Events im OLCAP-HTML-Dokument

OLCAP-Client-Applikationen unterstützen unterschiedliche Dokumenttypen. Unter diesen Typen sind auch ganz normale HTML-Dokumente. In der Client-Applikation wird dazu ein Internet Explorer ActiveX Control verwendet, welches das HTML-Dokument öffnet. Dabei sind auch Links zu tatsächlichen Homepages möglich. Dieser Dokumenttyp wird gerne in der Praxis zur Erstellung von Monitoren verwendet.

Grundsätzlich können solche Seiten jeden beliebigen HTML-Code beinhalten. Der eigentliche Datenaustausch mit der Client-Applikation bzw. mit dem OLCAP-Server erfolgt über Scripts in Javascript oder VBScript. Innerhalb der Funktionen eines Scripts werden die dazu passenden COM-Objekte erstellt und ausgeführt. Auf diese Art und Weise wird die HTML-Seite mit Daten gefüllt. Auch Technologien wie Silverlight oder Flash können in die HTML-Dokumente integriert werden.

# Eventsteuerung

Mit bisher besprochenen Techniken können zwar die HTML-Dokumente mit Leben gefüllt werden, jedoch bekommen die Dokumente nicht mit, wenn der Anwender beispielsweise eine Datenbanksitzung in der Client-Applikation wechselt oder wenn der Server neue Daten für den Client bereitstellt. Möchte man auf solche Events aktiv reagieren, bedarf es weiterer Komponenten, die man in die bestehende HTML-Seite einbauen muss.

Damit die Client-Applikation überhaupt weiß, dass die HTML-Seite über ein entsprechendes Eventhandling verfügt, bedarf es eines vordefinierten Textes im Title Tag der HTML-Seite. Man fügt dazu zum bestehenden Title den Text ***PP Script*** ein. Damit weiß die Client-Applikation, dass die HTML-Seite auf Events reagieren wird. Tritt ein Event am Client auf, wird es automatisch an den Script-Teil der HTML-Seite übergeben.

Als nächstes werden in die HTML-Seite die notwendigen Einsprungfunktionen implementiert. Es müssen im Folgenden alle Funktionen zumindest mit den Funktionsrümpfen verfügbar sein. Trifft dies nicht zu, kann es zum Absturz des Clients kommen. Folgende Funktionen müssen in der HTML-Seite eingefügt sein.

function OnMessageForActiveListener(nType, nElementId, nDbIndex, nPointx, nPointy, sText){}

function OnRefreshClient(nDBHandle, nRefreshType, sMessage){}

function OnRefreshClientEx(nDBHandle, nRefreshType, sMessage, DBInfo){}

function OnUpdateSession(nType, nKind, nDBIndex, bChangeActive) {}

Diese Code-Rümpfe sind in Javascript ausgeführt. Innerhalb dieser Code-Rümpfe kann man nun die eigenen Aktualisierungen durchführen.

# Eventfunktionen

Die Beschreibung der einzelnen Funktionen mit ihren Parametern erfolgt im Anschluss.

## OnMessageForActiveListener

Diese Funktion wird immer dann aufgerufen, wenn ein allgemeines Event am Client anliegt. Die Funktion hat folgende Parameter:

**nType** Eventtype (int)

Der Eventtyp gibt an, wer den Event ausgelöst hat, und kann folgende Ausprägungen haben:

* 10000 aktiver Datenbanksitzungswechsel
* 10001 Eintragung in einer Datenbanksitzung geändert
* 10010 Doubleclick (DBLClk) auf Gridcontrol
* 10011 DBLClk oder Drag&Drop mit linker Maustaste aus dem Organisationsbaum
* 10012 DBLClk oder Drag&Drop mit rechter Maustaste aus dem Organisationsbaum
* 10013 DBLClk oder Drag&Drop mit linker Maustaste aus dem Zeitbaum
* 10014 DBLClk oder Drag&Drop mit rechter Maustaste aus dem Zeitbaum
* 10015 DBLClk oder Drag&Drop mit linker Maustaste aus dem Sitzungsbaum
* 10016 DBLClk oder Drag&Drop mit rechter Maustaste aus dem Sitzungsbaum
* 10017 Periodenauswahl aus der Periodenauswahlcombobox (Toolbar)
* 10018 Strukturauswahl aus der Strukturauswahlbox (Toolbar)

**nElementId** Selektierte Strukturelement-ID des Steuerungselements, das den Event ausgelöst hat.

**nDBIndex** Aktive Datasetnummer

**xPoint, yPoint** Derzeit nicht verwendet, immer 0.

**sText** Textinhalt des ausgewählten Elements aus dem Baum, der den Event ausgelöst hat.

## OnRefreshClient

Dieser Event wird ausgelöst, wenn am OLCAP-Server neue Daten vorliegen. Die Funktion hat folgende Parameter:

**nDBHandle** Verweis auf das Dataset, welches den Refresh ausgelöst hat. Achtung: nicht identisch mit der Datasetnummer aus dem Sitzungsbaum.

**nRefreshType** Art des Refresh. Der Refresh-Typ ist bitcodiert und kann folgende Ausprägungen (auch kombinierte) haben.

