Basic Operations

Mongoose has a flexible API and provides many ways to accomplish a task. We will not focus on the variations because that is out of the scope of this course. However, try to remember that most of the operations can be done in more than one way either syntactically or via the application's architecture.

Create Record

Let's create an instance of the email model and save it to the database:

```
let EmailModel = require('./email')
let msg = new EmailModel({
  email: 'ADA.LOVELACE@GMAIL.COM'
})
msg.save()
   .then(doc => {
     console.log(doc)
   })
   .catch(err => {
     console.error(err)
   })
```

The result is a document that is returned upon a successful save:

```
{
  _id: 5a78fe3e2f44ba8f85a2409a,
  email: 'ada.lovelace@gmail.com',
  __v: 0
}
```

The following fields are returned (internal fields are prefixed with an underscore):

1. The `_id` field is auto-generated by Mongo and is a primary key of the collection. Its value is a unique identifier for the document.
2. The value of the `email` field is returned. Notice that it is lower-case because we specified the `lowercase:true` attribute in the schema.
3. `__v` is the versionKey property set on each document when first created by Mongoose. Its value contains the internal revision of the document.

If you try to repeat the save operation above, you will get an error because we have specified that the email field should be unique.

311790:0

Basic Operations

Fetch Record

Let's try to retrieve the record we saved to the database earlier. The model class exposes several static and instance methods to perform operations on the database. We will now try to find the record that we have created previously using the find method and pass the email as the search term.

```
EmailModel
  .find({
    email: 'ada.lovelace@gmail.com'    // search query
  })
  .then(doc => {
    console.log(doc)
  })
  .catch(err => {
    console.error(err)
  })
```

The document returned will be similar to what was displayed when we created the record:

```
{
  _id: 5a78fe3e2f44ba8f85a2409a,
  email: 'ada.lovelace@gmail.com',
```

```
  __v: 0
}
```

Basic Operations

Update Record

Let's modify the record above by changing the email address and adding another field to it, all in a single operation. For performance reasons, Mongoose won't return the updated document so we need to pass an additional parameter to ask for it:

```
EmailModel
  .findOneAndUpdate(
    {
      email: 'ada.lovelace@gmail.com'  // search query
    },
    {
      email: 'theoutlander@live.com'   // field:values to update
    },
    {
      new: true,                       // return updated doc
      runValidators: true              // validate before update
    })
  .then(doc => {
    console.log(doc)
  })
  .catch(err => {
    console.error(err)
  })
```

The document returned will contain the updated email:

```
{
  _id: 5a78fe3e2f44ba8f85a2409a,
  email: 'theoutlander@live.com',
```

```
  __v: 0
}
```

Basic Operations

Delete Record

We will use the `findOneAndRemove` call to delete a record. It returns the original document that was removed:

```
EmailModel
  .findOneAndRemove({
    email: 'theoutlander@live.com'
  })
  .then(response => {
    console.log(response)
  })
  .catch(err => {
    console.error(err)
  })
```

‹ Previous next ›