

# Strategy: DOCUMENT\_BY\_UNID

Use this strategy to get document from the database by UNID.

Allowed methods

- GET
- POST
- DELETE

Parameters:

- `databaseName(String databaseName)`: name of the database to read data from, use "server!!path/database.nsf" format; when omitted, current database is used
- `keyVariableName(String keyVariableName)`: name of the variable name in route to read value from; mandatory
- `formName(String formName)`: name of the Form used when creating new document via @new

## GET method

```
router.GET('topics/{id}') {
  strategy(DOCUMENT_BY_UNID) {
    keyVariableName('id')
  }
  mapJson 'id', json:'id', type:'STRING', isformula:true, formula:'@DocumentUniqueID'
  mapJson 'date_created', json:'date_created', type:'DATETIME', isformula:true, formula:'@Created'
  mapJson 'topic', json:'topic', type:'STRING'
  mapJson 'author', json:'author', type:'STRING', isformula:true, formula:'@Name([CN]; @Author)'
  mapJson 'categories', json:'categories', type:'ARRAY_OF_STRING'
}
```

With route defined above, URL `http://server.name/path-to/db.nsf/xsp/.xrest/topics/54502859C07299C7C12580D8006404F4` will return this JSON:

```
{
  "date_created": "2017-03-03T18:28+00:00",
  "author": "Martin Jinoch",
  "categories": [
    "category1",
    "category2"
  ],
  "topic": "test"
}
```

## POST method

Creating new document

With route defined below, you can create new document by POSTING to URL `http://server.name/path-to/db.nsf/xsp/.xrest/topics/@new`

```

router.POST('topics/{id}') {
  strategy(DOCUMENT_BY_UNID) {
    keyVariableName("id")
    formName("MainTopic")
  }
  mapJson "id", json:'id', type:'STRING', isformula:true, formula:'@DocumentUniqueId', readonly:true
  mapJson "Subject", json:'topic', type:'STRING'
  mapJson "body", json:'content', type:'MIME'
  mapJson 'categories', json:'categories', type:'ARRAY_OF_STRING', writeonly:true
  mapJson "date", json:'date', type:'DATETIME', isformula:true, formula:'@Created', readonly:true
  mapJson "author", json:'author', type:'STRING', isformula:true, formula:'@Name([CN]; From)', readonly:true

  events PRE_SAVE_DOCUMENT: {
    context, document ->
    nsfHelp = context.getNSFHelper()
    nsfHelp.computeWithForm(document)
  }
}

```

Let's have a detailed look at the definition above: note `readonly:true` part on the line 5. With this you can define the "output only" flag for fields. So SmartNSF doesn't expect and read field "id" from POSTed JSON data, but it includes it in the returned JSON.

Analogically you can define "input only" fields by specifying `writeonly:true`. Those are then not included in the output.

In the example above we are using another interesting feature: `events`. `PRE_SAVE_DOCUMENT` event, as name suggests, is executed before the document is saved. In this case we are simply grabbing the context and calling `computeWithForm()` method (provided by `NSFHelper` class) on the document.

So POSTING following JSON to URL `http://server.name/path-to/db.nsf/xsp/.xrest/topics/@new`

```

{
  "topic": "created from rest",
  "content": "<h3>test</h3>",
  "categories": [
    "cat1",
    "cat2",
    "cat3"
  ]
}

```

should return something like this

```

{
  "content": "<h3>test</h3>",
  "author": "martin jinoch",
  "id": "EF09FEA7F9FF7265C12580E40065E4D6",
  "date": "2017-03-15T18:57+00:00",
  "topic": "created from rest"
}

```

#### Updating existing document

With the same route (see above) you can update data in existing document, when you specify its UNID in the URL, so for example POSTING

```

{
  "topic": "updated from rest",
  "content": "<h3>updated test</h3>",
  "categories": [
    "cat1",
    "cat2",
    "cat3"
  ]
}

```

to URL `http://server.name/path-to/db.nsf/xsp/.xrest/topics/EF09FEA7F9FF7265C12580E40065E4D6` will update document with UNID `EF09FEA7F9FF7265C12580E40065E4D6` and return this JSON:

```
{
  "content": "<h3>updated test</h3>",
  "author": "martin jinoch",
  "id": "EF09FEA7F9FF7265C12580E40065E4D6",
  "date": "2017-03-15T18:57+00:00",
  "topic": "updated from rest"
}
```

## DELETE method

You can delete document using route such as this

```
router.DELETE('topics/{id}') {
  strategy(DOCUMENT_BY_UNID) {
    keyVariableName('id')
  }
}
```

Sending DELETE to URL `http://server.name/path-to/db.nsf/xsp/.xrest/topics/EF09FEA7F9FF7265C12580E40065E4D6` will delete document with specified UNID.

## Related articles

- [Strategy: CUSTOM](#)
- [Strategy: DOCUMENT\\_BY\\_UNID](#)
- [Strategy: DOCUMENT\\_FROM\\_VIEW\\_BY\\_KEY](#)
- [Strategy: DOCUMENTS\\_BY\\_FORMULA](#)
- [Strategy: DOCUMENTS\\_BY\\_FORMULA\\_PAGED](#)
- [Strategy: DOCUMENTS\\_BY\\_SEARCH\\_FT](#)
- [Strategy: DOCUMENTS\\_BY\\_SEARCH\\_FT\\_PAGED](#)
- [Strategy: DOCUMENTS\\_BY\\_VIEW](#)
- [Strategy: DOCUMENTS\\_BY\\_VIEW\\_PAGED](#)
- [Strategy: DOCUMENTS\\_FROM\\_VIEW\\_BY\\_KEY](#)
- [Strategy: DOCUMENTS\\_FROM\\_VIEW\\_BY\\_KEY\\_PAGED](#)
- [Strategy: VIEWENTRIES](#)
- [Strategy: VIEWENTRIES\\_BY\\_CATEGORY](#)
- [Strategy: VIEWENTRIES\\_BY\\_CATEGORY\\_PAGED](#)
- [Strategy: VIEWENTRIES\\_PAGED](#)