Bhargav Lad

CS Graduate Student.

Mongoose CheatSheet

Posted 1 year ago by **Bhargav Lad**

Mongoose Operations Cheat sheet

Schema

Types

- `String`
- `Boolean`
- `Number`
- `Date`
- Array
- `Buffer`
- `Schema.Types.Mixed`
- `Schema.Types.ObjectId`

```
const UserSchema = new mongoose.Schema({
  fullname: {
    type: String, // Data Type
    min:6,
    max:12,
    required: [true, "Please enter your fullname"], // Required with error
    trim: true,
  },
  followersCount: {
    type: Number,
    default: 0,
  },
  followers: [{ type: mongoose.Schema.ObjectId, ref: "User" }], // Array of Object Ids and
ref to schema
  createdAt: {
    type: Date,
    default: Date.now,
  },
  drink: {
    type: String,
    enum: ['Coffee', 'Tea', 'Water',]
  }
}

 module.exports = mongoose.model("User", UserSchema);
```

Queries are not promises

Mongoose queries are **not** promises. They have a `.then()` function for co and async/await as a convenience. If you need a fully-fledged promise, use the `.exec()` function.

```
const query = Band.findOne({name: "Guns N' Roses"});
assert.ok(!(query instanceof Promise));

// A query is not a fully-fledged promise, but it does have a `.then()`.
query.then(function (doc) {
  // use doc
});

// `.exec()` gives you a fully-fledged promise
const promise = query.exec(); //or use await on promise
assert.ok(promise instanceof Promise);

promise.then(function (doc) {
  // use doc
});
```

Methods

Mongoose Model Methods

- `find(criteria, [fields], [options], [callback])` : find document; callback has error and documents arguments
- `count(criteria, [callback]))` : return a count; callback has error and count arguments
- `findById(id, [fields], [options], [callback])` : return a single document by ID; callback has error and document arguments
- `findByIdAndUpdate(id, [update], [options], [callback])` : executes MongoDB's findAndModify to update by ID
- `findByIdAndRemove(id, [options], [callback])` : executes MongoDB's findAndModify to remove
- `findOne(criteria, [fields], [options], [callback])` : return a single document; callback has error and document arguments
- `findOneAndUpdate([criteria], [update], [options], [callback])` : executes MongoDB's findAndModify to update
- `findOneAndRemove(id, [update], [options], [callback])` : executes MongoDB's findAndModify to remove
- `update(criteria, update, [options], [callback])` : update documents; callback has error, and count arguments
- `create(doc(s), [callback])` : create document object and save it to database; callback has error and doc(s) arguments
- `remove(criteria, [callback])` : remove documents; callback has error argument

Examples

```
// --------------------------------- Find ----------------------------------------
User.find({ author : bob._id }).exec()
// ---------------------------- Find in list -------------------------------------------
User.find()
    .where("_id")
    .in(following.concat([req.user.id]))
    .exec();
// ------------------------------Find and Populate-----------------------------------------
--
Post.find()
    .populate({
      path: "comments",
      select: "text",
      populate: { path: "user", select: "avatar fullname username" },  // Two level populate
    })
    .populate({ path: "user", select: "avatar fullname username" })
    .sort("-createdAt")
    .where("_id")
```

Bhargav Lad

CS Graduate Student.

```
    .exec();

// ------------------------------  Find Regex --------------------------------------------
  const regex = new RegExp(req.query.username, "i");
  const users = await User.find({ username: regex });
// -------------------------------- Find One -------------------------------------------
User.findOne({ title: 'Bob goes sledding' })
        .select("-password")
  .populate('author')
  .exec()
// -------------------------------Find by Id ------------------------------------------
User.findById(bob._id)
.populate({ path: "posts", select: "files commentsCount likesCount" })
    .populate({ path: "savedPosts", select: "files commentsCount likesCount" })
    .populate({ path: "followers", select: "avatar username fullname" })
    .populate({ path: "following", select: "avatar username fullname" })
    .lean()  //Enabling the lean option tells Mongoose to skip instantiating a full Mongoose
document and just give you the POJO (So you cannot use Save() and other document methods)
    .exec();
// -------------------------------Find By Id and update ------------------------------------
-------
User.findByIdAndUpdate(req.params.id, {
    $push: { followers: req.user.id },
    $inc: { followersCount: 1 },
  });

// ----------------------------Find by Id and Update with options --------------------------
-----------
User.findByIdAndUpdate(
    req.user.id,
    {
      $set: { ...fieldsToUpdate, website, bio },
    },
    {
      new: true,
      runValidators: true,
    }
  ).select("avatar username fullname email bio website");

// ---------------------------Find and Pull ---------------------------------------------
User.findByIdAndUpdate(req.user.id, {
    $pull: { posts: req.params.id },
    $inc: { postCount: -1 },
  });


// ------------------------------Find and Populate exec ------------------------------------
-----
  let post = await Post.create({ caption, files, tags, user });

  await User.findByIdAndUpdate(req.user.id, {
    $push: { posts: post._id },
    $inc: { postCount: 1 },
  });

  post = await post
    .populate({ path: "user", select: "avatar username fullname" })
    .execPopulate(); // execPopulate is used to explicitly execute populate on query
```

Mongoose Document Methods

- `save([callback])` : save the document; callback has error, doc and count arguments
- `set(path, val, [type], [options])` : set value on the doc's property
- `get(path, [type])` : get the value
- `isModified([path])` : check if the property has been modified
- `populate([path], [callback])` : populate reference

- `validate(callback)` : validate the document

```
// ---------------------------------------------------------------------------
  let post = await Post.create({ caption, files, tags, user });

  await User.findByIdAndUpdate(req.user.id, {
    $push: { posts: post._id },
    $inc: { postCount: 1 },
  });

  post = await post
    .populate({ path: "user", select: "avatar username fullname" })
    .execPopulate(); // execPopulate is used to explicitly execute populate on query
```

Learning, Javascript

mongoose mongodb orm cheatsheet

Share:

OLDER
First missing positive

NEWER
Mordern C++ Tricks

Further Reading

1 year ago
TypeORM CheatSheet
Setting up TypeORM: Cheatsheet [TOC] Connect with database 1 2 …

1 year ago
20 Days of openCV - Day 1
20 Days of openCV - Day 1 This blog is based on Ardrian R. Face…

1 year ago
Visualize Sorting Algorithm using React
Introduction We all have implemented some sorting…

Bhargav Lad

CS Graduate Student.