Universidade Cruzeiro do Sul

Sistemas Cliente/Servidor

Conteúdo:

- Processamento de banco de dados com JSP utilizando PreparedStatement
- Exemplo
- Exercício

Statement e PreparedStatement

Ao trabalhar com instruções sql utilizando a interface Statement, percebemos a complexidade em montar instruções sql, veja o exemplo abaixo:

```
...
stm = con.createStatement();
sql = "INSERT INTO alunos (rgm,nome,curso,email) VALUES
   ('"+request.getParameter("rgm")+"','"+request.getParameter("nome")+"','"
   +request.getParameter("curso")+"','"+request.getParameter("email")+"')";
int retorno = stm.executeUpdate(sql);
...
```

Neste pequeno exemplo, é visível a dificuldade em montar a instrução sql devido a quantidade de aspas simples (‘), aspas duplas (“) e sinais de +.

PreparedStatement

- JDBC fornece uma maneira de executarmos comandos SQL parametrizados
  - Interface PreparedStatement, subclasse de Statement.
- Vantagens de PreparedStatement
  - código mais legível
  - melhor performance pois os comandos são pré compilados e otimizados.
  - ganhos de seguranca minimizando ataques de injecão de SQL.
- Com PreparedStatement, a string contendo o comando SQL deve ser fornecida no momento de criacão do comando e não no momento de execucão do comando:

```
...
Connection con = null;
PreparedStatement st = con.prepareStatement("Select * FROM pessoa where nome = ?");
st.setString(1, "Rodrigo");
resultado = st.executeQuery( );
...
```

PreparedStatement

- Os parâmetros de um PreparedStatement são indicados por sinais de interrogacão dentro da string que contém o comando SQL.
  - A interface PreparedStatement fornece uma série de métodos do tipo setInt, setDate, etc. que permitem passar os valores dos parâmetros antes da execução
- Os parâmetros são identificados pela sua posição dentro da string SQL, iniciando por 1
- Devido às vantagens de se ter o código mais simples e mais seguro, a maioria dos desenvolvedores prefere usar PreparedStatement o tempo todo, mesmo para comandos SQL que serão executados uma única vez e onde portanto não haveria ganho de performance

PreparedStatement (Vantagens)

- Interface que representa uma operação SQL pré-compilada;
- Para criar um objeto do tipo **PreparedStatement usa-se o método** prepareStatement(String sql) do objeto Connection;
- Diferente da criação do objeto Statement, é necessário passar o comando SQL como parâmetro no método;
- O uso do objeto PreparedStatement pode acelerar a execução dos comandos SQL, pois estes estarão pré-compilados;
- O objeto PreparedStatement permite o uso de parâmetros nos comandos SQL, permitindo o uso do mesmo comando SQL, mas alterando alguns valores dinamicamente

Classificação Funcional da API JDBC

Categoria	*Classe ou Interface*	*Fornecido por*	*Comentario*
Conexão com uma fonte de dados	Classe DriverManager	java.sql (Sun)	Gerencia um conjunto de drivers JDBC
	Interface Driver	Fabricante do SGBD	Fornece comunicacão com o banco de dados
	Interface Connection	Fabricante do SGBD	Gerencia uma sessão com o bancode dados
Envio de instrucões SQL	Interface Statement	java.sql	Instrucão SQL estática Objeto empacotador
	Interface PreparedStatement	java.sql	Instrucão SQL pré-compilada Objeto empacotador
	Interface CallableStatement	java.sql	Procedimento armazenado Objeto empacotador
Recuperacão de resultados de uma consulta	Interface ResultSet	java.sql	ResultSet do Banco de Dados Objeto empacotador

PreparedStatement

Veja como ficaria o exemplo ao invés de utilizarmos Statement, utilizando essa interface:

...

```
sql = "INSERT INTO alunos (rgm,nome,curso,email) VALUES (?,?,?,?)";
pstm = con.prepareStatement(sql);
pstm.setString(1,request.getParameter(“rgm”);
pstm.setString(2,request.getParameter(“nome”);
pstm.setString(3,request.getParameter(“curso”);
pstm.setString(4,request.getParameter(“email”);
int retorno = stm.executeUpdate();
```

...

PreparedStatement

- Com a interface PreparedStatement, podemos montar nossa instrução SQL sem complicações, nos locais onde serão inseridos os valores passados para a instrução, utilizamos o ponto de interrogação.

```
sql = "INSERT INTO alunos
   (rgm,nome,curso,email) VALUES (?,?,?,?)";
```

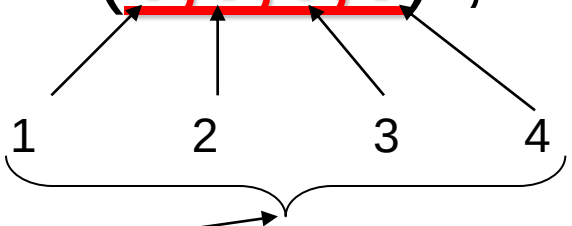
PreparedStatement

- Após a definição da instrução SQL, devemos passar os parâmetros que irão substituir cada ponto de interrogação, para a passagem desses parâmetros, utilizamos métodos associados aos tipos predefinidos no banco.
  Ex: pstm.setString(...)
  pstm.setInt(...)
  pstm.setFloat(...)

- Os parâmetros solicitados por todos os métodos set da interface PreparedStatement são dois, o primeiro indica o indice da interrogação e o segundo o valor que será atribuído, o indice neste caso começa com o valor 1
  Ex: pstm.setString(1,”33333-3”);

PreparedStatement

- sql = "INSERT INTO alunos (rgm,nome,curso,email) VALUES (?,?,?,?)";




```
pstm.setString(1,request.getParameter(“rgm”);
pstm.setString(2,request.getParameter(“nome”);
pstm.setString(3,request.getParameter(“curso”);
pstm.setString(4,request.getParameter(“email”);
```

PreparedStatement

```
sql = "UPDATE alunos SET rgm = ?,nome = ?,curso = ?,email = ? WHERE id = ?";
```

```
pstm = con.prepareStatement(sql);

pstm.setString(1,request.getParameter("rgm"));
pstm.setString(2,request.getParameter("nome"));
pstm.setString(3,request.getParameter("curso"));
pstm.setString(4,request.getParameter("email"));
pstm.setInt(5,Integer.parseInt(request.getParameter("id")));
```

Referência

- Consulte mais informações em

http://download.oracle.com/javase/tutorial/jdbc/basics/prepared.html

http://download.oracle.com/javase/1.4.2/docs/api/java/sql/PreparedStatement.html

Exemplo

- Vamos testar e verificar o código do exemplo da aula sobre cadastro de alunos utilizando agora a interface PreparedStatement.

Exercício

- Faça um cadastro de produtos utilizando a interface PreparedStatement, neste exercício implemente as seguintes funcionalidades:
  - Cadastrar um novo produto
  - Listar todos os produtos cadastrados ordenados por nome
  - Pesquisar um produto através do nome
  - Apagar um produto cadastrado
  - Alterar um produto cadastrado