* @Controller: annotation đánh dấu lớp này như 1 Spring bean có thể xử lý các HTTP khác nhau dựa trên việc mapping cụ thể trên class hoặc là các phương thức controller riêng biệt
* @RequestMapping: annotation được sử dụng để liên kết các web requests cụ thể vào các lớp cụ thể và(hoặc) các method cụ thể
* ModelMap: là một implement của Map. Trc kia ta hay dùng request.getAttribute hoặc request.setAttribute. Với ModelMap nó tiện lợi hơn, cung cấp một cách get/set các attributes từ/ tới các request hoặc session.

(Xem giá trị trả ra của các phương thức này) những giá trị này sẽ có tiền tố và hậu tố được định nghĩa trong view resolver (trong spring-servlet.xml)

* File cấu hình Spring chứa cấu trúc để vận hành Spring. Tạo một file spring-servlet.xml bên dưới thư mục WEB-INF. Chú ý chúng ta thể đăt tên file này là gì cũng được nhưng phải đảm bảo là nó được khai báo trong web.xml
* Cấu hình web.xml:
* <servlet>
* <servlet-name>dispatcher</servlet-name>
* <servlet-class>
* org.springframework.web.servlet.DispatcherServlet
* </servlet-class>
* <init-param>
* <param-name>contextConfigLocation</param-name>
* <param-value>/WEB-INF/spring-servlet.xml</param-value>
* </init-param>
* <load-on-startup>1</load-on-startup>
* </servlet>
* <servlet-mapping>
* <servlet-name>dispatcher</servlet-name>
* <url-pattern>/</url-pattern>
* </servlet-mapping>

Mọi thứ đều được tạo ra từ DispatcherServlet và contextConfigLocation. DispatcherServlet là một front controller nhận các request và điều hướng các request tới các controller thích hợp. Nó đồng thời chịu trách nhiệm cho việc

điều hướng từ controller tới view thích hợp.

Hãy xem xét cẩn thận contextConfigLocation init-param. Tham số này có thể đặt tên file cấu hình spring là bất cứ tên gì bạn muốn, đặt nó ở bất cứ đâu bạn muốn, mặc dù bạn có thể cung cấp nhiều file.

Cấu hình một file spring-servlet.xml đơn giản:

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

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

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

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*

*http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

*http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd"*>

<context:component-scan base-package=*"dam.tan.springmvc"*/>

<mvc:annotation-driven/>

<bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"*>

<value>/WEB-INF/views/</value>

</property>

<property name=*"suffix"*>

<value>.jsp</value>

</property>

</bean>

</beans>

Trong đó:

<mvc:annotation-driven/> nói rằng chúng ta có thể định nghĩa các spring beans mà không cần chỉ rõ tập hợp các bean trong xml hoặc implement 1 interface hoặc extend một lớp cơ bản. Ví dụ: với @Controller, Spring biết rằng lớp cụ thể đó chứa các phương thức mà xử lý HTTP requests, không cần định nghĩa như 1 bean trong xml.

<context:component-scan base-package=”dam.tan.springmvc”/> nói với Spring tìm kiếm tất cả các lớp được chỉ rõ trong base-package, và tìm kiếm mỗi class đó xem nó có được annotated cùng với các annotation cụ thể của Sprin[@Controller, @Service, @Repository, @Component,…] và nếu nó có thì sẽ đăng kí lớp đó với bean factory như bạn đã nhập vào trong file cấu hình xml.

Chúng ta click vào project -> Run As -> Maven install. Khi đó trong target chúng ta sẽ thấy sinh ra một loạt các file, trong đó có Spring4MVCHelloWorld.war file.

Deloy file war này trong 1 container. Có nhiều cách để làm điều này. Trong trường hợp Tomcat, đặt file war này bên trong thư mục webapp của tomcat installation của bạn (tomcat/webapp) và click vào start.bat file(bên dưới tomcat/bin).

Để nhanh chóng, tôi sử dụng plugin eclipse jetty để chạy nó trên jetty. Click chuột phải vào file war -> run as -> run-jetty.