


Java y Dispositivos Móviles




Sesión 6:
Fuentes de datos

Java y Dispositivos Móviles

© 2003-2009 Depto. Ciencia Computación e IA

Entrada/Salida-1

Índice




- Introducción a JDBC
- Los drivers
- Conectar a la base de datos
- Operaciones contra la base de datos
- Fuentes de datos con servidores web

Java y Dispositivos Móviles

© 2003-2009 Depto. Ciencia Computación e IA

Entrada/Salida-2

Fuentes de datos



- Introducción a JDBC
- Los drivers
- Conectar a la base de datos
- Operaciones contra la base de datos
- Fuentes de datos con servidores web

Java y Dispositivos Móviles

© 2003-2009 Depto. Ciencia Computación e IA

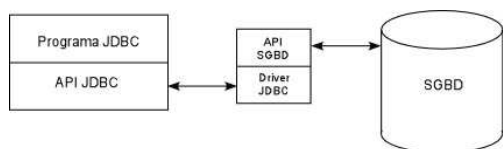
Entrada/Salida-3

Introducción a JDBC



- En muchas aplicaciones deberemos extraer/introducir información de bases de datos
 - Supongamos que nuestra BD es MySQL y que luego la empresa decide cambiar a Oracle
 - TENDRÍAMOS QUE REESCRIBIR EL CÓDIGO???
- Con JDBC no es necesario reescribir código si cambiamos de BD
- Incluso podemos conectar con BDs y servidores diferentes desde el mismo programa

Introducción a JDBC



Fuentes de datos



- Introducción a JDBC
- Los drivers
- Conectar a la base de datos
- Operaciones contra la base de datos
- Fuentes de datos con servidores web

Concepto de driver



- Para que un programa Java pueda conectar con una base de datos, deberá hacer uso del driver específico que dicha base de datos proporciona para poder conectar por JDBC
- Para instalar el driver lo único que debemos hacer en nuestro programa es incluir una línea de código con el nombre del driver:
 - `Class.forName("nombre_clase_driver");`
- La clase del driver deberá estar en el CLASSPATH para poder ejecutar el programa después
- Esta será la ÚNICA línea que deberemos cambiar si luego cambiamos de un sistema de base de datos a otro

Ejemplos de carga de drivers



- Para conectar con una base de datos MySQL:
 - `Class.forName("com.mysql.jdbc.Driver");`
- Para conectar con una BD que tiene una conexión ODBC:
 - `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`
 - Esta clase viene ya incorporada en Java. El resto de drivers deberemos conseguirlos (mediante ficheros JAR) de cada proveedor

Fuentes de datos



- Introducción a JDBC
- Los drivers
- Conectar a la base de datos
- Operaciones contra la base de datos
- Fuentes de datos con servidores web

Conectar a la base de datos



- Tras cargar el driver, obtenemos un objeto *Connection*, indicando la URL de la BD (dependiendo del servidor que se trate), y login y password para acceder, si son necesarios

Conectar a la base de datos



- Por ejemplo, para conectar a la BD “prueba” de MySQL, con usuario “root” y password “root”:

```
> Connection con = DriverManager.getConnection(
    "jdbc:mysql://localhost/prueba", "root", "root");
```

- Para conectar a esa misma BD a través de ODBC (siempre que exista el puente ODBC ya):

```
> Connection con = DriverManager.getConnection(
    "jdbc:odbc:prueba", ...);
```

Fuentes de datos



- Introducción a JDBC
- Los drivers
- Conectar a la base de datos
- Operaciones contra la base de datos
- Fuentes de datos con servidores web

Operaciones contra la base de datos



- Deberemos obtener un objeto *Statement* a partir de la *Connection* creada antes

```
> Statement stmt = con.createStatement();
```

- A partir de ahí, lo demás depende de si queremos hacer una consulta (SELECT) o una actualización (INSERT, UPDATE, DELETE)

Consultas



- Utilizamos el método *executeQuery* del *Statement*, que devuelve un objeto *ResultSet* con los registros encontrados

```
> ResultSet rs =  
    stmt.executeQuery("SELECT * from alumnos");
```

- Para recorrer los registros, utilizamos el método *next* del *ResultSet*, que pasa al siguiente registro mientras haya más:

```
> while (rs.next())  
{  
    ... Explorar registro  
}
```

Consultas (II)



- La clase *ResultSet* tiene métodos *getXXXX* para obtener los campos del registro, según el tipo que sean, pasándole como parámetro el nombre del campo, o la posición que ocupa en el registro:

```
> while (rs.next())  
{  
    int expediente = rs.getInt("exp");  
    String nombre = rs.getString(2);  
}
```

Actualizaciones



- Utilizamos el método *executeUpdate* del *Statement*, que devuelve un entero indicando cuántos registros se han visto afectados por la operación (será 0 si no se pudo hacer):

```
> int res = stmt.executeUpdate("DELETE FROM alumnos  
WHERE exp = 1234");
```

Fuentes de datos



- Introducción a JDBC
- Los drivers
- Conectar a la base de datos
- Operaciones contra la base de datos
- Fuentes de datos con servidores web

BDs en servidores web



- Cuando estamos en una aplicación web, podemos acceder a las bases de datos como hemos visto, o valiéndonos de ficheros de configuración XML (en Tomcat, fichero *META-INF/context.xml*)

```
<?xml version="1.0" encoding="ISO-8859-1"?>  
<Context>  
  <Resource name="jdbc/alumnos" type="javax.sql.DataSource" auth="Container"  
    username="root" password="root" driverClassName="com.mysql.jdbc.Driver"  
    url="jdbc:mysql://localhost:3306/alumnos?autoReconnect=true"/>  
  
  <ResourceParams name="miBD">  
    <parameter <name>maxActive</name> <value>20</value> </parameter>  
    <parameter <name>maxIdle</name> <value>5</value> </parameter>  
    <parameter <name>maxWait</name> <value>10000</value> </parameter>  
  </ResourceParams>  
</Context>
```



- Podemos utilizar este fichero, y obtener la conexión (*Connection*) a la BD con este código:

```
Context initCtx = new InitialContext();
Context envCtx = (Context)initCtx.lookup("java:comp/env");

// Buscamos la BD con el nombre asignado en el XML
DataSource ds = (DataSource)envCtx.lookup("jdbc/alumnos");

// Obtenemos la conexión
Connection con = ds.getConnection();

// A partir de aquí, lo demás es igual
```
