**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**…………….🙚O🙘…………….**

Cần Thơ, ngày tháng 04 năm 2015

Giảng viên hướng dẫn

ThS. Lâm Chí Nguyện

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

**…………….🙚O🙘…………….**

Cần Thơ, ngày tháng 04 năm 2015

Giảng viên phản biện

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

**…………….🙚O🙘…………….**

Cần Thơ, ngày tháng năm 2015

Giảng viên phản biện

# LỜI CẢM ƠN

**…………….🙚O🙘…………….**

Trong suốt quá trình thực hiện đề tài luận văn tốt nghiệp “Xây dựng hệ thống quản lý, điều khiển các dịch vụ mạng bằng giao diện web” em luôn nhận được sự giúp đỡ từ giảng viên, bạn bè và các anh chị để em có thể thực hiện tốt đề tài của mình.

Trước tiên, em xin gửi lời cảm ơn chân thành đến thầy Lâm Chí Nguyện đã tận tình hướng dẫn và chỉ ra con đường tốt để em có thể được làm luận văn đồng thời học hỏi và rèn luyện thêm nhiều kinh nghiệm.

Trên hết em xin gửi lời cảm ơn sâu sắc tới quý thầy cô trường Đại học Cần Thơ nói chung và quý thầy cô khoa Công nghệ thông tin và Truyền thông nói riêng, quý thầy cô đã truyền cho em những kiến thức, kinh nghiệm quý báu trong suốt 4 năm qua để em có đủ hành trang và nghị lực bước vào đời và có thể trở thành người có ích cho xã hội.

Kế đến là xin cảm ơn Cộng đồng ZendVN, Study And Share (Các anh chị đã đi làm) đã tận tình hướng dẫn cho em hiểu biết về công nghệ mới, cách áp dụng vào đề tài.

Mặc dù đã cố gắng hoàn thành luận văn trong phạm vi và khả năng nhưng chắc chắn sẽ không tránh khỏi những thiếu sót và cũng như chưa thể hoàn thiện 100% yêu cầu đề tài đặt ra trong khoảng thời gian ngắn. Em kính mong nhận được sự thông cảm và tận tình chỉ bảo của quý thầy cô và các bạn, hi vọng đề tài của em sẽ được phát triển và hoàn thiện để đưa vào sử dụng.

Cần Thơ, ngày tháng năm 2015

Sinh viên thực hiện

# LỜI NÓI ĐẦU

**…………….🙚O🙘…………….**

# MỤC LỤC

**…………….🙚O🙘…………….**

# KÍ HIỆU VÀ VIẾT TẮT

**…………….🙚O🙘…………….**

|  |  |
| --- | --- |
| **Từ/kí hiệu viết tắt** | **Nguyên bản** |
| API | Application Programming Interface |
| JSON | JavaScript Object Noattion |
| SSH | Secure Shell |
| CLI | Command Line Interface |
| POM | Project Object Model |
| JMX | Java Management Extensions |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# DANH MỤC HÌNH VẼ

**…………….🙚O🙘…………….**

# ABSTRACT

**…………….🙚O🙘…………….**

# TÓM TẮT

**…………….🙚O🙘…………….**

Hiện nay, với giá thành tương đối rẻ, và sự tiện dụng cùng với sự ra đời của Cloud, thì rất dễ dàng để có thể sử dụng một dịch vụ VPS, Cloud VPS, hoặc Server (sau đây xin gọi chung là “Server”) với giá chỉ từ 100.000đ một tháng với Cloud VPS. Bên cạnh đó vấn đề quản lý tập trung các Server ở nhiều nhà cung cấp và đối với người không chuyên cũng trở nên khó khăn cấp thiết. Để giải quyết phần nào khó khăn đó chúng tôi đã thực hiện đề tài “Xây dựng hệ thống quản lý, điều khiển các dịch vụ mạng bằng giao diện WEB”. Mục đích của đề tài là xây dựng một công cụ quản trị tập trung các dịch vụ mạng trên Server từ xa, hỗ trợ người quản trị có thể quản lý các Server của mình một cách hiệu quả và nhanh chóng.

# PHẦN 1: GIỚI THIỆU

### ĐẶT VẤN ĐỀ

Với một người quản trị, hoặc một webmaster sử dụng nhiều dịch vụ Server khác nhau tại nhiều nhà cung cấp khác nhau thì việc cấu hình, quan sát đều phải thực hiện trên mỗi Server được cài đặt và nếu có nhiều Server cài đặt nhiều dịch vụ thì việc quản lý sẽ trở nên khó khăn. Chắc chắn là sẽ gây ra lãng phí về công sức và thời gian để thực hiện tốt công việc đó nếu như không có một giải pháp hỗ trợ. Hiện tại cũng có không ít phần mềm, dịch vụ có thể điều khiển cài đặt các dịch vụ vào Server, nhưng đa phần phải cài đặt vào trong chính Server đó hoặc là hệ thống quản lý của mỗi nhà cung cấp riêng biệt, hay các ứng dụng chạy trên máy tính, sử dụng bằng dòng lệnh phức tạp, chưa thích hợp cho người dùng không chuyên. Để người quản lý có thể nhanh chóng cấu hình, cài đặt dịch vụ trên Server mọi lúc, mọi nơi và thích hợp cho người dùng không chuyên thì một dịch vụ web quản lý tập trung là cần thiết.

Sẽ thật sự rất khó khăn và tốn một thời gian khá lâu để cài đặt hoặc cấu hình nhiều dịch vụ trên các Server nếu không chuyên hoặc không có một hệ thống trợ giúp. Vì vậy, một hệ hống quản lý tập trung các Server, cho phép cấu hình cơ bản các dịch vụ cần thiết bằng các thao tác đơn giản. Hơn nữa còn có khả năng đáp ứng mọi lúc, mọi nơi là thật sự cần thiết.

Hiện tại cũng có một số sản phẩm thương mại lẫn miến phí có hỗ trợ quản lý Server. Tuy nhiên, đa phần là các ứng dụng riêng lẽ chạy trên máy tính hoặc do chính nhà cung cấp dịch vụ đó hổ trợ, ngoài ra nhằm để hiểu được phương thức hoạt động, các công nghệ hiện hữu, khai thác và tỉm hiểu công nghệ nên cần xây dựng hệ thống dựa trên các công nghệ đó.

### LỊCH SỬ GIẢI QUYẾT

Hiện nay đã có nhiều công cụ hỗ trợ quản trị Server từ xa bằng dòng lệnh, hoặc quản trị chuyên sâu được cài đặt trên chính Server đó, nhưng ứng dụng Cấu hình và cài đặt dịch vụ từ xa trền nền web thì rất ít.

Trên thế giới có dịch vụ ServerPilot.io thành lập bởi Justin Samuel vào năm 2012 cung cấp chức năng tương tự, nhưng giá mỗi tháng sử dụng các dịch vụ cơ bản là 10$ hoặc cấu hình các dịch vụ nâng cao hơn với gia 49$.

Ngoài ra các công cụ nổi tiếng như cPanel (cpanel.com), Directadmin (directadmin.com), Webmin (webmin.com),… là các công cụ quản trị, cấu hình chuyên nghiệp từ trả phí đến miễn phí, nhưng đòi hỏi phải cài đặt vào mỗi server, và phải mua license sử dụng cho từng server khác nhau nếu là phần mềm trả phí.

Ở Khoa Công nghệ thông tin và truyền thông – Trường đại học Cần Thơ, có luận văn “Thiết kế hệ thống tích hợp hiển thị thông tin, quản lý thiết bị mạng và dịch vụ mạng” – Nguyễn Thành Nhân, Nguyễn Hữu Nhật - 2012 đã cho ra ứng dụng chức năng như vậy nhưng chỉ dùng trên máy tính cá nhân, và còn hạn chế là phải sử dụng tài khoản root để quản trị dịch vụ trên Server.

Các sản phẩm, đề tài trên là những ứng dụng Desktop chạy trên máy tính Windows, không thể truy cập mọi lúc mọi nơi trên nhiều máy và nhiều hệ điều hành khác nhau, và các ứng dụng chỉ chạy trên chính Server được cài đặt,không thể cùng quản lý đồng thời nhiều máy chủ. Vì vậy, với đề tài này sẽ cho phép tích hợp các chức năng quản trị từ xa, quản lý tập trung nhiều Server và chạy trên nền web để có thể truy cập mọi lúc, mọi nơi và đa dạng hệ điều hành, thiết bị khác nhau vì chạy trên nền Web.

### MỤC TIÊU ĐỀ TÀI

**Mục đích tổng quát**: Tìm hiểu và vận dụng các công nghệ để xây dựng hệ thống điều khiển, cấu hình và theo dõi Server từ xa qua giao thức SSH.

Việc xây dựng hệ thống được tiến hành theo cách chia Module bao gồm 2 thành phần chính: Module giao diện hệ thống web với nhiệm vụ quản lý người dùng quản lý thông tin Server và kết nối các thành phần, Module cung cấp thư viện để thao tác cài đặt và theo dõi các dịch vụ trên Server.

Dựa trên phân công, module được đảm trách trong báo cáo này là “Hệ thống giao diện quản lý thông tin và thao tác với server”

**Mục tiêu cụ thể trong Module này**:

* Xây dựng hệ thống để triển khai các module cấu hình dịch vụ.
* Cho phép nhiều người dùng sử dụng hệ thống.
* Cung cấp cho người dùng quản lý thông tin Server.
* Hổ trợ quan sát thông tin cơ bản của Server đang hoạt động.
* Hỗ trợ người dùng cài đặt và cấu hình dịch vụ trên giao diện.

### ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu trong luận văn này là công nghệ lập trình web với JSP, Servlet, Hệ quản trị MongoDB, Bootstrap Framework. Xây dựng hệ thống có các chức năng cơ bản sau:

* Thiết kế hệ thống trên nền JSP
* Quản lý người dùng.
* Quản lý các chức năng điều khiển: Network Interfaces, SSH
* Quản lý cài đặt dịch vụ (module chỉ giới hạn ở 2 dịch vụ): DHCP, FTP

### NỘI DUNG NGHIÊN CỨU

* Nghiên cứu cấu hình JSP, spring MVC.
* Nghiên cứu và thiết kế cấu trúc Template với Spring MVC framework.
* Nghiên cứu cấu hình và thao tác với MongoDB.
* Nghiên cứu tạo ssh connection bằng java thông qua jsch
* Nghiên cứu và sử dụng GIT vào việc quản lý mã nguồn.

### BỐ CỤC LUẬN VĂN

Bố cục luận văn được trình bày gồm 2 phần:

**Phần 1: Nội dung**

* Chương 1: Mô tả bài toán
* Chương 2: Thiết kế và cài đặt giải pháp
* Chương 3: Kiểm thử và đánh giá

**Phần 2: Kết luận**

* Kết quả đạt được
* Hướng phát triển

Tài liệu tham khảo

Phụ lục: Hướng dẫn cài đặt, sử dụng chương trình/giải pháp

# PHẦN 2: NỘI DUNG

## 2.5 CÁC CÔNG CỤ HỖ TRỢ KHÁC

### Json converter (org.json)

#### Khái niệm

JSON (JavaScript Object Notation) là 1 định dạng hoán vị dữ liệu nhanh. Chúng dễ dàng cho chúng ta đọc và viết. Dễ dàng cho thiết bị phân tích và tạo ra. Chúng là cơ sở dựa trên tập hợp của Ngôn Ngữ Lập Trình JavaScript. JSON là 1 định dạng kiểu text mà hoàn toàn độc lập với các ngôn ngữ hoàn chỉnh, thuộc họ hàng với các ngôn ngữ họ hàng C, gồm có C, C++, C#, Java, JavaScript, Perl, Python, và nhiều ngôn ngữ khác.

#### Cấu trúc

JSON được xây dựng trên 2 cấu trúc:

* Là tập hợp của các cặp tên và giá trị name - value. Trong những ngôn ngữ khác nhau, đây được nhận thấy như là 1 đối tượng (object), bản ghi (record), cấu trúc (struct), từ điển (dictionary), bảng băm (hash table), danh sách khoá (keyed list), hay mảng liên hợp.
* Là 1 tập hợp các giá trị đã được sắp xếp. Trong hầu hết các ngôn ngữ, nó được nhận thấy như là 1 mảng, véctơ, tập hợp hay là 1 dãy sequence.

Trong JSON, chúng có những thứ trên các định dạng:

* Một đối tượng là 1 hổn độn của các cặp tên và giá trị. Một đối tượng bắt đầu bởi dấu ngoặc đơn trái { và kết thúc với dấu ngoặc đơn phải }. Từng tên được theo sao bởi dấu 2 chấm : và các cặp tên/giá trị được tách ra bởi dấu phẩy ,.

*{"nextHop":"192.168.50.100","destIp":"192.168.50.10","destMask":"255.255.255.0"}*

* Một mảng là 1 tập hợp các giá trị đã được sắp xếp. 1 mảng bắt đầu bởi dấu mở ngoặc vuông [ và kết thúc với dấu ngoặc vuông phải ]. Các giá trị được cách nhau bởi dấu phẩy ,. Một mảng là 1 tập hợp các giá trị đã được sắp xếp. Một mảng bắt đầu bởi dấu mở ngoặc vuông [ và kết thúc với dấu ngoặc vuông phải ]. Các giá trị được cách nhau bởi dấu phẩy ,.

*[*

*{"nextHop":"192.168.50.100","destIp":"192.168.50.10","destMask":"255.255.255.0"}, {"nextHop":"192.168.60.130","destIp":"192.168.60.70","destMask":"255.255.255.0"}, {"nextHop":"192.168.100.10","destIp":"192.168.100.150","destMask":"255.255.255.0"}]*

#### Cách sử dụng org.json framework

Với org.json framework và java, việc chuyển đổi qua lại giữa một object và chuỗi định dạng json là đơn giản, dưới đây là 4 dạng cần thiết cơ bản khi chuyển đổi:

* Một object sang chuỗi định dạng json:

**public** **static** String parseToString(IpRoute ipRoute) {

JSONObject jsonObject = **new** JSONObject(ipRoute);

**return** jsonObject.toString();

}

* Một danh sách object sang chuỗi định dạng json:

**public** **static** String parseToString(List<IpRoute> ipRoutes) {

JSONArray jsonArray = **new** JSONArray();

**for** (IpRoute ipRoute : ipRoutes) {

JSONObject jsonObject = **new** JSONObject(ipRoute);

jsonArray.put(jsonObject);

}

**return** jsonArray.toString();

}

* Một chuỗi định dạng json dang object:

**public** **static** IpRoute parseToIpRoute(String str) {

**try** {

JSONObject jsonObject = **new** JSONObject(str);

String destIp = jsonObject.getString("destIp");

String destMask = jsonObject.getString("destMask");

String nextHop = jsonObject.getString("nextHop");

**return** **new** IpRoute(destIp, destMask, nextHop);

} **catch** (JSONException e) {

System.***out***.println(e.getMessage());

}

**return** **null**;

}

* Một chuỗi định dạng json sang danh sách object:

**public** **static** List<IpRoute> parseToIpRoutes(String str) {

List<IpRoute> ipRoutes = **new** ArrayList<IpRoute>();

**try** {

JSONArray jsonArray = **new** JSONArray(str);

**for** (**int** i = 0; i < jsonArray.length(); i++) {

JSONObject jsonObject = jsonArray.getJSONObject(i);

IpRoute ipRoute = *parseToIpRoute*(jsonObject.toString());

ipRoutes.add(ipRoute);

}

**return** ipRoutes;

} **catch** (JSONException e) {

System.***out***.println(e.getMessage());

}

**return** **null**;

}

### Java Wrapper service

Tham khảo phụ lục A – Cài đặt Windows service với wrapper.

