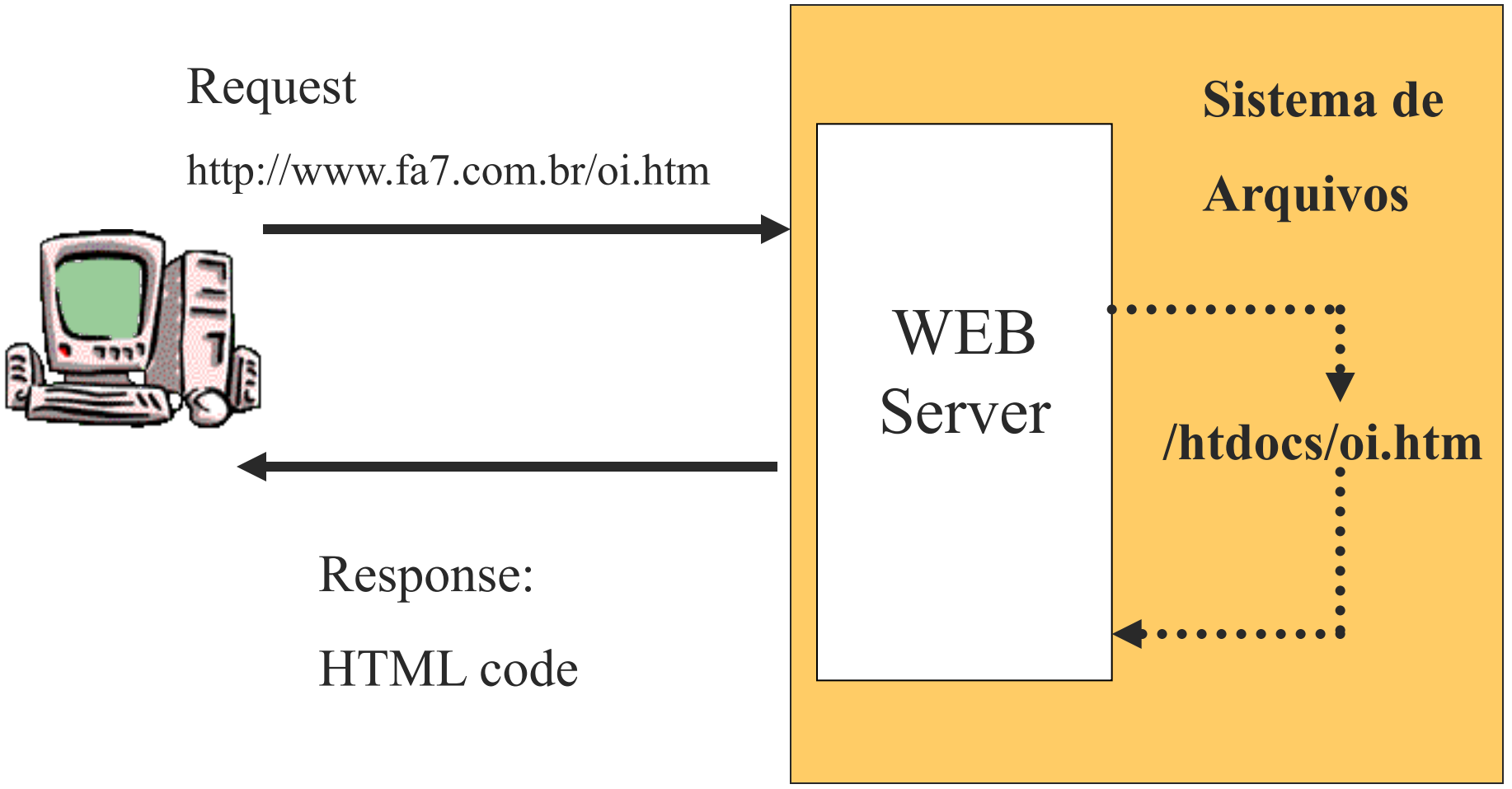
Aplicações Web com Servlets/JSP

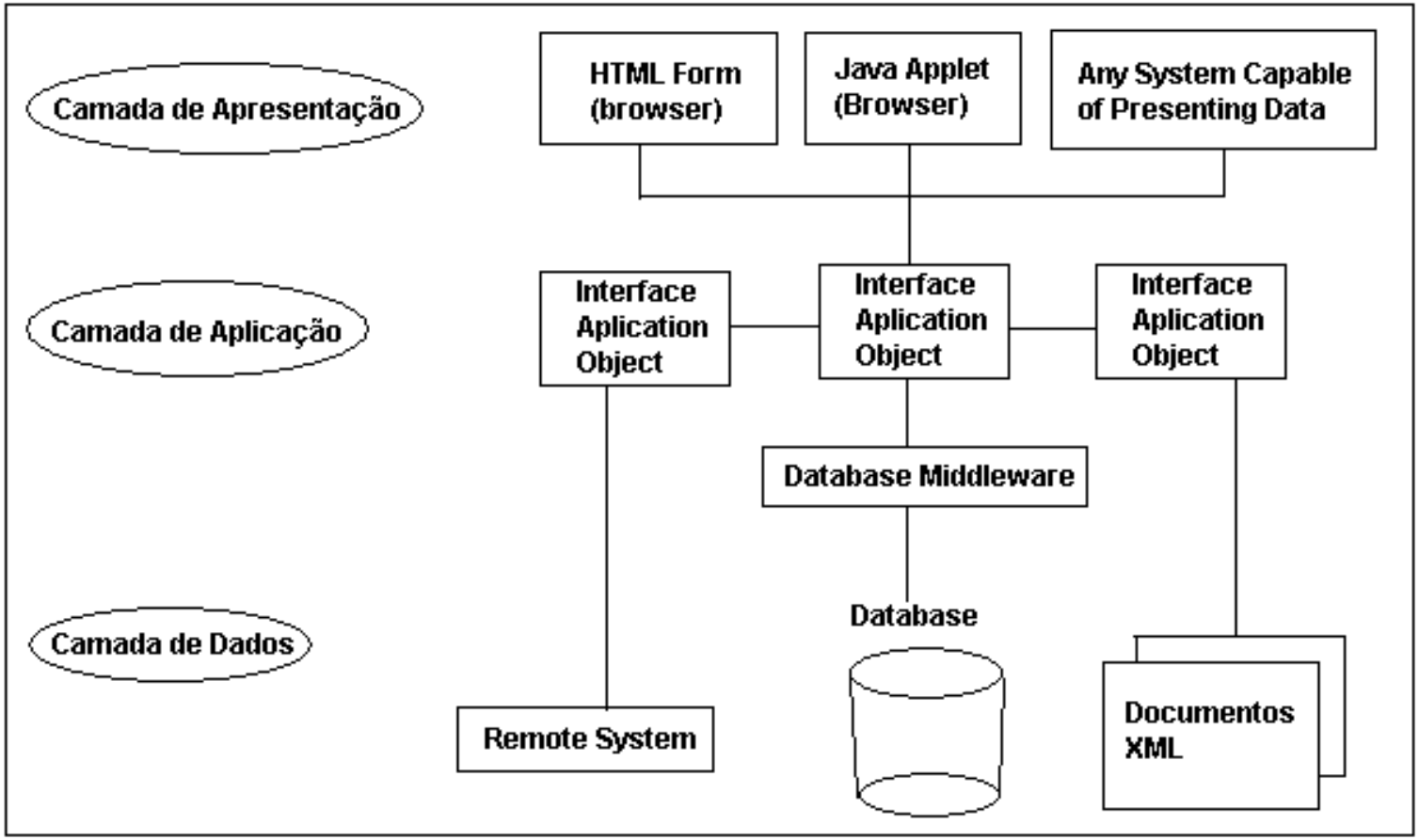
Prof. Serra
IFCE

Aplicações Web

Arquitetura da Web

