Stored Procedures & Triggers

Introduzione

Le *Stored Procedures* così come i *Trigger* non fanno parte della natura del server eXtraWay ma ne rappresentano una forma dinamica d'estensione.
Da molto tempo a questa parte esiste la possibilità di realizzare una “DLL d'archivio”, vale a dire una libreria dinamica che viene chiamata dal server, se presente e configurata, per svolgere azioni specializzate in alcuni punti ben definiti[1)] per i quali sono stati realizzati degli *entry point* detti Morsetti.
Col passare del tempo è risultato evidente che una simile realizzazione scarica su 3D Informatica tutto l'onere dello sviluppo e della manutenzione anche per azioni tutto sommato molto semplici, da svolgersi sia per completare/perfezionare i record che vengono salvati sia per *decorare* quelli che vengono mostrati.
Per questa ragione, sempre con la logica della libreria dinamica da evocare al momento giusto nel modo giusto, è stata realizzata una coppia di librerie dinamiche, liblua.[dll|so] e libluafunc.[dll|so], che assolve a questi compiti.

Componenti dell'architettura software

Le due librerie devono essere collocate come segue:

File	Collocazione
liblua.[dll|so]	Directory `plug-in` presente dentro alla directory degli eseguibili eXtraWay, ovvero nella directory ove si trova il server.
xwlua.lua	Stessa collocazione di liblua.[dll|so]. Questo file rappresenta il collegamento tra i comandi base citati nella successiva Reference
libluafunc.[dll|so]	Directory dei binari di eXtraWay, quindi la stessa in cui si trova il server.

Linguaggio di Scripting

Per introdurre *Stored Procedures* e *Triggers* tra le funzionalità del Server eXtraWay abbiamo ritenuto di doverci avvalere di un linguaggio di *scripting* potente e di ampia diffusione.
Per tale ragione la scelta è caduta su Lua.
Si assume quindi che il lettore che voglia intraprendere la stesura di proprie *Stored Procedures* o *Triggers* acquisisca una ragionevole padronanza del linguaggio avvalendosi del suo Tutorial o del Reference.

Registrazione di una Stored Procedure o di un Trigger

Perché siano disponibili presso un archivio, le *Stored Procedure* ovvero i *Trigger* devono essere opportunamente registrati presso l'archivio stesso.

Nel file di configurazione dell'archivio devono essere previste le due tipologie documentali che descrivono ciò che verrà associato a queste azioni.
Si veda il seguente esempio

```
  <primary_node ud_name="xwStoredProcedure" ud_container="xwStoredProcedureList">
     <unique_rule search_rule="[XML,/xwStoredProcedure/@package]"/>
     <key name="XML,/xwStoredProcedure/" key_style="skip" path_style="container"/>
     <key name="XML,/xwStoredProcedure/@package" key_style="single"/>
     <key name="XML,/xwStoredProcedure/func/@id" key_style="single"/>
     <file_location mode="rule" rule="stored$/@XML,/xwStoredProcedure/@package@$;stored"
move_always="yes"/>
  </primary_node>
  <primary_node ud_name="xwTrigger" ud_container="xwTriggerList">
     <unique_rule search_rule="[XML,/xwTrigger/@package]"/>
     <key name="XML,/xwTrigger/" key_style="skip" path_style="container"/>
     <key name="XML,/xwTrigger/@package" key_style="single"/>
     <file_location mode="rule" rule="trigger$/@XML,/xwTrigger/@package@$;trigger"
move_always="yes"/>
  </primary_node>
```

Come si può notare, entrambe le tipologie prevedono di essere identificate tramite un `@package` e, nel solo caso delle *Stored Procedures*, ciascuna funzione appartenente ad un *Package* avrà un nome ben preciso.

Iniziamo col descrivere le *Stored Procedures* in quanto i *Trigger* sono sostanzialmente identici ma con dei vincoli intrinsechi.

Stored Procedures

Vediamo un esempio di record di configurazinoe di una *Stored Procedure*

```
<xwStoredProcedure package="unifolder.decorator" xmlns:xw="http://www.3di.it/ns/xw-200203121136">
  <func id="loadRecordByNRecord">
```

```
    <parms>
      <in>
        <parm name="nRecord"/>
      </in>
      <out>
        <parm name="result"/>
      </out>
    </parms>
  </func>
  <func id="loadRecordByID">
    <parms>
      <in>
        <parm name="nDoc"/>
      </in>
      <out>
        <parm name="result"/>
      </out>
    </parms>
  </func>
  <xw:file name="./lua/decorator.lua"/>
</xwStoredProcedure>
```

