Acessando Banco de Dados em Java (PARTE 3)

comentários: 13

A interface Connection
Como o exemplo anterior (Acessando Bancos de Dados em Java - PARTE 2) demonstrou, as interfaces fundamentais para obter acesso a um repositório de dados em Java são:

1. Connection
2. Statement e suas variantes: PreparedStatement e CallableStatement
3. ResultSet

A Interface Connection é responsável por manter a conexão estabelecida com o repositório de dados através da chamada a DriverManager.getConnection(). A interface Connection tem 3 responsabilidades essenciais: criar ou preparar statements (sentenças SQL), executar o controle de transações com o repositório e criar objetos DatabaseMetaData, que permitem pesquisar dinamicamente que capacidades estão presentes no repositório de dados.

Recapitulando, vimos que a API JDBC usa o padrão de chamada FactoryMethod, para possibilitar que a criação dos diversos objetos de controle de uma transação com um repositório de dados seja adiada para o momento da execução, após o JDBC Driver que possibilita o acesso ser registrado em DriverManager. Desta forma, cada objeto na cadeia de execução de JDBC cria o seu sucessor na cadeia. Ou seja:

DriverManager -> Connection -> Statement (ou variantes) -> ResultSet

Connection pode criar Statement, PreparedStatement e CallableStatement através de um conjunto de chamadas:

createStatement() -> cria uma instância de Statement para execução de SQL no repositório

createStatement(tipo de cursor, concorrência) -> idem ao anterior, especificando o tipo de cursor a criar e o nivel de concorrência da conexão.

Antes da versão 2.0, só se podia conectar um banco de dados JDBC com cursores unidirecionais. Isto é, uma aplicação que lia a linha 1 e em seguida a linha 2 da tabela não podia retornar à linha 1 sem estabelecer um novo result set. A partir da versão 2, se o repositório de dados suportar esta característica, podemos estabelecer cursores bidirecionais de dois tipos: sensiveis, em que as alterações havidas previamente na linha são refletidas na releitura, ou insensíveis, em que as alterações havidas na linha não se refletem na releitura. O tipo de cursor que o repositório suporta é uma informação retornada por DatabaseMetadata.

O controle de transações compreende basicamente 3 grupos de funções:

getAutoCommit() e **setAutoCommit(boolean toSet)** -> permitem examinar ou definir que cada solicitação ao banco tenha commit imediato.

getTransactionIsolation() e **setTransactionIsolation(int transIso)** -> permitem examinar ou definir com que tipo de isolamento de transações aquela conexão deve operar.

commit() e **rollback()** -> se o banco não está em modo autocommit, commit confirma a transação no banco enquanto que rollback cancela as alterações da transação.

Um grupo de funções pouco explorado da interface Connection é o grupo que trata de ResultSets read-only, composto das funções isReadOnly() e setReadOnly(boolean setRead). Estas funções permitem estabelecer trancas de apenas leitura no banco de dados, evitando que o banco seja saturado por trancas de escrita (trancas de bloqueio), o que permite uma maior concorrência no acesso ao banco.

Publicado por daltoncamargo

Últimos artigos

Trabalhando com arquivos Microsoft Word em Java

Torre de Brahma (Torre de Hanoi) - Simulador em Swing

Exemplo simples de Singleton

NetBeans 6.5, jtree - treeModel e defaultMutableTreeNode

Master Detail baseado em JComboBox

Computação Soberana

Exemplo de aplicação JSF usando o NetBeans 6.5 em 20 passos

Exemplo de CardLayout no netbeans6.5

Pegando o endereço MAC do computador.

Instalando e configurando o Java no Linux Ubuntu

Aplicando Transparencia no panel

Apresentar Resultado de Consulta SQL em JTable

Gerador de Código para JTable

Componente Visual Para Selecionar Datas

Vagas em TI

Blogs em TI

Leia também:
Acessando Banco de Dados em Java (PARTE 1)
Acessando Banco de Dados em Java (PARTE 2)

Dalton Milkvicz de Camargo
dalton@javafree.com.br