

# Desenvolvimento Web II

## Aula 10 - Projeto (Parte 1): Incorporando o Banco de Dados

# Apresentação

---

Nas aulas anteriores, você aprendeu diversas tecnologias para desenvolvimento de sistemas web. Além disso, na disciplina de Desenvolvimento Web I, você desenvolveu uma aplicação web de Livraria Virtual em JSP. A partir desta aula, você integrará o recurso de banco de dados nessa aplicação JSP e nas aulas mais a frente você irá aprender a adicionar banco de dados em na aplicação SITURB em JSF. Assim teremos aplicações, tanto em JSP como em JSF mais completas que possam até servir como ponto de partida para você desenvolver um produto comercial.

Nesta aula, você irá, por conta própria, implementar melhorias na sua Livraria Virtual. Não se preocupe se o seu projeto não está completo pois será disponibilizado o projeto base da livraria para você utilizar durante as próximas aulas. Em particular, essas melhorias serão implementadas para fazer com que o seu sistema possa armazenar os dados em um banco de dados. Iremos descrever em detalhes os novos requisitos que devem ser implementados no sistema e você deve analisá-los e implementá-los da melhor maneira possível, usando o conhecimento que foi aprendido ao longo desta e de outras disciplinas do curso.

Aproveitem!



**Vídeo 01** - Apresentação

## Objetivos

- Saber utilizar um banco de dados em um projeto de sistema web.
- Entender de que maneira implementar a classe RepositorioLivrosJDBC, responsável por fazer acesso ao banco de dados.

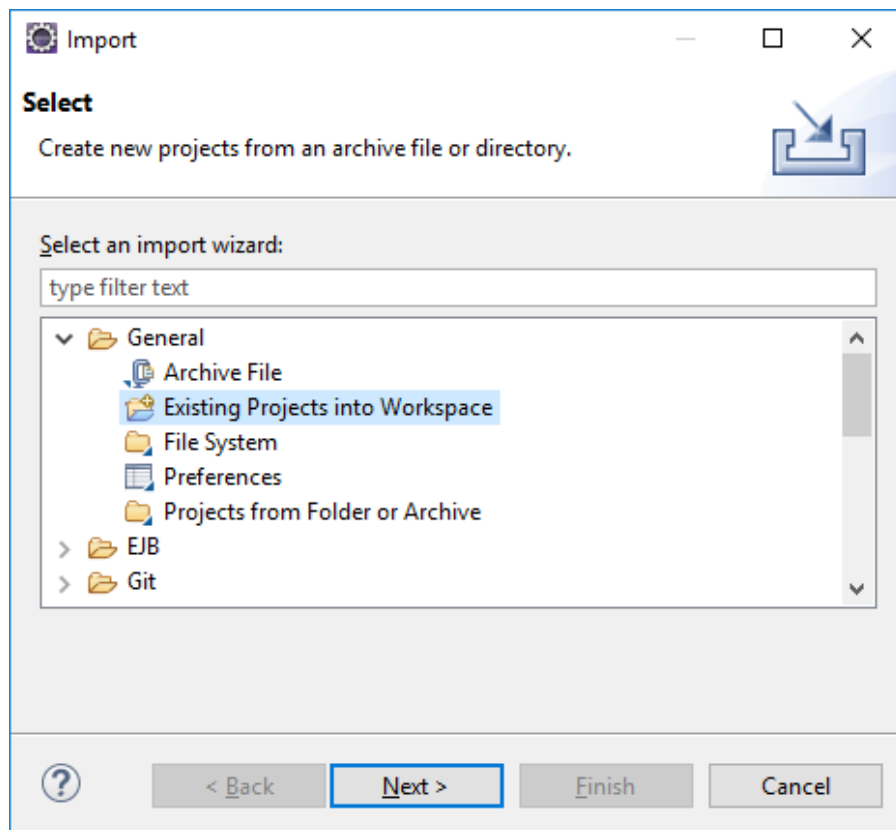
# Utilizar o Banco de Dados

---

Para acompanhar as próximas aulas, baixe o arquivo livraria.zip no link [aqui](#) com um projeto base da livraria para que possamos trabalhar no mesmo. Para adicionar esse projeto no Eclipse siga os seguintes passos:

1. Inicialmente o descompacte o arquivo livraria.zip na pasta Documentos
2. Eclipse escolha a opção File -> Import... e na janela que aparece escolha General -> Existing Projects into Workspace.

