Protección de recursos JAX-RS utilizando anotaciones

Last Updated: 2021-06-25

Puede proteger los recursos JAX-RS (Java™ API for RESTful Web Services) utilizando anotaciones que especifiquen valores de seguridad.

Antes de empezar

Esta tarea presupone que ha desarrollado la aplicación e identificado los recursos JAX-RS que desea proteger utilizando anotaciones para la seguridad.

Acerca de esta tarea

Puede proteger los recursos JAX-RS utilizando anotaciones para la seguridad soportadas por JSR 250. Puede utilizar las anotaciones siguientes para añadir la semántica de autorización a los recursos de aplicación JAX-RS:

- @PermitAll : especifica que se permiten todos los roles de seguridad para acceder a los recursos JAX-RS
- @DenyAll : especifica que no se permite ningún rol de seguridad para acceder a los recursos JAX-RS
- @RolesAllowed : especifica los roles de seguridad que se permiten para acceder a los recursos JAX-RS

Puede elegir anotar en el nivel de clase o en el nivel de método. Las reglas siguientes rigen las anotaciones para la seguridad:

Las anotaciones de nivel de método tienen prioridad sobre las anotaciones en el nivel de clase.

En el siguiente fragmento de código, el recurso JAX-RS al que hace referencia la anotación @GET y @Path de `/direcciones` y el método `getList ()` correspondiente no está restringido y abierto para el consumo público. Sin embargo, el recurso al que hacen referencia las anotaciones @PUT y @Path del `/direcciones` y el método `updateList ()` correspondiente requiere el rol de `Gestor`; por ejemplo:

```
@Path(value="/addresses")
        @PermitAll
        public class  AddressBookResource {

          @GET
          @Produces(value="text/plain")
          public String getList() {
          }

          @PUT
          @RolesAllowed("Manager")
          to public void updateList(String[] books) {
          }
        }
```

Las anotaciones para la seguridad se excluyen mutuamente.

Esto significa que cada recurso sólo se rige por como máximo una de las anotaciones para la seguridad. Por ejemplo, el ejemplo siguiente no es válido porque se han especificado @PermitAll y @RolesAllowed :

```
         @Path(value="/addresses")
         @PermitAll
         @RolesAllowed("Employee")
         public class  AddressBookResource {

        @GET
        @Produces(value="text/plain")
        public String getList() {
        }
      }
```

En el ejemplo de código anterior, la anotación @RolesAllowed tiene prioridad y la anotación @PermitAll se ignora. De forma similar, si se especifica la anotación @RolesAllowed y la anotación @DenyAll , la anotación @DenyAll tiene prioridad.

De forma similar, si las anotaciones @PermitAll y @DenyAll se especifican en el método o en el nivel de clase, la anotación @DenyAll tiene prioridad, ya que garantiza la seguridad conforme al principio predeterminado seguro.

Si las anotaciones @PermitAll, @DenyAll y @RolesAllowed están presentes en el método o nivel de clase, la anotación @DenyAll tiene prioridad sobre @RolesAllowed y @PermitAll. El orden de prioridad de estas anotaciones es el siguiente:

1. @DenyAll
2. @RolesAllowed
3. @PermitAll

Regla de herencia

Las anotaciones JSR 250 que se añaden a nivel de clase sólo afectan a las clases que anotan y a los métodos correspondientes para subrecursos. Las anotaciones que se especifican a nivel de clase no afectan a los recursos heredados de una superclase.

Regla para método (s) de alteración temporal

Las anotaciones en recursos que corresponden a métodos alterados temporalmente en subclases tienen prioridad sobre las anotaciones que se incluyen en la clase padre. En el siguiente fragmento de código, el rol `Administrador local` se utiliza para acceder al subrecurso `/dirección/local` ; por ejemplo:

```
s@Path(value="/addresses")
@PermitAll
public class  AddressBookResource {

   @GET
   @Produces(value="text/plain")
   public String getList() {
   }

   @PUT
```

```
  @RolesAllowed("Administrator")
  public void updateList(String books) {

  }
}

@Path(value="/addresses")
@PermitAll
public class  LocalAddressBookResource
              extends AddressBookResource {

  @PUT
  @RolesAllowed("LocalAdministrator")
  @Path(value="local")
  public void updateList(String books){

  }
}
```

@RolesAllowed consideración

No puede tener varias anotaciones @RolesAllowed simultáneamente en un recurso. Por ejemplo, puede lograr:

```
@RolesAllowed("role1")
 @RolesAllowed("role2")
  public String foo() {
  }
```

utilizando el siguiente fragmento de código:

```
@RolesAllowed({"role1", "role2"})
 public String foo() {
  }
```

Consideraciones sobre el uso de anotaciones para la seguridad y la configuración de restricciones de seguridad

Las anotaciones para la seguridad siguen el modelo de seguridad declarativo. Las restricciones de seguridad que se configuran en el descriptor de despliegue, el archivo web.xml, tienen prioridad sobre las restricciones de seguridad que se anotan mediante programación en la aplicación. Es importante que los desarrolladores de recursos JAX-RS consideren un equilibrio entre restricciones de seguridad configurables y restricciones de seguridad anotadas. Las restricciones anotadas son adicionales a cualquier restricción de seguridad configurada. El entorno de ejecución de JAX-RS comprueba las restricciones anotadas después de que el entorno de ejecución del contenedor web haya comprobado las restricciones de seguridad que se han configurado en el archivo web.xml.

