Video 4.6

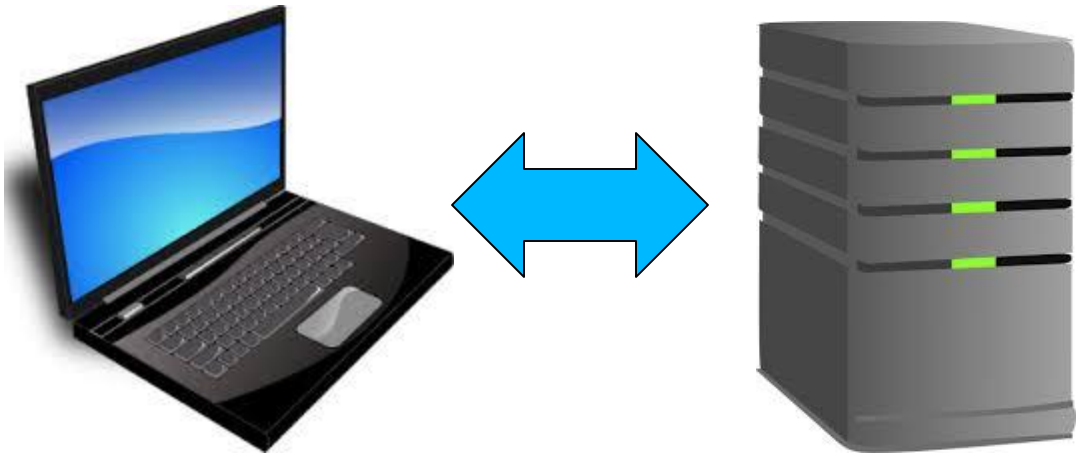
MongoDB: Setup and Insert

Chris Murphy

Review

- Node.js and Express allow us to build server-side web apps in JavaScript

- We can get data from the user via the URL query parameters or from form data



Client **Server**

Client **Server** **Database**

What is a NoSQL Database?

- A **NoSQL Database** is a database that does not use SQL, the traditional method of storing data

- In SQL, data is stored in tables and rows. This is also known as a relational database

- NoSQL Databases attempt to address some of the shortcomings of SQL and other relational databases by organizing and storing data differently

What is MongoDB?

- **MongoDB** is one NoSQL Database that is designed for use with JavaScript apps

- MongoDB stores **collections** of **documents** rather than tables of rows

SQL Database

- A SQL table of Users might look like this:

Name	Age	Country	Occupation
Jane Doe	30	United States	Programmer
John Smith	25	Canada	Doctor
Kim Jones	27	France	Painter

MongoDB Documents

- A MongoDB **collection** of User documents might look like this:

```
{
    name: 'Jane Doe',
    age: 30,
    country: 'United States',
    occupation: 'Programmer'
}
```

```
{
    name: 'John Smith',
    age: 25,
    country: 'Canada',
    occupation: 'Doctor'
}
```

```
{
    name: 'Kim Jones',
    age: 27,
    country: 'France',
    occupation: 'Painter'
}
```

Using MongoDB with a Node.js app

1. Install MongoDB locally or create an account on a remote service

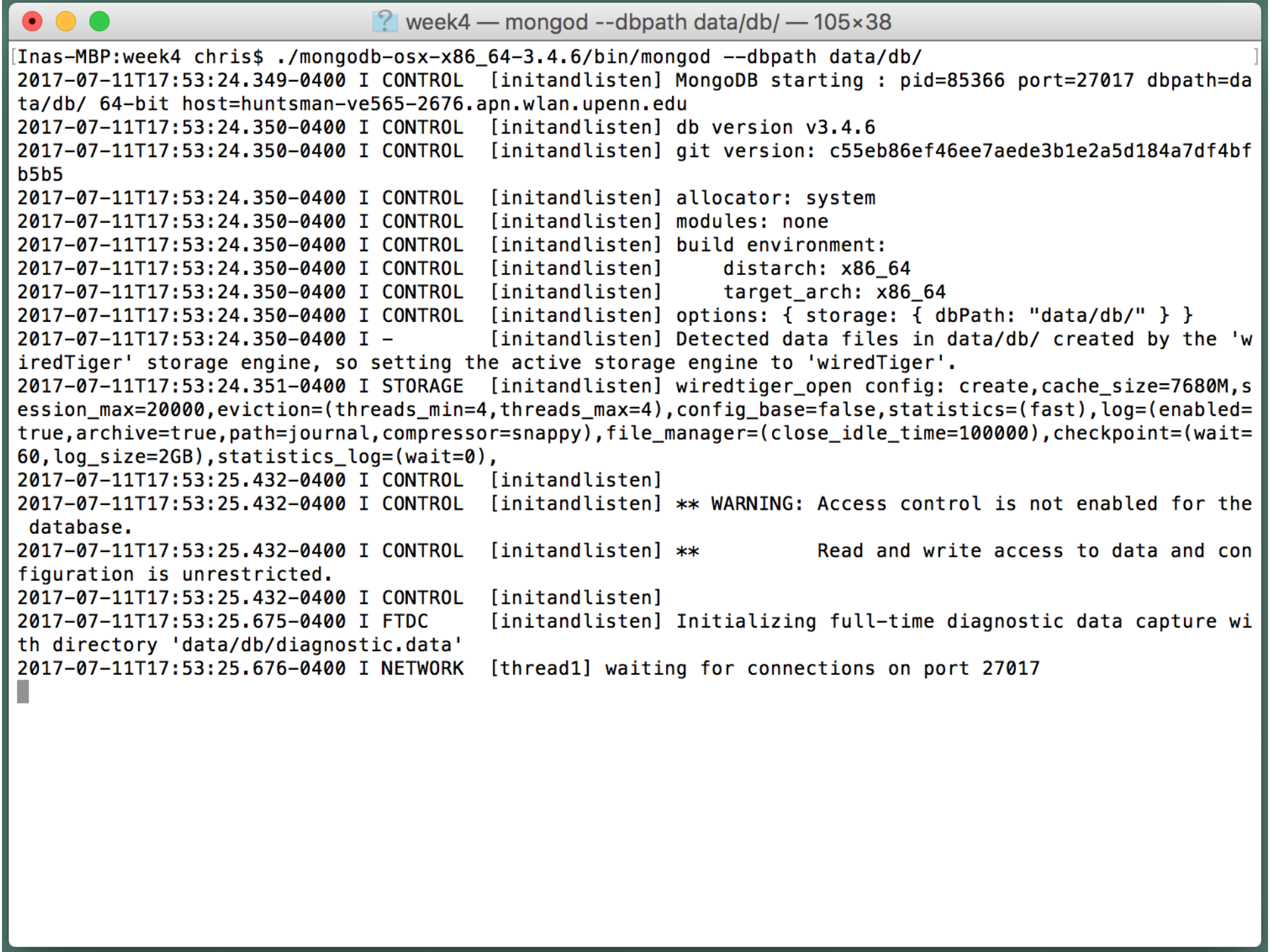
2. Install packages locally to allow your JavaScript programs to access your MongoDB instance

3. Write JavaScript to describe the Schema (blueprint for Documents) that you will use in the Collection

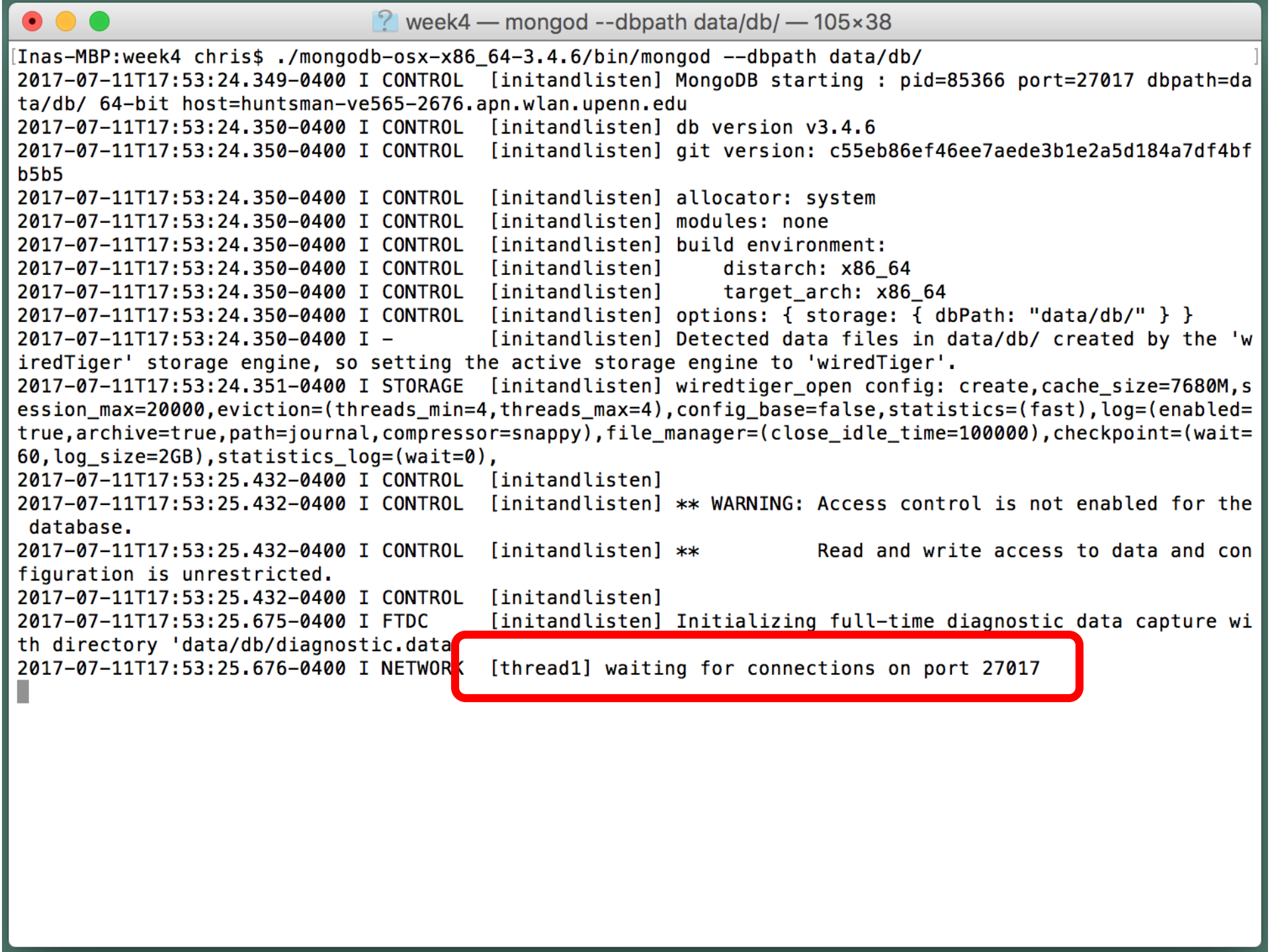
4. Use the Schema to access MongoDB in your app