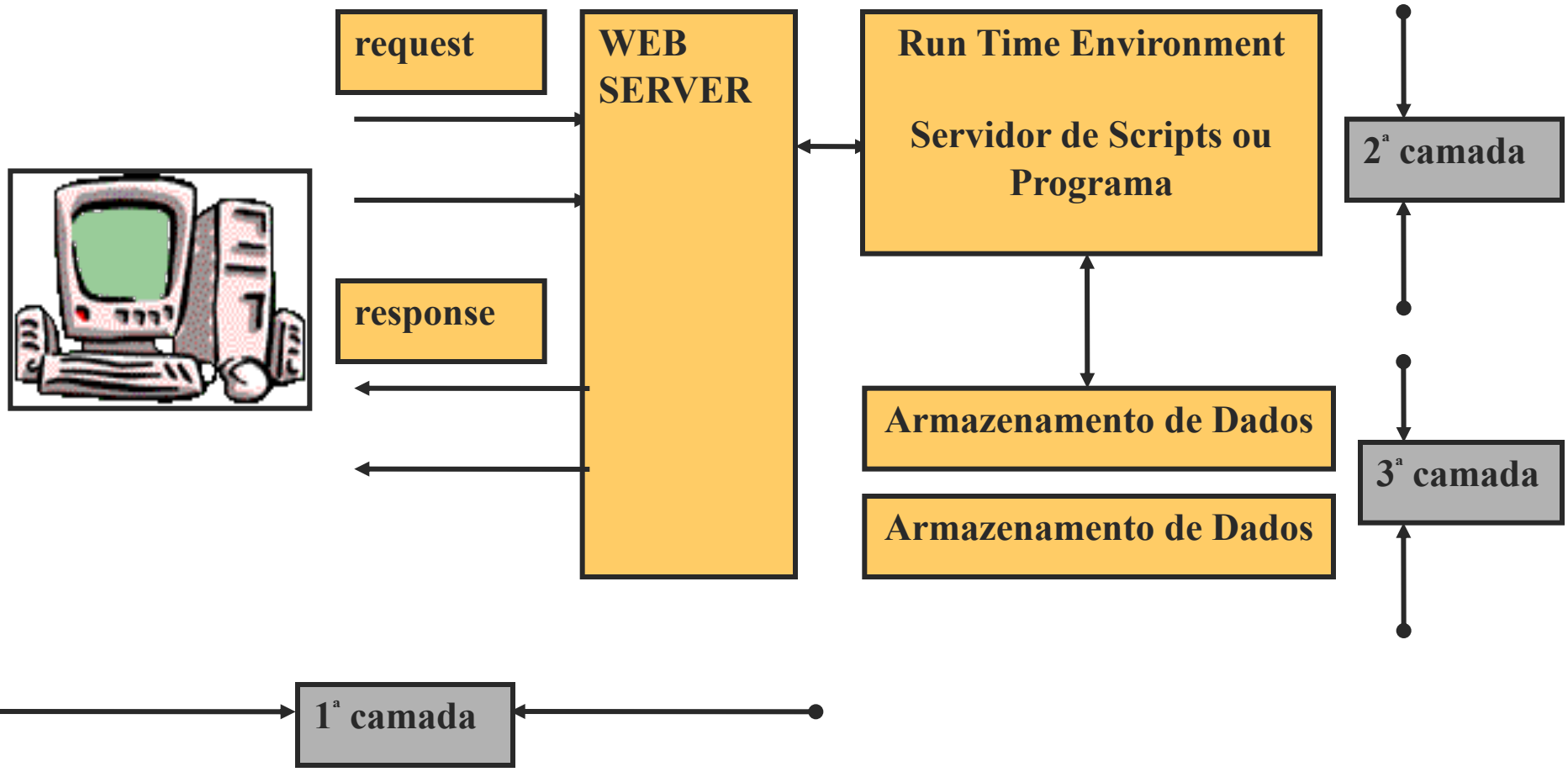


Aplicações n-camadas



Aplicações em n-camadas

Aplicaçoes Web - Arquitetura



Requisiçoes

- http://www.ifce.edu.br/cgi-bin/oi?name=serra

- HEAD - dados de cabeçalho;
- Get:
  - usa o cache;
  - tamanho da URL pequeno;
- Post:
  - permite um conjunto maior de dados;
  - envia sempre a requisição.

