JDBC

Lo standard

Le classi base

Utilizzo nelle applicazioni

Novità della versione 3.0

Matteo Baccan

(baccan@infomedia.it)

http://www.infomedia.it/artic/Baccan

Milano, 24 novembre 2000

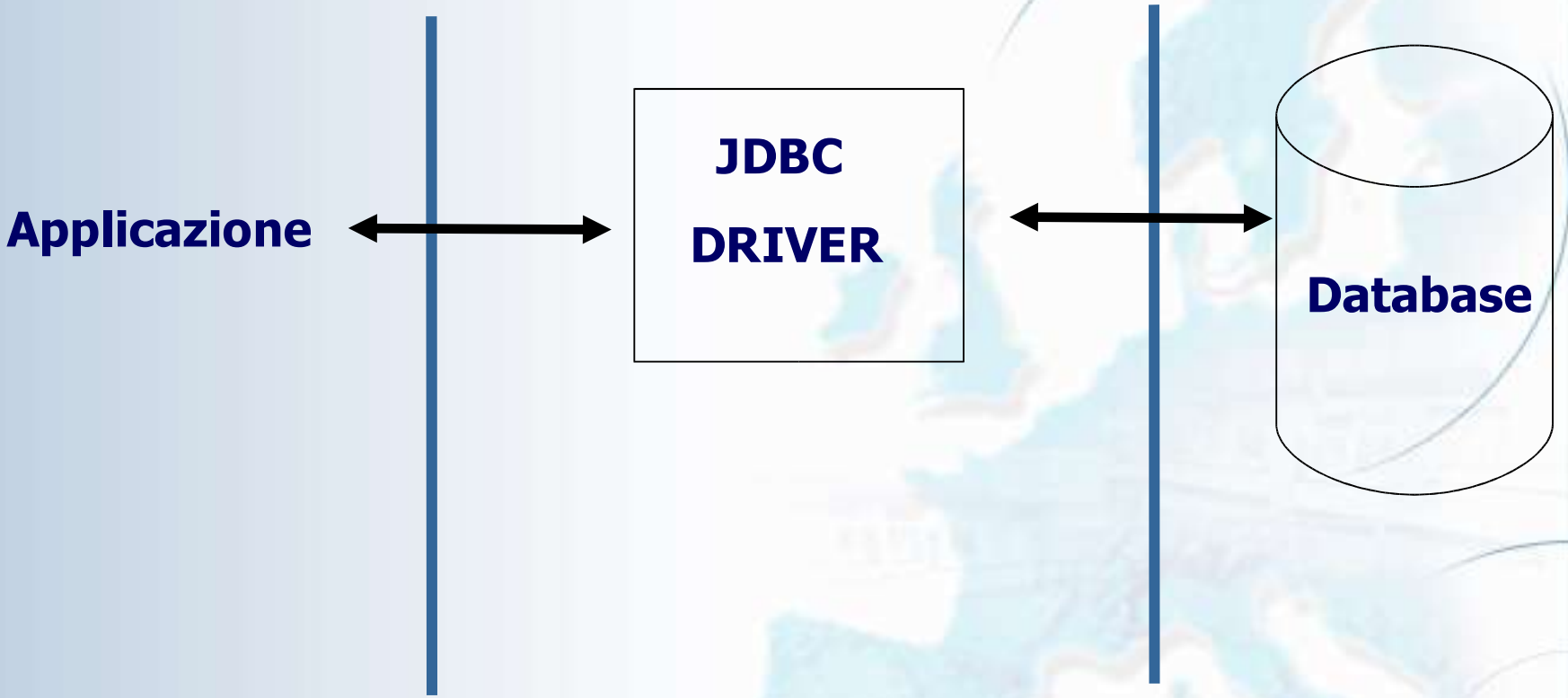
JDBC

- Le API JDBC sono le classi fondamentali per l'accesso ai dati da parte di applicazioni Java.
- Sono parte fondamentale di J2SE (Java 2 Standard Edition) e di J2EE (Java 2 Enterprise Edition.
- Sono divise in due package: java.sql, lo standard e javax.sql le estensioni. Entrambe incluse in J2SE e J2EE

Lo standard

- Nascita 1997
- Uniformità di accesso
- Portabilità
- Apertura del linguaggio verso il mondo dei database, fondamentale in ogni applicazione
- Con la versione 3.0 supporto per lo standard **SQL99** che verrà largamente supportato nei prossimi anni