Installing MongoDB

- You can find download/installation instructions for MongoDB at https://mongodb.com/download
- Follow these instructions to create a new empty database and run the MongoDB server
- When you start it, it will tell you which port it is using


week4 — mongod --dbpath data/db/ — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongod --dbpath data/db/
2017-07-11T17:53:24.349-0400 I CONTROL [initandlisten] MongoDB starting : pid=85366 port=27017 dbpath=data/db/ 64-bit host=huntsman-ve565-2676.apn.wlan.upenn.edu
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] db version v3.4.6
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bfb5b5
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] allocator: system
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] modules: none
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] build environment:
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] distarch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] target_arch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] options: { storage: { dbPath: "data/db/" } }
2017-07-11T17:53:24.350-0400 I - [initandlisten] Detected data files in data/db/ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-07-11T17:53:24.351-0400 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7680M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.675-0400 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'data/db/diagnostic.data'
2017-07-11T17:53:25.676-0400 I NETWORK [thread1] waiting for connections on port 27017

```
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongod --dbpath data/db/
2017-07-11T17:53:24.349-0400 I CONTROL  [initandlisten] MongoDB starting : pid=85366 port=27017 dbpath=data/db/ 64-bit host=huntsman-ve565-2676.apn.wlan.upenn.edu
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten] db version v3.4.6
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bfb5b5
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten] allocator: system
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten] modules: none
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten] build environment:
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten]     distarch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten]     target_arch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL  [initandlisten] options: { storage: { dbPath: "data/db/" } }
2017-07-11T17:53:24.350-0400 I -        [initandlisten] Detected data files in data/db/ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-07-11T17:53:24.351-0400 I STORAGE  [initandlisten] wiredtiger_open config: create,cache_size=7680M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.675-0400 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'data/db/diagnostic.data'
2017-07-11T17:53:25.676-0400 I NETWORK  [thread1] waiting for connections on port 27017
```


week4 — mongod --dbpath data/db/ — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongod --dbpath data/db/
2017-07-11T17:53:24.349-0400 I CONTROL [initandlisten] MongoDB starting : pid=85366 port=27017 dbpath=data/db/ 64-bit host=huntsman-ve565-2676.apn.wlan.upenn.edu
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] db version v3.4.6
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] git version: c55eb86ef46ee7aede3b1e2a5d184a7df4bfb5b5
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] allocator: system
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] modules: none
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] build environment:
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] distarch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] target_arch: x86_64
2017-07-11T17:53:24.350-0400 I CONTROL [initandlisten] options: { storage: { dbPath: "data/db/" } }
2017-07-11T17:53:24.350-0400 I - [initandlisten] Detected data files in data/db/ created by the 'wiredTiger' storage engine, so setting the active storage engine to 'wiredTiger'.
2017-07-11T17:53:24.351-0400 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=7680M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),statistics_log=(wait=0),
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.675-0400 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'data/db/diagnostic.data'
2017-07-11T17:53:25.676-0400 I NETWORK [thread1] waiting for connections on port 27017

Installing MongoDB

- You can find download/installation instructions for MongoDB at https://mongodb.com/download

- Follow these instructions to create a new empty database and run the MongoDB server
- When you start it, it will tell you which port it is using

Installing MongoDB

- You can find download/installation instructions for MongoDB at https://mongodb.com/download

- Follow these instructions to create a new empty database and run the MongoDB server
- When you start it, it will tell you which port it is using

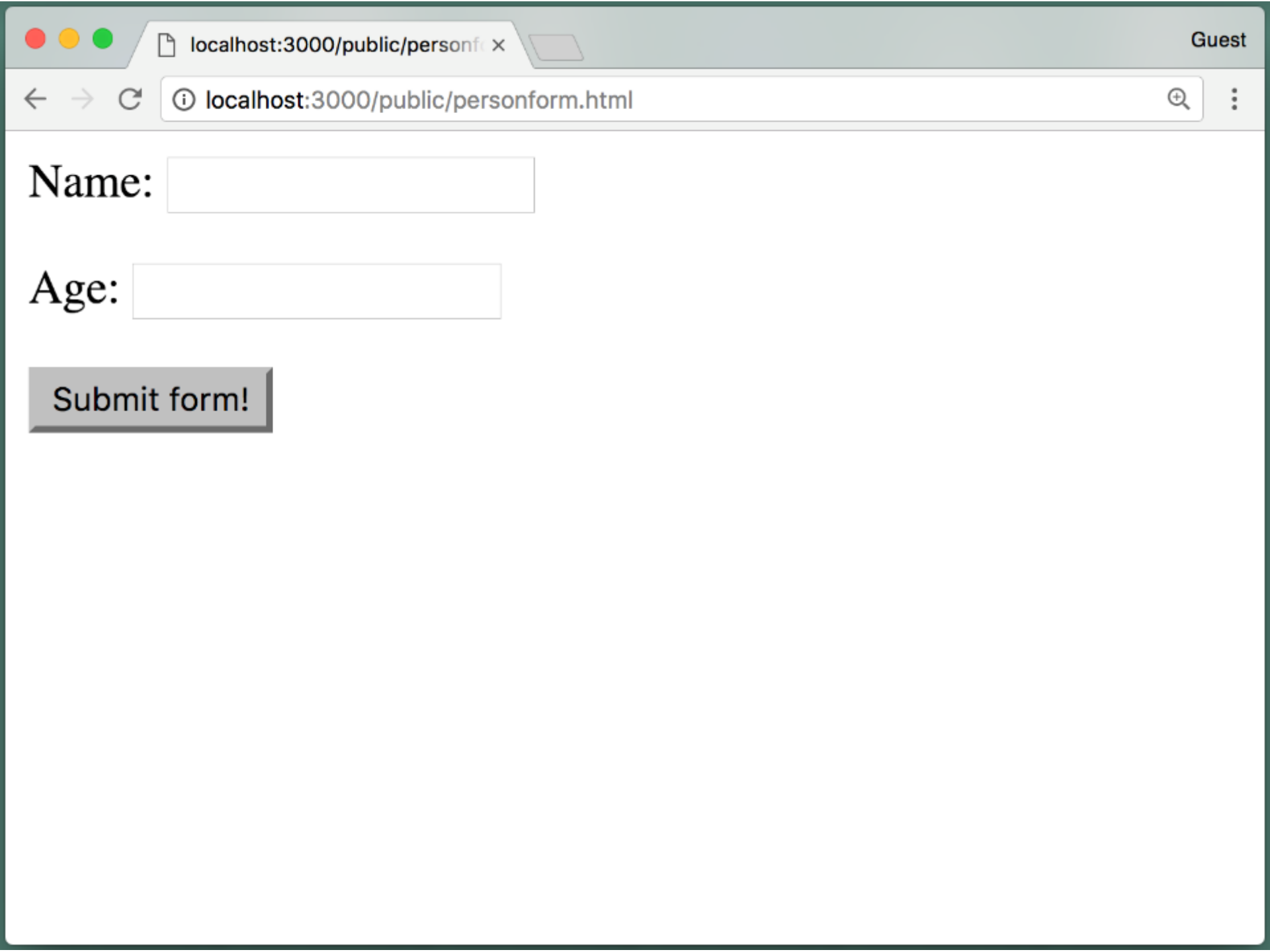
- Alternatively, you may use an online service, e.g. MongoDB Atlas

Installing Drivers for MongoDB

- You can access MongoDB directly from your Node app using the **MongoClient**

- Alternatively, you can install helper packages such as **Mongoose** to simplify some tasks:

```
npm install mongoose --save
```


localhost:3000/public/personf
Guest
localhost:3000/public/personform.html
Name:
Age:
Submit form!

