**Phiên bản tiếng việt**

**Author:**

# Mục đích tài liệu

Tài liệu hướng dẫn sử dụng: Framework Spring MVC Annotation– API web service restful.

# Phạm vi áp dụng

Bản thiết kế hiện tại dựng trên khả năng phát triển cho các ứng dụng web app dạng quản lí ứng dụng. Tuy nhiên project có khả năng mở rộng phát triển cho các ứng dụng web app dạng thương mại, chỉ cần cấu hình chuyển đường dẫn và mở rộng các thành phần là có ngay 1 app khác, với các khả năng xác thực và phân quyền khác nhau.

# Thuật ngữ viết tắt

|  |  |  |
| --- | --- | --- |
| Tên viết tắt | Tên đầy đủ | Ghi chú |
| WS | Web service |  |
| WA | Web application server |  |
| HQTCSDL | Hệ quản trị cơ sở dữ liệu |  |
| CSDL | Cơ sở dữ liệu |  |

# Lí do sử dụng

Tại sao nên sử dụng Framework

* Phù hợp với các quá trình khác trong quá trình phát triển hiện tại. Ngoài ra có khả năng mở rộng cao và tích hợp thêm các thành phần khác và tích hợp với các hệ thống khác dễ dàng mà không cần phải xây dựng lại từ đầu.

## Ưu điểm

* Mạnh mẽ, dễ phát triển, dễ bảo trì, sửa chữa, dễ cho lập trình do sử dụng các công nghệ hiện tại hiện đang rất phổ biến, nhiều tài liệu hướng dẫn code, fixbug như: **Spring MVC + Spring Security + Java + Jsp + jstl + Jquery + Javascript**.
* Hỗ trợ tốt cho bảo mật, sử dụng cơ chế xác thực hash bảo mật, có thể thay đổi thuật toán dễ dàng dựa trên nhu cầu mong muốn sử dụng. Sử dụng token cho WS. Có thể dễ dàng kết hợp nhiều cơ chế xác thực tùy vào nhu cầu.
* Dễ dàng mở rộng khi yêu cầu của bài toán lớn.
* Dễ dàng tích hợp với các ứng dụng khác.

## Nhược điểm

* Vì dung Java nên lượng code sẽ khá dài. Tuy nhiên việc này đã được khắc phục 1 phần trong framework có sử dụng design pattern, thiết kế đảm bảo sao cho việc code java là ít nhất và nhanh hơn có thể. Ngoài ra các IDE (Eclipse + Intelij IDA) support tốt việc recommend cho coder dễ dàng phát triển.
* Mỗi lần chỉnh sửa cần redeploy: Sử dụng tomcat cơ chế auto deploy, khi triển khai chỉ cần đưa lên là dk.

# Tại sao nên sử dụng mô hình MVC

Việc sử dụng HQLCSDL + CSDL là cần thiết và quan trọng đối với các ứng dụng. Chính ta không thể cứ lưu vào file và nó phức tạp và không linh hoạt. Do vậy từ đó cũng sinh ra mô hình MVC

Model + View + Controller

Model: 1 loại object được thiết kế để tương thích với table, view trong database.

View: Là nơi chứa các dữ liệu cần thiết và template theme gửi tới người dùng cuối.

Controller: Điều khiển tương tác giữa Model lấy dữ liệu gửi ra View.

