**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

**…………….🙚🕮🙘…………….**

Cần Thơ, ngày tháng năm 2015

Giảng viên hướng dẫn

Ts.

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

**…………….🙚🕮🙘…………….**

Cần Thơ, ngày 3 tháng 12 năm 2014

Giảng viên phản biện

**NHẬN XÉT CỦA GIẢNG VIÊN PHẢN BIỆN**

**…………….🙚🕮🙘…………….**

Cần Thơ, ngày tháng năm 2015

Giảng viên phản biện

# LỜI CẢM ƠN

**…………….🙚🕮🙘…………….**

Trong suốt quá trình thực hiện đề tài luận văn tốt nghiệp “” em luôn nhận được sự giúp đỡ quý báo từ thầy cô, bạn bè và các anh chị để em có thể hoàn thành đề tài của mình.

Trước tiên, em xin gửi lời cảm ơn chân thành đến thầy … đã tận tình hướng dẫn và chỉ ra con đường tốt để em có thể được làm luận văn và đồng thời học hỏi 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.

Mặc dù đã cố gắng hoàn thành luận văn trong phạm vi và khả năng cho phép nhưng chắc chắn sẽ không tránh khỏi những thiếu sót. Em kính mong nhận được sự thông cảm và tận tình chỉ bảo của quý thầy cô và các bạn.

Cần Thơ, ngày tháng năm 2015

Sinh viên thực hiện

# LỜI NÓI ĐẦU

**…………….🙚🕮🙘…………….**

# MỤC LỤC

**…………….🙚🕮🙘…………….**

# KÍ HIỆU VÀ VIẾT TẮT

**…………….🙚🕮🙘…………….**

|  |  |
| --- | --- |
| **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

**…………….🙚🕮🙘…………….**

# TÓM TẮT

**…………….🙚🕮🙘…………….**

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, hoặc Server (sau đây xin gọi chung là Server – Máy chủ) với giá chỉ từ 5$ một tháng. Và với người quản trị, hoặc một webmaster đã và đang sử dụng nhiều dịch vụ máy chủ khác nhau tại nhiều nhà cung cấp khác nhau, thì cần có một dịch vụ để có thể tập trung để điều khiển, quản lý và theo dõi tình trạng hoạt động của những Server là rất cần thiết. Hệ thống phải luôn theo dõi tình hình Server và đưa ra các cảnh báo cần thiết để người quản lý kịp thời khắc phục các sự cố. Hiện tại cũng có không ít phần mềm, dịch vụ có thể điều khiển và theo dõi Server, nhưng đa phần dừng lại ở mức phải cài đặt vào trong chính Server đó hoặc của mỗi nhà cung cấp riêng biệt,và là các dịch vụ của nước ngoài. Để người quản lý có thể nhanh chóng cấu hình, phát hiện sự cố, để khắc phục nhanh chóng, mọi lúc, mọi nơi thì một dịch vụ web quản lý tập trung các Server là cần thiết.

Trong phạm vi luận văn, hệ thống lấy ý tưởng từ công cụ Webmin,công vụ quản lý Server trên web. Và áp dụng các công nghệ mới, cùng với các framework hỗ trợ như: spring, maven, … Sử dụng Spring để làm ứng dụng web. Xây dựng server service cung cấp công cụ theo dõi hệ thống khách. Phía server sử dụng ssh làm con đường chính để thực hiện thao tác cấu hình remote đến Server. Sử dụng Google Cloud Platform và DigitalOcean Cloud để thực hiện mở các VPS chạy demo.

# PHẦN MỞ ĐẦU

### ĐẶT VẤN ĐỀ

Ngày nay sự ra đời của Công nghệ điện toán đám mây – Cloud Computing. Và các dịch vụ Cloud VPS, Cloud Server ngày càng phổ biến.Chính vì số lượng thiết bị ngày càng nhiều nên việc quản lý và cấu hình các thiết bị là công việc khó khăn đòi hỏi người quản trị phải am hiểu về mạng máy tính và nhạy bén trong việc giải quyết sự cố.

Sẽ thật sự rất khó khăn và tốn một thời gian khá lâu nếu một trong số hàng nghìn thiết bị bị sự cố nếu không có một hệ thống trợ giúp. Một hệ thống quản lý số lượng lớn các thiết bị, đưa ra cảnh báo khi thiết bị lỗi, cho phép cấu hình từ cơ bản để những cấu hình năng cao, phức tạp. Hơn nữa còn có khả năng đáp ứng mọi lúc, mọi nơi là thật sự cần thiết.

* Hiện tại có một số sản phẩm thương mại hiện hữu có hỗ trợ các tác vụ quản lý thiết bị mạng. Tuy nhiên, để hiểu được phương thức hoạt động, công nghệ hiện hữu, khai thác và làm chủ được công nghệ… cần phải xây dựng một hệ thống quản lý thiết bị mạng dựa trên các công nghệ đó.

### NHỮNG NGHIÊN CỨU LIÊN QUAN

### MỤC TIÊU ĐỀ TÀI

**Mục đích tổng quát**: Vận đụng các giải pháp hiện hữu để có thể làm chủ việc quản lý thiết bị mạng.

**Mục tiêu cụ thể trong đề tài này**: Xây dựng hệ thống quản lý thiết bị mạng theo mô hình client – server:

* Hỗ trợ lấy và hiển thị thông tin thiết bị theo biểu mẫu.
* Cấu hình thay đổi thông tin trên thiết bị.
* Lắng nghe và hiển thị thông báo từ thiết bị.
* Tiện dụng (hướng phát triển).

### ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu trong luận văn này phần mềm gns3 tạo môi trường router ảo để thực hiện các cấu hình thông thường:

* Cấu hình interface: mở/tắt port, cấu hình ip
* Cấu hình vạch đường cho thiết bị
* Quản lý các tcp connection
* Cấu hình định thời tác vụ cho server

### NỘI DUNG NGHIÊN CỨU

* Nghiên cứu cấu hình spring jmx, đăng ký mbean với hệ điều hành.
* Nghiên cứu thread pool trong java, cho phép thực thi hàm một cách song song
* Nghiên cứu việc gởi gói tin snmp bằng java thông qua snmp4j framework
* Nghiên cứu tạo ssh connection bằng java thông qua jsch framework
* Tìm hiểu và áp dụng maven vào việc đóng gói project java
* Nghiên cứu cách tạo và đăng ký service với java
* Tìm hiểu cơ bản về cách sử dụng giao thức SNMP để thao tác trên thiết bị

### BỐ CỤC LUẬN VĂN

Bố cục luận văn được trình bày gồm 4 chương:

* Chương 1: Tổng quan
* Chương 2: Cơ sở lý thuyết
* Chương 3: Sơ đồ chức năng
* Chương 4: Kết quả đạt được

# CHƯƠNG 1: TỔNG QUAN

## SỰ PHÁT TRIỂN MẠNG MÁY TÍNH

Vào giữa những năm 1970, các thiết bị đầu cuối sử dụng những phương pháp liên kết qua đường cáp nằm trong một khu vực đã được ra đời. Với những ưu điểm từ nâng cao tốc độ truyền dữ liệu và qua đó kết hợp được khả năng tính toán của các máy tính lại với nhau. Để thực hiện việc nâng cao khả năng tính toán với nhiều máy tính các nhà sản xuất bắt đầu xây dựng các mạng phức tạp.

Vào những năm 1980 các hệ thống đường truyền tốc độ cao đã được thiết lập ở Bắc Mỹ và Châu Âu và từ đó cũng xuất hiện các nhà cung cấp các dịnh vụ truyền thông với những đường truyền có tốc độ cao hơn nhiều lần so với đường dây điện thoại. Với những chi phí thuê bao chấp nhận được, người ta có thể sử dụng được các đường truyền này để liên kết máy tính lại với nhau và bắt đầu hình thành các mạng một cách rộng khắp. Ở đây các nhà cung cấp dịch vụ đã xây dựng những đường truyền dữ liệu liên kết giữa các thành phố và khu vực với nhau và sau đó cung cấp các dịch vụ truyền dữ liệu cho những người xây dựng mạng. Người xây dựng mạng lúc này sẽ không cần xây dựng lại đường truyền của mình mà chỉ cần sử dụng một phần các năng lực truyền thông của các nhà cung cấp.

Từ đó đến nay đã có rất nhiều công ty đưa ra các sản phẩm của mình, đặc biệt khi các máy tính cá nhân được sử dụng một cánh rộng rải. Khi số lượng máy vi tính trong một văn phòng hay cơ quan được tăng lên nhanh chóng thì việc kết nối chúng trở nên vô cùng cần thiết và sẽ mang lại nhiều hiệu quả cho người sử dụng.

Ngày nay với một lượng lớn về thông tin, nhu cầu xử lý thông tin ngày càng cao. Mạng máy tính hiện nay trở nên quá quen thuộc đối với chúng ta, trong mọi lĩnh vực như khoa học, quân sự, quốc phòng, thương mại, dịch vụ, giáo dục... và ở nhiều nơi mạng đã trở thành một nhu cầu không thể thiếu được.

Hiện nay việc làm sao có được một hệ thống mạng chạy thật tốt, thật an toàn với lợi ích kinh tế cao đang rất được quan tâm. Một vấn đề đặt ra có rất nhiều giải pháp về công nghệ, một giải pháp có rất nhiều yếu tố cấu thành, trong mỗi yếu tố có nhiều cách lựa chọn. Như vậy để đưa ra một giải pháp hoàn chỉnh, phù hợp thì phải trải qua một quá trình chọn lọc dựa trên những ưu điểm của từng yếu tố, từng chi tiết rất nhỏ.

## GIAO THỨC MẠNG

### Giao thức ssh

#### Định nghĩa:

SSH (tiếng Anh: Secure Shell) là một giao thức mạng dùng để thiết lập kết nối mạng một cách bảo mật. SSH hoạt động ở lớp trên trong mô hình phân lớp TCP/IP. Các công cụ SSH (như là OpenSSH, PuTTy,…) cung cấp cho người dùng cách thức để thiết lập kết nối mạng được mã hoá để tạo một kênh kết nối riêng tư. Hơn nữa tính năng tunneling (hoặc còn gọi là port forwarding) của các công cụ này cho phép chuyển tải các giao vận theo các giao thức khác. Do vậy có thể thấy khi xây dựng một hệ thống mạng dựa trên SSH, chúng ta sẽ có một hệ thống mạng riêng ảo VPN đơn giản. Mỗi khi dữ liệu được gửi bởi một máy tính vào mạng, SSH tự động mã hoá nó. Khi dữ liệu được nhận vào, SSH tự động giải mã nó. Kết quả là việc mã hoá được thực hiện trong suốt, người dùng có thể làm việc bình thường, không biết rằng việc truyền thông của họ đã được mã hoá an toàn trên mạng.

#### Lịch sử phát triển:

SSH1 và giao thức SSH-1 được trình bày năm 1995 bởi Tatu Ylõnen, một nhà nghiên cứu ở trường đại học kĩ thuật Helsinki của Phần Lan. Sau khi mạng trường đại học của ông ta là nạn nhân của một cuộc tấn công đánh cắp password vào đầu năm đó. Tháng 7 năm 1995, SSH1 được phát hành rộng rải dưới dạng một phần mềm miễn phí có source code, cho phép mọi người sao chép và sử dụng mà không thu phí.

Năm 1998, SCS phát hành sản phẩm phần mềm “SSH Secure Shell” (SSH2), dựa trên giao thức SSH-2. Tuy nhiên, SSH2 không thay thế SSH1 trong một số lĩnh vực, có 2 lí do:

* Thứ nhất, SSH2 không có một số tiện ích, các đặc điểm có ích và cấu hình tuỳ chọn như SSH1.
* Thứ hai, SSH2 có nhiều giới hạn về việc đăng kí. Bản chính SSH1 đã có sẵn miễn phí từ Ylõnen và trường đại học kĩ thuật Helsinki. Phiên bản mới hơn của SSH1 từ SCS vẫn có sẵn miễn phí cho hầu hết người dùng, thậm chí cả cấu hình thương mại cũng miễn phí chỉ cần phần mềm đó không được trực tiếp bán cho việc thu lợi nhuận hoặc được tặng như là một dịch vụ cho khách hàng.

Vì thế, tuy SSH2 đã xuất hiện, nhưng hầu hết những người đang sử dụng SSH1 đều nhận ra vài ưu điểm của SSH1 so với SSH2 và tiếp tục sử dụng SSH1, ba năm sau khi SSH2 ra đời thì SSH1 vẫn là phiên bản được sử dụng phổ biến trên Internet và vượt qua cả SSH2 là giao thức tốt hơn và bảo mật hơn. Tuy nhiên, SSH2 cũng có hai sự phát triển hưa hẹn, đó là một bản nới lỏng của SSH2 bản quyền và sự xuất hiện SSH-2 bổ sung.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## NGÔN NGỮ LẬP TRÌNH – JAVA

### Giới thiệu

Java là một [ngôn ngữ lập trình](http://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh) dạng [lập trình hướng đối tượng](http://vi.wikipedia.org/wiki/L%E1%BA%ADp_tr%C3%ACnh_h%C6%B0%E1%BB%9Bng_%C4%91%E1%BB%91i_t%C6%B0%E1%BB%A3ng) (OOP). Khác với phần lớn ngôn ngữ lập trình thông thường, thay vì [biên dịch](http://vi.wikipedia.org/wiki/Tr%C3%ACnh_bi%C3%AAn_d%E1%BB%8Bch) [mã nguồn](http://vi.wikipedia.org/wiki/M%C3%A3_ngu%E1%BB%93n) thành [mã máy](http://vi.wikipedia.org/wiki/Ng%C3%B4n_ng%E1%BB%AF_m%C3%A1y) hoặc [thông dịch](http://vi.wikipedia.org/wiki/Tr%C3%ACnh_th%C3%B4ng_d%E1%BB%8Bch) mã nguồn khi chạy, Java được thiết kế để biên dịch mã nguồn thành [bytecode](http://vi.wikipedia.org/w/index.php?title=Bytecode&action=edit&redlink=1), bytecode sau đó sẽ được môi trường thực thi (runtime environment) chạy.

Trước đây, Java chạy chậm hơn những ngôn ngữ dịch thẳng ra mã máy như C và [C++](http://vi.wikipedia.org/wiki/C%2B%2B), nhưng sau này nhờ công nghệ "biên dịch tại chỗ" - Just in time compilation, khoảng cách này đã được thu hẹp, và trong một số trường hợp đặc biệt Java có thể chạy nhanh hơn. Java chạy nhanh hơn những ngôn ngữ thông dịch như [Python](http://vi.wikipedia.org/wiki/Python), [Perl](http://vi.wikipedia.org/wiki/Perl), [PHP](http://vi.wikipedia.org/wiki/PHP) gấp nhiều lần. Java chạy tương đương so với [C#](http://vi.wikipedia.org/wiki/C), một ngôn ngữ khá tương đồng về mặt cú pháp và quá trình dịch/chạy.

[Cú pháp](http://vi.wikipedia.org/wiki/C%C3%BA_ph%C3%A1p_h%E1%BB%8Dc) Java được vay mượn nhiều từ [C](http://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)) & [C++](http://vi.wikipedia.org/wiki/C%2B%2B) nhưng có cú pháp hướng đối tượng đơn giản hơn và ít tính năng xử lý cấp thấp hơn. Do đó việc viết một chương trình bằng Java dễ hơn, đơn giản hơn, đỡ tốn công sửa lỗi hơn. Lập trình C rất hay xảy ra lỗi và khó sửa.

Trong Java, hiện tượng rò rỉ bộ nhớ hầu như không xảy ra do bộ nhớ được quản lí bởi Java Virtual Machine (JVM) bằng cách tự động "dọn dẹp rác" - Garbage collection. Người lập trình không phải quan tâm đến việc cấp phát và xóa bộ nhớ như C, C++. Tuy nhiên khi sử dụng những tài nguyên mạng, file IO, database (nằm ngoài kiểm soát của JVM) mà người lập trình không đóng (close) các streams thì rò rỉ dữ liệu vẫn có thể xảy ra.

### Lịch sử phát triển

Java được khởi đầu bởi [James - Gosling](http://vi.wikipedia.org/wiki/James_Gosling) và bạn đồng nghiệp ở [Sun Microsystems](http://vi.wikipedia.org/wiki/Sun_Microsystems) năm [1991](http://vi.wikipedia.org/wiki/1991). Ban đầu ngôn ngữ này được gọi là Oak (có nghĩa là [cây sồi](http://vi.wikipedia.org/w/index.php?title=C%C3%A2y_s%E1%BB%93i&action=edit&redlink=1)) do bên ngoài cơ quan của ông Gosling có trồng nhiều loại cây này), họ dự định ngôn ngữ đó thay cho [C++](http://vi.wikipedia.org/wiki/C%2B%2B), nhưng các tính năng giống [Objective C](http://vi.wikipedia.org/w/index.php?title=Objective_C&action=edit&redlink=1). Không nên lẫn lộn Java với [JavaScript](http://vi.wikipedia.org/wiki/JavaScript), hai ngôn ngữ đó chỉ giống tên và loại [cú pháp](http://vi.wikipedia.org/wiki/C%C3%BA_ph%C3%A1p_h%E1%BB%8Dc) như [C](http://vi.wikipedia.org/wiki/C_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)). Công ty Sun Microsystems đang giữ bản quyền và phát triển Java thường xuyên. Tháng 04/2011, công ty Sun Microsystems tiếp tục cho ra bản JDK 1.6.24.

