**Mục Lục**

[1. Giới thiệu công nghệ trước đây khi lập trình web với java : jsp -servlet 1](#_Toc511440724)

[2. Giới thiệu Spring MVC Framework 2](#_Toc511440725)

[3. Kiến trúc Spring Framework 2](#_Toc511440726)

[5. Thư Viện : 6](#_Toc511440727)

[6. ServletDispatcher : Bên trong web.xml : ServletDispatcher 6](#_Toc511440728)

[7. Controller : 7](#_Toc511440729)

[8. Tạo project hello world : 10](#_Toc511440730)

[9. Lấy thư viện : http://repo.spring.io/release/org/springframework/spring/ 10](#_Toc511440731)

[10. Tạo controller : HelloController 10](#_Toc511440732)

[11. Tạo view : 11](#_Toc511440733)

[12. Truyền dữ liệu : Controller => View 11](#_Toc511440734)

[13. Ánh xạ nhiều action : 11](#_Toc511440735)

[14. Ánh Xạ ( Rõ hơn ) 11](#_Toc511440736)

[15. Làm việc với tham số : 4 phương pháp 12](#_Toc511440737)

[16. Làm việc đối tượng web ( Web Object ) : 13](#_Toc511440738)

[17. Action Results : 14](#_Toc511440739)

[18. Phân Biệt Forward và Redirect : 14](#_Toc511440740)

[19. Layout : 14](#_Toc511440741)

[20. Thủ Thuật BaseController : 16](#_Toc511440742)

[21. Trang welcome : mặc định index.jsp 17](#_Toc511440743)

1. **Giới thiệu công nghệ trước đây khi lập trình web với java : jsp -servlet**
   * Khó khăn do lập trình ( code ) không theo một quy luật nào.
   * JSP, Servlet : có thể tổ chức theo MVC, MVVM tùy sở thích mỗi người và tùy thuộc vào mỗi dự án, không có ràng buộc theo 1 quy chuẩn nào.
   * Khó quản lý mã nguồn ( code ), khó bảo trì, lộn xộn.
   * Ở một số cty có sử dụng jsp-servlet để xây dựng website, tuy nhiên mỗi công ty có thể có cách tổ chức structure các nhau.

=> Ra Spring MVC Framework :

* + - Để lập trình có quy chuẩn, quy trình hẳn hoi.
    - Làm việc chặt chẽ, dễ quản lý code.
    - Dễ nắm bắt cho dù nhận sự có thay đổi.
    - Teamwork hiệu quả, tốc độ nhanh hơn, tái sử dụng.
    - Có sẵn khung sườn chỉ việc sử dụng lại => giảm chi phí thời gian phân tích.

1. **Giới thiệu Spring MVC Framework**
   * Ra đời sau Structs ( hiện tại structs 2 ).
   * Trước đây có 1 nền tảng JavaEE : lập trình mô hình đa tầng ( enterprise java bean ) cho các ứng dụng lớn. Áp dụng ứng dụng nhỏ thì quá nặng. => ứng dụng lớn ( dành cho doanh nghiệp lớn ), Nhỏ dùng JSP, Servlet.
   * Spring MVC ra đời để giải quyết ứng dụng có mức độ trung bình. Sử dụng rộng rãi nhất hiện nay ( phiên bản mới nhất 5 ( 12-2017 ) )
     + **Nhỏ, Trung, To** : số lượng người truy cập, doanh nghiệp đặt ( doanh nghiệp lớn là doanh nghiệp đặt trên toàn cầu ), khả năng tương tác của nhiều người.
     + **Lớn :** không phải là nhiều class, code mà là 1 trang web có 100 triệu người vào, không lag => lớn. Khả năng tải của người dùng, tần suất sử dụng.
   * Viết 1 website, 1 tương tác là tomcat chạy. Enterprise Java Bean không chỉ tomcat, jonas, jboss, glassfish ( ở mức test – không phải mức triển khai – không đủ mạnh triển khai ) cài thêm dịch vụ để xử lý tương tác người dùng. Tomcat không phải là JavaEE Server.
   * Spring MVC chỉ cần tomcat ( 1 server duy nhất – có thể là 2 ( CSDL ) ) – người dùng sẽ tương tác tomcat.