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

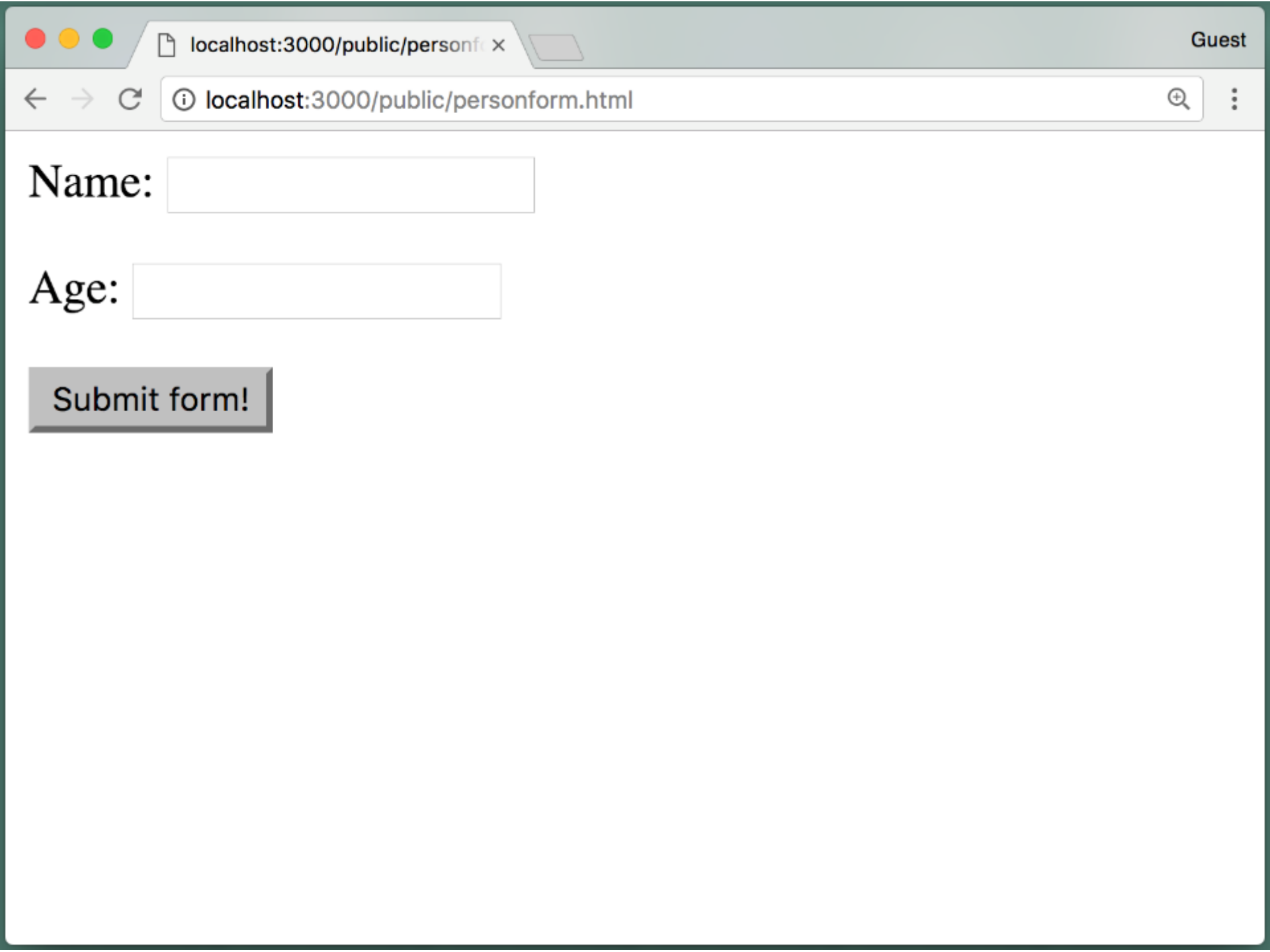
```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

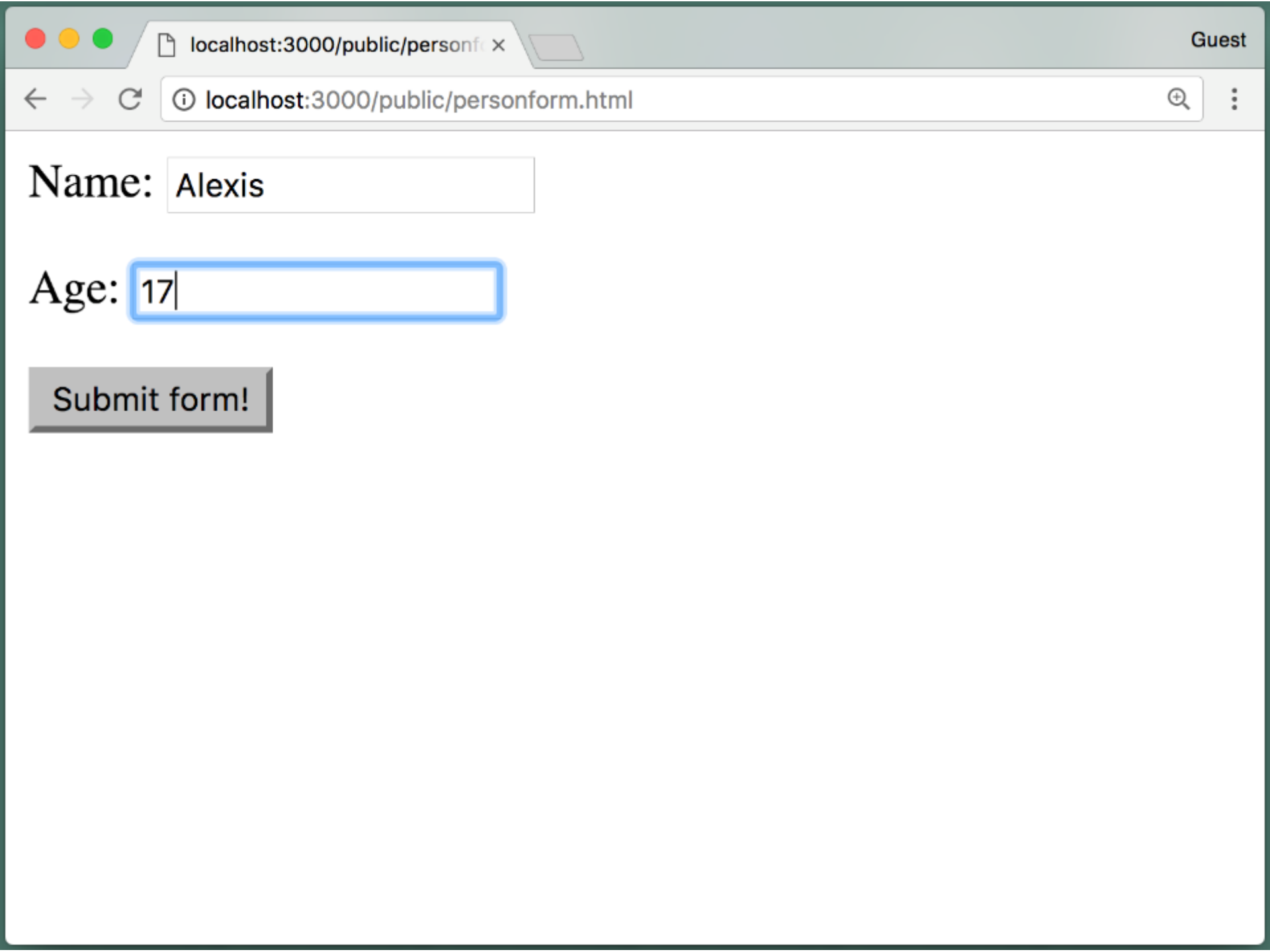
```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

