index.js

- ¶

Main init file for webframework.

- ¶

Requires express (light-weight web application framework), bodyParser (parse info from MongoDB), routes (import given enpoints from routes.js file), and http (inits a http webservice)

```
var express     = require('express'),
    bodyParser  = require('body-parser'),
    routes      = require('./routes'),
    http        = require('http');
```

- ¶

Inits an express app onto port 3000

```
    var app = express();
    app.set('port', process.env.PORT || 3000);
    app.set('env', process.env.NODE_ENV || 'development');
```

- ¶

Parses through encoded data

```
    app.use(bodyParser.urlencoded({ extended: false }));
    app.use(bodyParser.json());
```

- ¶

Exports routes.js to utilize our express framework

```
    routes(app);
```

- ¶

Creates our express server on http (localhost) and outputs that it is listening on port 3000

```
    var server = http.createServer(app);

  server.listen(app.get('port'), function(){
    console.log("Listening on port " + app.get('port') + " in "+ app.get('env') + " mode");
  });
```

- ¶

Export this module as app, so mongo.js and routes.js can use this.

```
module.exports = app;
```

mongo.js

- ¶

Main MongoDB init and connections

- ¶

  Requires mongoose for MongoDB data pullling

```
var mongoose    = require('mongoose');
```

- ¶

  dotenv is a node module that stores our credentials

```
require('dotenv').load();

mongoose.connect(process.env.DB);
```

- ¶

  Inits a basic 'any' schema with our database

```
var AnySchema = new mongoose.Schema({ any: mongoose.Schema.Types.Mixed },{ strict: false });
```

- ¶

  Next three functions init the 'any' schema on the three collections we have, which are Rooms, Class-Events, and User

```
function roominit(){
  return mongoose.model('ROOM', AnySchema, 'ROOM');
}

function classeventinit(){
  return mongoose.model('CLASS-EVENT', AnySchema, 'CLASS-EVENT');
}

function userinit(){
  return mongoose.model('USER', AnySchema, 'USER');
}
```

- ¶

  Export db schemas to other files

```
exports.roominit = roominit;
exports.classeventinit = classeventinit;
exports.userinit = userinit;
```

routes.js

- ¶

All Endpoints for hitting the database

- ¶

Requires database inits from mongo.js and all json data files

```
var db        = require('./mongo.js');
var data_room = require('../data_room.json')
var data_class_event = require('../data_class_event.json')
```

- ¶

Imports mongo.js DB schemas to manipulate

```
var Room = db.roominit();
var Class_Event = db.classeventinit();
var User = db.userinit();

module.exports = function(app){
```

- ¶

Returns COUNT of collection room and class-event

```
  app.get('/room/count', function(req,res){
    Room.count(function(err,count){
      res.status(200).send(count.toString())
    });
  });
  app.get('/class/count', function(req,res){
    Class_Event.count(function(err,count){
      res.status(200).send(count.toString())
    });
  });
```

- ¶

GET SINGLE - returns a single room details based on query of RoomID

```
  app.get('/room/get/:id', function(req,res){
    Room.find({RoomID:req.params.id}).find(function(err,doc){
      if(err) res.send(err);
      res.status(200).send(doc);
    });
  });

  app.get('/class/get/:id', function(req,res){
    Class_Event.find({ClassID:req.params.id}).find(function(err,doc){
      if(err) res.send(err);
      res.status(200).send(doc);
    });
  });
```

- ¶

GET ALL - outputs everything in the certain collection

```
  app.get('/room/all', function(req,res){
    Room.find(function(err,doc){
      if(err) res.send(err);
      res.status(200).send(doc);
    });
  });

  app.get('/class/all', function(req,res){
    Class_Event.find(function(err,doc){
      if(err) res.send(err);
      res.status(200).send(doc);
    });
  });

  app.get('/user/all', function(req,res){
    User.find(function(err,doc){
      if(err) res.send(err);
      res.status(200).send(doc);
    });
  });
```

- ¶

POST/INSERT - appends a single new document to a database

```
  app.post('/room/post', function(req,res){
    Room.count(function(err,count){
      Room.create({RoomID:req.query.RoomID,Wing:req.query.Wing,RoomNum:req.query.RoomNum,loc:req.query.loc}, function(err,doc){
        if(err) res.send(err);
        res.status(200).send(doc);
      });
    });
  });

  app.post('/class/post', function(req,res){
    Class_Event.count(function(err,count){
      Class_Event.create({RoomID:req.query.RoomID,ClassID:req.query.ClassID,Dept:req.query.Dept,ClassNum:req.query.ClassNum,Name:req.query.Name}, function(err,doc){
        if(err) res.send(err);
        res.status(200).send(doc);
      });
    });
  });
```

- ¶

DELETE - delete a document in certain database based on the ID given

```
  app.delete('/room/delete/:id', function(req,res){
    Room.remove( {RoomID:req.params.id}, function(err,Room){
      if(err) res.send(err);
      res.status(200).send("removed "+req.params.id);
    });
  });

  app.delete('/class/delete/:id', function(req,res){
    Class_Event.remove( {ClassID:req.params.id}, function(err){
      if(err) res.send(err);
      res.status(200).send("removed "+req.params.id);
    });
  });
```