2.1 : Spring là 1 framework chung : có thể viết app, web. Web : Spring MVC.

\_ Chung là 1 core chung. Chỉ tập trung web => gọi là Spring MVC.

\_ Tìm hiểu dự án Spring, bằng tay trước khi dùng **Maven**, có controller, view, cấu hình, model, kỹ thuật ánh xạ ( Request Mapping ) – action – POST|GET – tham số.

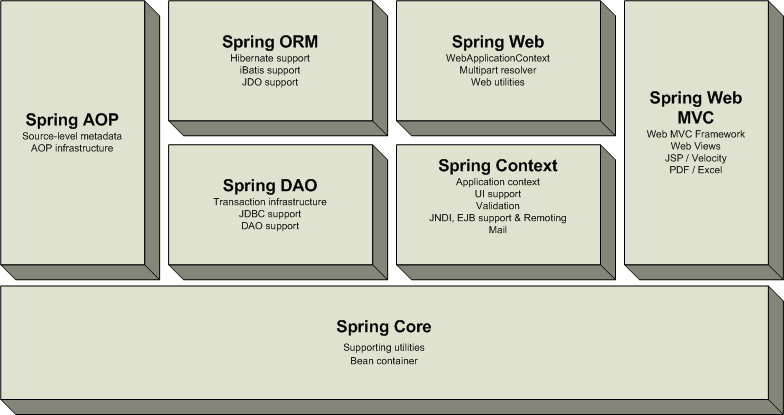
\_ Nhận dữ liệu từ người dùng.

\_ Làm việc với đối tượng web : Request, response, session, application, cookie.

1. **Kiến trúc Spring Framework**

\_ Spring Core, Spring AOP ( lập trình hướng khía cạnh : listener, filter, viết ra để chạy ở 1 thời điểm nào đó, Sẽ học Lá chắn Interceptor để bảo vệ trang admin ) , Spring ORM ( hibernate ), Spring Web ( structs ), Sptring DAO ( transaction, lập trình cơ sở dữ liệu nếu ko xài hibernate : JDBC ), Spring Context, Spring MVC. Chạy chỉ cần file JAR.





1. **Xử lý tương tác : khi người dùng yêu cầu .**



\_ Khi người dùng gửi yêu cầu ( mọi yêu cầu ) sẽ có 1 servlet tiếp nhận : **DispatcherServlet** ( Có sẵn ). Tất cả mọi request từ người dùng DS sẽ nhận hết. DS sẽ điều hành qua các bước 1, 2, 3, 4 để làm việc.

B1 : Sau khi có 1 yêu cầu HTTP Request .Nhận yêu cầu, đưa xuống **Handler Mapping** phân tích yêu cầu vàtrả kết quả về, DS sẽ biết gọi phương thức nào trong class ( controller ) nào.

B2 : Sẽ gọi phương thức trong controller tương ứng. Phương thức sẽ trả về 1 kết quả về cho DS. DS tiếp tục phân tích.