### JSCh framework

#### Giới thiệu:

JSCH(Java Secure Channel) là một framework hỗ trợ thực hiện các kết nối thông qua SSH đến thiết bị bằng ngôn ngữ Java. JSCH hiện tại hỗ trợ phiên bản SSH v2.

#### Sử dụng:

Để sử dụng jsch framework, cần thêm dependency trong pom.xml, tùy phiên bản hoặc vào trang chủ maven repository để lấy phiên bản mới nhất:

<!-- Java Secure Chanel -->

<dependency>

<groupId>com.jcraft</groupId>

<artifactId>jsch</artifactId>

<version>0.1.50</version>

</dependency>

Sau khi thêm thư viện, ta có thể sử dụng 1 số hàm cơ bản của jsch để tạo kết nối và giữ session cũng như thực thi một số lệnh cơ bản trên thiết bị có hỗ trợ. Sau đây là 1 class chủ đạo trong việc thực thi Command Line Interface (CLI) trong hệ thống.

**public** **abstract** **class** AbstractCLI {

**protected** **static** **final** Integer ***DEFAULT\_PORT*** = 22;

**protected** **static** **final** String ***DEFAULT\_USER*** = "nhoangdieu";

**protected** **static** **final** String ***DEFAULT\_PASS*** = "thesis";

**protected** **static** **final** String ***DEFAULT\_CHANNEL*** = "shell";

**protected** **static** **final** Integer ***DEFAULT\_TIMEOUT*** = 10000;

**protected** **static** **final** Integer ***DEFAULT\_SLEEP\_COMMAND*** = 200;

**protected** Session session;

**protected** Channel channel;

**protected** PrintStream ps;

**public** AbstractCLI(String host) **throws** JSchException, IOException {

JSch jsch = **new** JSch();

session = jsch.getSession(***DEFAULT\_USER***, host, ***DEFAULT\_PORT***);

session.setPassword(***DEFAULT\_PASS***);

UserInfo userInfo = **new** MyUserInfo();

session.setUserInfo(userInfo);

session.connect(***DEFAULT\_TIMEOUT***);

channel = session.openChannel(***DEFAULT\_CHANNEL***);

channel.connect(***DEFAULT\_TIMEOUT***);

ps = **new** PrintStream(channel.getOutputStream());

}

**public** **void** execCommand(String command) {

ps.print(command + "\n");

ps.flush();

sleepCommand();

}

**public** **void** close() {

channel.disconnect();

session.disconnect();

}

**public** **void** sleepCommand() {

**try** {

Thread.*sleep*(***DEFAULT\_SLEEP\_COMMAND***);

} **catch** (InterruptedException e) {

System.***out***.println(e.getMessage());

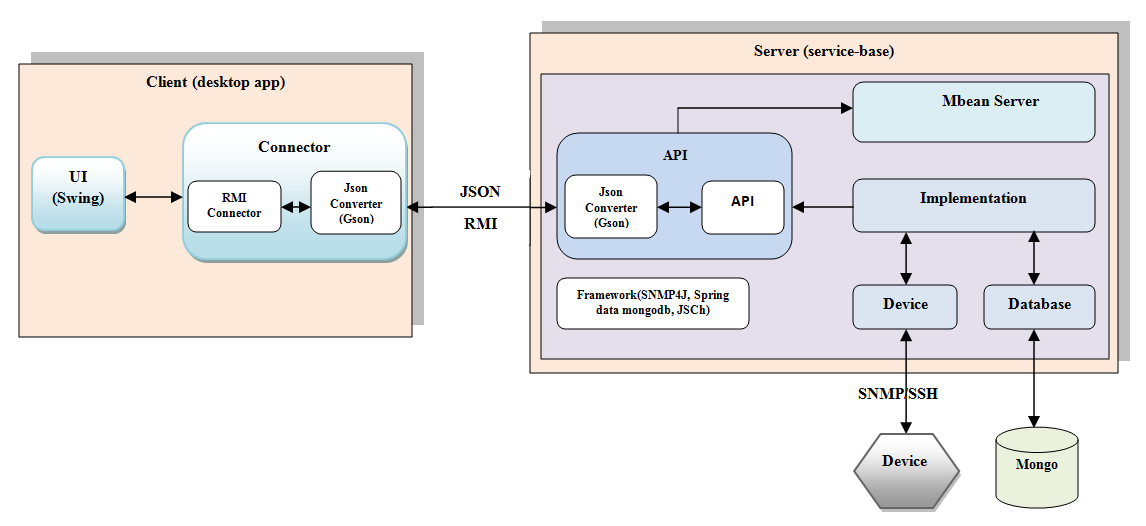
}

}

}

Lớp này tạo các kết nối đến router và cung cấp 1 số hàm cơ bản để đơn giản hóa việc gõ lệnh trên thiết bị.

# CHƯƠNG 3: SƠ ĐỒ CHỨC NĂNG



Hình 3. 1 - Mô hình tổng quát hệ thống

Hình 3.1 là mô hình tổng quát về hệ thống, hệ thống được chia thành 2 thành phần: client và server.

**Server:**

Server được thiết kế theo dạng service-base, sản phẩm là 2 service chạy trên nền Windows (có thể chạy trên Linux). Một service để lưu trữ cơ sở dữ liệu (có tên T-NightMongoDB) và một server để phụ vụ MbeanServer (T-NightService).

* T-NightMongoDB: Là service chạy cơ sở dữ liệu mongodb. Lưu trữ những thông tin cần thiết phía server.
* T-NightService: Tạo và đăng ký các hàm gọi từ xa cho phép thao tác xuống các cấu hình cơ bản trên router (Cisco). Server sử dụng 1 số framework chủ yếu như: snmp4j (thao tác gởi gói tin snmp xuống thiết bị), jsch (thực hiện các câu lệnh thông qua ssh), spring data mongodb (thao tác xuống mongodb) và cốt lỗi là spring/spring jmxrmi (tạo các hàm và đăng ký port với hệ điều hành, lắng nghe các kết nối và thực thi các lời gọi hàm từ xa trả kết quả cho client). Phía server sử dụng wrapper để đăng ký service (tham khảo phụ lục A). Server chỉ cung cấp các api cho phía client thực thi mà không quan tâm đến mọi vấn đề khác, vì thế ở server không quan tâm nhiều đến sơ đồ giải thuật.

**Client:**

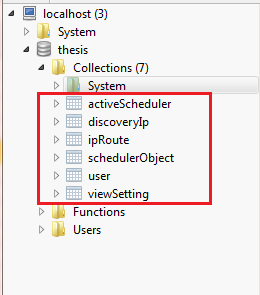
Client được thiết kế theo dạng desktop application, sử dụng java swing là thiết kế chính cho giao diện. Client chỉ sử dụng 2 framework chính là bộ chuyển đổi định dạng json và phần code chung của server cung cấp (được hỗ trợ bởi maven). Sử dụng lớp trong javax tạo kết nối, gọi hàm trên server và nhận kết quả trả về.

Dự liệu truyền giữa client và server với cấu trúc phức tạp được chuyển đổi về định dạng json để dể trong việc kiểm tra lỗi và cũng dể dàng trong việc phát triển ứng dụng lên dạng web-base theo chuẩn REST-API.

## SERVER

### Database

Database sử dụng mongodb nên phần dữ liệu lưu trữ cũng không có cấu trúc, chỉ lưu những thông tin cần thiết như: user, ip route, discovery ip, view setting… Hình 3.6 là toàn bộ documents (table) của hệ thống dưới sự hiển thị của mongodb tool (Robomongo).



Hình 3. 6 - Cấu trúc cơ sở dữ liệu server

Chức năng mỗi document trong hệ thống:

**User**: lưu trữ thông tin cơ bản của một người dùng, bao gồm username, password (được mã hóa dưới dạng md5) và phần tùy chọn có phải admin hay không.

*{*

*"\_id" : ObjectId("543e75a6d8146adf3408f5fb"),*

*"\_class" : "edu.ctu.cit.thesis.mbeans.model.User",*

*"username" : "admin",*

*"password" : "202cb962ac59075b964b07152d234b70",*

*"isAdmin" : true*

*}*

**ViewSetting**: lưu thông tin về việc hiển thị (phía client với mỗi client có chế độ hiển thị khác nhau) và thứ tự sắp xếp trên main form của client, khi client khởi động main form sẽ thực hiện một kết nối tới mbean server và gọi hàm để lấy về thông tin hiển thị dựa vào ip hiện tại của máy, nếu tồn tại giá trị thì main form sẽ hiển thị theo nếu không có thì sẽ hiển thị với giá trị mặc định và đồng thời thêm vào cơ sở dữ liệu với ip hiện tại.

*{*

*"\_id" : ObjectId("54317e044750a1a77128f1b8"),*

*"\_class" : "edu.ctu.cit.thesis.model.ViewSetting",*

*"order" : "asc",*

*"sortBy" : "ip",*

*"hostIp" : "192.168.137.6"*

*}*

**Ip** **Route**: lưu thông tin về các đối tượng ip route được tạo từ client bao gồm ip đích, netmask đích và nexthop.

*{*

*"\_id" : ObjectId("543ab35eb5d40c1709f2d7f4"),*

*"\_class" : "edu.ctu.cit.thesis.mbeans.model.IpRoute",*

*"destIp" : "1.0.0.0",*

*"destMask" : "255.0.0.0",*

*"nextHop" : "Ethernet1/5"*

*}*

**Discovery** **Ip**: lưu danh sách các ip đã được discovery ra để sử dụng cho việc lấy thông tin về client khi không cần phải discovery lại.

*/\* 0 \*/*

*{*

*"\_id" : ObjectId("543029a67c4028212de6bef3"),*

*"\_class" : "edu.ctu.cit.thesis.mbeans.model.DiscoveryIp",*

*"ip" : "192.168.100.10"*

*}*

*/\* 1 \*/*

*{*

*"\_id" : ObjectId("54353e29a176abe0c5927eb6"),*

*"\_class" : "edu.ctu.cit.thesis.mbeans.model.DiscoveryIp",*

*"ip" : "192.168.100.20"*

*}*

*/\* 2 \*/*

*{*

*"\_id" : ObjectId("54436c1545dd4f805834d068"),*

*"\_class" : "edu.ctu.cit.thesis.mbeans.model.DiscoveryIp",*

*"ip" : "192.168.100.30"*

*}*

**Scheduler** **Object**: lưu trữ các đối tượng scheduler để được sử dụng khi kích hoạt chế độ định thời cho server, mỗi scheduler được phân định dựa vào tên và tình trạng kích hoạt hay chưa được dựa vào document *activeScheduler*.

*{*

*"\_id" : ObjectId("54353974a176a13ad4b9aaaf"),*

*"\_class" : "edu.ctu.cit.thesis.mbeans.model.SchedulerObject",*

*"name" : "first",*

*"startIp" : "192.168.100.10",*

*"endIp" : "192.168.100.100",*

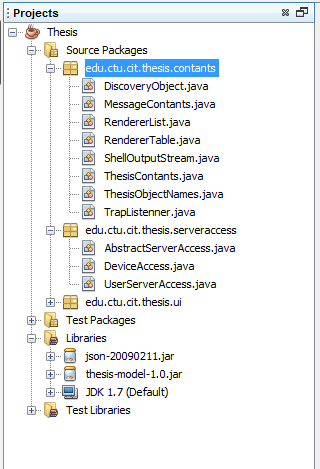
*"mask" : "255.255.255.0"*

*}*

**Active** **Scheduler**: lưu danh sách các sheduler đã được kích hoạt và đang thực thi trên server.

## CLIENT

Cấu trúc client được chia làm 3 phần: contants, serveraccess, ui và 1 file cấu hình địa chỉ kết nối đến server jmx.



Hình 3. 7 - Cấu trúc client

* Contants: là phần code định nghĩa các tính năng chung, các class model.
* Serveraccess: bao gồm 3 class để phục vụ truy cập và gọi các hàm từ server cung cấp.
* UI: là phần giao diện sử dụng Java Swing kết hợp look-and-feel tạo sự thân thiện với người dùng.
* File cấu hình: là một file properties chứa 2 thông số là địa chỉ server và jmx port.

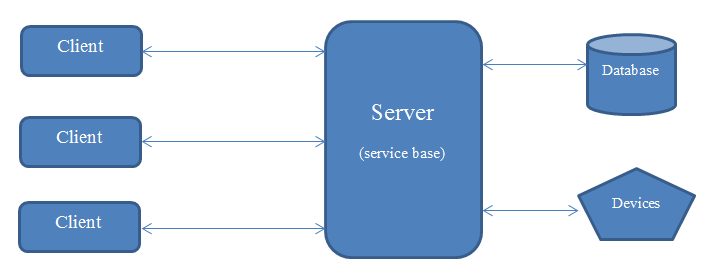
ip=192.168.100.100

port=1234

* Thư viện của client chỉ sử dụng framework chuyển đổi định dạng json là “json-20090211” và thư viện “thesis-model-1.0” là phần các class chung cho cả server và client, nó cũng bao gồm các interface cấu hình mbean server để client dự vào đấy mà gọi hàm.

## 3.3 CHỨC NĂNG VÀ CÁCH THỨC HOẠT ĐỘNG CỦA HỆ THỐNG

Hình 3.8 mô tả tổng quát cách thức hoạt động của hệ thống.



Hình 3. 8 - Chức năng tổng quát hệ thống

Hệ thống bao gồm 2 thành phần:

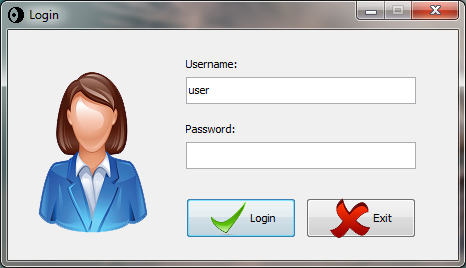
* Một server (dưới dạng service) luôn lắng nghe các kết nối RMI và thực thi các hàm do client gọi, tương tác với database hoặc thiết bị và trả về kết quả cho client.
* Một hoặc nhiều client được thiết kế dựa trên các hàm được cung cấp sẵn phía server, tạo kết nối đến server và thực thi các hàm (từ xa) tương tác xuống thiết bị cũng như trên database.

### Chức năng login

Đây là chức năng đầu tiên của hệ thống, để sử dụng được các chức năng khác của hệ thống yêu cầu phải login thành công.

Sử dụng object name phía server là *thesis.user* cho client kết nối và thực thi hàm.

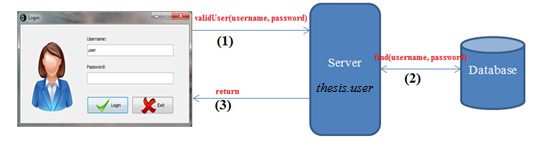
**Giao diện**:



Hình 3. 9 - Giao diện login

**Cách thức hoạt động**: Ở giao diện login có 2 chức năng chính login và exit.

Login:



Hình 3. 10 - Chức năng login

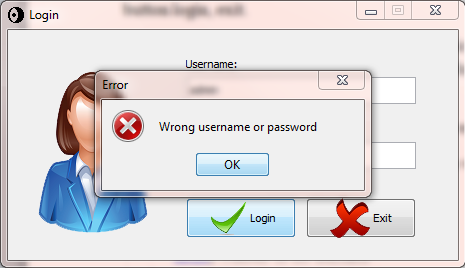
* Sau khi nhập thông tin đăng nhập (username và password), client thực hiện kết nối đến server và thực thi hàm *validUser(username, password).*
* Server khi nhận được yêu cầu xác thực sẽ thực hiện 1 truy vấn xuống database (mongodb) để tìm tài khoản với thông tin được truyền từ client.