**Figura 01** - Importando o projeto



3. Em Select root Directory clique em "Browse..." e escolha a pasta onde está o projeto da livraria descompactado.
4. Clique em Finish para adicionar o projeto no seu workspace e na lista do Project Explorer.

Nesta etapa, será necessário usar um SGBD para armazenar os dados da Locadora. Como você já viu exemplos na disciplina de Banco de Dados usando o MySQL, sugerimos usá-lo aqui também. Apesar disso, se você optar pelo uso de outro tipo de banco de dados, poucos ajustes serão necessários (normalmente, só o procedimento de conexão que deve mudar).

A seguir, são descritos os passos necessários para cumprir o objetivo desse novo requisito:

**Passo 1:** Instalar o SGBD MySQL <<http://dev.mysql.com/downloads/mysql/>>, caso não esteja instalado em sua máquina.

**Passo 2:** Baixar o driver do MySQL.

Para obter o driver do MySQL, siga os passos:

- a. Acesse o site <<http://dev.mysql.com/downloads/connector/j/>>.
- b. Em "Select Operating System:" escolha a opção "Platform Independent".
- c. Baixe a segunda opção (mysql-connector-java-8.0.11.zip). Antes do download iniciar será apresentado uma página solicitando que você faça login mas isso não é necessário. Clique somente na opção "No thanks, just start my download." e o download iniciará.
- d. Salve e descompacte esse arquivo na sua pasta Documentos.
- e. Será criada uma pasta chamada "mysql-connector-java-8.0.11". Dentro dela existe o arquivo "mysql-connector-java-8.0.11.jar". Esse JAR é o que queremos. Copie ele diretamente para a pasta Documentos para que seja mais fácil de o localizar posteriormente.

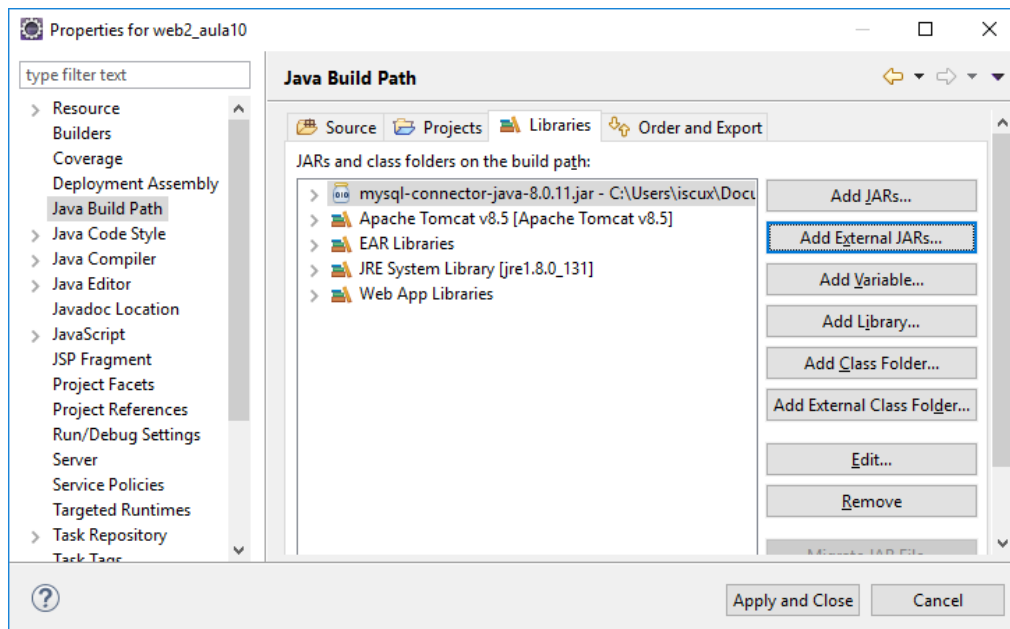
A versão que aparecerá para download é sempre a mais recente e na data em que foi escrita essa aula é a 8.0.11.

**Passo 3:** Configurar o driver no seu ambiente de desenvolvimento (iremos mostrar como fazer no Eclipse, mas isso pode ser feito em qualquer outro ambiente de forma similar).

- a. Clique com o botão direito do mouse em cima de seu projeto web no Eclipse.

