changemachine

Release 0.4.9

May 24, 2016

Contents

ChangeMachine was written to handle the CouchDB change processing required in Steelmesh. It is responsible for monitoring and responding to changes in a number of couchdb instances and taking appropriate actions in response to those changes.

The implementation of ChangeMachine is reasonably simple thanks to the flatiron neuron queueing library and through leveraging changemate notifiers.

CHAPTER 1

Contents

1.1 Getting Started

ChangeMachine is designed to work in similar fashion to manufacturing plant, whereby a plant is made up of many machines that perform a single function, and items (or materials) enter the machine, get processed and leave the machine.

In most cases, a machine processes an item successfully:

```
machine.on('process', function(item) {
        // do something with the item

        // mark the item as done
        item.done();
});
```

But sometimes things can go wrong:

```
machine.on('process', function(item) {
        try {
                // do something with the item

                // mark the item as done
                item.done();
        }
        catch (e) {
                // mark the item as failed
                item.fail({ error: e });
        }
});
```

Additionally, sometimes you might receive an error when something is done but not want to have to call a separate fail function. You can do this by passing an error in the options for any of the *done*, *fail* or *skip* methods and the status will be remapped to *fail*:

```
machine.on('process', function(item) {
        // write the item data to a file
        writeItemData(item, function(err) {
                item.done({ error: err });
        });
});
```

Regardless of how you flag that an item has failed, you probably want to know about it:

```
machine.on('fail', function(item, err) {
        console.log('got a failed item: ' + item.id);
});
```

1.2 Use Cases

To be completed

1.3 ChangeMachine Examples

This section of the documentation comments on the examples that are included with the ChangeMachine source.

1.3.1 Simple Example using Changemate Notifier

This simple example demonstrates responding to changes from an external couchdb:

```
var cm = require('../'),
    machine = new cm.Machine('<:couch:> http://sidelab.iriscouch.com/seattle_neighbourhood', {
        concurrency: 25 // override neurons default concurrency of 50
    });

machine.on('enter', function(item) {
    console.log('   entered: ' + item.id, machine.stats());
});

// perform actions for each of the
machine.on('process', function(item) {
    console.log('processing: ' + item.id, machine.stats());

    setTimeout(function() {
        item.done();
        console.log('      done: ' + item.id, machine.stats());
    }, Math.random() * 5000);
});
```

If it is all working nicely, you should see output similar to simple.output.txt.

1.3.2 Simple Example demonstrating ready queue

In the example above, items existed in either the `waiting` or `processing` queues. This because the machine had a `process` event that could be used to process the items as they enter the machine. In a case where a `process` event handler is not defined, however, the items ready for processing (according to neuron's concurrency setting) will be placed in the `ready` queue.

Let's modify the previous example to wire up the process event after 5 seconds:

```
var cm = require('../'),
    machine = new cm.Machine('<:couch:> http://sidelab.iriscouch.com/seattle_neighbourhood', {
        concurrency: 25 // override neurons default concurrency of 50
    });

machine.on('enter', function(item) {
```

```
    console.log('   entered: ' + item.id, machine.stats());
});

setTimeout(function() {
    // perform actions for each of the
    machine.on('process', function(item) {
        console.log('processing: ' + item.id, machine.stats());

        setTimeout(function() {
            item.done();
            console.log('      done: ' + item.id, machine.stats());
        }, Math.random() * 5000);
    });
}, 5000);
```

In the output for this example, you should see that before processing starts a number of items are reported in the `ready` queue. Once the `process` event is connected however, the items move directly from a `waiting` status to `processing`.

1.3.3 Checkpointing

At this stage ChangeMachine implements very simple checkpointing storage but it works nicely and event attempts to synchronously persist data when the process *exit* event is detected.

Below is an example that demonstrates how a state store is configured:

```
const cm = require('../');
const path = require('path');
const sourceUrl = '<:couch:> http://fluxant.cloudant.com/seattle_neighbourhood'

// create a new changemachine instance that will read updates from a remote db
const machine = new cm.Machine(sourceUrl, {
  // as updates are processed, we will keep a checkpoint of whether we are up
  // to using the following changemachine storage
  storage: new cm.JsonStore({ filename: path.resolve(__dirname, 'checkpoint.json') })
});

let counter = 0;

// perform actions for each of the
machine.on('process', function(item) {
  console.log('processing item sequence: ' + item.seq);

  counter++;
  item.done();

  // if we have processed 10 items, then stop
  if (counter >= 10) {
    machine.notifier.close();
  }
});
```

1.3.4 Serializing Data from CouchDB

If you wanted to extract all the JSON data from documents in a couch database (not the attachments though - although it could be combined with attachmate to achieve that) the following example is probably of interest:

```
var cm = require('changemachine'),
    fs = require('fs'),
    path = require('path'),
    machine = new cm.Machine('<:couch:> http://sidelab.iriscouch.com/seattle_neighbourhood', {
        include_docs: true,
        concurrency: 10 // override neurons default concurrency of 50
    }),
    dataPath = path.resolve(__dirname, 'data');

// make the data directory
fs.mkdir(dataPath);

// perform actions for each of the
console.log('waiting for change information');
machine.on('process', function(item) {
    try {
        var text = JSON.stringify(item.doc);

        fs.writeFile(path.join(dataPath, item.id + '.json'), text, 'utf8', function(err) {
            if (err) {
                item.fail(err);
            }
            else {
                item.done();
            }

            console.log('wrote ' + item.id + '.json', machine.stats());
        });
    }
    catch (e) {
        console.log('failed writing: ' + item.id, e);
        item.fail(e);
    }
});
```

1.3.5 Non Notifier Change Machines

While ChangeMachine is designed to work in conjuction with changemate, it is possible to create items and process them manually also.

Example to be completed

1.3.6 Machine Chaining

Machines in ChangeMachine are very chain friendly.

Example to be completed

CHAPTER 2

Indices and tables

- genindex
- modindex
- search