ADDING A MONGO DB

Agenda

- Last Week
- What is MongoDB?
- Creating a View Page
- Deleting Items
- Adding a Database
- Assignment #1
- In-Class

What did we cover last week?

What was the significance?

Home About Templates Contact Add A Dog

Our List of Pups

Name: Franky

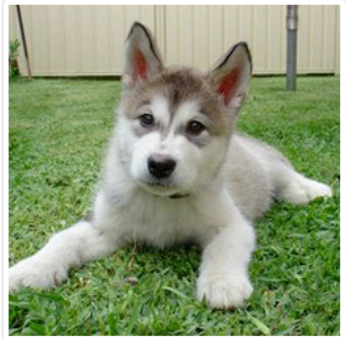
Owner: The Hoff

Details

Name: Milo

Owner: Rob

Details



Name: Isis

Owner: Pam Anderson

Details

Name: Billie

Owner: Annie Jones

Details

Completion of our Dog App

We added a view a single record and a delete button.

Today we will add a database to our dog application.

We will look at how to view, add and delete with MongoDB.

Adding a database with Nodejs

You can connect to your DB using NPM. Multiple databases are supported. “app.js”.

https://expressjs.com/en/guide/database-integration.html

NoSQL Databases

The characteristics of a NoSQL database are:

- Data stored in key-value pairs
- Easily scalable (Using multiple servers)
- Fast data write capabilities

MongoDB Advantages

Some advantages of MongoDB:

- Data stored in key-value pairs
- Schema less – MongoDB uses documents. Number of fields, content and size of the document can differ from one document to another.
- Deep query-ability; almost as powerful as SQL.
- Easy to scale across multiple servers.
- Fast access of data.
- Data is stored in the form of JSON style documents.

How many of you installed MongoDB?

Be honest…

How to Install MongoDB

https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-os-x/

http://treehouse.github.io/installation-guides/mac/mongo-mac.html

http://treehouse.github.io/installation-guides/windows/mongo-windows.html

Download the binary files for the desired release of MongoDB.

Download the binaries from https://www.mongodb.org/downloads.
For example, to download the latest release through the shell, issue the following

```
curl -O https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-3.2.11.tgz
```

Extract the files from the downloaded archive

For example, from a system shell, you can extract through the tar command:

```
tar -zxvf mongodb-osx-x86_64-3.2.11.tgz
```

Copy the extracted archive to the target directory

Copy the extracted folder to the location from which MongoDB will run.

```
mkdir -p mongodb
cp -R -n mongodb-osx-x86_64-3.2.11/ mongodb
```

How to Run MongoDB

https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-os-x/

Create the data directory

Before you start MongoDB for the first time, create the directory to which the **mongod** process will write data.

The following example command creates the data directory in your home folder

```
mkdir -p ~/data
```

NOTE: You can save the data to a different location by changing the directory.

Run the Mongo daemon

This will start the mongo server.

–dbpath=data instructs the **daemon** to use a custom data directory.
–port 27017 instructs the **daemon** to use a custom port

```
~/mongodb/bin/mongod --dbpath=data --port 27017
```