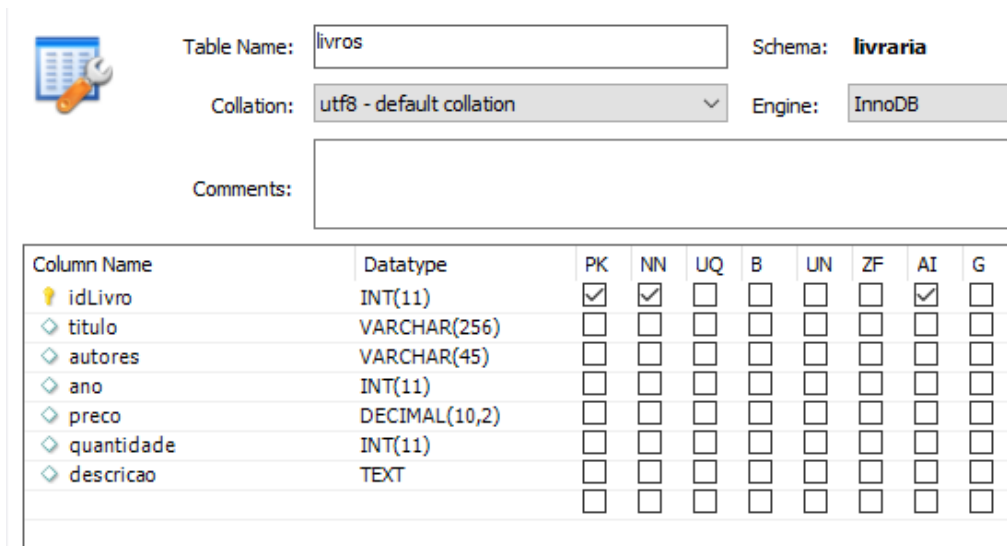
- b. Selecione a opção Properties (ou Propriedades).
- c. Depois, selecione o item “Java Build Path”, do lado esquerdo.
- d. Em seguida, selecione a aba “Libraries” (ou Bibliotecas), conforme mostrado na Figura 2.
- e. Depois, clique no botão “Add External Jars”, selecione o arquivo .jar do driver, que você acabou de descompactar, e, por fim, clique em OK.
- f. Para maior compatibilidade adicione também o arquivo mysql-connector-java-8.0.11.jar na pasta WebContent/WEB-INF/lib do projeto. Nas versões dos serviços que estamos utilizando esse passo é necessário para se carregar o driver JDBC do MySQL

**Figura 02** - Bibliotecas do projeto no Eclipse



**Passo 4:** Com o conector MySQL adicionado no seu projeto utilize as técnicas aprendidas na disciplina de Banco de Dados e crie no MySQL um novo banco (schema) chamado “livraria” e em seguida crie uma nova tabela chamada “livros” com as colunas correspondentes, como visto na Figura 3.

**Figura 03** - Colunas da tabela livros.



The screenshot shows the MySQL Workbench interface for creating a table named 'livros' in the 'livraria' schema. The table is using the 'utf8 - default collation' and 'InnoDB' engine. The columns are defined as follows:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G
idLivro	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
titulo	VARCHAR(256)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
autores	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
ano	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
preco	DECIMAL(10,2)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
quantidade	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
descricao	TEXT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Durante o processo de criação da tabela livros você deve ter visto o comando SQL de criação da tabela como o a seguir:

```
1 CREATE TABLE `livros` (  
2   `idLivro` varchar(20),  
3   `titulo` varchar(256) DEFAULT NULL,  
4   `autores` varchar(45) DEFAULT NULL,  
5   `ano` integer,  
6   `preco` decimal(10,2) DEFAULT NULL,  
7   `quantidade` integer DEFAULT 0,  
8   `descricao` text,  
9   PRIMARY KEY (`idLivro`)  
10 ) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

Repare que utilizamos os tipos VARCHAR para os campos de texto com os limites especificados, usamos varchar(20) para o idLivro já que esse campo representa seu ISBN, usamos DECIMAL(10,2) para o “preco” já que esse é um valor monetário relativamente baixo só precisa de um máximo de 10 dígitos e 2 casas decimais. É também uma boa prática utilizar chaves primárias inteiras do tipo INT AUTO\_INCREMENT já que esses campos são automaticamente preenchido, entretanto no nosso exemplo vamos utilizar o código ISBN do livro como chave já que o projeto da livraria foi desenvolvido dessa forma.

**Passo 5:** Revisar com atenção se você fez tudo corretamente antes de iniciar a próxima etapa.

**Passo 6:** No projeto “livraria” que você está utilizando como base, fazer o seguinte teste para ver se o driver JDBC do MySQL está sendo carregado:

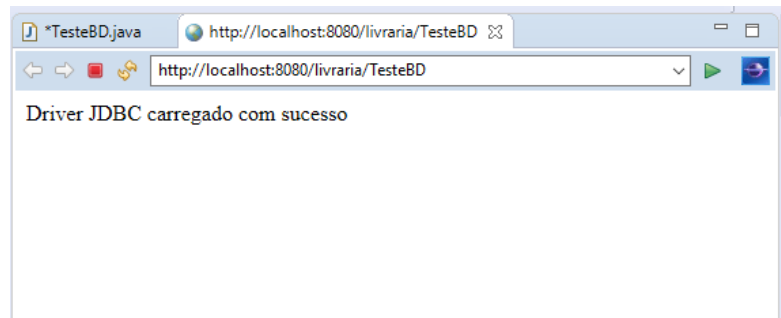
- a. Crie um servlet chamado TesteBD dentro do pacote livraria.servlet com o seguinte código:

