(http://baeldung.com)

Tutorial Spring MVC

Última modificación: 25 de enero de 2018

por Eugen Paraschiv (http://www.baeldung.com/author/eugen/)
(http://www.baeldung.com/author/eugen/)

Primavera (http://www.baeldung.com/category/spring/) +
Spring MVC (http://www.baeldung.com/category/spring-mvc/)

1. Descripción general y Maven

Este es un sencillo **tutorial de Spring MVC que** muestra cómo configurar un proyecto Spring MVC, tanto con la configuración basada en Java como con la configuración XML.

Los artefactos Maven para el proyecto Spring MVC se describen en detalle en el artículo sobre dependencias Spring MVC (/spring-with-maven#mvc) .

2. El *web.xml*

Esta es una configuración simple de *web.xml* para un proyecto Spring MVC:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance (http://www.w3.org/2001/XMLSchema-in
   xmlns="http://java.sun.com/xml/ns/javaee (http://java.sun.com/xml/ns/javaee)"
   xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_3_1.xsd (http://java.sun.com/xml/ns/javaee
   xsi:schemaLocation="
      http://java.sun.com/xml/ns/javaee (http://java.sun.com/xml/ns/javaee)
      http://java.sun.com/xml/ns/javaee/web-app_3_1.xsd (http://java.sun.com/xml/ns/javaee/web-app
   id="WebApp_ID" version="3.1">

   <display-name>Spring MVC Java Config App</display-name>

   <servlet>
      <servlet-name>mvc</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
      <servlet-name>mvc</servlet-name>
      <url-pattern>/</url-pattern>
   </servlet-mapping>

   <context-param>
      <param-name>contextClass</param-name>
      <param-value>
         org.springframework.web.context.support.AnnotationConfigWebApplicationContext
      </param-value>
   </context-param>
   <context-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>org.baeldung.spring.web.config</param-value>
   </context-param>
   <listener>
      <listener-class>
         org.springframework.web.context.ContextLoaderListener
      </listener-class>
   </listener>

</web-app>
```

Estamos utilizando **la configuración basada en Java** , por lo que estamos utilizando *AnnotationConfigWebApplicationContext* como la clase de contexto principal - esto acepta *@Configuration* clases anotadas como entrada. Como tal, solo necesitamos especificar el paquete donde se encuentran estas clases de configuración, a través de *contextConfigLocation* .

Para mantener este mecanismo flexible, también se pueden configurar varios paquetes, simplemente delimitados por espacios:

```
context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
      org.baeldung.spring.web.config org.baeldung.spring.persistence.config
    </param-value>
</context-param>
```

Esto permite que los proyectos más complejos con múltiples módulos administren sus propias clases de configuración de Spring y los aporten al contexto general de Spring en tiempo de ejecución.

Finalmente, el servlet se asigna a / - lo que significa que se convierte en el **servlet predeterminado** de la aplicación y recogerá cada patrón que no tenga otra coincidencia exacta definida por otro servlet.

Nota : Existen múltiples enfoques para configurar un proyecto Spring MVC. En lugar de web.xml como se describió anteriormente, podemos tener un proyecto 100% configurado en Java usando el inicializador. Para obtener más detalles, consulte nuestro artículo (http://www.baeldung.com/spring-xml-vs-java-

config) existente .

3. La configuración Spring MVC - Java

La configuración Spring MVC Java es simple: utiliza el soporte de configuración MVC introducido en Spring 3.1:

```
@EnableWebMvc
@Configuration
public class ClientWebConfig extends WebMvcConfigurerAdapter {

   @Override
   public void addViewControllers(ViewControllerRegistry registry) {
      super.addViewControllers(registry);

      registry.addViewController("/sample.html");
   }

   @Bean
   public ViewResolver viewResolver() {
      InternalResourceViewResolver bean = new InternalResourceViewResolver();

      bean.setViewClass(JstlView.class);
      bean.setPrefix("/WEB-INF/view/");
      bean.setSuffix(".jsp");

      return bean;
   }
}
```

Muy importante aquí es que podemos registrar controladores de vista que crean una asignación directa entre la URL y el nombre de la vista, **sin necesidad de ningún controlador** entre los dos ahora que estamos usando la configuración de Java.

4. La configuración Spring MVC - XML

Alternativamente a la configuración de Java anterior, también podemos usar una configuración puramente XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans (http://www.springframework.org/schema/b
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance (http://www.w3.org/2001/XMLSchema-instanc
    xmlns:mvc="http://www.springframework.org/schema/mvc (http://www.springframework.org/schema/mv
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans (http://www.springframework.org/schema/beans)
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd (http://www.springframewc
        http://www.springframework.org/schema/mvc (http://www.springframework.org/schema/mvc)
        http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd (http://www.springframework.c

    <bean id="viewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/view/" />
        <property name="suffix" value=".jsp" />
    </bean>

    <mvc:view-controller path="/sample.html" view-name="sample" />

</beans>
```

5. Las vistas JSP

Definimos anteriormente un controlador de vista básico - *sample.html* - el recurso jsp correspondiente es:

```
<html>
   <head></head>

   <body>
      <h1>This is the body of the sample view</h1>
   </body>
</html>
```

Los archivos de vista basados en JSP se encuentran en la carpeta */ WEB-INF* del proyecto, por lo que solo se puede acceder a la infraestructura de Spring y no a través del acceso directo de URL.