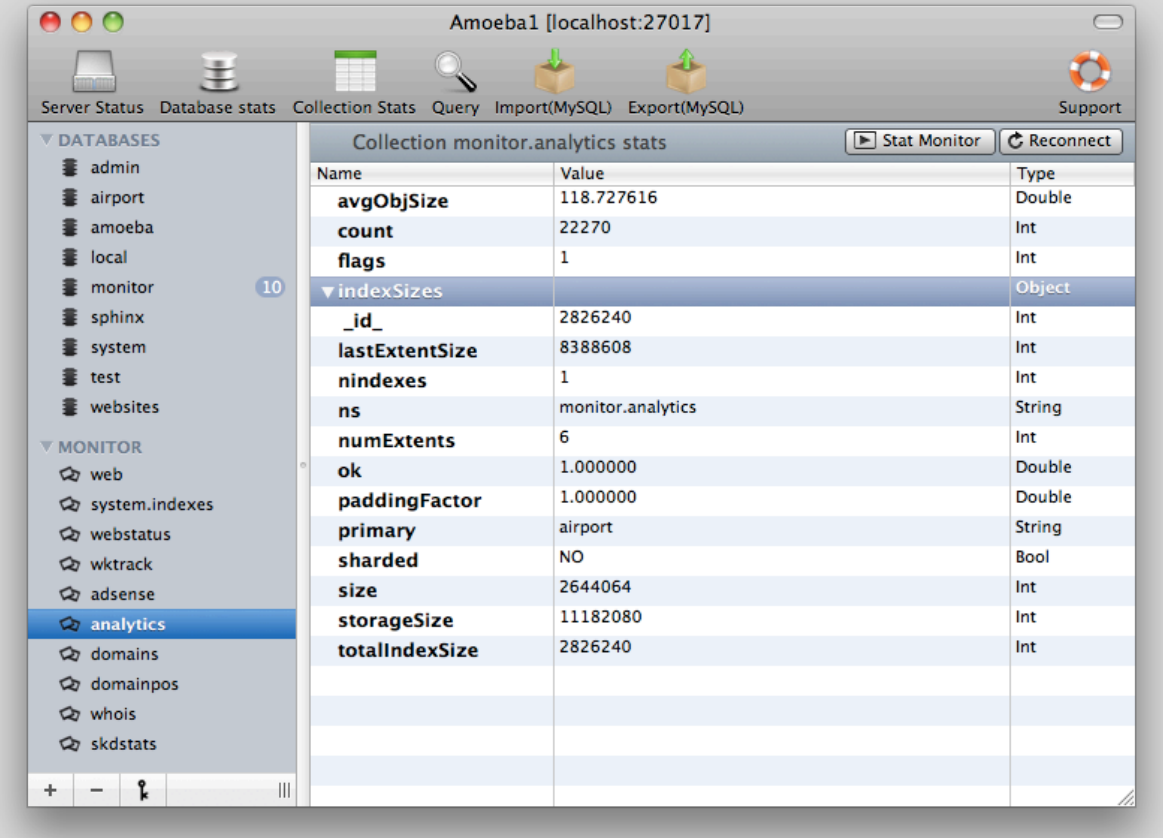
Confirm Your MongoDB Port

All Example code uses 27017. If yours is different, you need to update the port in all code examples.

NOTE: THIS STEP IS OPTIONAL

MongoHub

is a native mac GUI application for mongodb, designed for mongodb administration.



NOTE: THIS STEP IS OPTIONAL

Run the Mongo shell (if you don't like MongoHub)

This will start the mongo shell where you can interact directly with your MongoDB.

For a list of commands check out the following link:
https://docs.mongodb.com/v3.2/reference/mongo-shell/

```
~/mongodb/bin/mongo
```

Let's Connect to Our MongoDB

Make sure you are running the mongo daemon

Install the MongoDB module

Use NPM to install the module.

```
> npm install mongodb
```

Load the MongoDB Packages

Open the corresponding file and add the following code (shown in blue).

```
// ADD MONGODB PACKAGES CODE BELOW -- START

var mongodb = require('mongodb');
var MongoClient = mongodb.MongoClient;
var url = 'mongodb://localhost:27017/dog_list';

// ADD CODE ABOVE THE COMMENTS -- END
```

wk5-01.js

Important connection notes

Make sure your ports match.

```
// ADD MONGODB PACKAGES CODE BELOW -- START

var mongodb = require('mongodb');
var MongoClient = mongodb.MongoClient;
var url = 'mongodb://localhost:27017/dog_list';

// ADD CODE ABOVE THE COMMENTS -- END
```