User one = mongoTemplate.findOne(new Query(Criteria.where(USERNAME).is(username)), User.class);

* Nếu tìm thấy user trong database, server trả về kết quả *true*, nếu không tìm thấy server trả về *false*.
* Nếu client nhận về kết quả từ server là “*true*”, client khởi tạo form tiếp theo (*discovery*), nếu nhận kết quả trả về “*false*” tiếp tục cho người dùng nhập lại thông tin.

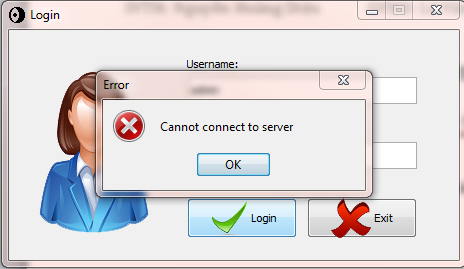
Kết quả trả về từ server:

* *False*: nếu user không tồn tại, hiển thị thông báo và cho phép nhập lại.



Hình 3. 11 - Sai thông tin đăng nhập

* *Null*: nếu không nhận được kết quả trả về sau thời gian timeout (server chưa run hoặc cấu hình kết nối sai địa chỉ server).



Hình 3. 12 - Không thể kết nối server

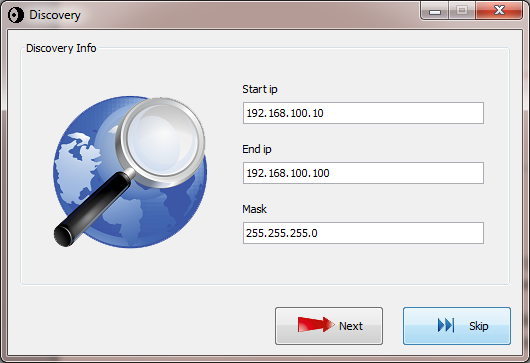
Exit: đóng ứng dụng.

### Chức năng discovery

Sau khi login thành công vào hệ thống, giao diện discovery hiển thị. Tại giao diện này cho phép người dùng nhập vào dãy ip để hệ thống tìm ra các thiết bị được cấu hình phù hợp bằng cách quét qua toàn bộ dãy ip. Tại đây, người dùng có thể chọn “*skip*” để sử dụng thông tin thiết bị được lưu ở server mà không cần phải tìm lại.

Sử dụng object name phía server là *thesis.device* cho client kết nối và thực thi hàm.

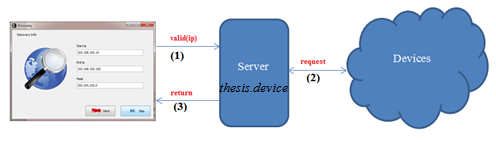
**Giao diện:**



Hình 3. 13 - Giao diện discovery

**Cách thức hoạt động**: Ở giao diện discovery gồm 2 chức năng chính là next và skip.

Next:



Hình 3. 14 - Sơ đồ chức năng discovery

* Sau khi nhập dãy ip, client tạo 1 kết nối đến server. Lặp qua tất cả ip, với mỗi ip client thực hiện gọi hàm valid(ip) từ server và chờ kết quả trả về.
* Khi được yêu cầu kiểm tra sự tồn tại của thiết bị, server thực hiện gởi đi gói tin (UDP theo định dạng gói SNMP) yêu cầu trả về tên thiết bị đến ip được truyền từ client.
* Nếu server nhận được kết quả trả về từ thiết bị, server trả về cho client giá trị “*true”* đồng thời lưu ip vừa tìm thấy vào database để sử dụng cho lần tiếp theo, nếu sau thời gian timeout mà server không nhận được trả lời từ thiết bị server sẽ trả về kết quả là “*false*” cho client.
* Với những ip mà server trả về “*true*” client sẽ lưu vào bộ nhớ để quản lý và truy vấn ở giao diện chính.
* Trong quá trình discovery, với các ip có tồn tại kết quả trả về sẽ là rất nhanh. Với các ip không có thiết bị hoặc với lỗi nào đó mà việc yêu cầu hoặc trả lời không hoàn thành thì thời gian chờ sẽ khá lâu. Chính vì thế trong quá trình discovery phía client cho chạy cùng lúc 10 luồng (thread/thread pool trong java) để thực thi song song nhầm giảm thiểu thời gian chờ (số lượng luồng có thể tăng lên để giảm thời gian). Do phía server chạy các RMI nên 1 hàm có thể được thực thi 1 cách song song và phục vụ cho nhiều client.

Skip: Bỏ qua quá trình discovery mà sử dụng dụng thông tin đã được lưu trên server đồng thời gán giá trị cho biến skip bằng “*true*” để xác nhận khi main form khởi tạo sẽ lấy danh sách thiết bị từ server.

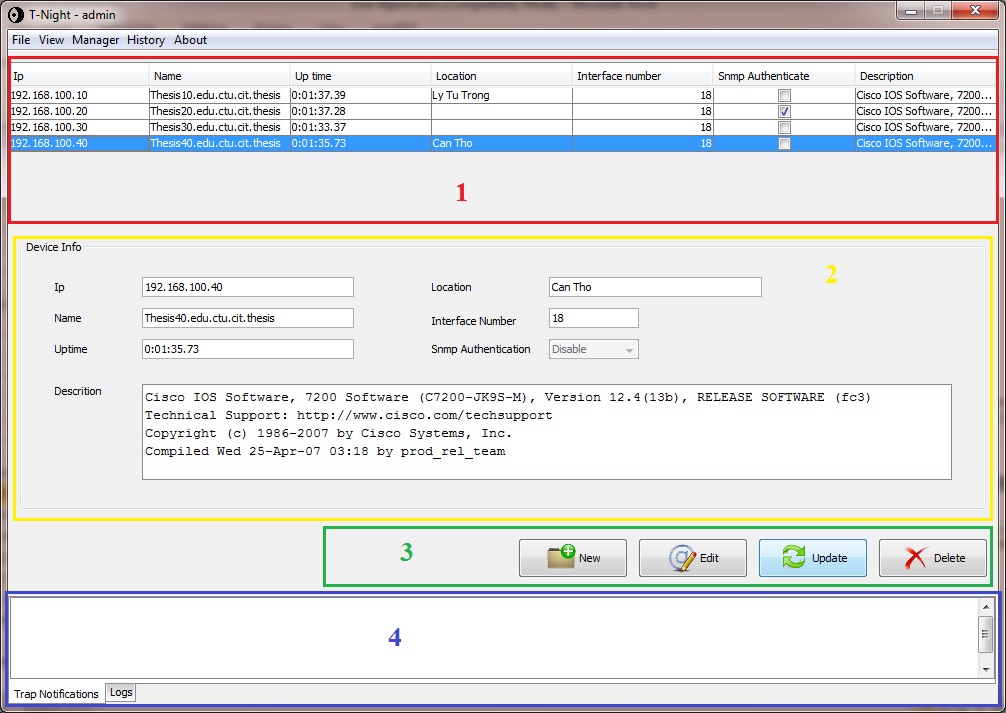
* Sau khi quá trình discovery hoàn tất hoặc người dùng chọn *skip*, giao diện chính (*MainForm*) khởi tạo và hiển thị thông tin tổng quát của những thiết bị đã được tìm thấy hoặc lấy về từ server.

### Chức năng hiển thị thông tin tổng quát (main form)

Đây là giao diện chính của hệ thống, bao gồm việc hiển thị danh sách các thiết bị và thông tin tổng quát của từng thiết bị. Một số chức năng thao tác cơ bản với thiết bị. Tại giao diện này với thanh menu, phím tắt và popup menu cho phép thao tác toàn bộ các chức năng của hệ thống.

Sử dụng 2 object name là *thesis.user* và *thesis.device* phía server cho client kết nối và thực thi hàm.

**Giao diện:**

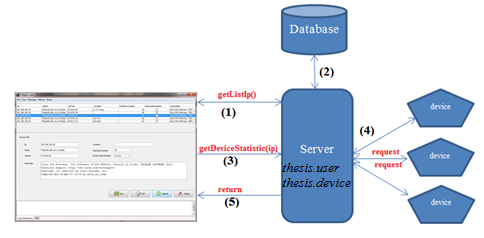


Hình 3. 15 - Giao diện chính

Giao diện gồm 5 thành phần chính:

* (1) Là bảng chứa danh sách các thiết bị có thể được sắp xếp cho tiện dụng.
* (2) Là thông tin cụ thể của từng thiết bị tương ứng với bảng phía trên.
* (3) Là 4 nút chức năng cơ bản.
* (4) Là phần hiển thị thông tin cảnh báo từ thiết bị (trap) và logs từ server.
* Trên cùng là thanh menu bao gồm tất cả các chức năng của hệ thống.

**Cách thức hoạt động:**



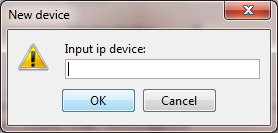
Hình 3. 16 - Mô hình hoạt động khi khởi tạo mainform

Khi khởi tạo form:

* Main form kiểm tra giá trị *skip*, nếu bằng “*true*” client thực hiện kết nối đến server và yêu cầu nhận về danh sách ip (gọi hàm *getListIp())* được lưu trên database.
* Khi server nhận được yêu cầu, thực hiện truy vấn database và trả về danh sách ip (có thể null) cho client.
* Khi có được danh sách các ip, client lặp qua toàn bộ và gởi yêu cầu lấy thông tin chi tiết từng thiết bị (sử dụng hàm *getDeviceStatistic(ip)).*
* Server khi nhận được yêu cầu lấy thông tin thiết bị, thực hiện gởi các gói tin đến ip yêu cầu lấy thông tin (sử dụng snmpget) từ thiết bị và trả về cho client (thông tin được truyền dưới định dạng json).

New:

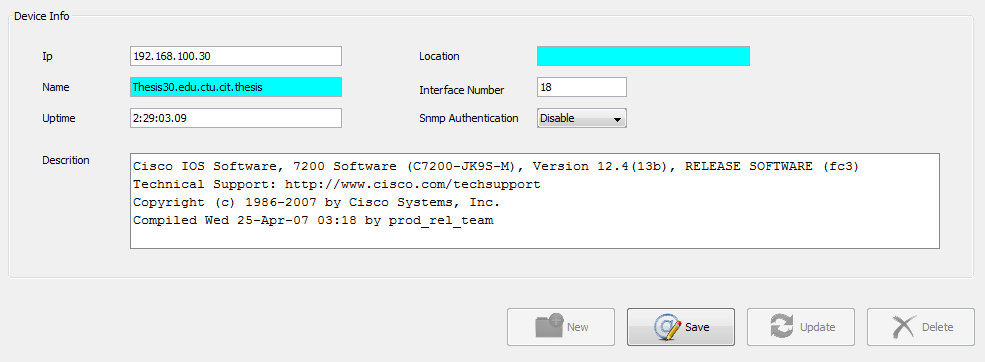
Hiển thị input cho phép nhập vào ip mới.



* Sau khi nhập ip, client thực thi hàm valid(ip) để kiểm tra việc tồn tại thiết bị tại ip đó.
* Nếu kết quả trả về từ server là có (*true*), client tiếp tục yêu cầu lấy thông tin chi tiết và cập nhật vào bảng devices cũng như lưu vào bộ nhớ để quản lý, phía server sẽ thêm ip này vào database cho việc sử dụng lại.

Edit:

Cho phép thay đổi một số thông tin trên thiết bị như name, location.



Hình 3. 17 - Giao diện edit - mainform

* Sau khi bấm nút “Save” client lấy thông tin, chuyển thành định dạng json và gọi hàm *updateDevice(json).*
* Server nhận được yêu cầu, chuyển tham số dạng json dang đối tượng và tạo gói tin snmp (snmp set) gởi đến thiết bị yêu cầu thay đổi giá trị.
* Sau khi thực hiện xong, client tiến hành cập nhật lại thông tin danh sách thiết bị.

Update:

Tiến hành cập nhật lại thông tin toàn bộ thiết bị, giống với việc lấy thông tin lúc khởi tạo form.

Delete:

* Tiến hành xóa device ra khỏi bộ nhớ quản lý của client.
* Gởi yêu cầu xóa ip đã lưu trong database bằng việc gọi hàm *removeDiscoveryIp(ip).*
* Server tiến hành xóa ip ra khỏi database.

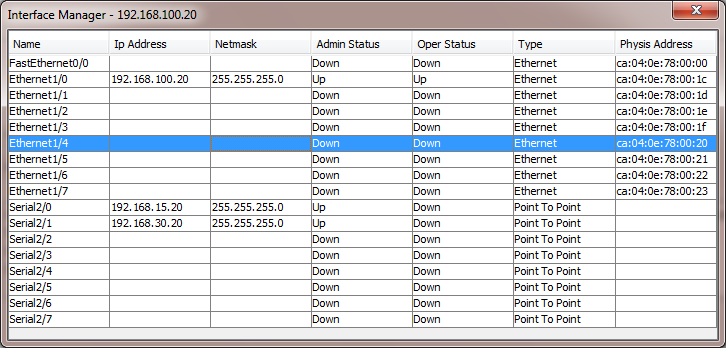
mongoTemplate.remove(new Query(Criteria.where("ip").is(ip)), DiscoveryIp.class);

### Chức năng interface manager

Đây là một chức dùng để quản lý các interface của router, cho phép lấy thông tin liên quan đến interface của thiết bị, sắp xếp và hiển thị cho người dùng. Tại đây người dùng có thể thay đổi một số giá trị như: đặt địa chỉ ip, bật/tắt giao diện. Để sử dụng chức năng này từ menu *Manager 🡪 Interface Manager*.

Sử dụng object name phía server là *thesis.device* cho client kết nối và thực thi hàm.

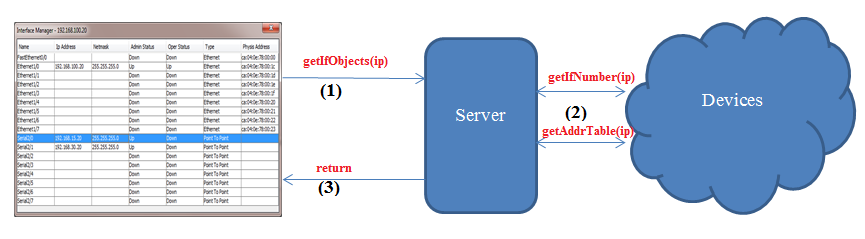
**Giao diện:**



Hình 3. 18 - Giao diện interface manager

**Cách thức hoạt động:**

Khởi tạo form:



Hình 3. 19 - Cách thức hoạt độn interface manager

* Khi khởi tạo form, client bắt sự kiện người dùng đang chọn vào thiết bị nào và lấy ip từ thiết bị đó.
* Client tiến hành kết nối server và thực thi hàm *getIfObjects(ip)* để nhận về danh sách các interface của thiết bị.
* Server nhận được yêu cầu, gởi các gói tin (*snmp get*) yêu cầu nhận về danh sách interface và cấu hình ip tương ứng với interface, tổng hợp thành đối tượng và chuyển thành định dạng json trả về cho phía client.

