Khanh Nguyen

Tôi yêu thích Java và chỉ chuyên sâu về Java.

Follow

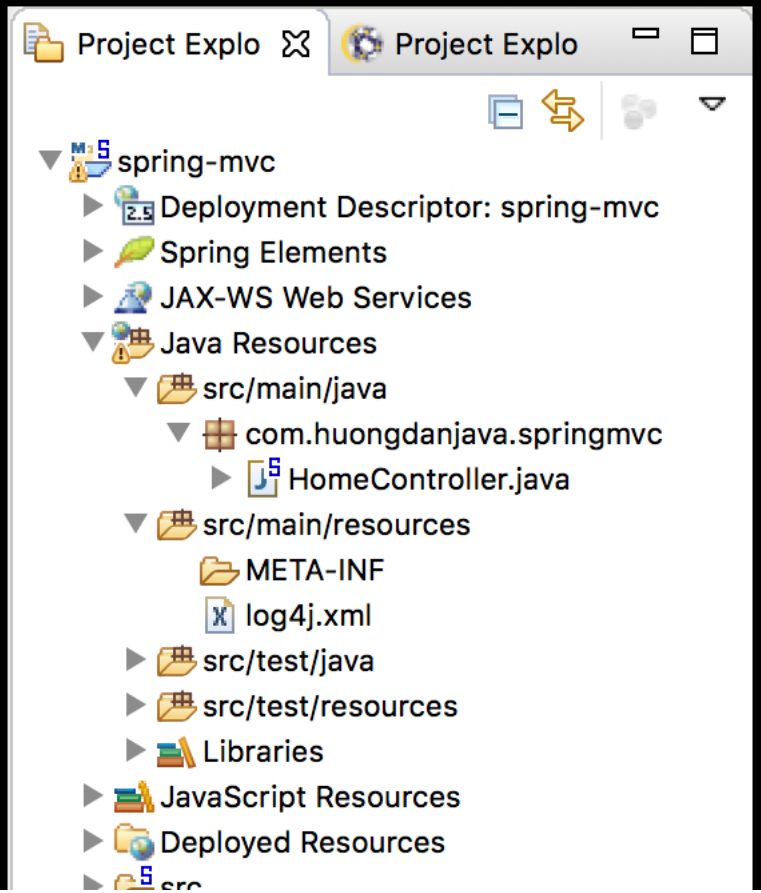
Khởi tạo và cấu hình DispatcherServlet trong Spring MVC

Posted on 4 Tháng Tám, 2016 in Spring MVC | Updated on 19 Tháng Mười Một, 2018

Views: 2.194

Giống như các servlet khác, để **DispatcherServlet** có thể nhận được và xử lý request, chúng ta cần phải cấu hình để web server container có thể khởi tạo và ánh xạ URL cho nó. Trong bài viết này, mình sẽ giới thiệu với các bạn cách khởi tạo và cấu hình **DispatcherServlet** trong Spring MVC!

Để làm ví dụ, mình sẽ sử dụng project mà mình đã tạo ở bài viết trước, cấu trúc project của mình như sau:



Với Servlet 3.0, chúng ta có một số cách để cấu hình và đăng ký một servlet như:

- Sử dụng tập tin **web.xml**.
- Sử dụng tập tin **web-fragment.xml**.
- Sử dụng **javax.servlet.ServletContainerInitializer.**

Spring MVC còn cho phép chúng ta sử dụng the **org.springframework.web.WebApplicationInitializer** để cấu hình và đăng ký một servlet.

Nhưng trong project ví dụ, chúng ta đang sử dụng cách phổ biến đó là sử dụng một tập tin **web.xml** nên trong bài viết này mình chỉ đề cập đến cách này. Nội dung của tập tin web.xml như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSche
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

  <!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/spring/root-context.xml</param-value>
  </context-param>

  <!-- Creates the Spring Container shared by all Servlets and Filters -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <!-- Processes application requests -->
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

Các bước để khởi tạo và cấu hình cho **DispatcherServlet** bao gồm:

- Đăng ký khởi tạo **DispatcherServlet** trong web server container và ánh xạ URL.
- Sau khi khởi tạo xong, **DispatcherServlet** sẽ sử dụng **org.springframework.web.context .WebApplicationContext** để cấu hình cho nó.