Những chi tiết về ngôn ngữ, [máy ảo](http://vi.wikipedia.org/w/index.php?title=M%C3%A1y_%E1%BA%A3o_Java&action=edit&redlink=1) và [API](http://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng) của Java được giữ bởi [Cộng đồng Java](http://vi.wikipedia.org/w/index.php?title=C%E1%BB%99ng_%C4%91%E1%BB%93ng_Java&action=edit&redlink=1) (do [Sun](http://vi.wikipedia.org/w/index.php?title=Sun&action=edit&redlink=1) quản lý). Java được tạo ra vào năm[1991](http://vi.wikipedia.org/wiki/1991) do một số kỹ sư ở [Sun](http://vi.wikipedia.org/wiki/Sun_Microsystems), bao gồm ông James Gosling, một phần của Dự án Xanh (Green Project). Java được phát hành vào năm [1994](http://vi.wikipedia.org/wiki/1994), rồi nó trở nên nổi tiếng khi [Netscape](http://vi.wikipedia.org/wiki/Netscape) tuyên bố tại hội thảo SunWorld năm [1995](http://vi.wikipedia.org/wiki/1995) là trình duyệt [Navigator](http://vi.wikipedia.org/wiki/Netscape_Navigator) của họ sẽ hỗ trợ Java. Về sau Java được hỗ trợ trên hầu hết các trình duyệt như [Internet Explorer](http://vi.wikipedia.org/wiki/Internet_Explorer) ([Microsoft](http://vi.wikipedia.org/wiki/Microsoft)), [Firefox](http://vi.wikipedia.org/wiki/Mozilla_Firefox) ([Mozilla](http://vi.wikipedia.org/wiki/Mozilla)), [Safari](http://vi.wikipedia.org/wiki/Safari) ([Apple](http://vi.wikipedia.org/wiki/Apple_Inc.))... Java được sử dụng chủ yếu trên môi trường NetBeans và Oracle. Sau khi [Oracle](http://vi.wikipedia.org/wiki/Oracle) mua lại [công ty](http://vi.wikipedia.org/wiki/C%C3%B4ng_ty) [Sun Microsystems](http://vi.wikipedia.org/wiki/Sun_Microsystems) năm [2009](http://vi.wikipedia.org/wiki/2009) - [2010](http://vi.wikipedia.org/wiki/2010), [Oracle](http://vi.wikipedia.org/wiki/Oracle) đã mô tả họ là "người quản lý công nghệ Java với cam kết không ngừng để bồi dưỡng một cộng đồng tham gia và minh bạch".

Lịch sử phiên bản java:

* JDK 1.0 (23 tháng 01, 1996)
* JDK 1.1 (19 tháng 2, 1997)
* J2SE 1.2 (Playground) 08 tháng 12, 1998
* J2SE 1.3 (Kestrel) 08 tháng 5, 2000
* J2SE 1.4.0 (Merlin) 06 tháng 02, 2002
* J2SE 5 (1.5.0) (Tiger) 30 tháng 9, 2004
* Java SE 6 (còn gọi là Mustang), được công bố [11 tháng 12](http://vi.wikipedia.org/wiki/11_th%C3%A1ng_12) năm [2006](http://vi.wikipedia.org/wiki/2006).
* Java SE 7 (còn gọi là Dolphin), được bắt đầu từ tháng 8 năm 2006 và công bố ngày 28 tháng 7 năm 2011.
* JDK 8, 18 tháng 3 năm 2014
* Phiên bản dự kiến tiếp theo: Java 9 dự kiến ra đời năm 2016.

### 2.1.4 Cài đặt và thiết lập môi trường Java

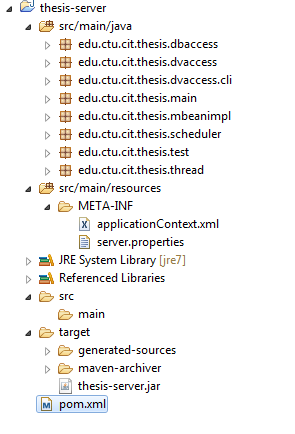
Tham khảo phụ lục C – Cài đặt và thiết lập môi trường Java

## CÔNG CỤ BUILD – MAVEN

### Giới thiệu

Maven là công cụ quản lý và thiết lập tự động 1 dự án phần mềm. Chủ yếu dùng cho các lập trình viên java, nhưng nó cũng có thể được dùng để xây dựng và quản lý các dự án dùng C#, Ruby, Scala hay ngôn ngữ khác. Maven phục vụ mục đích tương tự như [Apache Ant](http://vi.wikipedia.org/w/index.php?title=Apache_Ant&action=edit&redlink=1), nhưng nó dựa trên khái niệm khác và cách hoạt động khác. Maven được quản lý bởi [Apache Software Foundation](http://vi.wikipedia.org/w/index.php?title=Apache_Software_Foundation&action=edit&redlink=1), nó là một phần của dự án [Jakarta Project](http://vi.wikipedia.org/w/index.php?title=Jakarta_Project&action=edit&redlink=1).

Maven dùng khái niệm Project Object Model (POM) để mô tả việc build project, các thành phần phụ thuộc và các module. Nó định nghĩa trước các target cho việc khai báo task, trình biên dịch, đóng gói và thứ tự hoạt động để mọi việc diến ra tốt nhất. Maven tải các thư viện, plug-in từ một hoặc nhiều repositories. Maven cung cấp việc hỗ trợ tải file từ repository trung tâm của Maven và các repositories khác. Cũng có thể upload các sản phẩm lên repository sau khi đã build xong.



Hình 2. - Cấu trúc cơ bản maven project

Maven được xây dựng dựa trên kiến trúc plugin - cho phép sử dụng bất kỳ ứng dụng điều khiển nào thông qua đầu vào tiêu chuẩn. Về mặt lý thuyết, điều này cho phép bất kỳ ai cũng có thể viết các plugin để giao tiếp với các build tool (trình biên dịch, unit test tool, etc) cho bất kỳ ngôn ngữ nào khác. Nhưng trong thực tế, hỗ trợ và sử dụng cho các ngôn ngữ khác đã bị thu hẹp. Hiện tại thì plugin cho .Net đã có và vẫn được duy trì, và một plugin cho [C](http://vi.wikipedia.org/w/index.php?title=C_(programming_language)&action=edit&redlink=1)/[C++](http://vi.wikipedia.org/wiki/C%2B%2B) cũng được duy trì cho Maven.

Một project Maven được cấu hình sử dụng Project Object Model, nó được lưu trong file pom.[xml](http://vi.wikipedia.org/wiki/XML). Một file pom.xml được định nghĩa trong project hình 2.2:



Hình 2. - Cấu trúc cơ bản của pom.xml

### Lợi ích

* **Tự động hóa toàn bộ quá trình release project**:

Khởi tạo, cập nhật thư viện, build & test, release và tạo tài liệu mô tả. Maven hỗ trợ lập trình viên phân chia project ra nhiều thành phần nhỏ để phát triển độc lập và có thể gắn kết chúng lại với nhau để trở thành project hoàn chỉnh một cách dể dàng.

* **Tự động hóa toàn bộ qui trình release là yêu cầu hàng đầu**:

Chỉ cần 1 sự chậm trể, sai lầm ở một công đoạn có thể làm đổ vỡ cả quá trình release của project. Cho nên việc tự động hóa toàn bộ là ưu tiên hàng đầu, nhờ đó là việc lên kế hoạch hiệu quả và chính xác hơn, việc tích hợp và triển khai cũng đơn giản hơn rất nhiều.

* **Tự động hóa quá trình test với các plugin**:

Toàn bộ quá trình test được tự động, từ đó các lỗi được phát hiện sớm hơn. Thậm chí lỗi có thể phát hiện ở giai đoạn phát triển, do các developer đều có thể chạy các test case một cách dễ dàng bằng Maven.

* **Quản lý các dependencies (các thư viện) trong project một các ưu việt**:

Tự động cập nhật, mở rộng dễ dàng, đóng gói (có thể hiểu là phụ thuộc các thư viện bắc cầu). Maven có một repository lớn chứa toàn bộ các thư viện cần thiết (<http://mvnrepository.com>), nó không chỉ lưu trữ gói mà còn lưu cả version, các version này được cập nhật liên tực bởi người phát triển ra nó. Chính vì thế khi cần thay đổi version thì rất nhanh chóng.

* **Phân chia 1 project lớn thành các module nhỏ**:

Từ đó cho phép làm việc đồng thời trên các module khác nhau, đồng thời vẫn tạo được tính thống nhất.

* **Không phụ thuộc vào IDE**:

Cho phép chuyển đổi tử Eclipse sang NetBeans,.. một cách dễ dàng. Maven hỗ trợ một số command để tạo ra project tương thích với IDE muốn sử dụng (ví dụ: mvn eclipse:eclipse – để tạo project với eclipse).

### Maven command

Tham khảo phần mục lục f – Maven commands

### 2.2.4 Cài đặt maven

Tham khảo phần mục lục e - Cài đặt và thiết lập môi trường Maven

## SPRING FRAMEWORK

### Tổng quan về Spring framework

Spring Framework, hay ngắn hơn là Spring, là một cấu trúc dùng để xây dựng chương trình ứng dụng mã nguồn mở dành cho [ngôn ngữ lập trình Java](http://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)). Phiên bản đầu tiên của nó do [Rod Johnson](http://vi.wikipedia.org/w/index.php?title=Rod_Johnson&action=edit&redlink=1) viết và đã được ông xuất bản cùng với quyển sách đầu tay [Expert One-on-One Java EE Design and Development](http://www.wrox.com/WileyCDA/WroxTitle/productCd-0764543857.html) (Nhà xuất bản Wrox Press, Tháng 10 năm [2002](http://vi.wikipedia.org/wiki/2002)).

Kiến trúc của Spring Framework được ra mắt công chúng lần đầu tiên hồi tháng 6 năm [2003](http://vi.wikipedia.org/wiki/2003) dưới [Giấy phép Apache - phiên bản 2.0](http://vi.wikipedia.org/wiki/Gi%E1%BA%A5y_ph%C3%A9p_Apache). Phiên bản 1.0 đánh dấu mốc thành đạt đầu tiên được xuất bản vào tháng 3 năm [2004](http://vi.wikipedia.org/wiki/2004) và tiếp đó vào tháng 9 năm 2004, tháng 3 năm 2005.

Tuy Spring Framework không bắt buộc người ta phải tuân theo một [mô hình lập trình](http://vi.wikipedia.org/wiki/M%E1%BA%ABu_h%C3%ACnh_l%E1%BA%ADp_tr%C3%ACnh) cụ thể nào, song nó lan truyền rộng rải trong cộng đồng những người viết chương trình dùng Java, như một hình thức chủ yếu thay thế cho mô hình [Enterprise JavaBean](http://vi.wikipedia.org/w/index.php?title=Enterprise_JavaBean&action=edit&redlink=1). Theo thiết kế, bộ framework này giải phóng lập trình viên dùng Java, cho phép họ nhiều quyền tự do hơn và đồng thời cung cấp một giải pháp tiện lợi, đầy đủ dẫn chứng bằng tài liệu, dễ dàng sử dụng, phù hợp với những thực hành thông dụng trong công nghệ phần mềm. Bên cạnh những đặc trưng nền tảng của Spring Framework là những cái có thể dùng được trong bất cứ một chương trình ứng dụng Java nào, rất nhiều các mở rộng và tiến bộ trong việc kiến tạo các trình ứng dụng dành cho nền tảng mạng web (web-based application) dựa trên [nền Java Enterprise](http://vi.wikipedia.org/wiki/J2EE) cũng tồn tại nữa. Spring Framework nổi tiếng cũng một phần do chính đặc thù kể trên và được giới thương mại công nhận như một nền tảng kiến trúc có tầm quan trọng trong chiến lược kiến tạo phần mềm.

### Lịch sử phát triển

Phần đầu tiên của Spring Framework ban đầu nguyên được [Rod Johnson](http://vi.wikipedia.org/w/index.php?title=Rod_Johnson&action=edit&redlink=1) viết vào năm 2000. Vào năm 2001, những mô hình lập trình cho các trình ứng dụng cho mạng WEB được đại đa số sử dụng đều do [Java Servlet API](http://vi.wikipedia.org/w/index.php?title=Java_Servlet_API&action=edit&redlink=1) và [Enterprise JavaBean](http://vi.wikipedia.org/w/index.php?title=Enterprise_JavaBean&action=edit&redlink=1) cung cấp. Năm 2003 thành lập dự án phát tại Sourceforge để phát triển Spring. Sau khi phát triển trền nền tảng ban đầu hơn một năm họ đã phát hành phiên bản đầu tiên (1.0) vào tháng 3 năm 2004.

Spring Framework đã làm cho những kỹ thuật vốn không được biết đến mấy trở nên những kỹ thuật được mọi người ưa chuộng trong một thời gian ngắn ngủi. Một trong những kỹ thuật nổi tiếng hơn cả là kỹ thuật [đảo ngược quyền khống chế](http://vi.wikipedia.org/w/index.php?title=%C4%90%E1%BA%A3o_ng%C6%B0%E1%BB%A3c_quy%E1%BB%81n_kh%E1%BB%91ng_ch%E1%BA%BF&action=edit&redlink=1) (Inversion of Control, IoC). Nội trong 2004, Spring Framework được rất nhiều người hưởng ứng và làm theo và cũng do tự chính nó cống hiến framework [lập trình định dạng](http://vi.wikipedia.org/w/index.php?title=L%E1%BA%ADp_tr%C3%ACnh_%C4%91%E1%BB%8Bnh_d%E1%BA%A1ng&action=edit&redlink=1) (aspect-oriented programming). Spring Framework đã tạo cơ hội cho mẫu hình lập trình này trở nên được ưa thích trong cộng đồng Java.

Năm 2005 cho thấy mức độ hưởng ứng nâng cao hơn những năm trước, nguyên do cũng vì những xuất bản mới được ra với những cột mốc đáng kể và những tính năng mới được thêm vào. [Diễn đàn Spring](http://forum.springframework.org/) (Spring Forum) ra đời cuối năm 2004 cũng góp phần không nhỏ trong việc nâng cao tính phổ biến của bộ framework và từ đó đến nay đã sinh trưởng trở thành một nguồn thông tin quan trọng, giúp đỡ cho người sử dụng.

Vào tháng 12 năm 2005, hội thảo đầu tiên về Spring Framework đã được tổ chức tại [Miami, Florida](http://vi.wikipedia.org/wiki/Miami,_Florida) thu hút 300 nhà phát triển trong 3 ngày và tiếp theo đó cuộc hội thảo ở [Antwerp](http://vi.wikipedia.org/wiki/Antwerpen) vào tháng 6 năm 2006, thu hút hơn 400 người.

### Các module của Spring

Spring Framework được coi như là một tổ hợp của nhiều framework nhỏ, hay nói cách khác nó là tổ hợp của framework trong framework. Đại bộ phận các framework này được thiết kế để hoạt động biệt lập với các framework khác, song lại tạo nên nhiều chức năng tốt hơn khi được phối hợp làm việc với nhau. Những framework này được phân loại theo các khối nền cơ bản mà những chương trình phần mềm phức tạp điển hình thường sử dụng:

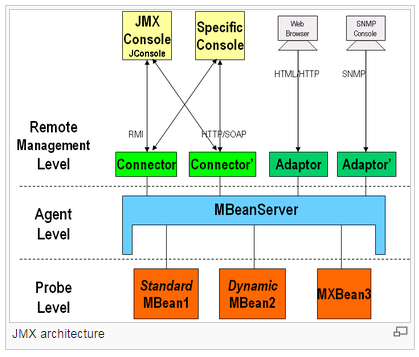
* **Inversion of Control container**: Sự cài đặt các thành phần của chương trình ứng dụng và quản lý chu trình của các đối tượng trong Java (lifecycle management of Java objects).
* **Aspect-oriented programming framework**: Spring AOP module tích hợp chức năng lập trình hướng khía cạnh vào Spring framework thông qua cấu hình của nó. Spring AOP module cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong bất kỳ ứng dụng nào sử dụng Spring.
* **Data access framework**: Tầng JDBC và DAO đưa ra một cây phân cấp exception để quản lý kết nối đến database, điều khiển exception và thông báo lỗi được ném bởi vendor của database. Tầng exception đơn giản điều khiển lỗi và giảm khối lượng code mà chúng ta cần viết như mở và đóng kết nối. Module này cũng cung cấp các dịch vụ quản lý giao dịch cho các đối tượng trong ứng dụng Spring.
* **Transaction processing/Transaction management framework**: Sự đồng hòa các API quản lý giao dịch và sự điều hợp quản lý giao dịch đối với các đối tượng Java dùng phương pháp cài đặt.
* **Model - view - controller framework**: Framework dựa trên nền tảng [HTTP](http://vi.wikipedia.org/wiki/Hypertext_Transfer_Protocol) và [Servlet](http://vi.wikipedia.org/w/index.php?title=Java_Servlet_API&action=edit&redlink=1) cung cấp rất nhiều móc điểm có thể mở rộng và cải biên (extension and customization).
* **Remote Access framework**: Tạo các đối tượng Java trên mạng lưới truyền thông hỗ trợ các giao thức dựa trên nền [HTTP](http://vi.wikipedia.org/wiki/Hypertext_Transfer_Protocol), sử dụng [RPC](http://vi.wikipedia.org/w/index.php?title=RPC&action=edit&redlink=1) (Remote Procedure Call) thông qua cài đặt, như [RMI](http://vi.wikipedia.org/w/index.php?title=RMI&action=edit&redlink=1) (Java remote Method Invocation - khởi động phương pháp từ xa trong Java), [CORBA](http://vi.wikipedia.org/w/index.php?title=CORBA&action=edit&redlink=1) và [dịch vụ mạng](http://vi.wikipedia.org/w/index.php?title=D%E1%BB%8Bch_v%E1%BB%A5_m%E1%BA%A1ng&action=edit&redlink=1) (web services - [SOAP](http://vi.wikipedia.org/wiki/SOAP)).
* **Authentication and authorization framework**: Áp dụng spring security, điều hợp các quá trình xác thực và ủy quyền thông qua cài đặt hỗ trợ nhiều tiêu chuẩn công nghiệp và những tiêu chuẩn được sử dụng phổ biến, cũng như các giao thức, công cụ và các thói quen thực hành phổ dụng.
* **Remote Management framework**: Quản lý tài nguyên ứng dụng thông qua [JMX](http://vi.wikipedia.org/w/index.php?title=JMX&action=edit&redlink=1) (Java Management Extension) cho phép quản lý các thông tin như: heap size, memory, threads...
* **Messaging framework**: Đăng ký các đối tượng lắng nghe thông điệp dùng để tiếp thu và sử dụng thông điệp từ các [hàng thông điệp](http://vi.wikipedia.org/w/index.php?title=H%C3%A0ng_th%C3%B4ng_%C4%91i%E1%BB%87p&action=edit&redlink=1) (message queue) thông qua [JMS](http://vi.wikipedia.org/w/index.php?title=JMS&action=edit&redlink=1) (Java Message Service), cải tiến việc truyền gửi thông điệp dùng tiêu chuẩn trong [API](http://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng) của [JMS](http://vi.wikipedia.org/w/index.php?title=JMS&action=edit&redlink=1).
* **Software testing framework**: Những chủ thể hỗ trợ việc viết các kiểm thử đơn vị và các kiểm thử tích hợp (unit tests and integration tests).

### Spring JMXRMI

#### Giới thiệu

JMX (Java Management Extensions) là một công nghệ Java cung cấp các công cụ để quản lý và giám sát các ứng dụng, các đối tượng hệ thống, thiết bị và mạng lưới dịch vụ chuyên nghiệp.

