(/)

Anular propiedades en las pruebas de primavera

Última modificación: 8 de julio de 2019

> por Łukasz Ryś (https://www.baeldung.com/author/lukasz-rys/) (https://www.baeldung.com/author/lukasz-rys/)

Primavera (https://www.baeldung.com/category/spring/) + Pruebas (https://www.baeldung.com/category/testing/)

Si tiene algunos años de experiencia en el ecosistema de Java y le interesa compartir esa experiencia con la comunidad (y, por supuesto, recibir un pago por su trabajo), eche un vistazo a la página "Escriba para nosotros" (/contribution-guidelines) . Aclamaciones. Eugen

1. Información general

En este tutorial, veremos varias formas de anular las propiedades en las pruebas de Spring.

En realidad, Spring proporciona varias soluciones para esto, por lo que tenemos mucho que explorar aquí.

2. Dependencias

Por supuesto, para trabajar con las pruebas de primavera, necesitamos agregar una dependencia de prueba:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <version>2.1.6.RELEASE</version>
    <scope>test</scope>
</dependency>
```

La dependencia de *prueba de arranque de arranque (https://search.maven.org/search?q=a:spring-boot-starter-test%20AND%20g:org.springframework.boot)* contiene todo lo que necesitamos para anular el valor de la propiedad en las pruebas.

3. Configuración

Primero, tendremos que crear una clase en la aplicación que usará nuestras propiedades:

```
@Component
public class PropertySourceResolver {

    @Value("${example.firstProperty}") private String firstProperty;
    @Value("${example.secondProperty}") private String secondProperty

    public String getFirstProperty() {
        return firstProperty;
    }

    public String getSecondProperty() {
        return secondProperty;
    }
}
```

A continuación, asignémosles valores. Podemos hacer eso creando los *application.properties* en el *src / main / resources:*

```
example.firstProperty=defaultFirst
example.secondProperty=defaultSecond
```

4. Anulando un archivo de propiedad

Ahora, anularemos las propiedades poniendo el archivo de propiedades en los recursos de prueba. **Este archivo debe estar en el mismo classpath** que el predeterminado.

Además, debe **contener todas las claves de propiedad** especificadas en el archivo predeterminado. Por lo tanto, agregaremos el archivo *application.properties* en *src / test / java / resources* :

```
example.firstProperty=file
example.secondProperty=file
```

Agreguemos también la prueba que hará uso de nuestra solución:

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class TestResourcePropertySourceResolverIntegrationTest {

    @Autowired private PropertySourceResolver propertySourceResolver;

    @Test
    public void shouldTestResourceFile_overridePropertyValues() {
        String firstProperty = propertySourceResolver.getFirstProperty
        String secondProperty = propertySourceResolver.getSecondPrope

        assertEquals("file", firstProperty);
        assertEquals("file", secondProperty);
    }
}
```

Este método es muy efectivo cuando queremos anular varias propiedades del archivo.

Y, si no pusimos la propiedad *example.second* en el archivo, el contexto de la aplicación no descubriría esta propiedad.

5. Perfiles de primavera

En esta sección, aprenderemos cómo abordar nuestro problema utilizando los perfiles de Spring. **A diferencia del método anterior, éste combina las propiedades del archivo predeterminado y el archivo perfilado** .

Primero, vamos a crear una *aplicación - test.properties* en el archivo *src / test / java / resources:*

```
example.firstProperty=profile
```

Después de eso, crearemos una prueba que usará el perfil de *prueba* :

```
@RunWith(SpringRunner.class)
@SpringBootTest
@ActiveProfiles("test")
public class ProfilePropertySourceResolverIntegrationTest {

    @Autowired private PropertySourceResolver propertySourceResolver;

    @Test
    public void shouldProfiledProperty_overridePropertyValues() {
        String firstProperty = propertySourceResolver.getFirstProperi
        String secondProperty = propertySourceResolver.getSecondPrope

        assertEquals("profile", firstProperty);
        assertEquals("defaultSecond", secondProperty);
    }
}
```

Este enfoque nos permite utilizar valores predeterminados y de prueba. Por lo tanto, este es un método excelente cuando necesitamos **anular varias propiedades de un archivo, pero todavía queremos usar las predeterminadas** .

Además, podemos obtener más información sobre los perfiles de Spring en nuestro tutorial de *perfiles de Spring* (https://www.baeldung.com/spring-profiles) .

