

## Práctica 4

# MOM Message Oriented Middleware

24 de abril del 2018

### Descripción de la práctica

El objetivo de esta práctica es ver las características de ActiveMQ, un intermediario de mensajes entre diferentes aplicaciones o sistemas. En concreto, veremos cómo se instala, configura y comprobaremos su funcionamiento mediante la implementación de un caso de uso.

El caso de uso consiste en construir un sistema para control de temperatura e iluminación en un edificio inteligente, mediante el empleo de Arduino y tecnología MOM. Para ello haremos uso de Arduinos (en nuestro caso los simularemos mediante software), para ajustar los parámetros necesarios para controlar la temperatura y los valores de iluminación dentro del edificio inteligente.

Por una parte, tenemos 2 aplicaciones servidoras (las oficinas), para notificar a nuestra aplicación cliente del estado de los sensores y en caso de que nuestra aplicación cliente lo indique accionar los actuadores en los servidores (los cuales serán simulaciones).

Por otra parte, en la parte cliente tendremos una consola, que cada 5 segundos recibirá información de los sensores de las 2 oficinas y, en caso de ser necesario, notificará a los actuadores que se activen.

Tanto las oficinas (aplicaciones servidoras) como la consola (aplicación cliente) estarán implementadas en NodeJS, con ayuda de la librería [stompit](#).

## Requisitos del sistema

- ❑ [Node.js](#) 8.9.4
- ❑ [ActiveMQ](#)

## Archivos de la práctica

Junto a esta memoria, adjunto la aplicación cliente (consola central) llamado '*console.js*' y las dos aplicaciones servidoras (oficinas) llamadas '*office1.js*' y '*office2.js*' respectivamente en NodeJS.

La ubicación del proyecto es indiferente, pero es importante abrir 3 terminales en su ubicación para poder poner cada aplicación en marcha.

## Instalación del software necesario

Para el correcto funcionamiento de la práctica, resulta indiferente el SO empleado.

1. **Instalar NodeJS:** En su [web oficial](#) nos proporcionan una descarga directa y personalizada para nuestro sistema operativo. En la cuál, de las 2 opciones que nos proporcionan para descargar, recomiendo la versión *8.9.4 LTS - Recomendado para la mayoría*.

## Puesta en marcha

### 1. Iniciar ActiveMQ

Desde el directorio donde se ubica activeMQ, accedemos al directorio `/bin` y ejecutamos el comando:

```
./activemq start
```

### 2. Descargar dependencias e iniciar aplicaciones en Node

Para descargar las dependencias, necesarias para iniciar la API Rest en Node, necesitamos abrir una terminal en la ruta donde se halla el proyecto *mtis-p4* adjuntado en la práctica. En él ejecutamos (sólo la primera vez):

```
npm install
```

A continuación, iniciamos la aplicación cliente, *console.js*, con:

```
node console.js
```

En una segunda terminal, iniciamos el servidor de la primera oficina, *office1.js*, con:

```
node office1.js
```

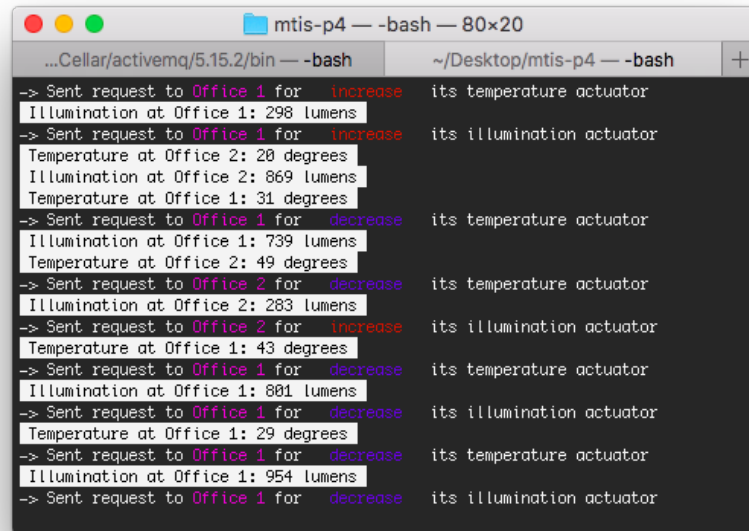
Y en una tercera terminal, iniciamos el servidor de la segunda oficina, *office2.js*, con:

```
node office2.js
```

### 3. Observar log emitido

Finalmente, podemos comprobar el funcionamiento de nuestra práctica desde cada terminal:

Consola Central



```
mtis-p4 — -bash — 80x20
...Cellar/activemq/5.15.2/bin — -bash  ~/Desktop/mtis-p4 — -bash +
-> Sent request to Office 1 for increase its temperature actuator
Illumination at Office 1: 298 lumens
-> Sent request to Office 1 for increase its illumination actuator
Temperature at Office 2: 20 degrees
Illumination at Office 2: 869 lumens
Temperature at Office 1: 31 degrees
-> Sent request to Office 1 for decrease its temperature actuator
Illumination at Office 1: 739 lumens
Temperature at Office 2: 49 degrees
-> Sent request to Office 2 for decrease its temperature actuator
Illumination at Office 2: 283 lumens
-> Sent request to Office 2 for increase its illumination actuator
Temperature at Office 1: 43 degrees
-> Sent request to Office 1 for decrease its temperature actuator
Illumination at Office 1: 801 lumens
-> Sent request to Office 1 for decrease its illumination actuator
Temperature at Office 1: 29 degrees
-> Sent request to Office 1 for decrease its temperature actuator
Illumination at Office 1: 954 lumens
-> Sent request to Office 1 for decrease its illumination actuator
```

