Programação para Web I
AULA 6

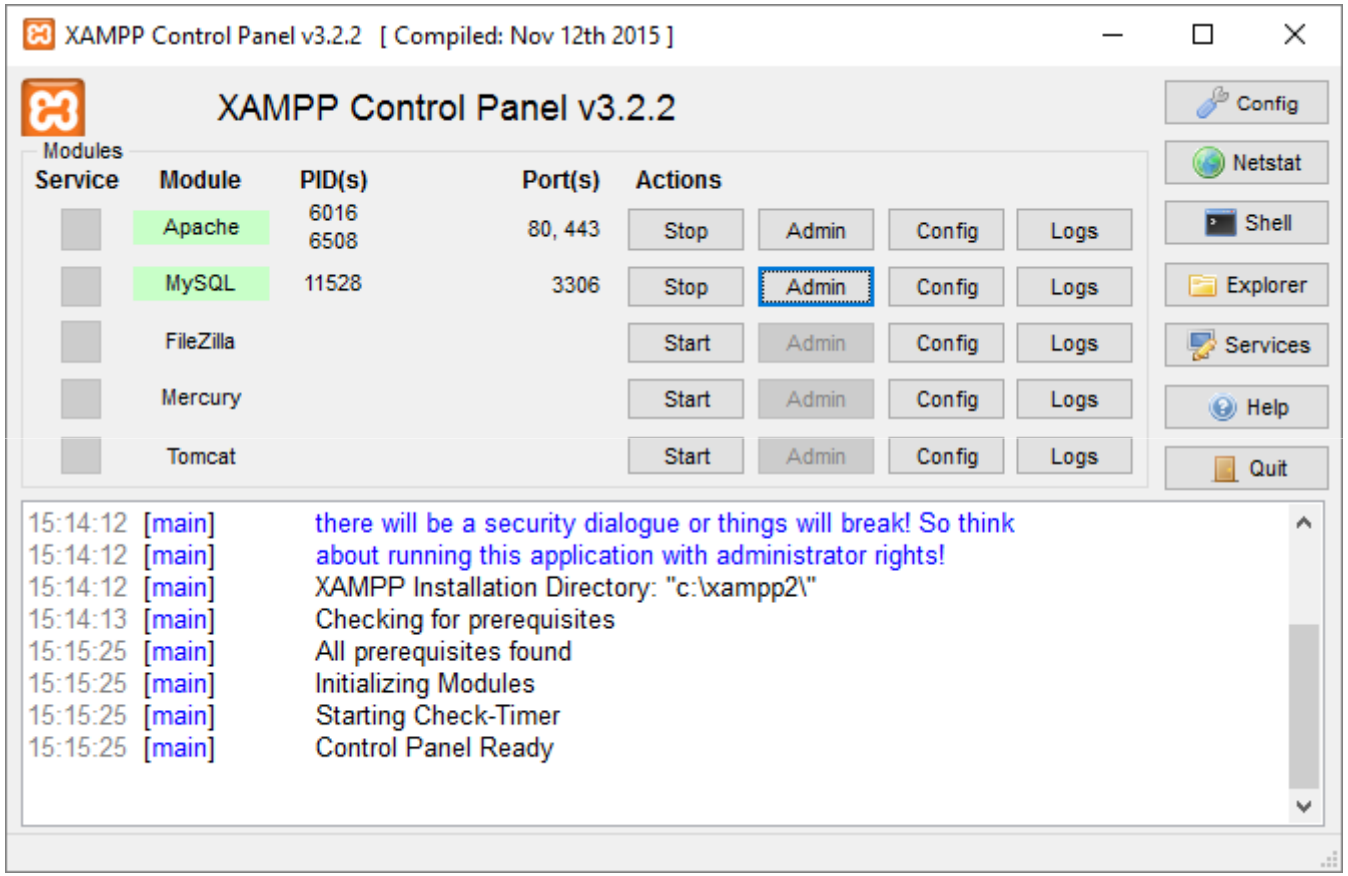
Profa. Silvia Bertagnolli

XAMPP

Primeiros Passos: no xampp

1. Acessar o XAMPP e iniciar o MySQL como serviço

2. Acessar as ferramentas de Admin


XAMPP Control Panel v3.2.2 [Compiled: Nov 12th 2015]
XAMPP Control Panel v3.2.2
Modules
Service
Module
PID(s)
Port(s)
Actions
Apache
6016
6508
80, 443
Stop
Admin
Config
Logs
MySQL
11528
3306
Stop
Admin
Config
Logs
FileZilla
Start
Admin
Config
Logs
Mercury
Start
Admin
Config
Logs
Tomcat
Start
Admin
Config
Logs
Config
Netstat
Shell
Explorer
Services
Help
Quit
15:14:12 [main] there will be a security dialogue or things will break! So think
15:14:12 [main] about running this application with administrator rights!
15:14:12 [main] XAMPP Installation Directory: "c:\xampp2\"
15:14:13 [main] Checking for prerequisites
15:15:25 [main] All prerequisites found
15:15:25 [main] Initializing Modules
15:15:25 [main] Starting Check-Timer
15:15:25 [main] Control Panel Ready