Spring JMX sử dụng kiến trúc 3 cấp:



Hình 2. - Mô hình phân cấp JMX

* **Probe level**: được gọi là mức cần thiết, mức thấp nhất, bao gồm các mbean cơ bản. Nó chỉ là các class thông thường được được implement các interface đã được spring đăng ký với JMX. Ở đây có các hàm được định nghĩa các công việc cần thực thi như thế nào, tương tác với các class nào.
* **Agent level**: được gọi là MBeanServer – mức cốt lỗi của JMX. Nó hoạt động như một trung gian của MBean và ứng dụng. Ở mức này, các interface định nghĩa các hàm (không có phần xử lý) được đăng ký với JMX để tạo ra các hàm cho client gọi từ xa. Các MBeanServer này được đăng ký với JMX để mở cổng và tạo các connector.
* **Remote management level**: cho phép các ứng dụng từ xa có thể truy cập đến MBeanServer thông qua các connector được tạo ra ở Agent level. Một kết nối cung cấp đầy đủ truy cập từ xa đến các API MBeanServer sử dụng truyền thông khác nhau (RMI, IIOP, JMS…), trong khi đó các bộ chuyển đổi sẽ chuyển đổi dữ liệu thành giao diện đồ họa trên web (HTML/HTTP, WML/HTTP,…).

#### Tạo Mbean server và đăng ký với JMX

Để tạo các mbean server và đăng ký với jmx cần thực hiện qua 3 bước: tạo interface, tạo implement và export với jmx. Sau đây là một ví dụ cơ bản về tạo và đăng ký jmx, sử dụng hàm tính toán cơ bản làm demo:

* Tạo interface: tạo một interface thông thường định nghĩa các hàm cần thiết, các hàm này sẽ được gọi từ xa sau khi đăng ký với jmx.

**public** **interface** Calculator {

**public** **int** add(**int** a, **int** b)

**public** **int** sub(**int** a, **int** b);

}

* Tạo implement: tạo một class implement interface Calculator và override tất cả các phương thức của interface, class này là class thực thi code và trả về giá trị khi được gọi từ xa.

**public** **class** CalculatorImpl **implements** Calculator {

@Override

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

@Override

**public** **int** sub(**int** a, **int** b) {

**return** a - b;

}

}

* Đăng ký với jmx: việc đăng ký với jmx bao gồm 2 bước (đăng ký với port gì và đăng ký connector cho client thực thi hàm từ xa):

Đăng ký port với jmx: việc đăng ký mbean server với 1 port jmx được spring hỗ trợ nên thực hiện một cách dễ dàng. Khởi tạo giá trị class RmiRegistryFactoryBean trong context.xml của project.

<bean id=*"registry"* class=*"org.springframework.remoting.rmi.RmiRegistryFactoryBean"*>

<property name=*"port"* value=*"1234"* />

</bean>

Khi thực thi ứng dụng jmx sẽ mở port 1234 nhưng ta chưa định nghĩa các interface cũng như các implement của nó nên client vẫn chưa thể thấy được các hàm. Để đăng ký các interface cũng như các implement cho jmx thấy thì cần khởi tạo giá trị cho class MbeanExporter.



Cần chú ý: ở dòng *<entry key="demo=calculator" value-ref="calculator" />,* key chính là key được đăng ký tại mbean server, value-ref là giá trị chỉ đến class implement của interface để thực thi các phương thức. Sở dĩ có key là do mbean server được phân thành nhiều cấp nhỏ hơn so với TCP truyền thống, với mbean để kết nối đến cần phải biết địa chỉ host và port, do trong cùng 1 host và 1 port có thể đăng ký được nhiều mbean server nên mbean server có thêm khái niệm key hay còn được gọi là object name để phân biệt giữa các mbean server với nhau. Trong mỗi object name sẽ có nhiều interface và nhiều implement được định nghĩa. Mức nhỏ nhất, với mỗi interface thì được định nghĩa nhiều hàm phục vụ cho công việc nào đó.

Đăng ký connector: Sau khi thực hiện các bước trên thì mbean server đã sẵn sàng phục vụ, nhưng việc gọi hàm chỉ được thực thi khi client gọi trên cùng 1 máy. Để có thể cho phép các thiết bị ở xa thực thi hàm thì cần phải đăng ký 1 (hoặc nhiều) connector. Để đăng ký connector chỉ cần khởi tạo giá trị cho class ConnectorServerFactoryBean của spring.

<bean id=*"serverConnector"*

class=*"org.springframework.jmx.support.ConnectorServerFactoryBean"*>

<property name=*"objectName"* value=*"connector:name=rmi"* />

<property name=*"serviceUrl"*

value=*"service:jmx:rmi:///jndi/rmi://localhost:1234/jmxrmi"* />

</bean>

Khi này mbean server đã có thể cho các thiết bị ở xa connect và gọi các hàm định nghĩa sẵng với ip host là ip của máy đang chạy mbean server và với port là 1234. Lúc này client sẽ connect đến mbean server theo đường dẫn: *service:jmx:rmi:///jndi/rmi://localhost:1234/jmxrmi,* tất nhiên ở client và server phải cùng chia sẽ các interface đăng ký mbean thì client mới có thể thực thi các hàm trong đó được.

#### Run Mbean server

Sau khi đã tạo đầy đủ các interface, implement và cấu hình đăng ký trong context.xml thì việc run một mbean server trở nên dể dàng với sự hỗ trợ của spring. Tạo hàm main và load context.xml là việc run mbean server cũng như đăng ký port, tạo connector đều được thực hiện. Sau đây là một ví dụ minh họa:

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**final** ConfigurableApplicationContext context = **new** ClassPathXmlApplicationContext("META-INF/demoContext.xml");

System.***out***.println("Loading context successful....!");

Runtime.*getRuntime*().addShutdownHook(**new** Thread() {

**public** **void** run() {

System.***out***.println("Clean up before close server!");

context.close();

System.***out***.println("Closed Server!");

}

});

}

}

Kết quả thực thi hàm main

SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".

SLF4J: Defaulting to no-operation (NOP) logger implementation

SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.

Loading context successful....!

Việc load context của spring sẽ thực hiện load qua tất cả những cấu hình trong file context.xml. Do được implement và khởi tạo giá trị cho 1 số class nên spring thực hiện đăng ký với jmx và mở port cộng với listener để phục vụ các kết nối.

#### Java JMX Tool

Jconsole là công cụ (thuộc Remote management level) cho phép tạo kết nối đến các Mbean server và thực thi hàm cũng như hiển thị kết quả trả về từ phía server (tham khảo mục lục G).

#### JMX client

Sau khi đã tạo được các mbean server và thực thi chúng, client cần biết được host ip, port, url đã được mbean server đăng ký và object name thì mới có thể kết nối vào được. Java core đã hổ trợ sẵn các thư viện để tạo kết nối mà không cần sự hỗ trợ từ spring. Một ví dụ về client jmx.

**public** **static** **void** main(String[] args) **throws** IOException, MalformedObjectNameException {

String strObjectName = "demo:thesis=calculator";

String url = "service:jmx:rmi:///jndi/rmi://localhost:1234/jmxrmi";

JMXServiceURL jmxUrl = **new** JMXServiceURL(url);

JMXConnector jmxConnector = JMXConnectorFactory.*connect*(jmxUrl);

MBeanServerConnection mbsc = jmxConnector.getMBeanServerConnection();

ObjectName objectName = **new** ObjectName(strObjectName);

Calculator cal = JMX.*newMBeanProxy*(mbsc, objectName, Calculator.**class**);

System.***out***.println("Result: " + cal.add(40, 20));

}

Kết quả thực thi client:

Result: 60

* strObjectName: là object name được tạo trong context.xml nếu object không tồn tại sẽ xảy ra exception MalformedObjectNameException.
* url: là url được cấu hình trong phần tạo connector của spring, nếu không tìm thấy hoặc url sai định dạng thì xảy ra exception IOException.
* Sau khi khởi tạo được giá trị các connector và object name, ta cần lấy được interface đã đăng ký phía mbean server (Calculator) và thực hiện gọi hàm như hàm cục bộ.

## 2.4 MONGODB

### No SQL

#### Khái niệm

NoSQL là một khái niệm chỉ về một lớp các hệ cơ sở dữ liệu không sử dụng mô hình quan hệ (RDBMS). RDBMS vốn tồn tại khá nhiều nhược điểm như có hiệu năng không tốt nếu kết nối dữ liệu nhiều bảng lại hay khi dữ liệu trong một bảng là rất lớn. NoSQL ra đời năm 1998 bởi Carlo Strozzi khi ông lập mới một hệ cơ sở dữ liệu quan hệ mã nguồn mở nhanh và nhẹ không liên quan đến SQL. Thuật ngữ NoSQL đánh dấu bước phát triển của thế hệ CSDL mới: phân tán (distributed) và không ràng buộc (non-relational).

#### Đặc điểm

Khác với cơ sở dữ liệu quan hệ, NoSql lưu thông tin dưới dạng json format.

* NoSQL lưu trữ dữ liệu của mình theo dạng cặp giá trị “key – value”. Sử dụng số lượng lớn các node để lưu trữ thông tin – Mô hình phân tán dưới sự kiểm soát phần mềm.
* Chấp nhận dữ liệu bị trùng lặp do một số node sẽ lưu cùng thông tin giống nhau.
* Một truy vấn sẽ được gửi tới nhiều máy cùng lúc, do đó khi một máy nào đó không phục vụ được sẽ không ảnh hưởng lắm đến chất lượng trả về kết quả
* Phi quan hệ – không có ràng buộc nào cho việc nhất quán dữ liệu.
* Tính nhất quán không theo thời gian thực: Sau mỗi thay đổi CSDL, không cần tác động ngay đến tất cả các CSDL liên quan mà được lan truyền theo thời gian.

Với các đặc điểm trên, NoSql chấp nhận dư thừa dữ liệu và cần phải được ràng buộc bởi người lập trình nhưng ngược lại NoSql có nhiều điểm lợi hơn so với cơ sở dữ liệu quan hệ, nó chập nhận số lượng truy cập và các truy cập được trả kết quả cực nhanh.

#### Các dạng NoSql cơ bản

**Key – value data stores**: Dữ liệu lưu dưới dạng cặp key – value. Giá trị được truy xuất thông qua key.

* Ví dụ : Redis, Dynomite, Project Voldemort.
* Thường cho: Content caching Applications.
* Ưu điểm: Tìm kiếm rất nhanh.
* Nhược điểm: Lưu dữ liệu không theo khuôn dạng (schema) nhất định.

**Column-based – Tabular**: Cơ sở dữ liệu tổ chức dưới dạng các bảng. Gần giống với mô hình RDBMS. Tuy nhiên, Chúng lưu dữ liệu bởi các cột chứ không phải bằng các dòng. Nó khá thích hợp với để hiển thị bằng các phần mềm quản lý kho dữ liệu

* Ví dụ : Apache Hbase, Apache Cassandra, Hypertable
* Thường cho: các hệ phân tán file
* Ưu điểm: Tìm kiếm nhanh, Phân tán dữ liệu tốt
* Nhược điểm: Hỗ trợ được với rất ít phần mềm

**Document-based**: Dữ liệu (bán cấu trúc hay semi-structured) được lưu trữ và tổ chức dưới dạng một tập hợp các document. Các document này linh hoạt, mỗi document có một tập nhiều trường.

* Ví dụ : Apache CouchDB và MongoDB
* Thường cho: Web applications
* Ưu điểm: Dùng khi dữ liệu nguồn không được mô tả đầy đủ
* Nhược điểm: Hiệu năng truy vấn, Không có cú pháp chuẩn cho câu truy vấn dữ liệu

**Graph-based data-stores**: Những CSDL này áp dụng lý thuyết đồ thị trong khoa học máy tính để lưu trữ và truy xuất dữ liệu. Chúng tập trung vào tính rời rạc giữa các phần dữ liệu. Các phần tử đơn vị dữ liệu được biểu thị như một nút và liên kết với các thành phần khác bằng các cạnh.

* Ví dụ : Neo4j, InfiniteGraph, DEX.
* Thường cho: Social networking, hệ trợ giúp.
* Ưu điểm: Ứng dụng các thuật toán trên đồ thị như Đường đi ngắn nhất, liên thông,…
* Nhược điểm: Phải duyệt nội bộ đồ thị, để trả lời lại các truy vấn. Không dễ để phân tán.

### MongoDB

#### Khái quát

MongoDB (“humongous”) là một hệ cơ sở dữ liệu NoSQL mã nguồn mở.

[](https://bigsonata.files.wordpress.com/2014/06/logo.png)

Hình 2. - Logo của mongodb

Thay cho việc lưu trữ dữ liệu vào các bảng có quan hệ với nhau như truyền thống, MongoDB lưu các dữ liệu cấu trúc dưới dạng giống với JSON(JavaScript Object Notation) và gọi tên là BSON. Dự án được bắt đầu triển khai vào tháng 10 năm 2007 bởi 10gen trong khi công ty này đang xây dựng một nền tảng như là dịch vụ (Platform as a Service) giống như Google App Engine. Phải đến năm 2009, dự án này được tách độc lập. Hệ thống có thể chạy trên Windows, Linux, OS X và Solaris.

**MongoDB có những ưu điểm sau đây:**

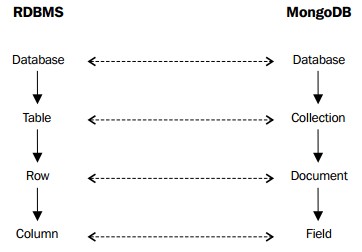
* Dễ học, có một số nét khá giống với CSDL quan hệ – Quản lý bằng command line hoặc bằng GUI như Robomongo, RockMongo hoặc phpMoAdmin.
* Linh động, không cần phải định nghĩa cấu trúc dữ liệu trước khi tiến hành lưu trữ nó -> rất tốt khi ta cần làm việc với các dạng dữ liệu không có cấu trúc.
* Khả năng mở rộng tốt (distributed horizontally), khả năng cân bằng tải cao, tích hợp các công nghệ quản lý dữ liệu vẫn tốt khi kích thước và thông lượng trao đổi dữ liệu tăng.
* Miễn phí.

#### Kiến trúc tổng quát

Một MongoDB Server sẽ chứa nhiều database. Mỗi database lại chứa một hoặc nhiều colection. Đây là một tập các documnents, về mặt logic thì chúng gần tương tự như các table trong CSDL quan hệ. Tuy nhiên, điểm hay ở đây là ta không cần phải định nghĩa trước cấu trúc của dữ liệu trước khi thao tác thêm, sửa dữ liệu… Một document là một đơn vị dữ liệu – một bản ghi (không lớn hơn 16MB). Mỗi chúng lại chứa một tập các trước hoặc các cặp key – value. Key là một chuỗi ký tự, dùng để truy xuất giá trị dạng : string, integer, double,…

Cấu trúc có vẻ khá giống JSON, tuy nhiên, khi lưu trữ document này ra database, MongoDB sẽ serialize dữ liệu thành một dạng mã hóa nhị phân đặc biệt – BSON. Ưu điểm của BSON là hiệu quả hơn các dạng format trung gian như XML hay JSON cả hệ tiêu thụ bộ nhớ lẫn hiệu năng xử lý. BSON hỗ trợ toàn bộ dạng dữ liệu mà JSON hỗ trợ (string, integer, double, Boolean, array, object, null) và thêm một số dạng dữ liệu đặc biệt như regular expression, object ID, date, binary, code.

Hình 2.5 thể hiện sự so sánh tương quan giữa MongoDB và cơ sở dữ liệu quan hệ.



Hình 2. - So sánh MongoDB và CSDL quan hệ

#### Truy vấn

Tham khảo phần 2 (Truy vấn) phục lục B: Cài đặt và cấu hình MongoDB.

#### Cài đặt và cấu hình MongoDB

Tham khảo phần 1 (Cài đặt) phụ lục B: Cài đặt và cấu hình MongoDB.

#### Sping data mongodb

**Giới thiệu**

Spring data mongodb là một framework mở rộng của spring hỗ trợ các truy cập đến mongodb đồng thời hỗ trợ quản lý phiên giao dịch và đơn giản quá công việc truy xuất đến mongodb.

**Cách sử dụng**

Để sử dụng spring data mongodb trước hết cần thêm dependency trong pom.xml như sau:

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-mongodb</artifactId>

<version>1.6.0.RELEASE</version>

</dependency>

Có thể lên trang chủ maven <http://mvnrepository.com/> để tìm phiên bản mới nhất.

Sau khi thêm thư viện spring data mongodb thì việc tiếp theo cần cấu hình các tham số cho kết nối trong context.xml bằng việc khởi tạo các giá trị cho class MongoTemplate:

<context:annotation-config />

<!-- config mongodb connection -->

<mongo:mongo host=*"localhost"* port=*"27017"* />

<mongo:db-factory dbname=*"thesis"* />

<bean id=*"mongoTemplate"* class=*"org.springframework.data.mongodb.core.MongoTemplate"*>

<constructor-arg name=*"mongoDbFactory"* ref=*"mongoDbFactory"* />

</bean>

* *context:annotation-config:* Để khi load file context, các class sử dụng autowrite mongotemplate sẽ được khởi tạo giá trị dựa vào thông tin cấu hình. Ví dụ:

**public** **abstract** **class** DbAccess {

@Autowired

**protected** MongoTemplate mongoTemplate;

}

* *Mongo host, port, dbname:* tương ứng với mongodb server đang thực thi.
* Ngoài MongoTemplate, spring data mongodb còn hỗ trợ các class truy xuất mongodb như: *MongoOperation* và *MongoRepository* với các api hỗ trợ nhiều hơn, nhưng trong phạm vi đề tài này để thấy được các câu lệnh gần gủi và cơ bản nên toàn bộ các truy xuất đến mongodb sẽ được thực hiện thông qua *MongoTemplate*.

Sau khi khởi tạo được giá trị *mongoTemplate* thì tất cả các api của *mongoTemplate* có thể sử dụng dể dàng.

## 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.

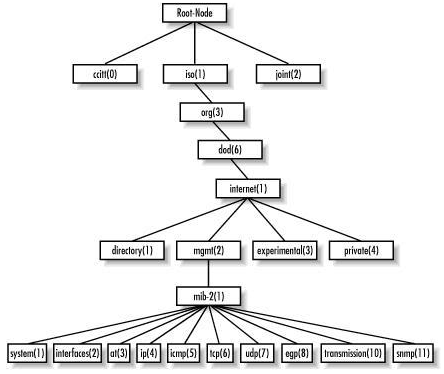
### SNMP4j framework

#### Giới thiệu

SNMP4j framework là một framework hỗ trợ java tạo ra những gói tin snmp và gởi tới thiết bị cũng như nhận lại trả lời. Các gói snmp truy xuất các đối tượng trên thiết bị đều dựa vào OID.