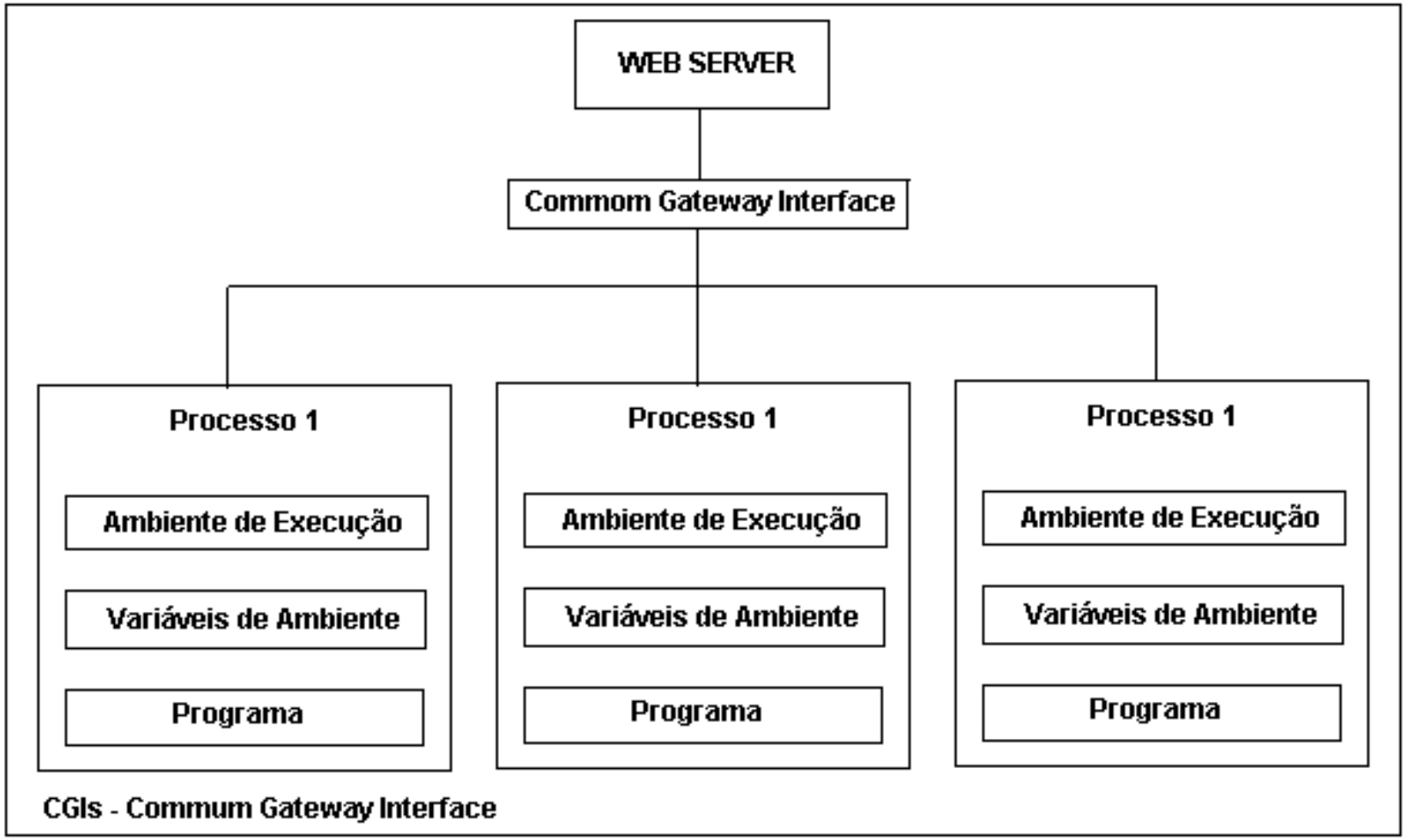
Tecnologias Lado Cliente

- HTML;
- Vbscript, JavaScript;
- DHTML;
- XML;
- Activex;
- Applets Java.

Tecnologias Lado Servidor

- Ágio para ser invocado pelo servidor;
- Ter que receber dados dos formulários;
- Deve ter como retornar o resultado ao servidor WEB.
- Exemplos:
  - CGIs
  - APIs (ISAPI,NSAPI);
  - ASP;
  - PHP;
  - Servlets;

GCIs


WEB SERVER
Commom Gateway Interface
Processo 1
Ambiente de Execução
Variáveis de Ambiente
Programa
Processo 1
Ambiente de Execução
Variáveis de Ambiente
Programa
Processo 1
Ambiente de Execução
Variáveis de Ambiente
Programa
CGIs - Commum Gateway Interface

APIs

- Bibliotecas compartilhadas;
- Carregadas no mesmo processo do WEB server na inicialização ou quando solicitadas;
- Desvantagens:
- não são multplataforma;
- podiam provocar um “crashing” no servidor WEB por estarem no mesmo processo.

ASP

- ASP utiliza HTML, Jscript e VBScript;
- Acessa qualquer componente no lado Servidor que tenha uma interface COM;
- Desvantagem:
- Não é multiplataforma.

Servlets Java e JSP

- Independência de plataforma;
- APIs da SUN para conexão com vários serviços;
- JAVA servlet:
  - um programa que serve requisições e respostas HTTP;
  - É multithead;
- JSP:
  - similar ao ASP da microsoft;
  - Contem HTML, java e JavaBean.

Porque a JAVA ?

- Independência de Plataforma;
- Eficiência;
- Acesso a APIs JAVA;
- Reutilização de código;
- Modularidade;

Revisão Linguagem Java

Linguagem JAVA

- orientada a objetos
- funciona em várias plataformas
- gera um bytecode
- executado pela JVM (Java Virtual Machine)

Aplicativos x Applets x Servlet

- **APLICATIVOS:** Programas que podem ser chamados de uma linha de comando.
- **APPLETS:** Programas embutidos em uma página na WEB.
- **SERVLETS**: Programas que se comunicam com um servidor web recebendo dados do browser, processando e devolvendo uma resposta.

Exemplo de um Programa JAVA

```
public class soma {
    public static void main (String args[]) {
        int a, b, c;
        a = 3;
        b = 2;
        c = a+b;
        System.out.println("resultado = "+c);
    }
}
```

Linguagem JAVA

Identificadores:

Podem ter qualquer tamanho, devem iniciar por uma letra, caractere de sublinhado ou cifrão e nas subsequentes podem conter dígitos.

Java E Unicode:

O Java utiliza o UNICODE que é um padrão que utiliza um conjunto de 16 bits que pode representar até 65.536.

Comentários:

// - Até o final da linha;
/* - inicia o comentário;
***/ - finaliza o comentário;**
/ - inicia um comentário que será utilizado pelo "javadoc"**

Obs: O javadoc – cria documentação sobre uma classe no formato HTML

Operadores

Operadores

++	- incremento;
--	- decremento;
!	- negação;
(nometipo)	- conversão de tipo;
*	- multiplicação;
/	- divisão;
>	- maior que;
<	- menor que;
==	- igual;
!=	- diferente;
&	- conjunção;
^	- ou exclusivo;
|	- ou inclusivo;
? :	- operador condicional;
=	- atribuição;