```
1 package livraria.servlet;
2
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet("/TesteBD")
13 public class TesteBD extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15
16     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException {
17         doGet(request, response);
18     }
19
20     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
21         String driverName = "com.mysql.cj.jdbc.Driver";
22         PrintWriter writer = response.getWriter();
23
24
25         try {
26             Class.forName(driverName);
27             writer.append("Driver JDBC carregado com sucesso");
28         } catch (ClassNotFoundException e) {
29             writer.append("Driver não carregado!");
30             e.printStackTrace();
31         }
32     }
33 }
```

- b. Execute o servlet e veja em localhost:8080/livraria/TesteBD que o resultado exibido deve ser “Driver JDBC carregado com sucesso”:



**Figura 04** - Resultado exibido.



**Passo 7:** Utilize seus conhecimentos de banco de dados para adicionar diretamente no banco alguns livros para teste com todos os campos preenchidos. Utilize os seguintes comandos para inserir 2 livros:

```
1 INSERT INTO `livraria`.`livros` (`idLivro`, `titulo`, `autores`, `ano`, `preco`, `quantidade`, `des  
2  
3 INSERT INTO `livraria`.`livros` (`idLivro`, `titulo`, `autores`, `ano`, `preco`, `quantidade`, `des
```

## Atividade 01

1. Realize os passos apresentados para ajustar o sistema de livraria virtual.

## Implementar a Classe RepositorioLivrosJDBC

Todo o acesso aos dados dos livros será feito pela classe RepositorioLivrosJDBC. Nesta classe, os livros não serão armazenados em memória (como era feito antes, usando-se uma lista do tipo List<Livro>). Os pontos a seguir detalham o passo a passo que deve ser feito para implementar essa nova classe.

1. Crie um nova classe chamada GerenciadorDeConexoes em um novo pacote chamado "livraria.bd". Essa classe será responsável por carregar o driver e abrir a conexão com o banco de dados. Nesta classe, você deve criar um método para fazer essa ação, conforme foi aprendido na disciplina de Banco de Dados e mostrado no trecho de código a seguir.

```

1 public static java.sql.Connection getConexao() {
2
3     Connection connection = null; // atributo do tipo Connection
4     String driverName = "com.mysql.cj.jdbc.Driver";
5     String serverName = "localhost"; // caminho do servidor do BD
6     String mydatabase = "livraria"; // nome do seu banco de dados
7     String url = "jdbc:mysql://" + serverName + "/" + mydatabase;
8     url += "?serverTimezone=UTC"; // Adicione/remova essa linha se tiver problemas em conectar co
9     String username = "root"; // nome de um usuário de seu BD
10    String password = "123456"; // sua senha de acesso
11
12    try {
13        Class.forName(driverName);
14        connection = DriverManager.getConnection(url, username, password);
15    } catch (Exception e) {
16        e.printStackTrace();
17    }
18    return connection;
19 }

```

2. Adicione um método main nesta classe para testar se sua classe consegue estabelecer a conexão com o banco de dados. Lembre também de adicionar os comandos "import java.sql.Connection;" e "import java.sql.DriverManager;" no início da classe. Confira também se o nome do banco, usuário e senha do banco que criou estão corretos. Depois desse método adicionado sua classe GerenciadorDeConexoes deve ter o seguinte código completo:

