Usando una contraseña cifrada con AES en WebSphere Liberty

Para usar una contraseña cifrada con AES desde el `server.xml` de WebSphere Liberty en el cliente Squirrel SQL, sigue estos pasos:

---

1. Descifrar la Contraseña Usando las Herramientas de WebSphere Liberty

La contraseña cifrada (por ejemplo, `password="{aes}ABC123..."`) debe ser descifrada primero. La herramienta `securityUtility` de Liberty puede hacerlo si tienes acceso al almacén de claves utilizado para el cifrado.

Comando para Descifrar:

```
# Navega al directorio bin de Liberty
cd /ruta/a/wlp/bin


# Ejecuta el comando decode de securityUtility
./securityUtility decode \
  --server=<RutaDelServidorLiberty> \
  --key=<AliasDeLaClave> \
  --keyPassword=<ContraseñaDelAlmacénDeClaves> \
  --source="{aes}ABC123..."
```

Reemplaza: - `<RutaDelServidorLiberty>`: Ruta a tu servidor Liberty (por ejemplo, `usr/servers/defaultServer`). - `<AliasDeLaClave>`: Alias de la clave en el almacén de claves (predeterminado: `default`). - `<ContraseñaDelAlmacénDeCla` Contraseña para el almacén de claves (predeterminado: `KeystorePass` si no se ha cambiado). - `{aes}ABC123...`: La contraseña cifrada de `server.xml`.

---

2. Usar la Contraseña Descifrada en Squirrel

1. Abre el cliente Squirrel SQL.
2. Crea un nuevo alias de conexión.
3. En las propiedades de la conexión:
    - **Controlador**: Elige tu controlador JDBC de Oracle.
    - **URL**: Usa la URL JDBC de `server.xml` de Liberty (por ejemplo, `jdbc:oracle:thin:@//host:port/service`).
    - **Nombre de usuario**: El nombre de usuario de `<dataSource>` de Liberty.
    - **Contraseña**: Introduce la **contraseña descifrada** obtenida del Paso 1.

Notas Importantes

- **Acceso al Almacén de Claves**: Necesitas el archivo del almacén de claves (`key.jks` o `key.p12`) y su contraseña. Por defecto, reside en el directorio `resources/security` de Liberty.
- **Seguridad**: Nunca almacenes la contraseña descifrada en texto plano. Introdúcela directamente en Squirrel sin guardarla en un archivo.
- **Ayuda del Administrador**: Si no tienes las credenciales del almacén de claves, contacta con tu administrador de Liberty para descifrar la contraseña de manera segura.

Enfoque Alternativo (No Recomendado)

Si la descifración no es factible, reemplaza temporalmente la contraseña cifrada en `server.xml` con una contraseña de texto plano (**solo para pruebas**). Reinicia Liberty y usa la contraseña de texto plano en Squirrel. Vuelve a la contraseña cifrada después.

```
<!-- Ejemplo: Contraseña de texto plano (evitar en producción) -->
<dataSource ...>
  <properties password="plaintextPassword" />
</dataSource>
```

Siguiendo estos pasos, puedes usar la contraseña descifrada en Squirrel mientras mantienes el cumplimiento con las mejores prácticas de seguridad.

Para conectar el cliente Squirrel SQL a una base de datos Oracle utilizando la configuración especificada en el archivo `server.xml` de un perfil WebSphere Liberty, especialmente cuando las propiedades del controlador JDBC de Oracle incluyen una contraseña cifrada con AES, sigue estos pasos. El proceso implica extraer los detalles de conexión del archivo `server.xml`, manejar la contraseña cifrada y configurar Squirrel SQL adecuadamente. A continuación se presenta una guía completa:

Paso 1: Entender la Configuración de `server.xml`

En WebSphere Liberty Profile, las conexiones a bases de datos se definen en el archivo `server.xml` utilizando un elemento `<dataSource>`. Este elemento especifica las propiedades de la fuente de datos, incluyendo las de una base de datos Oracle mediante un elemento `<properties.oracle>` anidado. Una configuración de ejemplo podría verse así:

```
<dataSource jndiName="jdbc/myOracleDS">
    <jdbcDriver libraryRef="OracleLib"/>
    <properties.oracle url="jdbc:oracle:thin:@//localhost:1521/orcl" user="scott" password="{aes}contraseña_ci
</dataSource>
<library id="OracleLib">
    <fileset dir="${server.config.dir}/lib" includes="ojdbc6.jar"/>
</library>
```

Aquí: - `url`: La URL JDBC para conectarse a la base de datos Oracle (por ejemplo, `jdbc:oracle:thin:@//localhost:1521/`
- `user`: El nombre de usuario de la base de datos (por ejemplo, `scott`). - `password`: La contraseña, cifrada con AES, con el prefijo `{aes}` (por ejemplo, `{aes}contraseña_cifrada`). - `<jdbcDriver>`: Hace referencia al archivo JAR del controlador JDBC de Oracle.