#### SNMP OID

Các đối tượng trong snmp được quản lý thành các chuổi octet dùng BER (Basic Encoding Rules). BER xây dựng cách mã hóa và giải mã để truyền các đối tượng qua các môi trường truyền như Ethernet. Tên hay OID được tổ chức theo dạng cây. Tên của một đối tượng được thành lập từ một dãy các số nguyên hay chữ dựa theo các nút trên cây, phân cách nhau bởi dấu chấm. Hình 2.6 là 1 ví dụ về phân cấp của OID:



Hình 2. - Mô hình phân cấp OID

MIB-II có 10 nhánh con được định nghĩa trong RFC 1213, kế thừa từ MIB-I trong RFC 1066. Mỗi nhánh có một chức năng riêng:

* **system** (1.3.6.1.2.1.1) Định nghĩa một danh sách các đối tượng gắn liền với hoạt động của hệ thống như: thời gian hệ thống khởi động tới bây giờ, thông tin liên lạc của hệ thống và tên của hệ thống.
* **interfaces** (1.3.6.1.2.1.2) Lưu giữ trạng thái của các interface trên một thực thể quản lý. Theo dõi một interface “up” hoặc “down”, lưu lại các octet gửi và nhận, octet lỗi hay bị hủy bỏ.
* **at** (1.3.6.1.2.1.3) Nhóm at (address translation) bị phản đối, nó chỉ cung cấp khả năng tương thích ngược. Nhóm này được bỏ từ MIB-III trở đi.
* **ip** (1.3.6.1.2.1.4) Lưu giữ nhiều thông tin liên quan tới giao thức IP, trong đó có phần định tuyến IP.
* **icmp** (1.3.6.1.2.1.5) Lưu các thông tin như gói ICMP lỗi, hủy.
* **tcp** (1.3.6.1.2.1.6) Lưu các thông tin khác dành riêng cho trạng thái các kết nối TCP như: đóng, lắng nghe, báo gửi…
* **udp** (1.3.6.1.2.1.7) Tập hợp các thông tin thống kê cho UDP, các datagram vào và ra, …
* **egp** (1.3.6.1.2.1.8) Lưu các tham số về EGP và bảng EGP lân cận.
* **transmission** (1.3.6.1.2.1.10) Không có đối tượng nào trong nhóm này, nhưng nó định nghĩa các môi trường đặc biệt của MIB.
* **snmp** (1.3.6.1.2.1.11) Đo lường sự thực thi của SNMP trên các thực thể quản lý và lưu các thông tin như số các gói SNMP nhận và gửi.

#### Sử dụng

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

<!-- SNMP4j -->

<dependency>

<groupId>org.snmp4j</groupId>

<artifactId>snmp4j</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

<groupId>org.snmp4j</groupId>

<artifactId>snmp4j-agent</artifactId>

<version>2.0.9</version>

</dependency>

Tuy nhiên ta cần phải thêm 1 định nghĩa repo cho snmp4j, repo mặc định có thể không thể download gói snmp4j về được.

<!-- specific repository for SNMP4j -->

<repositories>

<repository>

<id>snmp4j-repo</id>

<url>https://oosnmp.net/dist/release/</url>

</repository>

</repositories>

Sau khi đã thêm thư viện, ta có thể sử dụng 1 số hàm cơ bản để tạo và gởi gói snmp xuống thiết bị:

**public** **abstract** **class** DvAccess {

**protected** String DOT = ".";

**protected** Snmp snmp;

**protected** String ipAddress;

**protected** String defaultIp;

**protected** TransportMapping transport;

**protected** CommunityTarget target;

**public** DvAccess(String ipAddress) **throws** IOException {

**this**.defaultIp = ipAddress;

**this**.ipAddress = "udp:" + ipAddress + "/161";

target = getTarget();

transport = **new** DefaultUdpTransportMapping();

snmp = **new** Snmp(transport);

transport.listen();

}

**public** CommunityTarget getTarget() {

Address targetAddress = GenericAddress.*parse*(ipAddress);

target = **new** CommunityTarget();

target.setAddress(targetAddress);

target.setRetries(Community.***RETRIES***);

target.setTimeout(Community.***TIMEOUT***);

target.setVersion(SnmpConstants.***version2c***);

target.setCommunity(**new** OctetString(Community.***PRIVATE***));

**return** target;

}

**public** **void** close() **throws** IOException {

transport.close();

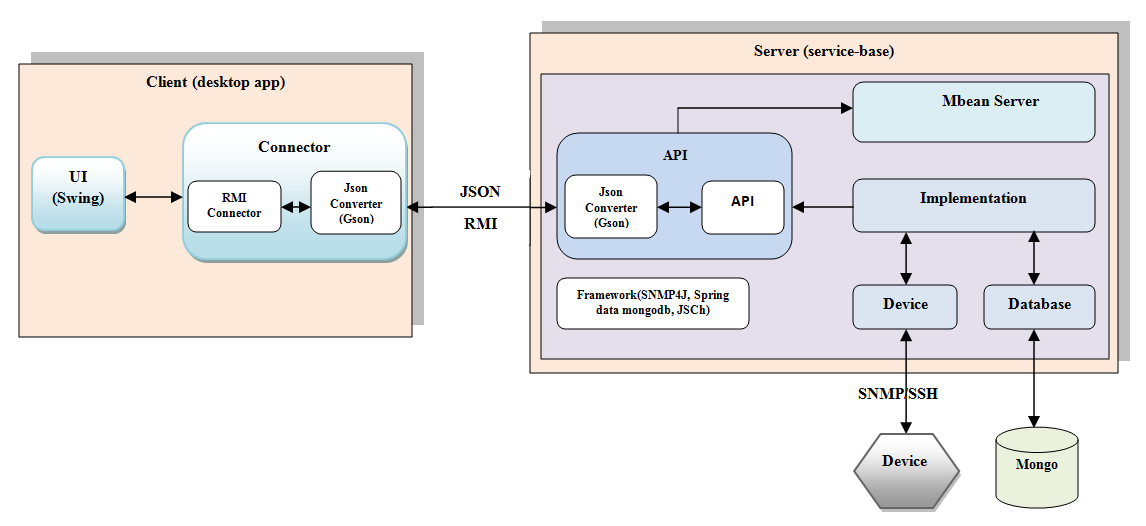
snmp.close();

}

}

Class DvAccess là 1 abstract class, thiết lập các kết nối cơ bản xuống thiết bị và lắng nghe trả lời từ thiết bị. Do là 1 abstract class nên để gọi được thể hiện của nó thì cần phải có lớp thừa kế. DvSet, DvGet, DvWalk là các class thừa kế và thực hiện các công việc tương tác với thiết bị thông qua snmpget, snmpset, snmpwalk.

# CHƯƠNG 3: SƠ ĐỒ CHỨC NĂNG



Hình 3. - 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.

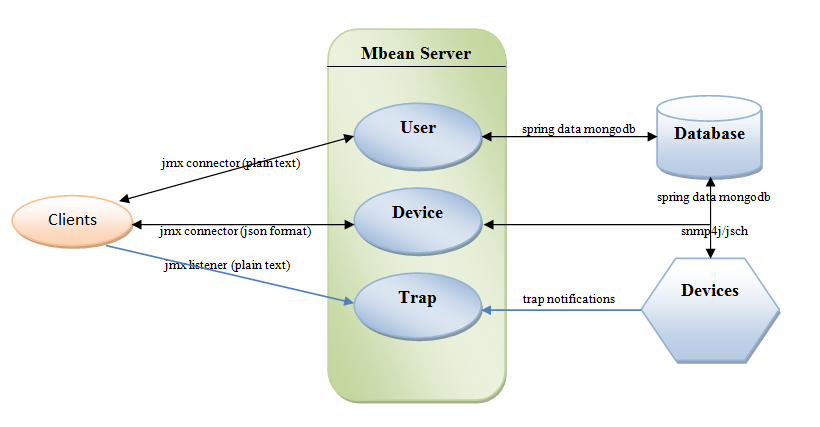
**Client:**

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

### MBean



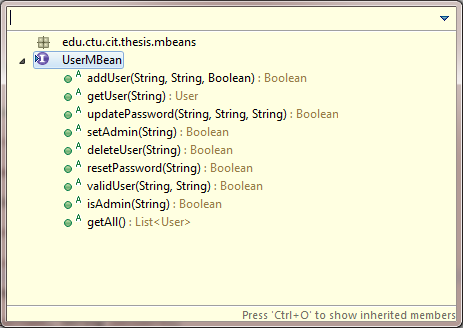
Hình 3. - Mô hình tổng quát MBeanServer

MbeanServer được chia thành 3 thành phần (object name) chủ đạo: user mbean, device mbean và trap mbean. Hình 3.2 là phần cấu hình đăng ký 3 mbean.



##### User mbean

Định nghĩa nên các hàm tương tác xuống mongodo, một số hàm xác thực người dùng, phân quyền, thêm mới hay xóa người dùng. User mbean được đăng ký với object name là “*thesis.user*”. Tất cả việc tương tác xuống cơ sở dữ liệu đều thông qua lớp MongoTemplate, thể hiện của nó là biến mongoTemplate.



Hình 3. - User mbean API

UserMbean là một interface định nghĩa các hàm cần thiết và phía client chỉ biết được interface này. Các code thực thi được implement lại tại class UserMbeanImpl, ví dụ về 2 phương thức add user và get user:

@Override

**public** Boolean addUser(String username, String pass, Boolean isAdmin) {

User user = **new** User(username, User.*crypt*(pass), isAdmin);

**if** (!exists(user)) {

mongoTemplate.save(user);

**return** **true**;

}

**return** **false**;

}

@Override

**public** User getUser(String username) {

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

**return** one;

}

##### Device mbean

Là phần trọng tâm và cũng là phần được dành nhiều thời gian thiết kế. Bao gồm tất cả các api tương tác xuống thiết bị, mongodb và một vài chức năng tương tác với Trap mbean. Device mbean được đăng ký với object name là “*thesis.device*”. DeviceMbean sử dụng 4 class chính là:

DvAccess: là một abstract class, định nghĩa các thông tin chung cho gói snmp như: ip, port, url, timeout…

**public** **abstract** **class** DvAccess {

**protected** String DOT = ".";

**protected** Snmp snmp;

**protected** String ipAddress;

**protected** String defaultIp;

**protected** TransportMapping transport;

**protected** CommunityTarget target;

**public** DvAccess(String ipAddress) **throws** IOException {

**this**.defaultIp = ipAddress;

**this**.ipAddress = "udp:" + ipAddress + "/161";

target = getTarget();

transport = **new** DefaultUdpTransportMapping();

snmp = **new** Snmp(transport);

transport.listen();

}

**public** CommunityTarget getTarget() {

Address targetAddress = GenericAddress.*parse*(ipAddress);

target = **new** CommunityTarget();

target.setAddress(targetAddress);

target.setRetries(Community.***RETRIES***);

target.setTimeout(Community.***TIMEOUT***);

target.setVersion(SnmpConstants.***version2c***);

target.setCommunity(**new** OctetString(Community.***PRIVATE***));

**return** target;

}

**public** **void** close() **throws** IOException {

transport.close();

snmp.close();

}

}

DvGet: là lớp thừa kế từ DvAccess, định nghĩa các hàm cần thiết để lấy thông tin từ thiết bị, sử dụng phương thức snmpget và DvWalk trong việc truy cập thông tin. Một số hàm cơ bản trong DvGet:

**public** String getSysName() **throws** IOException {

String asString = getAsString(OIDObject.***sysName***);

**return** asString;

}

**public** String getPhysisAddress(String ip) **throws** IOException {

String physicAddress = OIDObject.*physicAddress*(ip);

**return** getAsString(physicAddress);

}

**public** IpObject getIpAddress(String ifDescrOid) **throws** IOException {

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

IpObject result = **null**;

List<IpObject> ipTableObjects = dvWalk.getIpTableObject();

**for** (IpObject ipObject : ipTableObjects) {

**if** (ipObject.getIfOid().equals(ifDescrOid)) {

result = ipObject;

**break**;

}

}

dvWalk.close();

**return** result;

}

**public** IpObject getIpObject(Integer ifIndex) {

String oid = OIDObject.***ifIndexIp*** + ifIndex;

**try** {

**return** getIpAddress(oid);

} **catch** (IOException e) {

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

}

**return** **null**;

}

}

DvSet: cũng là 1 lớp được thừa kế từ DvAccess, mục tiêu tạo ra các gói snmp và thay đổi giá trị trên thiết bị. Sử dụng chủ yếu snmpset. Một số hàm cơ bản trong DvSet:

**public** Boolean setLocation(String name) **throws** IOException {

OID oid = **new** OID(OIDObject.***sysLocation***);

OctetString var = **new** OctetString(name);

VariableBinding varBind = **new** VariableBinding(oid, var);

createPDU();

pdu.add(varBind);

ResponseEvent response = snmp.set(pdu, target);

**return** checkResponse(response);

}

**public** Boolean setSnmpAuthenTrap(Boolean boo) **throws** IOException {

OID oid = **new** OID(OIDObject.***snmpAuthen***);

Integer value = boo ? 1 : 2;

Integer32 var = **new** Integer32(value);

VariableBinding varBind = **new** VariableBinding(oid, var);

createPDU();

pdu.add(varBind);

ResponseEvent response = snmp.set(pdu, target);

**return** checkResponse(response);

}

DvWalk: tương tự DvGet, cũng là lớp thừa kế từ DvAccess, sử dụng phương thức snmpget, nhưng việc lấy dữ liệu của DvWalk là một bảng. Hàm lấy bảng ip table từ thiết bị:

**public** List<IpObject> getIpTableObject() {

List<IpObject> lsIpObjects = **new** ArrayList<IpObject>();

TreeUtils treeUtils = **new** TreeUtils(snmp, **new** DefaultPDUFactory());

List<TreeEvent> events = treeUtils.walk(target, **new** OID[] { **new** OID(OIDObject.***ipAddressTableObject***) });

String[] args = { OIDObject.***ifOid***, OIDObject.***ipAddress***, OIDObject.***ipNetmask*** };

List<List<String>> valuess = **new** ArrayList<List<String>>();

**for** (String str : args) {

List<String> values = getVariableTableObject(events, str);

valuess.add(values);

}

IpObject[] ipObjects = **new** IpObject[valuess.get(0).size()];

// oid of IpAddress base on ifDescription oid

List<String> lsOids = valuess.get(0);

**for** (**int** i = 0; i < valuess.get(0).size(); i++) {

ipObjects[i] = **new** IpObject();

ipObjects[i].setIfOid(DOT + OIDObject.***ifDescrition*** + lsOids.get(i));

}

List<String> lsIpAddress = valuess.get(1);

**for** (**int** i = 0; i < valuess.get(0).size(); i++) {

ipObjects[i].setIpAddress(lsIpAddress.get(i));

}

List<String> lsIpNetmask = valuess.get(2);

**for** (**int** i = 0; i < valuess.get(0).size(); i++) {

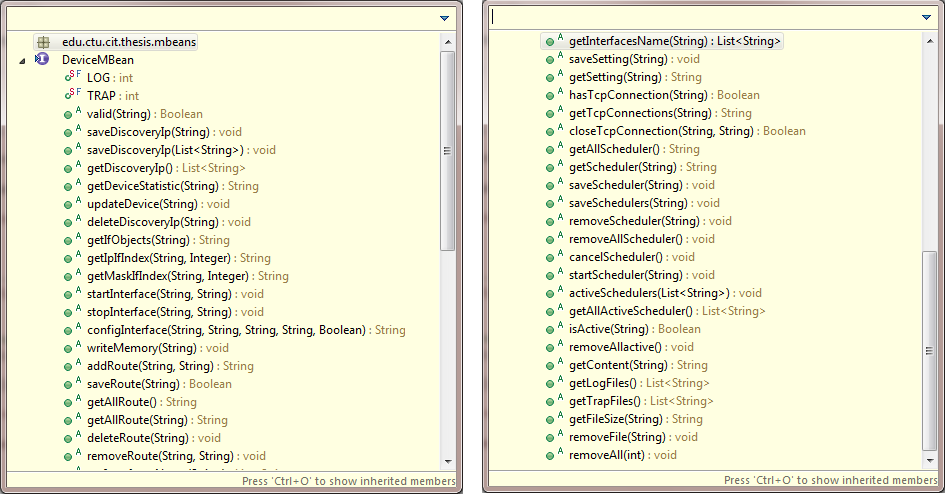
ipObjects[i].setIpNetmask(lsIpNetmask.get(i));

lsIpObjects.add(ipObjects[i]);

}

**return** lsIpObjects;

}



Hình 3. - Device mbean API

##### Trap mbean

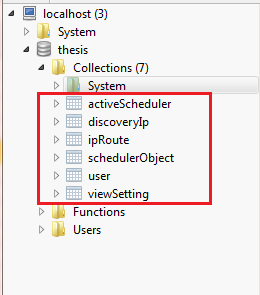
Là phần mbean đơn giản nhất, thực thi và lắng nghe các thông báo từ thiết bị cũng như các sự thay đổi từ người dùng được Device mbean gởi qua. Sử dụng jmx notification để thông báo đến toàn bộ client đang kết nối đến. Trap mbean được đăng ký với object name là “*thesis.trap*”.



Hình 3. - Trap mbean API

### 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. - 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.

### Config file

Server lưu thông tin cấu hình ở file “*server.properties*” bao gồm ba thông tin: host, port và logTraps.

server.properties:

ip=192.168.100.100

port=1234

logTraps=D:/logs/

* Host: lưu trữ ip hiện tại của máy đang thực thi mbean server, có thể thực thi việc get ip và không cần file cấu hình nhưng trong trường hợp máy có nhiều card mạng và có sự lựa chọn cho server.
* Port: lựa chọn port để mở jmx mbean server, thông số bắt buộc phải có.
* LogTraps: là phần chọn đường dẫn để lưu các file log và trap của server, thông số bắt buộc phải có và đường dẫn phải tồn tại trên server để đảm bảo hoạt động chính xác.
* Trong tất cả các phần của server, các thành phần liên quan đến jmx mbean đều được đọc và tạo đường dẫn từ file *server.properties* vì thế khi thay đổi các thông số thì không phải đổi nhiều nơi tạo nên sự thuận tiện và linh hoạt.