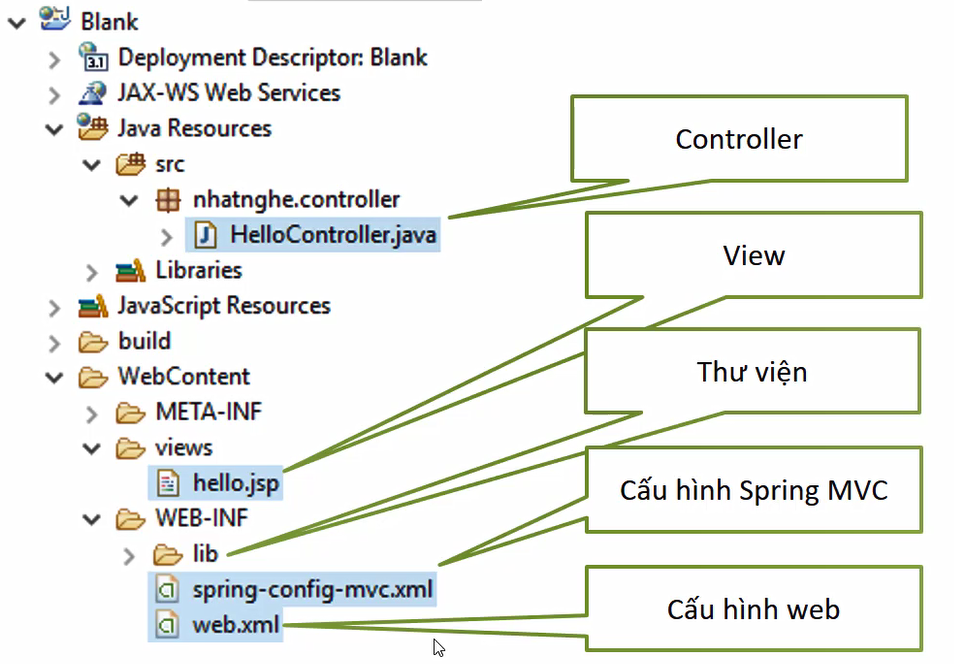
B4 : Gửi kết quả qua cho View Resolver => VR sẽ tìm view nào cần thiết để hiển thị cho người dùng. Trả kết quả qua cho DS.

B5 : Nhận kết quả, sau khi view đã hoàn tất ! DS truyền dữ liệu của model vào view ! cuối cùng hiển thị ra trình duyệt.

Ta cần cấu hình DispatcherServlet biết : Controller ?? View Resolver ?? Handler Mapping và View Resolver có sẵn.

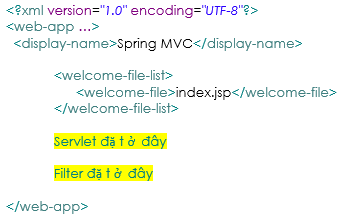
**Cấu hình là tâm của công nghệ ( XML )**

* Rắc rối quy trình, không phải mặt lập trình.



Web.xml : khai báo ServletDispatcher cho hệ thống biết. Cho dispatcher biết controller, view, viewresolver ?

Trong spring filter utf-8 có sẵn ! chỉ cần khai báo !

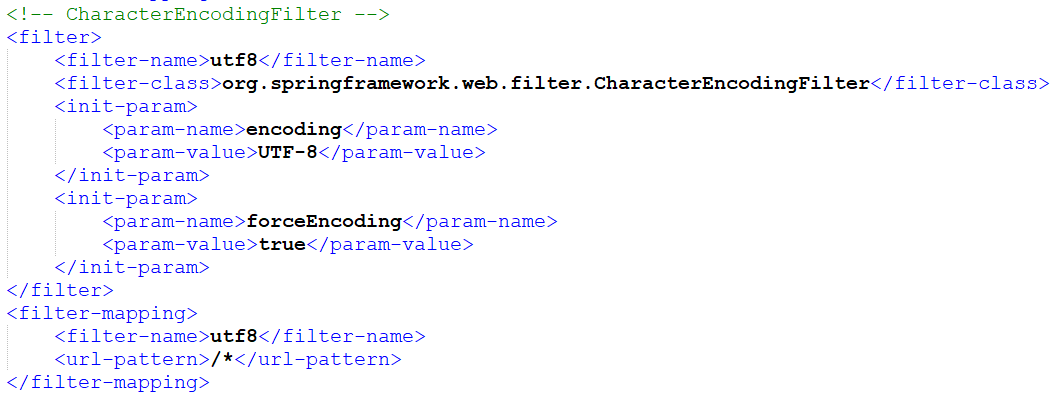


1. **Thư Viện :** 
   * Java Servlet API
   * Spring Web MVC
   * Junit
   * Hibernate ( tại trang chủ )
2. **ServletDispatcher :**   
   Bên trong web.xml : **ServletDispatcher**