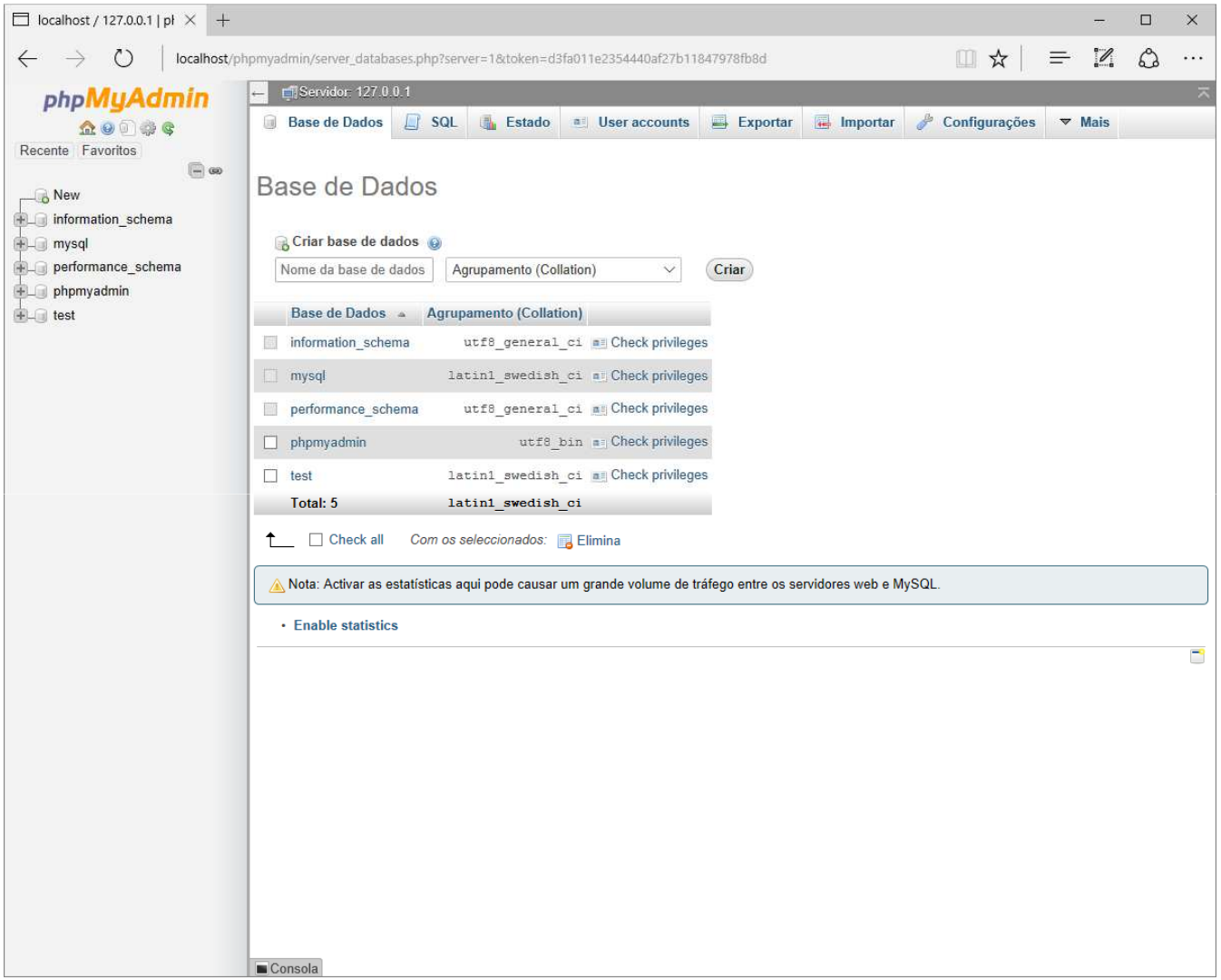
Primeiros Passos: no MySQL

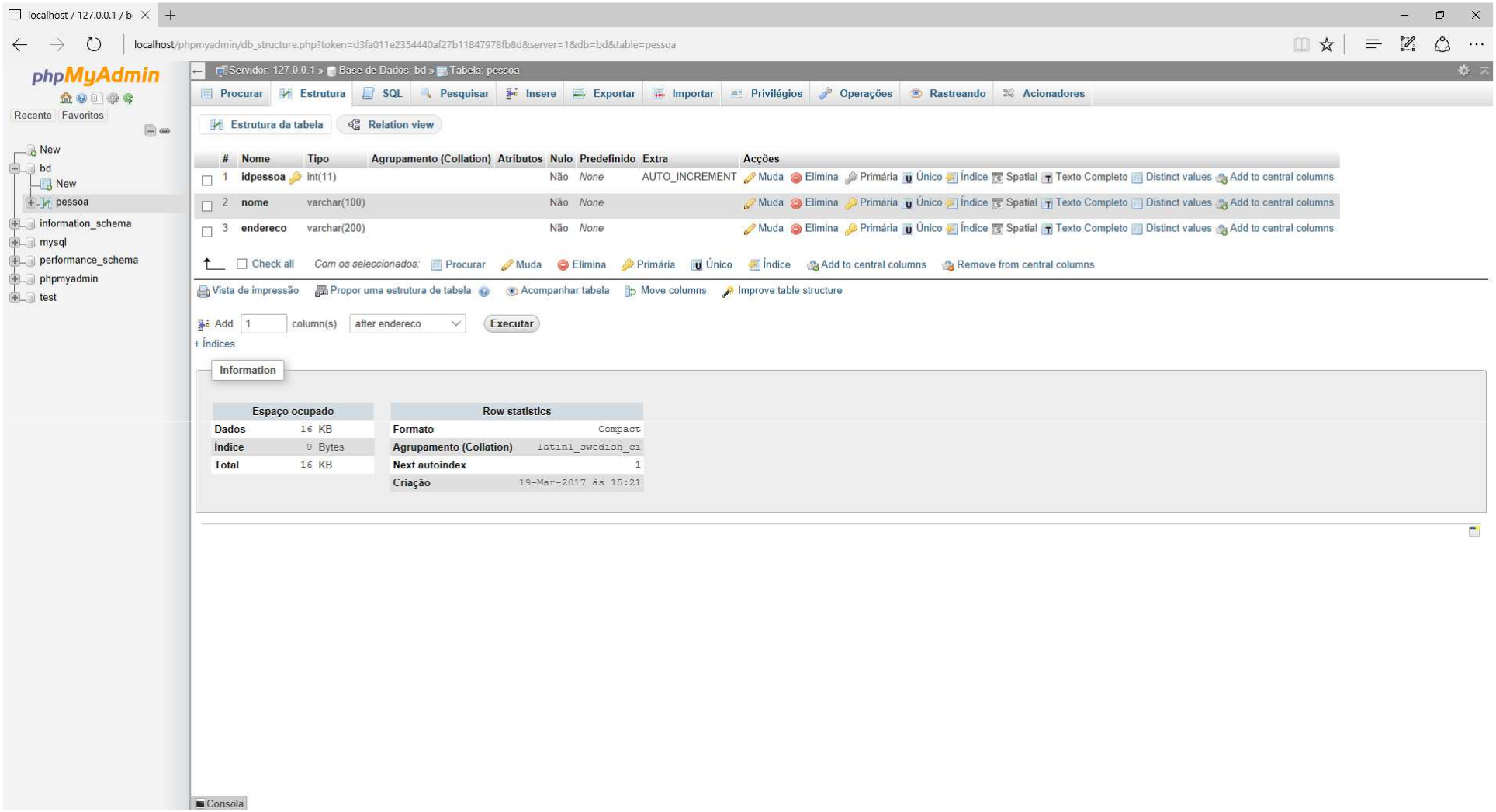
1. Criar o banco de dados **bd**
2. Criar a tabela pessoa com as colunas abaixo (importe o arquivo pessoa.sql):

   idpessoa= int, chave primária, auto incremento e not null

   nome = varchar(100) e not null

   endereco = varchar(200) e not null


localhost / 127.0.0.1 | pl
localhost/phpmyadmin/server_databases.php?server=1&token=d3fa011e2354440af27b11847978fb8d
phpMyAdmin
Recente
Favoritos
New
information_schema
mysql
performance_schema
phpmyadmin
test
Servidor: 127.0.0.1
Base de Dados
SQL
Estado
User accounts
Exportar
Importar
Configurações
Mais
Base de Dados
Criar base de dados
Nome da base de dados
Agrupamento (Collation)
Criar
Base de Dados
Agrupamento (Collation)
information_schema utf8_general_ci Check privileges
mysql latin1_swedish_ci Check privileges
performance_schema utf8_general_ci Check privileges
phpmyadmin utf8_bin Check privileges
test latin1_swedish_ci Check privileges
Total: 5 latin1_swedish_ci
Check all
Com os seleccionados:
Elimina
Nota: Activar as estatísticas aqui pode causar um grande volume de tráfego entre os servidores web e MySQL.
Enable statistics
Consola


localhost / 127.0.0.1 / b
localhost/phpmyadmin/db_structure.php?token=d3fa011e2354440af27b11847978fb8d&server=1&db=bd&table=pessoa
