MongoDB Projection

Summary: in this tutorial, you'll how to use the MongoDB projection that allows you to select fields to return from a query

Introduction to the MongoDB projection

In MongoDB, projection simply means selecting fields to return from a query.

By default, the `find()` (https://mongodbtutorial.org/mongodb-crud/mongodb-find/) and `findOne()` (https://mongodbtutorial.org/mongodb-crud/mongodb-findone/) methods return all fields in matching documents. Most of the time you don't need data from all the fields.

To select fields to return from a query, you can specify them in a document and pass the document to the `find()` and `findOne()` methods. This document is called a projection document.

To determine if a field is included in the returned documents, you use the following syntax:

```
{ <field>: value, ...}
```

If the `value` is `1` or `true`, the `<field>` is included in the matching documents. In case the `value` is `0` or `false`, it is suppressed from the returned documents.

If the `projection` document is empty `{}`, all the available fields will be included in the returned documents.

To specify a field in an embedded document, you use the following dot notation:

```
{ "<embeddedDocument>.<field>": value, ... }
```

Similarly, to include a `<field>` from an embedded document located in an array, you use the following dot notation syntax:

```
{ "<arrayField>.field": value, ...}
```

MongoDB projection examples

We'll use the following `products` collection for the projection examples.

```
db.products.insertMany([
    { "_id" : 1, "name" : "xPhone", "price" : 799, "releaseDate": ISODate("2011-0
    { "_id" : 2, "name" : "xTablet", "price" : 899, "releaseDate": ISODate("2011-
    { "_id" : 3, "name" : "SmartTablet", "price" : 899, "releaseDate": ISODate("2
    { "_id" : 4, "name" : "SmartPad", "price" : 699, "releaseDate": ISODate("2020
    { "_id" : 5, "name" : "SmartPhone", "price" : 599,"releaseDate": ISODate("202
 ])
```

1) Returning all fields in matching documents

If you don't specify the projection document, the `find()` method will include all fields in the matching documents.

For example, the following query returns all fields from all documents in the `products` collection where the `price` is `899` :

```
db.products.find({price: 899});
```

Output:

```
[
  {
    _id: 2,
```

```
    name: 'xTablet',
    price: 899,
    releaseDate: ISODate("2011-09-01T00:00:00.000Z"),
    spec: { ram: 16, screen: 9.5, cpu: 3.66 },
    color: [ 'white', 'black', 'purple' ],
    storage: [ 128, 256, 512 ],
    inventory: [ { qty: 300, warehouse: 'San Francisco' } ]
  },
  {
    _id: 3,
    name: 'SmartTablet',
    price: 899,
    releaseDate: ISODate("2015-01-14T00:00:00.000Z"),
    spec: { ram: 12, screen: 9.7, cpu: 3.66 },
    color: [ 'blue' ],
    storage: [ 16, 64, 128 ],
    inventory: [
      { qty: 400, warehouse: 'San Jose' },
      { qty: 200, warehouse: 'San Francisco' }
    ]
  }
]
```

2) Returning specified fields including the _id field

If you specify the fields in the projection document, the `find()` method will return only these fields including the `_id` field by default.

The following example returns all documents from the `products` collection. However, its result includes only the `name`, `price`, and `_id` field in the matching documents:

```
db.products.find({}, {
    name: 1,
```

```
    price: 1
});
```

Output:

```
[
  { _id: 1, name: 'xPhone', price: 799 },
  { _id: 2, name: 'xTablet', price: 899 },
  { _id: 3, name: 'SmartTablet', price: 899 },
  { _id: 4, name: 'SmartPad', price: 699 },
  { _id: 5, name: 'SmartPhone', price: 599 }
]
```

To suppress the `_id` field, you need to explicitly specify it in the projection document like this:

```
db.products.find({}, {
    name: 1,
    price: 1
,
    _id: 0
});
```

Output:

```
[
  { name: 'xPhone', price: 799 },
  { name: 'xTablet', price: 899 },
  { name: 'SmartTablet', price: 899 },
  { name: 'SmartPad', price: 699 },
  { name: 'SmartPhone', price: 599 }
]
```

3) Returning all fields except for some fields

If the number of fields to return is many, you can use the projection document to exclude other fields instead.

The following example returns all fields of the document `_id 1` except for `releaseDate` , `spec` , and `storage` fields:

```
db.products.find({_id:1}, {
    releaseDate: 0,
    spec: 0,
    storage: 0
})
```

Output:

```
[
  {
    _id: 1,
    name: 'xPhone',
    price: 799,
    color: [ 'white', 'black' ],
    inventory: [ { qty: 1200, warehouse: 'San Jose' } ]
  }
]
```

4) Returning fields in embedded documents

The following example returns the `name` , `price` , and `_id` fields of document `_id 1` . It also returns the `screen` field on the `spec` embedded document:

```
db.products.find({_id:1}, {
    name: 1,
    price: 1,
```

```
    "spec.screen": 1
})
```

Output:

```
[ { _id: 1, name: 'xPhone', price: 799, spec: { screen: 6.5 } } ]
```

MongoDB 4.4 and later allows you to specify embedded fields using the nested form like this:

```
db.products.find({_id:1}, {
    name: 1,
    price: 1,
    spec : { screen: 1 }
})
```

5) Projecting fields on embedded documents in an array

The following example specifies a projection that returns:

- the `_id` field (by default)
- The `name` field
- And `qty` field in the documents embedded in the `inventory` array.

```
db.products.find({}, {
    name: 1,
    "inventory.qty": 1
});
```

Output:

```
[
  { _id: 1, name: 'xPhone', inventory: [ { qty: 1200 } ] },
  { _id: 2, name: 'xTablet', inventory: [ { qty: 300 } ] },
```

```
  {
    _id: 3, name: 'SmartTablet', inventory: [ { qty: 400 }, { qty: 200 } ]
  },
  { _id: 4, name: 'SmartPad', inventory: [ { qty: 1200 } ] },
  { _id: 5, name: 'SmartPhone' }
]
```

Summary

- Use the `{<field>: 1}` to include the `<field>` in the matching documents and `{<field>: 0}` to exclude it.
- Use the `{ <embeddedDocument>.<field>: 1}` to include the `<field>` from the `<embeddedDocument>` in the matching document and `{ <embeddedDocument>.<field>: 0}` to suppress it.
- Use `{ <arrayField>.<field>: 1}` to include the `<field>` from the embedded document in an array in the matching document and `{ <arrayField>.<field>: 0}` to exclude it.