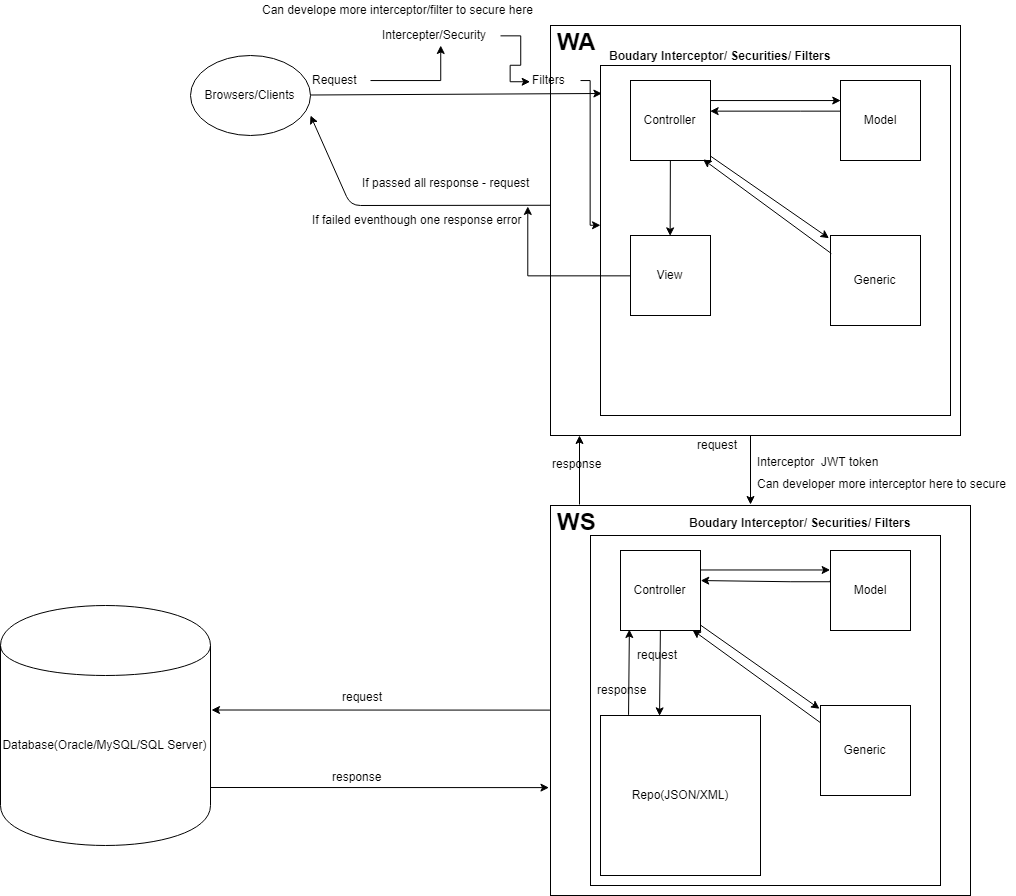
Hướng phát triển hiện tại: Đang sử phương pháp gọi là combine, nghĩa là kết hợp cả 3 (MVC) trong 1 cách thức gọi.

Lí do ta nên chuyển MVC: Khi chỉ muốn thay đổi giao diện, ví dụ người dùng không thích hoặc sau 1 thời gian dài muốn thay theme thì ta chỉ đơn giản thay đổi View thay vì thay đổi tất cả mọi thứ. Việc này tiết kiệm rất nhiều thời gian và công sức. Ngoài ra, việc thay đổi Controller Service cũng ko ảnh hưởng giao diện => tiết kiệm chi phí. Thay đổi model tương tự 🡺 tiết kiệm. Việc phân tách cũng dễ dàng khi phát triển hướng losing couple, các thành phần không ảnh hưởng tới nhau, khi dự án lớn, coder nào mạnh phần front end có thể phát triển riêng front end, coder nào mạnh phần backend có thể phát triển backend. Việc này cực kì là hiệu quả khi phân chia người.

* Vậy việc chuyển sang MVC là cực kì hợp lí ngay cả khi dự án không lớn (biết đâu 1 ngày nó cần lớn)

# Mô hình tổng quát

Mô hình tổng quan WA-WS-DB



# Mô hình chi tiết

## Web App Server

## Web App Service

# Công nghệ sử dụng

**Java + Spring MVC Annotation framework base + JSP + Jquery + JSTL + Javascript**

Các thành phần công nghệ và ngôn ngữ này hiện đang là phổ biến nhất với nhiều tài liệu, forum hỗ trợ code và fixbug, tránh dùng các công nghệ cũ và ít tài liệu support

# Hướng dẫn sử dụng chi tiết

## Web App Server

Sơ lược ứng dụng: Sử dụng Spring MVC annotation. Lí do, Spring support khá nhiều cho các programmer về cách thức sử dụng, khi dùng file cấu hình, annotation, hoặc kết hợp cả 2. Việc này đối với newbie đôi khi là một sự khó khăn trong việc kiểm soát, và khó thể hiểu. Vậy nên , ở đây quyết định chỉ sử dụng annotation cho việc phát triển ứng dụng. Và tại sao không sử dụng Spring boot, thường thì dự án phát triển web, sửa chữa khá nhiều, mỗi lần cần build lại, redeploy, sau mỗi lần như vậy sẽ mất khá thời gian. Nếu sử dụng Spring boot dạng application, ứng dụng cần tắt đi bật lại rất ngại, nên thay vì đó sử dụng cơ chế auto deploy của tomcat để nó tự rebuild, redeploy tiện cho quá trình phát triển không gây nhàm chán.