```

1 package livraria.bd;
2
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5
6 public class GerenciadorDeConexoes {
7     public static java.sql.Connection getConexao() {
8
9         Connection connection = null; // atributo do tipo Connection
10        String driverName = "com.mysql.cj.jdbc.Driver";
11        String serverName = "localhost"; // caminho do servidor do BD
12        String mydatabase = "livraria"; // nome do seu banco de dados
13        String url = "jdbc:mysql://" + serverName + "/" + mydatabase;
14        url += "?serverTimezone=UTC"; // Adicione/remova essa linha se tiver problemas em conectar
15        String username = "root"; // nome de um usuário de seu BD
16        String password = "123456"; // sua senha de acesso
17
18        try {
19            Class.forName(driverName);
20            connection = DriverManager.getConnection(url, username, password);
21        } catch (Exception e) {
22            e.printStackTrace();
23        }
24        return connection;
25    }
26
27    public static void main(String[] args) {
28        Connection conexao = getConexao();
29    }
30 }

```

3. Crie a classe `RepositorioLivrosJDBC` dentro do pacote *livraria.bd*, com os métodos:

- **Construtor:** Instancia uma variável de conexão utilizando o `GerenciadorDeConexoes`
- **public Livro getLivro(String idLivro):** Obtém do banco o livro com o código passado por parâmetro
- **public List<Livro> getLivros():** Obtém do banco todos os livros
- **public int atualizaLivro(String idLivro, Livro livro):** Atualiza os dados no banco de um livro originalmente com o *idLivro* passado e com os valores dos campos igual aos atributos do objeto *livro* também passado por parâmetro e

com seus atributos diferentes dos que estão no banco de dados.

- **public static void main(String[] args):** Método de testes para os recursos implementados.

O código do RepositorioLivrosJDBC deve ficar assim:

```

1 package livraria.bd;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import java.util.ArrayList;
8 import java.util.List;
9
10 import livraria.negocio.Livro;
11 import livraria.negocio.excecoes.LivroNaoEncontradoException;
12
13 public class RepositorioLivrosJDBC {
14     private Connection conexaoBD;
15     public RepositorioLivrosJDBC() {
16         this.conexaoBD = GerenciadorDeConexoes.getConexao();
17     }
18
19     public Livro getLivro (String idLivro) throws LivroNaoEncontradoException {
20
21         Livro livro = null;
22
23         try {
24             //1. Executa a consulta SQL pra recuperar o livro com esse id
25             PreparedStatement st;
26             st = this.conexaoBD.prepareStatement("SELECT * FROM livros WHERE idLivro=?");
27             st.setString(1, idLivro);
28             ResultSet result = st.executeQuery();
29
30             //2. Recupera cada um dos campos do livro, a partir do objeto ResultSet
31             if (result.first()) {
32                 //3. Instancia o livro com os valores que foram recuperados
33                 livro = new Livro();
34                 livro.setIdLivro(result.getString("idLivro"));
35                 livro.setTitulo(result.getString("titulo"));
36                 livro.setAutores(result.getString("autores"));
37                 livro.setAno(result.getInt("ano"));
38                 livro.setPreco(result.getDouble("preco"));
39                 livro.setQuantidade(result.getInt("quantidade"));
40                 livro.setDescricao(result.getString("descricao"));
41             }
42             st.close();
43         } catch (SQLException e) {
44             e.printStackTrace();
45             throw new LivroNaoEncontradoException("Não foi possível encontrar o livro: " + idLivro);
46         }
47
48         return livro;
49     }
50
51     public List<Livro> getLivros() {

```