Quello che ci dice questo record XML di configurazione è che esiste un *Package*, denominato unifolder.decorator sono presenti due *Stored Procedures*, rispettivamente loadRecordByNRecord e loadRecordByID.
Ciascuna *Stored Procedure* prevede uno o più parametri in ingresso ed in uscita[2)].
ciascuno parametro ha un name cui viene associato un valore verboso e può avere un valore sottinteso (default). In tal caso sarà presente un attributo default atto ad indicare tale valore.
In assenza del valore di default il parametro viene inizializzato con un valore nullo. Questo vale solo per i parametri in ingresso in quanto per quelli in uscita non è previsto un valore predefinito.

```
<xwStoredProcedure    package="unifolder.util">
  <func id="closeFolder">
    <parms>
      <in>
        <parm name="nRecord"/>
        <parm default="0" name="force"/> <!-- Valore di default del parametro 'force' -->
      </in>
      <out>
        <parm name="result"/>
      </out>
    </parms>
  </func>
  <xw:file name="./lua/decorator.lua"/>
 </xwStoredProcedure>
```

Al record XML di configurazione è associato uno ed un solo allegato di tipo LUA che deve contenere tutte le *Stored Procedures* indicate[3)].

Secondo quanto detto, quindi, ogni *Stored Procedure* è identificabile per mezzo del suo *fully qualified name*. Nel nostro esempio avremo unifolder.decorator.loadRecordByNRecord e unifolder.decorator.loadRecordByID.

Triggers

Vediamo un esempio di record di configurazinoe di una *Trigger*

```
<xwTrigger package="doc.beforeSave" xmlns:xw="http://www.3di.it/ns/xw-200203121136">
  <func id="docCheck"/>
  <xw:file name="./lua/doctrigger.lua"/>
</xwTrigger>
```

Le analogie con le *Stored Procedures* sono numerose ma con l'introduzione di una serie di vincoli.
Mentre per le *Stored Procedures* il nome del package è del tutto libero, il nome package di un *Trigger* è la combinazione della denominazione del *Primary Node Element* cui il *Trigger* si riferisce e l'evento cui il *Trigger* risponde.
Oltre a questo, i parametri in ingresso ed uscita sono fissati dalla natura del *Trigger* stesso e quindi non vanno elencati in questo record.
In assenza di un *Trigger* per un determinato evento occorso su un determinato tipo record, nessuna funzionalità verrà eseguita e la cosa in se non costituisce errore.

Vengono invece elencati i nomi di tutte le funzioni che devono essere chiamate e che verranno eseguite nello stesso ordine in cui gli elementi `func` sono presenti nell'XML, questo nell'ipotesi che si possano dover compiere diverse operazioni[4)].
In presenza di molteplici chiamate in sequenza, i valori di output della funzione precedente possono essere usati per valorizzare i parametri in input della funzione successiva.

Anche in questo caso, al record XML di configurazione è associato uno ed un solo allegato di tipo LUA che deve contenere tutte le funzioni citate dal *Trigger* indicate. Per esso valgono le stesse considerazioni fatte per le *Stored Procedures*.

Secondo quanto detto quindi questo *Trigger* risponde all'evento beforeSave sul tipo record doc evocando la funzione docCheck.

Esempi

Vediamo rapidamente un esempio di come dev'essere realizzato il codice LUA prendendo spunto dal *Trigger* `docCheck`.
Le funzioni utilizzate sono meglio descritte nella successiva Reference.

```
function docCheck(toBeSaved, numDoc)

    local newRecord = xw.prepareRecord(toBeSaved, nil) ;
    local codFasc = newRecord:getFirstNodeValue("//doc/rif_interni/rif/@cod_fasc", nil) ;
    local docType = newRecord:getFirstNodeValue("//doc/extra/content/@docType", nil) ;

    if ( 0 ~= #codFasc ) then -- only if a valid cod_fasc is present.

        local isComplete = isDocumentComplete(newRecord) ;

        -- Fascicolo will be changed anyway since i must set/reset record state each time it
changes.
        local nodeNRecord = newRecord:selectFirstNode("//doc/@nrecord", nil) ;
        if ( nodeNRecord:isValid() ) then
            local nRecordValue = nodeNRecord:getValue() ;
            -- Got the NRecord of this "doc". Look for the corresponding "fascicolo" and change
it.
            local fascicolo =
xw.loadFirstRecordByQuery("([/fascicolo/@numero/]=\""..codFasc.."\")", "", "") ;
            fascicolo:lock() ;
            local docRefNode =
fascicolo:selectFirstNode("//fascicolo/extra/content/documents/document/ref[@nrecord='"..nRecordVa
lue.."']", nil) ;
            -- Define "complete" attribute value
            local completeStr = xw.choose((0 ~= isComplete), "1", "0") ;
            log.msg("This record completeness: "..completeStr) ;
            -- When saving the 'fascicolo' record I will check for completeness.
            if ( docRefNode:isValid() ) then
                -- Got it!
                docRefNode:setAttribute("complete", completeStr) ;
                fascicolo:save() ;
            else
                -- add the document reference lost.
                docRefNode =
fascicolo:selectFirstNode("//fascicolo/extra/content/documents/document[@id='"..docType.."']",
nil) ;
                if ( docRefNode:isValid() ) then
                    log.error("Document ref "..nRecordValue.." still not present in 'fascicolo'
"..codFasc) ;
                    local refNode = docRefNode:appendChild("ref", "") ;
                    refNode:setAttribute("nrecord", nRecordValue) ;
                    refNode:setAttribute("complete", completeStr) ;
                    fascicolo:save() ;
                else
                    log.error("No document of type '"..docType.."' are expected in 'fascicolo'
"..codFasc.."!") ;
                    fascicolo:unlock() ; -- No condition to change 'fascicolo' record.
                end
            end
        end
    end
    return "" ; -- The doc record will be unchanged.
```

```
end
```