phpMyAdmin
Recente Favoritos
New
bd
New
pessoa
information_schema
mysql
performance_schema
phpmyadmin
test
Servidor: 127.0.0.1 » Base de Dados: bd » Tabela: pessoa
Procurar Estrutura SQL Pesquisar Insere Exportar Importar Privilégios Operações Rastreando Acionadores
Estrutura da tabela Relation view
Nome Tipo Agrupamento (Collation) Atributos Nulo Predefinido Extra Acções
1 idpessoa int(11) Não None AUTO_INCREMENT Muda Elimina Primária Único Índice Spatial Texto Completo Distinct values Add to central columns
2 nome varchar(100) Não None Muda Elimina Primária Único Índice Spatial Texto Completo Distinct values Add to central columns
3 endereco varchar(200) Não None Muda Elimina Primária Único Índice Spatial Texto Completo Distinct values Add to central columns
Check all Com os seleccionados: Procurar Muda Elimina Primária Único Índice Add to central columns Remove from central columns
Vista de impressão Propor uma estrutura de tabela Acompanhar tabela Move columns Improve table structure
Add 1 column(s) after endereco Executar
+ Índices
Information
Espaço ocupado
Dados 16 KB
Índice 0 Bytes
Total 16 KB
Row statistics
Formato Compact
Agrupamento (Collation) latin1_swedish_ci
Next autoindex 1
Criação 19-Mar-2017 às 15:21
Consola

