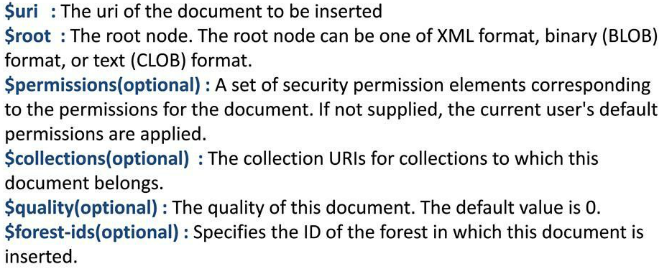


|  |  |
| --- | --- |
| |  | | --- | | **The xdmp:document-load options include:**  **<uri>**  The URI of the document to be loaded. If omitted, then the location is used for the URI.  **<permissions>**  Security permission elements corresponding to the permissions for the document. If not supplied, the current user's default permissions are applied. The default value used for $permissions can be obtained by calling xdmp:default-permissions(). A document that is created by a non-admin user (that is, by any user who does not have the admin role) must have at least one update permission, otherwise the creation will throw an XDMP-MUSTHAVEUPDATE exception.  **<collections>**  The collection URIs for collections to which this document belongs. If not supplied, the document is added to the current user's default collections (the collections returned from xdmp:default-collections()). For each collection that is protected, the user must have permissions to update that collection or have the any-collection privilege. For each unprotected collection, the user must have the unprotected-collections privilege. The <collections> element consists of one or more <collection> child elements.  **For example:**  <collections>  <collection>myCollection1</collection>  <collection>myCollection2</collection>  </collections>  **<quality>**  The quality of this document. A positive value increases the relevance score of the document in text search functions. The converse is true for a negative value. The default value is 0.  **<default-namespace>**  The namespace to use if there is no namespace at the root node of the document. The default value is "".  **<repair>**  A value of full specifies that malformed XML content be repaired. A value of none specifies that malformed XML content is rejected. If no repair option is explicitly specified, the default is implicitly specified by the XQuery version of the caller. In XQuery 1.0 and 1.0-ml the default is none. In XQuery 0.9-ml the default is full. This option has no effect on binary or text documents.  **<format>**  A value of text specifies to get the document as a text document, regardless of the URI specified. A value of binary specifies to get the document as a binary document, regardless of the URI specified. A value of xml specifies to get the document as an XML document, regardless of the URI specified.  **<default-language>**  The language to specify in an xml:lang attribute on the root element node if the root element node does not already have an xml:lang attribute. If default-language is not specified, then nothing is added to the root element node.  **<forests>**  Specifies the ID of the forest in which this document is inserted. Each forest ID is in a <forest> child element and is of type xs:unsignedLong. If the document already exists in the database, it will remain in its existing forest. If no such forest exists or if no such forest is attached to the context database, an error is raised. If multiple forests are specified, the document is inserted into one of the specifed forests. If the document already exists and the forest in which it is stored is set to delete-only, then you must specify the forest IDs to include one or more forests that allow updates, otherwise an exception is thrown.  If you have local disk failover enabled, specify the ID of the master forest. In the event of a failover, MarkLogic server will automatically redirect documents to the replica forest. Specify the ID of the replica forest will result in a "forest not in database" error.  **<encoding>**  Specifies the encoding to use when reading the document into MarkLogic Server. Supported values include UTF-8, ISO-8859-1, as well as many other popular encodings. See the *Search Developer's Guide* for a list of character set encodings by language. All encodings will be translated into UTF-8 from the specified encoding. The string specifed for the encoding option will be matched to an encoding name according to the Unicode Charset Alias Matching rules (<http://www.unicode.org/reports/tr22/#Charset_Alias_Matching>). An automatic encoding detector will be used if the value auto is specified. If no encoding can be detected, the encoding defaults to UTF-8. If no encoding option is specified and you are using a path with an http:// scheme, the encoding defaults to the encoding specified in the http header (if an encoding header is specified), otherwise it defaults to UTF-8. | |
| **Required Privilege:**  ***http://marklogic.com/xdmp/privileges/xdmp-document-load***  If a new document is inserted, you also need the unprotected-uri privilege (only if the URI is not protected), the any-uri privilege, or an appropriate URI privilege.  If adding an unprotected collection to a document, the unprotected-collections privilege (http://marklogic.com/xdmp/privileges/unprotected-collections) is needed; if adding a protected collection, the user must have either permissions to update the collection or the any-collection privilege (http://marklogic.com/xdmp/privileges/any-collection). |
| **Usage Notes:**  When selecting documents over HTTP (where the $location parameter begins with http://), the response from the webserver is loaded into the database, regardless of what the headers returned from the webserver indicate. For example, if the webserver returns a 404 (file not found), then the response page that says "file not found" is loaded into the database. If you want to examine the headers before loading the document, use xdmp:http-get (combined with xdmp:document-insert) instead, as xdmp:http-get allows you to examine the headers returned from the HTTP server. |
| **Example:**  xdmp:document-load("c:\myFile.xml",  <options xmlns="xdmp:document-load">  <uri>/documents/myFile.xml</uri>  <repair>none</repair>  <permissions>{xdmp:default-permissions()}</permissions>  </options>)  **=>** Loads the document with a URI "/documents/myFile.xml"  and does not perform tag repair during the load. |
| **Example:**  xdmp:document-load("http://myCompany.com/file.xml",  <options xmlns="xdmp:document-load"  xmlns:http="xdmp:http">  <uri>/documents/myFile.xml</uri>  <repair>none</repair>  <permissions>{xdmp:default-permissions()}</permissions>  <format>xml</format>  <http:authentication>  <http:username>user</http:username>  <http:password>pass</http:password>  </http:authentication>  </options>)  **=>** Loads the document with a URI "/documents/myFile.xml"  from the server http://myCompany.com, sending the  credentials user/pass. Tag repair is not performed  during the load, and the document is loaded as xml. |
|  |