## 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. - 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.

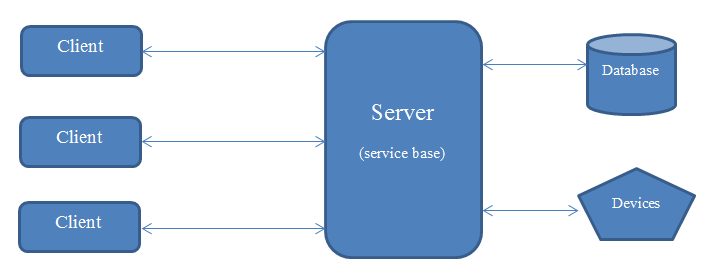
ip=192.168.100.100

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. - Chức năng tổng quát hệ thống

Hệ thống bao gồm 2 thành phần:

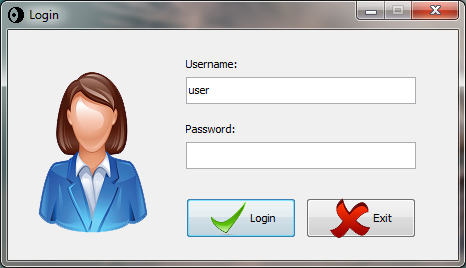
* Một server (dưới dạng service) luôn lắng nghe các kết nối RMI và thực thi các hàm do client gọi, tương tác với database hoặc thiết bị và trả về kết quả cho client.
* Một hoặc nhiều client được thiết kế dựa trên các hàm được cung cấp sẵn phía server, tạo kết nối đến server và thực thi các hàm (từ xa)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

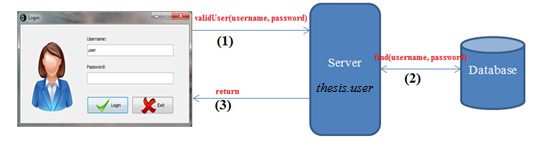
**Giao diện**:



Hình 3. - Giao diện login

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

Login:



Hình 3. - Chức năng login

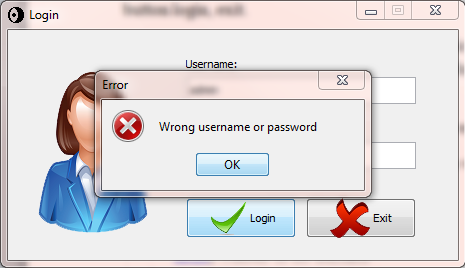
* Sau khi nhập thông tin đăng nhập (username và password), client thực hiện kết nối đến server và thực thi hàm *validUser(username, password).*
* Server khi nhận được yêu cầu xác thực sẽ thực hiện 1 truy vấn xuống database (mongodb) để tìm tài khoản với thông tin được truyền từ client.

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.

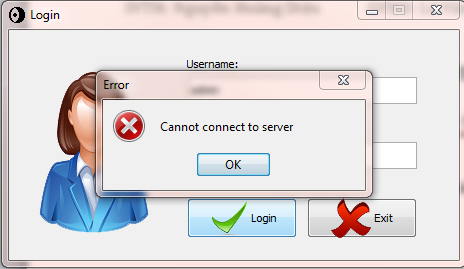
Kết quả trả về từ server:

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



Hình 3. - 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. - Không thể kết nối server

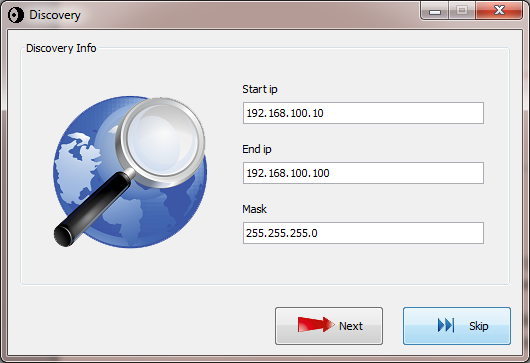
Exit: đóng ứng dụng.

### 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.

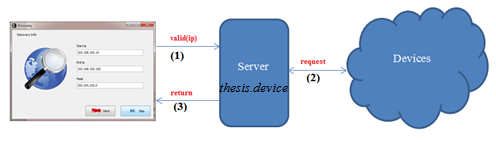
**Giao diện:**



Hình 3. - Giao diện discovery

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

Next:



Hình 3. - 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.

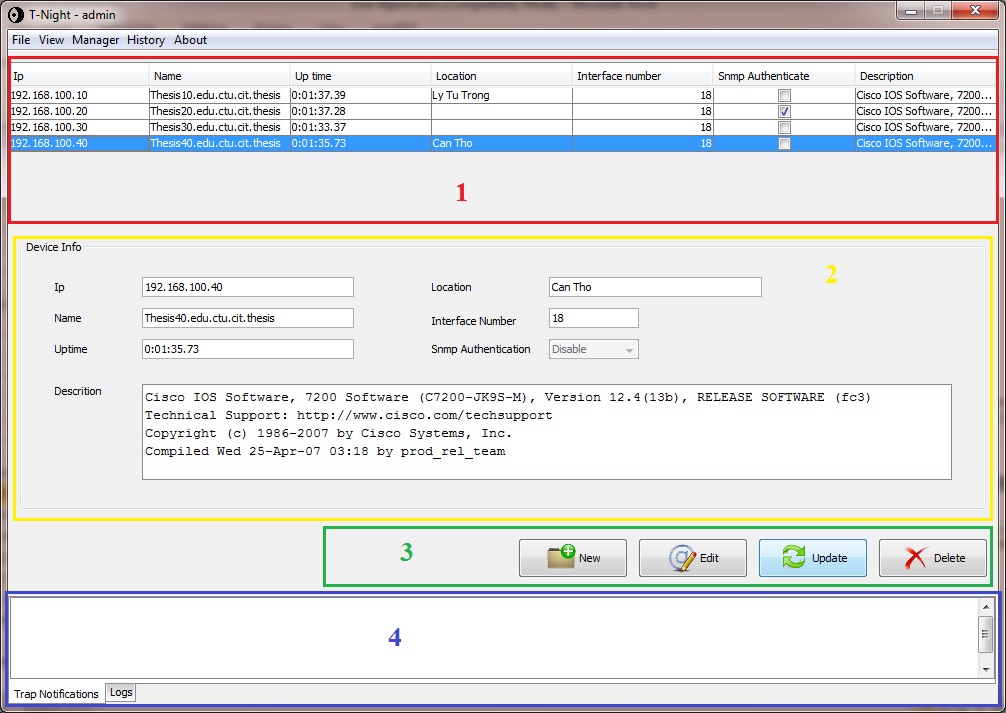
* Sau khi quá trình discovery hoàn tất hoặc người dùng chọn *skip*, giao diện chính (*MainForm*) khởi tạo và hiển thị thông tin tổng quát của những thiết bị đã được tìm thấy hoặc lấy về từ server.

### 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:**

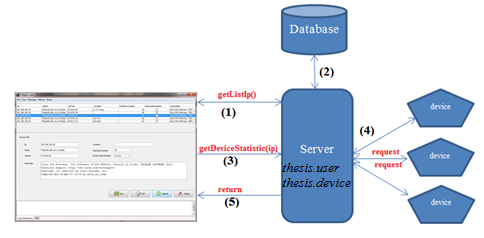


Hình 3. - Giao diện chính

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:**



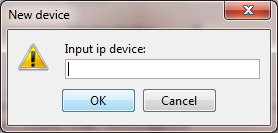
Hình 3. - Mô hình hoạt động khi khởi tạo mainform

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:

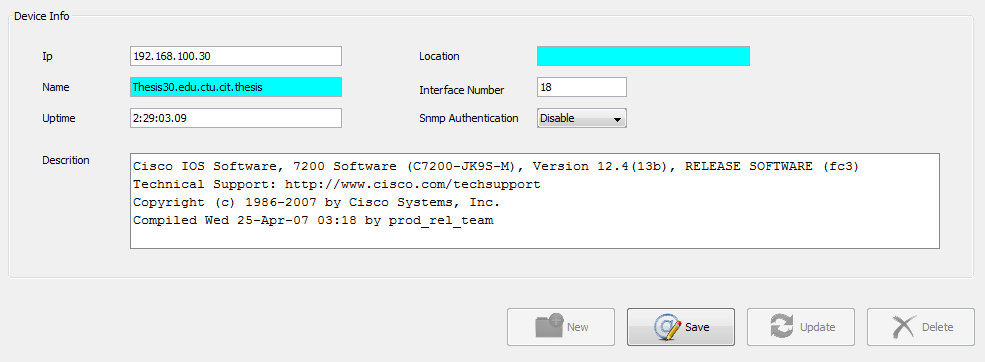
Hiển thị input cho phép nhập vào ip mới.



* 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. - 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.

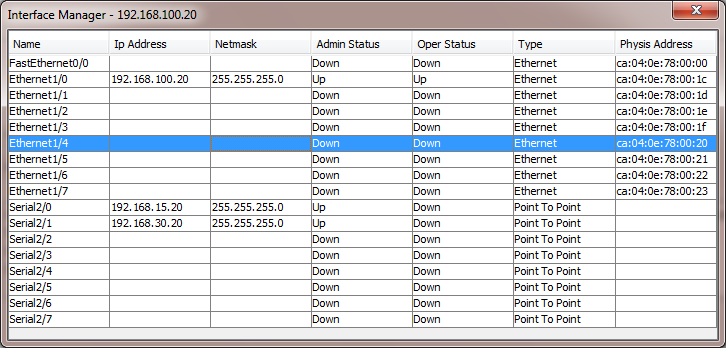
mongoTemplate.remove(new Query(Criteria.where("ip").is(ip)), DiscoveryIp.class);

### 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.

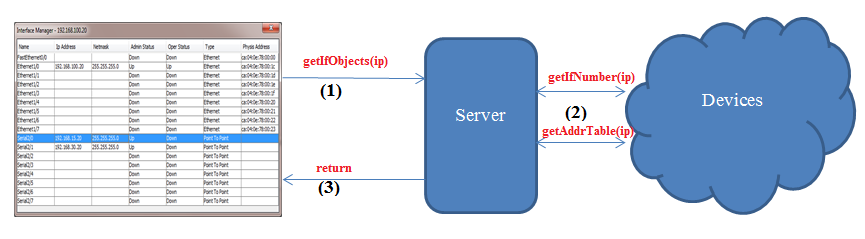
**Giao diện:**



Hình 3. - Giao diện interface manager

**Cách thức hoạt động:**

Khởi tạo form:



Hình 3. - 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:

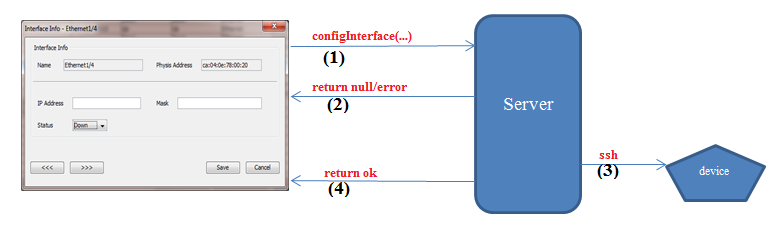
Khi chọn vào 1 interface trên giao diện, 1 popup panel được khởi tạo và hiển thị.



Hình 3. - Giao diện popup - interface manager

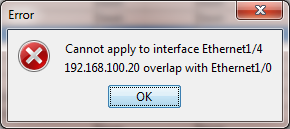
Giao diện hiển thị các thông tin của interface được người dùng chọn. Tại đây, người dùng có thể thiết lập ip (nếu chưa có ip) hoặc thay đổi ip (nếu đã có ip) trên giao diện hiện tại, phần status chọn up/down để bật/tắt giao diện.

Save:



Hình 3. - 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.

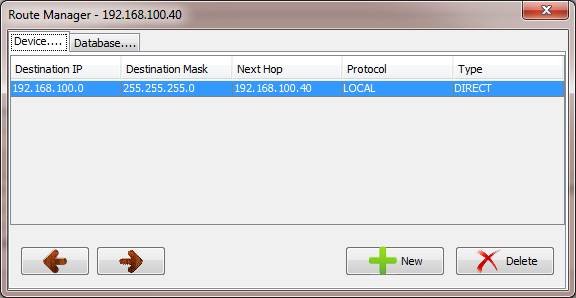
### Chức năng route manager

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:**

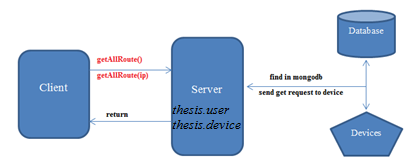


Hình 3. - Giao diện route manager

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. - 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) {

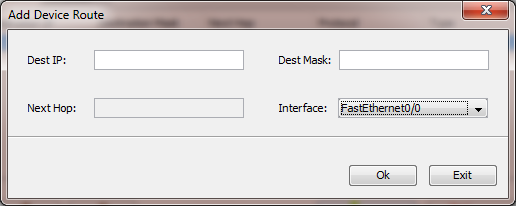
**return** IpRoute.*parseToString*(ipRoutes);

}

**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. - 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)*).

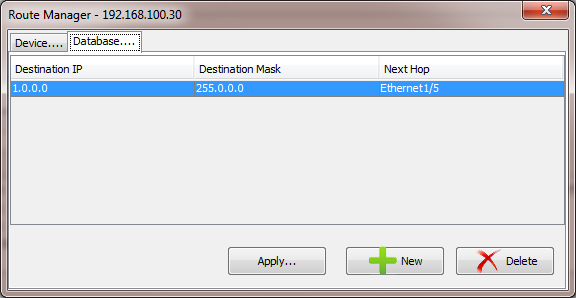
CLISet cliSet = **new** CLISet(hostIp);

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.



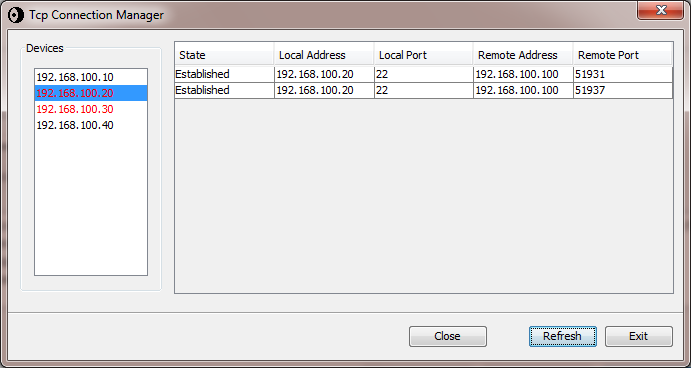
Hình 3. - Tab database

### 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. - 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:

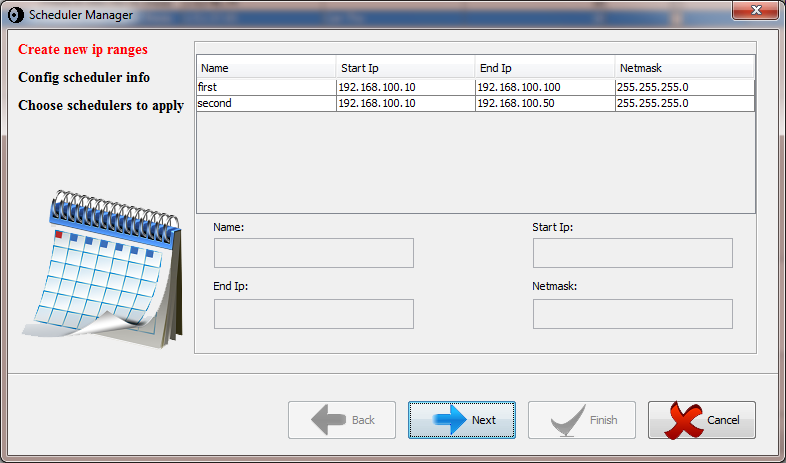
Thoát chức năng, quay trở lại main form.

### 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. - Giao diện scheduler manager

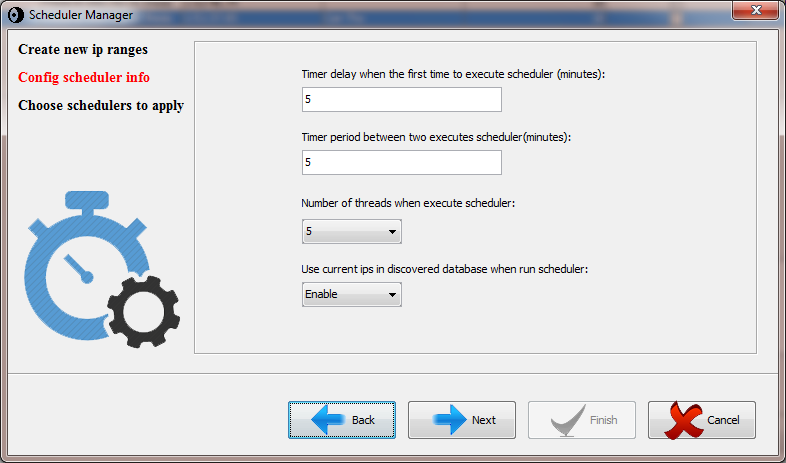
Để khởi động scheduler cần cấu hình qua 3 bước: tạo dãy ip (nếu cần), cấu hình thời gian, chọn dãy ip (nhiều) để thực thi scheduler. Hình 3.31 là giao diện đầu tiên của sheduler manager.

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*).

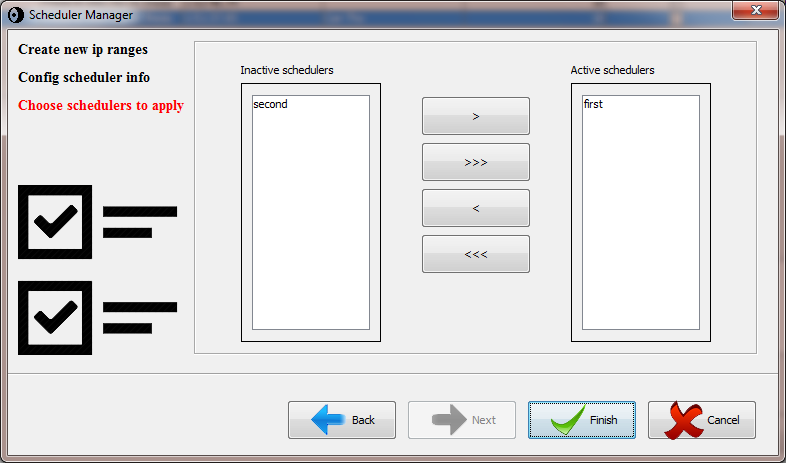
Bước 2: định nghĩa thời gian thực thi.