* REFRESH\_NORMAL 0x0001 Werte erneuern
* REFRESH\_STRUCTURE 0x0002 Strukturelement gelöscht oder neu hinzugefügt
* REFRESH\_INDEX 0x0004 Indexwert wurde verändert
* REFRESH\_TIMEINDEX 0x0008 Zeitindex wurde verändert.
* REFRESH\_STATISTIK 0x0010 Änderung von Statistikdaten
* REFRESH\_DBCOMMIT 0x0020 wird gesetzt wenn die alle Serverdaten in die SQL Datenbank zurückgeschrieben wurden.
* REFRESH\_DBADJUST 0x0040 wird bei Datenbankumstellung erzeugt
* REFRESH\_MESSAGE 0x0080 wird bei einer Fehlermeldung oder Nachricht gesetzt
* REFRESH\_MEMO 0x0100 Memoupdate
* REFRESH\_CHAT 0x0200 Nachrichtendienst

**sMessage** Optional mitgegebene Fehlermeldung

## OnRefreshClientEx

Dieser Event wird ausgelöst, wenn am OLCAP-Server neue Daten vorliegen. Die Funktion entspricht der Funktion OnRefreshClient, hat jedoch einen zusätzlichen Parameter. Sie hat insgesamt folgende Parameter:

**nDBHandle** Verweis auf das Dataset, welches den Refresh ausgelöst hat. Achtung: nicht identisch mit der Datasetnummer aus dem Sitzungsbaum.

**nRefreshType** Art des Refresh. Der Refresh-Typ ist bitcodiert und kann folgende Ausprägungen (auch kombinierte) haben.

* REFRESH\_NORMAL 0x0001 Werte erneuern
* REFRESH\_STRUCTURE 0x0002 Strukturelement gelöscht oder neu hinzugefügt
* REFRESH\_INDEX 0x0004 Indexwert wurde verändert
* REFRESH\_TIMEINDEX 0x0008 Zeitindex wurde verändert.
* REFRESH\_STATISTIK 0x0010 Änderung von Statistikdaten
* REFRESH\_DBCOMMIT 0x0020 wird gesetzt wenn die alle Serverdaten in die SQL Datenbank zurückgeschrieben wurden.
* REFRESH\_DBADJUST 0x0040 wird bei Datenbankumstellung erzeugt
* REFRESH\_MESSAGE 0x0080 wird bei einer Fehlermeldung oder Nachricht gesetzt
* REFRESH\_MEMO 0x0100 Memoupdate
* REFRESH\_CHAT 0x0200 Nachrichtendienst

**sMessage** Optional mitgegebene Fehlermeldung

**DBInfo** Optional mitgelieferte Struktur. Die Struktur enthält Daten über ein Dataset und hat folgenden Aufbau.

typedef struct \_DB\_INFO

{

unsigned short DBPath[ 256 ]; // Datasetpfad

unsigned short BudgetName[ 80 ]; // Datasetname

unsigned short Connectstring[ 256 ]; // Connectstring zum Dataset

unsigned short ServerName[ 128 ]; // OLCAP Servername

unsigned short Prefix[ 80 ]; //Verwendeter Vorspannname in der SQL Datenbank

int DBPlace; //optional nicht verwendet

int DBType; // Datasettyp

} DB\_INFO;

## OnUpdateSession

Die Funktion wird immer dann aufgerufen, wenn im Sitzungsbaum eine Änderung durchgeführt wird. Sie hat folgende Parameter:

**nType** Art der Nachricht. Dieser Parameter ist mit dem Parameter nType aus der Funktion OnMessageForActiveListener identisch.

**nKind** Gibt an, was geändert wurde und kann folgende Ausprägungen haben:

* CH\_ALL 1 //kompletter Neuaufbau
* CH\_DELETED 2 //Datenbank wurde gelöscht
* CH\_ADDED 3 //Datenbank wurde hinzugefügt
* CH\_DBNAME 4 //Datenbankname wurde geändert
* CH\_TRANSFER 5 //Datenbank wurde verschoben/kopiert
* CH\_SESSNAME 6 //Sitzungsname wurde geändert
* CH\_SESSDELETED 7 //Sitzung wurde gelöscht

**nDBIndex** Aktive Datasetnummer

**bActiveChange** Gibt an, ob die Änderung aktiv durch den Anwender (true) oder fremdgesteuert (false) ausgeführt wurde.

# Zusammenfassung

Beachten Sie bitte, dass folgende Bedingungen im HTML-Dokument gegeben sein müssen, damit ein einwandfreies Funktionieren der Eventfunktionen gewährleistet ist:

1. Im HTML Tag Title muss der Wortlaut ***PP Script*** vorkommen (Leerzeichen beachten).
2. Alle oben beschriebenen Eventfunktionen müssen zumindest mit ihren Funktionsrümpfen in Jacascript implementiert sein. Allerdings müssen nicht alle Eventfunktionen einen weiteren Code ausführen.
3. Die Eventfunktionen werden nur dann ausgeführt, wenn das HTML-Dokument innerhalb der Client-Applikation gestartet wird. Ein Starten in einem externen Browser bedient die Eventfunktionen nicht.
4. Bei einem Event können auch mehrere Eventfunktionen gleichzeitig ausgelöst werden. Funktional werden die Funktionen aber sequenziell aufgerufen.
5. Innerhalb des Javascript-Codes kann sowohl die Client-Applikation als auch die OLCAP-Server-Applikation über COM-Objekte angesprochen werden. Beachten Sie dabei bitte, dass dies wiederum einen Event auslösen kann, was zu einer Rekursion führen könnte. Die Applikation würde dadurch in eine Endlosschleife geraten und letztendlich einen Stack Overflow erzeugen.

In der Praxis hat es sich bewährt, die beiden Eventfunktionen OnMessageForActiveListener (für Events vom Client) und OnRefreshClient (für Events vom Server) zu verwenden. Die beiden anderen Funktionen liefern lediglich detaillierter Daten, die in der Regel allerdings nicht benötigt werden.