**Inserts a new document into the database if a document with the specified URI does not already exist. If a document already exists at the specified URI, the function replaces the content of the existing document with the specified content (the $root parameter) as an update operation.** In addition to replacing the content, xdmp:document-insert replaces any permissions, collections, and quality with the ones specified (or with the default values for these parameters, if not explicitly specified). Also, if a properties document exists at the same URI, that properties document (including any content it contains) is preserved.



**$forest-ids**

If the document already exists in the database and if $forest-ids is not specified, it will remain in its existing forest. If no such forest exists or if no such forest is attached to the context database, an error is raised. If multiple forests are specified, the document is inserted into one of the specifed forests. If the document exists and the forest in which it is stored is set to delete-only, then you must set $forest-ids to include one or more forests that allow updates, otherwise an exception is thrown.

If you have local disk failover enabled, specify the ID of the master forest. In the event of a failover, MarkLogic server will automatically redirect documents to the replica forest. Specify the ID of the replica forest will result in a "forest not in database" error.

**Example**

xdmp:document-insert(

"/example.xml", <a>aaa</a>,

(xdmp:permission("editor", "read"),

xdmp:permission("editor", "update")),

"http://examples.com")

**Example**

xdmp:document-insert(

"/example.xml",

<a>aaa</a>,

xdmp:default-permissions(),

xdmp:default-collections(),

10)

**Example**

(:

Specify the forest IDs to move a delete-only document

from one forest to another (assuming at least one of

the forests allows updates).

:)

xdmp:document-insert(

"/example.xml",

<root>new content here</root>, (), (), 0,

xdmp:database-forests(xdmp:database()) )

