Roteiro para a configuração de DataSource em Servidor JBoss com JavaDB

Uma aplicação web normalmente necessita conectar-se a um Banco de Dados a fim de armazenar e obter informações, seja para salvar dados de uma tela ou mesmo emitir um relatório ou ainda efetuar um cálculo.

A conexão ao Banco de Dados em Java utiliza a tecnologia JDBC e para ser estabelecida necessita de algumas informações:

- URL de conexão que identifica o servidor a porta de comunicação a instância de dados;
- Class Java que implementa o Driver de acesso ao Banco de Dados e que é fornecida normalmente pelo desenvolvedor do Banco de Dados;
- Credenciais de conexão que são o nome de um usuário com acesso ao banco e a respectiva senha.

A classe que implementa a transação Web seja Servlet ou JSP pode ter também a responsabilidade por criar a conexão ao banco de dados e para isto necessitará utilizar dos mesmos recursos que são usados numa classe construída para a plataforma JavaSE ou ainda fazer uso de JSTL (Java Standard Tag Library) no namespace “sql”, vide exemplos abaixo:

```
try {
  Connection conn =  DriverManager.getConnection("jdbc:derby:DerbyTestDB");
  Statement s = conn.createStatement();
  ResultSet rs = s.executeQuery("SELECT Cidade, Estado, Cep FROM cepTable");

  StringBuilder sb = new StringBuilder("Resultado:\n\n");
  while(rs.next()) {
    sb.append("Cidade   : " + rs.getString(1));
    sb.append(" Estado  : " + rs.getString(2) + "\t");
    sb.append(" Cep: " + rs.getString(3) + "\n");
   }

  JOptionPane.showMessageDialog(null, sb.toString());

  rs.close();
  s.close();
  conn.close();
 } catch(Exception ex) {
  System.out.println("Erro: " + ex.getMessage());
  ex.printStackTrace();
 }
}
```

Codificação em JavaSE para acessar informações em um Banco de Dados

```
<sql:setDataSource var="datasource" driver="org.apache.derby.jdbc.ClientDriver"
                   url="jdbc:derby://localhost/escola" user="root" password="senai" />
<sql:query var="lista" dataSource="${datasource}"
           sql="SELECT * FROM amigo" />
<c:choose>
<c:when test="${lista.rowCount == 0}">
   <p>N&atilde;o existem Amigos Cadastrados!<br><br>
</c:when>
<c:otherwise>
   <p>Total de registros encontrados: <c:out value="${lista.rowCount}"/>
   <form action="processaDel.jsp" method="post">
   <table border="1">
    <tr><th>Del</th><th>Nome</th><th>E-Mail</th></tr>
    <c:forEach var="amigo" items="${lista.rows}">
    <tr><td><input type="checkbox" name="amigo" value="${amigo.idAmigo}" /></td>
    <td><c:out value="${amigo.nome}" /></td>
    <td><c:out value="${amigo.email}" /></td></tr>
    </c:forEach>
   </table><br>
   <input type="submit" value="Deletar os itens selecionados"> 
   </form>
</c:otherwise>
</c:choose>
```

Codificação em JSTL para efetuar acesso ao Banco de Dados

Embora seja possível a implementação da conexão ao Banco de Dados diretamente nas classes que a utilizam isto não é recomendado para aplicações Web, pois o custo de estabelecer uma conexão é alto e a classe normalmente é utilizada por vários usuários. Assim a recomendação é a criação de um DataSource no Servidor onde a aplicação é publicada.

DataSource representa um recurso configurado que provê e gerencia estas conexões além de manter um pool de conexões que é otimizado para ser mais eficiente na entrega deste recurso para as aplicações.