Configure las restricciones de autenticación en el archivo web.xml. En el siguiente archivo web.xml de ejemplo, se define la restricción de seguridad de `SecurityConstraint_1` . Esta restricción se utiliza para requerir autenticación en la aplicación. Además, la restricción de seguridad de `SecurityConstraint_1` define restricciones en los patrones de URL correspondientes a los recursos JAX-RS. Cuando se accede a un recurso JAX-RS que corresponde a una de estas restricciones, se realizan comprobaciones de autorización. Las comprobaciones de acceso se realizan para las anotaciones de seguridad declarativas sólo después de verificar las restricciones configuradas.

```
<web-app id="WebApp_someID">
<servlet>
  <servlet-name>AddressBookAppSample</servlet-name>
  <servlet-class>
       org.apache.wink.server.internal.servlet.RestServlet
  </servlet-class>
        <init-param>
            <param-name>javax.ws.rs.Application</param-name>
            <param-value>jaxrs.sample.AddressBookApplication
            </param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
  </servlet>
    <servlet-mapping>
        <servlet-name>AddressBookApp</servlet-name>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>
    <security-constraint id="SecurityConstraint_1">
      <web-resource-collection id="WebResourceCollection_1">
        <web-resource-name>AddressBookAppSample</web-resource-name>
        <description>Protection area for Rest Servlet</description>
        <url-pattern>/*</url-pattern>
```

```
            <http-method>GET</http-method>
            <http-method>POST</http-method>
            <http-method>PUT</http-method>
          </web-resource-collection>
          <auth-constraint id="AuthConstraint_1">
             <description>Role1 for this rest servlet</description>
             <role-name>Role</role-name>
          </auth-constraint>
          <user-data-constraint id="UserDataConstraint_1">
                <transport-guarantee>CONFIDENTIAL</transport-guarantee>
          </user-data-constraint>
       </security-constraint>
       <security-role id="SecurityRole_1">
            <description>This Role is used to drive authentication
            </description>
            <role-name>Role1</role-name>
       </security-role>
       <login-config>
           <auth-method>BASIC</auth-method>
           <realm-name>test realm</realm-name>
       </login-config>
    </web-app>
```

En el archivo web.xml de ejemplo anterior, se utiliza `Role1` para toda la aplicación. Si sólo está definiendo anotaciones de seguridad declarativas y no está utilizando restricciones de autorización del archivo web.xml, puede correlacionar este rol para la aplicación JAX-RS con el sujeto especial AllAuthenticated para la autenticación de usuario.

Procedimiento

1. Determine si hay restricciones de seguridad definidas por el archivo web.xml para la aplicación JAX-RS.
2. Configure el archivo web.xml para añadir restricciones de seguridad.
   Las restricciones de seguridad que se configuran en el descriptor de despliegue, el archivo web.xml, tienen prioridad sobre las restricciones de seguridad que se anotan mediante programación en la aplicación.
3. Determine si desea añadir anotaciones para la seguridad, además de las restricciones del archivo web.xml.

Decida si desea añadir una de las anotaciones @PermitAll, @DenyAll y @RolesAllowed para proporcionar seguridad adicional para los recursos JAX-RS. Tenga en cuenta las reglas para añadir anotaciones para la seguridad, como la prioridad y la herencia descritas anteriormente.

Resultados

Ha definido recursos JAX-RS seguros utilizando anotaciones de seguridad declarativas.

Ejemplo

El siguiente fragmento de código muestra cómo puede utilizar las anotaciones de seguridad para proteger los recursos JAX-RS. En este ejemplo, el recurso raíz `/direcciones` se asocia con una anotación @PermitAll y, por lo tanto, el subrecurso que corresponde a los métodos `@GET` y `@Produces(value="text/plain ")` está permitido a todos los usuarios porque este recurso no introduce anotaciones de seguridad propias. Sin embargo, el subrecurso que corresponde al método @PUT está asociado con su propia anotación @RolesAllowed y requiere el rol `Administrador` .

```
@Path(value="/addresses")
 @PermitAll
 public class  AddressBookResource {

   @GET
   @Produces(value="text/plain")
       public String getList() {
       }


       @RolesAllowed("Administrator")
       @PUT
       public void updateList(String books) {

       }
 }
```