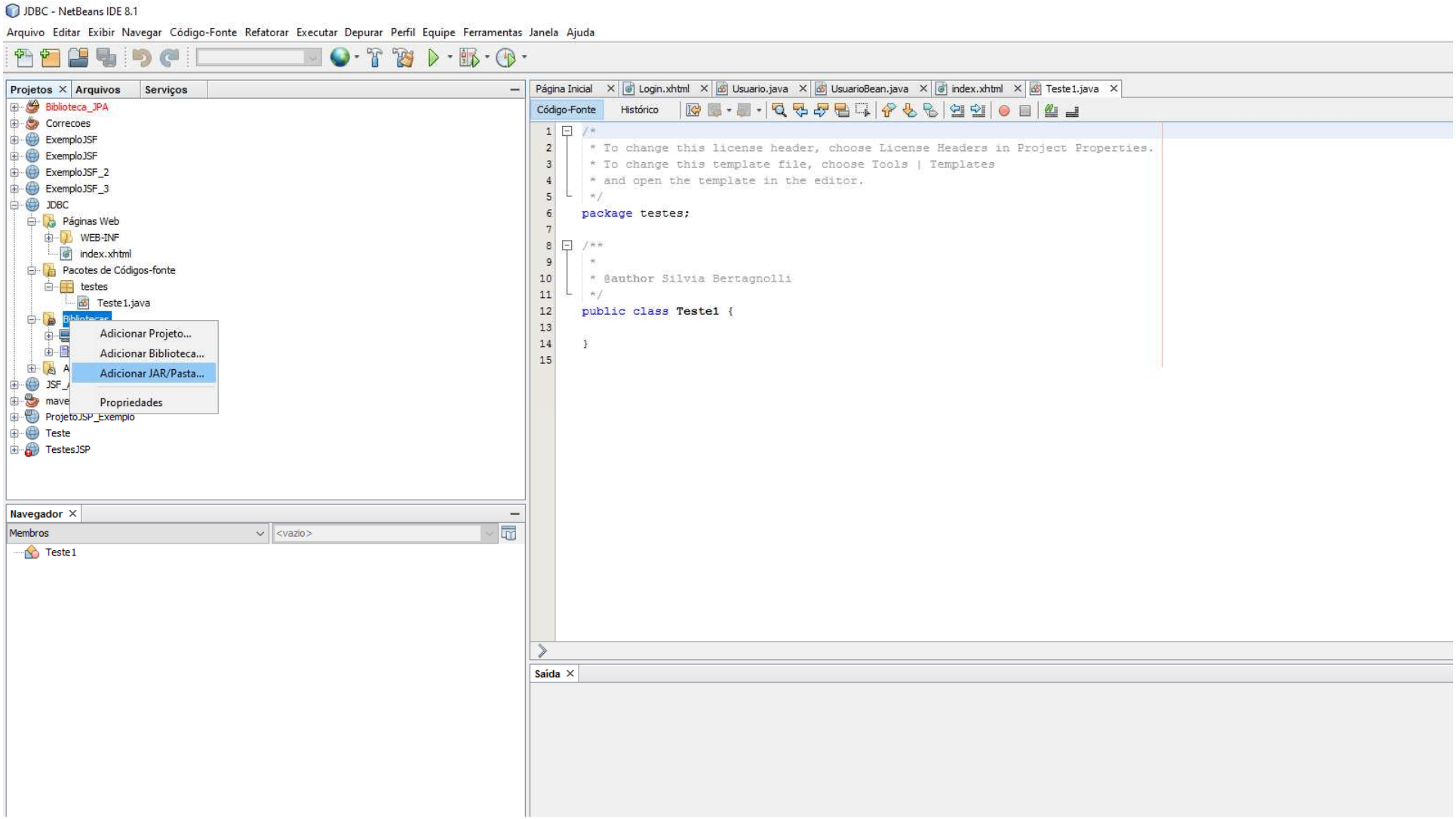
Primeiros Passos: no NetBeans

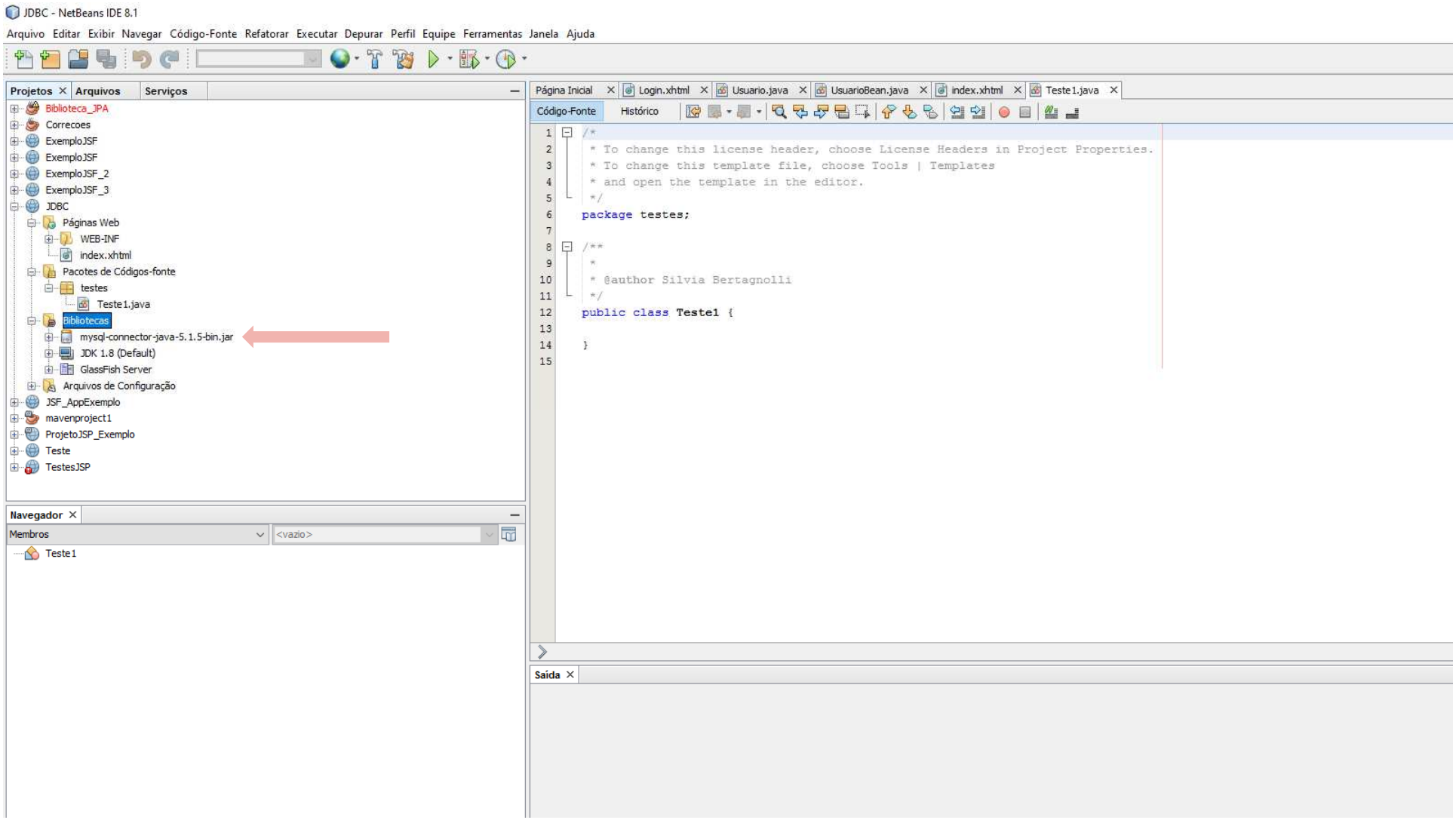
1. Criar um projeto no NetBeans

   Vá em Java -> Aplicação Java

2. Anexar o driver do MySQL ao projeto

   Vá em Bibliotecas -> Adicionar Jar/Pasta


JDBC - NetBeans IDE 8.1
Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda
Projetos
Arquivos
Serviços
Biblioteca_JPA
Correcoes
ExemploJSF
ExemploJSF
ExemploJSF_2
ExemploJSF_3
JDBC
Páginas Web
WEB-INF
index.xhtml
Pacotes de Códigos-fonte
testes
Teste1.java
Bibliotecas
Adicionar Projeto...
Adicionar Biblioteca...
Adicionar JAR/Pasta...
Propriedades
ProjetoJSP_Exemplo
Teste
TestesJSP
Navegador
Membros
<vazio>
Teste1
Página Inicial
Login.xhtml
Usuario.java
UsuarioBean.java
index.xhtml
Teste1.java
Código-Fonte
Histórico
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package testes;
7
8 /**
9 *
10 * @author Silvia Bertagnolli
11 */
12 public class Teste1 {
13
14 }
15
Saída


JDBC - NetBeans IDE 8.1
Arquivo Editar Exibir Navegar Código-Fonte Refatorar Executar Depurar Perfil Equipe Ferramentas Janela Ajuda
Projetos
Arquivos
Serviços
Biblioteca_JPA
Correcoes
ExemploJSF
ExemploJSF
ExemploJSF_2
ExemploJSF_3
JDBC
Páginas Web
WEB-INF
index.xhtml
Pacotes de Códigos-fonte
testes
Teste1.java
Bibliotecas