### Yêu cầu: Để làm được, chúng ta nên có hiểu biết về Spring MVC (đọc tài liệu web, có rất nhiều)

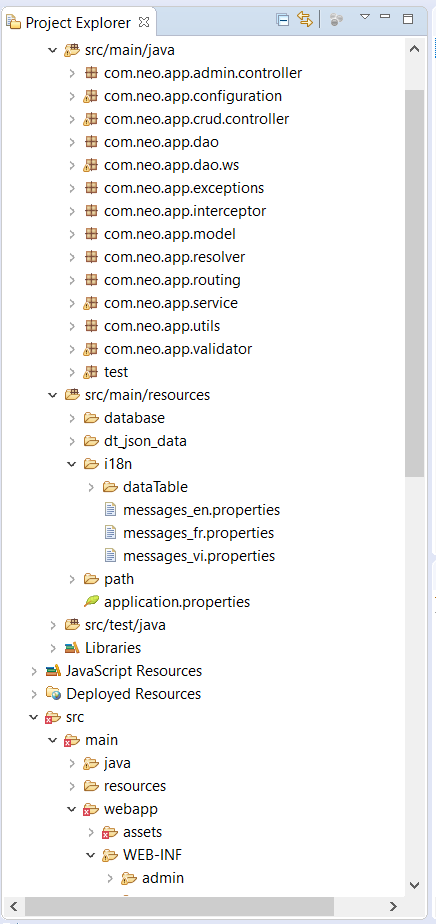
### Môi trường phát triển:

JDK + Eclipse Oxygen trở lên (Thường có tích hợp maven đủ hỗ trợ hoặc có thể lên chợ eclipse download plugin maven cho eclipse).

Trường hợp <User>/.m2 :chưa tồn tại, hãy tham khảo trên mạng và cài maven vào máy. Trang o7planning, support cực tốt.

Để sử dụng được cho thuận tiện và dễ dàng chúng ta nên dùng một vài IDE mạnh mẽ như Eclipse, Intelij IDEA. Hiện tại đang sử dụng Eclipse.

### Cấu trúc project



Thư mục src/main/java: Chứa các class về java cho việc tạo nên framework và phát triển thêm.

* Package …admin.controller: Chứa các điều khiển của admin
* Package …configuraton: Chứa các cấu hình cho toàn bộ ứng dụng.
* Package ….dao : Chứa các repository khi chúng ta sử dụng local repo bằng cách gọi trực tiếp tới database thì sẽ call ở đây.
* Package …dao.ws: Chứ các repo khi chúng ta sử dụng web service.
* Package …exceptions: Cấu hình các ngoại lệ khi muốn đẩy ra lỗi.
* Package …intercepter : Chứa các bộ lọc filter cho ứng dụng.
* Package …model: Chứa các thực thể cho ứng dụng
* Package …resolver: Chứa điều khiển cấu hình servlet
* Package …routing: Chứa các điều khiển cho việc mapping với nhiều database trong trường hợp dùng local repo.
* Package …service: Thành phần liên kết giữa controller ⬄ dao(repo).
* Package …utils : Chứa các tiện ích cần thiết cho ứng dụng

Thư mục src/main/resources:

* Database: chứa cấu hình cho database local.
* I18n: Cấu hình đa ngôn ngữ
* I18n/dataTable: Cấu hình đa ngôn ngữ cho dataTable
* Path: Cấu hình giao tiếp giữa url path ⬄ thư mục chứa View. Ví dụ url là menu/index.html và ta muốn để code tận đẩu đâu đó như admin/template/menu. Ta sẽ cấu hình như sau : menu= admin/template/menu
* Applications.properties: Chứa các cấu hình cho ứng dụng.

Webapp: như dynamics web thông thường mà ta biết.