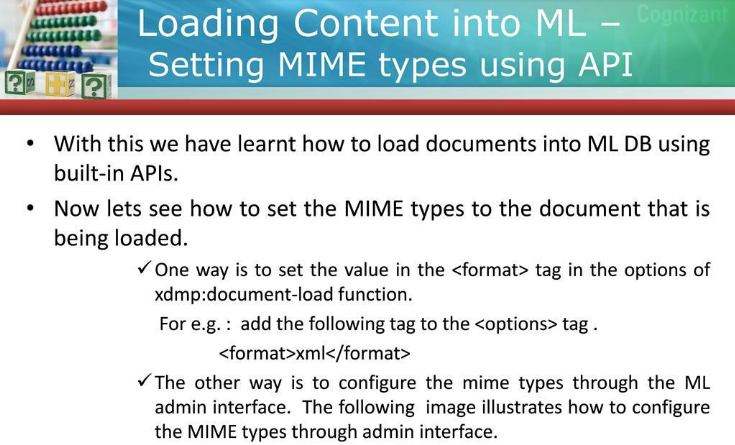
**Example**

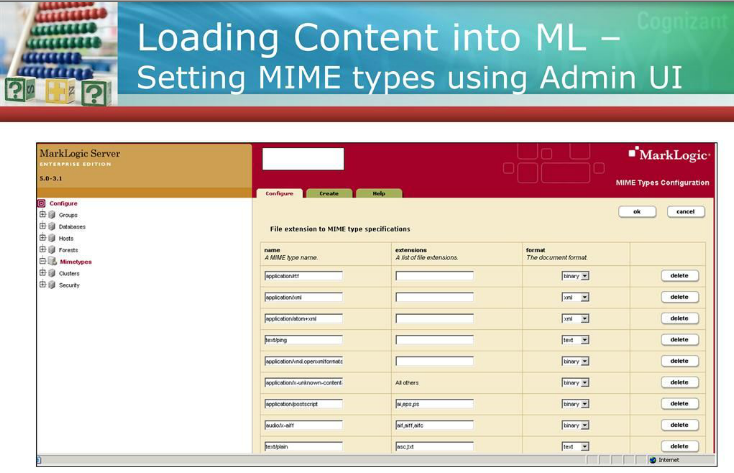
xquery version "1.0-ml";

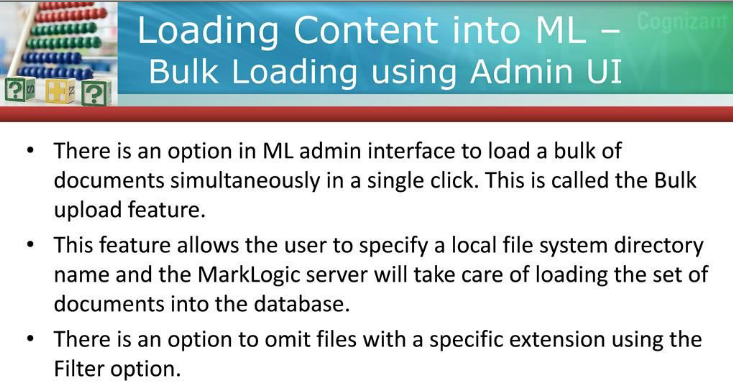
(: create a text document :)

xdmp:document-insert("/text-doc.txt",

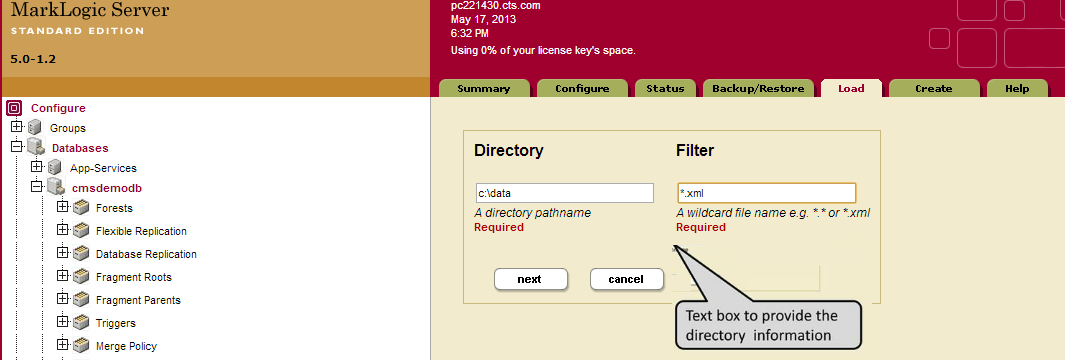
text { "This is a text document." } )