mysql-connector-java-5.1.5-bin.jar
JDK 1.8 (Default)
GlassFish Server
Arquivos de Configuração
JSF_AppExemplo
mavenproject1
ProjetoJSP_Exemplo
Teste
TestesJSP
Navegador
Membros
<vazio>
Teste1
Página Inicial
Login.xhtml
Usuario.java
UsuarioBean.java
index.xhtml
Teste1.java
Código-Fonte
Histórico
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package testes;
7
8 /**
9 *
10 * @author Silvia Bertagnolli
11 */
12 public class Teste1 {
13
14 }
15
Saída

JDBC

JDBC

Há várias tecnologias que permitem a persistência de dados em Java, nesta aula vamos usar JDBC

JDBC é uma API para persistência de objetos em banco de dados relacionais

Essa API reúne classes e interfaces escritas em Java que permitem a conexão através de um driver específico do banco de dados desejado

Todas as suas classes estão descritas e podem ser encontradas no pacote: java.sql

JDBC

O driver permite executar instruções SQL de qualquer tipo de banco de dados relacional

Para fazer a comunicação entre a aplicação e o SGBDs é necessário possuir um driver para a conexão desejada

```
MySQL => Class.forName("com.mysql.jdbc.Driver");

Acess => Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

Oracle => Class.forName("oracle.jdbc.driver.OracleDriver");

SQL Server =>
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
```