Tipos Primitivos de Dados

Boolean - true ou false;
int - -2147483648 a 2147483647
Long - -2^{63} a $2^{63}-1$
Byte - -128 a 127
Short - -32768 a 32767
Double - 17000..(307 zeros)...0
Float - -3.4E38 a 3.4E38 (6 a 7 dígitos de precisão)
Char - 0 a 65535

Tipos não Primitivos

String: é uma classe em java cujo objeto contém uma série de caracteres adjacentes;
Ex: “”, “ok \n”

Criação: String nome = “Serra”;

Concatenação: “jõao”+” e ”+”maria”

Instruções de Seleção

- **if (expr) instrução [else instrução]**

Obs: = significa atribuição enquanto == significa igualdade;
{ } delimita um bloco de instruções (o mesmo que begin e end)

em algumas situações pode ser utilizado: Expr ? instr1 : instr2

Exemplo:

```
if (opcao.equals("V1")) {
        query = query+" and to_char(data,'yyyy/mm/dd')='"+texto+"'";
}
if (opcao.equals("V2")) {
        query = query+" and local='"+texto+"' ";
}
```

Instruções de Seleção

```
switch (expr) {
        case const1: instr1; break;
        case const2: instr2; break;
        .
        dafault: instr; break;
```

Exemplo:

```
int operacao = Integer.parseInt(req.getParameter("Carga"));
switch (operacao) {
       case 0 : ListarBoletins(req,res);
                break;
       case 1 : CriarBoletim(req,res);
                break;
       case 2 : GravarBoletim(req,res);
                break;
}
```

Instruções de Repetição

For (inicial; teste, incremento) instrução;

Exemplo:

```
for (i=0; i<100; i++) {
        System.out.println(i)
}
```

obs: For (;;) – loop infinito;

Instruções de Repetição

While (expr) instrução

Exemplo:

```
int i = 0;
While i < 10 {
        System.out.print (“oi”);
        I=+1;
}
```

Instruções de Repetição (cont...)

do instrução while (expr);

Exemplo:

```
int i=0;
Do {
    System.out.print(“oi”)
    I=+1;
} While i<10;
```

Obs: pode se utilizar as estruturas de desvio de fluxo:

```
Continue;
Break;
```

Linguagem JAVA

Inicialização

recebem valores 0, nulo, false, “” etc na criação.

Classes Embutidas:

TIPO	**CLASSE**
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Exemplo:

```
Integer manga;
int i = 43;
Manga = new integer(i);
i = manga.intvalue();
```

Linguagem JAVA

Pacotes

Um pacote é um conjunto de classes que estão associadas e cujos arquivos .class ficam armazenados em um mesmo diretório.

CLASSPATH

Indica a localização das classes para o compilador.

Exemplo:

SET CLASPATH = C:\sun ; c:\projeto; c:\projeto\teste\fonte

Linguagem JAVA

Import

```
import java.net.*;
import java.math.*;
import java.io.*;
import java.util.*;
```

Indica que pacotes serão utilizados no programa.

Obs: se não for definido é necessário utilizar a referência completa da classe:

Exemplo:

```
Class torta {
        Java.util.date faitaem;
        Duble peso;
```

Classes Java (.zip, .tar)

Arquivos de Classe “.jar”

As classes podem vir agrupadas em um arquivo compactado (.jar).

Neste caso a variável SET CLASSPATH deve ser:

