mongoose-paginate-v2

1.0.13 • Public • Published 7 hours ago

Readme

1 Dependencies

3 Dependents

5 Versions

install

```
npm i mongoose-paginate-v2
```

weekly downloads

420

version	license
1.0.13	MIT
open issues	**pull requests**
0	0
homepage	**repository**
github.com	github
last publish	
7 hours ago	

collaborators

Test with RunKit | Report a vulnerability

mongoose-paginate-v2

npm v1.0.13 | dependencies up to date | devDependencies up to date | contributions welcome | hits 3241

> A cursor based custom pagination library for Mongoose with customizable labels.

Installation

```
npm install mongoose-paginate-v2
```

Usage

Add plugin to a schema and then use model `paginate` method:

```
var mongoose          = require('mongoose');
var mongoosePaginate = require('mongoose-paginate-v2');

var mySchema = new mongoose.Schema({
    /* your schema definition */
});

mySchema.plugin(mongoosePaginate);

var myModel = mongoose.model('SampleModel',  mySchema);

myModel.paginate().then({}) // Usage
```

Model.paginate([query], [options], [callback])

Returns promise

Parameters

- `[query]` {Object} - Query criteria. Documentation
- `[options]` {Object}
  - `[select]` {Object | String} - Fields to return (by default returns all fields). Documentation
  - `[sort]` {Object | String} - Sort order. Documentation
  - `[populate]` {Array | Object | String} - Paths which should be populated with other documents. Documentation
  - `[lean=false]` {Boolean} - Should return plain javascript objects instead of Mongoose documents? Documentation
  - `[leanWithId=true]` {Boolean} - If `lean` and `leanWithId` are `true`, adds `id` field with string representation of `_id` to every document
  - `[offset=0]` {Number} - Use `offset` or `page` to set skip position
  - `[page=1]` {Number}
  - `[limit=10]` {Number}
  - `[customLabels]` {Object} - Developers can provide custom labels for manipulating the response data.
- `[callback(err, result)]` - If specified the callback is called once pagination results are retrieved or when an error has occurred

Return value

Promise fulfilled with object having properties:

- `docs` {Array} - Array of documents
- `totalDocs` {Number} - Total number of documents in collection that match a query
- `limit` {Number} - Limit that was used
- `hasPrevPage` {Bool} - Availability of prev page.
- `hasNextPage` {Bool} - Availability of next page.
- `page` {Number} - Current page number
- `totalPages` {Number} - Total number of pages.
- `offset` {Number} - Only if specified or default `page` / `offset` values were used
- `prevPage` {Number} - Previous page number if available or NULL
- `nextPage` {Number} - Next page number if available or NULL

Please note that the above properties can be renamed by setting customLabel attribute.

Sample Usage

Return first 10 documents from 100

```
const options = {
    page: 1,
    limit: 10
};

Model.paginate({}, options, function(err, result) {
    // result.docs
    // result.totalDocs = 100
    // result.limit = 10
    // result.page = 1
    // result.totalPages = 10
    // result.hasNextPage = true
    // result.nextPage = 2
    // result.hasPrevPage = false
    // result.prevPage = null

});
```

With custom return labels

Now developers can specify the return field names if they want. Below are the list of attributes whose name can be changed.

- totalDocs
- docs
- limit
- page
- nextPage
- prevPage
- totalPages

You should pass the names of the properties you wish to changes using `customLabels` object in options.

Same query with custom labels

```
const myCustomLabels = {
    totalDocs: 'itemCount',
    docs: 'itemsList',
    limit: 'perPage',
    page: 'currentPage',
    nextPage: 'next',
    prevPage: 'prev',
    totalPages: 'pageCount'
};

const options = {
    page: 1,
    limit: 10,
    customLabels: myCustomLabels
};

Model.paginate({}, options, function(err, result) {
    // result.itemsList [here docs become itemsList]
    // result.itemCount = 100 [here totalDocs becomes itemCoun
    // result.perPage = 10 [here limit becomes perPage]
    // result.currentPage = 1 [here page becomes currentPage]
    // result.pageCount = 10 [here totalPages becomes pageCoun
    // result.next = 2 [here nextPage becomes next]
    // result.prev = null [here prevPage becomes prev]

    // result.hasNextPage = true [not changeable]
    // result.hasPrevPage = false [not changeable]
});
```

Other Examples

Using `offset` and `limit`:

```
Model.paginate({}, { offset: 30, limit: 10 }, function(err, re
```

```
    // result.docs
    // result.totalPages
    // result.limit - 10
    // result.offset - 30
});
```

With promise:

```
Model.paginate({}, { offset: 30, limit: 10 }).then(function(re
    // ...
});
```

More advanced example

```
var query   = {};
var options = {
    select:   'title date author',
    sort:     { date: -1 },
    populate: 'author',
    lean:     true,
    offset:   20,
    limit:    10
};

Book.paginate(query, options).then(function(result) {
    // ...
});
```

Zero limit

You can use `limit=0` to get only metadata:

```
Model.paginate({}, { offset: 100, limit: 0 }).then(function(re
    // result.docs - empty array
    // result.totalDocs
    // result.limit - 0
```

```
    // result.offset - 100
});
```

Set custom default options for all queries

config.js:

```
var mongoosePaginate = require('mongoose-paginate-v2');

mongoosePaginate.paginate.options = {
    lean:  true,
    limit: 20
};
```

controller.js:

```
Model.paginate().then(function(result) {
    // result.docs - array of plain javascript objects
    // result.limit - 20
});
```

Thanks

This is a advanced version of mongoose-paginate forked from Mongoose Paginate.
Thanks to the initial author Edward Hotchkiss

License

MIT

Keywords

mongoose pagination plugin mongodb paginate paging next prev nextpage prevpage total paginator plugin

You Need Help

Documentation
Support / Contact Us
Registry Status
Report Issues
npm Community Site
Security

About npm

About npm, Inc
Jobs
npm Weekly
Blog
Twitter
GitHub

Terms & Policies

Terms of Use
Code of Conduct
Package Name Disputes
Privacy Policy
Reporting Abuse
Other policies

npm loves you