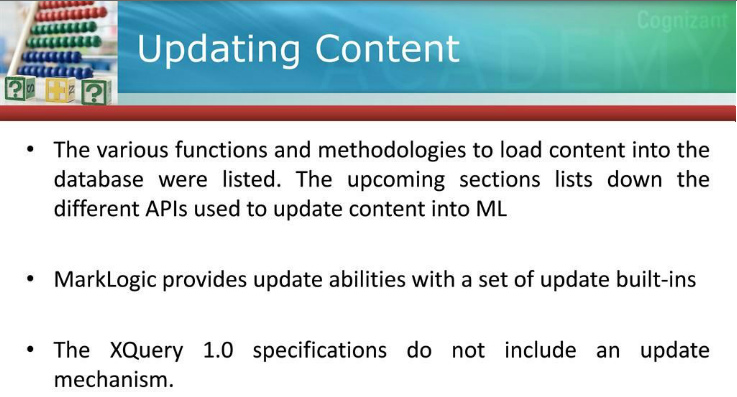


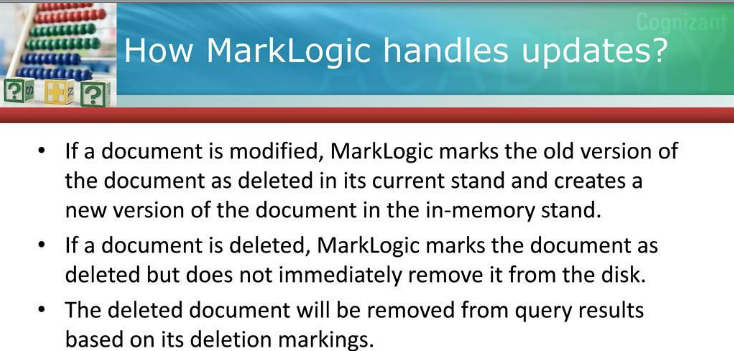


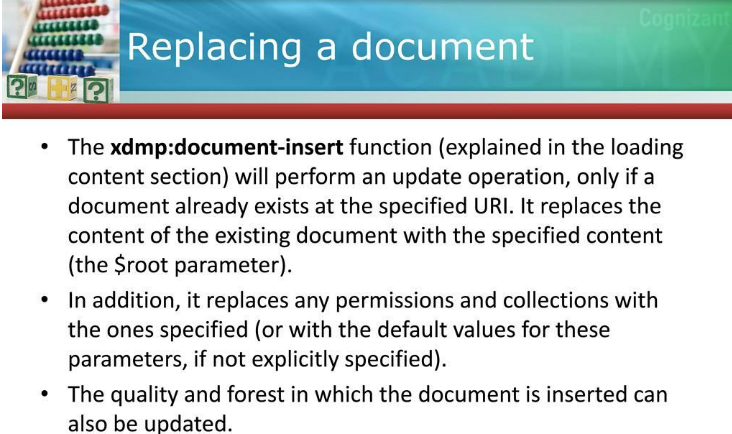


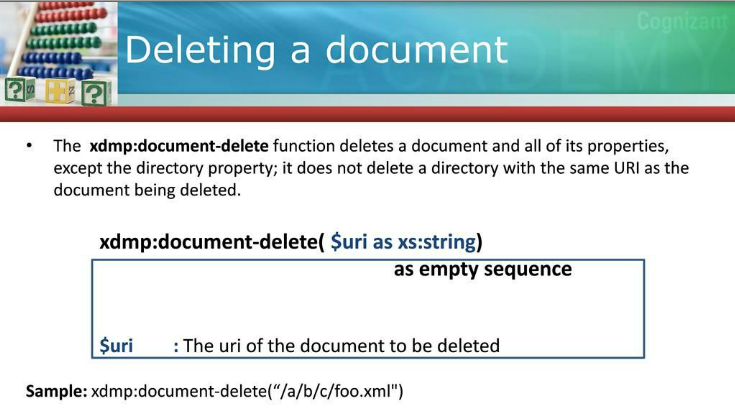
**From here we can upload the documents in bulk, we have specified the type of document as “xml” here.**