DvGet dvGet = **new** DvGet(ip);

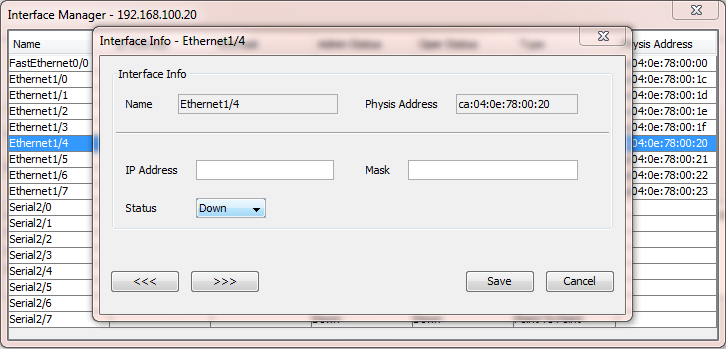
List<IfObject> ifTableObject = dvGet.getIfObjects();

dvGet.close();

**return** IfObject.*parseToString*(ifTableObject);

Click chọn vào 1 interface:

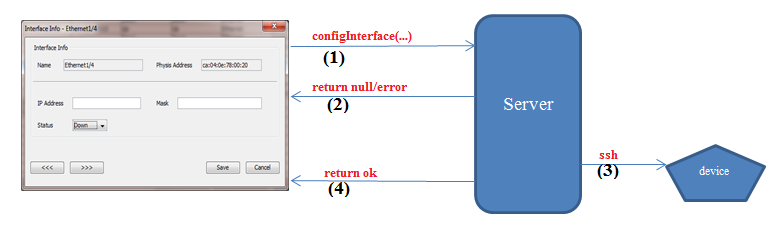
Khi chọn vào 1 interface trên giao diện, 1 popup panel được khởi tạo và hiển thị.



Hình 3. 20 - Giao diện popup - interface manager

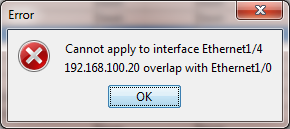
Giao diện hiển thị các thông tin của interface được người dùng chọn. Tại đây, người dùng có thể thiết lập ip (nếu chưa có ip) hoặc thay đổi ip (nếu đã có ip) trên giao diện hiện tại, phần status chọn up/down để bật/tắt giao diện.

Save:



Hình 3. 21 - Sơ đồ hoạt động chức năng save

* Client tiến hành lấy các thông tin như: name, ip, mask, status và kết nối đến server thực thi hàm *configInterface(name, ip, mask, status).*
* Server nhận được yêu cầu cấu hình giao diện, tiến hành kiểm tra các thông tin được truyền từ client và thực thi.
* Nếu thông tin không đúng, server trả về null
* Nếu thông tin đúng nhưng ip và mask trùng (nhánh mạng) với 1 giao diện nào đó, server trả về câu thông báo lỗi.



* Nếu server kiểm tra mọi thứ đều đúng, server tiến hành tạo 1 kết nối ssh đến thiết bị và thực hiện cấu hình trên giao diện đó với các thông tin được truyền từ client.

CLISet cliSet = **new** CLISet(hostIp);

cliSet.setIpInterface(interfaceName, ip, mask, status);

cliSet.close();

Cancel:

Đóng popup panel và trở về giao diện interface manager.

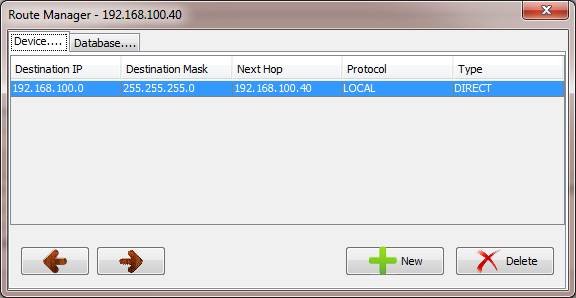
### Chức năng route manager

Một chức năng chính tiếp theo là chức năng route manager, là chức năng cho phép hiển thị thông tin vạch đường của thiết bị, cho phép tạo thêm và apply xuống thiết bị, có hỗ trợ apply 1 danh sách thiết bị với thông tin vạch đường giống nhau.

Sử dụng 2 object name phía server là *thesis.user* và *thesis.device* cho client kết nối và thực thi hàm.

Để vào giao diện route manager (hình 3.26), từ menu chính chọn *manager 🡪 route manager*.

**Giao diện:**

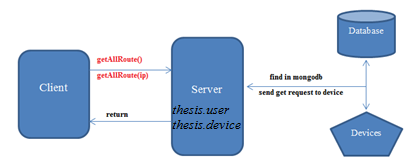


Hình 3. 22 - Giao diện route manager

Route manager bao gồm 2 tabs: device và database:

* Device: các route đang được áp dụng trên thiết bị, có thể xóa được nếu loại (type) không phải là “DIRECT”.
* Database: các route đã được cấu hình và lưu trên server, có thể đã áp dụng xuống thiết bị hoặc chưa. Cho phép apply xuống cùng lúc nhiều device với thông tin vạch đường giống nhau.

**Cách thức hoạt động:**



Hình 3. 23 - Cách thức hoạt động route manager

Device:

Giao diện bao gồm 2 chức năng chính: new (thêm mới và áp dụng xuống thiết bị), delete (xóa thông tin vạch đường trên thiết bị).

* Khi khởi tạo form:
* Clien bắt sự kiện ip đang được chọn lấy ip thiết bị, thực hiện kết nối đến server và thực thi hàm getAllRoute(ip).
* Server tiến hành gởi gói tin (snmp get) đến ip yêu cầu nhận về thông tin route từ thiết bị.
* Server nhận thông tin từ thiết bị, chuyển đổi thành định dạng json và trả về cho client.

List<IpRoute> ipRoutes = mongoTemplate.findAll(IpRoute.**class**);

**if** (ipRoutes.size() != 0) {

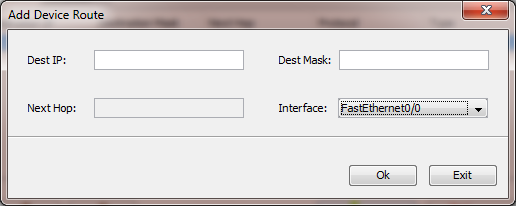
**return** IpRoute.*parseToString*(ipRoutes);

}

**return** **null**;

* Nút New:

Hiển thị popup form cho phép nhập thông tin route như: địa chỉ đích, mạng đích và next hop.



Hình 3. 24 - Giao diện add route

* Dest Ip: địa chỉ mạng đích.
* Dest Mask: mặt nạ mạng đích.
* Next Hop/Interface: chọn 1 trong 2, có thể nhập địa chỉ ip trong khung next hop hoặc chọn tên giao diện thay cho địa chỉ.
* Chọn Ok, client lấy thông tin và chuyển định dạng json, thực thi hàm addRoute(ip, json).
* Server kiểm tra thông tin và apply xuống thiết bị (bằng ssh), trả về thông báo lỗi nếu có lỗi xảy ra.

CLISet cliSet = **new** CLISet(hostIp);

cliSet.setIpInterface(interfaceName, ip, mask, status);

cliSet.close();

* Nút Delete:

Xóa thông tin route trên thiết bị (chỉ xóa được nếu type không phải DIRECT).

Server thực hiện xóa route trên thiết bị (gọi hàm *removeRoute(ip, json)*).

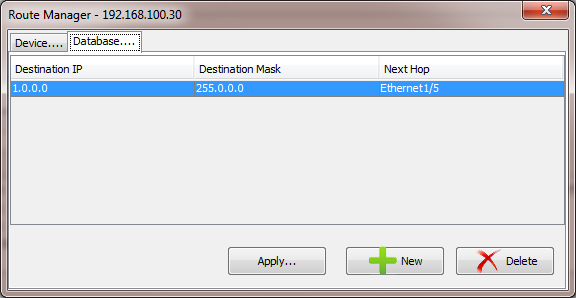
CLISet cliSet = **new** CLISet(hostIp);

cliSet.removeRoute(ipRoute.getDestIp(), ipRoute.getDestMask(), ipRoute.getNextHop());

cliSet.close();

Database:

Tương tự với tab device, thông tin vạch đường được lưu trong database và cho phép apply xuống cùng lúc nhiều thiết bị với thông tin vạch đường giống nhau.



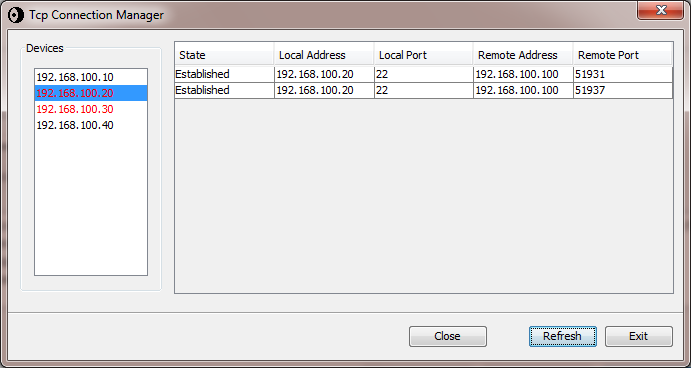
Hình 3. 25 - Tab database

### Chức năng connection manager

Là chức năng quản lý và theo dõi các tcp connection đến thiết bị. Để sử dụng chức năng này, từ menu *Manager 🡪 Tcp Connection*.

Sử dụng object name phía server là *thesis.device* cho client kết nối và thực thi hàm.

**Giao diện:**



Hình 3. 26 - Giao diện tcp connection manager

Giao diện bao gồm 2 thành phần: danh sách thiết bị và danh sách các kết nối (nếu có) đến thiết bị. Để dể quản lý và theo dõi với các thiết bị có kết nối sẽ được đổi màu đỏ.

**Cách thức hoạt động:**

* Khởi tạo form:

Form được khởi tạo được truyền danh sách thiết bị đang được quản lý từ main form.

* Click vào thiết bị:
* Client thực hiện kết nối server và thực thi hàm *getTcpConnections(ip)* để lấy về danh sách các kết nối đến thiết bị.
* Server thực hiện gởi gói snmp walk lấy về *TcpConnectionObject* chuyển về định dạng json và trả về cho client.

DvWalk dvWalk = **new** DvWalk(hostIp);

List<TcpConnectionObject> tcpConnectionObjects = dvWalk.getTcpConnectionObjects();

dvWalk.close();

**if** (tcpConnectionObjects == **null**) {

**return** **null**;

}

**return** TcpConnectionObject.*parseToString*(tcpConnectionObjects);

* Nút Close:

Buộc đóng 1 kết nối.

* Nút Refresh:

Cập nhật lại thông tin các kết nối của thiết bị

* Nút Exit:

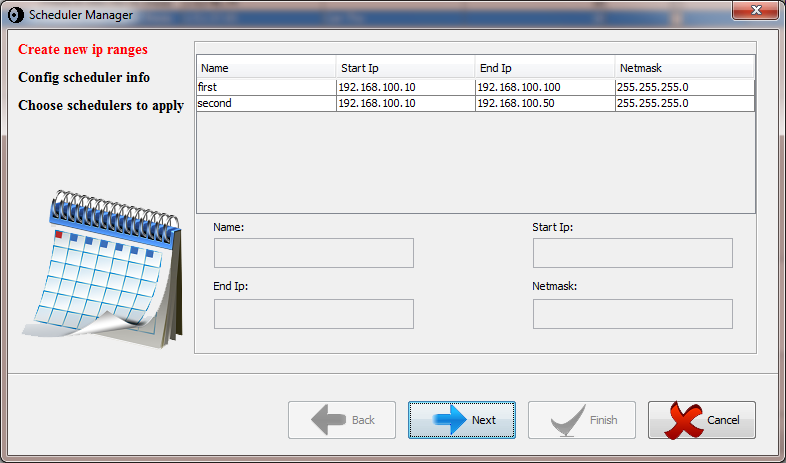
Thoát chức năng, quay trở lại main form.

### Chức năng scheduler manager

Một chức năng khá quan trọng và cần thiết trong 1 hệ thống. Cho phép định nghĩa các dãy ip và thời gian để server tự động discovery dãy ip, tìm ra các ip đang hoạt động cũng như loại bỏ việc quản lý với những ip đã tắt. Là hướng phát triển cho phía server, giúp giảm thiểu thời gian làm việc khi có yêu cầu từ phía client.

Sử dụng 2 object name là *thesis.user* và *thesis.device* phía server cho client kết nối và thực thi hàm.

Để sử dụng chức năng này (cần có quyền admin) từ giao diện chính *Manager 🡪 Scheduler* Manager hoặc sử dụng phím tắt “*control + shift + s*”. Mặc định khi server khởi động sẽ không chạy scheduler.



Hình 3. 27 - Giao diện scheduler manager

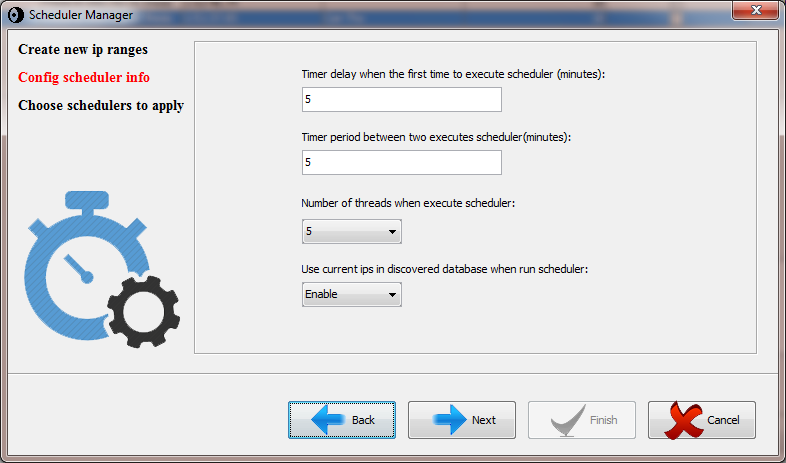
Để khởi động scheduler cần cấu hình qua 3 bước: tạo dãy ip (nếu cần), cấu hình thời gian, chọn dãy ip (nhiều) để thực thi scheduler. Hình 3.31 là giao diện đầu tiên của sheduler manager.

Để cấu hình scheduler cho server cần thực hiện qua 3 bước:

Bước 1: định nghĩa dãy ip.

* Client tạo kết nối đến server và lấy các scheduler được tạo sẵn phía server (gọi hàm *getAllScheduler()*) và hiển thị chúng dưới dạng bảng. Tại đây, ta có thể tạo thêm dãy ip bằng phía tắt “*control + n*” (vì lí do thẩm mỹ nên không sử dụng button).
* Tạo mới, nhập đầy đủ thông tin và chọn save (*control + s*) hoặc hủy (*control + c*).