UNIX : setenv CLASSPATH /java/lib/classes.zip:.:/meucodigo

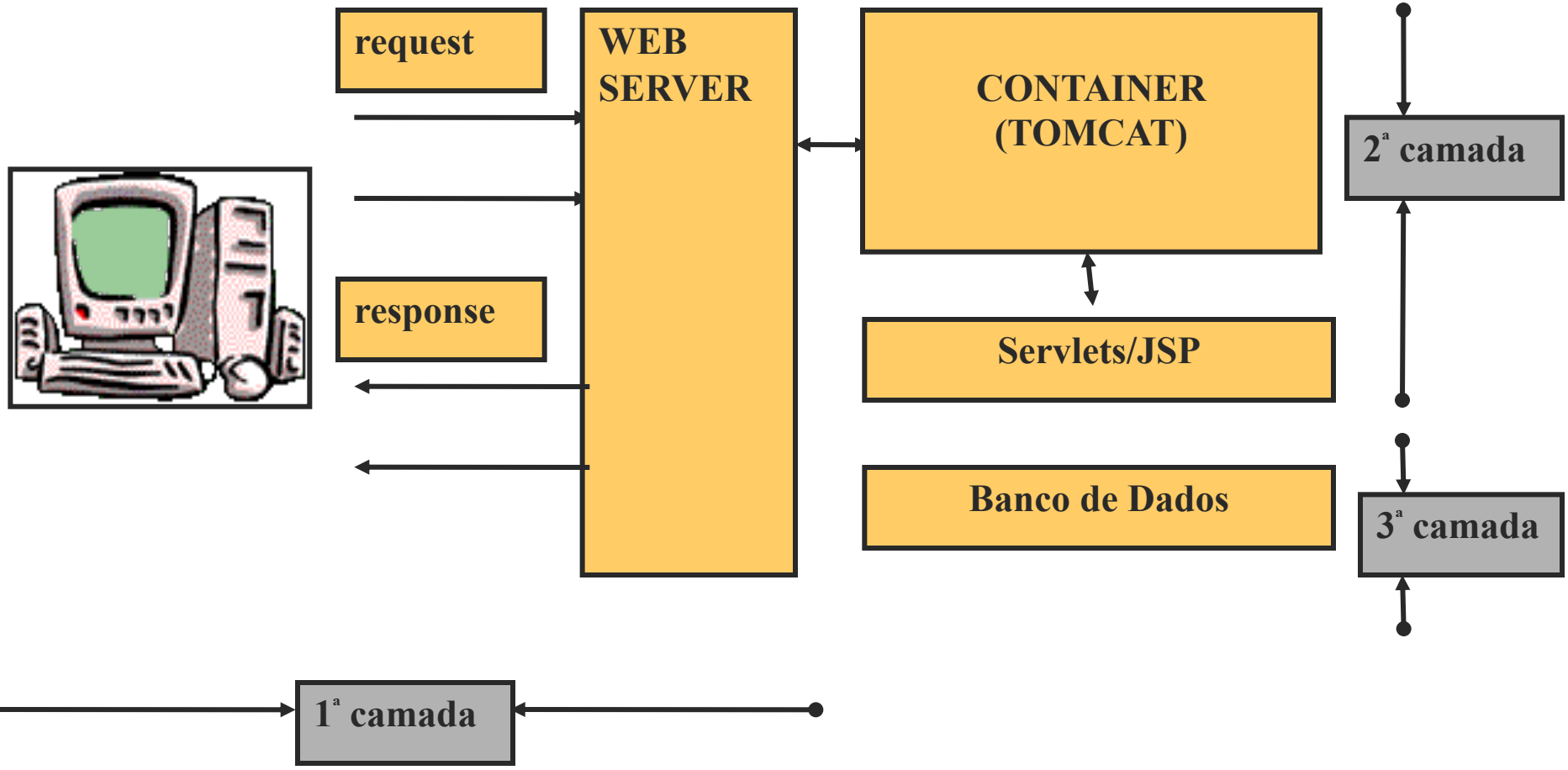
WINDOWS: set CLASSPATH=c:\java\lib\dt.jar; c:\java\lib\tolls.jar; c:\meucodigo

Servlets Java

Servlets

- Servlet API
  - Contem as classes e interfaces necessárias ao desenvolvimento de servlets:
  - Classe: javax.servlet.http.HttpServlet define o cliclo de vida de um servlet;
- www.java.sun.com/products/servlets

Servlets/JSP - Arquitetura



Servlet Container

Cria Instância do Servlet

Chama o Método Init

Instancia o método service() quando recebe uma requisição

Chama o Método destroy

Remove a Instância do Servlet

Exemplo (Hello World)

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class HelloWorld extends HttpServlet {
   public void doGet(HttpServletRequest req, HttpServletResponse res)
         throws ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      out.println("<HTML>");
      out.println("<HEAD>");
      out.println("<TITLE>Hello World Sample Servlet</TITLE>");
      out.println("</HEAD>");
      out.println("<BODY>");
      out.println("<CENTER><H1>Hello World!</H1></CENTER>");
      out.println("</BODY>");
      out.println("</HTML>");
      out.close();
   }
}
```

Resposta do Servlet

Passando Parâmetros

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class HelloYou extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException  {
    String nome = req.getParameter("nome");
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<HTML>");
    out.println("<HEAD>");
    out.println("<TITLE>Hello World Sample Servlet</TITLE>");
    out.println("</HEAD>");
    out.println("<BODY>");
    out.println("<CENTER><H1>Hello "+nome+"</H1></CENTER>");
    out.println("</BODY>");
    out.println("</HTML>");
    out.close();
  }
}
```