Dado que Squirrel SQL es un cliente independiente y no puede acceder directamente a la fuente de datos gestionada por WebSphere (por ejemplo, mediante una búsqueda JNDI), necesitas configurarlo manualmente utilizando los mismos detalles de conexión.

Paso 2: Extraer Detalles de Conexión de `server.xml`

Localiza el elemento `<dataSource>` en tu archivo `server.xml` que corresponde a tu base de datos Oracle. Del elemento `<properties.oracle>`, toma nota de lo siguiente: - **URL JDBC**: Encontrado en el atributo `url` (por ejemplo, `jdbc:oracle:thin:@//localhost:1521/orcl`). - **Nombre de usuario**: Encontrado en el atributo `user` (por ejemplo, `scott`). - **Contraseña cifrada**: Encontrada en el atributo `password` (por ejemplo, `{aes}contraseña_cifrada`).

La URL JDBC especifica cómo conectarse a la base de datos Oracle, generalmente en uno de estos formatos: - `jdbc:oracle:thin:@//hostname:port/service_name` (usando un nombre de servicio) - `jdbc:oracle:thin:@hostname:port:SID` (usando un SID)

Verifica tu `server.xml` para confirmar la URL exacta.

Paso 3: Decodificar la Contraseña Cifrada con AES

La contraseña en el `server.xml` está cifrada con AES, como indica el prefijo `{aes}`. WebSphere Liberty Profile cifra las contraseñas por seguridad, pero Squirrel SQL requiere la contraseña en texto plano para establecer una conexión. Para decodificar la contraseña cifrada:

1. **Usar la Herramienta `securityUtility` de WebSphere**:
    - Esta herramienta está incluida con tu instalación de WebSphere Liberty, generalmente ubicada en el directorio `bin` (por ejemplo, `<ruta_de_instalación_de_liberty>/bin/`).
    - Ejecuta el siguiente comando en una terminal o símbolo del sistema desde el directorio `bin`:

      ```
      securityUtility decode --encoding=aes <contraseña_cifrada>
      ```

Reemplaza `<contraseña_cifrada>` con la cadena cifrada real del atributo `password` (todo después de `{aes}`). Por ejemplo:

```
securityUtility decode --encoding=aes contraseña_cifrada
```

- La herramienta mostrará la contraseña en texto plano.

2. **Alternativa**:
   - Si no tienes acceso a la instalación de WebSphere Liberty o a la herramienta `securityUtility`, necesitarás obtener la contraseña en texto plano de tu administrador del sistema o de la persona que configuró la fuente de datos.

Guarda la contraseña decodificada de manera segura, ya que la necesitarás para Squirrel SQL.

Paso 4: Configurar el Controlador JDBC de Oracle en Squirrel SQL

Squirrel SQL requiere el controlador JDBC de Oracle para conectarse a la base de datos. Necesitarás el mismo archivo JAR del controlador referenciado en el elemento `<library>` de `server.xml` (por ejemplo, `ojdbc6.jar`).

1. **Obtener el Archivo JAR del Controlador**:
   - Localiza el archivo JAR del controlador JDBC de Oracle especificado en el elemento `<fileset>` del `server.xml` (por ejemplo, `ojdbc6.jar` en `${server.config.dir}/lib`).
   - Si no lo tienes, descárgalo de la versión adecuada del sitio web de Oracle (por ejemplo, `ojdbc6.jar` o `ojdbc8.jar`, coincidiendo con tu versión de la base de datos).
2. **Agregar el Controlador a Squirrel SQL**:
   - Abre Squirrel SQL.
   - Ve a la pestaña **Drivers** en el lado izquierdo.
   - Haz clic en el botón **+** para agregar un nuevo controlador.
   - Configura el controlador:
     - **Nombre**: Introduce un nombre (por ejemplo, “Controlador JDBC de Oracle”).
     - **URL de ejemplo**: Introduce una URL de ejemplo (por ejemplo, `jdbc:oracle:thin:@//localhost:1521/orc`
     - **Nombre de la clase**: Introduce `oracle.jdbc.OracleDriver`.
     - **Ruta de clase adicional**: Haz clic en **Add**, luego navega y selecciona el archivo JAR del controlador JDBC de Oracle.
   - Haz clic en **OK** para guardar el controlador.

