Mongoose CRUD Methods: A Detailed Guide (Using Promises)

Mongoose offers several methods to perform CRUD operations. This guide compares these methods using promise-based syntax, which is more modern and clearer for students.

1. Insertion Methods

a. Using doc.save()

- Usage:

1. Create a new instance.
2. Save it using .save(), which returns a promise.

- Example:

```
const MyModel = require('./models/myModel');
const doc = new MyModel({ field: 'value' });
doc.save()
 .then(savedDoc => {
 console.log('Saved via save():', savedDoc);
 })
 .catch(err => {
 console.error('Error in save():', err);
 });
```

b. Using Model.create()

- Usage: Directly creates and saves a document in one call. Accepts a single object or an array.
- Example:

```
const MyModel = require('./models/myModel');
MyModel.create({ field: 'value' })
 .then(createdDoc => {
 console.log('Created via create():', createdDoc);
 })
 .catch(err => {
 console.error('Error in create():', err);
 });
```

Comparison:

- Instantiation:
- .save(): Requires explicit instantiation before saving.
- .create(): Does both creation and saving in one step.
- Middleware: Both trigger pre-save middleware.
- Batch Input: .create() can handle an array of objects.

2. Retrieval Methods

a. Model.find()

- Usage: Retrieves an array of documents matching the query.
- Example:

```
MyModel.find({ field: 'value' })
 .then(docs => {
 console.log('Found documents:', docs);
 })
 .catch(err => {
 console.error('Error in find():', err);
 });
```

b. Model.findOne()

- Usage: Retrieves the first document matching the query.
- Example:

```
MyModel.findOne({ field: 'value' })
 .then(doc => {
 console.log('Found document:', doc);
 })
 .catch(err => {
 console.error('Error in findOne():', err);
 });
```

c. Model.findById()

- Usage: Retrieves a document by its unique _id.
- Example:

```
MyModel.findById('60d123abc456')
 .then(doc => {
 console.log('Found by ID:', doc);
 })
 .catch(err => {
 console.error('Error in findById():', err);
 });
```

3. Update Methods

a. Using Document Modification and .save()

- Usage: Retrieve a document, modify its properties, then call .save().
- Example:

```
MyModel.findById('60d123abc456')
 .then(doc => {
 if (!doc) {
 throw new Error('Document not found');
 }
 doc.field = 'new value';
 doc.updatedAt = Date.now();
 return doc.save();
 })
 .then(updatedDoc => {
 console.log('Updated via save():', updatedDoc);
 })
 .catch(err => {
 console.error('Error in update via save():', err);
 });
```

b. Using Model.updateOne()

- Usage: Updates one document matching the filter. Returns the result (not the updated document by default).
- Example:

```
MyModel.updateOne({ _id: '60d123abc456' }, { field: 'new value',
updatedAt: Date.now() })
 .then(result => {
 console.log('Update result:', result);
 })
 .catch(err => {
 console.error('Error in updateOne():', err);
 });
```

c. Using Model.findOneAndUpdate()

- Usage: Finds a document, updates it, and returns the updated document.
- Example:

```
MyModel.findOneAndUpdate(
 { _id: '60d123abc456' },
 { field: 'new value', updatedAt: Date.now() },
 { new: true }
)
 .then(updatedDoc => {
 console.log('Updated via findOneAndUpdate:', updatedDoc);
 })
 .catch(err => {
 console.error('Error in findOneAndUpdate():', err);
 });
```

d. Using Model.findByIdAndUpdate()

- Usage: A shorthand method to update a document by its _id.
- Example:

```
MyModel.findByIdAndUpdate(
 '60d123abc456',
 { field: 'new value', updatedAt: Date.now() },
 { new: true }
)
 .then(updatedDoc => {
 console.log('Updated via findByIdAndUpdate:', updatedDoc);
 })
 .catch(err => {
 console.error('Error in findByIdAndUpdate():', err);
 });
```

4. Deletion Methods

a. Using Model.deleteOne()

- Usage: Deletes a single document matching the filter.
- Example:

```
MyModel.deleteOne({ _id: '60d123abc456' })
 .then(result => {
 console.log('Deleted via deleteOne():', result);
 })
 .catch(err => {
 console.error('Error in deleteOne():', err);
 });
```

b. Using Model.deleteMany()

- Usage: Deletes all documents that match the filter.
- Example:

```
MyModel.deleteMany({ field: 'value' })
 .then(result => {
 console.log('Deleted via deleteMany():', result);
 })
 .catch(err => {
 console.error('Error in deleteMany():', err);
 });
```

Summary:

- Insertion:
- Use .save() for an instance that is already created.
- Use Model.create() for a one-step creation and save.
- Retrieval:
- Use find(), findOne(), or findById() based on whether you need multiple documents or a single document.
- Updating:
- Modify the document instance and call .save(), or use update methods (updateOne(), findOneAndUpdate(), findByIdAndUpdate()) for one-step updates.
- Ensure to set updatedAt to Date.now() if you want to update the timestamp.
- Deletion:
- Use deletion methods like deleteOne(), deleteMany(), findOneAndDelete(), or findByIdAndDelete() to remove documents.