La funzione determina in primo luogo due valori del presente record di tipo `doc`, valori necessari per conoscere a quale fascicolo esso appartenga e quale sia la sua natura.
Solo qualora il documento sia associato ad un fascicolo si procede a richiamare una funzione, isDocumentComplete(newRecord) che compie delle verifiche sul documento stesso.
Quale che sia l'esito della suddetta verifica, si procede a ricercare il fascicolo avente corrispondente codFasc assumendo di identificarne uno ed uno soltato. In esso si determina la presenza di un riferimento al documento corrente (identificato tramite il suo nrecord.
A questo punto ci sono due casi: il fascicolo cita correttamente questo documento, ed in tal caso si provvede ad aggiungere al nodo del riferimento un attributo complete col giusto valore ('0' o '1'), oppure il fascicolo non cita ancora questo documento[5].
In questo secondo caso, si identifica la porzione del fascicolo ove sono raccolti i riferimenti alle topologie di documento corrispondenti al tipo del documento corrente e si correda tale nodo di un nuovo elemento ref che costituisca il legame con il presente documento tramite il suo nrecord apponendo analogamente al caso precedente anche la condizione di completezza nell'attributo complete.

Se quanto detto va regolarmente a buon fine, il fascicolo così modificato viene salvato. Si noti che il salvataggio del record di tipo fascicolo può a sua volta causare l'esecuzione di un *Trigger* per un proprio evento. Chi realizzerà i *Trigger* deve tenere conto del rischio di causare una eventuale condizione di *loop* o un effetto a catena.

"Reference" dei metodi e dei tipi disponibili

La stesura di una *Stored Procedure* ovvero di un *Trigger* in Lua può essere effettuata avvalendosi dei seguenti metodi e tipi complessi.

Iniziamo dalla descrizione dei tipi complessi che si sommano ai tipi standard di Lua.

Tipo	Finalità
xwLuaRecord_t	Rappresenta un singolo record di un archivio eXtraWay. Esso può essere sia un record esistente al quale si accede sia un record del tutto nuovo che potrà essere salvato nell'archivio.
xwLuaRecordSet_t	Rappresenta un insieme di `xwLuaRecord_t`, tipicamente esito di un'operazione di selezione sull'archivio eXtraWay.
xwLuaNode_t	Dal momento che i `xwLuaRecord_t` sono degli XML essi sono composti da *nodi* di un *DOM*. Il `xwLuaNode_t` rappresenta proprio uno di tali nodi, nella più generica delle accezioni possibili.
xwLuaNodeSet_t	Analogamente ai casi precedenti, un `xwLuaNodeSet_t` rappresenta un insieme di `xwLuaNode_t`, normalmente frutto dell'applicazione di un *XQuery* sul record.

Ciascuno di tali tipi evoluti ha metodi propri, così come esistono metodi generali, riferiti tipicamente a chiamate dirette al server eXtraWay.

Evocare un metodo di un tipo evoluto prevede una sintassi lievemente differente da quella da adottare per evocare un metodo generico.
Mentre il metodo generico prevede la sintassi...

`<namespace>.<nmetodo>(…)`

...il metodo evoluto richiede la forma...

`<tipo evoluto>:<metodo>(…)`

...per compiere una chiamata analoga.

Si suggerisce quindi la massima attenzione nell'uso di **`.`** e **`:`**.

Di seguito il *reference* di dettaglio dei metodi con una notazione Lua.

Legenda

Per ciascuno dei seguenti metodi verranno utilizzati i seguenti stili:

Il colore verde	Indica il nome del metodo
Il colore rosso	Indica i parametri obbligatori dei metodi
Il colore arancio	Indica per il parametri facoltativi dei metodi Per i parametri facoltativi, essi possono essere impostati a "" ovvero nil ovvero omessi la dove la sintassi della chiamata del metodo lo consenta

Metodi del Namespace 'xw'

user defined type xw.choose(<espressione Lua>, <true value>, <false value>)

Questo metodo intende semplificare alcuni processi di verifica evitando l'onere di scrivere un'espressione completa. Provvede a verificare il valore di cui ad `espressione Lua`. Se tale valore è lecito, corretto, in una parola vero, si torna il `true value`, altrimenti il `false value`.	
param <espressione Lua>	L'espressione da valutare
param <true value>	Valore tornato in caso di esito positivo
param <false value>	Valore tornato in caso di esito negativo
return	Torna il `true value` o il `false value`
example	return xw.choose(nil==value, "", value) ; return xw.choose(record:isValid(), record, xw.prepareRecord("<?xml version="1.0"?><doc/>") ;

xwLuaRecordSet_t xw.executeQuery(<String> query, <String> sortRule, <String> arcName)