wk5-01.js

Important connection notes

The name of your database is the last part of the URL. IF the database doesn't exist, MongoDB will automatically create it for you.

```
// ADD MONGODB PACKAGES CODE BELOW -- START

var mongodb = require('mongodb');
var MongoClient = mongodb.MongoClient;
var url = 'mongodb://localhost:27017/dog_list';

// ADD CODE ABOVE THE COMMENTS -- END
```

wk5-01.js

Connect to the MongoDB

Open the corresponding file and add the following code (shown in blue).

```
// ADD THE MONGODB CONNECTION CODE BELOW -- START

MongoClient.connect(url, function (err, db) {
    if (err) {
        console.log('Unable to connect to the mongoDB server. Error:', err);
    }
    else {
        console.log('Connection established to', url);
        app.locals.db = db;
    }
});

// ADD CODE ABOVE THE COMMENTS -- END
```

wk5-01.js

Important Connection Notes

Our URL is the variable we created in the previous step. So we use a variable instead of 'mongodb://localhost:27017/dog_list'

```
// ADD THE MONGODB CONNECTION CODE BELOW -- START

MongoClient.connect(url, function (err, db) {
    if (err) {
        console.log('Unable to connect to the mongoDB server. Error:', err);
    }
    else {
        console.log('Connection established to', url);
        app.locals.db = db;
    }
});

// ADD CODE ABOVE THE COMMENTS -- END
```

wk5-01.js

app.locals creates a Global Variable

This enables us to access the db data elsewhere in our code.

```
// ADD THE MONGODB CONNECTION CODE BELOW -- START

MongoClient.connect(url, function (err, db) {
    if (err) {
        console.log('Unable to connect to the mongoDB server. Error:', err);
    }
    else {
        console.log('Connection established to', url);
        app.locals.db = db;
    }
});

// ADD CODE ABOVE THE COMMENTS -- END
```

wk5-01.js

Launch the app.js

Open a browser and visit: http://localhost:3000. Check the console of your node terminal. You should see a connection message. Also look at your MongoDB terminal window. It should show a new connection.

```
$ node app.js
```

That's the Hard Part!

If we are connected to our MongoDB, we can delete those pesky JSON files and use less code!

Populate Our MongoDB with Sample Data

Run the “wk5_sample_data.js” in node. This is a program that will create some sample content for our database. We only need to run this once.

```
$ node wk5_sample_data.js
```

A MongoDB Has Collections & Documents

A single database can have multiple collections, each containing different document structures.

Database: dog_list
Collection: dogs
Documents: {JSON data}

Look at our “wk5_sample_data.js”

Here we can see the database, collection and documents.

Line 4: Database

```
var url = 'mongodb://localhost:27017/dog_list';
```

Line 14: Collection

```
var collection = db.collection('dogs');
```

Line 16-39: Documents

```
var user1 = {"name": "Milo", "breed": "Lab/Sharpei" ... "owner":"Rob"};
```

Let's Load Our Dogs from MongoDB

For the purposes of this lecture, and cleaner code, I have deleted the opening/closing of JSON files.

Loading All The Documents From a Collection

We can use collection.find() to search a MongoDB for all documents.

```
// ADD OUR GET.("/") - HOMEPAGE CODE BELOW -- START
app.get('/', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-02.js

With MongoDB, We Don't Need the Load Our Dogs from MongoDB

For the purposes of this lecture, and cleaner code, I have deleted the opening/closing of JSON files.

Use Our Global Variable to Access the Database

We create a local variable to make our code shorter. We could write req.app.locals.db.collection but that is super long!

```
// ADD OUR GET.("/") - HOMEPAGE CODE BELOW -- START
app.get('/', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-02.js

Search Our dogs Collection

The .toArray() converts the data to an JSON array.

```
// ADD OUR GET.("/") - HOMEPAGE CODE BELOW -- START
app.get('/', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-02.js

Send the Data to Render A Pug Template

This part should look familiar.