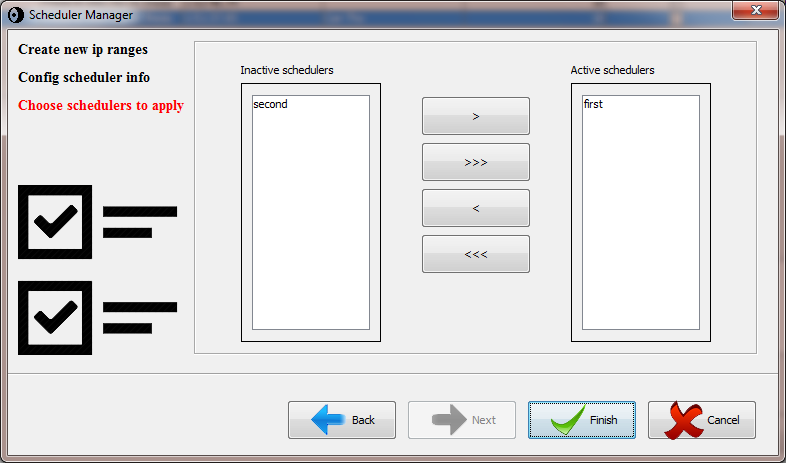
Bước 2: định nghĩa thời gian thực thi.



Hình 3. 28 - Bước 2 cấu hình scheduler manager

Ở bước 2, ta sẽ cấu hình 4 thông tin: thời gian bắt đầu thực thi scheduler, thời gian lặp lại, số thread server cần sử dụng (càng cao server thực thi càng nhanh nhưng chiếm tài nguyên server) và tùy chọn có sử dụng danh sách ip hiện tại.

Bước cuối: chọn danh sách dãy ip để thực thi.



Hình 3. 29 - Bước 3 cấu hình sheduler manager

Ở bước này, client lấy danh sách các scheduler từ server (kể cả đã apply và chưa apply) và hiển thị ở 2 khung chọn khác nhau. Cho phép chọn danh sách các dãy ip để apply cho server. Khi apply server sẽ hủy tất cả scheduler củ và tiến hành thực thi scheduler mới.

* Nếu trong khung “selected” không có tên nào được chọn, server sẽ hủy scheduler.

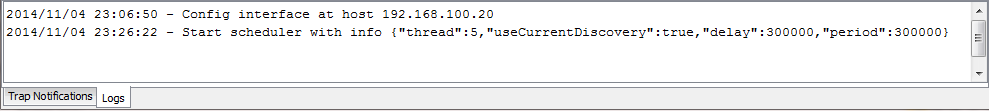
Sau khi chọn Finish, client lấy toàn bộ dữ liệu chuyển thành định dạng json và gọi hàm thực thi scheduler (*startScheduler(json)*). Server tiến hành hủy các scheduler đang thực thi (nếu có) và khởi động scheduler mới.

timer = **new** Timer();

task = **new** SchedulerTask(schedulerInfo.getUseCurrentDiscovery(), schedulerInfo.getThread());

timer.schedule(task, schedulerInfo.getDelay(), schedulerInfo.getPeriod());

Sau khi khởi động scheduler, server gởi đi một thông báo (jmx notification) đến toàn bộ client đang hoạt động thông tin scheduler – hình 3.34 là phần hiển thị thông báo tại khung thông báo của main form khi nhận được thông báo phía servver.



Hình 3. 30 - Thông báo khởi động scheduler mới

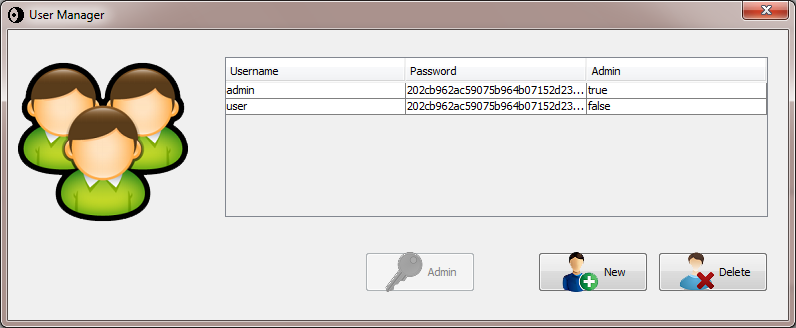
### Chức năng user manager

Là chức năng dùng để quản lý user, bao gồm các thao tác cơ bản như: thêm, sửa, xóa và gán user trở thành admin (với user chưa phải admin). Để có thể sử dụng được chức năng này cần phải login với quyền admin.

Sử dụng object name phía server là *thesis.user* cho client kết nối và thực thi hàm.

Để sử dụng chức năng, từ giao diện chính chọn Manager 🡪 Users hình 3.35 là giao diện User manager.

**Giao diện:**



Hình 3. 31 - Giao diện user manager

**Cách thức hoạt động:**

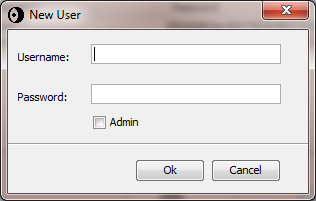
* Khởi tạo form:

Khi khởi tạo, client thực hiện kết nối đến server thực thi hàm *getAll()* để lấy về danh sách toàn bộ user.

Server khi nhận được yêu cầu, thực hiên truy vấn xuống database lấy về danh sách user và trả về cho client.

* Nút New:

Hiển thị form nhập thông tin user



Hình 3. 32 - Add user

Sau khi chọn Ok, client lấy thông tin kết nối đến server và thực thi hàm addUser(username, password, isAdmin) để thêm user vào cơ sở dữ liệu.

Server nhận được yêu cầu tiến hành lưu thông tin vào database.

*mongoTemplate.save(user);*

Trả về thông báo lỗi nếu username đã tồn tại.

* Nút Delete:

Client lấy username, kết nối server thực thi hàm deleteUser(username).

Server tiến hành xóa user ra khỏi database.

mongoTemplate.remove(new Query(Criteria.where(USERNAME).is(username)), User.class);

* Nút Admin:

Client lấy username, kết nối server thực thi hàm setAdmin(username).

Server tiến hành set username trở thành admin.

mongoTemplate.updateFirst(new Query(Criteria.where(USERNAME).is(username)), Update.update(ISADMIN, true), User.class);

# CHƯƠNG 4: KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN

**Kết quả đạt được:**

Hệ thống mạng ngày càng rộng lớn, thiết bị mạng ngày càng trở nên đa dạng và số lượng tăng nhanh. Việc quản lý, cấu hình và phát hiện sự cố ngày càng trở nên là vấn đề phức tạp đối với các nhà quản trị mạng. Việc xây dựng được hệ thống hỗ trợ quản lý thiết bị và tiện dụng là vấn đề hết sức cần thiết.

Luận văn này, nghiên cứu tìm hiểu các công nghệ mới và nghiên cứu từ dự án thực tế để rút kết lại những phần cốt lỗi mở ra hướng phát triển ứng dụng một cách nhanh chóng và dể dàng trên các công nghệ mới đó.

Hệ thống nếu được đầu tư và phát triển đến mức toàn diện có thể trở thành hệ thống thực sự cần thiết với các nhà quản trị mạng và tiện dụng đến mức có thể quản trị, cấu hình hay nhận thông báo xử lý lỗi của một hệ thống lớn chỉ với thiết bị cầm tay (smartphone).

Những vấn đề đạt được:

* Nắm bắt được phần cốt lỗi của spring framework và triễn khai được spring jmxrmi hỗ trợ cho gọi hàm từ xa.
* Hiểu và áp dụng được giao thức SNMP cho cấu hình cũng như lấy thông tin từ thiết bị. Đồng thời xử lý các gói tin TRAP từ thiết bị.
* Nắm bắt được công cụ build và áp dụng một cách uyển chuyển để quản lý project một cách thuận tiện.

**Hướng phát triển:**

Hệ thống nếu phát triển một các toàn diện:

* Về phía server, sẽ cài đặt được nhiều hàm hỗ trợ thao tác xuống thiết bị hơn và để quản lý số lượng thiết bị cực lớn và cần rút ngắn thời gian thì server sẽ được thiết kế theo dạng phân tán với sự hỗ trợ của “akka framework”.
* Về phía client, sẽ được thiết kế dưới dạng web application và có thể tương thích với các thiết bị cầm tây để thuận tiện trong việc nhận thông báo sự cố từ thiết bị và cấu hình quản trị một cách dể dàng thuận tiện với sự hỗ trợ của JavaScript MVC (AngularJS hay CanJS framework).
* Về tính năng: có thể phát triển lên hầu hết các thiết bị có hỗ trợ snmp và chứ không chỉ là router của cisco.

# HẠN CHẾ

Do phạm vi luận văn có giới hạn, nên kết quả chỉ dừng lại ở mức desktop application thay vì là web application và tương thích với các thiết bị cầm tay.

Chỉ sử các thiết bị trên mô hình với sự hỗ trợ của GNS3 nên không được hỗ trợ đầy đủ các tính năng của thiết bị, vì thế trong quá trình phát triển mô hình chỉ phát triển các tính năng mà thiết bị ảo hỗ trợ.

Do không có thiết bị thực tế để kiểm thử nên việc phát triển ứng dụng cũng chỉ dựa trên các dòng router của cisco thay vì các thiết bị khác và mỡ rộng hơn là tất cả những thiết bị có hỗ trợ snmp.

Với các tính năng thật sự cần thiết nhưng do thiết bị ảo không hỗ trợ (file mib trong việc lấy thông tin đối tượng snmp) nên phải sử dụng ssh thời gian thực thi có phần chậm hơn.

# TÀI LIỆU THAM KHẢO

1. *http://www.oidview.net/tools/index.html*
2. *http://techdive.in/snmp/snmp4j-snmp-get-example*
3. *http://www.ehow.com/how\_5001442\_configure-snmp-cisco-router.html*
4. *http://download.intermapper.com/thirdparty/libsmi-0.4.8/mibs/dirlist.php*
5. *http://www.iphostmonitor.com/mib/mibs/IP-ACCESS-LIST-MIB.html*
6. *http://tools.cisco.com/ITDIT/MIBS/servlet/index*
7. *http://www.gns3.net/documentation/gns3/connecting-gns3-to-real-networks/*
8. *http://www.oidview.com/mibs/1751/IP-ACCESS-LIST-MIB.html*
9. *http://www.shivasoft.in/blog/java/snmp/create-snmp-client-in-java-using-snmp4j/*
10. *http://www.javacodegeeks.com/2012/07/jmx-and-spring-part-2.html*
11. *http://www.journaldev.com/1352/what-is-jmx-mbean-jconsole-tutorial*
12. *http://nikunjp.wordpress.com/2011/07/30/remote-ssh-using-jsch-with-expect4j/*
13. *http://www.ashishpaliwal.com/blog/2008/12/multi-threaded-trap-receiver-using-snmp4j/*
14. *http://techdive.in/snmp/snmp4j-trap-receiver*
15. *http://www.cisco.com/c/en/us/td/docs/ios/netmgmt/command/reference/nm\_book/nm\_20.html*
16. *http://www.netcraftsmen.net/resources/archived-articles/370-configuring-snmp-in-cisco-routers.html*
17. *http://ducnd94.wordpress.com/2014/07/18/bai-2-tong-quan-ve-spring-framework/*
18. *http://laptrinh.vn/d/3910-gioi-thieu-ve-spring-framework.html*
19. *http://nodejs.vn:4567/topic*
20. *http://doc.edu.vn/tai-lieu/de-tai-giao-thuc-tcpip-va-mang-internet-7731/*
21. *http://bigsonata.com/mongodb/*

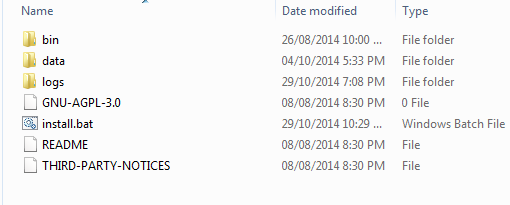
# PHỤ LỤC

## PHỤ LỤC: CÀI ĐẶT VÀ CẤU HÌNH MONGO DATABASE

### CÀI ĐẶT

Tải về phiên bản mongodb tương thích ở địa chỉ: <http://www.mongodb.org/downloads>

Sau khi tải về, gải nén ra thư mục. vào thư mục mongodb tạo 2 thư mục data và logs để lưu dữ liệu và log. Cấu trúc thư mục mongodb:



Tạo file install.bat trong thư mục mongodb với nội dung cấu hình như sau:

*mongod.exe --install --journal*

*--dbpath D:\softs\mongodb-win32-i386-2.6.4\data\db*

*--logpath D:\softs\mongodb-win32-i386-2.6.4\logs\mongodb.logs*

*--serviceName T-NightMongoDb*

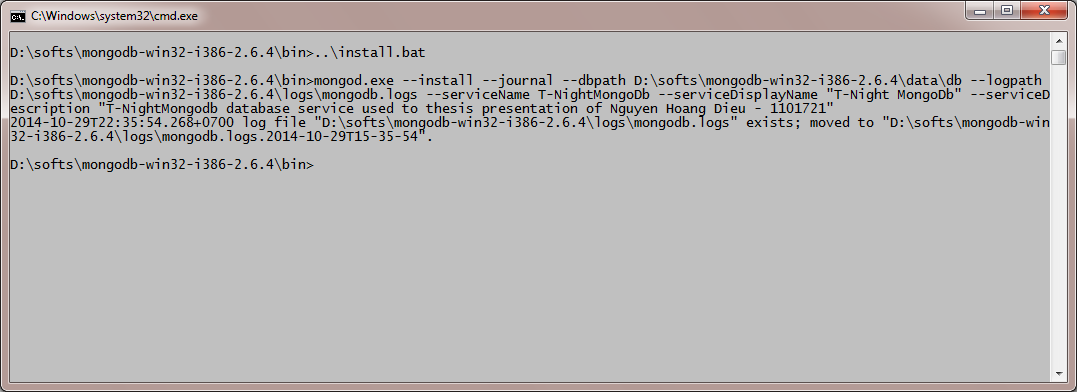
*--serviceDisplayName "T-Night MongoDb"*

*--serviceDescription "T-NightMongodb database service used to thesis presentation of Nguyen Hoang Dieu - 1101721"*

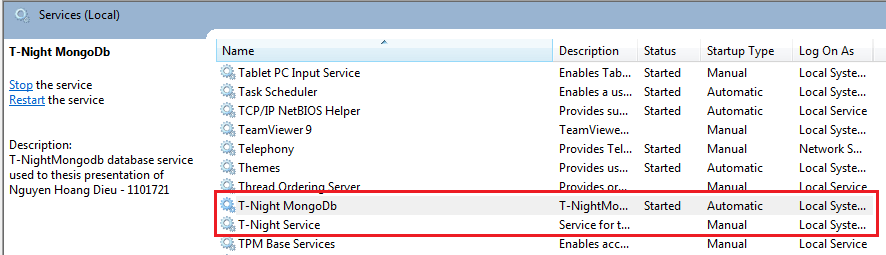
Giải thích tham số trong install.bat:

* *dbpath*: là đường dẫn đến thư mục chứa dữ liệu (data), nên chọn đường dẫn tuyệt đối để tránh phát sinh lỗi.
* *logpath*: là đuồng dẫn đến thư mục chứa log.
* *serviceName*: là tên service sẽ được hệ điều hành quản lý khi cài đặt thành công thành service.
* *serviceDisplayName*: là tên hiển thị của service.
* *serviceDescription*: là phần thông tin chi tiết của service.

Sau khi tạo file install.bat, mở cmd tại thư mục bin của mongodb và thực thi install.bat.

