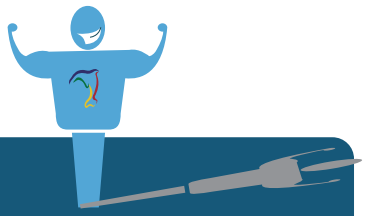
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

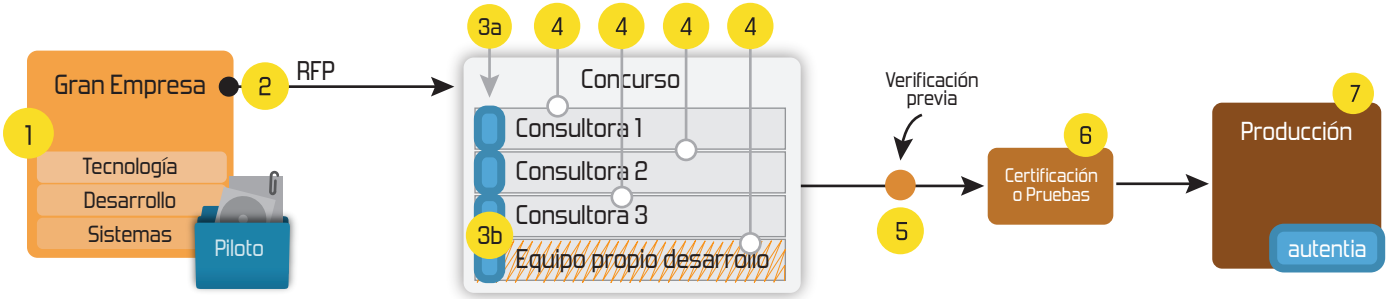
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conocimiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

autentia
real business solutions
Hosting patrocinado por
enredados

DESARROLLADO POR:
Saúl García Díaz

Consultor tecnológico de desarrollo de proyectos informáticos.

Charla sobre LiquiBase

Charla sobre Alfresco Community Edition

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Fecha de publicación del tutorial: 2011-08-04

Share | Regístrate para votar

Usando CallableStatements para ejecutar procedimientos almacenados

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno
- 3. Interfaz CallableStatements
- 4. Ejemplo
- 5. Referencias
- 6. Conclusiones

1. Introducción

JDBC (Java Database Connectivity) es un API que nos proporciona una serie de interfaces para poder acceder a cualquier base de datos que la implemente. El API nos abstrae de los detalles específicos de cada base de datos permitiendonos conectar practicamente de igual manera independientemente de la bases de datos.

De entre todas las interfaces que nos proporciona el API nos centraremos en la interfaz CallableStatements que nos permitirá ejecutar procedimientos almacenados. Esto puede venir bien sobre todo cuando estamos trabajando en proyectos en los que por distintos motivos gran parte del modelo de negocio se encuentra en base de datos y se quiere por ejemplo explotar ese negocio desde una nueva aplicación desarrollada en otra tecnología, Java concretamente.

Para el tutorial de hoy se ha utilizado MySQL con lo que necesitaremos el driver correspondiente Conector/J que actualmente esta en la versión 5.1 . Normalmente es cada base de datos la que pone a nuestra disposición el driver para JDBC que permitirá relizar la conexión de manera sencilla.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware : Portátil Mac Book Pro 15" (2,6 Ghz Intel Core i7, 4 GB DDR3).
- Sistema Operativo:Mac OS X Snow Leopard 10.6.7
- Eclipse Helios
- MySQL 5.1.35
- Java 1.6

3. Interfaz CallableStatements

Lo primero de todo cometaros que podemos obtener más información acerca de la síntaxis de los procedimientos almacenados de MySQL desde el siguiente enlace Procedimientos almacenados y funciones.

A partr de MySQL 5.0 cuando se usa con Conector/J 3.1.1 o posterior la interfaz CallableStatements esta completamente implementada excepto el metodo getParameterMetaData().

