



Yo Libros Hablando Menciones

Custom Search

Diseña tu propio arranque Spring Boot - parte 2

últimas publicaciones

14 de febrero de 2016

Nicolas Fränkel

En la última [publicación](#), traté de describir el funcionamiento interno del arranque de Spring Boot. ¡Ahora es el momento de desarrollar el nuestro!

Como ejemplo, usaremos [XStream](#), un serializador XML / JSON (de) de pura materia prima ofrecido por Thoughtworks. Se aconseja a los lectores que solo usan JAXB y Jackson echar un vistazo a XStream, es extremadamente eficiente y su API es bastante fácil de usar.

Como se vio en nuestra última publicación, el punto de entrada de un iniciador se encuentra en el `META-INF/spring.factories` archivo. Vamos a crear un archivo así, con el contenido adecuado:



```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=ch.frankel.blog.xstream.XStreamAutoConfiguration
```

Ahora, creemos la clase a la que se hace referencia arriba. Como hemos visto anteriormente, una clase de autoconfiguración es solo una clase de configuración regular. Está bien mantenerlo vacío por el momento.

```
@Configuration
public class XStreamAutoConfiguration {}
```

XStream se basa en la `XStream` clase con nombre adecuado, que es un punto de entrada en sus características de serialización. Thoughtworks lo diseñó para evitar métodos estáticos, por lo que necesita una `XStream` instancia. Crear una instancia es una tarea aburrida y repetitiva sin valor: parece ser un objetivo perfecto para un frijol de primavera. Creemos esta instancia en la clase de configuración automática como un bean singleton para que las aplicaciones cliente puedan usarlo. Nuestra clase de configuración se convierte en:

```
@Configuration
public class XStreamAutoConfiguration {

    @Bean
    public XStream xstream() {
        return new XStream();
    }
}
```

Hay otras alternativas al constructor de X-Starts no-args documentado en el sitio web de XStream, por *ejemplo*, uno para StaX, otro para JSON, etc. Nuestro iniciador debe permitir a las aplicaciones cliente usar su propia instancia, creando así solo si no hay ninguna proporcionada en el contexto. Eso suena como un condicional al bean perdido:

```
@Configuration
public class XStreamAutoConfiguration {

    @Bean
    @ConditionalOnMissingBean(XStream.class)
    public XStream xstream() {
        return new XStream();
    }
}
```

XStream se basa en convertidores, una forma de convertir de un valor tipeado a una cadena con formato JSON / XML (y viceversa). Hay muchos convertidores preinscritos listos para usar, pero los clientes pueden registrar los suyos propios. En ese caso, debería ser posible proporcionarlos en el contexto para que se registren con la instancia proporcionada.

Para hacerlo, cree un `@Bean` método que tome tanto una instancia de XStream como una colección de conversores como argumentos inyectados. Este método solo debe invocarse si hay al menos una `Converter` instancia en el contexto. Esto se puede configurar fácilmente con la `@ConditionalOnBean` anotación.

```
@Bean
@ConditionalOnBean(Converter.class)
```

- [Haz tu vida más fácil Kotlin stdlib](#)
- [Aprovechar al máximo conferencias](#)
- [En la escasez de desarrolladores](#)
- [¿Es la programación orientada a objetos compatible con un enterprise?](#)
- [Los múltiples usos rebase --onto](#)
- [Navegador alternativo Vaadin](#)
- [Migración de una a Spring Boot a Java Módulos](#)
- [Migración de una a Spring Boot a Java Compatibilidad](#)
- [Construcciones verdaderamente in](#)
- [Pasar por alto las comprobaciones de Javascript](#)

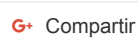
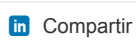
```
public Collection<Converter> converters(XStream xstream, Collection<Converter> converters) {  
    converters.forEach(xstream::registerConverter);  
    return converters;  
}
```

En este punto, cualquier convertidor personalizado proporcionado en el contexto de Spring por las aplicaciones del cliente se registrará en la instancia de XStream.

Con esto concluye esta publicación sobre la creación de iniciadores de Spring Boot. ¡Es bastante fácil y directo! Antes de seguir adelante con la suya, no se olvide de verificar los arrancadores existentes, ya que hay un montón de elementos listos para usar y de la comunidad.

El código fuente completo para esta publicación se puede encontrar en [Github](#).

Java

[arranque de primavera](#)

0 Comments

A Java Geek

Javier Martín Alon... ▾

Recommend

Share

Sort by Newest ▾



Start the discussion...

Be the first to comment.

Subscribe Add Disqus to your site Add Disqus Privacy

[Gestión de registros en Spring Boot](#)[Diseñando su propio motor de arranque Spring Boot - parte 1](#)