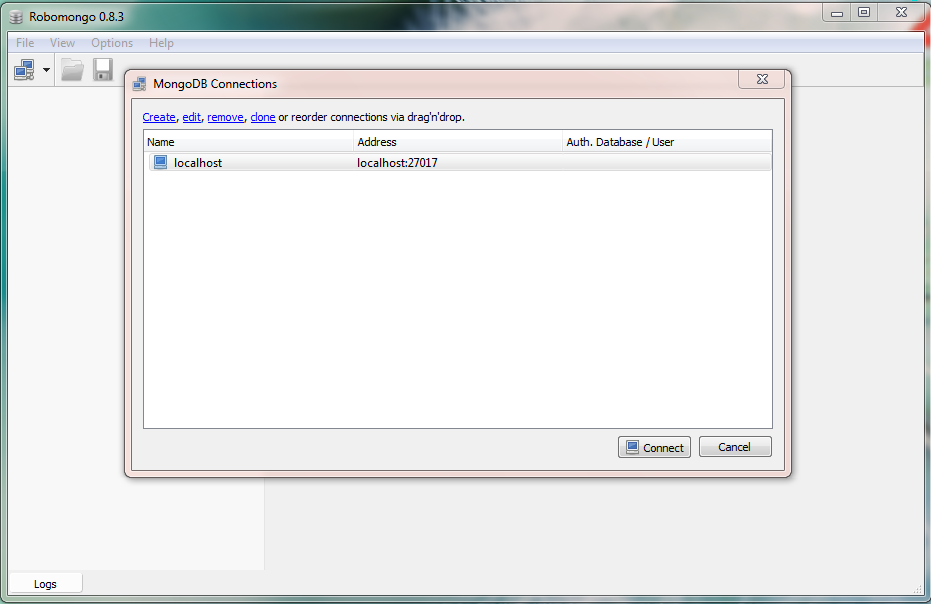


Như vậy, service mongodb đã được cài đặt thành công.

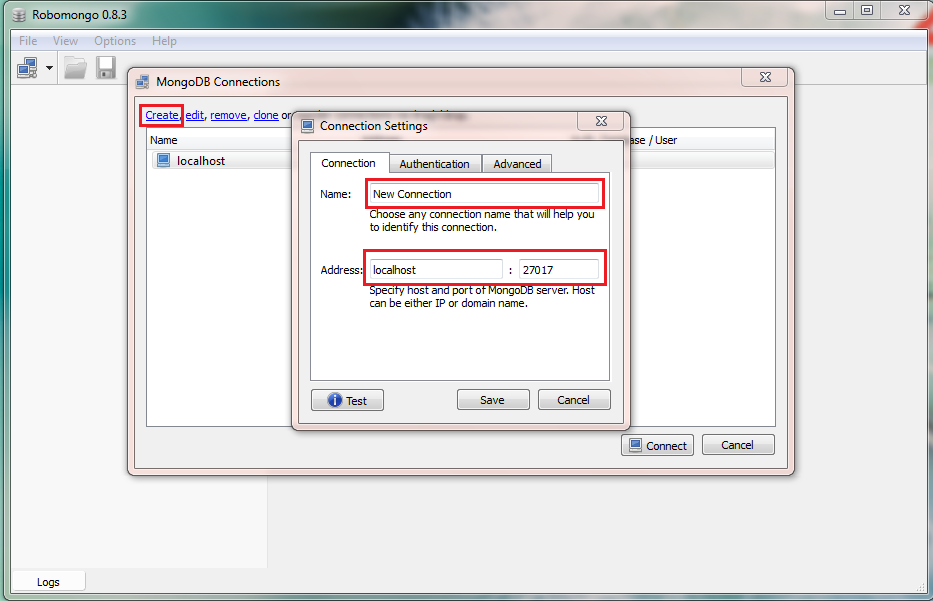


### TRUY VẤN

Giống như CSDL quan hệ, MongoDB cũng hỗ trợ truy vấn với các câu điều kiện phức tạp. Robomongo là một công cụ cho phép thiết lập kết nối và thực hiện các truy vấn cũng như hiển thị kết quả với các câu truy vấn MongoDB, hình dưới là giao diện tổng quá của Robomongo.



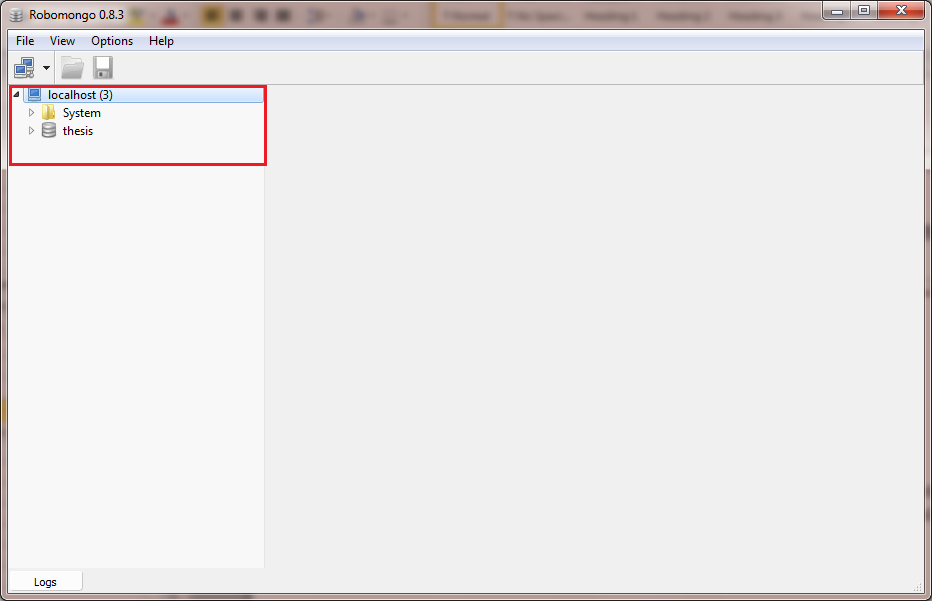
Giao diện tổng quát Robomongo



Tạo mới connection

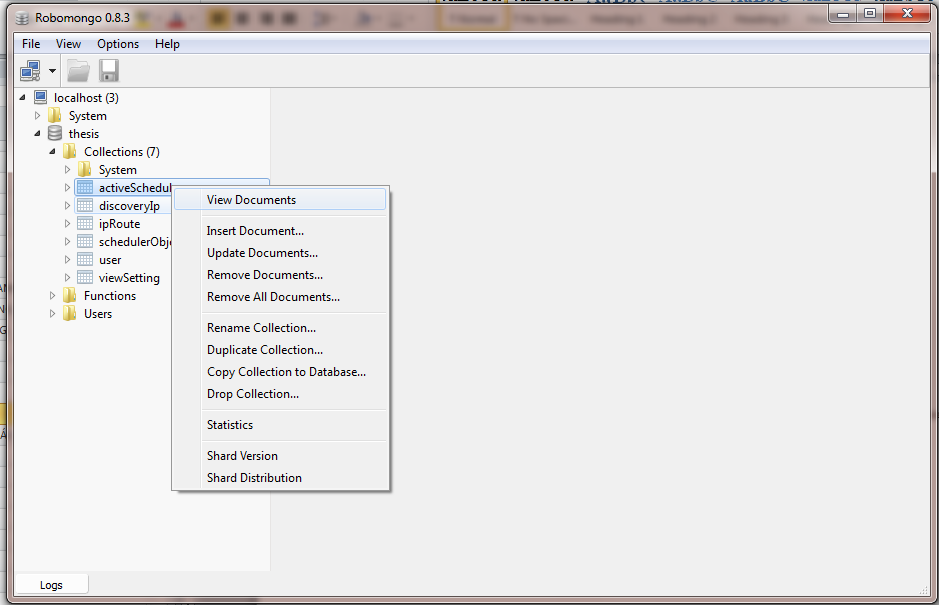
Để tạo kết nối đến MongoDB cần tạo mới 1 connection với 1 số thông tin như sau:

* Chọn Create, 1 popup hiện lên và điền các thông tin cần thiết
* Address: địa chỉ ip server MongoDB
* Port: mặc định là 27017
* Nếu database có cần xác thực thì nhập thông tin xác thực và chọn database, sau đó thực hiện kết nối. Hình 2.14 là giao diện Robomongo khi kết nối thành công đến MongoDB.



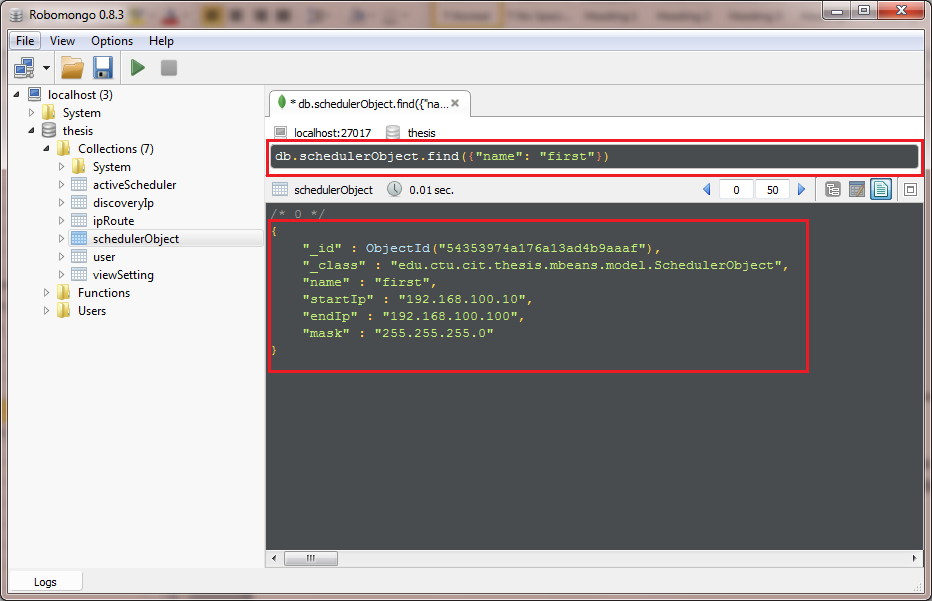
Giao diện kết nối thành công MongoDB

Để hiển thị toàn bộ dữ liệu của một document (table) của một database, chọn vào document và chọn “View Documents”.



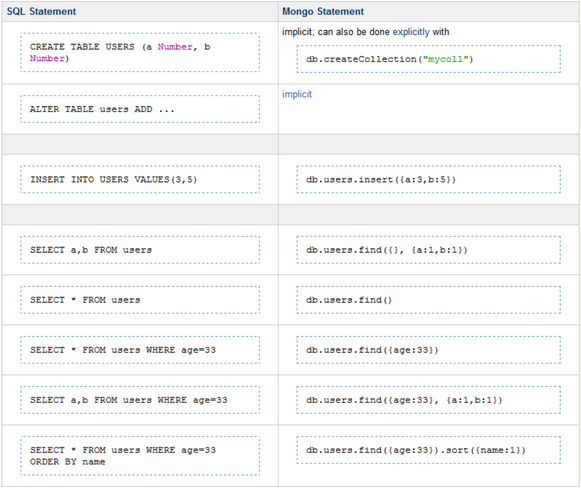
Hiển thị document trong Robomongo

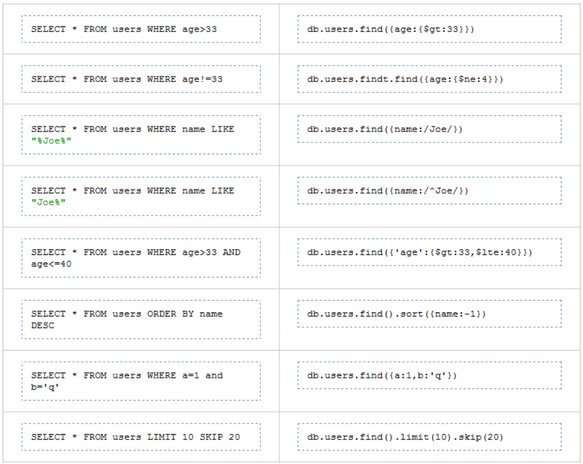
Sau khi hiển thị dữ liệu của 1 document nào đó, ta có thể thực hiện các câu truy vấn như CSDL quan hệ. Các câu truy vấn cần nhập vào khung truy vấn và nhấn f5 để hiển thị kết quả.



Truy vấn trong Robomongo

Sau đây là phần so sánh giữa các câu lệnh truy vấn cơ bản của CSDL quan hệ (mysql) và MongoDB.







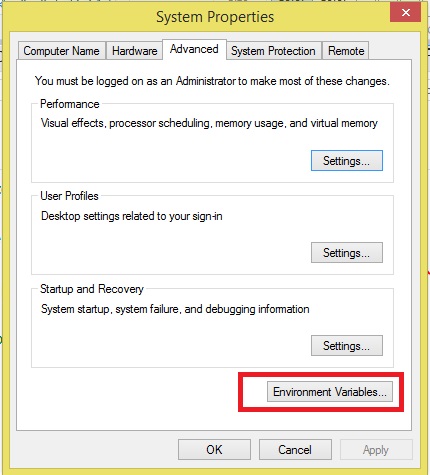
## PHỤ LỤC: CÀI ĐẶT VÀ THIẾT LẬP MÔI TRƯỜNG JAVA

Vào trang : [*http://www.oracle.com/technetwork/java/javase/downloads/index.html*](http://www.oracle.com/technetwork/java/javase/downloads/index.html) để tải về phiên bản Java phù hợp với hệ điều hành. Sau khi download về cài đặt vào máy tính như bình thường.

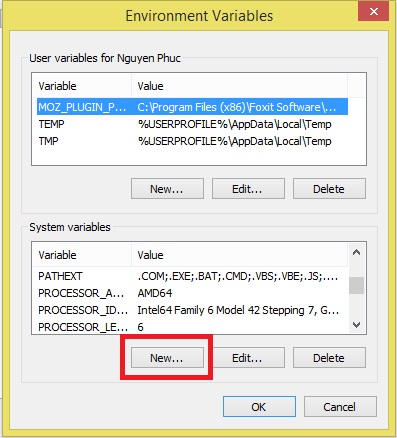
Bây giờ bạn tiếp tục thiết lập môi trường cho JAVA trên máy bạn. Bạn R-Click vào Mycomputer trên Desktop chọn Properties.