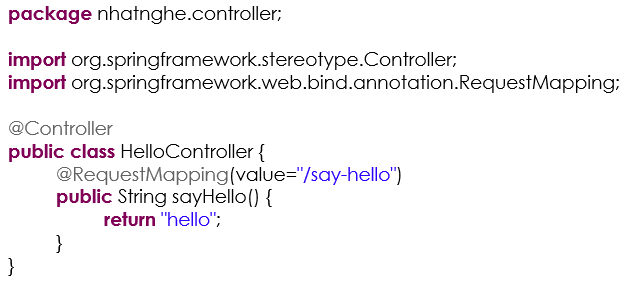


\_ .htm là gọi trang web .htm thì nó sẽ chạy !

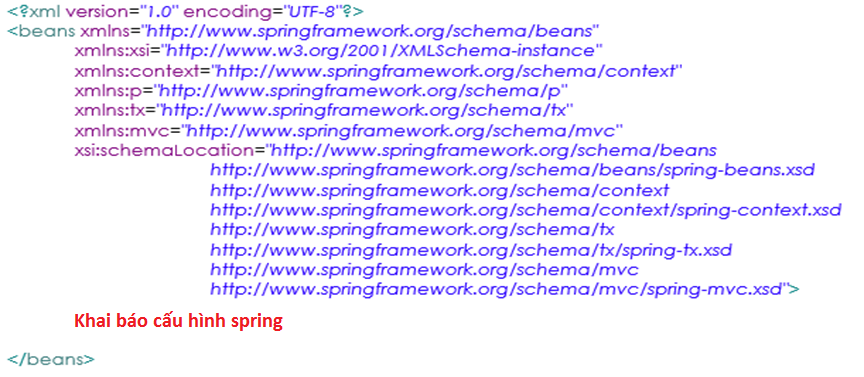
\_ /WEB-INF/spring-config-\*.xml ( \* là gì cũng được – đuôi xml ) : chỉ ra chỗ tất cả file cấu hình.



1. **Controller :**



* + Trước đây trong servlet chỉ có doGet, doPost.
  + Là 1 class tên bất kỳ - được ánh xạ.
  + Gõ địa chỉ : tên-project/say-hello.htm thì phương thức sayHello sẽ chạy.
  + Trả về view là hello. ( Trước đây là forward và chỉ đầy đủ tên bao gồm .jsp) // đuôi htm do mình khai báo.
  + Để hệ thống nhận biết được **hello** là file hello.jsp ta cần phải cấu hình **( quan trọng )** .





\_ Lưu ý : ta phải cấu hình để khi return về view : hiểu được tên view bao gồm : nằm ở đâu, có đuôi là gì . Ngoài ra : khai báo thư mục controller . Chỉ cần hiểu các dòng cấu hình.

**Bổ sung để khai báo được annotation-driven :**

*xmlns:mvc="http://www.springframework.org/schema/mvc"*

*http://www.springframework.org/schema/mvc*

*http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd"*

\_ **Spring MVC Annotation :** cú pháp mới annotation => 2 dòng này là bắt buộc !

\_ **View Resolver** :