Architettura



Compatibilità

- JDBC 1.0
  Introduce CallableStatement, Connection, PreparedStatement, ResultSet, Statement
- JDBC 2.0
  Estende DatabaseMetaData, ResultSetMetaData
- JDBC 3.0
  Introduce ParameterMetaData, Savepoint
  Estende DatabaseMetaData

Tipi di driver

- 1. JDBC-ODBC. Il driver utilizza un "ponte" creato da Sun, che permette l'utilizzo di driver di tipo ODBC, all'interno di applicazioni Java.
- 2. JDBC basato su funzioni scritte in maniera nativa, cioè usando JNI: Java Native Interface.
- 3. JDBC-Net, driver nativo, pure Java, che non comunica direttamente con un DBMS, ma con un middleware, in grado di prendere le chiamate e convertirle in qualcosa che il database è in grado di capire, rispondendo poi di conseguenza.
- 4. JDBC, pure Java, con accesso diretto. In questo caso non occorre nulla, se non il driver e il DBMS al quale collegarsi.

Interfaccia

Classi base

- Driver
- Connection
- Statement
- ResultSet

Caricamento

Class.forName(“nome driver”);

com.ibm.db2.jdbc.app.DB2Driver

…..

Gestore: DriverManager

Connessione

Richiesta al DriverManager per una nuova connessione

Connection con =
DriverManager.getConnection(url, user, pwd):

url = indirizzo di connessione
user = utente
pwd = password d'accesso

URL

- Jdbc:<protocollo>:<nome_database>
- protocollo: db2, oracle, inetdae, odbc, etc
- nome_database: ESEMPIO, LOCAL, TEST, DATI

Esempio

```
Class.forName( “COM.ibm.db2.jdbc.app.DB2Driver” );

Connection con;
con = DriverManager.getConnection( “jdbc:db2:local”, “user”, “pwd” );

Statement stmt = con.createStatement();
ResultSet tableSQL =
    stmt.executeQuery("SELECT * FROM Customers");

while( tableSQL.next() ){
    System.out.println( tableSQL.getString( "CompanyName" ) );
}
```

Esempio

Un po’ di pratica

Transazioni

Le transazioni sono usate per poter avere integrità di informazioni sul database SQL. Tutti i driver JDBC devono supportare le transazioni.

Le transazioni si debbono attivare sul driver e devono essere finalizzate con una commit o rimosse con una rollback

Transazioni

Di default i driver JDBC effettuano la COMMIT di ogni modifica al DBMS

```
connection.setAutoCommit(false);

connection. commit();

connection. rollback();
```

Utilizzo dei driver

In base all'utilizzo occorre fare attenzione a quali tipi di driver si usano e quali limiti ci sono nelle varie configurazioni.

Esempi classici

1) Applicazione
2) Applet
3) Scripting server side JSP/Servlet

Applet

- Problemi di sicurezza. L'applet deve essere certificata per poter effettuare certe operazioni
- JDBC:ODBC è improponibile
- Driver di tipo 3 o 4
- ThinDriver. Il peso ha la sua importanza

Esempio applet

Esempio di uso JDBC:ODBC

Esempio di applicazione di accesso ai dati frmQuery

JSP

JSP acronimo di Java Server Pages

Sono la proposta SUN per le pagine dinamiche server side.

Permettono di mischiare codice HTML con codice Java

JSP

Esempio di uso di JSP e JDBC

Ottimizzazioni

- Connection pool.
- CachedRowSet. Possibilità di fare una cache contenente un resultset e riutilizzarla durante il programma anche non connessi al database e anche salvando il dato su disco in modo serializzato
- Stored Procedure e PreparedStatement
- Gestione di Commit e Rollback

Ottimizzazioni

Esempio di CachedRowSet

Esempio PreparedStatement

DEBUG

Il debug non è semplicissimo, dato che il driver cela tutte le chiamate SQL al suo interno.

Come aiuto, la classe DriverManager offre la possibilità di effettuare un LOG delle operazioni che vengono eseguite

```
DriverManager.setLogStream(System.out);
```

JDBC 3.0

- Supporto per i savepoint. Possibilità di definire rollback parziali per le transazioni
- Nuove proprietà per ConnectionPoolDataSource
- Transazioni distribuite: XADataSource, XAConnection. two-phase commit

JDBC 3

Esempio

Materiale didattico

JDK per Win32, Linux e SUN Solaris

http://java.sun.com

http://java.sun.com/products/jdbc/download.html

JDataStore, Jbuilder

http://www.borland.com

MS Java SDK

http://www.microsoft.com

Questa presentazione

http://www.infomedia.it/artic/Baccan

DEV, Login, CP e Mokabyte

http://www.infomedia.it

Domande

Termine presentazione