Partimos de la situación inicial de un procedimiento almacenado que retorna el valor de un inOutParam incrementado en 1 y la cadena de caracteres pasada via inputParam como Resultset:

```
view plain  print  ?
01. delimiter //
02.
03. CREATE PROCEDURE demoSp(IN inputParam VARCHAR(255), INOUT inOutParam INT)
04. BEGIN
05.     DECLARE z INT;
06.     SET z = inOutParam + 1;
07.     SET inOutParam = z;
08.
09.     SELECT inputParam;
10.
11.     SELECT CONCAT('zyxw', inputParam);
12. END//
```

Catálogo de servicios Autentia

Get ADOBE FLASH PLAYER

Últimas Noticias

- Pirineos on Tour
- VII Autentia Cycling Day
- Autentia patrocina la charla sobre Java SE 7 en Madrid
- Alfresco Day 2011
- XVII Charla Autentia - Grails - Vídeos y Material

Histórico de NOTICIAS

Últimos Tutoriales

- Android: Leer correos de Gmail
- Exportación del contenido de un dataTable de JSF con el soporte de Primefaces
- Spring schedule annotations
- Primeros pasos con Google+
- Spring Security, Single Sign-On bajo plataformas Windows con Active Directory y Kerberos V5

Últimos Tutoriales del Autor

- JSTL: Leyendo atributos de la cabecera de una petición
- Informes dinámicos con DynamicJasper
- Liquibase-Incorporación del histórico de cambios en una BBDD existente
- Liquibase-Gestión De Cambios En Base De Datos
- Accediendo al repositorio de Alfresco via CIFS/SMB , FTP y WebDAV

Síguenos a través de:

Web2PDF
converted by Web2PDFConvert.com

Para usar el procedimiento almacenado demoSP seguimos los siguientes pasos:

- 1. Preparamos el callable statement usando Connection.prepareCall().

  Tened siempre en cuenta que hemos de usar la sintaxis de escape de JDBC siendo obligatorios los paréntesis que rodean los parámetros.

```
view plain  print  ?
01.  import java.sql.CallableStatement;
02.
03.     ...
04.
05.     //
06.     // Prepare a call to the stored procedure 'demoSp'
07.     // with two parameters
08.     //
09.     // Notice the use of JDBC-escape syntax ({call ...})
10.     //
11.
12.     CallableStatement cStmt = conn.prepareCall("{call demoSp(?, ?)}");
13.     cStmt.setString(1, "abcdefg");
14.
15.     ...
```

  Nota: Connection.prepareCall() es un método costoso, debido a la petición de metadatos que hace el driver para suportar los parámetros de salida.
  Por razones de rendimiento, intentad minimizar llamadas innecesarias a Connection.prepareCall() reusando instancias de CallableStatement.

- 2. Registramos los parámetros de salida (si existen)

  Para recibir los valores de los parámetros de salida (parámetros especificados como OUT o INOUT cuando se crea el procedimiento), JDBC requiere que esten especificados antes de la ejecución del statement usando los distinitos modos de registrar un parámetro con el método registerOutputParameter():

```
view plain  print  ?
01.    ...
02.
03.  // Registers the second parameter as output, and
04.  // uses the type 'INTEGER' for values returned from
05.  // getObject()
06.  //
07.
08.  cStmt.registerOutParameter(2, Types.INTEGER);
09.
10.  //
11.  // Registers the named parameter 'inOutParam', and
12.  // uses the type 'INTEGER' for values returned from
13.  // getObject()
14.  //
15.
16.  cStmt.registerOutParameter("inOutParam", Types.INTEGER);
17.  ...
18.
```

- 3. Especificamos los parámetros de entrada (si existen)

  Los parámtros de entrada y de entrada/salida se especifican como en los objetos PreparedStatement . Sin embargo, CallableStatement también soporta especificar los parámetros por nombre:

```
view plain  print  ?
01.  ...
02.
03.    //
04.    // Set a parameter by index
05.    //
06.
07.    cStmt.setString(1, "abcdefg");
08.
09.    //
10.    // Alternatively, set a parameter using
11.    // the parameter name
12.    //
13.
14.    cStmt.setString("inputParameter", "abcdefg");
15.
16.    //
17.    // Set the 'in/out' parameter using an index
18.    //
19.
20.    cStmt.setInt(2, 1);
21.
22.    //
23.    // Alternatively, set the 'in/out' parameter
24.    // by name
25.    //
26.
27.    cStmt.setInt("inOutParam", 1);
28.
29.  .
```

Últimas ofertas de empleo

2011-07-06
Otras Sin catalogar - LUGO.

2011-06-20
Comercial - Ventas - SEVILLA.

2011-05-24
Contabilidad - Expecialista Contable - BARCELONA.

2011-05-14
Comercial - Ventas - TARRAGONA.

2011-04-13
Comercial - Ventas - VALENCIA.

Web2PDF
converted by Web2PDFConvert.com

- 4. Ejecutamos CallableStatement , y recibimos cualquier conjunto de resultados o parámetros de salida.