Kế đến chọn**Advanced System Setings**:

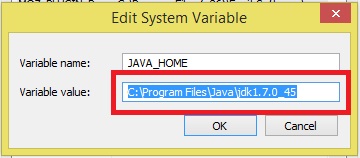
Trong Cửa Số System Properties (sau khi bạn chọn Advanced system settings) chọn Tab Advanced và chọn**Environment Variables….**

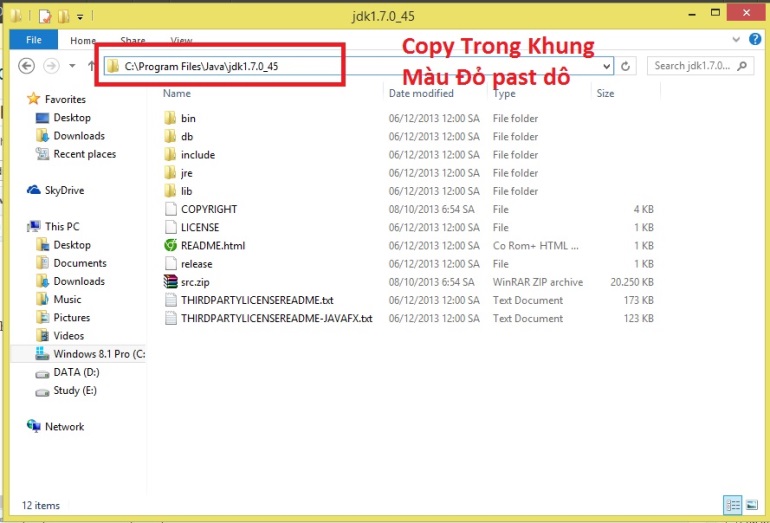
[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled2.jpg)

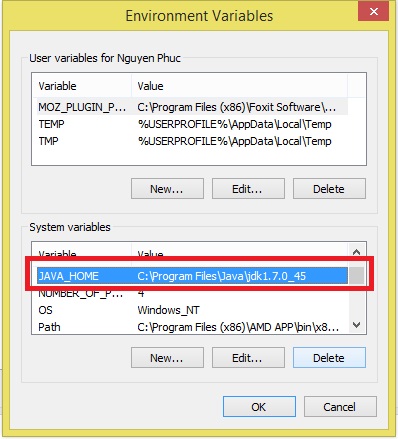
Trong cửa sổ**Environment Variables** chọn **New** như trong hình dưới:

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled3.jpg)

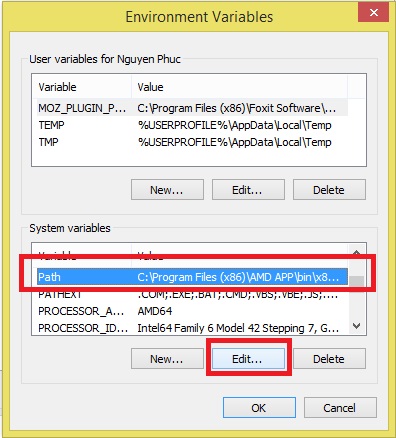
Thiết lập như hình bên dưới: Lưu ý phần trong khung đỏ để chính xác tuyệt đối bạn vào (thư mục cài đặt JAVA bạn thiết lập – ở đây tôi cài mặc định) C:\Program Files\JAVA\….. để xem

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled4.jpg)

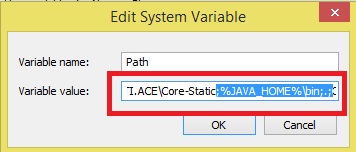
[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled5.jpg)

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled6.jpg)

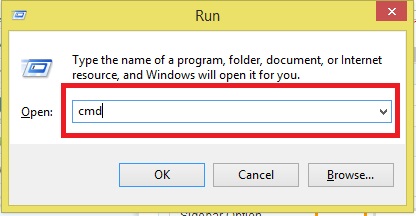
Bây giờ bạn cần tìm đến **Path** trong **System Variables** và chọn edit:

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled7.jpg)

Trong cửa sổ **PATH** bạn **THÊM**vào sau những dòng sau: “ **%JAVA\_HOME%\bin”**

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled8.jpg)

Mở của sổ cmd để kiểm tra hoạt động của Java.

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled9.jpg)

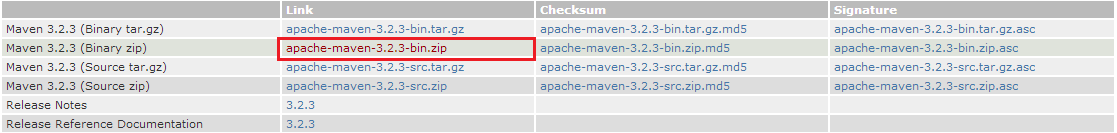
Của sổ CMD xuất hiện. Nhập lệnh “ **Java -version**” để kiểm tra

[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled10.jpg)

* Như vậy Java đã được cài đặt và hoạt động tốt.

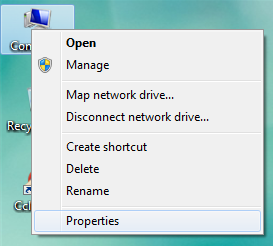
## PHỤ LỤC: CÀI ĐẶT VÀ THIẾT LẬP MAVEN

Tải về phiên bản (phù hợp với hệ điều hành) maven tại địa chỉ: [*http://maven.apache.org/download.cgi*](http://maven.apache.org/download.cgi)*,* chọn bản portable (.zip) để tiện sử dụng:

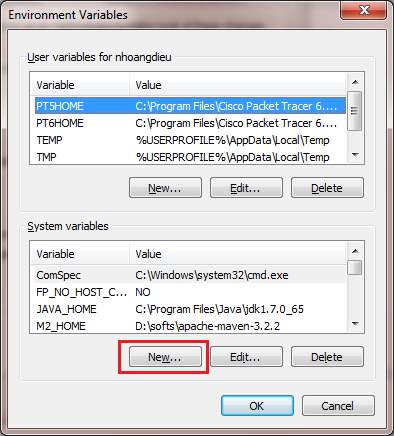


Sau khi tải về giải nén và tiến hành cài đặt biến môi trường:

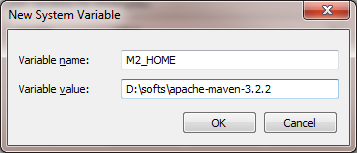
Chuột phải vào biểu tượng MyComputer 🡪 chọn properties,



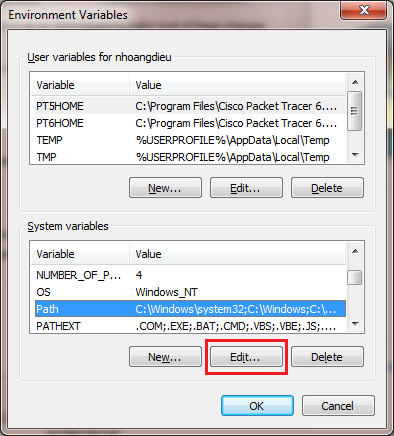
Tiếp theo chọn: Advance system settings 🡪 Enviroment variables, giao diện biến môi trường hiển thị. Chọn new:



Nhập vào giá trị M2\_HOME và đường dẫn tới thư mục cài đặt maven, chọn OK:



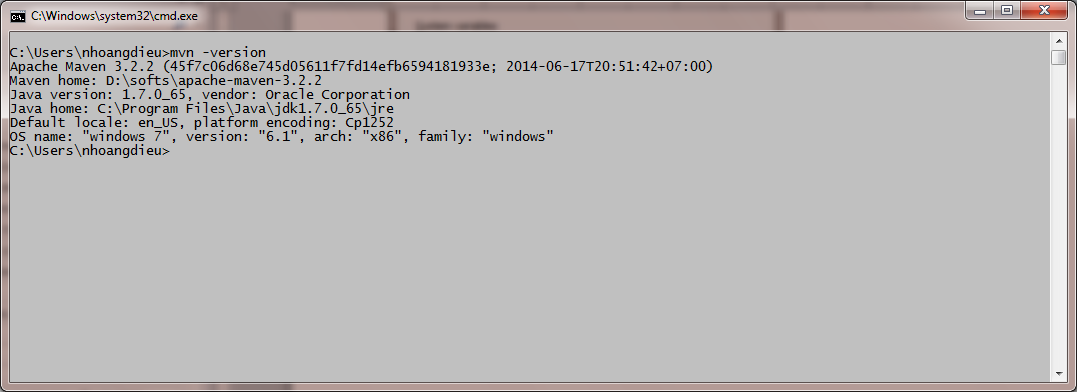
Tiếp theo, tìm đến path và thêm đường dẫn đến thư mục bin của maven:



Thêm vào phần cuối, %M2\_HOME%\bin, chọn OK.

* Lưu ý, khi cài đặt maven thì bắt buộc phải cài đặt và cấu hình các biến môi trường cần thiết của Java trước, tham khảo mục lục C.

Các bước cài đặt coi như hoàn tất, để kiểm tra hoạt động của maven, mở của sổ cmd thực thi câu lệnh: *mvn –version.*



* Như vậy maven đã được cài đặt và hoạt động tốt.
* Để sử dụng maven có thể tham khảo phần phụ lục F: Maven Command

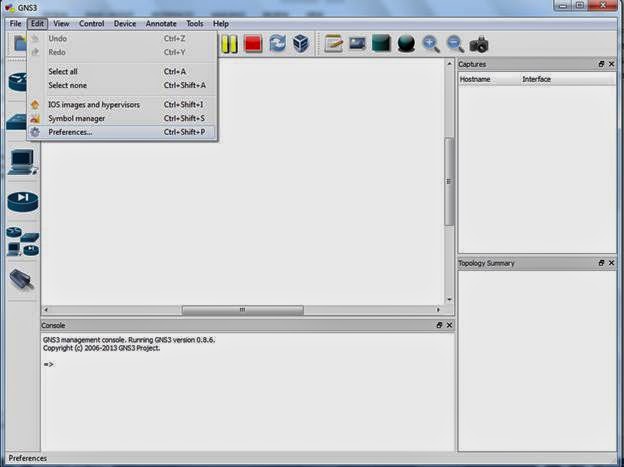
## PHỤ LỤC: THIẾT LẬP MÔI TRƯỜNG ẢO TRÊN GNS3

### CÀI ĐẶT GNS3

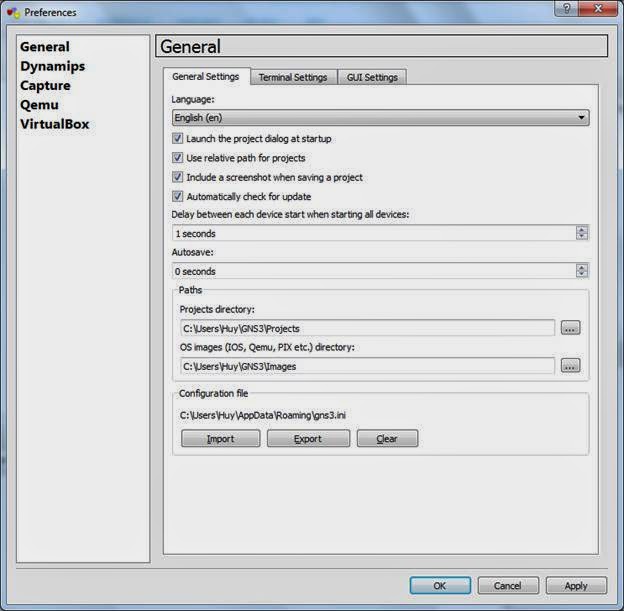
Download phiên bản mới nhât của GNS3 tại: [*http://www.gns3.net/download*](http://www.gns3.net/download)*.* Sau khi cài download phần mềm GNS3 tiến hành cài đặt như bình thường:

Chỉnh sửa sau khi cài đặ:

Sau khi cài đặt GNS3, phần quan trọng đầu tiên là kiểm tra dynamip có hoạt động hay không. Vào Edit 🡪 Preferences…



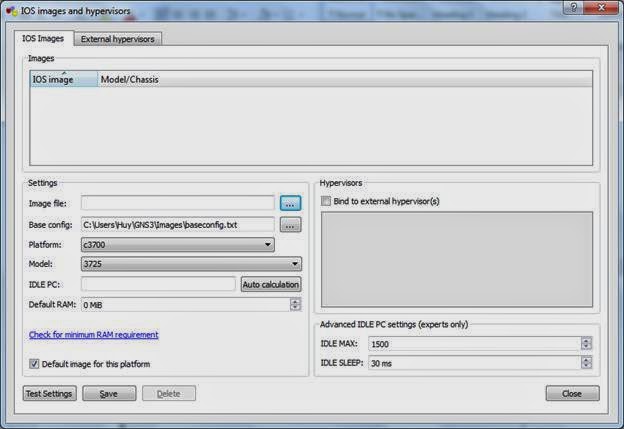
Chọn phần Dynamips bên cột trái.

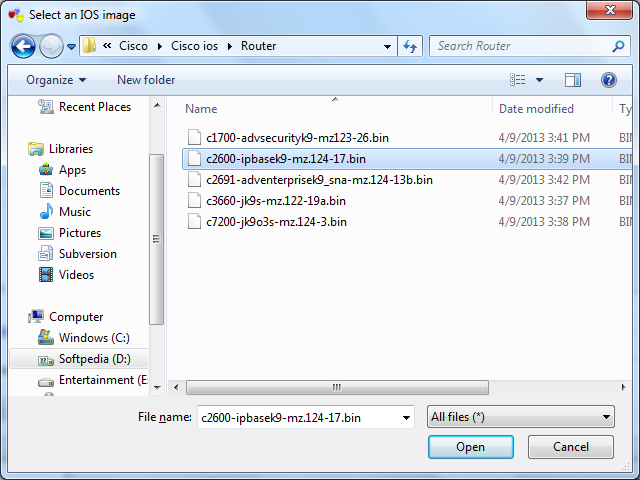


Chỉnh sửa các thông số bấm nút Test Settings để kiểm tra hoạt động, nếu chữ màu đỏ và báo fail thì phải reset máy và cài lại, với các bản GNS3 cũ khi cài lên Windows 8 thì sẽ không cài được WinPCap khi test sẽ lỗi, phải tìm các bản WinPcap cho Windows 8 rồi cài lại.

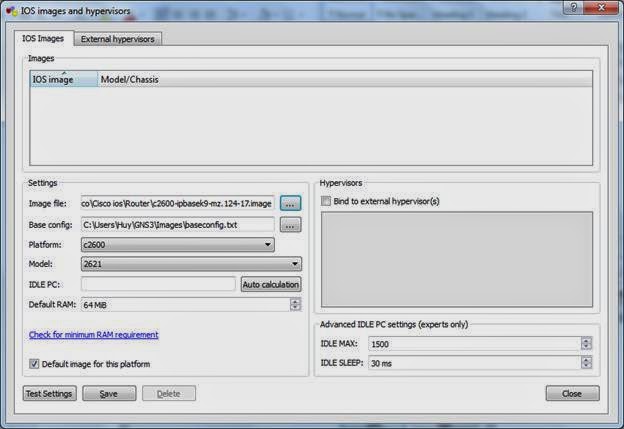
Để cấu hình IOS cho các Router chọn Edit 🡪 IOS image…

Browwse tới file IOS mà download về:





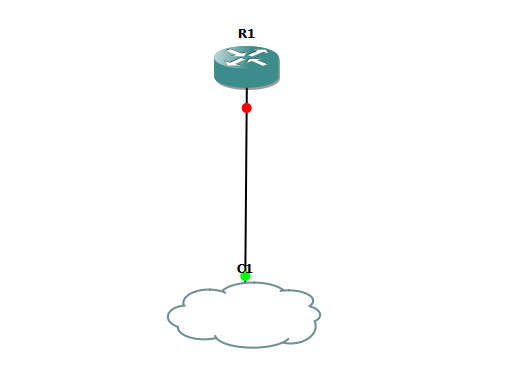
Sau khi chọn thì GNS3 sẽ tự động nhận diện đó là IOS của model router nào.



Bấm save, sau khi có IOS thì mục chọn router sẽ enable router theo model đã chọn. Thực hiện kéo thả router ra hình trống để thí nghiệm.