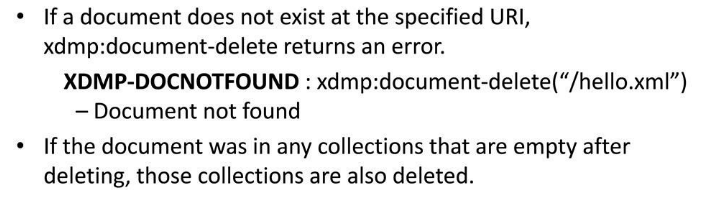




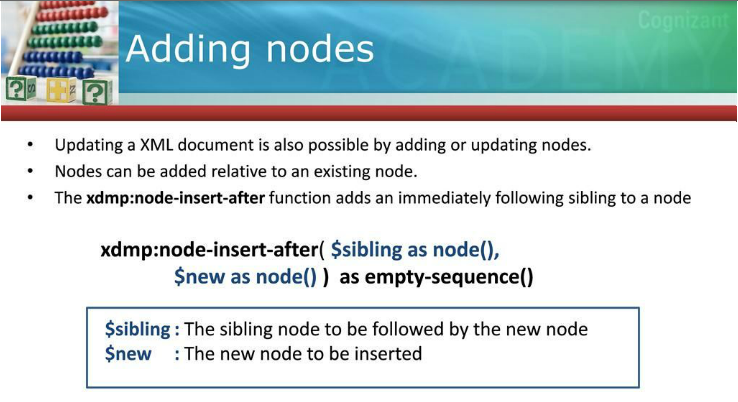




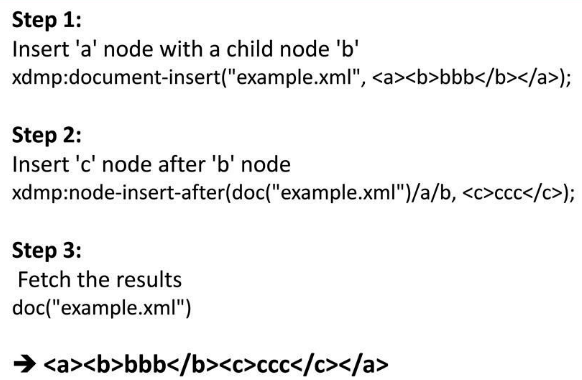




**Node insert after a node [Inserting node must be sibling of existing node]:**



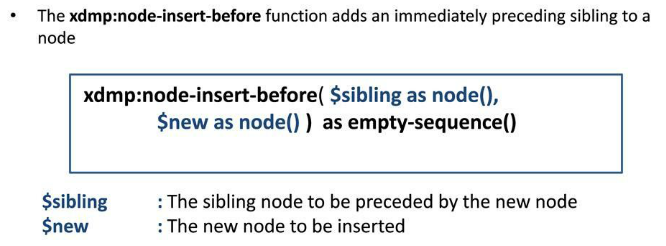
**Example:**



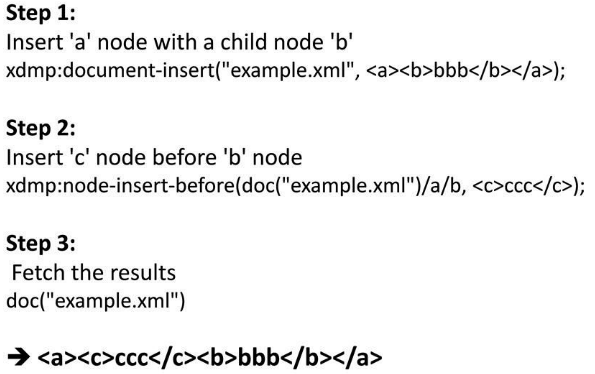
**While Adding Nodes (node-insert-after(..)):**



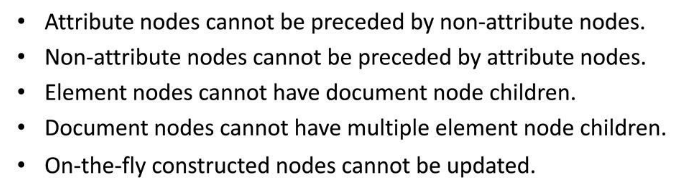
**Node insert before a node [Inserting node must be sibling of existing node]:**