Compie una selezione, eventualmente ordinata, sull'archivio eXtraWay. L'esito è rappresentato da un `xwLuaRecordSet_t` eventualmente vuovo.	
param <String> query	L'espressione di ricerca sulla base della quale compiere la selezione
param <String> sortRule	La regola di ordinamento
param <String> arcName	Il nome, logico o fisico, dell'archivio su cui svolgere la ricerca se diverso da quello corrente
return <xwRecord_t>	Torna l'insieme dei record selezionati dall'espressione di ricerca impostata
see	xw.refineQuery()

String xw.getCurrentArcName()

return <String>	Torna la stringa corrispondente al nome completo dell'archivio per il quale è stata registrata la `Stored Procedure` o il `Trigger` in corso di esecuzione

xwLuaRecord_t xw.loadFirstRecordByQuery(<String> query, <String> sortRule,<String> arcName)

param <String> query	L'espressione di ricerca sulla base della quale compiere la selezione
param <String> sortRule	La regola di ordinamento
param <String> arcName	Il nome, logico o fisico, dell'archivio su cui svolgere la ricerca se diverso da quello corrente
return <xwRecord_t>	Torna il primo dei record selezionati dall'espressione di ricerca impostata o un record non valido
see	xw.executeQuery(), xw.loadRecordById()

xwLuaRecord_t xw.loadRecordById(<String> Id, <String> arcName)

param <String> Id	L'identificatore (numero) fisico del documento che si intende caricare
param <String> arcName	Il nome, logico o fisico, dell'archivio su cui svolgere la ricerca se diverso da quello corrente
return <xwRecord_t>	Torna il record richiesto se presente nell'archivio e non cancellato o un record non valido
see	xw.loadRecordById()

xwLuaRecord_t xw.prepareRecord(<String> XML_Text, <String> arcName)

Consente la creazione di un tipo `xwRecord_t` da utilizzarsi come strumento transitorio o per compiere un salvataggio sull'archivio eXtraWay. L'impostazione dell'archivio, facoltativa, è proprio finalizzata ad un successivo salvataggio e quindi può essere omessa ma se assente si assume comunque l'archivio corrente.	
param <String> XML_Text	Una stringa che contenga una porzione di XML valida sulla quale costituire il record
param <String> arcName	Il nome, logico o fisico, dell'archivio su cui svolgere la ricerca se diverso da quello corrente
return <xwRecord_t>	Torna il record costituito sulla base del testo fornito

xwLuaRecordSet_t xw.refineQuery(<xwLuaRecordSet_> recordSet, <String> query, <String> sortRule, <String> arcName)

Raffina l'esito della ricerca precedentemente compiuta e registrata in `recordSet` con un'ulteriore espressione di ricerca e torna un nuovo `xwLuaRecordSet_t` contenente l'insieme dei record selezionati.	
param <String> recordSet	La precedente selezione da raffinare
param <String> query	L'espressione di ricerca sulla base della quale compiere il raffinamento
param <String> sortRule	La regola di ordinamento
param <String> arcName	Il nome, logico o fisico, dell'archivio su cui svolgere la ricerca se diverso da quello corrente
return <xwRecord_t>	Torna l'insieme dei record raffinati applicando l'espressione di ricerca impostata
see	xw.executeQuery()

Metodi del Namespace 'log'

void log.error(<Sting> errMsg)

Registra quanto indicato in `errMsg` nel log del server con una registrazione di tipo `[E]`.	
param <String> errMsg	Il messaggio da porre nel log

void log.msg(<Sting> infoMsg)

Registra quanto indicato in `infoMsg` nel log del server con una registrazione di tipo `[I]`.	
param <String> infoMsg	Il messaggio da porre nel log

void log.warning(<Sting> warnMsg)

Registra quanto indicato in `warnMsg` nel log del server con una registrazione di tipo `[W]`.	
param <String> warnMsg	Il messaggio da porre nel log

Metodi del Tipo 'xwLuaRecordSet_t'

numeric #recordSet

Torna la dimensione del *Record Set* ovvero il numero di record che esso cita.	
return <numeric>	Numero di xwLuaRecord_t presenti nel *Record Set*. Il valore '0' indica un *Record Set* vuoto o non valido.

xwLuaRecord_t recordSet:getAbsolute(<numeric> Position)