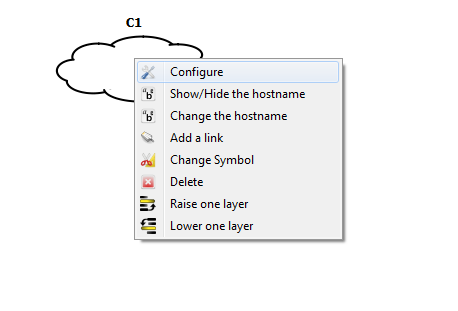
### CẤU HÌNH KẾT NỐI VỚI THIẾT BỊ THẬT

Để có thể kết nối router ảo với các thiết bị thật thì các router cần được kết nối với cloud.

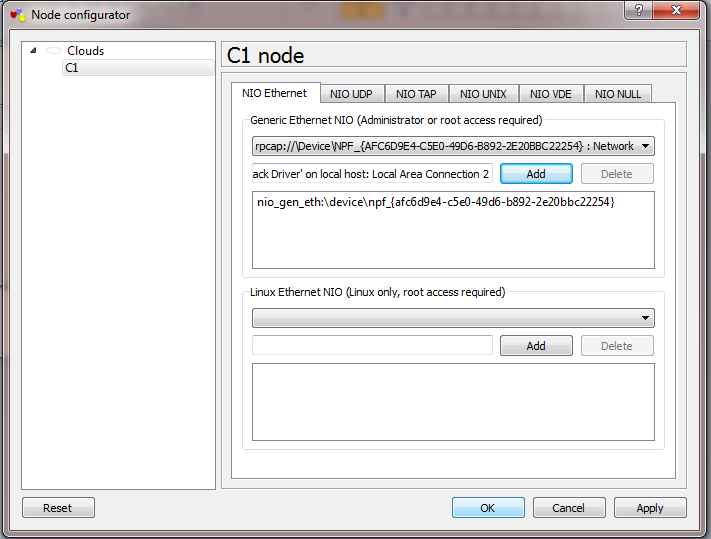


**Cấu hình cloud:**

Để cấu hình cloud kết nối vào thiết bị thật, chuột phải vào cloud 🡪 configure



Ở tab NIO Ethernet, chọn card mạng muốn kết nối (ở đây tôi tạo card mạng ảo tên “Local Area Connection 2”) để thí nghiệm. Bấm vào nút add và click ok để hoàn thành.



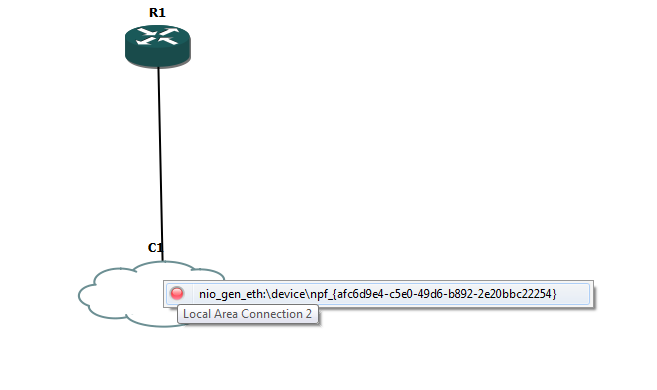
Lưu ý: ở bước này có một số lỗi phát sinh:

* Nếu không run GNS3 với quyền admin thì trong tab NIO Ethernet sẽ không thấy danh sách card mạng để add, phải thoát ứng dụng và run với quyền admin để tiếp tục.
* Nếu sử dụng card mạng ảo được tạo ra sau khi cài đặt GNS3 thì trong danh sách card mạng sẽ không thấy card ảo, phải reset lại máy hoặc nếu không được phải gỡ GNS3 và cài lại sau khi cài card mạng ảo.

**Cấu hình router kết nối vào cloud:**

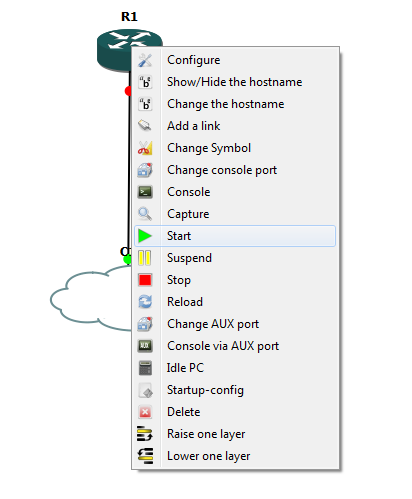
Router muốn kết nối được với mạng thật phải kết nối thông qua cloud, dùng link ethernet kết nối vào cloud.

Trong giới hạn đề tài, card mạng được nối vào cloud là card mạng ảo có địa chỉ ip là 192.168.100.1/24 và các ip router kết nối với cloud được đặt theo thứ tự +10.

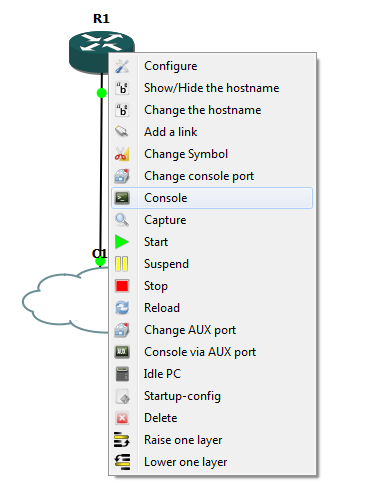


Chọn vào card mạng mà muốn kết nối nếu lúc cấu hình cloud có nhiều card mạng được add.

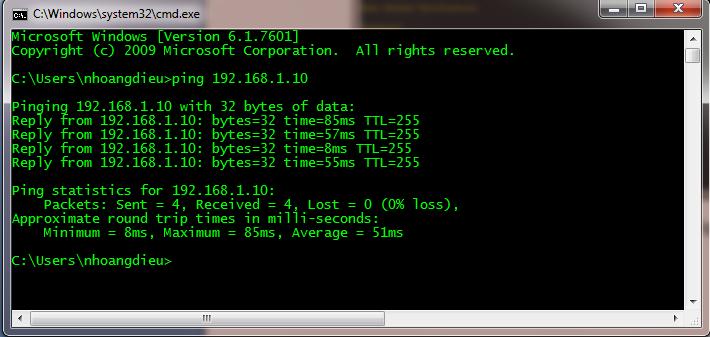
Start router và tiến hành cấu hình port cổng ethernet vừa kết nối.



Tiến hành mở console router và cấu hình ip(cùng nhánh mạng với card mạng được add vào cloud) port đang được kết nối với cloud cũng như mở port lên.



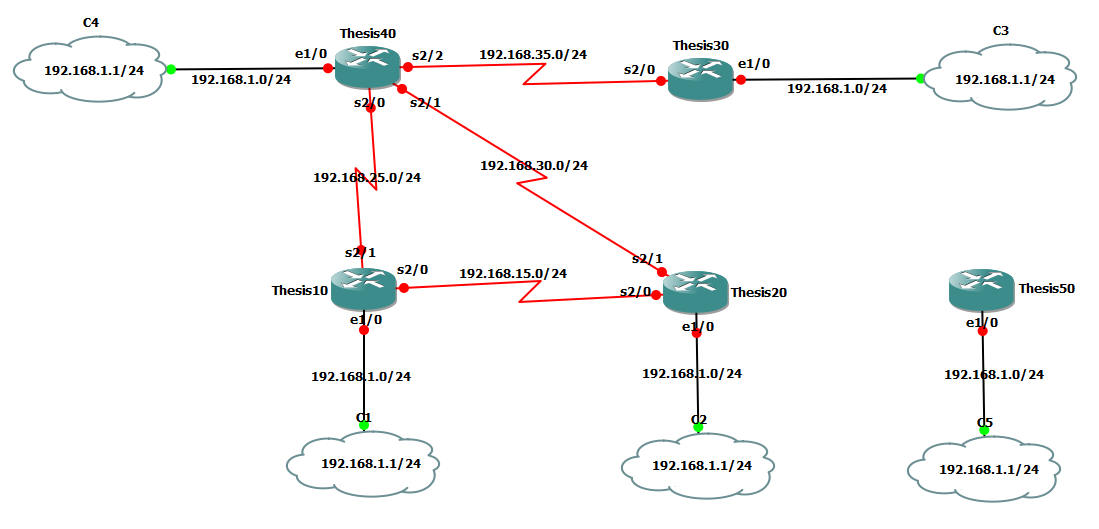
Bây giờ có thể dùng cmd của Windows ping vào ip router vừa đặt để kiểm tra kết nối router ra thiết bị thật được chưa.



Router ảo đã kết nối ra thiết bị thật, nhưng muốn có thể telnet hay ssh vào router thì cần cấu hình enable các chức năng trên của router.

### XÂY DỰNG MÔ HÌNH THỬ NGHIỆM

Mô hình ảo ban đầu cơ bản như sau:



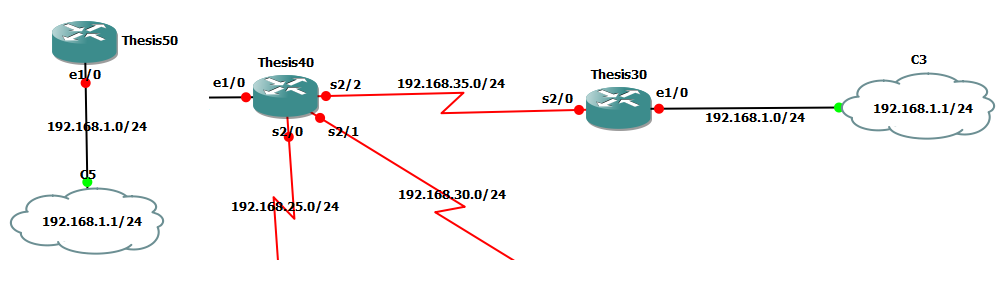
Mô hình gồm 5 router C7200 cisco để thử nghiệm:

Tất cả 5 router đều được kết nối với cloud để kết nối ra card mạng “Local Area Connection 2” của máy thật dùng cho việc telnet/ssh hoặc sử dụng snmp.

Card mạng “Local Area Connection 2” có địa chỉ ip là: 192.168.100.1/24

Các router kết nối vào cloud với cổng ethernet1/0(e1/0) và có địa chỉ ip cùng nhánh (192.168.100.0/24) với card mạng trên máy thật, ip ethernet1/0 và các port khác của các router được đặt theo quy tắc để tiện việc quản lý và thử nghiệm (xem phần giải thích mô hình bên dưới).

Giải thích mô hình:



* 192.168.100.1/24: ip của card mạng trên máy thật.
* 192.168.100.0/24: ip nhánh mạng được kết nối giữa router và cloud (phải cùng nhánh mạng với ip của card mạng trên máy thật.
* Các router được đặt tên ThesisXX với XX là số thứ tự (+10) và được bắt đầu từ 10.
* Nhánh mạng giữa 2 router đựa đặt dựa vào việc kết nối giữa 2 router nào với nhau theo quy tắc: 192.168.xxx.0/24 với x là trung bình cộng giữa số thứ tự tên router. Ví dụ nhánh mạng giữa Thesis40 và Thesis 30 sẽ có ip là 192.168.35.0/24.
* Các port serial kết nối giữa 2 router được đánh ip giống với quy tắc đánh ip của E1/0.
* Với quy tắc đánh ip như trên, đảm bảo các router được đánh ip phù hợp và hoạt động được cũng như không có sự trùng lặp ip và tiện trong việc thử nghiệm chương trình.

## PHỤ LỤC: MAVEN COMMAND

Maven hỗ trợ khá nhiều câu lệnh tùy vào mục đích sử dụng:

**Mvn compile:**

D:\nhoangdieu\>mvn compile

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building laptrinh.vn 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] Nothing to compile - all classes are up to date

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.738s

[INFO] Finished at: Sat Aug 23 11:49:49 ICT 2014

[INFO] Final Memory: 4M/15M

[INFO] ------------------------------------------------------------------------

Lần đầu tiên thực thi command này, maven sẽ download toàn bộ các pluins và các dependencies liên quan. Nếu bạn thực thi lại lệnh này, maven sẽ không download lại những gì nó đã có từ lần trước, vậy nên command sẽ được thực thi nhanh hơn. Các class được compile sẽ được đưa vào ${basedir}/target/classes. Với maven vẫn còn vài lỗi phát sinh, nếu trong quá trình download các gói thư viện được định nghĩa trong pom.xml nhưng phát sinh lỗi do sự cố mạng hay 1 lỗi nào đó mà gói không được tải trọn vẹn, thì ở lần build sau khi maven tìm thấy gói trong repo trên máy lúc ấy maven sẽ dùng gói đó hỗ trợ build và việc build sẽ faild, trong trường hợp này cần phải tìm đến gói đó và xóa nó đi, thực hiện build lại nếu trong quá trình download lại gói đó không phát sinh lỗi thì build sẽ thành công.

**mvn test**

Khi có một số unit test được định nghĩa, để compile và thực thi các test case ấy. Trong quá trình thực thi nếu các thư viện còn thiếu maven sẽ download về và thực hiện compile project và run các test case.

D:\nhoangdieu>mvn test

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building laptrinh.vn 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO]

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running vn.laptrinh.AppDemoTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.073 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 4.300s

[INFO] Finished at: Sat Aug 23 11:59:14 ICT 2014

[INFO] Final Memory: 6M/15M

[INFO] ------------------------------------------------------------------------

**mvn test-compile**

Dùng để test compile code nhưng không muốn thực thi code:

D:\nhoangdieu>mvn test-compile

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building laptrinh.vn 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 0.617s

[INFO] Finished at: Sat Aug 23 12:00:09 ICT 2014

[INFO] Final Memory: 4M/15M

[INFO] -----------------------------------------------------------------------

**mvn package**

Được dùng khi muốn export project ra 1 gói jar để cho người khác sử dụng. Được định nghĩa trong pom.xml, nếu được định nghĩa jar thì khi export ra gói .jar, nếu không mặc định là .war (web app).

D:\nhoangdieu>mvn package

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building laptrinh.vn 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ laptrinh.v

-------------------------------------------------------

T E S T S

-------------------------------------------------------

Running vn.laptrinh.AppDemoTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.038 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 8.595s

[INFO] Finished at: Sat Aug 23 12:00:33 ICT 2014

[INFO] Final Memory: 6M/16M

[INFO] ------------------------------------------------------------------------

**mvn install**

Với command “mvn package” gói jar chỉ được export ra ${basedir}/target, nếu muốn gói jar ấy được đưa lên repo và các project các có thể dependency tới thì cần sử dụng lệnh “mvn install”.

C:\nhoangdieu>mvn install

[INFO] Scanning for projects...

[INFO]

[INFO] ------------------------------------------------------------------------

[INFO] Building laptrinh.vn 0.0.1-SNAPSHOT

[INFO] ------------------------------------------------------------------------

[INFO]

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 1.640s

[INFO] Finished at: Sat Aug 23 12:01:15 ICT 2014

[INFO] Final Memory: 5M/15M

[INFO] ------------------------------------------------------------------------

**mvn clean**

Dùng để xóa thư mục target, xóa các gói jar củ.

**mvn clean install**

Thực hiện xóa thư mục target và build lại.

**mvn site**

Tạo 1 document site cho project, hiển thị các thông tin liên quan đến project (ví dụ như các dependency, các plugin được sử dụng – khai báo trong pom)

**mvn eclipse:eclipse**

Để tạo project với eclipse. Còn 1 số tham command tương tự như:

*mvn eclipse:clean* – để clean project