  Mientras CallableStatement soporta llamar a cualquiera de los métodos de ejecución de Statement (executeUpdate(), executeQuery() o execute()), el método más flexible es execute(), ya que no necesita saber de antemano si el procedimiento almacenado retorna un conjunto de resultados:

```
view plain  print  ?
01.
02.         ...
03.
04.     // Process all returned result sets
05.         cStmt.execute();
06.         final ResultSet rs = cStmt.getResultSet();
07.
08.         while (rs.next()) {
09.             // process result set
10.                 ...
11.         }
12.
13.      //
14.      // Retrieve output parameters
15.      //
16.      // Connector/J supports both index-based and
17.      // name-based retrieval
18.      //
19.
20.      int outputValue = cStmt.getInt(1); // index-based
21.
22.      outputValue = cStmt.getInt("inOutParam"); // name-based
23.
24.  ...
25.
26.
```

4. Ejemplo.

A continuación os muestro un simple ejemplo completo para ver un poquito lo que hemos explicado anteriormente. Será necesario añadir al buildPath el jar correspondiente del driver de MySQL.
Los fuentes aqui.

Por un lado tenemos una clase de apoyo que nos ayuda a establecer la conexión con la base de datos:

```
view plain  print  ?
01.
02.  package com.autentia.tutos;
03.
04.  import java.sql.Connection;
05.  import java.sql.DriverManager;
06.
07.  public class ConnectionMySQL{
08.
09.      public static Connection con_mysql;
10.
11.      public static Connection conectar(String host, String user, String password, String db)
12.              String databaseURL = "jdbc:mysql://" + host + "/" + db;
13.              Class.forName("com.mysql.jdbc.Driver");
14.              con_mysql = DriverManager.getConnection(databaseURL, user, password);
15.          return con_mysql;
16.      }
17.
18.  }
```

y por otro lado la clase que realiza la llamada al procedimiento almacenado:

```
view plain  print  ?
01.
02.  package com.autentia.tutos;
03.
04.  import java.sql.CallableStatement;
05.  import java.sql.Connection;
06.  import java.sql.ResultSet;
07.  import java.sql.SQLException;
08.  import java.sql.Types;
09.
10.  public class SampleStoredProcedures {
11.
12.      public static void main(String[] args) throws SQLException{
13.
14.          Connection conn = null;
15.          try {
16.              // Creamos la conexion
17.              conn=ConnectionMySQL.conectar("localhost", "user", "password", "****");
18.              conn.setAutoCommit(false);
19.
20.               CallableStatement cStmt = conn.prepareCall("{call demoSp(?, ?)}");
21.               cStmt.setString(1, "abcdefg");
22.               cStmt.setInt(2, 0);
23.               cStmt.registerOutParameter("inOutParam", Types.INTEGER);
24.
25.               cStmt.execute();
26.               final ResultSet rs = cStmt.getResultSet();
27.
```

Web2PDF
converted by Web2PDFConvert.com

```
28.                 while (rs.next()) {
29.                     System.out.println("Cadena de caracteres pasada como parametro de entrada="
30.                 }
31.
32.                 int outputValue = cStmt.getInt("inOutParam");
33.                 System.out.println("Parametro de salida incrementado="+outputValue);
34.
35.
36.
37.             }catch (Exception e) {
38.                 conn.rollback();
39.                 e.printStackTrace();
40.             }finally{
41.                 conn.close();
42.             }
43.         }
44.     }
```

Obteniendo como resultado:
Cadena de caractéres pasada como parametro de entrada=abcdefg
Parametro de salida incrementado =1

5. Referencias.

- http://dev.mysql.com/

6. Conclusiones.

Como hemos visto la interfaz CallableStatements nos permite ejecutar procedimientos almacenados e interactuar con el resultado de una manera rápida y sencilla posicionandose como una buena opción a la hora de trabajar con procedimientos almacenados. Espero que les sirva de utilidad, hasta próximos tutoriales.

Un saludo.

Saúl

sgdiaz@autentia.com

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Registrate** y accede a esta y otras ventajas «

COMENTARIOS

SOME RIGHTS RESERVED Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5

Copyright 2003-2011 © All Rights Reserved | Texto legal y condiciones de uso | Banners | Powered by Autentia | Contacto

XHTML 1.0 CSS XML RSS XML ATOM

Web2PDF
converted by Web2PDFConvert.com