Torna il xwLuaRecord_t richiesto. Il numero da esprimere dev'essere compreso tra '1' e #recordSet.	
return <xwLuaRecord_t>	Il xwLuaRecord_t richiesto, se disponibile, ovvero un xwLuaRecord_t non valido se la posizione indicata non è valida entro il xwLuaREcordSet_t. L'esito del comando va sempre verificato.
see	recordSet:getCurrent(), recordSet:getFirst(), recordSet:getLast(), recordSet:getNext(), recordSet:getPrev()

xwLuaRecord_t recordSet:getCurrent()
xwLuaRecord_t recordSet:getFirst()
xwLuaRecord_t recordSet:getLast()
xwLuaRecord_t recordSet:getNext()
xwLuaRecord_t recordSet:getPrev()

Torna il xwLuaRecord_t richiesto. La prassi d'utilizzo è una chiamata a `getFirst()` seguita da una o più chiamate a `getNext()`, ovvero una chiamata a `getLast()` seguita da una o più chiamate a `getPrev()`. Questo in quanto, in principio, il xwLuaRecordSet_t non ha una *posizione attuale* cui fare riferimento. Il metodo `getCurrent()` può essere utilizzato solo dopo un posizionamento.	
return <xwLuaRecord_t>	Il xwLuaRecord_t richiesto, se disponibile, ovvero un xwLuaRecord_t non valido. L'esito del comando va sempre verificato.
see	recordSet:getAbsolute()

boolean recordSet:isValid()

Indica se il *Record Set* è valido o meno. La validità si ha in presenza di un *Record Set* non vuoto. Non vi è modo di distinguere se esso dipenda da un'espressione di ricerca errata o semplicemente da un esito nullo.	
return <boolean>	`true` per *Record Set* valido, `false` in caso contrario

xwLuaRecordSet_t recordSet:refine(<String> searchExpression, <String> sortRule)

Genera un nuovo xwLuaRecordSet_t da quello corrente filtrando ulteriormente i suoi contenuti sulla base della `searchExpression` ed applicando, eventualmente, una `sortRule`. Il xwLuaRecordSet_t originale rimane invariato così come rimane invariata la posizione eventualmente raggiunta in esso.	
return <xwLuaRecordSet_t>	Il xwLuaRecordSet_t frutto del raffinamento. L'esito del raffinamento può produrre un *Result Set* vuoto.
see	xw.refineQuery()

xwLuaRecordSet_t recordSet:sort(<String> sortRule)

Genera un nuovo xwLuaRecordSet_t da quello corrente applicando una `sortRule`. Il xwLuaRecordSet_t originale rimane invariato così come rimane invariata la posizione eventualmente raggiunta in esso.	
return <xwLuaRecordSet_t>	Il xwLuaRecordSet_t frutto dell'ordinamento.

Metodi del Tipo 'xwLuaRecord_t'

numeric record:bindToArc(<String> arcName)

Assegna al record l'archivio al quale esso è destinato per successive operazioni di salvataggio. Il suo campo d'applicazione interessa sia i record generati ex-novo, sia record caricati da un archivio eXtraWay che si intenda salvare su un archivio differente. In ambo i casi, il numero fisico del record eventualmente impostato nel record stesso[6] viene annullato ed il salvataggio del record avrà luogo in forma di inserimento.	
param <String> arcName	Il nome dell'archivio cui viene associato il record.
return <numeric>	Torna 'non 0' in caso di successo e '0' in caso contrario
see	xw.prepareRecord(), record:save()

String record:getArc()

Torna il nome dell'archivio cui è stato associato il record corrente.

return <String>	Il nome dell'archivio
see	record:bindToArc(), record:getId()

String record:getFirstNodeValue(<String> xPath, <xwLuaNode_t>node)

Esegue sul xwLuaRecord_t indicato una XQuery basata sul XPath indicato e torna il valore del primo dei nodi identificati o una stringa vuota se l'espressione non conduce ad un set valorizzato E' possibile esprimere un nodo del DOM del record dal quale partire come parametro opzionale	
param <String> xPath	L'espressione da applicare sul DOM del record.
param <xwLuaNode_t> node	Nodo di origine al quale applicare il parametro xPath. In sua assenza, ovvero indicando un nodo non valido, l'espressione viene applicata all'intero record.
return <String>	Il contenuto del primo nodo identificato se disponibile
see	record:selectNodes(), record:selectFirstNode()

numeric record:getId()

Torna il numero fisico associato al record. Il numero fisico è un valore non nullo se il record è stato caricato da un archivio eXtraWay[7].	
return <numeric>	Il numero fisico assegnato al record ovvero '0' per record generati ex-novo o assegnati ad un nuovo archivio
see	record:bindToArc(), record:getArc(), xw.prepareRecord()

boolean record:isValid()

Indica se il *Record* è valido o meno. La validità si ha in presenza di un *Record* non vuoto, indipendentemente dal fatto che esso sia stato creato da un testo XML o caricato da un archivio eXtraWay.	
return <boolean>	`true` per *Record* valido, `false` in caso contrario

xwLuaRecord_t record:lock()

Impone il blocco del record corrente (che necessariamente dev'essere stato precedentemente caricato da un archivio eXtraWay). L'operazione può fallire qualora il record sia già stato bloccato da terzi.	
return <xwLuaRecord_t>	Torna il record stesso dopo averne compiuto il lock.