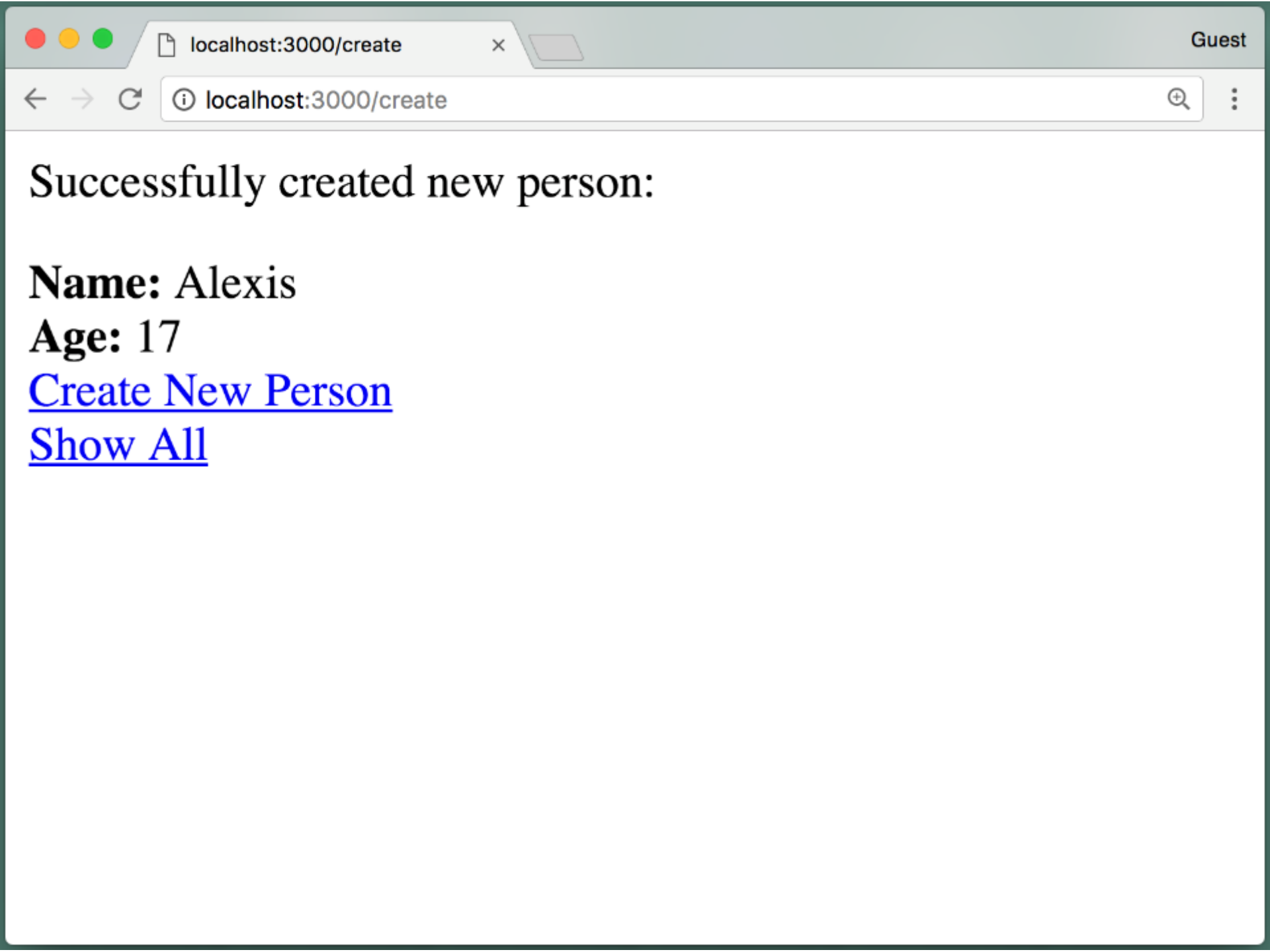
```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```


localhost:3000/public/personf
Guest
localhost:3000/public/personform.html
Name:
Age:
Submit form!


localhost:3000/public/personf
Guest
localhost:3000/public/personform.html
Name:
Alexis
Age:
17
Submit form!



Successfully created new person:

Name: Alexis
Age: 17
Create New Person
Show All

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

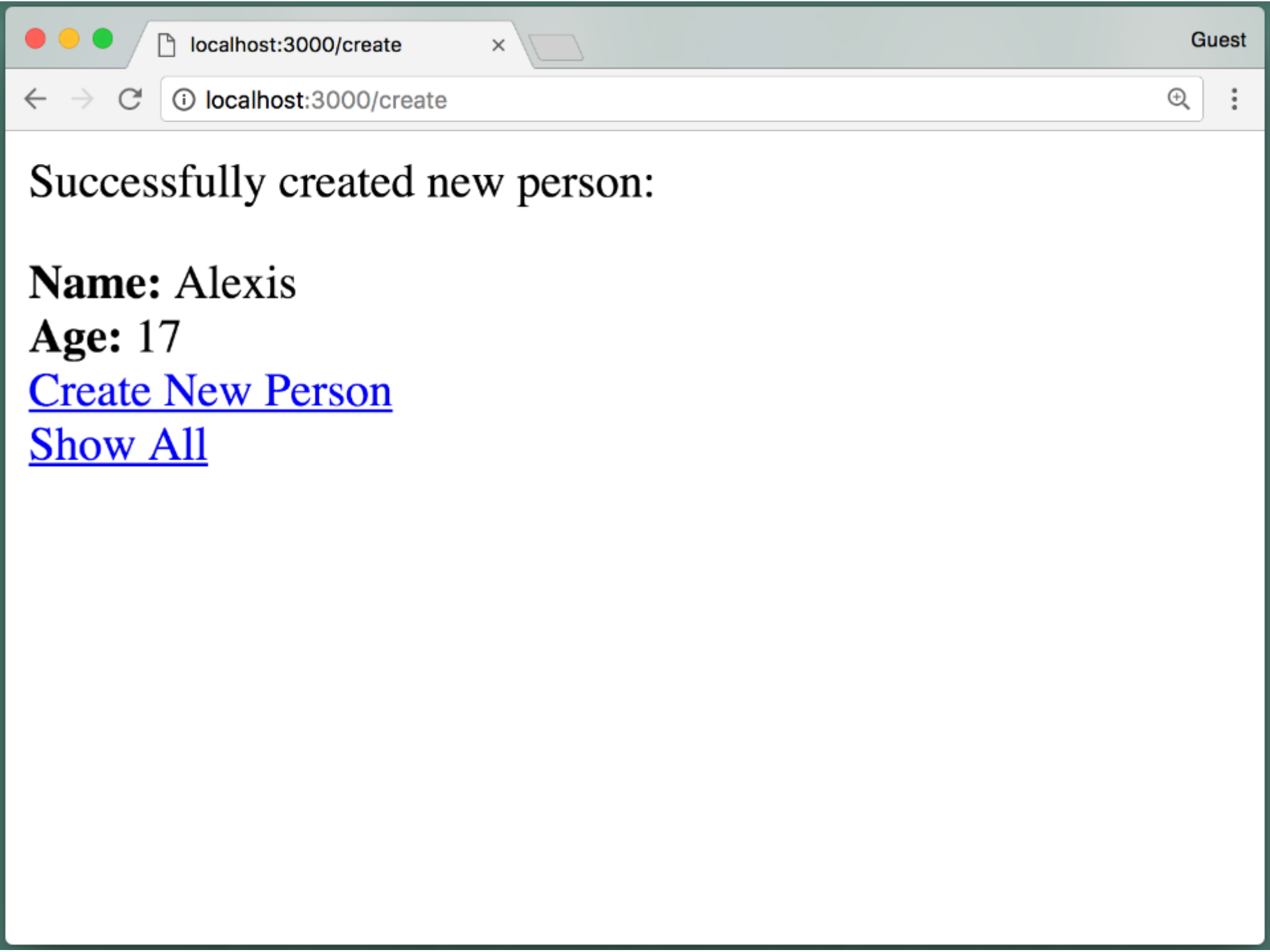
```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```

```
/* This is Person.js */

var mongoose = require('mongoose');

// note: your host/port number may be different!
mongoose.connect('mongodb://localhost:27017/myDatabase');

var Schema = mongoose.Schema;

var personSchema = new Schema( {
      name: {type: String, required: true, unique: true},
      age: Number
    } );


module.exports = mongoose.model('Person', personSchema);
```



Successfully created new person:

Name: Alexis
Age: 17
Create New Person
Show All

```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

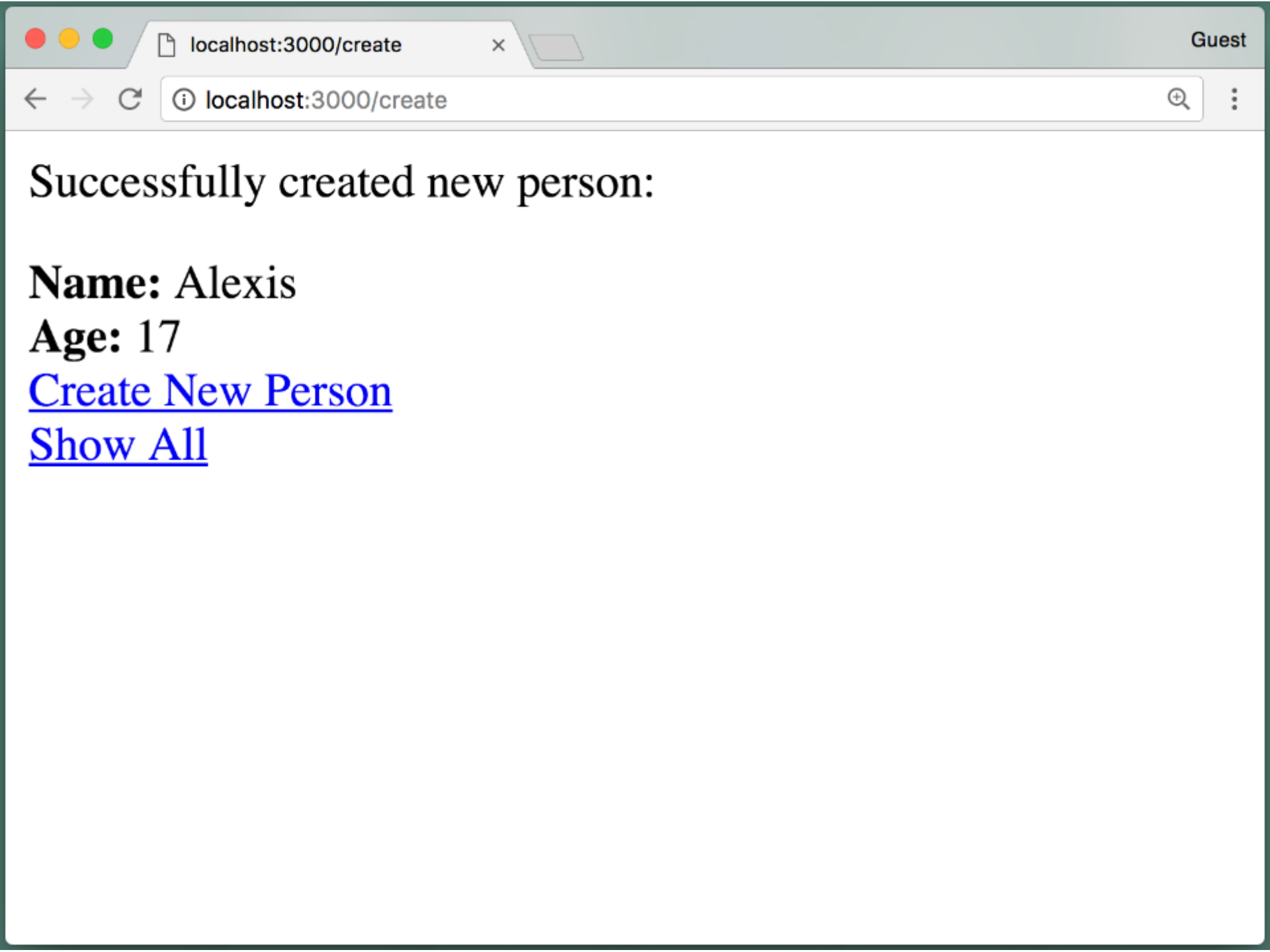
```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