Hình 3. - 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. - 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.

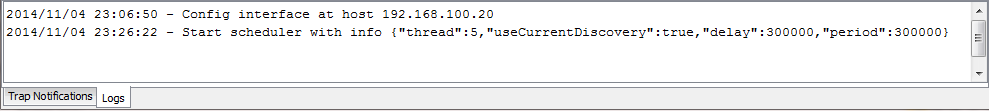
Sau khi chọn Finish, client lấy toàn bộ dữ liệu chuyển thành định dạng json và gọi hàm thực thi scheduler (*startScheduler(json)*). Server tiến hành hủy các scheduler đang thực thi (nếu có) và khởi động scheduler mới.

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. - Thông báo khởi động scheduler mới

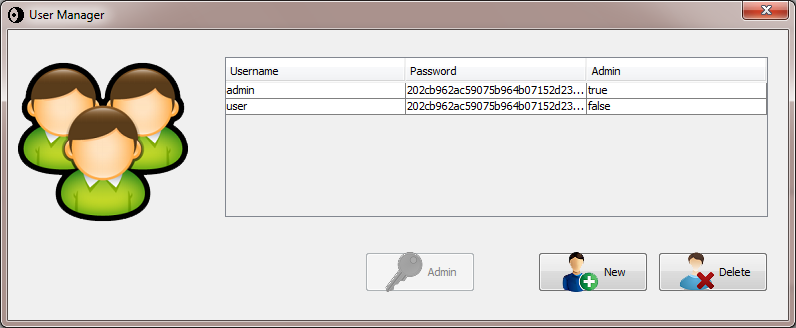
### Chức năng user manager

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. - Giao diện user manager

**Cách thức hoạt động:**

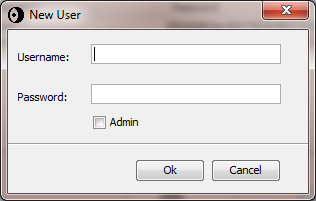
* Khởi tạo form:

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. - 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

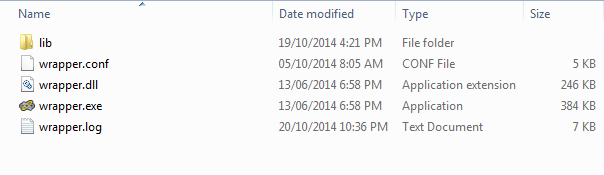
1. *http://www.oidview.net/tools/index.html*
2. *http://techdive.in/snmp/snmp4j-snmp-get-example*
3. *http://www.ehow.com/how\_5001442\_configure-snmp-cisco-router.html*
4. *http://download.intermapper.com/thirdparty/libsmi-0.4.8/mibs/dirlist.php*
5. *http://www.iphostmonitor.com/mib/mibs/IP-ACCESS-LIST-MIB.html*
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 A: CÀI ĐẶT WINDOWS SERVICE VỚI WRAPPER

Tải bản Wapper phù hợp với hệ điều hành về từ địa chỉ: <http://sourceforge.net/projects/wrapper/>

Sau khi tải về giải nén ra thư mục.



Ta cần sử dụng và chỉnh sửa file wapper.conf theo đúng chuẩn quy định, nếu file cấu hình để tên mặc định và cùng thư mục với wapper.exe thì khi thực thi wapper không cần chỉ đường dẫn đến file cấu hình, ở hệ thống hiện tại, mọi thư viện (\*.jar) đều được đặt trong thư mục lib, nếu để ở nhiều nơi thì cần phải thêm đường dẫn đến các thư mục đó nếu không hệ thống sẽ không thực thi được.

File *wapper.conf* của hệ thống:

*# Java Application*

*wrapper.java.command=%JAVA\_HOME%\bin\java*

*wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp*

*wrapper.java.classpath.1=.*

*wrapper.java.classpath.2=./lib*

*wrapper.java.classpath.3=./lib/\*.jar*

*wrapper.java.library.path.1=.*

*wrapper.java.additional.1=-Xms256m*

*wrapper.java.additional.2=-Xmx768m*

*wrapper.java.additional.3=-Djava.net.preferIPv4Stack=true*

*wrapper.app.parameter.1=edu.ctu.cit.thesis.main.Main*

*wrapper.jvm\_exit.timeout=0*

*wrapper.ping.timeout=300*

*wrapper.disable\_restarts=true*

*wrapper.console.format=PM*

*wrapper.console.loglevel=INFO*

*wrapper.logfile=wrapper.log*

*wrapper.logfile.format=LPTM*

*wrapper.logfile.loglevel=INFO*

*wrapper.logfile.maxsize=1m*

*wrapper.logfile.maxfiles=1*

*wrapper.syslog.loglevel=NONE*

*wrapper.console.title=T-Night Service*

*# Wrapper Windows NT/2000/XP Service Properties*

*wrapper.ntservice.name=t-night*

*wrapper.ntservice.displayname=T-Night Service*

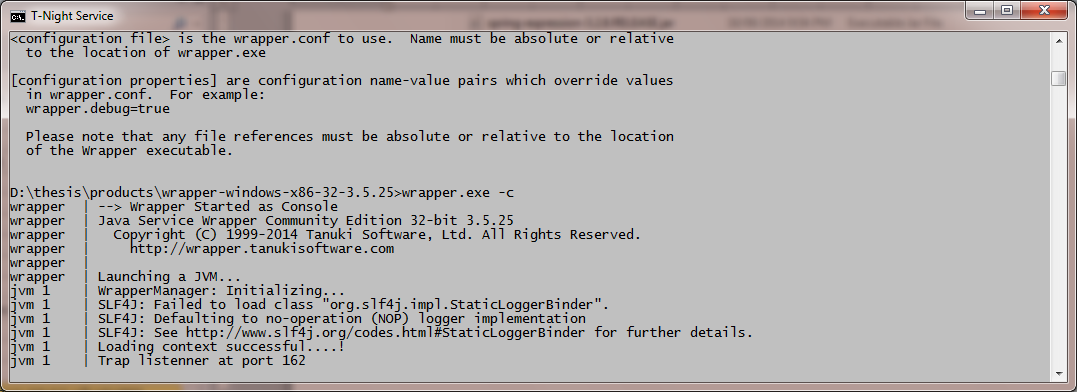
*wrapper.ntservice.description=Service for thesis presentation of Nguyen Hoang Dieu - 1101721*

*wrapper.ntservice.dependency.1=*

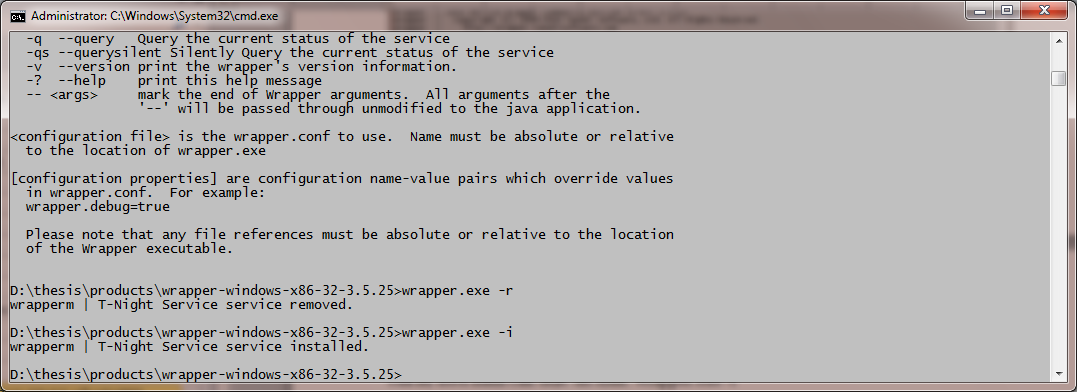
*wrapper.ntservice.starttype=DEMAND\_START*

*wrapper.ntservice.interactive=false*

Sau khi đã có được wapper và cấu hình thông tin cần thiết, ta tiến hanh chạy thử wrapper với câu lệnh: *wrapper.exe -c*



Như vậy, việc run wrapper với console đã chạy ổn định, để cài đặt wrapper thành service của hệ điều hành cần thực thi lệnh: *wrapper.exe –i*. Nếu service đã tồn tại sẽ báo lỗi, để gỡ bỏ service củ thực thi lệnh *wapper.exe –r*:



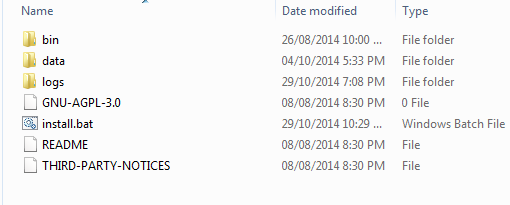
* Như vậy service đã cài đặt thành công.

## PHỤ LỤC B: 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*

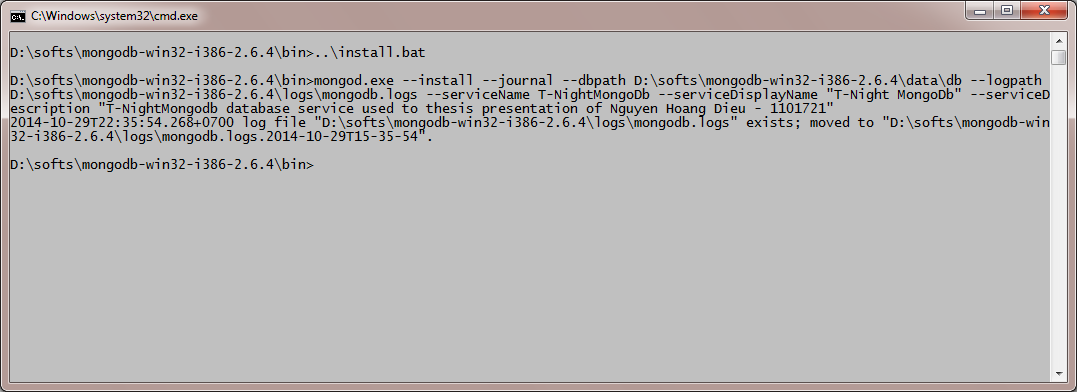
*--serviceDisplayName "T-Night MongoDb"*

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

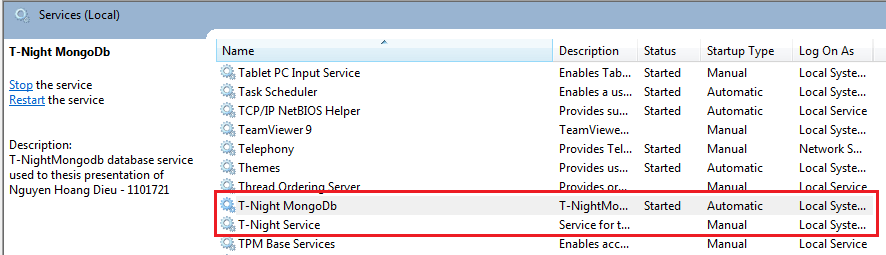
Giải thích tham số trong install.bat:

* *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.

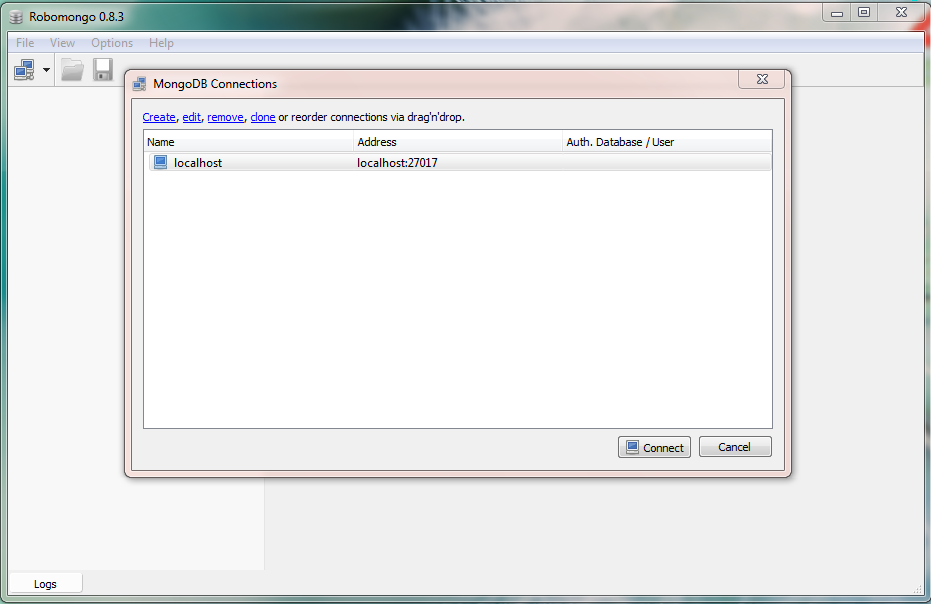


Như vậy, service mongodb đã được cài đặt thành công.

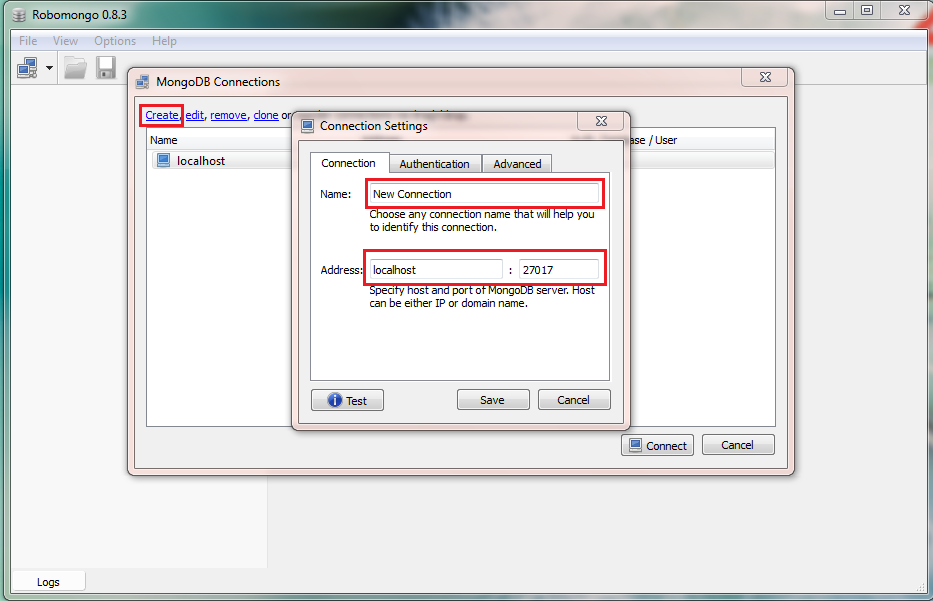


### TRUY VẤN

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.



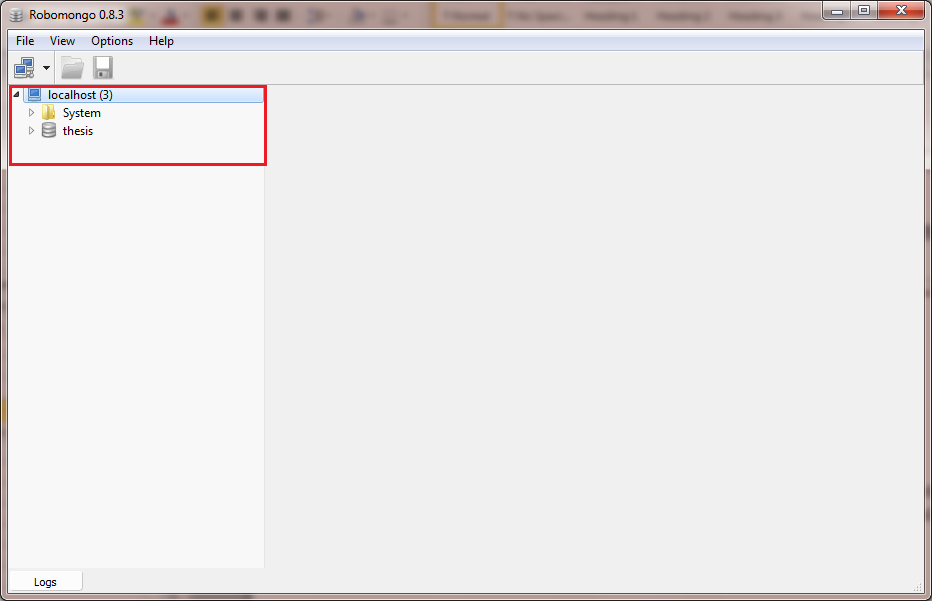
Giao diện tổng quát Robomongo



Tạo mới connection

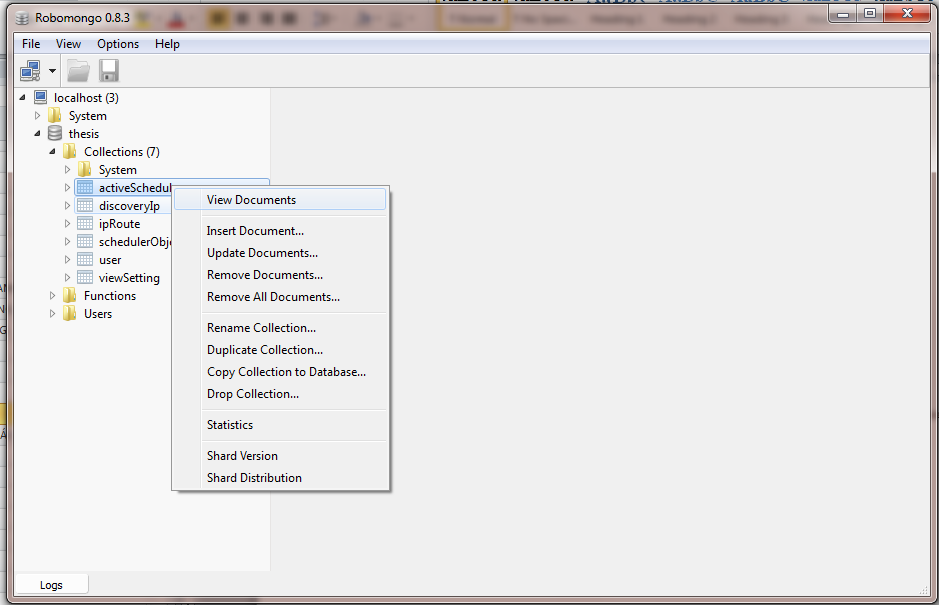
Để tạo kết nối đến MongoDB cần tạo mới 1 connection với 1 số thông tin như sau:

* 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.