numeric record:save()

Salva il record. L'operazione può avvenire nei seguenti 3 modi: - Salvataggio in un archivio eXtraWay di un record del tutto nuovo cui sia stato assegnato un valido identificativo d'archivio; - Salvataggio in un archivio eXtraWay della modifica di un record preceduta dal suo bloccaggio; - Salvataggio in un archivio eXtraWay della modifica di un record che non sia stato precedentemente bloccato. Quest'ultima possibilità può fallire qualora il record sia stato modificato da terzi nell'intervallo di tempo che intercorre tra il suo caricamento e l'operazione di salvataggio. L'operazione di salvataggio condotta a termine con successo compie autonomamente lo sbloccaggio del record stesso.	
return <numerico>	Torna '0' in caso di esito positivo, ogni altro valore numerico rappresenta un codice d'errore da analizzare.
see	xw.prepareRecord(), record:bindToArc(), record:lock(), record:unlock()

xwLuaNode_t record:selectFirstNode(<String> xPath, <xwLuaNode_t>node)

Esegue sul xwLuaRecord_t indicato una XQuery basata sul XPath indicato e torna il primo dei nodi identificati o un nodo non valido se l'espressione non conduce ad un set valorizzato E' possibile esprimere un nodo del DOM del record dal quale partire come parametro opzionale	
param <String> xPath	L'espressione da applicare sul DOM del record.
param <xwLuaNode_t> node	Nodo di origine al quale applicare il parametro xPath. In sua assenza, ovvero indicando un nodo non valido, l'espressione viene applicata all'intero record.
return <xwLuaNode_t>	Un xwLuaNode_t corrispondente al primo nodo (attributo o elemento) corrispondente all'espressione data
see	record:selectNodes(), record:getFirstNodeValue()

xwLuaNodeSet_t record:selectNodes(<String> xPath, <xwLuaNode_t>node)

Esegue sul xwLuaRecord_t indicato una XQuery basata sul XPath indicato. I nodi identificati sono tornati in un xwLuaNodeSet_t eventualmente vuoto. E' possibile esprimere un nodo del DOM del record dal quale partire come parametro opzionale	
param <String> xPath	L'espressione da applicare sul DOM del record.
param <xwLuaNode_t> node	Nodo di origine al quale applicare il parametro xPath. In sua assenza, ovvero indicando un nodo non valido, l'espressione viene applicata all'intero record.
return <xwLuaNodeSet_t>	Un insieme dei nodi selezionati, eventualmente vuoto, in forma di xwLuaNodeSet_t
see	record:selectFirstNode(), record:getFirstNodeValue()

numeric record:setContentFromRecord(<xwLuaRecord_t> sourceRecord)

Annulla il DOM del record e lo sostituisce con una copia integrale del DOM del record indicato.	
param <xwLuaRecord_t> sourceRecord	Il record dal quale copiare il contenuto in quello corrente.

return <numeric>	Torna 'non 0' in caso di assegnazione corretta, '0' in caso contrario.
see	record:setContentFromString()

numeric record:setContentFromString(<String> sourceString)

Annulla il DOM del record e lo sostituisce con un DOM realizzato dalla stringa indicato.	
param <String> sourceString	La stringa dalla quale alimentare il record corrente.
return <numeric>	Torna 'non 0' in caso di assegnazione corretta, '0' in caso contrario.
see	record:setContentFromRecord()

numeric record:setNodeValue(<String> xPath, <xwLuaNode_t>node, <String> value)

Esegue sul xwLuaRecord_t indicato una `XQuery` basata sul `XPath` indicato. Per tutti i nodi identificati compie l'assegnazione del valore impostato ovvero assegna loro un valore vuoto. E' possibile esprimere un nodo del DOM del record dal quale partire come parametro opzionale	
param <String> xPath	L'espressione da applicare sul DOM del record.
param <xwLuaNode_t> node	Nodo di origine al quale applicare il parametro `xPath`. In sua assenza, ovvero indicando un nodo non valido, l'espressione viene applicata all'intero record.
param <String> node	Il valore da assegnare a tutti i nodi identificati.
return <numeric>	torna il numero dei nodi per i quali è stata compiuta l'assegnazione del valore.

