(http://baeldung.com)

Datos de primavera con Spring Security

Última modificación: 18 de marzo de 2018

> por baeldung (http://www.baeldung.com/author/baeldung/)
> (http://www.baeldung.com/author/baeldung/)

Persistencia (http://www.baeldung.com/category/persistence/)

Seguridad (http://www.baeldung.com/category/security-2/)

Primavera (http://www.baeldung.com/category/spring/) +

1. Información general

Spring Security proporciona un buen soporte para la integración con Spring Data. Mientras que el primero maneja los aspectos de seguridad de nuestra aplicación, este último proporciona un acceso conveniente a la base de datos que contiene

los datos de la aplicación.

En este artículo, analizaremos cómo **se puede integrar Spring Security con Spring Data para permitir más consultas específicas del usuario** .

2. Spring Spring + Spring Data Configuration

En nuestra introducción a Spring Data JPA (http://www.baeldung.com/the-persistence-layer-with-spring-data-jpa) , vimos cómo configurar Spring Data en un proyecto de Spring. Para habilitar la seguridad del resorte y los datos del resorte, como de costumbre, podemos adoptar la configuración basada en Java o XML.

2.1. Configuración de Java

Recordemos que desde el formulario de inicio de sesión (http://www.baeldung.com/spring-security-login) de Spring Security (http://www.baeldung.com/spring-security-login) (secciones 4 y 5), podemos agregar Spring Security a nuestro proyecto usando la configuración basada en anotaciones:

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {
    // Bean definitions
}
```

Otros detalles de configuración incluirían la definición de filtros, beans y otras reglas de seguridad según sea necesario.

Para habilitar Spring Data en Spring Security, simplemente agregamos este bean a *WebSecurityConfig* :

```
@Bean
public SecurityEvaluationContextExtension securityEvaluationContextExtension
    return new SecurityEvaluationContextExtension();
}
```

La definición anterior permite la activación de la resolución automática de expresiones específicas de datos de primavera anotadas en las clases.

2.2. Configuración XML

La configuración basada en XML comienza con la inclusión del espacio de nombres Spring Security:

```
<beans:beans xmlns="http://www.springframework.org/schema/security (http://w
  xmlns:beans="http://www.springframework.org/schema/beans (http://www.sprin
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance (http://www.w3.org/20
  xsi:schemaLocation="http://www.springframework.org/schema/beans (http://ww
  http://www.springframework.org/schema/beans/spring-beans-4.3.xsd (http://w
  http://www.springframework.org/schema/security (http://www.springframework
  http://www.springframework.org/schema/security/spring-security.xsd (http:/
...
</beans:beans>
```

Al igual que en la configuración basada en Java, para la configuración basada en XML o espacio de nombres, agregamos bean **SecurityEvaluationContextExtension** al archivo de configuración XML:

```
<bean class="org.springframework.security.data.repository
  .query.SecurityEvaluationContextExtension"/>
```

La definición de *SecurityEvaluationContextExtension* hace que todas las expresiones comunes en Spring Security estén disponibles desde las consultas de Spring Data.

Dichas expresiones comunes incluyen *principal, autenticación, isAnonymous (), hasRole ([role]), isAhenthenticated,* etc.

3. Ejemplo de uso

Consideremos algunos casos de uso de Spring Data y Spring Security.

3.1. Restringir la *actualización de* campo de *AppUser*

En este ejemplo, vamos a ver a restringir *la aplicación del usuario* 's *lastlogin* actualización de campo para el usuario sólo autenticado actualmente.

Con esto, queremos decir que siempre que se *active el* método *updateLastLogin* , solo actualiza el *último* campo de *registro* del usuario autenticado actualmente.

Para lograr esto, agregamos la siguiente consulta a nuestra interfaz *UserRepository* :

```
@Query("UPDATE AppUser u SET u.lastLogin=:lastLogin WHERE"
  +" u.username = ?#{ principal?.username }")
public void updateLastLogin (Date lastLogin);
```

Sin la integración de Spring Data y Spring Security, normalmente tendríamos que pasar el nombre de usuario como argumento para *actualizarLastLogin* .

En el caso de que se proporcionen las credenciales de usuario incorrectas, el proceso de inicio de sesión fallará y no es necesario preocuparse por garantizar la validación del acceso.

3.2. Obtener *el* contenido específico del *usuario de la aplicación* con la *paginación*

Otro escenario donde Spring Data y Spring Security funcionan perfectamente mano a mano es un caso en el que tenemos que recuperar el contenido de nuestra base de datos que es propiedad del usuario autenticado actualmente.

Por ejemplo, si tenemos una aplicación de tweeter, es posible que deseemos mostrar tweets creados o apreciados por el usuario actual en su página de fuentes personalizadas.

Por supuesto, esto puede implicar la escritura de consultas para interactuar con una o más tablas en nuestra base de datos. Con Spring Data y Spring Security, esto es tan simple como escribir:

```
public interface TweetRepository extends PagingAndSortingRepository<Tweet, L
    @Query("select twt from Tweet twt  JOIN twt.likes as lk where lk ="+
      " ?#{ principal?.username } or twt.owner = ?#{ principal?.username }")
    Page<Tweet> getMyTweetsAndTheOnesILiked(Pageable pageable);
}
```

Como queremos que nuestros resultados *estén* paginados, nuestro *TweetRepository* extiende *PagingAndSortingRepository* en la definición de la interfaz anterior.

4. Conclusión

La integración de Spring Data y Spring Security ofrece una gran flexibilidad para gestionar estados autenticados en aplicaciones de Spring.

En esta sesión, hemos echado un vistazo a cómo agregar Spring Security a Spring Data. Se puede encontrar más información sobre otras características poderosas de Spring Data o Spring Security en nuestra colección de artículos de Spring Data (http://www.baeldung.com/spring-data) y Spring Security (http://www.baeldung.com/security-spring) .

Como de costumbre, los fragmentos de código se pueden encontrar en GitHub (https://github.com/eugenp/tutorials/tree/master/spring-data-spring-security) .

Deja una respuesta

¡Sé el primero en comentar!

Start the discussion

Suscribir

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)