Resposta Usando Parâmetros

Parâmetros Através do Form

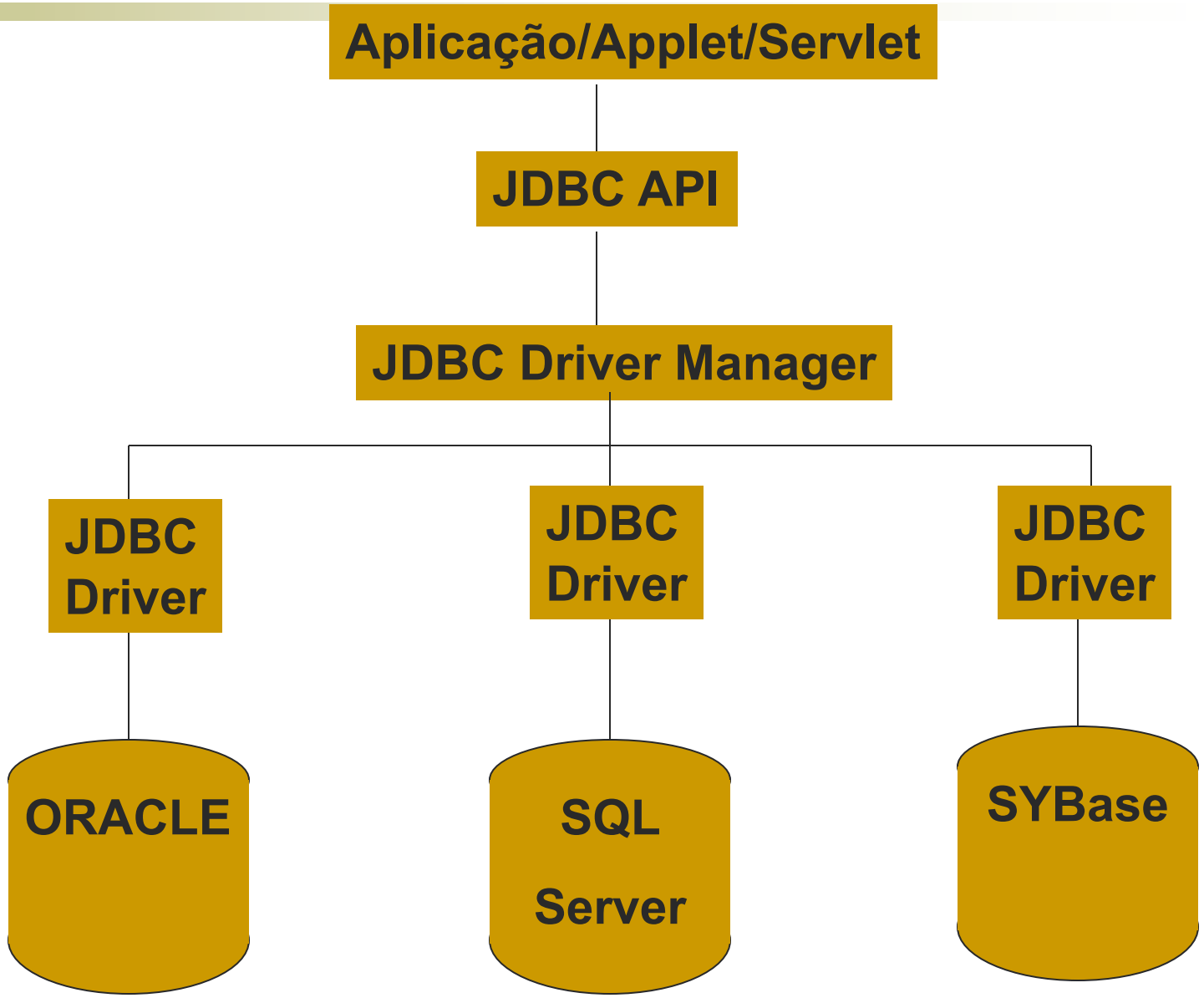
```
<html>
<head>
  <title> Exemplo de Um Formulário </title>
</head>
<body>
  <form method="GET" action="http://localhost:8080/servlet/HelloYou">
    Digite seu nome:<input type="text" name="nome">
    <input type="submit" value="Clique aqui">
    <input type="reset"  value="Limpe o campo">
  </form>
</body>
```

Integrando Servlets com Banco de Dados

Servlets e Bancos de Dados

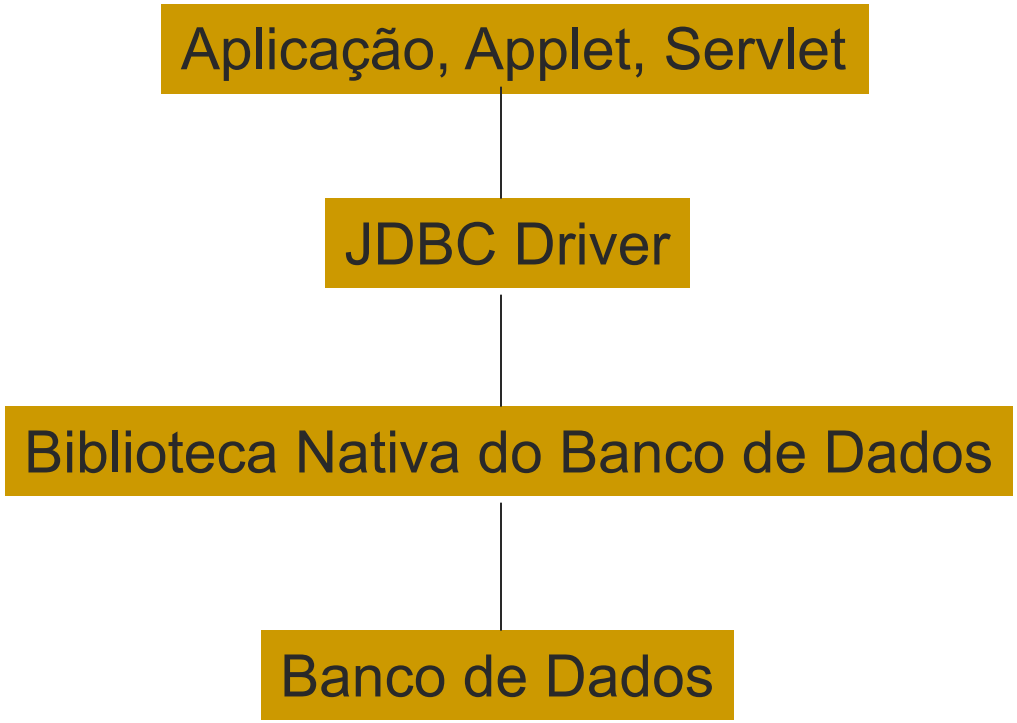
- JDBC é uma especificação de API que define:
  - como interagir com bancos de dados através de applets, aplicações e servlets;
  - como utilizar os drivers JDBC;
  - como escrever JDBC drives;
- Permite interoperabilidade