Chúng ta sẽ đi vào chi tiết từng bước các bạn nhé!

Khởi tạo và ánh xạ URL cho DispatcherServlet

Giả sử chúng ta chưa có tập tin **web.xml** trong project của chúng ta đi, thì bắt buộc các bạn phải tạo mới một tập tin **web.xml** nằm trong thư mục **/src/webapp/WEB-INF**. Đây là tập tin sẽ chứa tất cả các cấu hình web server container cần để khởi tạo servlet, listener hay filter.

Và để khởi tạo và ánh xạ URL cho **DispatcherServlet** các bạn chỉ cần khai báo như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSche
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

  <!-- Processes application requests -->
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

Mặc định khi **DispatcherServlet** được khởi tạo, nó sẽ khởi tạo một đối tượng **org.springframework.web.context.WebApplicationContext** với hiện thực là **org.springframework.web.context.support.XmlWebApplicationContext**. Đối tượng **XmlWebApplicationContext** này chứa cấu hình tất cả các beans mà chúng ta sẽ định nghĩa trong khung chứa của Spring. Đối tượng này sẽ sử dụng một tập tin cấu hình của Spring với tên gọi mặc định là **[tên-servlet]-servlet.xml** nằm trong thư mục **/src/webapp/WEB-INF**.

Nếu các bạn không định nghĩa tập tin cấu hình của Spring này thì khi chạy ứng dụng web của chúng ta, lỗi sẽ xảy ra:

```
Caused by:
java.io.FileNotFoundException: Could not open ServletContext resource [/WEB-INF/appServlet-servlet.xml]
    at org.springframework.web.context.support.ServletContextResource.getInputStream(ServletContextResourc
    at org.springframework.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanDefinition
    at org.springframework.beans.factory.xml.XmlBeanDefinitionReader.loadBeanDefinitions(XmlBeanDefinition
    at org.springframework.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(Abstract
    at org.springframework.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(Abstract
    at org.springframework.beans.factory.support.AbstractBeanDefinitionReader.loadBeanDefinitions(Abstract
    at org.springframework.web.context.support.XmlWebApplicationContext.loadBeanDefinitions(XmlWebApplicat
    at org.springframework.web.context.support.XmlWebApplicationContext.loadBeanDefinitions(XmlWebApplicat
    at org.springframework.context.support.AbstractRefreshableApplicationContext.refreshBeanFactory(Abstra
    at org.springframework.context.support.AbstractApplicationContext.obtainFreshBeanFactory(AbstractAppli
    at org.springframework.context.support.AbstractApplicationContext.refresh(AbstractApplicationContext.j
    at org.springframework.web.servlet.FrameworkServlet.configureAndRefreshWebApplicationContext(Framework
    at org.springframework.web.servlet.FrameworkServlet.createWebApplicationContext(FrameworkServlet.java:
    at org.springframework.web.servlet.FrameworkServlet.createWebApplicationContext(FrameworkServlet.java:
    at org.springframework.web.servlet.FrameworkServlet.initWebApplicationContext(FrameworkServlet.java:50
    at org.springframework.web.servlet.FrameworkServlet.initServletBean(FrameworkServlet.java:449)
    at org.springframework.web.servlet.HttpServletBean.init(HttpServletBean.java:133)
```

Ở đây, mình sẽ không định nghĩa tập tin này, vì chúng ta có thể sử dụng **DispatcherServlet** để định nghĩa tập tin cấu hình của Spring.

