* @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.

* Cấu hình với annotation: Lớp cấu hình java được xem như một sự thay thế spring-servlet.xml như việc nó chứa tất cả các thông tin được yêu cầu cho component-scanning và view resolver.

@Configuration: chỉ ra rằng lớp này chứa 1 hoặc nhiều phương thức bean được annotated @Bean đươc đưa ra bởi spring container.

@EnableWebMvc tương tự như mvc:annotation-driven trong XML. Nó có khả năng hỗ trợ annotated @Controller mà sử dụng @RequestMapping để map requests tới phương thức cụ thể.

@ComponentScan tương tự context:component-scan base-package=”” cung cấp nơi mà các bean được spring container quản lý.

Bài 3: Spring 4 MVC Form Validation and Resource Handling(Annotations)

Trong bài này chúng ta sẽ học sử dụng Spring Form Tags, Form Validation sử dụng JSR-303 validation annotations, hibernate-validators, cung cấp cài đặt ngôn ngữ sử dụng MessageSource và truy cập các nguồn tĩnh(css, js, images) trong view của chúng ta sử dụng ResourceHandlerRegistry, tất cả đều sử dụng annotation để cấu hình.

* @Controller chỉ ra rằng lớp này là một controller xử lý requests được map bởi @RequestMapping.
* @RequestMethod.POST sẽ xử lý request submit form Post. Chú ý các tham số và thứ tự của chúng trong phương thức.
* @Valid: sẽ yêu cầu spring validate đối tượng liên quan (student).
* BindingResult chứa kết quả của sự xác nhận (validate) này và bất kỳ lỗi nào có thể xảy ra trong quá trình validate.

Chú ý rằng trong trường hợp validate thất bại, error message mặc định sẽ đc show lên màn hình (ko mong muốn như thế). Thay vì thế, chúng ta cần cấu hình MessageSource trong lớp cấu hình ứng dụng và cung cấp các file thuộc tính chứa message chúng ta sẽ làm.

* Phương thức viewResolver cấu hình một view resolver để xác định view thực. Phương thức addResourceHandlers cấu hình ResourceHandler cho nguồn tĩnh.
* Method messageSource cấu hình một tập hợp Message từ file properties. Chú các tham số được cung cấp tới baseName. Spring sẽ tìm kiếm tệp có tên messages.properties trong class path của ứng dụng.

Chú ý message đc theo một mẫu cụ thể:

{ValidationAnnotationClass}.{modelObject}.{fieldName}

Ví dụ: Size.student.firstName=First Name must be between {2} and {1} characters long

Bài 4: Spring 4 MVC ContentNegotiatingViewResolver example

Tổng quan: Bài học minh họa sự hỗ trợ nhiều định dạng đầu ra của ứng dụng Spring MVC4 sử dụng ContentNegotiatingResolver của Spring. Chúng ta sẽ tạo đầu ra ứng dụng trong dạng XML, JSON, PDF, XLS và HTML. Tất cả sử dụng Annotation để cấu hình.

ContentNegotiatingViewResolver là một kế thừa của ViewResolver, được sử dụng cho media requests(dựa trên phần đuôi của file, định dạng đầu ra của tham số url cụ thể hoặc accept header) để chọn view phù hợp cho request.

* Spring-oxm được them vào để hỗ trợ việc tạo đầu ra XML (sử dụng JAXB). Jackson-databind và Jackson-annotations hỗ trợ cung cấp đầu ra dạng JSON. Itext việc tạo thư viện PDF để hỗ trợ đầu ra PDF. Apache POI sẽ giúp tạo đầu ra XLS.
* XML View Resolver: view này dựa trên JAXB2 Marshalling/unmashalling để sinh ra đầu ra XML. Domain class(lớp chính) cần được annotated với JAXB2 annotations.
* JSON View Resolver: view resolver này sử dụng Spring MappingJackson2JsonView để lấy ra ra view được sử dụng để chuyển POJO thành JSON.
* PDF View Resolver: View này sử dụng lowagie itext library để tạo ra đầu ra PDF.
* XLS View Resolver: View này sử dụng Apache POI library để tạo ra Microsoft XLS output. Chú ý rằng view này kế thừa từ Spring AbstractExcelView