```
// ADD OUR GET.("/") - HOMEPAGE CODE BELOW -- START
app.get('/', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-02.js

!important An Update to Our index.pug and all other templates

When we sent the data from our JSON files to our templates, we needed to JSON.parse the data. With MongoDB, we no longer need to do this.

Index.pug: 13 – Last Week's Files wk4/views/index.pug

```
-var dogs = JSON.parse(users)
```

Index.pug: 13 – This Week's Files wk5/views/index.pug

```
-var dogs = users
```

Launch the app.js

Open a browser and visit: http://localhost:3000. You should see the dogs from the sample content we added to our database.

```
$ node app.js
```

Let's Include the Add Page

For the purposes of this lecture, and cleaner code, I have deleted the opening/closing of JSON files.

Add a Get For the addDog Page

This is the same as our old code.

```
// ADD GET.("/addDog") - VIEW CODE BELOW -- START
app.get('/addDog', function (req, res) {
    res.render('add', {
        title: "Add A Dog"
    });
});
// ADD GET.("/addDog") - VIEW CODE ABOVE -- START
```

wk5-03.js

Add a Post For the addDog Page

We are doing two things. First we write the submitted information to the database using collection.insert(). Then, we use collection.find() to load all documents and load the index.pug template.

```
// ADD POST.("/") - CODE BELOW -- START
app.post('/', function (req, res) {
    req.body.id = randomID(10);
    req.body.img = "/images/doggo.gif";
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.insert(req.body, function (err, result) {
            console.log('Inserted %d documents into the "dogs" collection:', result.length, result);
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , success: "Your Dog Was Added to the List"
            });
        });
    });
});
// ADD POST.("/") - CODE ABOVE -- START
```

wk5-03.js

Insert the Posted Data to the MongoDB.

We use collection.insert(), to add the submitted data, req.body, to the dogs collection.

```
// ADD POST.("/") - CODE BELOW -- START
app.post('/', function (req, res) {
    req.body.id = randomID(10);
    req.body.img = "/images/doggo.gif";
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.insert(req.body, function (err, result) {
            console.log('Inserted %d documents into the "dogs" collection:', result.length, result);
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , success: "Your Dog Was Added to the List"
            });
        });
    });
});
// ADD POST.("/") - CODE ABOVE -- START
```

wk5-03.js

Search Our dogs Collection

The .toArray() converts the data to an JSON array.

```
// ADD POST.("/") - CODE BELOW -- START
app.post('/', function (req, res) {
    req.body.id = randomID(10);
    req.body.img = "/images/doggo.gif";
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.insert(req.body, function (err, result) {
            console.log('Inserted %d documents into the "dogs" collection:', result.length, result);
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , success: "Your Dog Was Added to the List"
            });
        });
    });
});
// ADD POST.("/") - CODE ABOVE -- START
```

wk5-03.js

Send the Data to Render A Pug Template

This part should look familiar.

```
// ADD POST.("/") - CODE BELOW -- START
app.post('/', function (req, res) {
    req.body.id = randomID(10);
    req.body.img = "/images/doggo.gif";
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.insert(req.body, function (err, result) {
            console.log('Inserted %d documents into the "dogs" collection:', result.length, result);
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , success: "Your Dog Was Added to the List"
            });
        });
    });
});
// ADD POST.("/") - CODE ABOVE -- START
```

wk5-03.js

Launch the app.js

Open a browser and visit: http://localhost:3000. You should be able to add a dog to your database.

```
$ node app.js
```

BREAK

See you in 10-15 minutes

Let's Add the Detail View

For the purposes of this lecture, and cleaner code, I have deleted the opening/closing of JSON files.

Add a Get For the Detail Pages

We will use the collection.find() to search for the req.params.id (dog id) in the database.

```
// ADD OUR GET.("/:id") - VIEW CODE BELOW -- START
app.get('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find({
            id: req.params.id
        }).toArray(function (err, item) {
            console.log(item);
            res.render('view', {
                users: item
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-04.js

Search Our dogs Collection for the id

The .toArray() converts the data to an JSON array.