6. Spring MVC con arranque

Spring Boot es una adición a Spring Platform que hace que sea muy fácil comenzar y crear aplicaciones autónomas de grado de producción. ***Boot* no pretende reemplazar Spring, sino hacer que trabajar con él sea más rápido y fácil.**

6.1. Arrancadores de arranque de primavera

El nuevo marco proporciona cómodas dependencias de inicio, que son descripciones de dependencia que pueden brindar toda la tecnología necesaria para una determinada funcionalidad.

Éstos tienen la ventaja de que ya no necesitamos especificar una versión para cada dependencia, sino que permitimos que el iniciador administre dependencias para nosotros.

La forma más rápida de comenzar es agregar el *pom.xml* principal de inicio de arranque de primavera (https://search.maven.org/#search%7Cga%7C1%7Ca%3A%22spring-boot-starter-parent%22) *:*

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.6.RELEASE</version>
</parent>
```

Esto se ocupará de la gestión de la dependencia.

6.2. Punto de entrada de arranque de primavera

Cada aplicación creada usando *Spring Boot* necesita simplemente definir el punto de entrada principal. Esta suele ser una clase de Java con el método *principal* , anotado con *@SpringBootApplication* :

```
@SpringBootApplication
public class Application {
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Esta anotación agrega las siguientes otras anotaciones:

- *@Configuration* - que marca la clase como fuente de definiciones de bean
- *@EnableAutoConfiguration* - que le dice al framework que agregue beans en función de las dependencias en el classpath automáticamente
- *@ComponentScan* - que busca otras configuraciones y beans en el mismo paquete que la clase *Application* o debajo

Con Spring Boot, podemos configurar frontend usando Thymeleaf o JSP sin usar ViewResolver como se define en la sección 4. Al agregar *spring-boot-starter-thymeleaf* dependencia a nuestro pom.xml, Thymeleaf se habilita, y no se necesita configuración adicional.

El código fuente de la aplicación Boot está, como siempre, disponible en GitHub (https://github.com/eugenp/tutorials/tree/master/spring-boot-bootstrap) .

Finalmente, si está buscando comenzar con Spring Boot, eche un vistazo a nuestra introducción aquí (http://www.baeldung.com/spring-boot-start) .

7. Conclusión

En este ejemplo configuramos un proyecto Spring MVC simple y funcional, usando la configuración de Java.

La implementación de este sencillo tutorial Spring MVC se puede encontrar en el proyecto GitHub (https://github.com/eugenp/tutorials/tree/master/spring-mvc-xml#readme) : este es un proyecto basado en Eclipse, por lo que debería ser fácil de importar y ejecutar tal cual.

Cuando el proyecto se ejecuta localmente, se puede acceder a *sample.html* en:

http: // localhost: 8080 / spring-mvc-xml / sample.html (http://localhost:8080/spring-mvc-xml/sample.html)

Hola Sujit,
sí, esa es la idea general. Ciertamente puede manejar todas estas asignaciones en la capa de Controlador, pero si no hay una lógica extra, generalmente es más fácil hacer todo aquí y no tener que definir muchos métodos adicionales sin una lógica real.
Espero que aclare las cosas. Saludos,
Eugen.

+ 1 −

hace 1 año

Huésped

Sujit Tripathy

Gracias Eugen.

+ 0 −

hace 1 año

CATEGORÍAS

PRIMAVERA (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)
HTTPCLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL "VOLVER A LO BÁSICO" DE JAVA (HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (HTTP://WWW.BAELDUNG.COM/JACKSON)
TUTORIAL DE HTTPCLIENT 4 (HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE)
REST CON SPRING TUTORIAL (HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/)
TUTORIAL DE SPRING PERSISTENCE (HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/)
SEGURIDAD CON SPRING (HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING)

ACERCA DE

ACERCA DE BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)
LOS CURSOS (HTTP://COURSES.BAELDUNG.COM)
TRABAJO DE CONSULTORÍA (HTTP://WWW.BAELDUNG.COM/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
EL ARCHIVO COMPLETO (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)
ESCRIBIR PARA BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)

CONTACTO (HTTP://WWW.BAELDUNG.COM/CONTACT)
INFORMACIÓN DE LA COMPAÑÍA (HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
TÉRMINOS DE SERVICIO (HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY)
EDITORES (HTTP://WWW.BAELDUNG.COM/EDITORS)
KIT DE MEDIOS (PDF) (HTTPS://S3.AMAZONAWS.COM/BAELDUNG.COM/BAELDUNG+-+MEDIA+KIT.PDF)