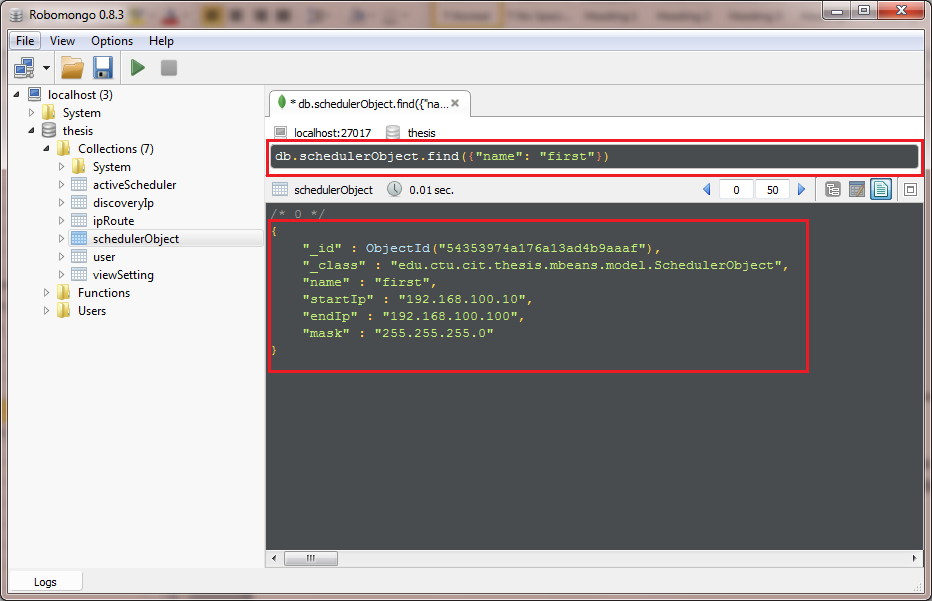
Giao diện kết nối thành công MongoDB

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



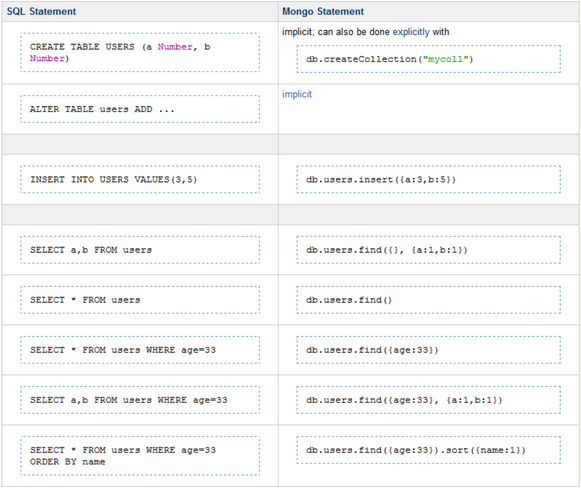
Hiển thị document trong Robomongo

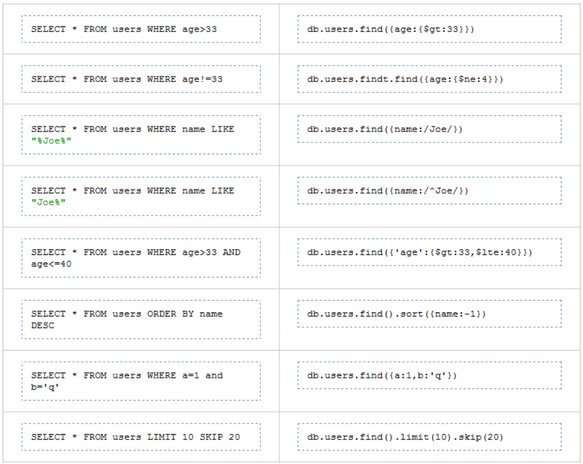
Sau khi hiển thị dữ liệu của 1 document nào đó, ta có thể thực hiện các câu truy vấn như CSDL quan hệ. Các câu truy vấn cần nhập vào khung truy vấn và nhấn f5 để hiển thị kết quả.



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.







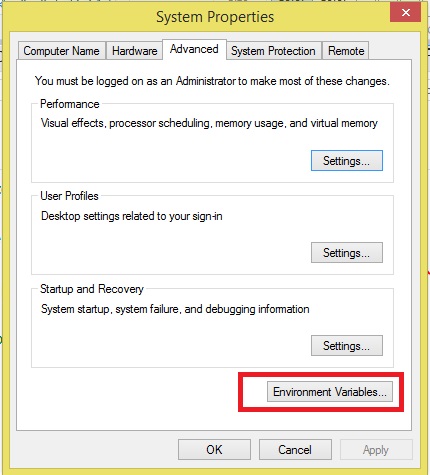
## PHỤ LỤC C: CÀI ĐẶT VÀ THIẾT LẬP MÔI TRƯỜNG JAVA

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.

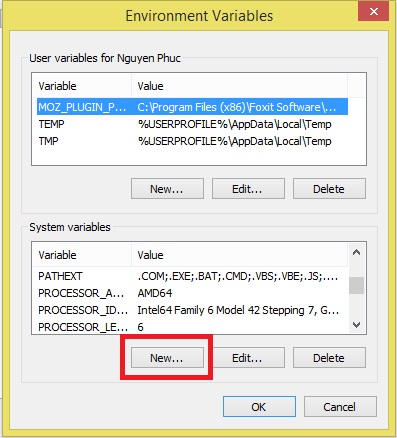
Bây giờ bạn tiếp tục thiết lập môi trường cho JAVA trên máy bạn. Bạn R-Click vào Mycomputer trên Desktop chọn Properties.

Kế đến chọn**Advanced System Setings**:

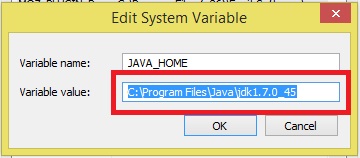
Trong Cửa Số System Properties (sau khi bạn chọn Advanced system settings) chọn Tab Advanced và chọn**Environment Variables….**

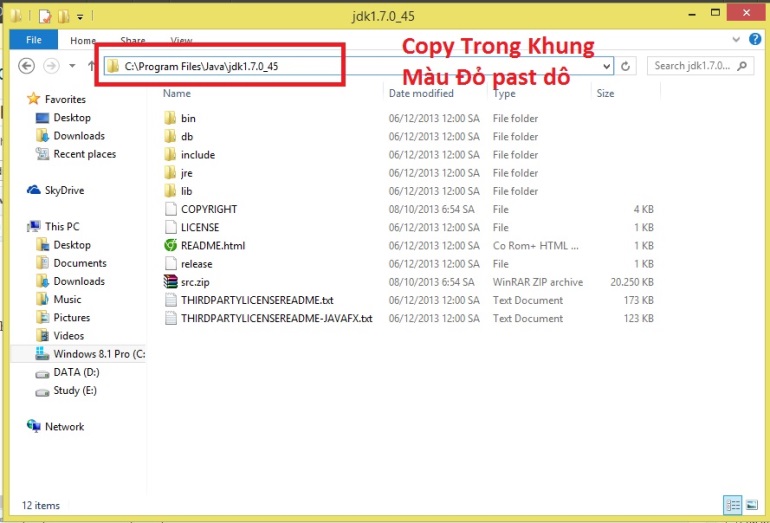
[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled2.jpg)

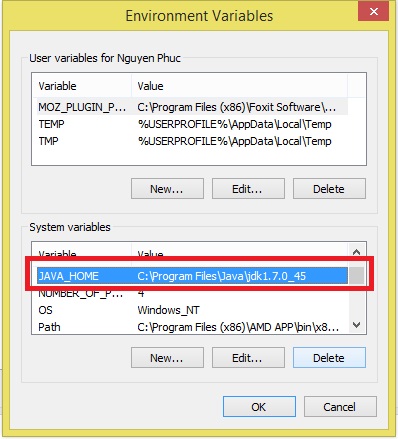
Trong cửa sổ**Environment Variables** chọn **New** như trong hình dưới:

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

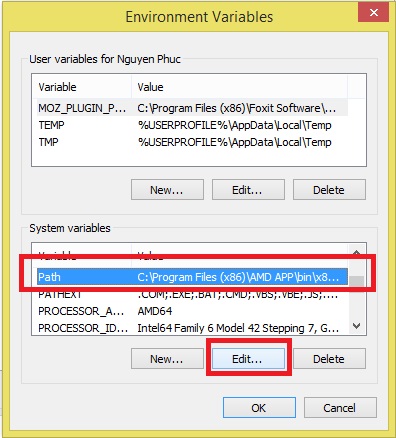
Thiết lập như hình bên dưới: Lưu ý phần trong khung đỏ để chính xác tuyệt đối bạn vào (thư mục cài đặt JAVA bạn thiết lập – ở đây tôi cài mặc định) C:\Program Files\JAVA\….. để xem

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

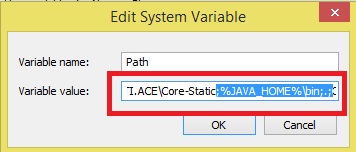
[](http://clbtinhoc.dntu.edu.vn/wp-content/uploads/2013/12/Untitled5.jpg)

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

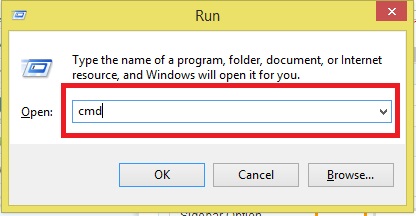
Bây giờ bạn cần tìm đến **Path** trong **System Variables** và chọn edit:

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

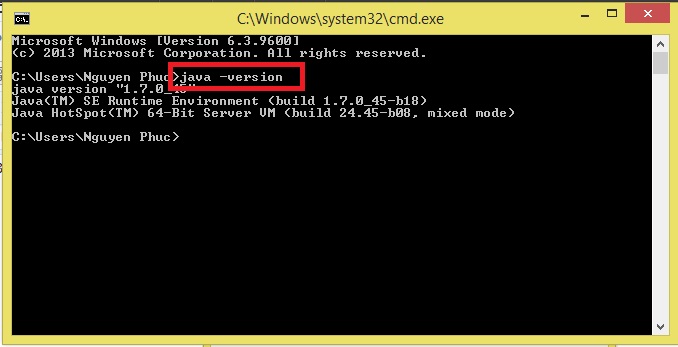
Trong cửa sổ **PATH** bạn **THÊM**vào sau những dòng sau: “ **%JAVA\_HOME%\bin”**

[](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)

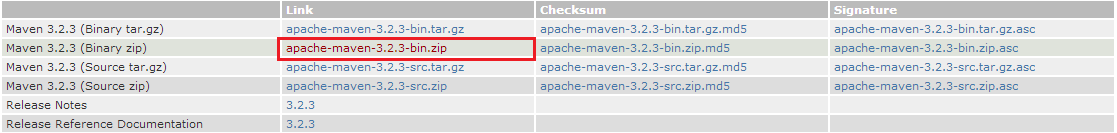
Của sổ CMD xuất hiện. Nhập lệnh “ **Java -version**” để kiểm tra

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

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

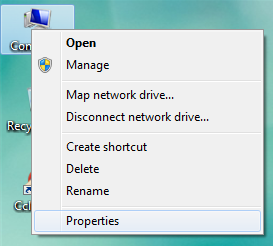
## PHỤ LỤC D: CÀI ĐẶT VÀ THIẾT LẬP MAVEN

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:

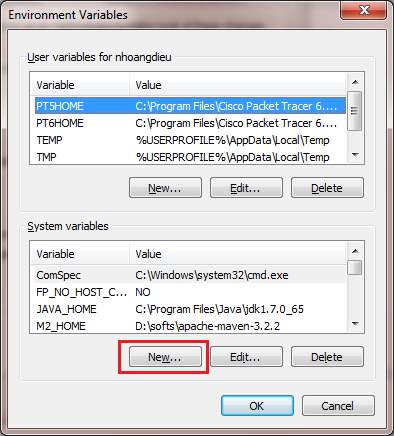


Sau khi tải về giải nén và tiến hành cài đặt biến môi trường:

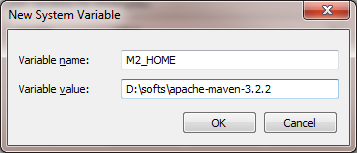
Chuột phải vào biểu tượng MyComputer 🡪 chọn properties,



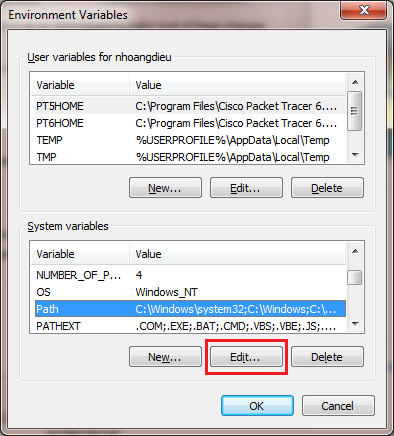
Tiếp theo chọn: Advance system settings 🡪 Enviroment variables, giao diện biến môi trường hiển thị. Chọn new:



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



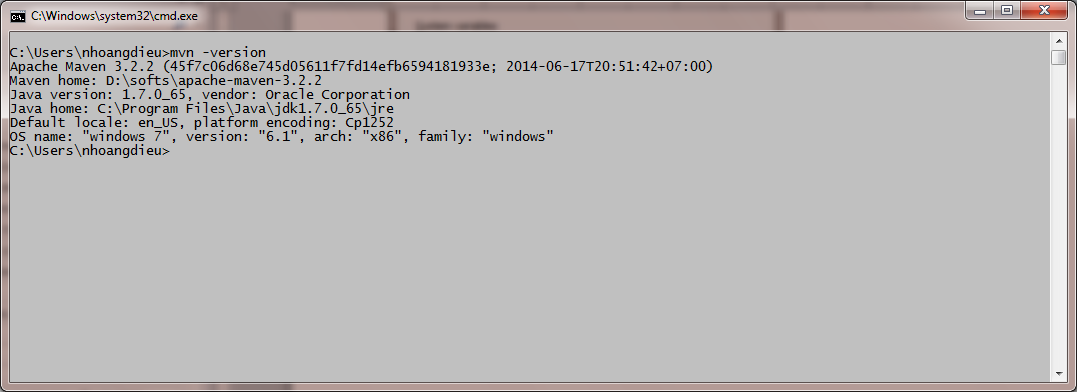
Tiếp theo, tìm đến path và thêm đường dẫn đến thư mục bin của maven:



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

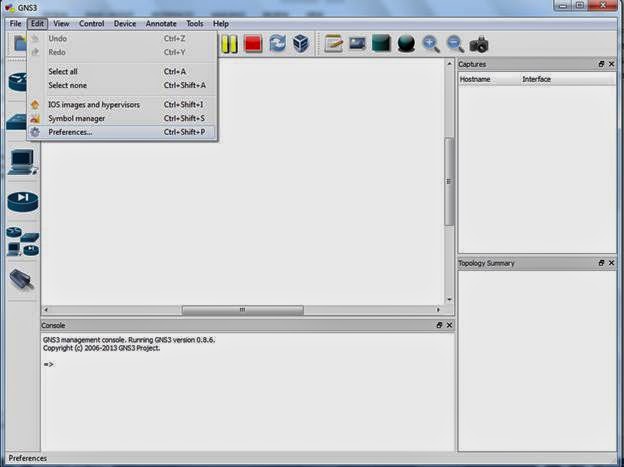
## PHỤ LỤC E: THIẾT LẬP MÔI TRƯỜNG ẢO TRÊN GNS3

### CÀI ĐẶT GNS3

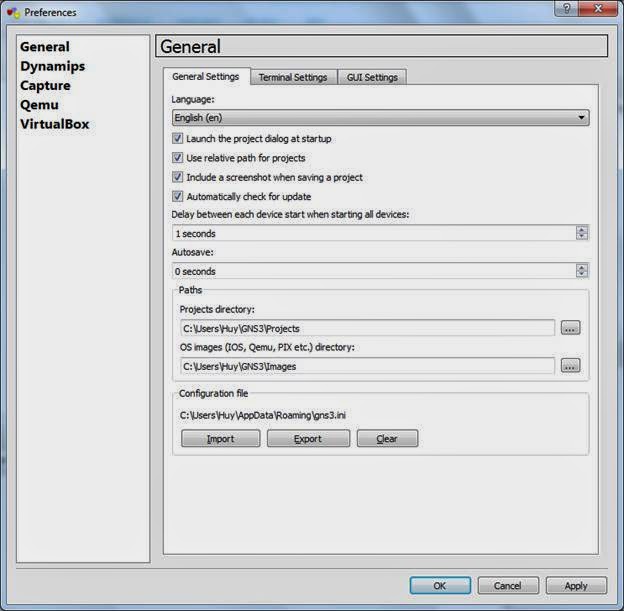
Download phiên bản mới nhât của GNS3 tại: [*http://www.gns3.net/download*](http://www.gns3.net/download)*.* Sau khi cài download phần mềm GNS3 tiến hành cài đặt như bình thường:

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…



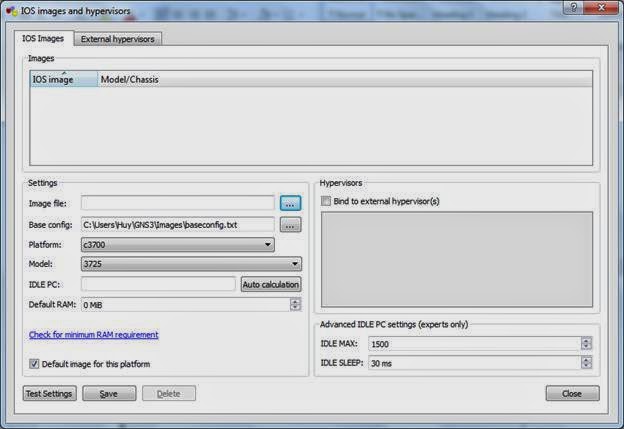
Chọn phần Dynamips bên cột trái.

