jejich použití lze nalézt v dokumentu k funkcím.

Návrhář výrazů

Při návrhu výrazů se zápis provádí v dialogu aplikace Form Designer, který obsahuje předdefinovanou sadu funkcí v menu návrháře, syntaktické zvýraznění zápisu, kontrolu a další užitečné pomůcky. Funkce jsou v nabídce rozřazeny do skupin a navíc jsou již vyfiltrovány podle aktuálního kontextu v dokumentu. Znamená to, že různé funkce jsou dostupné podle jejich možnosti použití – sady funkcí se liší na tlačítku, na vazebních vlastnostech a v dialogu pro zpracování certifikátů.

Příklady jednoduchých volání metod a funkcí:

Příklad výrazu pro spojení tří stringů:

Následující příklad ukazuje zapsání funkce concat pomocí javascriptových operátorů. Jako výsledek je do aktuálního elementu v datové větě, na kterém je tento concatenate volán, zapsán stringový součet jednotlivých elementů.

```
{var doc=GetDocument();  
  var elem=doc.GetElement();  
Result(elem.XPath("/d:root/d:red")+  
"+elem.XPath("/d:root/d:green")+  
"+elem.XPath("/d:root/d:blue")); }
```

Příklad na součet hodnot dvou elementů:

Tento příklad ukazuje možnost zapsat číselný součet pomocí javascriptu. Výsledek je opět zapsán do aktuálního elementu, v jehož výpočtové vazbě je tento výraz volán.

```
{  
var doc = GetDocument();  
var ele = doc.GetElement();  
var a = parseInt(ele.XPath("../d:a"));  
var b = parseInt(ele.XPath("../d:b"));  
Result(""+(a+b));}
```

Spuštění modálního okna Filleru:

Příklad na spuštění modálního okna Filleru pomocí metody `OpenModalFiller`. Výraz by měl být umístěn na tlačítku, po jehož stisku se spustí modální instance Filleru s formulářem `SignMaster`.

```
{var doc=GetDocument().OpenModalFiller(new Variant(VarStr,  
"<d:root xmlns:d='http://www.software602.cz/signmastercz'></d:root>"),  
new Variant());}
```

Podmínka

V tomto příkladu je použito větvení pomocí javascriptového příkazu `switch()`. Uvnitř jednotlivých případů jsou ještě navíc použity podmínky.

```
{  
var doc = GetDocument();  
var ele = doc.GetElement();  
var s = ele.XPath("/d:root/d:switch");  
switch (+s)  
{  
case 1:  
var a = doc.GetOpenFileNameDialog();  
if (a!=""){  
doc.OpenTab(new Variant(VarFN,a),new Variant(),OPEN_READONLY);  
break;}  
else{  
break;}  
case 2:  
var a = MessageBox("Pozor! Otevřený soubor bude po zavření automaticky  
smazán!", "Pozor!", MB_ICONWARNING|MB_OKCANCEL);  
if (a==1)  
{  
var b = doc.GetOpenFileNameDialog();  
if (b!=""){  
doc.OpenTab(new Variant(VarFN,b),new Variant(),OPEN_AUTODELETE);  
break;}  
else {  
break;}  
}  
else  
{  
break;
```

```
}  
case 3:  
var a = doc.GetOpenFileNameDialog();  
if (a!=""){  
doc.OpenTab(new Variant(VarFN,a),new Variant(),OPEN_NOACTIVATE);  
break;}  
else{  
break;}  
default:  
MessageBox("Vyber flag","Upozornění",MB_OK);  
break;  
}  
}
```

Cyklus

Tento příklad ukazuje použití jednoduchého cyklu pomocí příkazu for. Konkrétně slouží k uložení všech binárních příloh z opakovací sekce.

```
{var doc=GetDocument();  
var ele=doc.GetElement();  
  
a=0;  
var opak1=ele.XPathExpr("count(//*[@form602ct="bin"])");  
  
for (i=1;i<=opak1;i++){  
ele.SaveBinData("/fo:root/fo:page-  
sequence[1]/fo:flow[1]/fo:block[1]/fo:table[1]/fo:table-body[1]/fo:table-  
row["+i+"]/fo:table-  
cell[1],"c:\\a\\"+ele.XPath("../d:pril1_gr/d:pril1_item["+i+"]/d:pril1_file_  
name"));  
a++;  
} }
```

Otázky a odpovědi

Adresování hodnoty elementu z datové větě pomocí Xpath v metodách a funkcích

K tomuto slouží metoda elementu XPath, podobně jako ve starém mechanismu. Příklad získání cesty k uložení souboru z datové větě formuláře.

```
{var doc = GetDocument();  
var ele = doc.GetElement();  
var kam = ele.XPath("/d:root/d:kam");  
doc.SaveAsSigned(kam);}
```

Pozor na datový typ parametru funkcí a metod

Funkce a metody mají jasně deklarovaný datový typ parametru. V případě použití nesprávného debug panel zhlásí špatný typ parametru a je nutné použít správný. Například MessageBox, jako parametr lze použít pouze string, tedy není možné použít například číslo. Parametr lze převést na string pomocí:

```
{var a=1234;  
MessageBox(a,"Číslo sekce",MB_OK);}
```

Toto nebude fungovat, protože parametr funkce není string, ale hodnota. Lze převést například takto:

```
{var a="1234";  
MessageBox(a,"Číslo sekce",MB_OK);}
```

Nebo takto:

```
{var a=1234;  
MessageBox(a+"","Číslo sekce",MB_OK);}
```

Variant

Variant je zvláštní typ parametru. Nemá obecně deklarovaný datový typ, ale je na vývojáři jej specifikovat. Jeho možné hodnoty lze nalézt v dokumentaci k funkcím.

```
{  
var doc = GetDocument();  
var nazev = doc.GetOpenFileNameDialog();  
doc.Open(new Variant(VarFN,nazev),new Variant(),OPEN_NODLG);  
}
```

Result

Tato funkce se používá vždy, když je potřeba získat výsledek z výpočtu. Pokud je potřeba získat výsledek vyhodnocení.

```
{  
  var doc = GetDocument();  
  var elem = doc.GetElement();  
  
  Result(RCCheck(elem.XPath("/d:root/d:rc"))?"OK":"Bad");  
}
```

Převody na string a int

Převod hodnoty na string se provede pomocí `""+proměnná`, a převod stringu na int se provede pomocí `+proměnná`.

Kontrola výrazu

V zápisu výrazu na akci – tedy na tlačítko, nelze provést kontrolu výrazu.

Odesílací profily

Odesílací profily lze volat podobně jako v starém mechanismu díky použití příslušných metod a parametrizováním pomocí jména vytvořeného profilu.

```
{ var doc=GetDocument();  
var submit = doc.GetNamedSubmit("file_save") ;  
submit.Execute() ; }
```