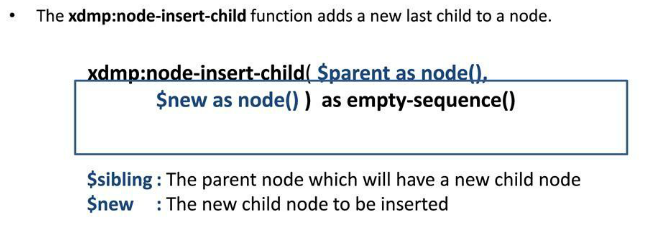
**Example:**



**While adding node (node-insert-before(..)):**

****

**Adding child nodes:**

****

**Inserting attribute node as a child:**

xdmp:node-insert-child(/books/book[@id="0"],<authorname>Holmen</authorname>),

xdmp:node-insert-child(/books/book[@id="0"]/authorname,attribute auid {"101"})

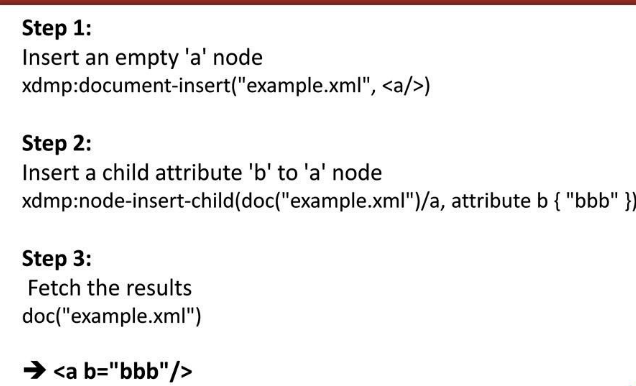
**O/p**>

<books>

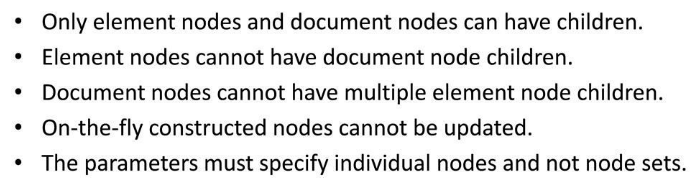
<book id="0">xml<authorname auid="101">Holmen</authorname></book>

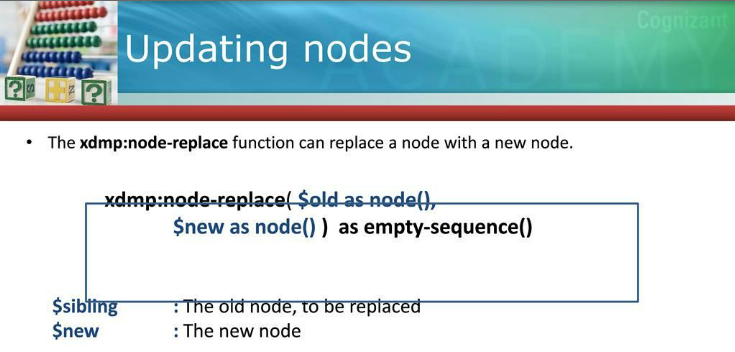
</books>

**Example:**

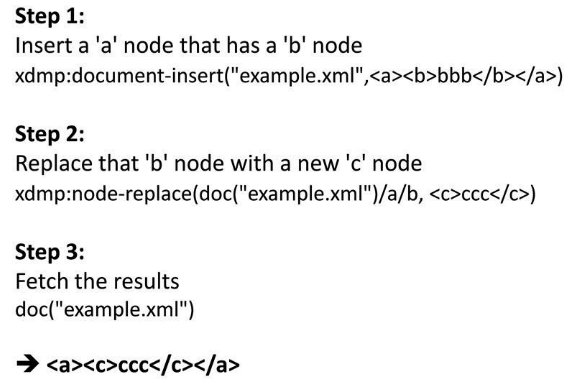
****

**While adding node (node-insert-child(..)):**

****



**Example:**



**Note:** You can replace more than one node at time but same node cannot be updated at same time. If you will try to update the same node more than once at a time then MarkLogic will throw “**XDMP-CONFLICTINGUPDATES exception**”.