6. *@SpringBootTest*

Otra forma de anular el valor de la propiedad es usar la anotación *@SpringBootTest* :

```
@RunWith(SpringRunner.class)
@SpringBootTest(properties = { "example.firstProperty=annotation" })
public class SpringBootPropertySourceResolverIntegrationTest {

    @Autowired private PropertySourceResolver propertySourceResolver;

    @Test
    public void shouldSpringBootTestAnnotation_overridePropertyValues
        String firstProperty = propertySourceResolver.getFirstProperi
        String secondProperty = propertySourceResolver.getSecondPrope

        Assert.assertEquals("annotation", firstProperty);
        Assert.assertEquals("defaultSecond", secondProperty);
    }
}
```

Como podemos ver, la *propiedad example.firstProperty* se ha invalidado mientras que la *propiedad example.secondProperty* no se ha *modificado* . Por lo tanto, esta es una gran solución cuando necesitamos anular solo propiedades específicas para la prueba. Este es el único método que requiere el uso de Spring Boot.

7. *TestPropertySourceUtils*

En esta sección, aprenderemos cómo sobrescribir las propiedades mediante el uso de la clase *TestPropertySourceUtils* en el *ApplicationContextInitializer.*

El *TestPropertySourceUtils* viene con dos métodos que podemos utilizar para definir un valor de propiedad diferente.

Vamos a crear una clase de inicializador que usaremos en nuestra prueba:

```
public class PropertyOverrideContextInitializer
  implements ApplicationContextInitializer<ConfigurableApplicationCor

    static final String PROPERTY_FIRST_VALUE = "contextClass";

    @Override
    public void initialize(ConfigurableApplicationContext configurabl
        TestPropertySourceUtils.addInlinedPropertiesToEnvironment(
          configurableApplicationContext, "example.firstProperty=" +

        TestPropertySourceUtils.addPropertiesFilesToEnvironment(
          configurableApplicationContext, "context-override-applicat
    }
}
```

A continuación, agregaremos el archivo *context-override-application.properties* en *src / test / resources:*

```
example.secondProperty=contextFile
```

Finalmente, deberíamos crear una clase de prueba que usará nuestro inicializador:

```
@RunWith(SpringRunner.class)
@ContextConfiguration(
  initializers = PropertyOverrideContextInitializer.class,
  classes = Application.class)
public class ContextPropertySourceResolverIntegrationTest {

    @Autowired private PropertySourceResolver propertySourceResolver;

    @Test
    public void shouldContext_overridePropertyValues() {
        final String firstProperty = propertySourceResolver.getFirstP
        final String secondProperty = propertySourceResolver.getSecon

        assertEquals(PropertyOverrideContextInitializer.PROPERTY_FIRS
        assertEquals("contextFile", secondProperty);
    }
}
```

La *propiedad example.firstProperty* ha sido anulada desde el método en línea.

La *propiedad example.secondProperty* se ha invalidado del archivo específico en el segundo método. Este enfoque nos permite definir diferentes valores de propiedad al inicializar el contexto.

8. Conclusión

En este tutorial, nos hemos centrado en las múltiples formas que podemos utilizar para anular las propiedades en nuestras pruebas.

También hemos descubierto cuándo usar cada solución o, en algunos casos, cuándo mezclarlas.

Nosotros, por supuesto, también tenemos la anotación *@TestPropertySource* (https://www.baeldung.com/spring-test-property-source) a nuestra disposición.

Como siempre, el código para ejemplos está disponible en GitHub (https://github.com/eugenp/tutorials/tree/master/testing-modules/spring-testing) .

Deja una respuesta

Start the discussion...

Suscribir

LAS CATEGORÍAS

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

KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL DE JAVA "BACK TO BASICS" (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
TUTORIAL HTTPCLIENT 4 (/HTTPCLIENT-GUIDE)
DESCANSO CON TUTORIAL DE PRIMAVERA (/REST-WITH-SPRING-SERIES)
TUTORIAL DE PERSISTENCIA DE PRIMAVERA (/PERSISTENCE-WITH-SPRING-SERIES)
SEGURIDAD CON SPRING (/SECURITY-SPRING)

ACERCA DE

ACERCA DE BAELDUNG (/ABOUT)
LOS CURSOS (HTTPS://COURSES.BAELDUNG.COM)
TRABAJO DE CONSULTORÍA (/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
EL ARCHIVO COMPLETO (/FULL_ARCHIVE)
ESCRIBIR PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORES (/EDITORS)
NUESTROS COMPAÑEROS (/PARTNERS)
PUBLICIDAD EN BAELDUNG (/ADVERTISE)

TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (/PRIVACY-POLICY)
INFORMACIÓN DE LA COMPAÑÍA (/BAELDUNG-COMPANY-INFO)
CONTACTO (/CONTACT)