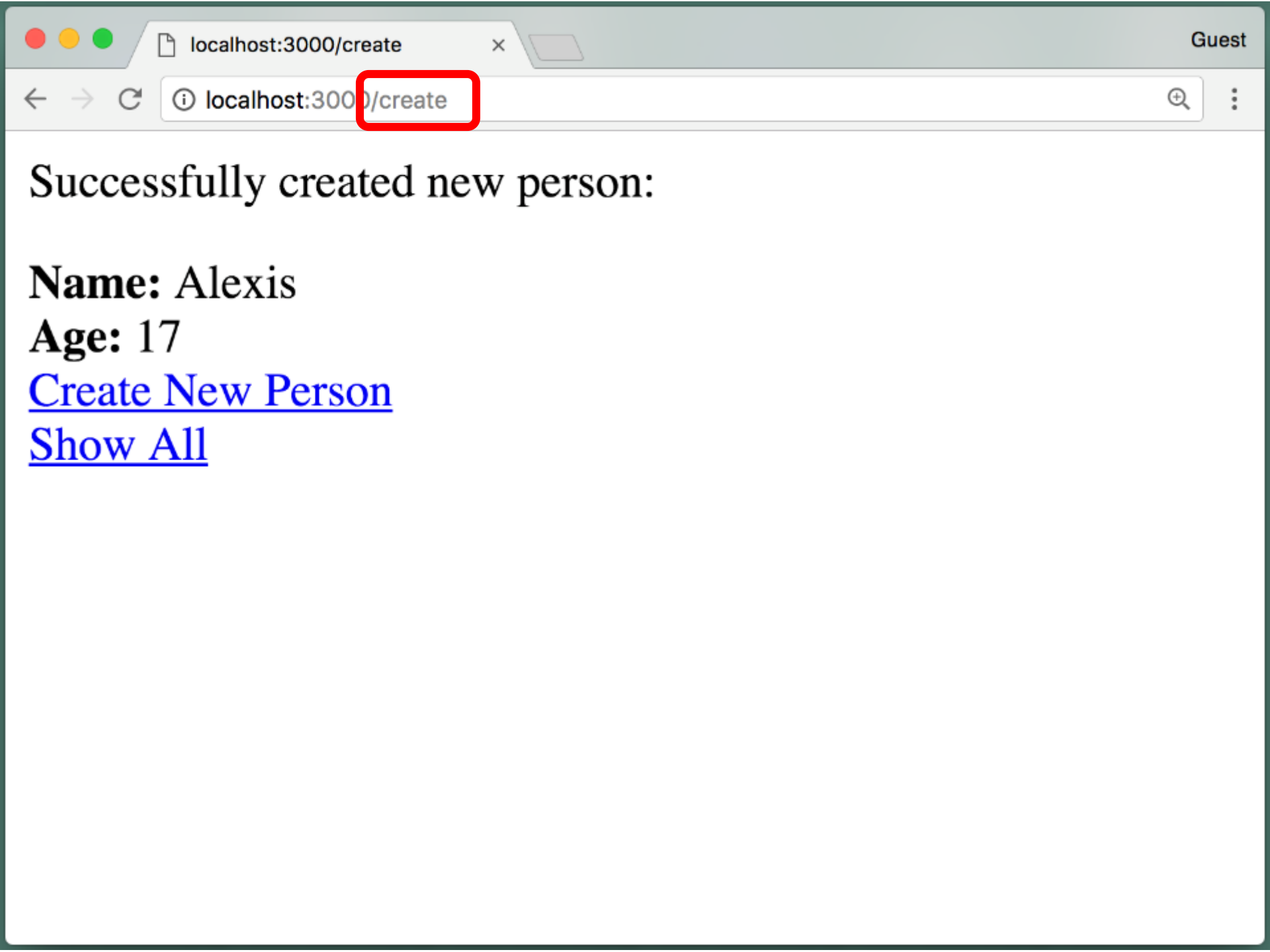
```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```

```
/* This is index.js */

var express = require('express');
var app = express();

app.set('view engine', 'ejs');

var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));

var Person = require('./Person.js');

. . .

// this is at the bottom

app.use('/public', express.static('public'));

app.use('/', (req, res) =>
   { res.redirect('/public/personform.html'); } );

app.listen(3000,  () =>
   { console.log('Listening on port 3000'); } );
```



Successfully created new person:

Name: Alexis
Age: 17
Create New Person
Show All


localhost:3000/create
Guest
localhost:3000/create
Successfully created new person:
Name: Alexis
Age: 17
Create New Person
Show All

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
<html>
<body>

<form action='/create' method='post'>

Name: <input name='name'>
<p>

Age: <input name='age'>
<p>

<input type=submit value='Submit form!'>

</form>

</body>
</html>
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

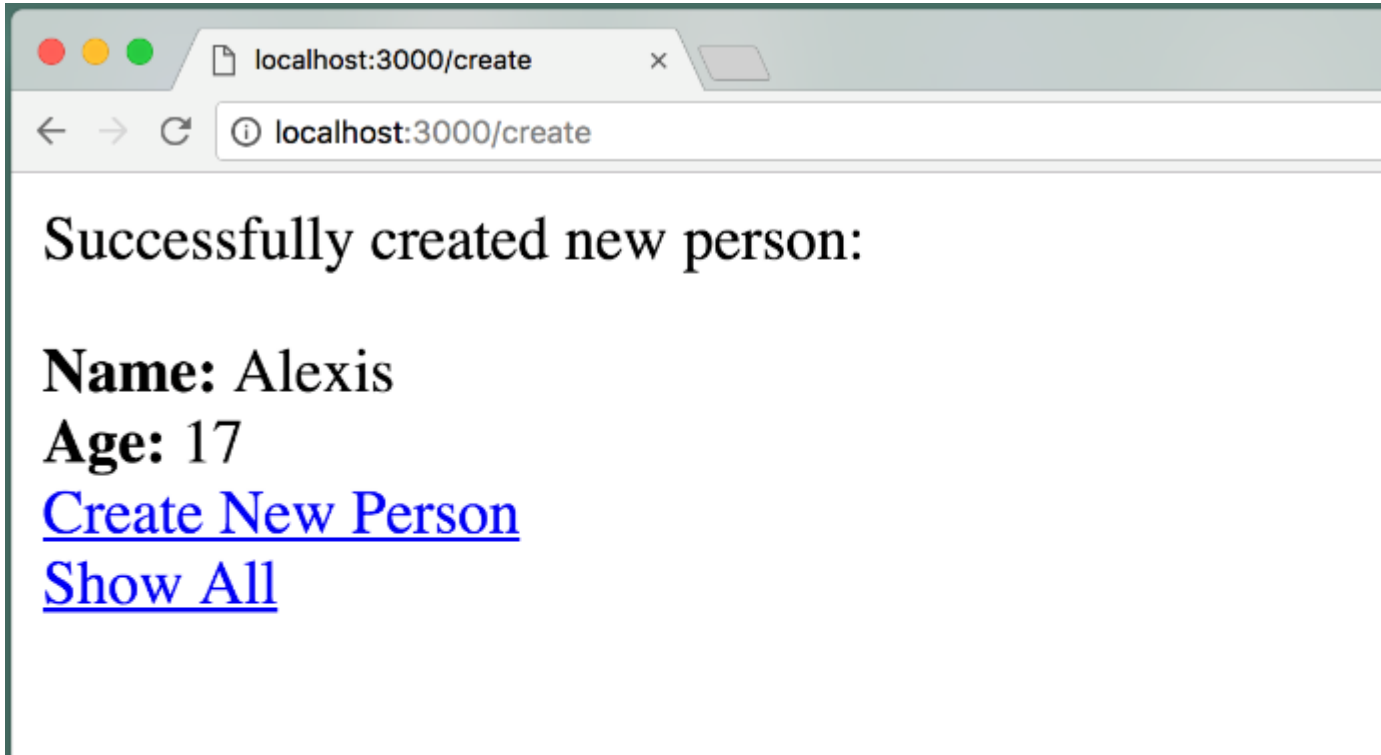
```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