**Example:**

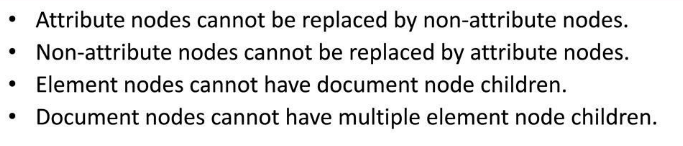
xdmp:node-replace(fn:doc("/books.xml")/books/book[@id="3"],<book id="3">core java</book>),

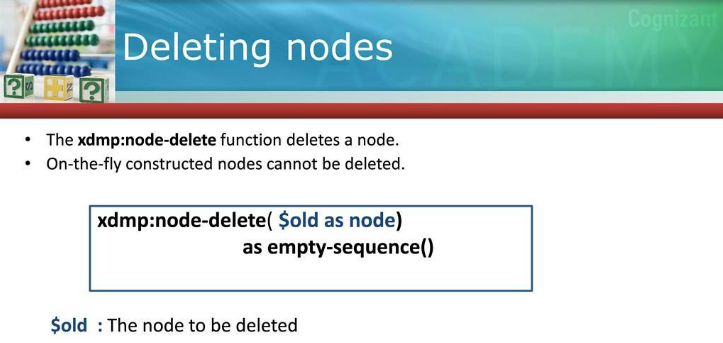
xdmp:node-replace(fn:doc("/books.xml")/books/book[@id="4"],<book id="4">j2ee</book>),

xdmp:node-insert-child(fn:doc("/books.xml")/books,<book id="5">struts 2.x</book>),

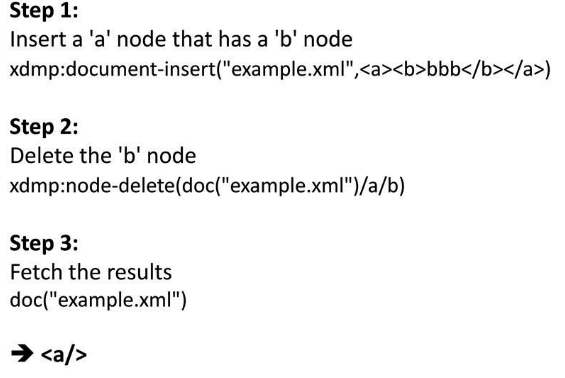
xdmp:node-delete(fn:doc("/books.xml")/books/book[@id="5" and fn:position()!=1])

**While updating node (node-replace(..)):**

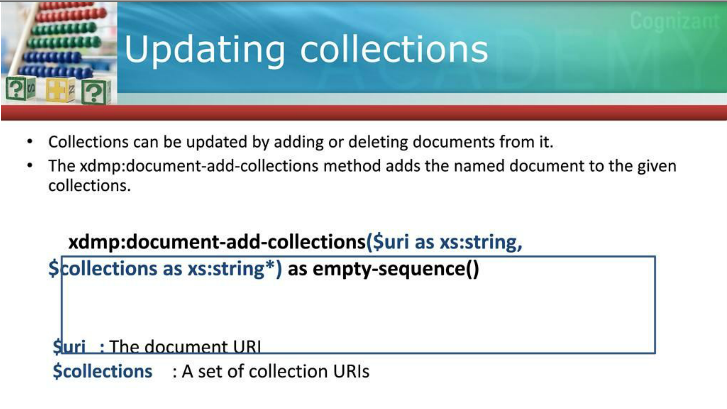




**Example:**

****

**Adding a document to one or more collections:**

****

**Example: xdmp:document-add-collections**

1. xdmp:document-insert("/books.xml",<books>

<book id="0">xml<authorname auid="101">Holmen</authorname><price>100rs</price></book>

<book id="1">xpath</book>

<book id="2">xquery</book>

<book id="3">core java</book>

<book id="4">j2ee</book>

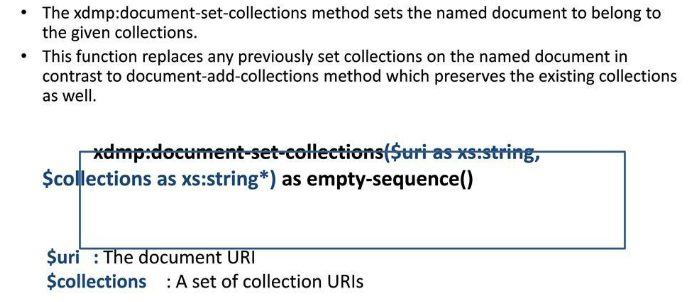
<book id="5">struts 2.x</book>

</books>,(),())

1. xdmp:**document-add-collections**("/books.xml",("/myBooks1/","/myBooks2","/myBooks3/"))

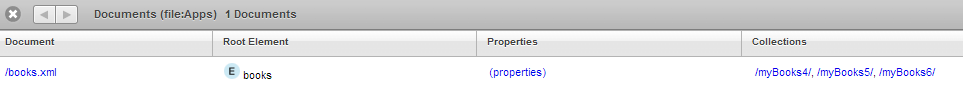
**Note:** Use **xdmp:document-add-collections** to preserve the existing collections. i.e. when you use this method, it will add the named document in the named collection and other collections having the same document will be preserved.