Cấu hình DispatcherServlet

Để thêm tập tin cấu hình của Spring, các bạn có thể sử dụng thuộc tính **contextConfigLocation** của **DispatcherServlet** và cấu hình nó trong tập tin **web.xml** như sau:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSche
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

  <!-- Processes application requests -->
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
```

```
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>appServlet</servlet-name>
    <url-pattern>/</url-pattern>
  </servlet-mapping>

</web-app>
```

Giá trị của thuộc tính **contextConfigLocation** chính là đường dẫn đến tập tin cấu hình cho Spring của chúng ta.

Cấu hình root servlet

Nhìn lại tập tin **web.xml** trong project ví dụ của chúng ta, các bạn sẽ thắc mắc những dòng cấu hình sau dành cho mục đích gì?

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>

<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Như các bạn đã biết, chúng ta có thể định nghĩa nhiều servlet trong một tập tin **web.xml**. Mỗi servlet có thể được khởi tạo, ánh xạ đến các URL khác nhau. Và do đó, chúng ta sẽ có nhiều Spring container tương ứng với từng servlet, với định nghĩa khác nhau.

Trong trường hợp đó, một số định nghĩa bean có thể lặp đi lặp lại trong nhiều Spring container. **org.springframework.web.context.ContextLoaderListener** được tạo ra để giải quyết vấn đề lặp đi lặp lại này bằng cách tạo ra một root **org.springframework.web.context.WebApplicationContext** sử dụng chung cho tất cả các servlet. **ContextLoaderListener** sẽ sử dụng tập tin được định nghĩa trong **contextConfigLocation** của thẻ **<context-param>**.

Trong tập tin **root-context.xml**, chúng ta sẽ định nghĩa các bean, các thuộc tính dùng chung giữa các khung chứa của Spring, mỗi khung chứa Spring trong mỗi servlet sẽ sử dụng các bean, các thuộc tính này cùng với những bean, những thuộc tính được định nghĩa riêng cho chính nó.

Previous Post: Tổng quan quy trình xử lý request trong Spring MVC
Next Post: Tổng quan về log4j 1.x

4 thoughts on “Khởi tạo và cấu hình DispatcherServlet trong Spring MVC”

congptc Reply

6 Tháng Mười, 2018 at 4:40 chiều

Bài viết của bạn rất hay và rất chi tiết , nhờ blog của bạn mà mình đã hiểu thêm rất nhiều về spring và java
Mong rằng bạn sẽ tiếp tục ra thêm những bài viết mới trong thời gian tới !
Cảm ơn bạn rất nhiều về những kiến thức bổ ích bạn đã chia sẻ .
Chúc bạn luôn mạnh khoẻ , và thành công 🙂

Khanh Nguyen Reply

7 Tháng Mười, 2018 at 1:39 chiều

Cảm ơn bạn 🙂

Minh Reply

4 Tháng Sáu, 2018 at 11:43 chiều

Em chào anh , Anh ơi ! anh có thể đưa ra file dispatcherservlet và giải thích rõ từng tag trong đó giúp được không ạ ? em cảm ơn

Khanh Nguyen Reply

5 Tháng Sáu, 2018 at 11:17 sáng

Bạn đang thắc mắc gì nhỉ? Bạn không hiểu chỗ nào thì hỏi nhé!

Add Comment

Name (required)

Email (required)

Website (optional)

Submit Comment

Fanpage

Hướng Dẫn Java
246 likes
hdJava.com

Xem nhiều

Spring Framework cơ bản (11.879)

Nói về Serialization trong Java (10.598)

Tổng quan về Node Package Manager trong

Phản hồi gần đây

Tan trong Sử dụng properties trong Spring với annotation @Value

Khanh Nguyen trong JDBC transaction management trong Spring

Get Updates

Subscribe to our newsletter to receive breaking news by email.

Enter your email

Node.js (9.094)

Hello World với Spark framework (9.004)

Bean autowiring sử dụng @Autowired annotation (8.090)

Hoang Hai trong JDBC transaction management trong Spring

Khanh Nguyen trong Khởi tạo các đối tượng trong Spring container sử dụng tập tin XML

tuyet trong Khởi tạo các đối tượng trong Spring container sử dụng tập tin XML

Liên kết

IT Phú Trần