Pom.xml : File chứa các import lib cần thiết cho ứng dụng, nếu chúng ta dùng thêm lib ngoài, search <tên lib> maven, copy đoạn command xml và paste vào đây.

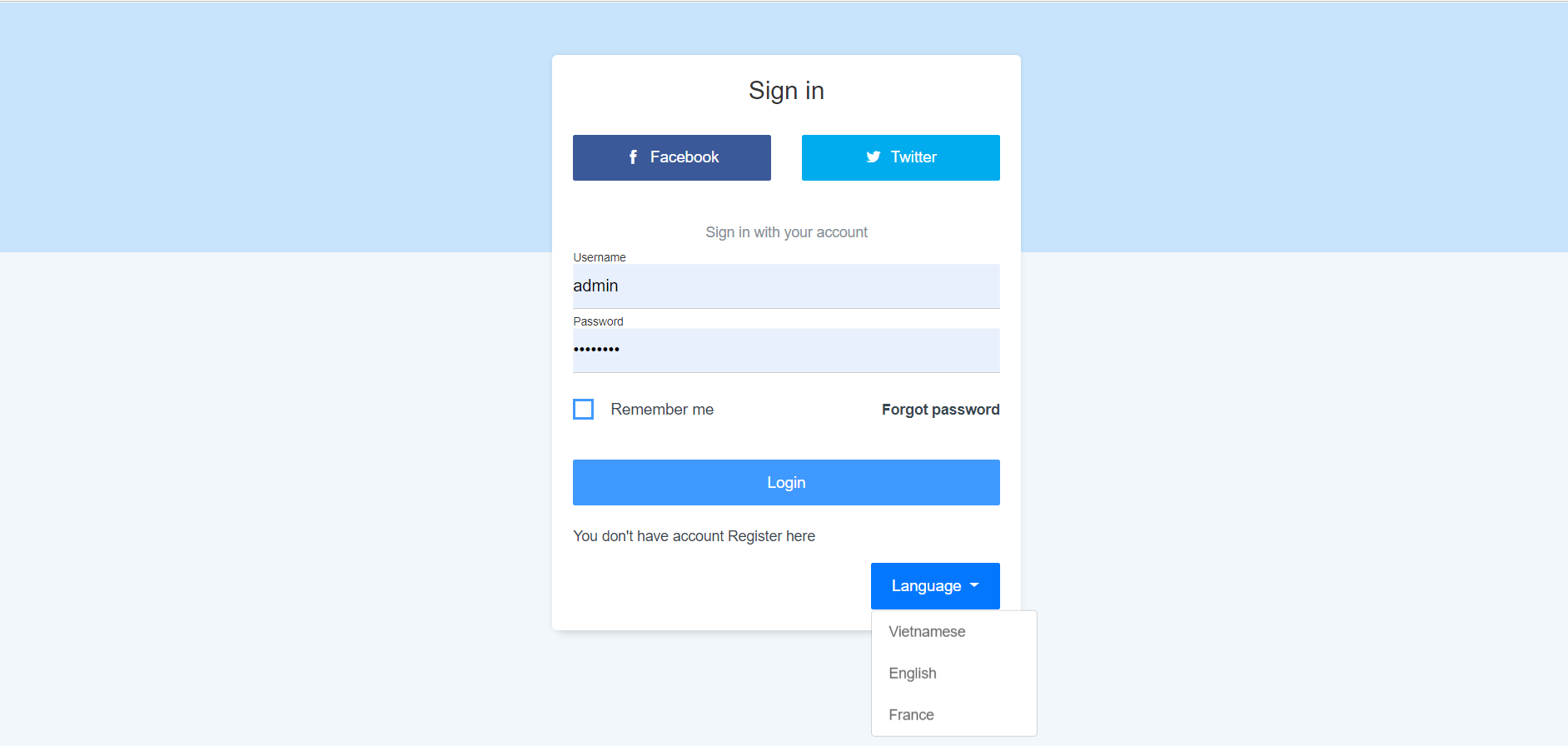
### Một vài hình ảnh mô tả template:

Hệ thống support: Đa ngôn ngữ, cookie ghi nhớ đăng nhập, cho phép đăng nhập 1 lần dùng lâu dài.