- ¶

  DROP - Removes everything from a certain collection

```
  app.get('/room/drop', function(req, res) {
    Room.remove({}, function(err){
      if (err) res.send(err);
      res.status(200).send("Database ROOM successfully dropped.");
    });
  });

  app.get('/class/drop', function(req, res) {
    Class_Event.remove({}, function(err){
      if (err) res.send(err);
      res.status(200).send("Database CLASS successfully dropped.");
    });
  });
```

- ¶

  ADD FROM FILE - Reads in the json file data_room.json or data_class_event.json and adds all the info as docs

```
  app.get('/room/add', function(req, res){
    for(var i = 0; i < data_room.length; i++){
      new Room(data_room[i]).save();
    }
    res.status(200).send("Database ROOM successfully added from data_room.json.");
  });

  app.get('/class/add', function(req, res){
    for(var i = 0; i < data_class_event.length; i++){
      new Class_Event(data_class_event[i]).save();
    }
    res.status(200).send("Database CLASS successfully added from data_class_event.json.");
  });
```

- ¶

  RESET - Clear a collection of all of its documents

```
  app.get('/room/reset', function(req, res){
    Room.remove({}, function(err){
      if (err) res.send(err);
       for(var i = 0; i < data_room.length; i++){
          new Room(data_room[i]).save();
        }
      res.status(200).send("Database ROOM successfully reset.");
    });
  });

  app.get('/class/reset', function(req, res){
    Class_Event.remove({}, function(err){
      if (err) res.send(err);
       for(var i = 0; i < data_class_event.length; i++){
          new Class_Event(data_class_event[i]).save();
        }
      res.status(200).send("Database CLASS successfully reset.");
    });
  });


};
```

test.js

- ¶

Testing connections with the MongoLab

- ¶

Requires lots of testing modules for MongoDB

```
var request = require('supertest'),
    chai = require('chai'),
    express = require('express'),
    app     = require('../server/index.js'),
    db      = require('../server/mongo.js'),
    data_room = require('../data_room.json')
    mongoose = require('mongoose');
```

- ¶

Chai module init into variables

```
    var expect = chai.expect,
        should = chai.should();

    if(process.env.NODE_ENV != 'development'){
        console.log('in development mode');
    }

    var Room = mongoose.model('ROOM'); // this is counting as a test, very good
```

- ¶

Tests below

- ¶

Most code is self explanatory

```
    describe('POST single class', function(){
      it('responds with a single class object in JSON', function(done){
        request(app).post('/class/post?RoomID=0001&ClassID=50009&Dept=CSCI&ClassNum=4308&Name=SeniorProjects')
            .end(function(err,res){
                if(err){ return err}
                expect(res).to.exist;
                expect(res.status).to.equal(200);
                expect(res.text).to.contain('"RoomID":"0001"');
                expect(res.text).to.contain('"ClassID":"50009"');
                expect(res.text).to.contain('"Dept":"CSCI"');
                done();
        });
      });
    });

    describe('DELETE single room' , function(){
      it('deletes a single room', function(done){
        request(app).del('/room/delete/0002').expect(200, done);
        request(app)
            .get('/room/get/0002')
            .end(function(err,res){
                if(err){ return err }
                expect(res.text).to.equal("[]")
        });
      });
    });
    describe('DELETE single class' , function(){
      it('deletes a single class', function(done){
        request(app).del('/class/delete/50002').expect(200, done);
        request(app)
            .get('/class/get/50002')
            .end(function(err,res){
                if(err){ return err }
                expect(res.text).to.equal("[]")
        });
      });
    });

    describe('GET all rooms', function(){
      it('responds with a list of room objects in JSON', function(done){
```

```
    request(app)
    .get('/room/all')
    .set('Accept', 'application/json')
    .expect('Content-Type', /json/)
    .expect(200, done);
  });
});

describe('DROP ROOM database', function(){
  it('drops the ROOM database..', function(done){
    request(app).get('/room/drop').expect(200, done);
    request(app).get('/room/count')
        .end(function(err,res){
            if(err){ return err }
            expect(res.text).to.equal("0")
        });
  });
});

describe('RESET ROOM database', function(){
  it('RESETS the ROOM collection..', function(done){
    request(app).get('/room/reset').expect(200);
    request(app).get('/room/count')
        .end(function(err,res){
            if(err){ return err }
            expect(res.text).to.equal("8")
        });
        done();
  });
});

describe('ADD ROOM database', function(){
  it('ADD the ROOM database..', function(done){
    request(app).get('/room/drop').expect(200);
    request(app).get('/room/add').expect(200);
    request(app).get('/room/count')
        .end(function(err,res){
            if(err){ return err }
            expect(res.text).to.equal("8")
        })
    done();
  });
});
```