Java Database Connectivity (JDBC)

Introduzione…

Uno dei motivi di successo di Java è dovuto alla possibilità di sviluppare applicazioni client/server indipendenti dalla piattaforma. L'indipendenza dalla piattaforma deve essere garantita anche per applicazioni che lavorano su basi di dati. Per questo è nato lo standard Java DataBase Connectivity (JDBC).

Uno dei problemi principali con le basi di dati è la non compatibilità dei linguaggi, infatti pur basandosi sullo Structured Query Language (SQL-92), ogni database management system aggiunge o modifica qualcosa allo standard.

…Introduzione…

JDBC è progettato per essere platform-independent. Per permettere ciò JDBC fornisce un *driver manager* che gestisce dinamicamente tutti gli oggetti driver di cui hanno bisogno le interrogazioni a database.

Pertanto se si hanno tre diversi DBMS allora necessiteranno tre diversi tipi di oggetti driver.

Gli oggetti driver si registrano presso il driver manager al momento del caricamento, che può essere forzato con il metodo ***Class.forName( ).***

Come tutte le API Java anche JDBC è stato progettato in modo da semplificare tutte le normali operazioni di interfacciamento con un database: connessione, creazione di tabelle, interrogazione e visualizzazione dei risultati.

Connessione ad un database…

Per aprire un database si deve creare un "*database URL*" che specifica:

- che si sta usando JDBC attraverso la stringa "*jdbc*";
- il "sottoprotocollo" ovvero il nome del driver o il nome di meccanismo di connettività al database.

…Connessione ad un database…

- L'identificatore del database. Questo varia a seconda del database usato, ma generalmente fornisce un nome logico che viene associato, dal software di amministrazione del database, ad una cartella fisica dove sono localizzate le tabelle. In genere è necessario registrare l'identificatore del database e tale processo varia con il tipo di piattaforma.

Queste informazioni sono combinate in una unica stringa chiamata *database URL*.

Connessione ad un database in passi …

Passo 1 – Trovare il driver JDBC

L'istruzione:

```
Class.forName("…");}
```

permette di caricare i driver per il database a cui si vuole accedere.

Per verificare che il caricamento del driver sta funzionando correttamente è opportuno inserire il codice in un blocco *try-catch*.

…Connessione ad un database in passi…

Passo 2 : Configurazione della connessione al database

Quando ci si deve connettere ad un database è necessario invocare il metodo **statico** *DriverManager.getConnection()*

passando come parametri la URL del database, lo ***user name*** e la ***password*** per accedervi.

Sarà restituito un oggetto della classe **Connection** che verrà poi usato per formulare *query* e/o manipolare la base di dati.

…Connessione ad un database in passi.

Passo 3. Generare la query SQL

Stabilita la connessione con l'istruzione *DriverManager.getConnection()*, è possibile usare l'oggetto *Connection* per creare un oggetto *Statement* attraverso il metodo *createStatement()* che permetterà di operare sul database.

A questo punto si chiama punto si invoca il metodo *executeUpdate* (o il metodo *executeQuery* per le query si selezione) per eseguire l'istruzione SQL.

Nel caso di *executeQuery*, verrà restituito un oggetto *ResultSet* che è un iteratore. Infatti attraverso il metodo *next()* è possibile scorrere l'insieme dei record restituiti dalla query.

Esempio

DBAccess.txt