```
// ADD OUR GET.("/:id") - VIEW CODE BELOW -- START
app.get('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find({
            id: req.params.id
        }).toArray(function (err, item) {
            console.log(item);
            res.render('view', {
                users: item
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-04.js

Send the Data to Render A Pug Template

This part should look familiar.

```
// ADD OUR GET.("/:id") - VIEW CODE BELOW -- START
app.get('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.find({
            id: req.params.id
        }).toArray(function (err, item) {
            console.log(item);
            res.render('view', {
                users: item
            });
        });
    });
});
// ADD OUR GET.("/") - HOMEPAGE CODE ABOVE -- END
```

wk5-04.js

!important An Update to Our view.pug template

When we loaded the data last week, we did not need the index of the array. This time we do.

view.pug: 12 – Last Week's Files wk4/views/view.pug

```
h1 All About #{dogs.name}
```

Index.pug: 13 – This Week's Files wk5/views/view.pug

```
h1 All About #{dogs[0].name}
```

Launch the app.js

Open a browser and visit: http://localhost:3000. You should be able to view the details of a dog from your database.

```
$ node app.js
```

Let's Delete a Document From Our Collection

For the purposes of this lecture, and cleaner code, I have deleted the opening/closing of JSON files.

Add a Delete

We will use the collection.deleteOne() to search for the req.params.id (dog id) and delete it.

```
// ADD OUR DELETE("/:id") - CODE BELOW -- START
app.delete('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.deleteOne({
            id: req.params.id
        }, function (err, result) {
           console.log('Deleted the record.');
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , deleted: "Your Dog Was Deleted From the List"
            });
        });
    });
});
// ADD OUR DELETE("/:id") - CODE ABOVE -- START
```

wk5-05.js

Search Our dogs Collection for the id and Delete

We use collection.deleteOne() to find and delete the requested dog, req.params.id

```
// ADD OUR DELETE("/:id") - CODE BELOW -- START
app.delete('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.deleteOne({
            id: req.params.id
        }, function (err, result) {
           console.log('Deleted the record.');
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , deleted: "Your Dog Was Deleted From the List"
            });
        });
    });
});
// ADD OUR DELETE("/:id") - CODE ABOVE -- START
```

wk5-05.js

Search Our dogs Collection

The .toArray() converts the data to an JSON array.

```
// ADD OUR DELETE("/:id") - CODE BELOW -- START
app.delete('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.deleteOne({
            id: req.params.id
        }, function (err, result) {
           console.log('Deleted the record.');
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , deleted: "Your Dog Was Deleted From the List"
            });
        });
    });
});
// ADD OUR DELETE("/:id") - CODE ABOVE -- START
```

wk5-05.js

Send the Data to Render A Pug Template

This part should look familiar.

```
// ADD OUR DELETE("/:id") - CODE BELOW -- START
app.delete('/:id', function (req, res) {
    const db = req.app.locals.db;
    db.collection('dogs', function (err, collection) {
        collection.deleteOne({
            id: req.params.id
        }, function (err, result) {
           console.log('Deleted the record.');
        });
        collection.find().toArray(function (err, items) {
            res.render('index', {
                users: items
                , deleted: "Your Dog Was Deleted From the List"
            });
        });
    });
});
// ADD OUR DELETE("/:id") - CODE ABOVE -- START
```

wk5-05.js

Assignment #1

You are going to use your knowledge of Node.js, Express, Pug and MongoDB to create an interactive ToDo List. The ToDo list will enable a user to VIEW all ToDo list items, DELETE an item and ADD a new item to the list. All ToDo list items are to be stored in a MongoDB. You need to create your ToDo list app using Node.js with Express and Pug templates.

See BlackBoard for more details.

DUE: FEBRUARY 10, 2017 @ 5:00 pm (at the end of class)

Do you need more time?