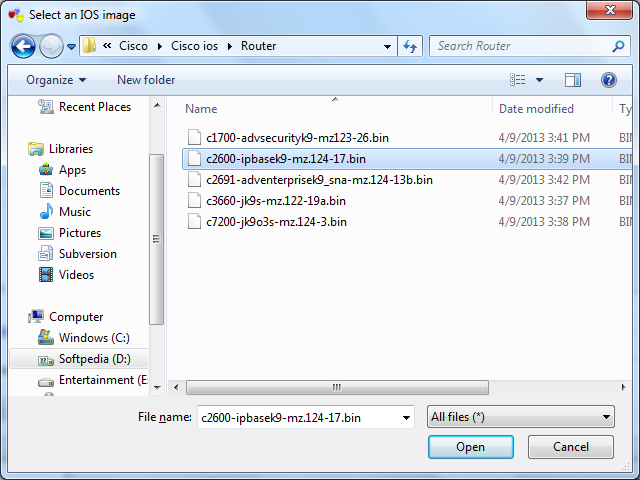


Chỉnh sửa các thông số bấm nút Test Settings để kiểm tra hoạt động, nếu chữ màu đỏ và báo fail thì phải reset máy và cài lại, với các bản GNS3 cũ khi cài lên Windows 8 thì sẽ không cài được WinPCap khi test sẽ lỗi, phải tìm các bản WinPcap cho Windows 8 rồi cài lại.

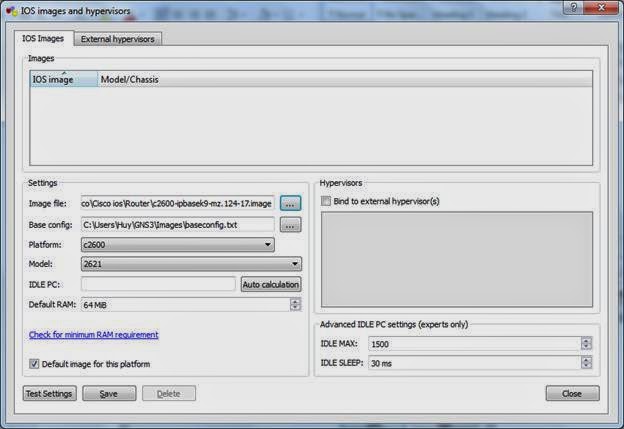
Để cấu hình IOS cho các Router chọn Edit 🡪 IOS image…

Browwse tới file IOS mà download về:





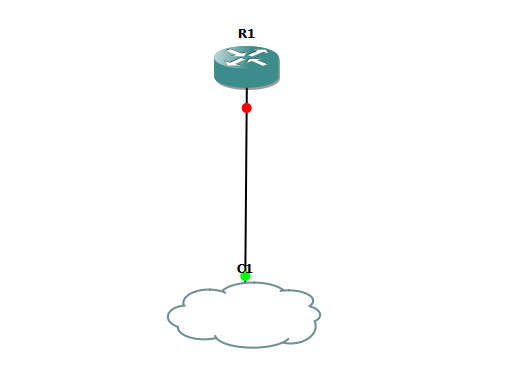
Sau khi chọn thì GNS3 sẽ tự động nhận diện đó là IOS của model router nào.



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.

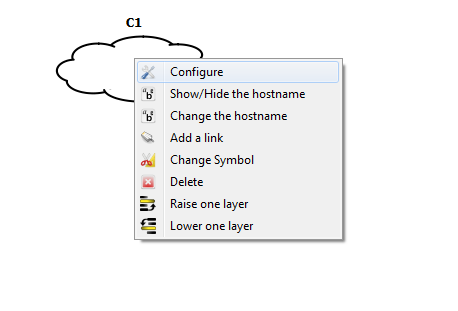
### CẤU HÌNH KẾT NỐI VỚI THIẾT BỊ THẬT

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

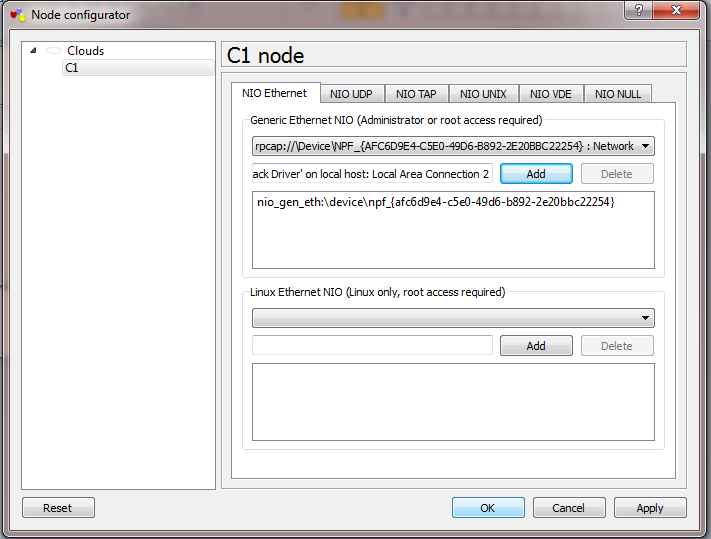


**Cấu hình cloud:**

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.



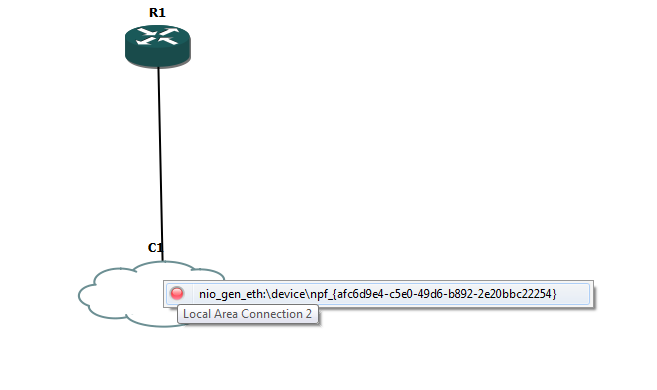
Lưu ý: ở bước này có một số lỗi phát sinh:

* 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:**

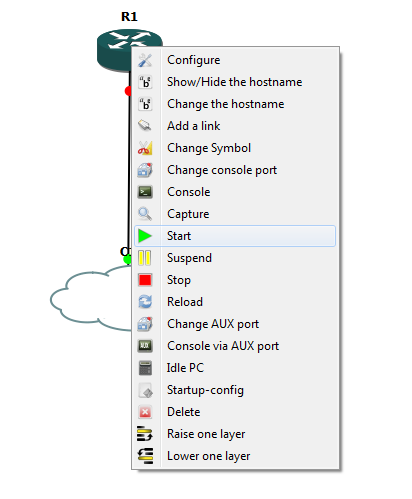
Router muốn kết nối được với mạng thật phải kết nối thông qua cloud, dùng link ethernet kết nối vào cloud.

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.

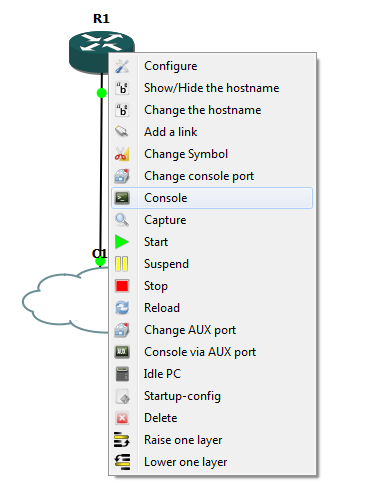


Chọn vào card mạng mà muốn kết nối nếu lúc cấu hình cloud có nhiều card mạng được add.

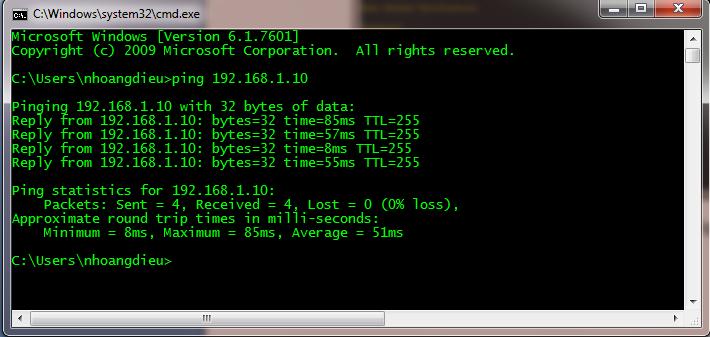
Start router và tiến hành cấu hình port cổng ethernet vừa kết nối.



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.



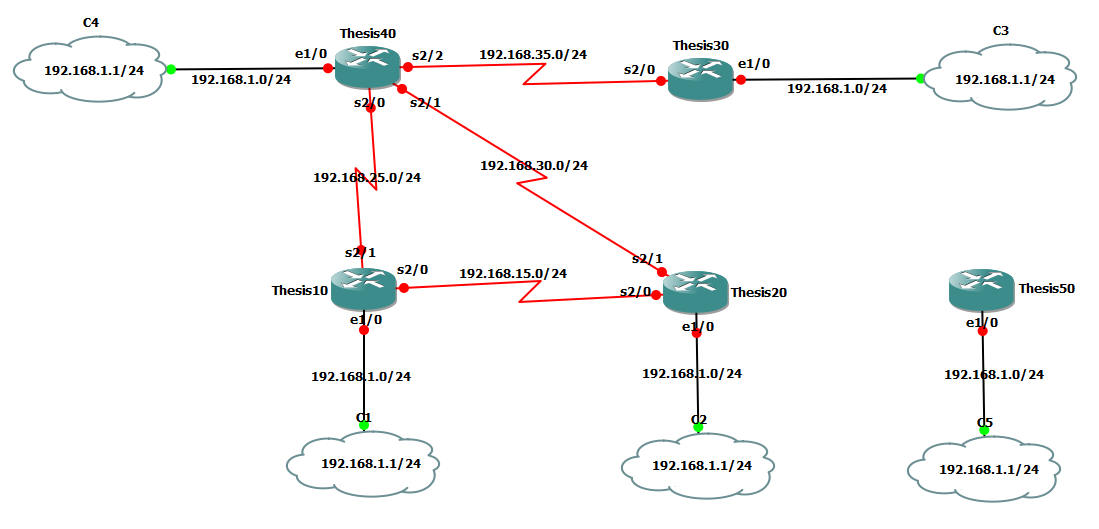
Bây giờ có thể dùng cmd của Windows ping vào ip router vừa đặt để kiểm tra kết nối router ra thiết bị thật được chưa.



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:



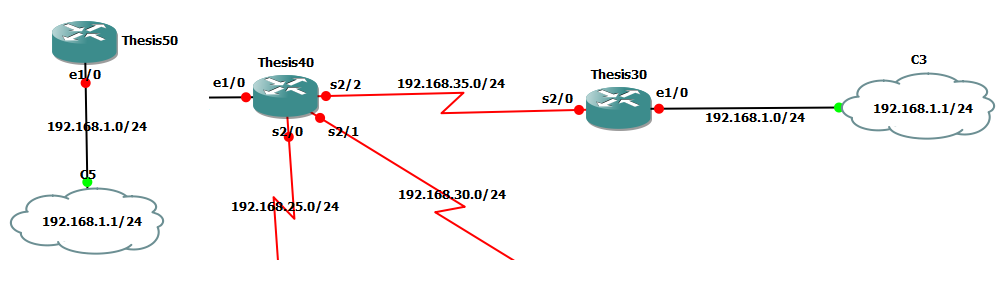
Mô hình gồm 5 router C7200 cisco để thử nghiệm:

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 F: 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)

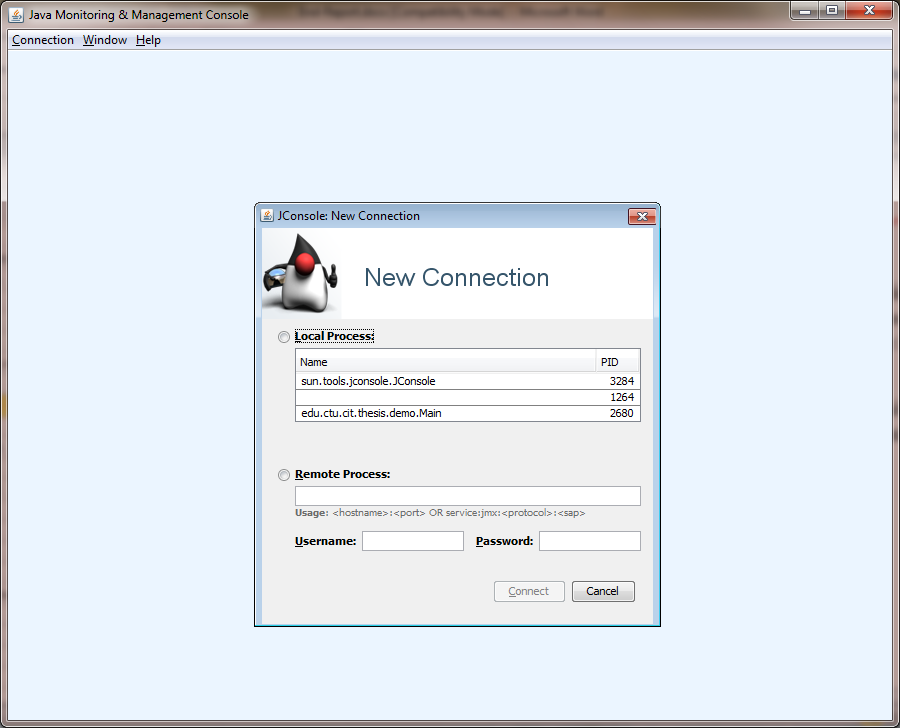
**mvn eclipse:eclipse**

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

*mvn eclipse:clean* – để clean project

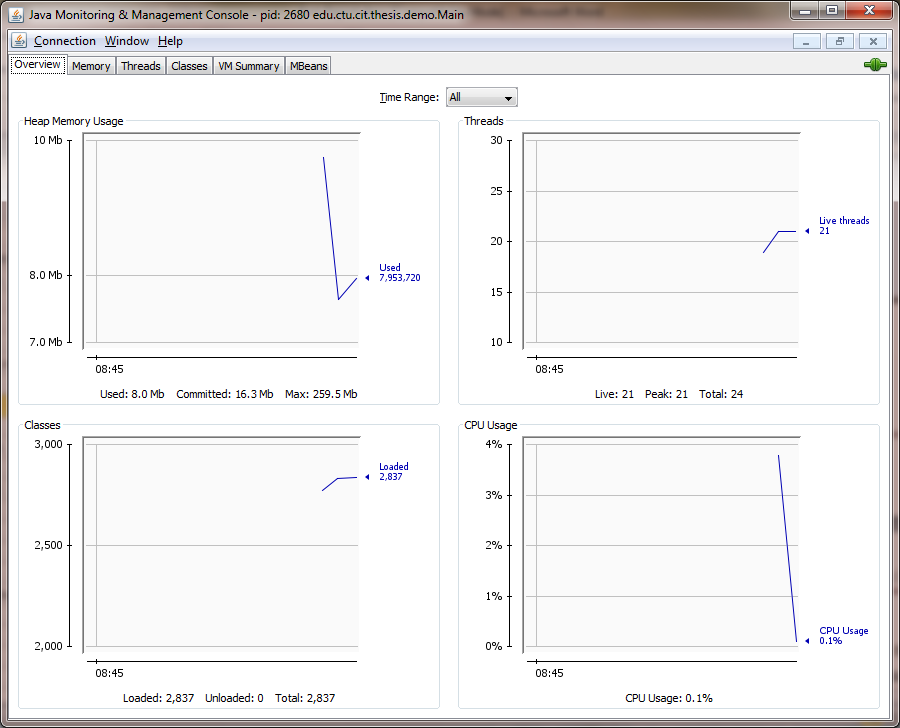
## MỤC LỤC G: SỬ DỤNG JCONSOLE ĐỂ THỰC THI HÀM TỪ XA

Với bản JDK hiện tại có hỗ trợ vài công cụ để kiểm thử các kết nối đến mbean server, jconsole *(%JAVA\_HOME%\bin\jconsole)* là 1 ví dụ cơ bản.

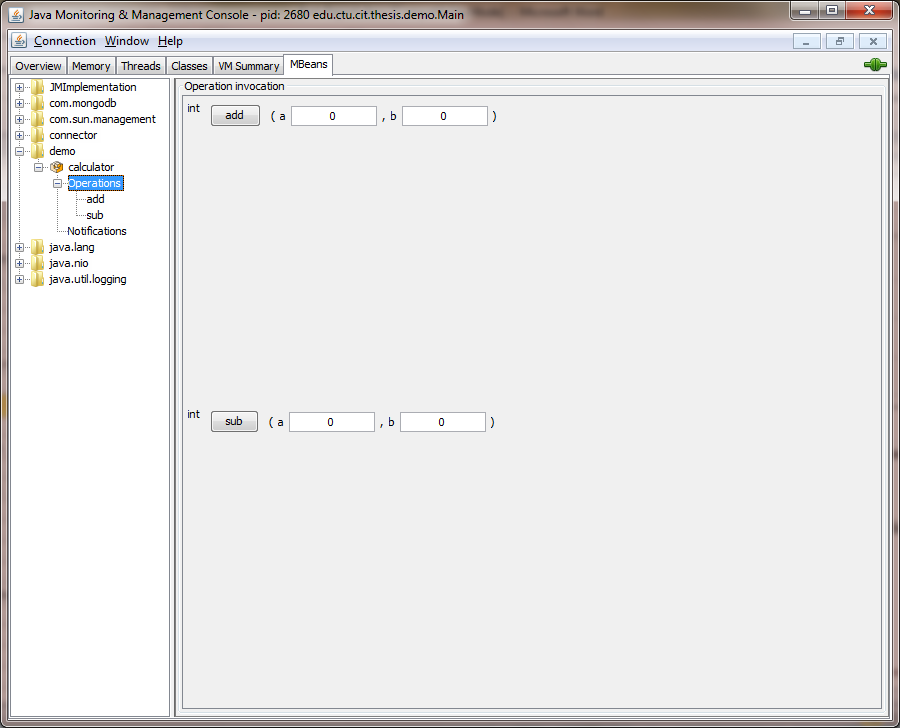


Khi run jconsole, nó sẽ tự động quét các mbean server trên máy cục bộ, nếu muốn thực thi một kết nối đến máy khác thì nhập url (định dạng jmx connector) vào khung “Remote Process” và bấm vào nút “Connect”.

Hình dưới là giao diện tổng quát sau khi kết nối thành công vào một mbean server. Có thể quan sát các thông tin tổng quát của một mbean server (mở rộng cho các project đang thực thi) như: heap memory usage, số lượng threads, số lượng class được load lên, phần trăm cpu sử dụng.



Để có thể nhìn thấy các object name đã được đăng ký và các hàm thì chọn vào tab “MBean”. Hình 2.7 là giao diện của tab mbean.



Phần bên trái là các object name của hệ thống và các object do người dùng định nghĩa và đang thực thi, bên phải là các hàm của 1 object name nào đó. Ở giao diện này, ta có thể thực thi các hàm đã được khai báo và kiểm tra giá trị.