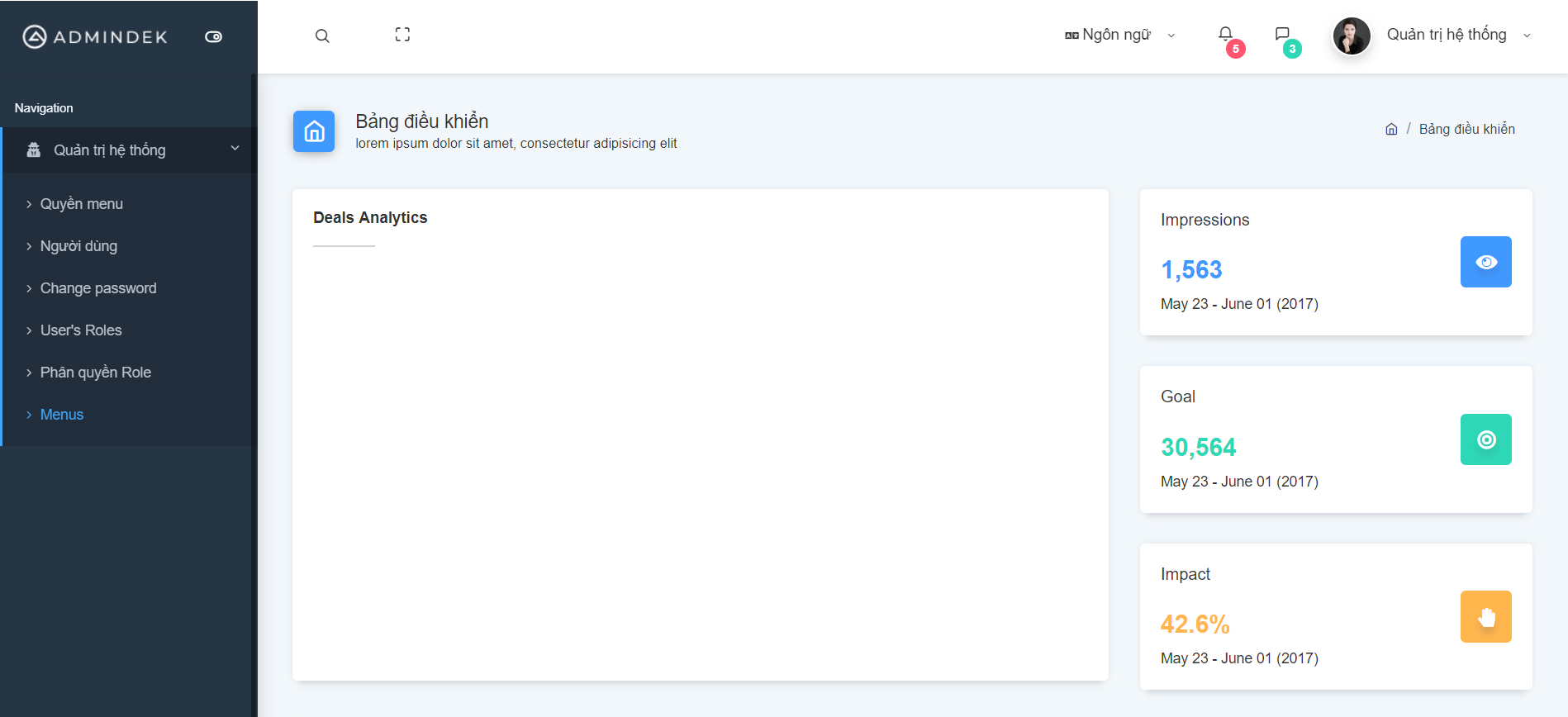
API tích hợp với social account(Facebook, twitter) có thể sẽ làm sau.

Hệ thống cho phép phân quyền menu, cấp quyền url path được xác thực, ngoài ra có thể xác thực mức sâu hơn dựa vào cấp độ user được phân quyền.

