Tópicos Especiais em Análise e Desenvolvimento de Sistemas

Prof. Rafael Odon (rafael.alencar@prof.una.br)

ORIENTAÇÕES PARA ENTREGA:

- O roteiro poderá ser feito **em dupla** ou **individual**. A dupla definida não poderá ser alterada para os próximos roteiros.
- Enviar até o início da aula de 26/02/2014 o código e as respostas de **todas as perguntas** desse roteiro para o e-mail do professor (rafael.alencar@prof.una.br) com o título:
  - `Roteiro JPA 01 – Nome do 1º Aluno / Nome do 2º Aluno`
- Lembre-se: **todas as informações** para responder as perguntas são dadas em **sala de aula**. Preste atenção na aula e faça notas pessoais!
- As **respostas** devem ser pessoais e **escritas com suas palavras**. Não serão aceitas respostas longas e/ou **copiadas** de forma literal do material de referência ou Internet.

Java Persistence API & Hibernate

Roteiro de Aula 01 – 19/02/2014

Esse Roteiro baseia-se no uso dos softwares:

- Netbeans 7.4.1 Java EE
- Java DB (Apache Derby)
- Java Development Kit (JDK) 1.7

Fase 1: Configurações de Banco de Dados

Antes de iniciar o Roteiro, é preciso certificar que temos um servidor de banco de dados acessível e rodando, bem como um banco de dados disponível para utilização. Vamos utilizar a tecnologia Java DB disponível no próprio Netbeans 7 Java EE que fornece um servidor de banco de dados feito também em Java, o Apache DB.

1. No Netbeans, acesse a aba **Serviços** no canto esquerdo. Caso ela não esteja visível, vá em **Janelas > Serviço (CTRL +5)**.

2. Expanda o item **Banco de Dados** da lista, clique com o botão direito no banco, **Java DB** > **Inicializar servidor**.

3. Caso seja a primeira vez que você inicia o JavaDB, uma janela será aberta requisitando informações sobre o diretório onde está o JavaDB e o diretório onde você deseja gravar seu banco de dados.

4. Aponte o item **Instalação do Java DB** para o diretório onde está instalado o seu JavaDB (geralmente fica na pasta do JDK ou na pasta do Glassfish):
   - **Instalação do Java DB:** `C:\Program Files\Java\jdk1.7.0_45\db`

5. Aponte o item **Localização do Banco de Dados** para qualquer diretório com permissão de escrita do seu usuário, como por exemplo uma nova pasta chamada JavaDB em Documentos:
   - **Localização do Banco de Dados:** `C:\Users\rafael.alencar\Documents\JavaDB`

6. Clique em **OK** e observe que no console se será criada uma aba **Processo banco de dados Java DB**, e um texto parecido com o abaixo deve aparecer:

```
Thu Aug 22 23:37:01 BRT 2013 : Security manager installed using the Basic server security policy.
Thu Aug 22 23:37:02 BRT 2013 : Apache Derby Network Server - 10.8.1.2 - (1095077) started and ready to accept connections on port 1527
```

7. Clique novamente com o botão direito no banco **Java DB** > **Criar banco de dados...**

8. Entre com as seguintes informações:
   Nome do Banco de Dados: exemploJPA
   Nome do Usuário: dba
   Senha: dba
   E clique em **OK**.

9. Observe que ainda dentro do item **Banco de Dados** aparecerá uma conexão com o texto:
   `jdbc:derby://localhost:1527/exemploJPA [dba em DBA]`
   Clique bom o botão direito sobre ela, **Conectar**

10. Clique com o botão direito na conexão mencionada e acesse a opção **Executar Comando.** Cole o script DDL/SQL abaixo e execute-o:

```
create table cores(id int, descricao varchar(40));
insert into cores values (1,'Verde');
insert into cores values (2,'Amarelo');
insert into cores values (3,'Azul');
insert into cores values (4,'Branco');
```

11. Você deverá ver como resultado da execução do script algo parecido com:

```
Executado com sucesso em 0,121 s, 0 linhas afetadas.
Linha 1, coluna 1

Executado com sucesso em 0,136 s, 1 linhas afetadas.
Linha 2, coluna 1

Executado com sucesso em 0,005 s, 1 linhas afetadas.
Linha 3, coluna 1

Executado com sucesso em 0,005 s, 1 linhas afetadas.
Linha 4, coluna 1

Executado com sucesso em 0,005 s, 1 linhas afetadas.
Linha 5, coluna 1

Execução finalizada após 0,272 s, 0 erro(s) encontrado(s).
```

12. Explore essa visualização do Netbeans. Ele permite gerenciar conexões com bancos e administrar suas tabelas. Encontre a tabela criada no script acima e exiba seus dados.

Fase 2: Conectando manualmente no banco com JDBC

Vamos testar a conexão de uma aplicação Java com o banco de dados SQL Server utilizando apenas o recurso do JDBC para relembrar alguns pontos.

