SD540 Server-Side Programming

Maharishi University of Management

Masters of Software Development

Associate Professor Asaad Saad

Maharishi International University - Fairfield, Iowa

Searching Within Rich Document (Array)

```
// { _id: 1, courses: [ "SD500", "SD540", "SD550" ] }
// find all documents where courses value contains "SD540"
await Model.find({ courses: "SD540" })

// { _id: 1, courses: [ {code: "SD500"}, {code: "SD540"}, {code: "SD550"} ] }
// find all documents where courses value contains "SD540"
await Model.find({ "courses.code": "SD540" })

// find all documents where courses value contains "SD540" or "SD550"
await Model.find({ courses: { $in: ["SD540", "SD550"] } })

// find all documents where courses value contains "SD540" and "SD550"
await Model.find({ courses: { $all: [ "SD540" , "SD550" ] } })
```

Array Search with `$elemMatch`

The **`$elemMatch`** operator matches documents that contain an array field with at least one element that matches all the specified query criteria.

```
{ _id: 1, cars: [ 33, 2, 5, 10 ] }
{ _id: 2, cars: [ 15, 8, 9, 10 ] }
{ _id: 3, cars: [ 10, 11, 12 ] }

await Model.find( { cars: { $elemMatch: { $gt: 5, $lt: 10 } } } )
// checks if there is one value matches the condition
// returns { _id: 2, cars: [ 15, 8, 9, 10 ] }
```

$ INDEX (Positional Operator)

Every time you search a first-level array, MongoDB provides the positional **$** operator as a reference to the **INDEX** of the element that matched.

When you use the positional **$** operator in projection, it limits the contents of an array to return the first element that matches the query condition on the array.

```
{ _id: 4, grades: [ {total: 80, student: 8}, {total: 85, student: 5}, {total: 85, student: 8} ]}

await Model.findOne( { _id: 4, "grades.total": 85 }, { _id: 0, "grades.$" : 1 } )

// { grades: [ {total: 85, student: 5} ]}
```

Array Pagination

Use the projection **`$slice`** operator to specify the skip and limit of how many elements are to be returned from the array.

```
{ _id: 4, grades: [ {total: 80, student: 8}, {total: 85, student: 5}, {total: 86, student: 8} ]}

await Model.findOne( { _id: 4 }, { "grades": { "$slice": 2 } } )

// First 2 elements { _id: 4, grades: [{total: 80, student: 8}, {total: 85, student: 5} ]}

await Model.findOne( { _id: 4 }, { "grades": { "$slice": -2 } } )

// Last 2 elements { _id: 4, grades: [ {total: 85, student: 5} , {total: 86, student: 8} ]}

await Model.findOne( { _id: 4 }, { "grades": { "$slice": [ 1, 1 ] } } )

// Skip 1, Limit 1 { _id: 4, grades: [ {total: 85, student: 5} ]}
```

Notes

- **`find`** scans all documents, return documents that match the query.
- **`findOne`** scans all documents, return the first document that matches the query.
- A **`find/findOne`** query without projection, return all properties of the document.
- A **`find/findOne`** query with projection, return the properties set in the projection condition only.
- You cannot use **`$`** in the query part, it can only be used as projection operator or when targeting an element to update.

Insert One Array Element

The **$push** operator appends a specified value to an array.

```
// { "_id": 1, "scores": [1, 2, 5, 4] }
 await Model.updateOne({_id: 1}, { $push: { "scores": 5 } })
// { "_id": 1, "scores": [1, 2, 5, 4, 5] }
```

The **$addToSet** operator adds a value to an array only if the value is not already present.

```
// { "_id": 1, "scores": [2, 4, 5] }
await Model.updateOne({_id: 1}, { $addToSet: { "scores": 4 } })
// { "_id": 1, "scores": [2, 4, 5] }
```

Insert Multiple Array Elements

Use **$push** or **$addToSet** with the **$each** modifier to append multiple values to the array field.

```
// { "_id": 1, "scores": [1, 2, 3, 4] }
 await Model.updateOne({_id: 1}, { $push: { "scores": { $each: [5, 6, 7] } } })
// { "_id": 1, "scores": [1, 2, 3, 4, 5, 6, 7] }
```

Delete Array Element(s)

The **`$pull`** operator removes from an existing array all instances of a value or values that match a specified condition.

```
// { "_id": 1, "scores": [12, 57, 14, 61, 48, 57] }
 await Model.updateOne({_id: 1}, { $pull: { "scores": 57 } })
// { "_id": 1, "scores": [12, 14, 61, 48] }
```

Update one Array Element

To update one element in Array we target the element with the **INDEX**

```
// { "_id": 1, "scores": [1, 2, 3, 4] }
await Model.updateOne({_id: 1}, { $set: { "scores.2": 5 } })
// { "_id": 1, "scores": [1, 2, 5, 4] }
```

```
// { _id: 4, grades: [{total: 80, student: 8}, {total: 85, student: 5}, {total: 85, student: 8}] }
await Model.updateOne({ _id: 4, "grades.total": 85 }, { $set: { "grades.$.student" : 6 } })
// { _id: 4, grades: [ {total: 80, student: 8}, {total: 85, student: 6}, {total: 85, student: 8} ]}
```

Update Multiple Array Elements

To perform an update on all embedded array elements of each document that matches your query, use the filtered positional operator **$[<identifier>]**. The filtered positional operator **$[<identifier>]** specifies the matching array elements in the update document. To identify which array elements to match, pair this operator with **<identifier>** in an **arrayFilters** object.

```
{ _id: 4, grades: [ {total: 80, student: 8}, {total: 85, student: 5}, {total: 86, student: 8} ]}
```

```
await Model.updateOne(
   { _id: 4 },
   { $set: { "grades.$[obj].student" : 0 } } ,
   { arrayFilters: [{ "obj.total": { $gte: 85 } }] }
)
```

```
// Update all elements in the array that match our condition
{ _id: 4, grades: [ {total: 80, student: 8}, {total: 85, student: 0}, {total: 86, student: 0} ]}
```