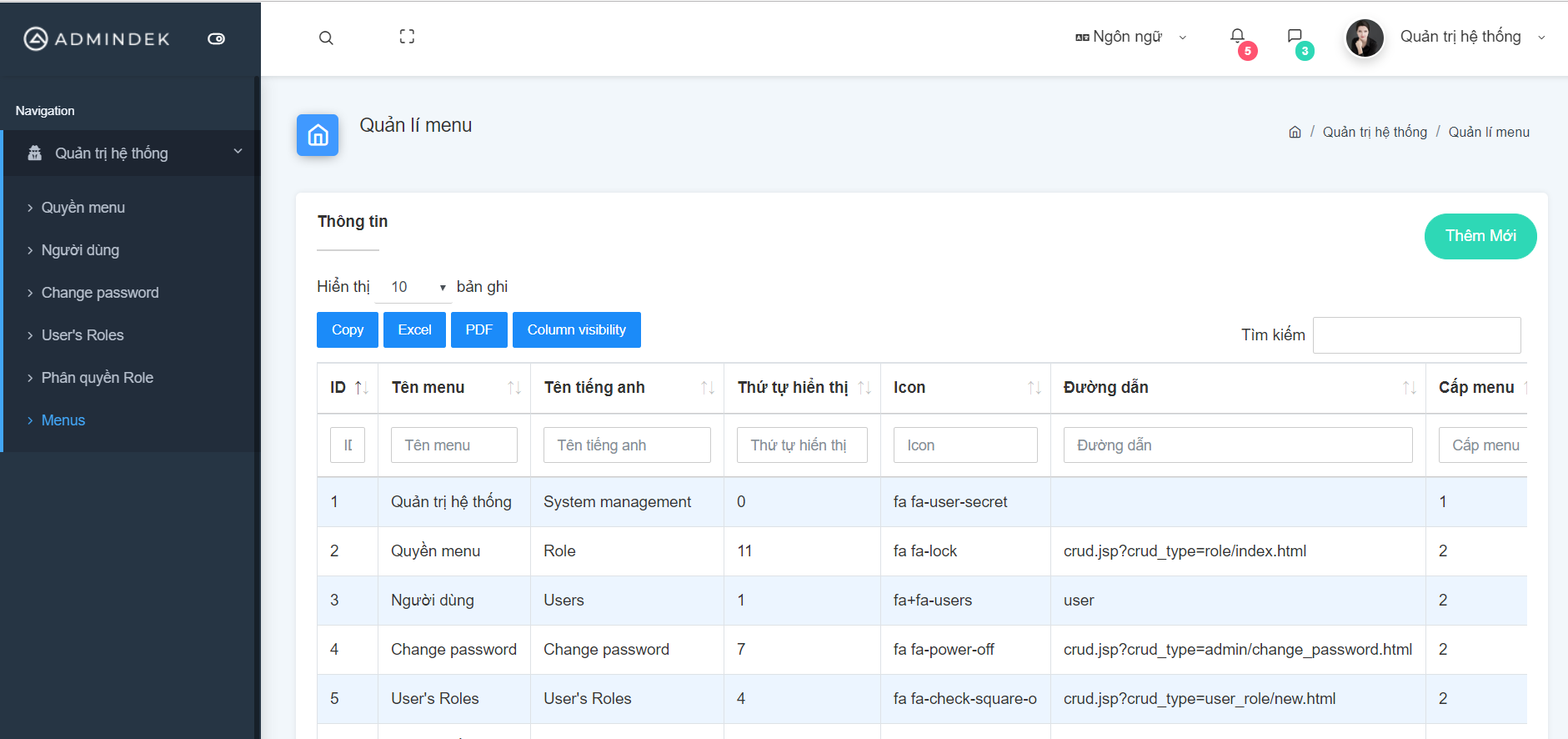
Login



Màn hình chính



Màn hình menu



### Giải thích code

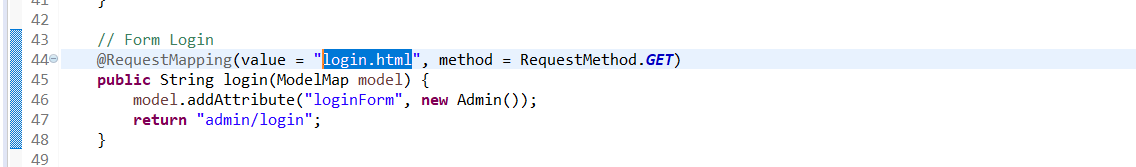
* Để hiểu được chúng ta cần đi từ đầu như sau:Có thể tóm lược sơ như sau. Khi spring khởi tạo bởi container, nó sẽ gọi hết vào 1 class trong package configuration như là gọi hàm main đầu vào ứng dụng. trong class này khai báo các yêu cầu cần thiết cho ứng dụng load tất cả các class được khai là Bean, configuration, annotation. Các file này sẽ chỉ tạo duy nhất singleton object trong toàn bộ container. Và chúng sẽ luôn là singleton trên toàn bộ chương trình, bất kể lúc nào ta gọi cũng chỉ 1 thể hiện(sẽ có điểm bất lợi ở đây, tuy nhiên điều này an toàn hơn cách chúng ta gọi thông thường). Vì sao lại là singleton?
* Cách chúng ta hiện tại là mỗi khi request lại khởi tạo thêm đối tượng => tăng heap => điều gì xảy ra. Có điểm tốt gì ở đây? Nhưng điểm bất lợi có nhiều hơn không.