**xdmp:document-set-collections:**

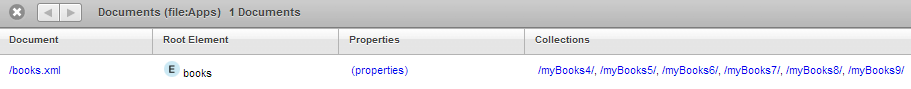
****

**Example:**

1. Before calling **xdmp:document-set-collections(..)**
2. ***xdmp:document-set-collections("/books.xml",("/myBooks4/","/myBooks5/","/myBooks6/"))***

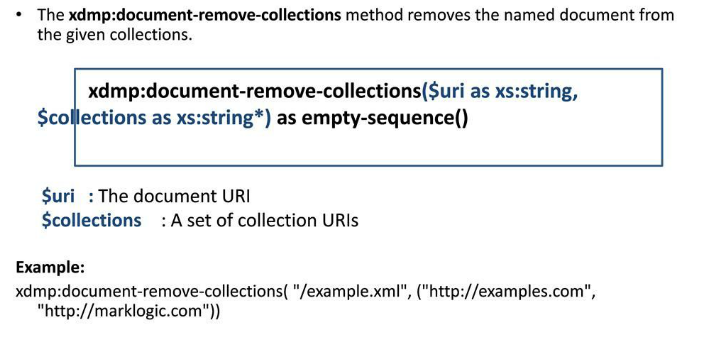
******

1. ***xdmp:document-add-collections("/books.xml",("/myBooks7/","/myBooks8/","/myBooks9/"))***

******

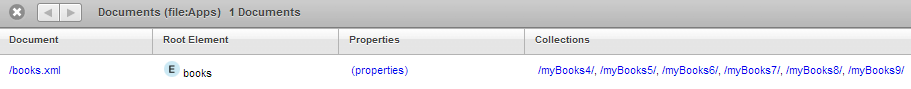
**Note:** Use **xdmp:document-set-collections** if don’t want preserve the existing collections.

**xdmp:document-remove-collections:**

****

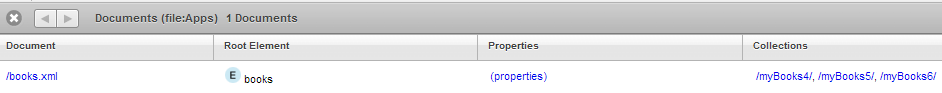
**Example:**

1. Before calling **xdmp:document-remove-collections(..)**

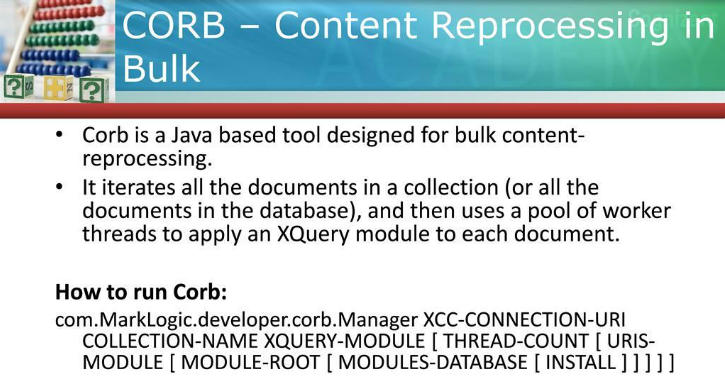
******

1. After calling **xdmp:document-remove-collections(..)**

xdmp:document-remove-collections("/books.xml",("/myBooks7/","/myBooks8/","/myBooks9/"))



**Bulk load:**

****