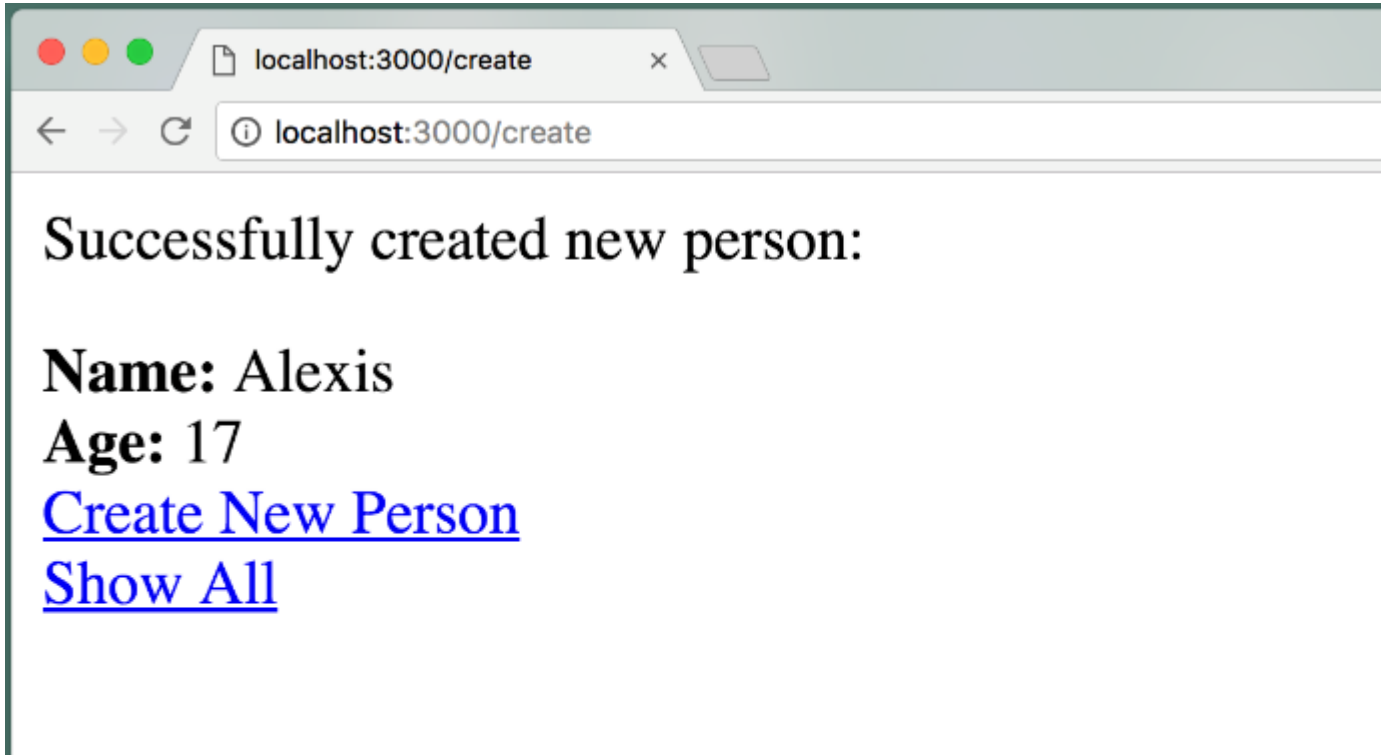
```
app.use('/create', (req, res) => {

   var newPerson = new Person ({ // defined in Person.js
      name: req.body.name,
      age: req.body.age,
   });

   newPerson.save( (err) => {
      if (err) {
         res.type('html').status(500);
         res.send('Error: ' + err);
      }
      else {
         res.render('created', { person: newPerson });
      }
   });
});
```

```
<!-- This is views/created.ejs -->

Successfully created new person:
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

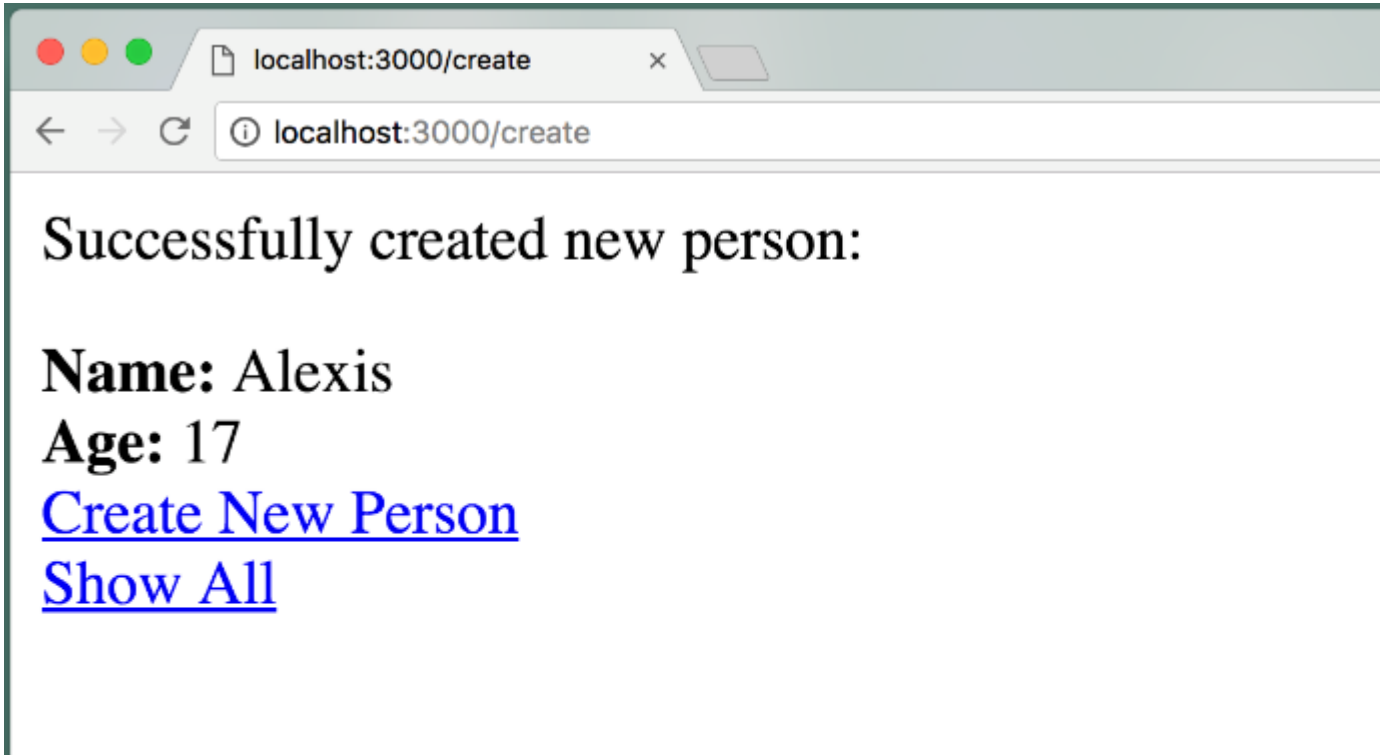


Penn Engineering

```
<!-- This is views/created.ejs -->

Successfully created new person:
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

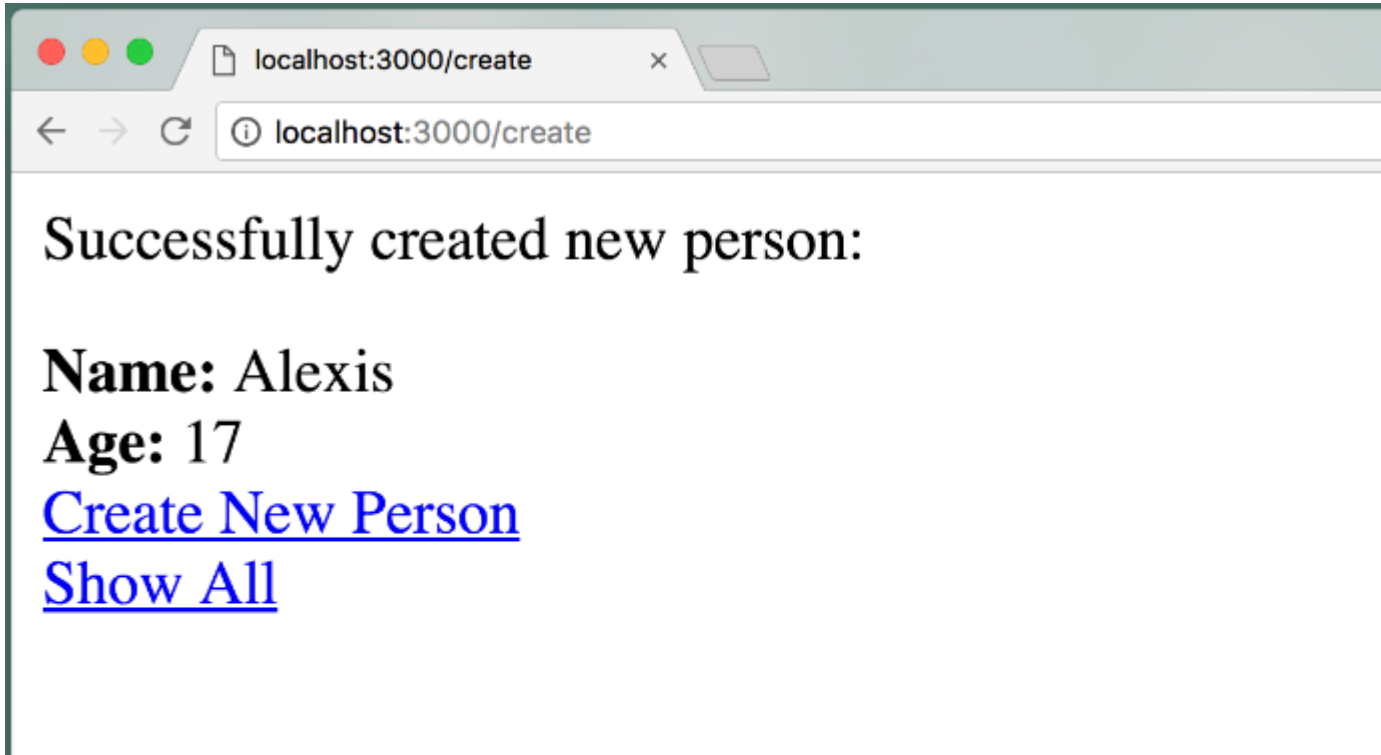


```
<!-- This is views/created.ejs -->

Successfully created new person:
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```



Penn Engineering

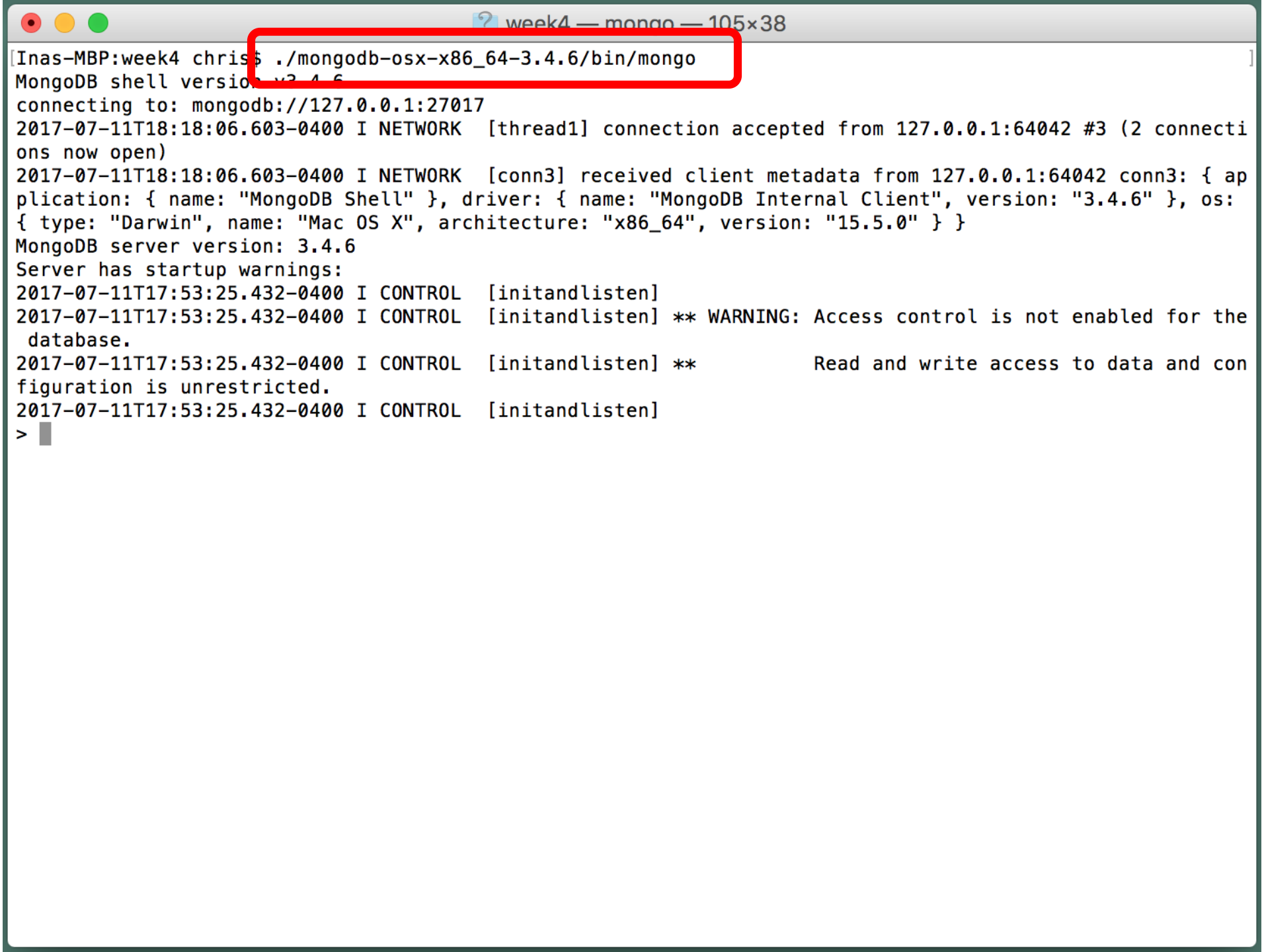
```
<!-- This is views/created.ejs -->

