Aggregating Data

Using a MongoDB aggregation pipeline, you can select and aggregate documents. A pipeline is an array of one or multiple stages that will be processed one after the other. Refer to the MongoDB Manual for more information on aggregation and pipelines.

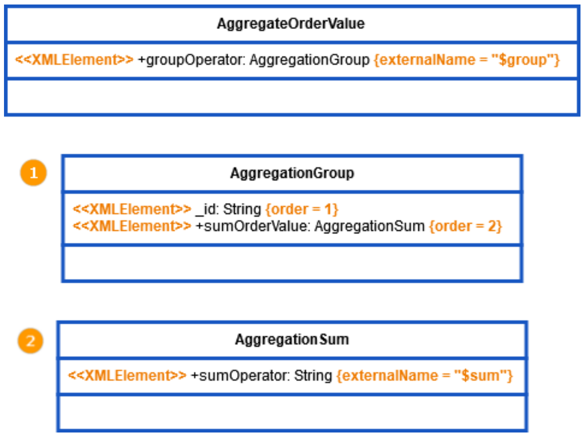
The example below shows a simple aggregation pipeline that consists of one single stage: a `$group` stage that groups documents and summarizes values

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

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

Creating an Aggregation Pipeline

Aggregation stages can be reflected in the Designer using the following class construct:


AggregateOrderValue
<<XMLElement>> +groupOperator: AggregationGroup {externalName = "$group"}
1
AggregationGroup
<<XMLElement>> _id: String {order = 1}
<<XMLElement>> +sumOrderValue: AggregationSum {order = 2}
2
AggregationSum
<<XMLElement>> +sumOperator: String {externalName = "$sum"}

The displayed class diagram defines aggregations stages to aggregate property **orderValue** per **country** for all or a selected country.

Class	Description
1	**Stage $group** Describes a group stage. • As a class property cannot have a name `$group`, you need to apply stereotype **XMLElement** and external name **$group**. • Attribute **_id** is fix and contains the name of the property to group by. In this example the property to group by is fix. It is **$address.country** which you need to set before creating the pipeline (see Building the Aggregation Pipeline below). • The attributes need to be in exact that order to build a correct group stage, therefore they have stereotype **XMLElement** and **order** applied.
2	**Sum Operator** • The structure below the **$group** key defines the `$sum` part of the grouping. • The sum operator (**sumOperator** with external name **$sum**) contains the name of the document property to summarize. In this example the property to summarize is fix. It is **$orderValue** which you need to set before creating the pipeline (see Building the Aggregation Pipeline below).

You can add other stages (e.g. a $match stage) to this structure using the same pattern.

On this Page:

Related Pages:

- Querying MongoDB
- Updating MongoDB Documents
- Aggregating Data
- Inserting and Deleting Documents
- MongoDB Adapter Reference

Related Documentation:

- MongoDB Manual
  - Aggregation

Aggregating Data

Building the Aggregation Pipeline

The action script below shows how to build the pipeline.

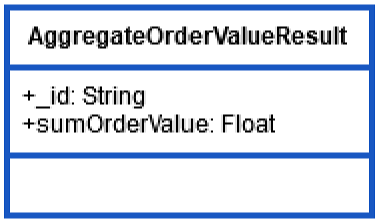
Action Script	Explanation
`buildGroupString(out pipelineStructure: AggregateOderValue, out grouping: AggregationGroup, out sum: AggregationSum, out pipeline: String[])`	Parameters of the action script operation.
`create pipeline;` `create pipelineStructure;` `create grouping;` `create sum;`	• Create the pipeline array. • Create an object of the pipeline structure you have defined before. In this example, this is **piplineStructure : AggregateOrderValue**. • Create the **$group** stage and it's contained **$sum** operator.
`set pipelineStructure.groupOperator = grouping;` `set pipelineStructure.groupOperator._id = "$address.country";`	• Assign the group stage to the pipeline structure. • Assign the name of the document property to group by to **_id**.
`set pipelineStructure.groupOperator.sumOrderValue = sum;` `set pipelineStructure.groupOperator.sumOrderValue.sumOperator = "$orderValue";`	• Assign the sum operator to the group stage. • Assign the name of the document property to summarize to **sumOperator**.
`append pipelineStructure.classToExtendedJSON() to pipeline;`	• Build the group stage using classToExtendedJSON() Operation. • Append the stage to the **pipeline** array.

The resulting aggregation pipeline will look like

```
{ "$group" : { "_id" : "$address.country", "sumOrderValue" : { "$sum" :
"$orderValue" } } }
```

Aggregation Result

`{ "orderVolume": [ { "_id": "USA", "sumOrderVolume": 1098.0 }, { "_id": "CA", "sumOrderVolume": 180.0 } ] }`	As a result of the aggregation, you will get a JSON document that contains the following order value aggregation.

 AggregateOrderValueResult +_id: String +sumOrderValue: Float	If you provide an array of a result structure as an output for the adapter call, the xUML Runtime will map the results accordingly.