As configurações de um DataSource num Application Server difere de uma para outro. No Jboss a criação de um DataSource pode ser feito através da publicação de um arquivo XML que descreve as configurações necessárias para o estabelecimento de tal conexão. Vide exemplo a seguir:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE datasources PUBLIC "-//JBoss//DTD JBOSS JCA Config 1.5//EN"
                     "http://www.jboss.org/j2ee/dtd/jboss-ds_1_5.dtd">
<datasources>
  <xa-datasource>
    <jndi-name>Escola</jndi-name>
    <track-connection-by-tx />
    <xa-datasource-class>org.apache.derby.jdbc.EmbeddedXADataSource</xa-datasource-class>
    <xa-datasource-property name="DatabaseName">/root/Desktop/escola</xa-datasource-property>
    <xa-datasource-property name="User">root</xa-datasource-property>
    <xa-datasource-property name="Password">senai</xa-datasource-property>
    <isSameRM-override-value>false</isSameRM-override-value>
    <min-pool-size>5</min-pool-size>
    <max-pool-size>20</max-pool-size>
    <idle-timeout-minutes>5</idle-timeout-minutes>
    <metadata>
      <type-mapping>Derby</type-mapping>
    </metadata>
  </xa-datasource>
</datasources>
```

XML de configuração de um DataSource para o Servidor JBoss

O arquivo XML acima declara um DataSource no elemento XML “jndi-name” é definido seu nome : “Escola”. A este nome é acrescentado “java:” uma vez este arquivo seja publicado. Assim teremos o nome “java:Escola” como o recurso JNDI (Java Naming and Directory Interface) a ser localizado no momento em que a aplicação necessitar.

Toda a vez que uma solicitação a um DataSource é recebida pelo servidor para o recurso chamado “java:Escola” uma conexão do pool de conexões é entregue em resposta. Se a quantidade de conexões excede o número mínimo declarado em “min-pool-size”, mais conexões são estabelecidas até o limite máximo definido em “max-pool-size”.

Quando uma conexão é liberada pela aplicação, esta volta ao pool de conexões. O número de conexões que excede o valor mínimo ficará ativa até o tempo definido em “idle-timeout-minutes” expirar.

Uma vez um DataSource esteja publicado, deveremos criar um mapeamento deste nome JNDI para que seja possível utilizá-la numa aplicação WEB. Este mapeamento é feito em dois arquivos XML, um chamado “jboss-web.xml” e outro “web.xml”, ambos ficam no diretório “WEB-INF” da aplicação que utilizará este recurso. O primeiro arquivo informa ao servidor Jboss qual será o nome que a aplicação irá utilizar quando necessitar deste recurso.

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-web>
  <resource-ref>
    <res-ref-name>jdbc/Escola</res-ref-name>
    <jndi-name>java:Escola</jndi-name>
  </resource-ref>
</jboss-web>
```

Arquivo jboss-web.xml de mapeamento para recursos Web

O segundo arquivo identifica o nome do recurso a ser publicado para a aplicação e o tipo de objeto Java que este representa.

```
<resource-ref>
  <res-ref-name>jdbc/Escola</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

Fragmento da configuração para a associação de um recurso a um DataSource

A partir do momento que utilizamos um DataSource para a ser responsabilidade do Servidor de Aplicações o estabelecimento das conexões ao Banco de dados bastando para a aplicação somente obter o acesso a este recurso, conforme demonstrado a seguir.

```
<sql:setDataSource var="datasource" dataSource="jdbc/Escola" />
<sql:query var="lista" dataSource="${datasource}"
        sql="SELECT * FROM amigo" />
<c:choose>
<c:when test="${lista.rowCount == 0}">
   <p>N&atilde;o existem Amigos Cadastrados!<br><br>
</c:when>
<c:otherwise>
   <p>Total de registros encontrados: <c:out value="${lista.rowCount}"/>
   <form action="processaDel.jsp" method="post">
   <table border="1">
    <tr><th>Del</th><th>Nome</th><th>E-Mail</th></tr>
    <c:forEach var="amigo" items="${lista.rows}">
    <tr><td><input type="checkbox" name="amigo" value="${amigo.idAmigo}" /></td>
    <td><c:out value="${amigo.nome}" /></td>
    <td><c:out value="${amigo.email}" /></td></tr>
    </c:forEach>
   </table><br>
   <input type="submit" value="Deletar os itens selecionados"> 
   </form>
</c:otherwise>
</c:choose>
```

Aplicação utilizando um DataSource publicado no servidor de Aplicações Web

A utilização de instruções SQL diretamente nas classes também foi delegado a outra tecnologia, onde nesta efetuamos o mapeamento de estruturas para classes java, declaramos os tipos dos dados, queries SQL nomeadas, etc.

Nas classes que utilizam os dados é utilizado um conjunto de métodos especializados em obter objetos a partir das consultas geradas, e desta forma o desenvolvedor utiliza somente objetos ao invés de ter que mapear os dados em algum tipo Java.

Esta tecnologia chama-se JPA (Java Persistence Application Program Interface), inicialmente nasceu como uma solução EJB (Enterprise Java Bean) e somente poderíamos utilizar se criássemos um EJB, que na época de seu lançamento era muito trabalhoso para construir.

Posteriormente a comunidade Open Source criou uma solução chamada de Hibernate, que foi adquirida pela Jboss e esta adquirida pela Red Hat. Outras soluções de mapeamento Objeto – Relacional também apareceram tal como JDO. Na época a Sun estava criando a especificação EJB 3 e aí nasceu o JPA.

Como resposta a comunidade a especificação JPA nasceu com o apoio da Jboss/Hibernate e Oracle (Hoje dona da Sun e do Java).

Sobre JPA tratarei noutro documento.