Successfully created new person:
<p>
<b>Name:</b> <%= person.name %>
<br>
<b>Age:</b> <%= person.age %>
<br><a href='/public/personform.html'>Create New Person</a>
<br><a href='/all'>Show All</a>
```

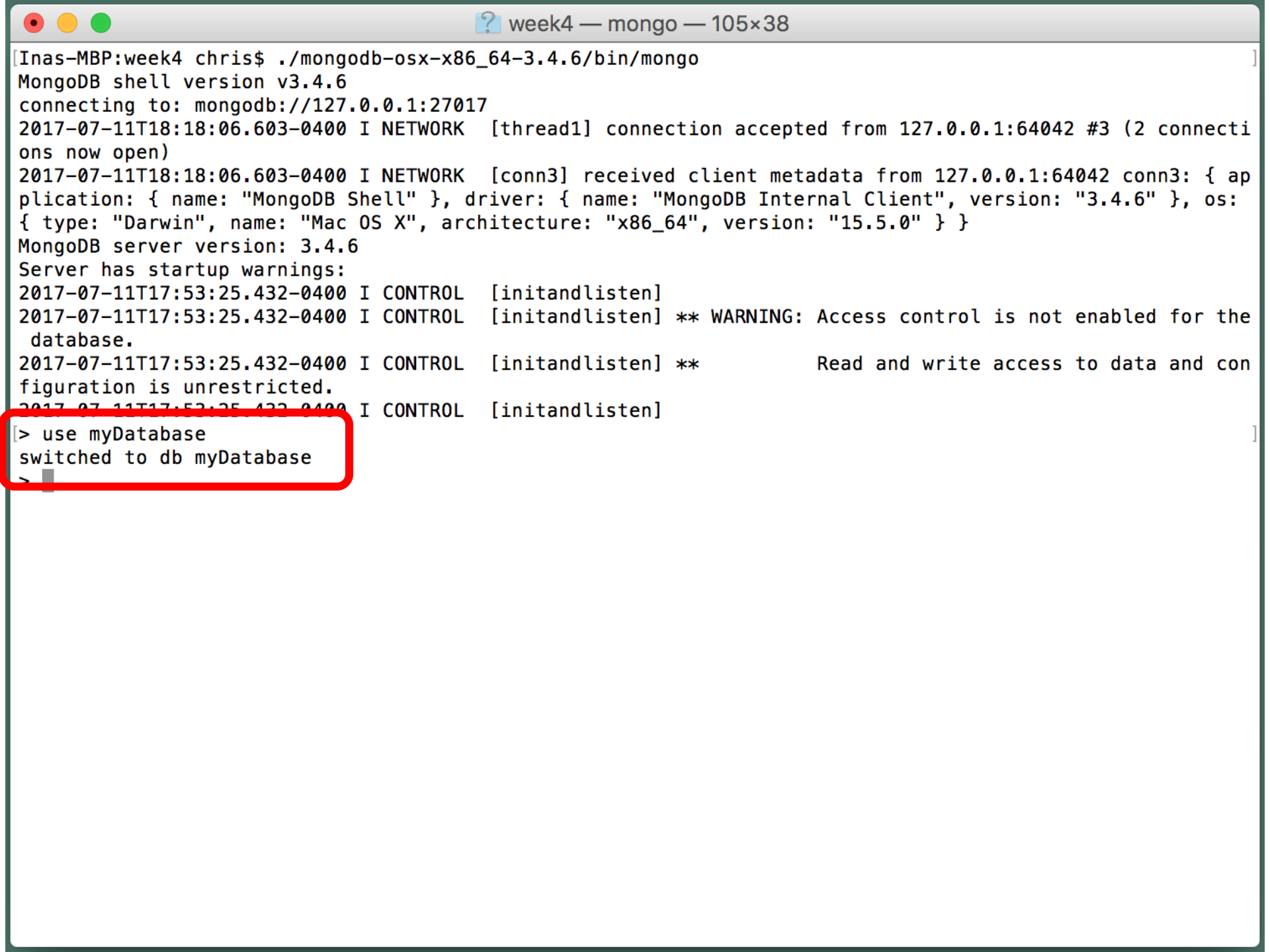


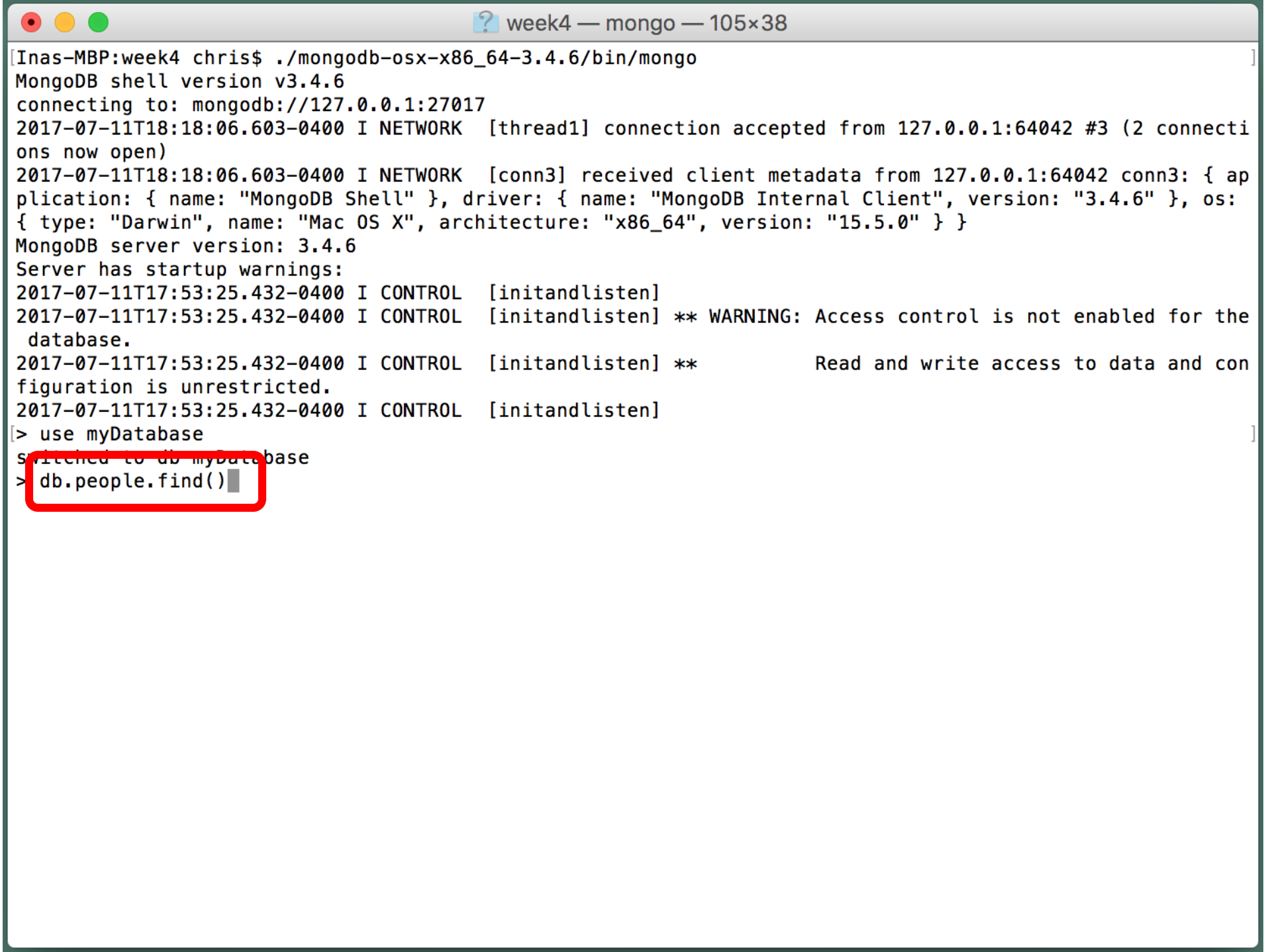
Penn Engineering

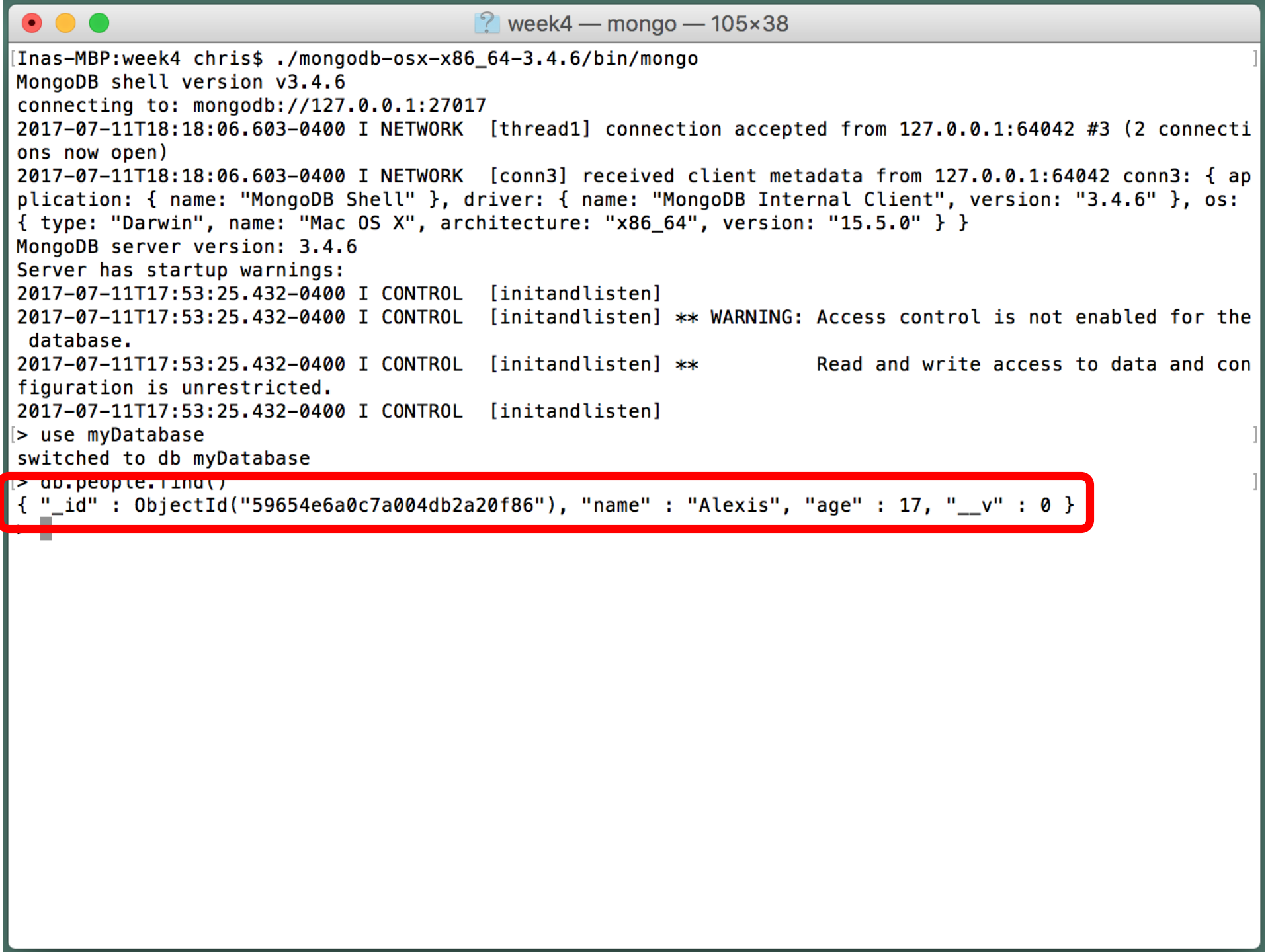
week4 — mongo — 105×38

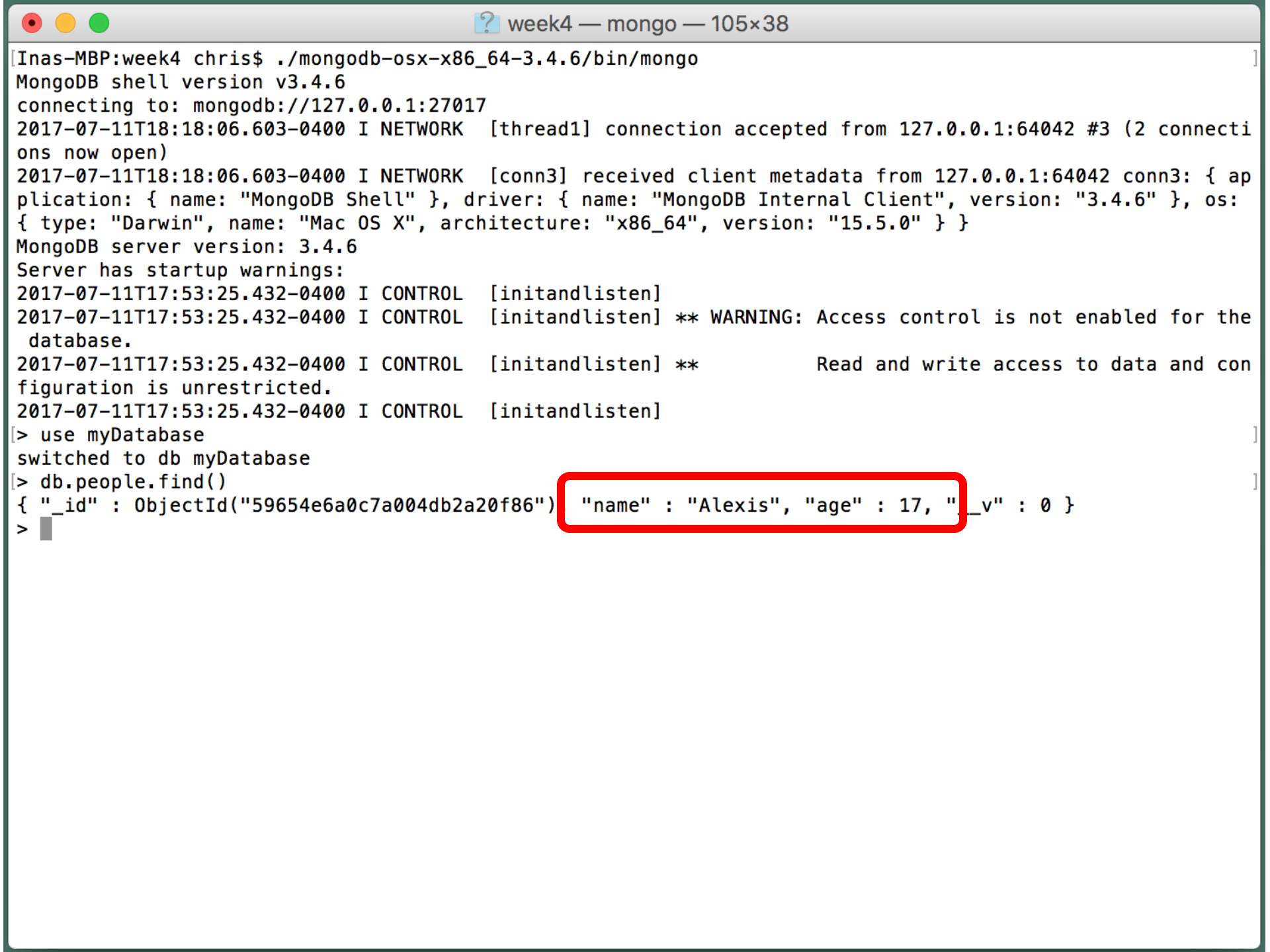
```
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK  [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connecti
ons now open)
2017-07-11T18:18:06.603-0400 I NETWORK  [conn3] received client metadata from 127.0.0.1:64042 conn3: { ap
plication: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os:
{ type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the
 database.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten] **          Read and write access to data and con
figuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL  [initandlisten]
> 
```


week4 — mongo — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
>


week4 — mongo — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
I CONTROL [initandlisten]
> use myDatabase
switched to db myDatabase
>


week4 — mongo — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
> use myDatabase
switched to db myDatabase
> db.people.find()


week4 — mongo — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