Paso 5: Crear una Conexión (Alias) en Squirrel SQL

Ahora, crea un alias de conexión utilizando los detalles extraídos:

1. **Agregar un Nuevo Alias**:
   - Ve a la pestaña **Aliases** en Squirrel SQL.
   - Haz clic en el botón **+** para agregar un nuevo alias.
   - Configura el alias:
     - **Nombre**: Introduce un nombre para la conexión (por ejemplo, "Base de Datos Oracle a través de WebSphere").
     - **Controlador**: Selecciona el controlador JDBC de Oracle que acabas de configurar.
     - **URL**: Introduce la URL JDBC del elemento `<properties.oracle>` de `server.xml` (por ejemplo, `jdbc:oracle:thin:@//localhost:1521/orcl`).
     - **Nombre de usuario**: Introduce el nombre de usuario del `server.xml` (por ejemplo, `scott`).
     - **Contraseña**: Introduce la contraseña en texto plano decodificada del Paso 3.
2. **Propiedades Adicionales (Opcional)**:
   - Si el elemento `<properties.oracle>` en `server.xml` incluye atributos adicionales (por ejemplo, `ssl="true"` o `connectionTimeout="30000"`), haz clic en la pestaña **Properties** en la configuración del alias y agrégalos como pares clave-valor.
3. **Probar la Conexión**:
   - Haz clic en **Test Connection** para verificar que Squirrel SQL puede conectarse a la base de datos.
   - Si es exitoso, haz clic en **OK** para guardar el alias.

La URL JDBC

La URL a usar en Squirrel SQL es el valor especificado en el atributo `url` del elemento `<properties.oracle>` en el archivo `server.xml`. Por ejemplo: - Si `server.xml` tiene `<properties.oracle url="jdbc:oracle:thin:@//localhost:152` `.../>`, entonces usa: `jdbc:oracle:thin:@//localhost:1521/orcl`

Esta URL, combinada con el nombre de usuario y la contraseña decodificada, permite a Squirrel SQL conectarse a la base de datos Oracle.

Notas

- **Contraseña cifrada**: No puedes usar la contraseña cifrada directamente en Squirrel SQL; debe ser decodificada a texto plano.
- **Alcance de DataSource**: El `<dataSource>` en `server.xml` está diseñado para aplicaciones que se ejecutan en WebSphere Liberty. Squirrel SQL, como cliente externo, requiere una configuración manual que refleje estos ajustes.
- **Compatibilidad del JAR**: Asegúrate de que la versión del archivo JAR del controlador JDBC de Oracle coincida con tu versión de la base de datos para la compatibilidad.

Siguiendo estos pasos, podrás configurar Squirrel SQL para conectarse a la base de datos Oracle utilizando las propiedades definidas en el archivo `server.xml` del perfil WebSphere Liberty.

Dado que el comando `securityUtility` en WebSphere Liberty no proporciona una opción de decodificación para contraseñas cifradas con AES (por ejemplo, aquellas con el prefijo `{aes}` en el archivo `server.xml`), necesitas usar un enfoque programático para descifrarlas. El método recomendado implica aprovechar la clase `com.ibm.websphere.crypto.PasswordUtil`, que es parte del entorno de ejecución de Liberty y puede decodificar tales contraseñas. A continuación se presentan dos soluciones prácticas para lograr esto:

Opción 1: Usar una Aplicación Web Temporal (Recomendada)

Desplegar una aplicación web simple en tu servidor Liberty te permite ejecutar la lógica de decodificación dentro del entorno del servidor, asegurando el acceso a la clave de cifrado correcta (predeterminada o personalizada, según lo definido en `server.xml`).

Pasos:

1. **Crear un Archivo JSP** Crea un archivo llamado `decode.jsp` con el siguiente contenido:

```
<%@ page import="com.ibm.websphere.crypto.PasswordUtil" %>
<%
    String encoded = request.getParameter("encoded");
    if (encoded != null) {
        try {
            String decoded = PasswordUtil.decode(encoded);
            out.println("Contraseña decodificada: " + decoded);
        } catch (Exception e) {
            out.println("Error al decodificar la contraseña: " + e.getMessage());
        }
    }
%>
```

2. **Desplegar el JSP**
   - Coloca `decode.jsp` en un directorio de aplicación web, como `wlp/usr/servers/yourServer/apps/myApp.war/WEB`
   - Si es necesario, crea un archivo WAR básico con este JSP y despliégalo usando la consola de administración de Liberty o colocándolo en el directorio `dropins`.
3. **Acceder al JSP**
   - Inicia tu servidor Liberty (`server start yourServer`).
   - Abre un navegador y navega a: `http://localhost:9080/myApp/decode.jsp?encoded={aes}tu_contraseña_cifra` Reemplaza `{aes}tu_contraseña_cifrada` con la contraseña cifrada real de `server.xml`.