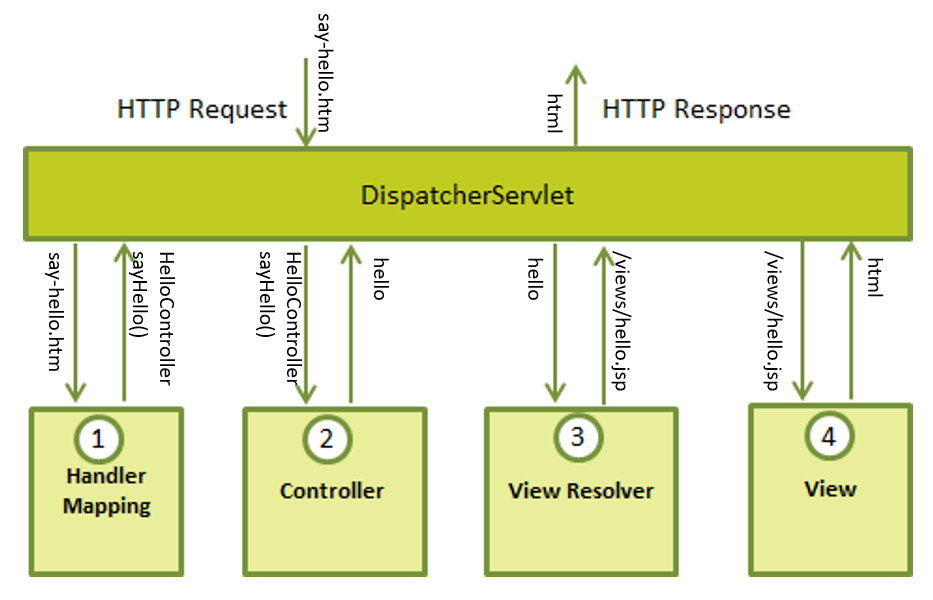
+ prefix : ghép phần đầu

+ suffix : ghép phần cuối => return view không cần .jsp

+ Bắt đầu dấu / là WebContent .

\_ **Chỉ ra Controller :** nằm ở đâu ?

\_ Quy trình :



1. **Tạo project hello world :**

B1 : Tạo dynamic web project

B2 : Cấu hình web.xml ( DispatcherServlet, CharacterEncodingFilter )

( Đã đính kèm txt )

+ servlet-mapping : khi gọi \*.php => DS sẽ chạy. Khi chạy nó sẽ cần các file cấu hình : /WEB-INF/spring-configs/\*.xml . Nơi lưu trữ file cấu hình hệ thống .

B3 : Tạo file spring-configs :

+ spring-config-mvc.xml : khai báo **Spring MVC Annotation** và **ViewResolver** và cấu hình **Controller** .

B4 : Trong WEB-INF : tạo views và tạo package controller như trong file cấu hình mvc.

B5 : Thư viện :

1. **Lấy thư viện :** <http://repo.spring.io/release/org/springframework/spring/>
2. **Tạo controller :** HelloController

\_ Là 1 class được đánh dấu bằng annotation : **@Controller**

\_ Viết phương thức trả về 1 **String**. ( return “hello” ) – tên bất kỳ. ( gọi là **action method )**

\_ Ánh xạ bằng **@RequestMapping(“tên-gọi”)** => tên-gọi.php ( gọi là **action )**

1. **Tạo view** :

\_ Tạo file view : **hello.jsp**

1. **Truyền dữ liệu : Controller => View**

\_ Trước đây : Request, Session, Application.

\_ Trong spring hỗ trợ : **Model**, **ModelMap** – model.**addAttribute**

VD : truyền dữ liệu từ Controller -> view và hiển thị sử dụng EL **${}**

1. **Ánh xạ nhiều action :**

\_ Giả sử : **/products/add** và **/products/edit** và **/products/delete**

\_ Gọm lại : sử dụng annotation ánh xạ cho controller **@RequestMapping(value=”/products ”)** và ánh xạ cho method phần còn lại.