> use myDatabase
switched to db myDatabase
> db.people.find()
{ "_id" : ObjectId("59654e6a0c7a004db2a20f86"), "name" : "Alexis", "age" : 17, "__v" : 0 }


week4 — mongo — 105×38
Inas-MBP:week4 chris$ ./mongodb-osx-x86_64-3.4.6/bin/mongo
MongoDB shell version v3.4.6
connecting to: mongodb://127.0.0.1:27017
2017-07-11T18:18:06.603-0400 I NETWORK [thread1] connection accepted from 127.0.0.1:64042 #3 (2 connections now open)
2017-07-11T18:18:06.603-0400 I NETWORK [conn3] received client metadata from 127.0.0.1:64042 conn3: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "3.4.6" }, os: { type: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "15.5.0" } }
MongoDB server version: 3.4.6
Server has startup warnings:
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten] ** Read and write access to data and configuration is unrestricted.
2017-07-11T17:53:25.432-0400 I CONTROL [initandlisten]
> use myDatabase
switched to db myDatabase
> db.people.find()
{ "_id" : ObjectId("59654e6a0c7a004db2a20f86") "name" : "Alexis", "age" : 17, "__v" : 0 }
>

Summary

- **MongoDB** is a NoSQL Database that is designed for use with JavaScript apps
- MongoDB stores **collections** of **documents**
- We can access MongoDB from our Node/Express app using libraries such as Mongoose
- We define a **Schema** and then can create new documents using the **`save`** function