```

52
53 List<Livro> livros = new ArrayList<Livro>();
54 try {
55     //1. Executa a consulta SQL pra recuperar todos os livros
56     PreparedStatement st;
57     st = this.conexaoBD.prepareStatement("SELECT * FROM livros");
58     ResultSet result = st.executeQuery();
59
60     //2. Para cada livro que for retornado como resultado:
61     //a. Recupera cada um dos campos do livro, a partir do objeto ResultSet
62     while (result.next()) {
63
64         //b. Instancia um livro com os valores que foram recuperados
65         Livro livro = new Livro();
66
67         livro.setIdLivro(result.getString("idLivro"));
68         livro.setTitulo(result.getString("titulo"));
69         livro.setAutores(result.getString("autores"));
70         livro.setAno(result.getInt("ano"));
71         livro.setPreco(result.getDouble("preco"));
72         livro.setQuantidade(result.getInt("quantidade"));
73         livro.setDescricao(result.getString("descricao"));
74
75         //c. Adiciona na lista de livros;
76         livros.add(livro);
77     }
78 } catch (SQLException e) {
79     e.printStackTrace();
80 }
81
82 return livros;
83 }
84
85 public int atualizaLivro(String idLivro, Livro livro) {
86     int result = 0;
87     try {
88         //1. Executa a consulta SQL pra recuperar o livro com esse id
89         PreparedStatement st;
90         st = this.conexaoBD.prepareStatement("UPDATE livros SET idLivro=?, titulo=?, autores=?, and
91         st.setString(1, idLivro);
92         st.setString(2, livro.getTitulo());
93         st.setString(3, livro.getAutores());
94         st.setInt(4, livro.getAno());
95         st.setDouble(5, livro.getPreco());
96         st.setInt(6, livro.getQuantidade());
97         st.setString(7, livro.getDescricao());
98         st.setString(8, livro.getIdLivro());
99         result = st.executeUpdate();
100     } catch (SQLException e) {
101         e.printStackTrace();
102     }

```

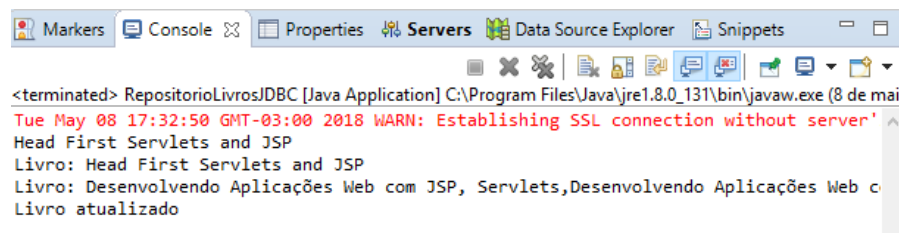
```

103
104     return result;
105 }
106
107
108 // Método main de teste
109 public static void main(String[] args) {
110     RepositorioLivrosJDBC repositorioLivros = new RepositorioLivrosJDBC();
111     try {
112         Livro livro = repositorioLivros.getLivro("0596005407");
113         System.out.println(livro.getTitulo());
114
115     } catch (LivroNaoEncontradoException e) {
116         System.out.println("Livro não encontrado");
117     }
118
119     try {
120         List<Livro> livros = repositorioLivros.getLivros();
121         for (Livro livro : livros) {
122             System.out.println("Livro: " + livro.getTitulo());
123         }
124     } catch (Exception e) {
125         System.out.println(e.getMessage());
126     }
127
128     try {
129         String idLivro = "0596005407";
130         Livro livro = repositorioLivros.getLivro(idLivro);
131         livro.setQuantidade( livro.getQuantidade() - 1);
132         int resultado = repositorioLivros.atualizaLivro(idLivro, livro);
133         if (resultado > 0) {
134             System.out.println("Livro atualizado");
135         }
136     } catch (Exception e) {
137         System.out.println(e.getMessage());
138     }
139 }
140 }

```

4. Realize o teste executando essa classe como um Aplicativo Java e veja o resultado no console do Eclipse como a figura abaixo:

**Figura 05** - Resultado no console.



## Atividade 02

---

1. Realize os passos apresentados para o sistema de livraria virtual.



# Resumo

---

Na aula de hoje, você praticou conhecimentos já adquiridos em outras aulas, inclusive em outras disciplinas para melhorar a Livraria Virtual que você vem desenvolvendo ao longo deste curso. Em especial, nesta aula, você preparou o seu banco de dados para armazenar os dados dos livros e implementou uma classe que será a responsável por acessar o banco de dados. Na próxima aula, você verá como integrar essa classe em seu sistema, tornando a sua livraria mais próxima de um sistema real de comércio eletrônico.

Até lá!

## Autoavaliação

---

1. O que você precisa fazer para poder utilizar um banco de dados em um sistema web?
2. Como se pode implementar a classe RepositorioLivrosJDBC, responsável por fazer acesso ao banco de dados?

## Referências

---

**JAVASCRIPT and HTML DOM reference.** Disponível em: <<http://www.w3schools.com/jsref/>>. Acesso em: 01 ago. 2012.

**W3C.** Disponível em: <<http://www.w3.org/>>. Acesso em: 01 ago. 2012.

**W3SCHOOL.** Disponível: <<http://www.w3schools.com/>>. Acesso em: 01 ago. 2012.

**ESPECIFICAÇÃO do Objeto XMLHttpRequest.** Disponível em: <<http://www.w3.org/TR/XMLHttpRequest/>>. Acesso em: 01 ago. 2012.

ESPECIFICAÇÃO dos códigos de respostas de um servidor HTTP. Disponível em: <<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>>. Acesso em: 01 ago. 2012.