JDBC: CLASSES E INTERFACES

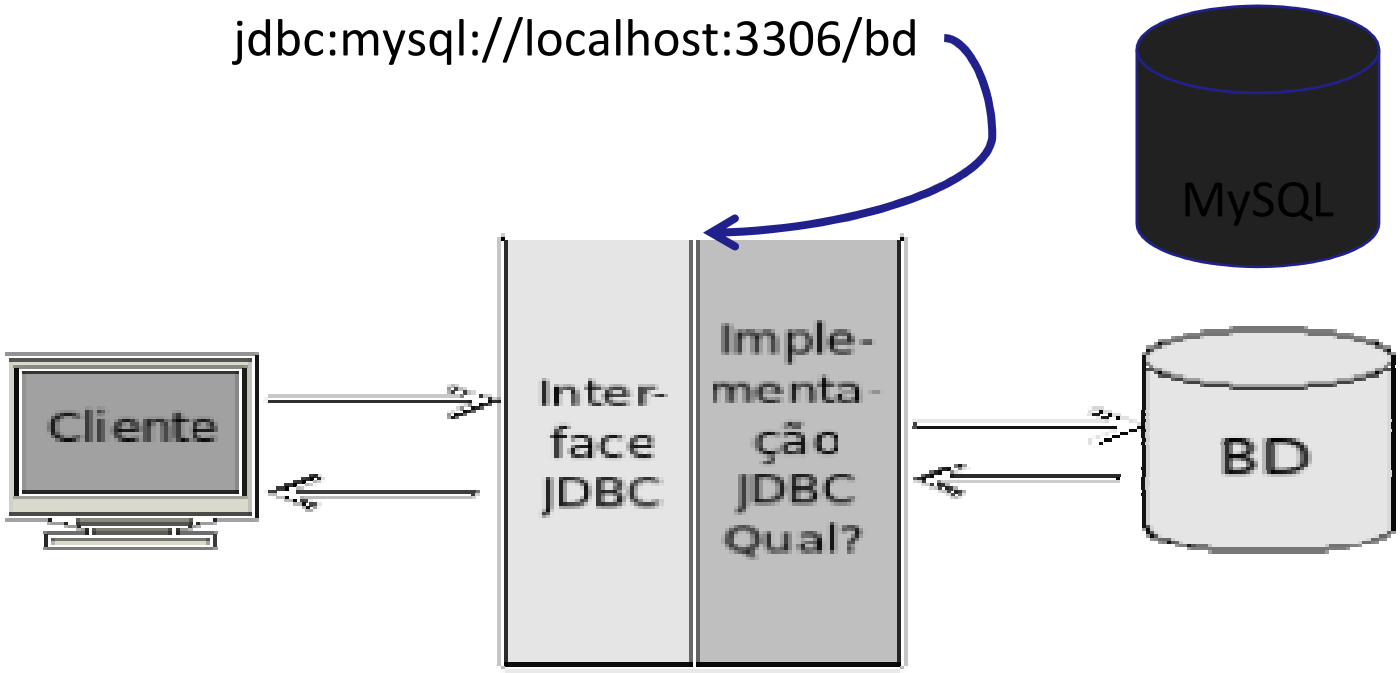
Connection - é a classe que representa a conexão com o BD; ela define métodos para executar consultas em um banco de dados

DriverManager - gerencia o driver e é responsável por localizar o driver de comunicação com o BD

Statement, PreparedStatement e CallableStatement - controlam e permitem executar consultas em um BD

ResultSet - contém o conjunto de dados retornado por uma consulta SQL; essa classe que permite navegar entre os registros

JDBC


jdbc:mysql://localhost:3306/bd
MySQL
Cliente
Inter-
face
JDBC
Imple-
menta-
ção
JDBC
Qual?
BD

Exemplo 1

```
public class Exemplo1{
   public static void main(String[] args)
                                    throws Exception{
     Class.forName("com.mysql.jdbc.Driver");
     String urlBD="jdbc:mysql://localhost:3306/bd";
     Connection conexao =
     DriverManager.getConnection(urlBD, "root", " ");
     System.out.println("Conectou!");
     conexao.close();
   }
}
```

A URL do BD

Sempre inicia com **jdbc**

nome do fabricante (jdbc:db2, jdbc:oracle, jdbc:mysql, etc)

caminho da fonte de dados (//localhost ou //meu-servidor)

nome da fonte de dados ou caminho da fonte de dados

Exercícios

Exercícios

Faça os exercícios 1 a 6 da lista de exercícios ListaExercicios_Aula6.pdf

Factory

Padrão de Projeto: Factory

Os padrões de projeto são uma solução genérica, que resolve algum problema na orientação a objetos

Factory é um padrão que “implementa o design pattern Factory que prega o encapsulamento da construção (fabricação) de objetos complicados”

Nesta aula o padrão funciona como fábrica de conexões com o BD

Ao chamar o método getConnection() é devolvida uma conexão com o BD

ConnectionFactory

```
public class ConnectionFactory {
    public Connection getConnection() {
      try {
        Class.forName("com.mysql.jdbc.Driver");
        String urlBD="jdbc:mysql://localhost:3306/bd";
        return DriverManager.getConnection(urlBD, "root", "");
      } catch (SQLException e) {
         System.out.println("Exceção SQL");
      } catch(ClassNotFoundException e){
         System.out.println("Exceção Classe não encontrada");
      }
      return null;
    }
}
```

Exemplo 2

```
import dao.*;
public class Exemplo2{
    public static void main(String[] args)
                                        throws SQLException{
      Connection connection =
                    new ConnectionFactory().getConnection();
      System.out.println("Conexão aberta!");
      connection.close();
    }
}
```