String record:toString()

Torna il contenuto XML del record in forma di stringa.	
return <String>	Il contenuto XML del xwLuaRecord_t

void record:unlock()

Sblocca il record precedentemente bloccato	
see	record:lock(), record:save()

Metodi del Tipo 'xwLuaNodeSet_t'

numeric #nodeSet

Torna la dimensione del *Node Set* ovvero il numero di xwLuaNode_t che esso contiene.	
return <numeric>	Numero di xwLuaNode_t presenti nel *Node Set*. Il valore '0' indica un *Node Set* vuoto o non valido.

numeric nodeSet:erase()

Cancella dal DOM del record cui appartengono tutti i xwLuaNode_t contenuti nel *Node Set* e tutti i loro eventuali nodi figli.	
return <numeric>	Torna 'non 0' in caso di operazione completata, '0' in caso contrario, quindi anche in caso di rimozione parziale.

boolean nodeSet:isValid()

Indica se il *Node Set* è valido o meno. La validità si ha in presenza di un *Node Set* non vuoto. Non vi è modo di distinguere se esso dipenda da un'espressione xPath errata o semplicemente da un esito nullo.	
return <boolean>	`true` per *Node Set* valido, `false` in caso contrario

xwLuaNode_t nodeSet:getAbsolute(<numeric> Position)

Torna il xwLuaNode_t richiesto. Il numero da esprimere dev'essere compreso tra '1' e #nodeSet.	
return <xwLuaNode_t>	Il xwLuaNode_t richiesto, se disponibile, ovvero un xwLuaNode_t non valido se la posizione indicata non è valida entro il xwLuaNodeSet_t. L'esito del comando va sempre verificato.
see	nodeSet:getCurrent(), nodeSet:getFirst(), nodeSet:getLast(), nodeSet:getNext(), nodeSet:getPrev()

xwLuaNode_t nodeSet:getCurrent()
xwLuaNode_t nodeSet:getFirst()
xwLuaNode_t nodeSet:getLast()
xwLuaNode_t nodeSet:getNext()
xwLuaNode_t nodeSet:getPrev()

Torna il xwLuaNode_t richiesto. La prassi d'utilizzo è una chiamata a `getFirst()` seguita da una o più chiamate a `getNext()`, ovvero una chiamata a `getLast()` seguita da una o più chiamate a `getPrev()`. Questo in quanto, in principio, il xwLuaNodeSet_t non ha una *posizione attuale* cui fare riferimento. Il metodo `getCurrent()` può essere utilizzato solo dopo un posizionamento.	
return <xwLuaNode_t>	Il xwLusNode_t richiesto, se disponibile, ovvero un xwLuaNode_t non valido. L'esito del comando va sempre verificato.
see	nodeSet:getAbsolute()

Metodi del Tipo 'xwLuaNode_t'

numeric node:erase()

Rimuove il nodoindicato.	
return <numeric>	Torna 'non 0' in caso di rimozione effettuata, '0' in ogni altro caso.

see	node:removeAttribute()

boolean node:isValid()

Indica se il nodo è valido o meno. La validità si ha in presenza di un nodo correttamente frutto di una precedente operazione di selezione nel DOM.	
return <boolean>	`true` per *Node Set* valido, `false` in caso contrario
see	xw.selectFirstNode(), record:selectNodes()

<String> node:getAttribute(<String> attrName)

Torna il valore dell'attributo del nodo corrente avente il nome indicato.	
return <String>	Torna il contenuto dell'attributo. Non è possibile distinguere i casi in cui l'attributo esista ma ha valore vuoto da quello in cui l'attributo non esiste
see	node:hasAttribute(), node:hasAttributes(), node:getAttributes()

<xwLuaNodeSet_t node:getAttributes()

Torna l'elenco degli attributi di questo nodo.	
return <xwLuaNodeSet_t>	Torna un xwLuaNodeSet_t, eventualmente vuoto, contenente gli attributi del nodo dato
see	node:hasAttributes(), node:hasChildren(), node:getAttribute(), node:getChildren()

<xwLuaNodeSet_t node:getChildren(<numeric> childType)

Torna l'elenco dei nodi figli di questo nodo. Si deve indiacre il tipo di nodi richiesti	
param <numeric> childType	Indica di quali figli si faccia richiesta. I valori ammessi sono. - xwLuaNodeType_Node: Nodi generici [node()] - xwLuaNodeType_Comment: Nodi Commento [comment()] - xwLuaNodeType_PI: Nodi *Processing Instruction* [processing-instruction()] - xwLuaNodeType_Text: Nodi di tipo testuale [text()]
return <xwLuaNodeSet_t>	Torna un xwLuaNodeSet_t, eventualmente vuoto, contenente i nodi direttamente figli del nodo dato che rispettino il tipo richiesto
see	node:hasAttributes(), node:hasChildren(), node:getAttributes()

String node:getValue()

Torna il contenuto del nodo.	
return <String>	Il conenuto testuale del nodo. Se applicato a nodi *non foglia* torna il contenuto testuale del nodo e di tutti i nodi di tipo `elemento` suoi figli
see	node:setValue()

boolean node:hasAttribute(<String> attributeName)

Indica la presenza dell'attributo richiesto.	
return <boolean>	Torna 'true' se il nodo ha un attributo avente il nome indicato, 'false' in caso contrario
see	node:hasAttributes(), node:getAttribute()

boolean node:hasAttributes()

Indica la presenza di attributi in questo nodo.	
return <boolean>	Torna il numero di nodi direttamente figli del nodo dato che rispettino il tipo richiesto
see	node:hasAttribute(), node:hasChildren(), node:getAttribute(), <color green>node:getAttributes(), <color green>node:getchildren()

boolean node:hasChildren(<numeric> childType)

