Full Stack III - Lecture 6.2

Mongoose CRUD & Queries

Topics

- **CRUD Operations**
- **Queries**

Mongoose Model CRUD

Mongoose CREATE Methods

- save() - is used to insert a record in collection. It is used with Model object. ie. we have to create a model object
- insert(), insertMany() or create() - are used to insert arrays of records into collection
- insertOne() - insert single record in collection, like save() but faster

```
let Newuser = new User(user); // this is modal object.
Newuser.save()
  .then((data)=> {
    console.log(data);
   })
  .catch((err)=> {
    console.log(err);
  })
```

Mongoose UPDATE Methods

- update() - used to update a record based on condition
- findOneAndUpdate() - combination of read and update a record in collection
- findByIdAndUpdate() - used to update a record based on Id

```
Tank.updateOne({ size: 'large' }, { name: 'T-90' }, function(err, res) {
  // Updated at most one doc, `res.modifiedCount` contains the number
  // of docs that MongoDB updated
});
```

Model.update(conditions, update, [options], [callback])

```
var Student = require('../models/student.model.js');

var condition = { name: 'Captain Jack' };
var update = { City: 'Montreal' };

Student.update(condition, update, function(err, numberAffected, rawResponse) {
   // abbr..
})

// Alternative, we can still find a document..then update it
Student.findOne({ name: 'Captain Jack'}, function(err, doc) {
    doc.City = 'Montreal';
    // update the result
    doc.save(function (err) {
    // Handle errors..
    });
});
```

Mongoose DELETE Methods

- delete() - is used to delete a document based on a condition
- deleteMany() - used to remove all documents matching given filter

```
Tank.deleteOne({ size: 'large' }, function (err) {
  if (err) return handleError(err);
  // deleted at most one tank document
});
```

Model.remove(conditions, [callback])

```
var Student = require('../models/student.model.js');

Student.remove(condition, function(err) {
    // Handle error here..
 })

 // Remove any document registered on or after given date
 var regDate = new Date(2018, 9, 5);
 Student.remove({ registrationDate: { $gte: regDate }}, function(err) {
    // Handle error here..
 });

 // Execute query without a callback function - does not wait for the response
 var query = Student.remove({ registrationDate: { $gte: regDate }});
 query.exec()

 // DELETE FROM Student WHERE registrationDate >= '2018-9-5'
```

Mongoose Queries

- `Model.deleteMany()`
- `Model.deleteOne()`
- `Model.find()`
- `Model.findById()`
- `Model.findByIdAndDelete()`
- `Model.findByIdAndRemove()`
- `Model.findByIdAndUpdate()`
- `Model.findOne()`
- `Model.findOneAndDelete()`
- `Model.findOneAndRemove()`
- `Model.findOneAndUpdate()`
- `Model.replaceOne()`
- `Model.updateMany()`
- `Model.updateOne()`

- Mongoose models provide several static helper functions for CRUD operations. Each of these functions returns a mongoose Query object.

- * Remember CRUD stands for **C**reate, **U**pdate, **D**elete.

Mongoose Queries

Model.find

Model.find(conditions, [fields], [options], [callback])

```
var Student = require('../models/student.model.js');

// No callback...Deferred execution
var query = Student.find();

// With callback... Executes query immediately
Student.find(function (err, results) {
   // abbr..
})

// With callback and query conditions
Student.find({ name: 'Captain Jack' }, function (err, results) {
   // abbr..
});

// Limit the return fields in query..
Student.find({ name: 'Polly' }, 'name isActive',
    function (err, results) {
   // abbr..
});
```

- The model class exposes several static and instance methods to perform operations on the database ie. fetch
- Using find without a callback, will return a query object to be executed at a later time
- Using find with a callback will execute the query right away and then handle results or error in callback
- *Student.find() = SELECT * FROM Students*
- *Student.find({ name: 'Captain Jack'} = SELECT * FROM Students WHERE name = 'Captain Jack'*

Model.findOne and Model.findById

Model.findOne(conditions, [fields], [options], [callback])

```
var Student = require('../models/student.model.js');

// No callback... No conditions..
var query = Student.findOne();

// exec the query object
query.exec(function (err, results) {
   // handle the error... or results
});

// With conditions..
var query = Student.findOne({ name: 'Black Beard' });
```

Model.findById(conditions, [fields], [options], [callback])

```
var Student = require('../models/student.model.js');

var id = '9875c4fefefedf9c44174734a6'

// By Id... No conditions..
var query = Student.findById(id);

// exec the query object
query.exec(function (err, results) {
   // handle the error... or results
});

// Chained above method calls together
Student.findById(id).exec(function (err, results) { //... });

// By Id .. return every field EXCLUDING name..
var query = Student.findById(id, '-name');
```

Query Selectors - Comparison Query Operators

```
// Example: find students with age >= 20
Student.find({ age: { $gte: 20 },
    function(err, results) {
      if (err) throw err;
        console.log(results);
})
```

Name	Description
$eq	Matches values that are equal to a specified value.
$gt	Matches values that are greater than a specified value.
$gte	Matches values that are greater than or equal to a specified value.
$in	Matches any of the values specified in an array.
$lt	Matches values that are less than a specified value.
$lte	Matches values that are less than or equal to a specified value.
$ne	Matches all values that are not equal to a specified value.
$nin	Matches none of the values specified in an array.

Logical Query Operators

```
Test.find({
    $and: [
        { $or: [{a: 1}, {b: 1}] },
        { $or: [{c: 1}, {d: 1}] }
    ]
}, function (err, results) {
    ...
}
```

Name	Description
$and	Joins query clauses with a logical AND returns all documents that match the conditions of both clauses.
$not	Inverts the effect of a query expression and returns documents that do *not* match the query expression.
$nor	Joins query clauses with a logical NOR returns all documents that fail to match both clauses.
$or	Joins query clauses with a logical OR returns all documents that match the conditions of either clause.

Model.where(path, [val])

```
Student.find({ age: { $gte: 20, $lt: 35 }, function(err, results) {
    if (err) throw err;
   console.log(results);
})

// where..chain the selectors and execute the callback right away.
Student.where('age').gte(20).lt(35).exec(function(err, results) {
    if (err) throw err;
   console.log(results);
})

// Chain multiple where methods together
Student.where('age').gte(20).lt(35)
  .where('city', 'Toronto')
  .exec(function(err, results) {
 if (err) throw err;
 console.log(results);
});

// SELECT * FROM Students WHERE age >= 20 AND age < 35 AND city = 'Toronto'
```