## Oficina 1

```
mtis-p4 — -bash — 59x21
# Illumination sensor: 298 lumens
-> Requested action: increase
Temperature actuator: 23 degrees
-> Requested action: increase
Illumination actuator: 450 lumens
# Temperature sensor: 31 degrees
# Illumination sensor: 739 lumens
-> Requested action: decrease
Temperature actuator: 22 degrees
# Temperature sensor: 43 degrees
# Illumination sensor: 801 lumens
-> Requested action: decrease
Temperature actuator: 21 degrees
-> Requested action: decrease
Illumination actuator: 425 lumens
# Temperature sensor: 29 degrees
# Illumination sensor: 954 lumens
-> Requested action: decrease
Temperature actuator: 20 degrees
-> Requested action: decrease
Illumination actuator: 400 lumens
```

## Oficina 2

```
mtis-p4 — -bash — 59x21
Temperature actuator: 17 degrees
-> Requested action: increase
Illumination actuator: 575 lumens
# Temperature sensor: 8 degrees
# Illumination sensor: 391 lumens
-> Requested action: increase
Temperature actuator: 18 degrees
-> Requested action: increase
Illumination actuator: 600 lumens
# Temperature sensor: 45 degrees
# Illumination sensor: 965 lumens
-> Requested action: decrease
Temperature actuator: 17 degrees
# Temperature sensor: 20 degrees
# Illumination sensor: 869 lumens
# Temperature sensor: 49 degrees
# Illumination sensor: 283 lumens
-> Requested action: decrease
Temperature actuator: 16 degrees
-> Requested action: increase
Illumination actuator: 625 lumens
```

## Pasos seguidos para la elaboración de la práctica

Primero, cree la aplicación cliente usando la librería [STOMPIT](#) para NodeJS. Para ello instale la dependencia:

```
npm install stompit
```

Posteriormente, consumí dicha dependencia importándola y asignando las opciones de conexión con ActiveMQ:

```
var stompit = require('stompit');
var readline = require('readline-sync');
var yesno = require('yesno');

let connectOptions = {
  'host': 'localhost',
  'port': 61613,
  'connectHeaders': {
    'host': '/',
    'login': '', //'arancha',
    'passcode': '', //'12345',
    'heart-beat': '5000,5000'
  }
};
```

Realizamos la conexión:

```
stompit.connect(connectOptions, function(error, client) {

  if (error) {
    console.log('\x1b[31m', 'Connect error: ' + error.message, '\x1b[0m');
    return false;
  } else {
    console.log('\x1b[32m', 'Central console: Connected successfully to ActiveMQ', '\x1b[0m')
  }
}
```

Si no se producen errores, nos suscribimos al sensor de temperatura de la oficina 1 y procesamos sus mensajes:

```
function readOffice1Temperature(client) {

  var subscribeHeaders = {
    'destination': '/mtis_p4/office1/temperature/sensor',
    'ack': 'client-individual'
  };

  client.subscribe(subscribeHeaders, function(error, message) {

    if (error) {
      console.log('Subscribe error: ' + error.message);
      return;
    }

    message.readString('utf-8', function(error, body) {

      if (error) {
        console.log('Read message error: ' + error.message);
        return;
      }

      let sensorValue = parseInt(body, 10)
      console.log('\x1b[7m', 'Temperature at Office 1: ' + sensorValue + ' degrees \x1b[0m');
```

Por otro lado, para la aplicación servidora de la oficina 1, el proceso de conexión a ActiveMQ es el mismo. En su caso, tengo configurado el envío de la información de los sensores cada 5 segundos:

```
setInterval( () => {
  sendTemperature(client)
  sendIllumination(client)
}, 5000 )
```

Para enviar datos, configuramos el endpoint definido, `client.send( sendHeaders )`, y con `frame.write(...)` escribimos el mensaje a enviar:

```
function sendTemperature(client) {
  var sendHeaders = {
    'destination': '/mtis_p4/office1/temperature/sensor',
    'content-type': 'text/plain'
  };

  let newTemp = Math.floor(Math.random() * 50) + 0;

  //console.log('hace ' + newTemp + ' grados de caloret')
  console.log('\x1b[7m # Temperature sensor: ' + newTemp + ' degrees \x1b[0m')
  var frame = client.send(sendHeaders);
  frame.write(newTemp.toString());
  frame.end();
}
```

El proceso de la oficina 1 para subscribirse a las acciones enviadas por la consola central es similar al seguido por la consola central:

```
function listenTemperatureActuator(client) {

  var subscribeHeaders = {
    'destination': '/mtis_p4/office1/temperature/actuator',
    'ack': 'client-individual'
  };

  client.subscribe(subscribeHeaders, function(error, message) {

    if (error) {
      console.log('Subscribe error: ' + error.message);
      return;
    }

    message.readString('utf-8', function(error, body) {

      if (error) {
        console.log('Read message error: ' + error.message);
        return;
      }
    })
  })
}
```