Quay lại vấn đề: Sau khi load thành công mọi thứ ứng dụng running.

Bây giờ 1 request tới, theo như mô tả tổng thể, nó sẽ đi vào interceptor=> pass=> controller(mapping tướng ứng với url path gọi tới)=> trả ra kết quả.

Cùng xem chức năng login để hiểu thêm.

url path : neo/login.html = > tìm trong code sẽ thấy đoạn,



Ok, như vậy nó sẽ mapping vào đây trong file AdminController(tên file khong quan trọng, đặt tùy ý, nhưng mang ý nghĩa sẽ gợi nhớ, còn chính là annotation tên @requestMapping). Tiếp đó chính ta sẽ login.Gọi tiếp vào



Một tập mọi thứ check liên quan và lấy ra dữ liệu. Vậy từ đây hãy đọc tiếp để hiểu thêm cơ chế nếu chưa rõ.

Để biết gọi web service như nào hãy xem class FuncServiceWS . Nó là service gọi các repo. Trong này chứa các hàm để gọi tương ứng với web service.

* Method qryListModel: Gọi các câu lệnh sql dạng là query select statement.
* Method refListModel: Gọi các function, procedure tương ứng trong database.
* 2 dạng trên có thể trả về String json hoặc list<Map> hoặc list<Model>, tùy vào cách gọi vào (mong muốn dùng).
* Để kiểm tra cách truyền vào String, xem chi tiết cụ thể với CrudController khi truyền vào String. Kiểm tra cách truyền vào List<Model> kiểm tra trong lúc xác thực login thành công, lấy ra danh sách quyền, danh sách menu có sử dụng. Khi nào thì truyền được List<Map> , bất cứ khi nào muốn lấy resultset thành list<map>, nó tương đương với list<model> mà thôi.
* Method valPostParam : Gọi các function, procedure trong db và trả về là object.
* Method update: Gọi các câu lệnh dạng insert, update , và return 1 nếu thành công, và 0 nếu ko thành công.

Tìm hiểu thêm: hãy đi chi tiết vào màn hình menu như minh họa và xem chi tiết về cách gọi.

+ Lí do sử dụng datatable. Datable support cực tốt cho json data. Chỉ cần lấy dữ liệu từ server trả về(không nên trả về toàn bộ trang, sẽ giảm đáng kể lượng data từ server trả ra, traffic sẽ giảm đáng kể).Vậy tại sao không dùng sức mạnh của js client thay cho server, sẽ giảm đáng kể traffic vô nghĩa.

Để xem chi tiết về dataTable, xem chi tiết luồn lấy ra danh sách menu. Bắt đầu từ file trong thư mục WEB-INF/admin/menu/index.jsp.

Nó sẽ gọi tới @RequestMapping(value = "{path}/index.html", method = RequestMethod.GET). Để load ra giao diện của trang, sau đó xem Script call ajax nó sẽ gọi tiếp tới @RequestMapping(value = "{type}/dt\_ajax.html", method = RequestMethod.GET) để lấy dữ liệu. Hãy xem chi tiết mục này để hiểu thêm.

+ Recommend: Đối với các table không quá 500 records: Không cần tốn công tạo ra các mục search giao diện để mỗi lần request lại, thay vào đó, trả toàn bộ và cho client xử lí việc này, DataTable support cực tốt việc này. Vậy nên ta thấy với các menu dạng thuần CRUD, dữ liệu không lớn đa phần nhỏ hơn 500 bản ghi, thì chỉ cần đổ dữ liệu, và từ đó tương tác toàn bộ là client cho các nhiệm vụ search.