4. **Obtener la Contraseña Decodificada** La página mostrará la contraseña descifrada, que luego puedes usar (por ejemplo, en Squirrel SQL para conectarte a una base de datos).
5. **Seguridad de la Aplicación** Después de obtener la contraseña, elimina o restringe el acceso al JSP para evitar su uso no autorizado.

¿Por Qué Funciona Esto? Ejecutar dentro del servidor Liberty asegura que `PasswordUtil.decode()` use la misma clave de cifrado (predeterminada o personalizada, especificada a través de `wlp.password.encryption.key` en `server.xml`) que se utilizó para codificar la contraseña.

Opción 2: Usar un Programa Java Autónomo

Si desplegar una aplicación web no es factible, puedes escribir un programa Java autónomo y ejecutarlo con las bibliotecas de tiempo de ejecución de Liberty en el classpath. Este enfoque es más complicado porque requiere el manejo manual de la clave de cifrado, especialmente si se utilizó una clave personalizada.

Código de Ejemplo:

```
import com.ibm.websphere.crypto.PasswordUtil;

public class PasswordDecoder {
    public static void main(String[] args) {
        if (args.length < 1 || args.length > 2) {
            System.out.println("Uso: java PasswordDecoder <contraseña_cifrada> [crypto_key]");
            return;
        }
        String encoded = args[0];
        String cryptoKey = args.length == 2 ? args[1] : null;
        try {
            String decoded;
            if (cryptoKey != null) {
                decoded = PasswordUtil.decode(encoded, cryptoKey);
            } else {
                decoded = PasswordUtil.decode(encoded);
            }
            System.out.println("Contraseña decodificada: " + decoded);
        } catch (Exception e) {
            System.err.println("Error al decodificar la contraseña: " + e.getMessage());
        }
```

```
    }
}
```

Pasos:

1. **Compilar el Programa**
   - Guarda el código como `PasswordDecoder.java`.
   - Compílalo usando los jars de Liberty:

     ```
     javac -cp /ruta/a/wlp/lib/* PasswordDecoder.java
     ```

     Reemplaza `/ruta/a/wlp` con tu directorio de instalación de Liberty (por ejemplo, `/opt/ibm/wlp`).
2. **Ejecutar el Programa**
   - Si la contraseña se cifró con la clave predeterminada:

     ```
     java -cp /ruta/a/wlp/lib/*:. PasswordDecoder "{aes}tu_contraseña_cifrada"
     ```
   - Si se utilizó una clave personalizada (por ejemplo, definida en `server.xml` como `<variable name="wlp.password.encryption.key" value="tuClave"/>`):

     ```
     java -cp /ruta/a/wlp/lib/*:. PasswordDecoder "{aes}tu_contraseña_cifrada" "tuClave"
     ```
3. **Manejar la Salida** El programa imprimirá la contraseña decodificada o un error si la clave es incorrecta.

Notas:

- Los jars de Liberty (por ejemplo, en `wlp/lib`) contienen `com.ibm.websphere.crypto.PasswordUtil` y sus dependencias.
- Si se utilizó una clave personalizada y no la proporcionas, la decodificación fallará. Verifica `server.xml` o los archivos de configuración incluidos para la clave.

Consideraciones Clave

- **Clave Predeterminada vs. Personalizada**:
  - Si no se especifica `wlp.password.encryption.key` en `server.xml`, se usa la clave predeterminada, y `PasswordUtil.decode(encoded)` debería funcionar sin parámetros adicionales.
  - Si se define una clave personalizada, debes proporcionarla explícitamente al decodificar fuera del servidor (Opción 2) o confiar en el contexto del servidor (Opción 1).
- **Seguridad**: Maneja la contraseña decodificada con cuidado, ya que estará en texto plano. Evita registrarla o exponerla innecesariamente.

- **Preferencia**: La Opción 1 (aplicación web) generalmente es más confiable porque se ejecuta dentro del entorno de Liberty, manejando automáticamente la clave y las dependencias.

Siguiendo cualquiera de los enfoques, puedes descifrar con éxito la contraseña cifrada con AES de `server.xml` a pesar de la falta de una opción de decodificación en `securityUtility`.