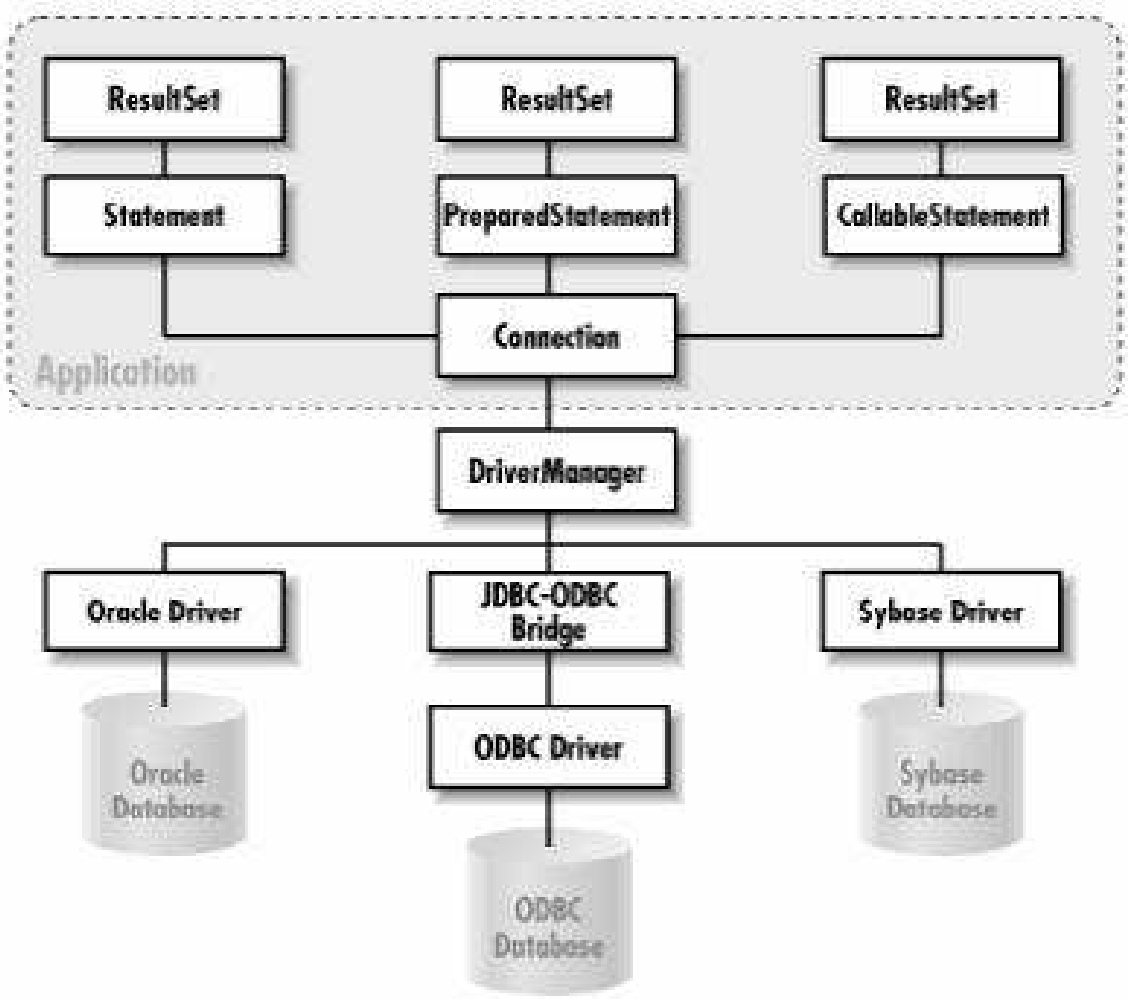
Exercícios

Exercícios

Faça os exercícios 7 e 8 da lista de exercícios

Manipulando Dados

Manipulando Dados



Statement

A maioria dos bancos de dados relacionais processa as consultas JDBC usando 4 passos:

1) Interpretar a consulta SQL
2) Compilar a consulta SQL
3) Planejar e otimizar o caminho de busca dos dados
4) Executar a consulta otimizada, buscando e retornando os dados

Um *Statement* sempre executa os passos acima para cada consulta SQL enviada para o banco

Statement

Statement não é seguro, pois é vulnerável à injeção de SQL

```
String sql = "select * from user where user='" + user +"' and
password ='" + password + "'";
//...
```

Se for informado para user => admin’ OR ‘1′=’1
O comando select ficaria: select * from user where username=’admin’ OR ‘1′=’1′ and password=”

O resultado desse comando retorna o usuário admin, ou o primeiro usuário cadastrado no banco

PreparedStatement

Essa classe pré-executa os passos (1) a (3)

Se uma mesma consulta for executada repetidas vezes mudando apenas os parâmetros de cada uma o uso dessa classe será mais rápido e com menos carga sobre o banco

Ajuda a evitar ataques de Injeção de SQL, pois usa métodos set() para definir os valores dinâmicos que fazem parte da consulta

CallableStatement

CallableStatement é usada para chamar de procedures e functions da base de dados

Com elas é possível colocar grande parte de regras de negócio no banco

As regras são definidas no BD, e se uma regra muda ou é inserida não é necessário fazer refatoração, recompilar e reimplantar o sistema

Selecionando Dados

Selecionando Dados da Tabela

```
Connection connection = new ConnectionFactory().getConnection();
String sql = “select * from pessoa";

Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery(sql);
while(rs.next()){
      int idPessoa = rs.getInt("idpessoa");
      String nome = rs.getString("nome");
      String endereco = rs.getString("endereco");
      System.out.println("Dados = "+idPessoa + " " + nome + " "
                                                + endereco);
}
```