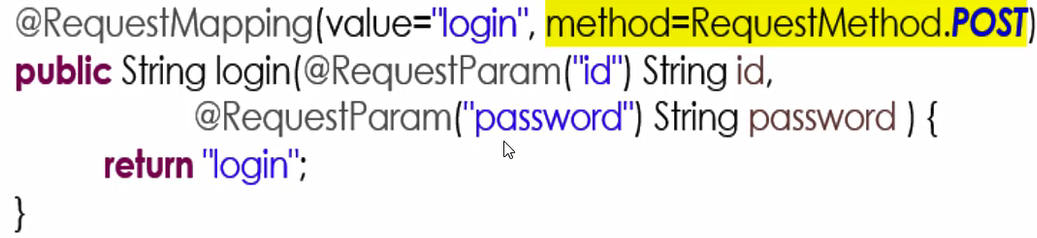
VD : /home/index và /home/about **và** view return về home/index ( home/about )

1. **Ánh Xạ ( Rõ hơn )**
   * **Phân biệt POST – GET**

Cú pháp : @RequestMapping(value=”login”, **method = RequestMethod.GET|POST** )

* **Lấy tham số : ( @RequestParam(“ name\_parameter ”) Data\_Type name ) – CHỌN PHƯƠNG THỨC ĐỂ GỌI BẰNG WEB METHOD.**

Kiểu dữ liệu ( Data\_Type ) tự động ép kiểu. Là tham số đầu vào của action method. Mặc định là get.



Lưu ý : Có thể sử dụng Request. GetParameter. Tuy nhiên phải ép kiểu.  
VD : sử dụng @RequestParam nhận tham số và Model để truyền từ C => V. Tạo form, button có **formmethod** là post hoặc getđể test .

* Action giống nhau : ( sử dụng **params** tại **@RequestMapping** ) **CHỌN PHƯƠNG THỨC ĐỂ GỌI BẰNG THAM SỐ**

**@RequestMapping(value=”list”, params=”get”)**

**@RequestMapping(value=”list”, params=”updateStatus”)**

Vd :

@RequestMapping(params=”get”) và params = post

public String insert và public String insert

1. **Làm việc với tham số :**  4 phương pháp
   * HttpServletRequest
   * @RequestParam
   * JavaBean
   * PathVariable

**\_ HttpServletRequest :** khai báo biến **HttpServletRequest** là tham số của **action method**.

Gọi request.**getParameter** như **servlet**

Nhược điểm : dài dòng, phải tự chuyển kiểu

**\_ @RequestParams.** Nhược điểm : số lượng tham số lớn.

**\_ Bean :** Tạo class có các thuộc tính trùng tên các tham số ! Có **getter, setter** . Sau đó khai báo là tham số của **action method.** Ưu điểm : tự chuyển kiểu

**\_ PathVariable :** nhận dữ liệu từ địa chỉ URL.

[http://zing.vn/news/detail/**2000**.php](http://zing.vn/news/detail/2000.php)

Giả sử muốn lấy giá trị 2000.

Cú pháp :

@RequestMapping(value=”detail/{ **name** }”)

Public String detail(**@PathVariable(“id”)** intvalue){ } // tự ép kiểu

1. **Làm việc đối tượng web ( Web Object ) :**

* Request ( HttpServletRequest )
* Response ( HttpServletResponse )
* Session ( HttpSession )
* Application ( ServletContext )
* Cookie

\_ Request, response, session

C1 : có thể khai báo là tham số của **action method.**

C2 : sử dụng Autowired như application.

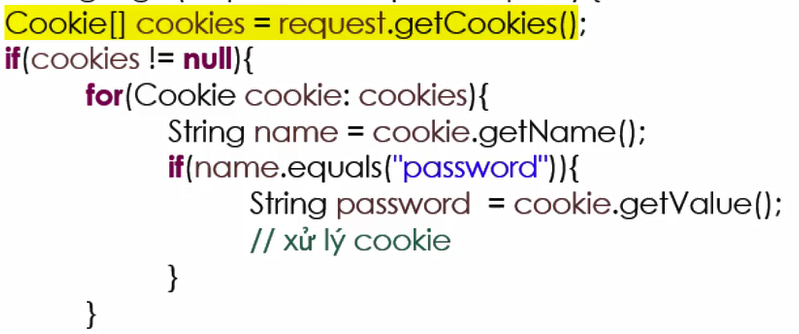
\_ Application sử dụng **@Autowired**

**@Autowired**

**ServletContext application;**

\_ Cookie có 2 cách :

C1 : thủ công – dựa vào name



Phân biệt theo name.