Đối với các dữ liệu có số bản ghi trên 500, khi đó dùng điều kiện lọc dữ liệu chuẩn là theo thời gian : Từ bắt dầu đến kết thúc, việc này sẽ giới hạn dữ liệu tránh overload. Ngoài ra, mỗi màn hình tùy vào mục đích sử dụng của khách hàng sẽ cho thêm các điều kiện lọc khác nhau.

Khi muốn thêm mới hoặc cập nhật dữ liệu. Lấy menu là ví dụ, tìm đoạn script thêm mới(trước khi thực hiện action ta sẽ cần validate dữ liệu ở client trước rồi, xem các hàm validate ở script), ta sẽ thấy nó call vào @RequestMapping(value = "crud\_ajax/{type}", method = RequestMethod.POST). Từ đây dữ liệu sẽ được ghi vào WS và ghi vào DB. Đây là ví dụ demo, nên gọi vào đây.

Thực tế để làm nhiệm vụ validate tốt hơn, chúng ta cần validate ở server nữa. việc này sẽ phát sinh thêm code Java (việc này là cần thiết) và nó sẽ giúp bảo vệ dữ liệu tốt hơn. Nếu làm theo cách này ta nên dùng với mỗi chức năng thêm mới hoặc cập nhật tạo ra 1 @RequestMapping riêng cho nó để tiện cho việc validate dữ liệu, khi đó hàm hoặc câu lệnh sql chỉ làm nhiệm vụ store dữ liệu tránh validate nhiều sẽ giảm được traffic của DB. Recommend gọi như sau: @RequestMapping(value = "menu/{type}", method = RequestMethod.POST)

public String actionValWithValidateServer(ModelMap model, @ModelAttribute("menu") Menu menu,

@RequestParam Map<Object, Object> params, @PathVariable("type") String type, HttpServletRequest request,

HttpServletResponse response, HttpSession session,BindingResult errors) {

Ta thấy ở trên tham số @ModelAttribute("menu"), menu là id của form, và thuộc tính của nó chính là thuộc tính của model Menu, BindingResult error, sẽ đẩy ra lỗi khi chúng ta validate có lỗi xảy ra. Có thể xem phần login xác thực để hiểu thêm cơ chế này.

## Web App Service

Việc phát triển web service đã gần như hoàn thiện, nên chỉ cần cấu hình gọi các mapping sql statement trong thư mục static/sql.properties là đủ. Cơ chế là key=value. Mapping 1-1 giữa web serice và statement trong SQL.

Xem ví dụ sau :

**edit\_menu= select edit\_menu(?,?,?);**

**edit\_menu.param=id,name,detail**

Ta thấy ở đâu là key: edit\_menu, và value là select edit\_menu(?,?,?). Chúng ta nên đặt tên gợi nhớ key = value tương đương nhau để so sánh ngay với statement trong SQL dễ dàng, tiện cho việc tracking lại để sửa lỗi.

Dòng lệnh thứ 2 nghĩa là, số parameter ta cần truyền tương ứng vào trong hàm là 3 tương đương với 3 tham số tên, Vậy việc ở client là chúng ta cần truyền đúng name của 3 tham số này.

Để hiểu thêm cách gọi, hãy xem ví dụ về Quản trị hệ thống => Menu, ta sẽ thấy mục tìm kiếm menu sẽ tuân theo qui tắc trên.

# Tool kèm theo

Khi làm theo đội nhóm phát triển ứng dụng, để tránh tình trạng code không được up-to-date peer to peer, chúng ta nên dùng các tool quản lí code như github(source tree, git tortoise) để quản lí code được dễ dàng, tránh code lẻ tẻ, khi merge vào nhau sẽ phức tạp, không đồng nhất gây mất thời gian, tốn tài nguyên.

# Yêu cầu phi chức năng

Theme đi kèm chứa các style css và js lib đi kèm mỗi page, cần sử dụng gì, ta sẽ xem mẫu và bổ sung tương ứng, để tương thích với



# Remind

Hiện tại vẫn đang phát triển nên có thể gặp nhiều lỗi, hi vọng được đóng góp nhiều hơn.

Trên đây là 1 vài note về cách sử dụng