Indica la presenza di nodi figli di questo nodo. Si deve indiacre il tipo di nodi richiesti	
param <numeric> childType	Indica di quali figli si faccia richiesta. I valori ammessi sono. - xwLuaNodeType_Node: Nodi generici [node()] - xwLuaNodeType_Comment: Nodi Commento [comment()] - xwLuaNodeType_PI: Nodi *Processing Instruction* [processing-instruction()] - xwLuaNodeType_Text: Nodi di tipo testuale [text()]
return <boolean>	Torna il numero di nodi direttamente figli del nodo dato che rispettino il tipo richiesto
see	node:hasAttributes(), node:getChildren(), node:getAttributes()

xwLuaNode_t node:insertAfter(<String> nodeName, <String> value)
xwLuaNode_t node:insertBefore(<String> nodeName, <String> value)
xwLuaNode_t node:insertNodeAfter(<xwLuaNode_t> node)
xwLuaNode_t node:insertNodeBefore(<xwLuaNode_t> node)

Inserisce un nuovo nodo prima o dopo il nodo corrente, quindi come fratello immediatamente precedente o seguente. Il nodo può essere creato per <nome>-<valore> ovvero copiato da un nodo esistente	
param <String> nodeName	Il nome del nodo da creare

param <String> value	Il valore da assegnare al nodo crato
...ovvero...	
param <xwNode_t> node	Il nodo da copiare
return <xwLuaNode_t>	Torna il nodo creato
see	node:insertFirstChild(), node:appendChild(), node:insertFirstNodeChild(), node:appendNodeChild(), node:replace()

xwLuaNode_t node:insertFirstChild(<String> nodeName, <String> value)
xwLuaNode_t node:appendChild(<String> nodeName, <String> value)
xwLuaNode_t node:insertFirstNodeChild(<xwLuaNode_t> node)
xwLuaNode_t node:appendNodeChild(<xwLuaNode_t> node)

Inserisce un nuovo nodo figlio, come primo dei figlio o ultimo dei figli del nodo corrente. L'inserimento può avvenire per coppia <nome>-<valore> o per copia di un nodo esistente	
param <String> nodeName	Il nome del nodo da creare
param <String> value	Il valore da assegnare al nodo crato
...ovvero...	
param <xwLuaNode_t node	Il nodo da copiare
return <xwLuaNode_t>	Torna il nodo creato
see	node:insertAfter(), node:insertBefore(), node:insertNodeAfter(), <color green>node:insertNodeBefore(), node:replace()

numeric node:removeAttribute(<String> attrName)

Rimuove l'attributo indicato.	
param <String> attrName	Il nome dell'attributo da rimuovere
return <numeric>	Torna 'non 0' in caso di rimozione effettuata, '0' in ogni altro caso.
see	node:setAttribute()

xwLuaNode_t node:replace(<xwLuaNode_t> node)

Sostituisce il nodo corrente con una copia di quello indicato, portando con se eventuali nodi figli.	
param <xwLuaNode_t> node	Il nodo da usare per la sostituzione
return <xwLuaNode_t>	Torna il nuovo nodo creato.
see	node:insertNodeAfter(), node:insertNodeBefore(), node:insertFirstNodeChild(), node:appendNodeChild(), node:erase()

numeric node:setAttribute(<String> attrName, <String> value)

param <String> attrName	Il nome dell'attributo, eventualmente da creare, cui assegnare il valore. Si noti che assegnare un valore vuoto non corrisponde a cancellare l'attributo.
param <String> value	Il valore da assegnare all'attributo
Assegna un valore all'attributo indicato. Se l'attributo non esiste esso viene creato.	
return <numeric>	Torna 'non 0' in caso di assegnazione positiva, '0' in ogni altro caso.
see	node:setValue(), node:removeAttribute()

numeric node:setValue(<String> value)

Assegna un valore al nodo indicato. Se il nodo contiene dei nodi figli essi vengono rimossi e viene mantenuto solo il contenuto testuale indicato.	
param <String> value	Il valore da assegnare al nodo
return <numeric>	Torna 'non 0' in caso di assegnazione positiva, '0' in ogni altro caso.
see	node:getValue()

1)
Tipicamente in sede di salvataggio dei record, di caricamento, di indicizzazione e di elaborazione dei titoli
2)
Seppure in uscita si genera solitamente un singolo risultato
3)
Per quanto insolito per gli archivi eXtraWay, in questo caso può essere vantaggioso, oltre che lecito, allegare lo stesso script LUA a più distinti files di configurazione, relativi a *packages* distinti ma che magari hanno alcune funzioni in comune
4)
Di cui magari qualcuna usata in altri *Trigger* dello stesso tipo record o di tipi record diversi.
5)
Seppure è vero l'opposto
6)
Avviene o per impostazione forzata o nel momento stesso in cui il record viene caricato dall'archivio eXtraWay
7)
...e mai assegnato ad un diverso archivio