ResultSet

Representa o conjunto dos resultados obtidos pela aplicação do SQL em uma tabela no banco de dados

Ele mantém o cursor apontando para a sua linha atual de dados, sendo que seu início fica posicionado na primeira linha

Médodos para acessar os diferentes tipos de dados: getInt, getString, getDouble, getFloat, getLong entre outros

Esses métodos permitem recuperar valores usando, por exemplo, o nome da coluna ou número da índice

ResultSet

Usando os índices do ResultSet

```
int idPessoa = rs.getInt(1);
String nome = rs.getString(2);
```

#	idpessoa 1	nome 2	endereco 3
1	1	Silvia	Rua X, 10
2	4	Eduardo	Rua Y, 30

Usando os nomes das colunas

```
int idPessoa = rs.getInt(“idPessoa”);
String nome = rs.getString(“nome”);
```

ResultSet

Para movimentar o cursor uma posição na tabela de dados, utiliza-se o método next():

next() - move o cursor um linha abaixo retornando true quando existe linha para o cursor ser posicionado; caso contrário retornará false, indicando que chegou ao fim dos registros

Exercícios

Exercícios

Faça os exercícios 9 a 11 da lista de exercícios

Inserindo Dados

Solução 1: Usar Statement

```
Connection connection = new ConnectionFactory().getConnection();
//Monta o SQL
String sql = "insert into pessoa (nome, endereco)" +
        " values ('" + nome + "', '" + endereco +  "')";
Statement s = connection.createStatement();
if(s.executeUpdate(sql)>0)System.out.println("Dados inseridos!");
else System.out.println("Erro ao inserir dados!");
```

Solução 2: Usar PreparedStatement

Essa classe pré-executa os passos (1) a (3)

Se uma mesma consulta for executada repetidas vezes mudando apenas os parâmetros de cada uma o uso dessa classe será mais rápido e com menos carga sobre o banco

Ajuda a evitar ataques de Injeção de SQL, pois usa métodos set() para definir os valores dinâmicos que fazem parte da consulta

Inserindo dados na Tabela

```
Connection connection = new ConnectionFactory().getConnection();

String sql = "insert into pessoa (nome,endereco) values(?,?)";

PreparedStatement stmt = connection.prepareStatement(sql);
// preenche os valores das ?
stmt.setString(1, "Silvia 2");
stmt.setString(2, "Rua X, 20");

// executa
stmt.execute();
stmt.close();
connection.close();
System.out.println("Dados Gravados!");
```

Exercícios

Exercícios

Faça o exercício 12 da lista de exercícios

Persistindo Objetos - BD

Salvando Objetos

Para começar a estruturar a solução usando camadas é necessário:

- Pegar os dados de um objeto e salvá-los no BD
- Ler os dados do banco de dados e armazená-los nos objetos

Criar Classe POJO

POJO = **Plain Old Java Objects** (significa “O Simples e Velho Objeto Java”)

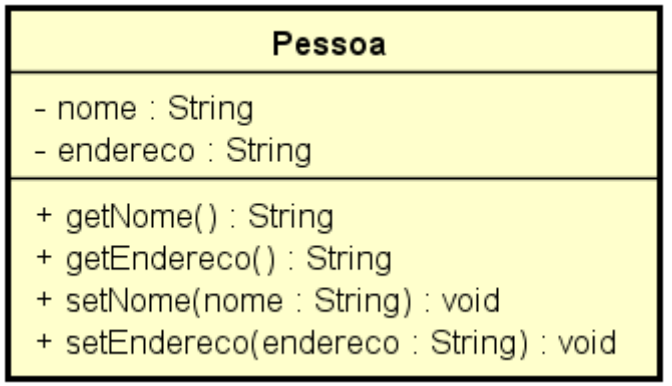
Objetivo é criar classes o mais simples possível

São objetos que não dependem da herança de interfaces ou classes de frameworks externos

Uma classe POJO segue definições rígidas de estrutura:

- Construtor sem argumentos;
- Propriedades com métodos get/set definidos

POJO: PESSOA




```
public class Pessoa{
    private String nome;
    private String endereco;
    public Pessoa(){}
    public Pessoa(String nome, String endereco){
            this.nome = nome;
            this.endereco = endereco;}
    public void setNome(String nome){ this.nome = nome; }
    public String getNome(){return nome;}
    //outros métodos get/set
}
```

Exemplo 6

```
PreparedStatement stmt = connection.prepareStatement("insert
into pessoa (nome,endereco) values (?,?)");

Pessoa p = new Pessoa("Silvia 3", "Rua X, 30");

stmt.setString(1, p.getNome());

stmt.setString(2, p.getEndereco());

stmt.execute();
```

Exercícios

Exercícios

Faça os exercícios 13 a 15 da lista de exercícios

Obtendo PK

Salvando Objetos e Obtendo sua PK

As classes Java para persistência de dados possibilitam que você salve dados e ao mesmo tempo retorne a chave primária gerada pela inserção

```
PreparedStatement stmt =
                          connection.prepareStatement(sql,
                  Statement.RETURN_GENERATED_KEYS);
stmt.setString(1, p.getNome());
stmt.setString(2, p.getEndereco());
stmt.execute();
ResultSet chaves = stmt.getGeneratedKeys();
if (chaves.next())  chavePrimaria= chaves.getInt(1);
System.out.println("PK = " + chavePrimaria);
```

Exercícios

Exercícios

Faça o exercício 16 da lista de exercícios