1. Crie um novo projeto Java **(não é Web)** no Netbeans

2. Adicione à lista de bibliotecas do projeto o arquivo .JAR do Driver JDBC do Apache Derby disponível no SOL (**derbyclient.jar ou similar**):
   Botão direito no projeto > Propriedades > Bibliotecas > Adicionar Jar/Pasta

3. Crie uma classe com o método abaixo. Não esqueça de importar as classes necessárias do pacote **java.sql**

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class ExemploJDBC {

    public static void main(String[] args) throws Exception {

        Class.forName("org.apache.derby.jdbc.ClientDriver");
```

```
        Connection conexao = DriverManager.getConnection("jdbc:derby://localhost:1527/exemploJPA", "dba",
                "dba");
        Statement statement = conexao.createStatement();
        ResultSet rs = statement.executeQuery("SELECT * FROM cores");
        while(rs.next()){
            int id = rs.getInt("id");
            String nome = rs.getString("descricao");
            System.out.println("id="+id+"; nome="+nome);
        }

        conexao.close();
    }
}
```

4. Execute o código acima para certificar que foi possível conectar no banco criado e listar as cores previamente cadastradas no banco.

5. Se tudo estiver funcionando, podemos passar adiante. O código criado até agora só foi utilizado para testes e para relembrar qual é a tecnologia fundamental por trás de toda conexão com banco de dados do Java, o **JDBC**. Relembre um pouco dessa tecnologia respondendo as perguntas abaixo.

RESPONDA:

I. Qual o papel do *Driver JDBC* no código acima?
II. O que é um *Statement*?
III. Qual a diferença entre a chamada *execute()* e *executeQuery()* no JDBC?
IV. O que é um ResultSet?
V. O que faz o método *next()* do ResultSet?
VI. Como se obtêm o valor de uma coluna de um registro do banco de dados com JDBC?

Fase 3: Configurando o JPA na aplicação

Agora faremos com que nossa aplicação possa utilizar o JPA efetivamente.

1. Adicione à lista de bibliotecas da aplicação já criada o item **JPA do Hibernate** disponibilizado pelo próprio Netbeans:
   Botão direito no projeto > Propriedades > Bibliotecas > Adicionar Biblioteca)

2. Crie uma nova Unidade de Persistência:
   Botão direito no projeto > Novo > Outro > Persistência > Unidade de Persistência.

3. Na tela seguinte, escolha os seguintes parâmetros:
   Nome da unidade de persistência: unidade-exemploJPA
   Biblioteca de persistência: Hibernate (JPA 2.0)
   Conexão com o banco de dados: `jdbc:derby://localhost:1527/exemploJPA [dba em DBA]` *(a mesma criada na Fase 1)*
   Estratégia de geração de tabela: Eliminar e criar

4. Clique em Finalizar.

5. Observe que agora seu projeto possui no código-fonte um novo arquivo:
   META-INF/persistence.xml
   Clique bom o botão direito sobre esse arquivo e escolha **Editar.**

6. Observe atentamente o conteúdo (código-fonte) do arquivo **persistence.xml**

7. Observe também que novas bibliotecas foram adicionadas ao seu projeto

RESPONDA:

I. O que é uma Unidade de Persistência?

II. Qual a relação do arquivo **persistence.xml** gerado na Fase 4 com a conexão JDBC criada na Fase 1?
III. Onde no arquivo **persistence.xml** podemos mudar o nome de uma Unidade de Persistência?
IV. Se quiséssemos conectar nossa aplicação com 2 bancos de dados distintos, o que poderíamos fazer?

Fase 4: Mapeando sua primeira entidade

Para exercitarmos o Mapeamento Objeto-Relacional (ORM), precisamos pensar na modelagem Orientada a Objeto. Mãos a obra!

1. Crie a classe **Carro** no pacote **exemplojpa.entidade**

2. Essa classe deve ser um JavaBean com as propriedades: **id** (long), **placa** (String), **cor** (String) e **modelo** (String). Não se esqueça dos getters e setters!

3. Inclua a anotação **@Entity** na classe **Carro,** e faça-a implementar a interface *java.io.Serializable*; Inclua as anotação **@Id** e **@GeneratedValue** no atributo id da classe Carro.

```
...
@Entity
public class Carro implements Serializable{

    @Id
    @GeneratedValue
    private Long id;

   ...
}
```

   Não esqueça de importar as classes necessárias do pacote **javax.persistence**

4. Crie a classe **ExemploJPA** no pacote **exemploJPA** com o seguinte método main:

```
public static void main(String [] args) throws Exception{

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("unidade-exemploJPA");
        EntityManager em = emf.createEntityManager();

        em.getTransaction().begin();

        Carro uno = new Carro();
        uno.setPlaca("ABC-0001");
        uno.setModelo("Uno");
        uno.setCor("Prata");

        em.persist(uno);

        Carro gol = new Carro();
        gol.setPlaca("ABC-0002");
        gol.setModelo("Gol");
        gol.setCor("Preto");

        em.persist(gol);

        em.getTransaction().commit();

        List<Carro> carros = em.createQuery("from Carro").getResultList();
        for(Carro c:carros){
            System.out.println(c.getPlaca());
        }

        System.out.println("Pressione uma tecla para continuar...");
        System.in.read();

        em.close();
        emf.close();
}
```

5. Execute a classe ExemploJPA. Deverá ser visto na saída as placas dos carros inseridos no banco:

```
ABC-0001
ABC-0002
```

6. Antes de pressionar alguma tecla para continuar, vá na aba **Serviços** novamente investigue no banco de dados **exemploJPA** criado anteriormente se existe alguma tabela armazenando os registros de Carro. Observe o nome da tabela, bem como o nome e o tipo das colunas e os registros existentes nela.

RESPONDA:

I. O que é um Entity Manager? Como o EntityManager foi criado? Que informação precisou ser informada na sua criação?

II. O que é uma transação de banco de dados? Quais linhas de código (duas) foram responsáveis por inciar e finalizar a transação?

III. Que método foi chamado para persistir uma entidade no banco de dados?

IV. Que método foi chamado para listar entidades do banco de dados? O que foi informado para esse método?

V. Quem criou o banco de dados?

VI. Compare as linhas de código utilizadas para listar os Carros através do JPA com as linhas de código utilizadas para listar as Cores através do JDBC. Que facilidades foram trazidas pelo JPA?