C2 : là tham số của **action method.** Đối tượng @CookieValue( value=”?”, defaultValue=”?”) String box)

Ex :

public String login(**@CookieValue( value=”pwd”, defaultValue=”123456” )** int password)

1. **Action Results :**

**\_ @ResponseBody :** trả về chuỗi. Mặc định trả về tên view. @**ResponseBody** Đánh dấu trên **@RequestMapping.**

**\_ return “redirect:/home/index.html”** // Chuyển trang

**\_ return “forward:/home/index.html”** // Chuyển trang

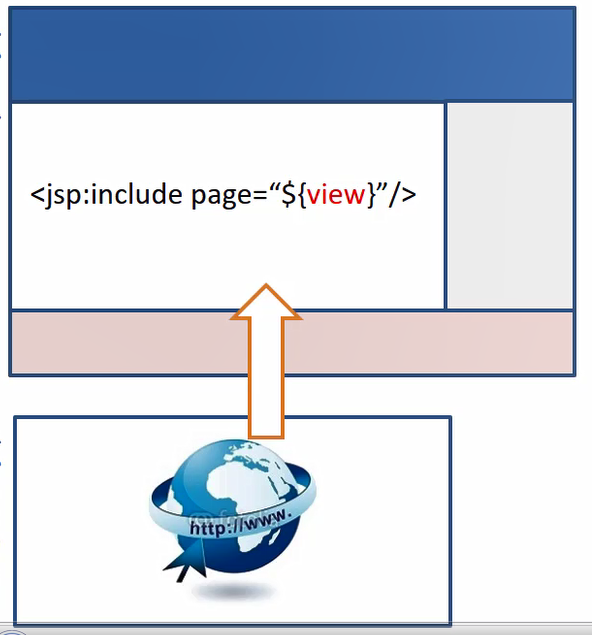
1. **Phân Biệt Forward và Redirect :**

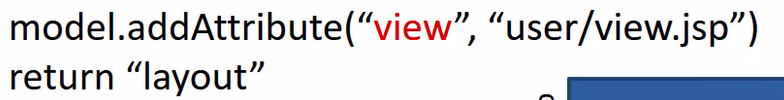
\_ Redirect : tạo ra 1 request khác.

\_ Forward : trên cùng 1 request.

1. **Layout :**

\_ Bao gồm các thành phần dùng chung. Phần thay đổi chỉ việc add vào Model.





B1 : Khai báo tại **controller :**

\_ Tiêm **HttpServletRequest**

\_ Thiết lập thuộc tính ( setAttribute ) với trang tương ứng.

\_ Tất cả các trang sẽ return về layout. ( VD : **return “user/layout”;** )

B2 : Khai báo tại layout :

**<jsp:include page=”${view}” />**

Tại đây có thể tách nhỏ ra : menu, footer, aside.

1. **Thủ Thuật BaseController :** 
   * **Xây dựng :**

\_ Thay vì tiêm vào từ controller **HttpServletRequest** ta tiêm duy nhất vào **BaseController** và cho kế thừa lại. Phạm vi truy cấp tối thiểu : **protected.**

\_ Lần lượt tiêm các đối tượng cần thiết vào : **HttpServletRequest** , **HttpServletResponse, ServletContext, HttpSession**.

\_ Khai báo phương thức thiết lập view tương ứng và layout. ( tái sử dụng )

protected String layout(String **view**){

**request.setAttribute("view", view);**

**return "user/layout";**

}

\_ Tại các controller : các action muốn trả về view tương ứng chỉ việc tạo biến **view** và gọi hàm **layout** :

**String view = “home/about.jsp”;**

**return layout(view);**

\_ Tương tự ta có thể làm layout cho phần **admin**.

* Hàm thiết lập **attribute** :

**protected void addMessage(String message){**

**request.setAttribute("message", message);**

**}**

1. Trang welcome : mặc định index.jsp

