Updating MongoDB Documents

You can use the MongoDB adapter to interact with a MongoDB and to insert, get and manipulate documents.

To update documents in a MongoDB database, you can use the **update** and **replace** operations. Updating MongoDB documents is twofold:

1. Identify the document(s) to update.
2. Perform the update.

Finding the Documents to Update

For all actions that refer to existing documents, you need to provide a query string (**queryString**) to identify them. A query string contains all properties of the document you want to use for selection.

Assume we have the following sample **Customer** document structure:

```
{
  "id": "ebd7c78b-44e0-4cbd-8164-d28431716942"

  "name": "John Snow",
  "company": "Winter & Partners",
  "address": {
    "street": "99, Malamute Street",
    "city": "Anchorage, AK 99506",
    "country:": "USA"
  },
  orderValue: "16323.00
}
```

The simplest way to create a query string is the following:

1. Create an object having the structure of the document (**Customer** in the example).

   ```
   create queryData;
   ```

2. Set all query values to this object (the **customerID** in the example).

   ```
   set queryData.id = customerID;
   ```

3. Provide this object as **queryString** by converting it to JSON using classToExtendedJSON().

   ```
   set queryString = queryData.classToExtendedJSON();
   ```

To build a query string, we recommend to **not** use concat() operations but to create a data structure that represents the update string and can be converted to JSON with classToExtendedJSON().

⚠ Building a query string manually (e.g. using concat()) is susceptible to code injection.

Updating Data

You can update complete documents or only parts of them, depending which operation you use.

Operation	Name	Type	Description	Link

Example File (Builder project Add-ons/MongoDB):

<your example path>\Add-ons\MongoDB\uml\simpleMongoDbAccess.xml

ⓘ

update	**updateString**	String	MongoDB update string as valid JSON. ⓘ In contrast to the MongoDB shell, the JSON keys must be quoted properly.	• Building an Update String • Updating a Document Via an Update String
update	**document**	Any \<document class\>	A data structure that contains the new document data. For this, the following rules apply: • unset top-level fields will be skipped • complex fields will be replaced entirely	• Updating a Document Using Parameter "Document"
replace	**document**	Any \<document class\>	A data structure representing the document you want to replace the selected document with. The old stored document will be replaced by the new one, the MongoDB **_id**, however, will not change.	• Replacing a Document

The following examples refer to the same MongoDB document structure as .

Update Using Update String

Building an Update String

Assume you have a customer database containing the following customer document:

```
{
  "id": "0000002b6f7b863000006daa1e5007007a842b9d"
  "name": "John Snow",
  "company": "Winter & Partners",
  "address": {
    "street": "99, Malamute Street",
    "city": "Anchorage, AK 99506",
    "country:": "USA"
  },
  orderValue: "16323.00
}
```

The changes that should be applied are propagated to MongoDB via an update string. Either you yourself can build an update string to be used with an **update** adapter operation, or the xUML Runtime translates a document object you have provided into an update string.

```{ "$set": { "name": "John Snowflake" } }```	A valid update string contains a **$set** key providing as value the changes to be applied.

To build an update string, we recommend to **not** use concat() operations but to create a data structure that represents the update string and can be converted to JSON with classToExtendedJSON().

⚠ Building an update string manually (e.g. using concat()) is susceptible to code injection.

Find below the class structure that represents an update string for the customer address in the example.
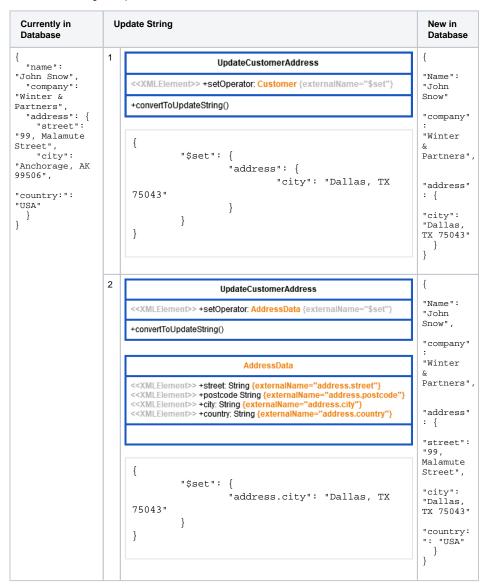
UpdateCustomer
<<XMLElement>> +setOperator: Customer {externalName="$set"}
+convertToUpdateString()

- As a class property cannot have a name `$set`, you need to apply stereotype **XMLElement** and an external name **$set**.
- The structure below the **$set** key reflects the document structure of a customer (**Customer**).

To set values within the update string, create an object of type **UpdateCustomer** and assign values to the properties you want to change.

If you want to update parts of a sub-structure, e.g. the customer address, you need to be careful. Have a look at the following example:

Currently in Database		Update String	New in Database

{
  "name": "John Snow",
  "company": "Winter & Partners",
  "address": {
    "street": "99, Malamute Street",
    "city": "Anchorage, AK 99506",
    "country:": "USA"
  }
}
	1	**UpdateCustomerAddress**
		<<XMLElement>> +setOperator: Customer {externalName="$set"}
		+convertToUpdateString()
		
		{
		        "$set": {
		                "address": {
		                        "city": "Dallas, TX 75043"
		                }
		        }
		}
			{
			  "Name": "John Snow"
			  "company": "Winter & Partners",
			  "address": {
			    "city": "Dallas, TX 75043"
			  }
			}

	2	**UpdateCustomerAddress**
		<<XMLElement>> +setOperator: AddressData {externalName="$set"}
		+convertToUpdateString()
		
		AddressData
		<<XMLElement>> +street: String {externalName="address.street"}
		<<XMLElement>> +postcode String {externalName="address.postcode"}
		<<XMLElement>> +city: String {externalName="address.city"}
		<<XMLElement>> +country: String {externalName="address.country"}
		
		{
		        "$set": {
		                "address.city": "Dallas, TX 75043"
		        }
		}
			{
			  "Name": "John Snow",
			  "company": "Winter & Partners",
			  "address": {
			    "street": "99, Malamute Street",
			    "city": "Dallas, TX 75043"
			    "country:": "USA"
			  }
			}

Using update string 1 you will update the complete address. All properties of address that are not listed in the statement will be removed from the target document.

> ⓘ This is also the way the Runtime will translate any given update document.

Using update string 2 only the **city** property of the target document will be updated. All other properties will be left intact.

Updating a Document Via an Update String

Using an **update** operation with the MongoDB adapter with parameter **updateString**, you can update all or dedicated properties in the selected document.

```
create updateAddress;
set updateAddress.setOperator.address = <new address object>;
set updateString = updateAddress.classToExtendedJSON();
```

In this example, the address part of a customer document is updated. Depending on the structure of the update string, the complete address is replaced, or single address properties are updated.
Refer to Building an Update String for more information on how to build an update string, and an example of the implications.

Update Using Document

Updating a Document Using Parameter "Document"

Using the **update operation** of the MongoDB adapter with parameter **document**, you can update all or dedicated properties in the selected document. Set the values you want to update in the object you provide to the **document** parameter.
The xUML Runtime will translate the **document** parameter to an update string. In this case, the same rules apply as for updates via update string, especially the rule concerning properties of complex structures within a document.

Replacing a Document

Using the **replace** operation of the MongoDB adapter you can replace a complete document in the database. The document provided with the **document** parameter replaces the document indicated by the query string as is. The MongoDB **_id**, however, stays intact.

> The structure of the old document does not necessarily need to match the structure of the new document.