A ponte JDBC-ODBC



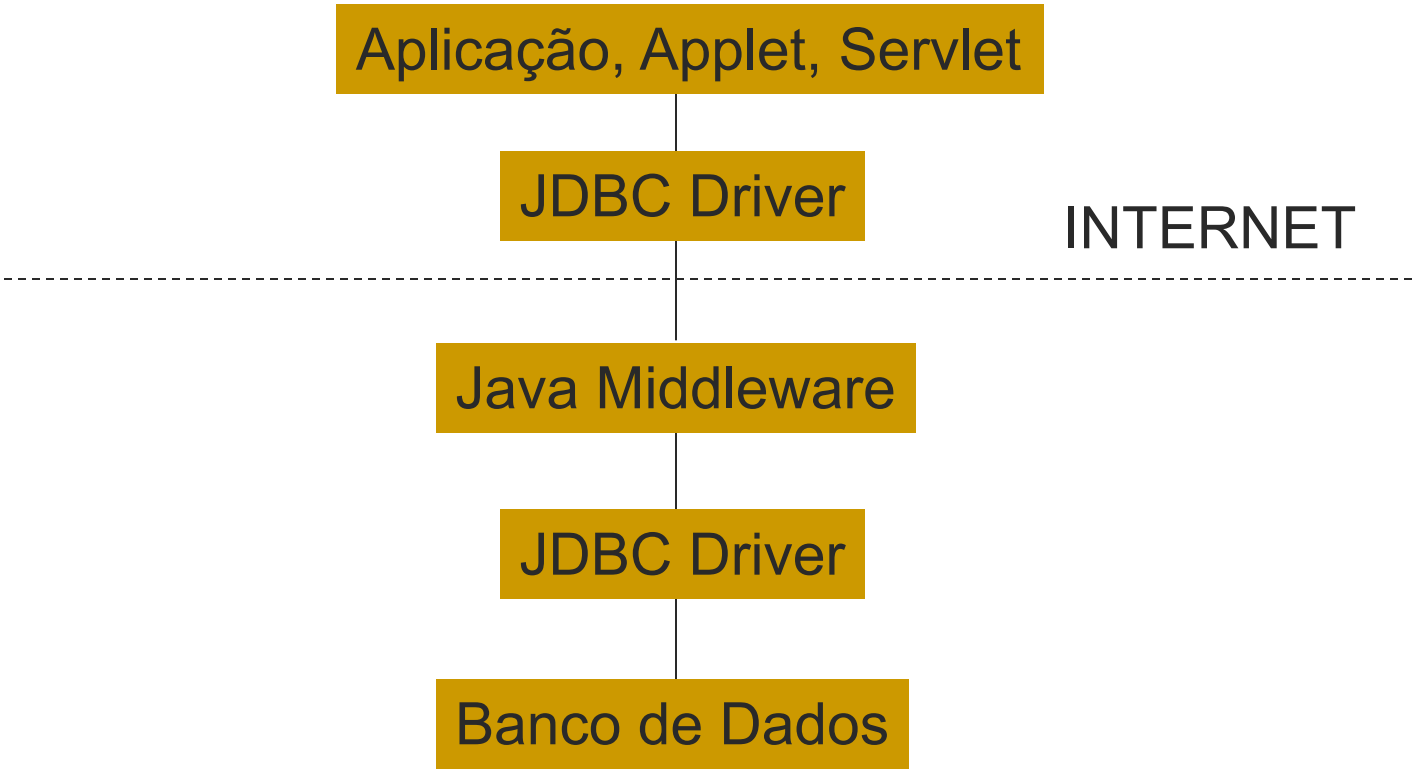
Acesso Nativo (API java)

- Faz uso de bibliotecas fornecidas pelo fabricante para acesso direto ao Banco de Dado;



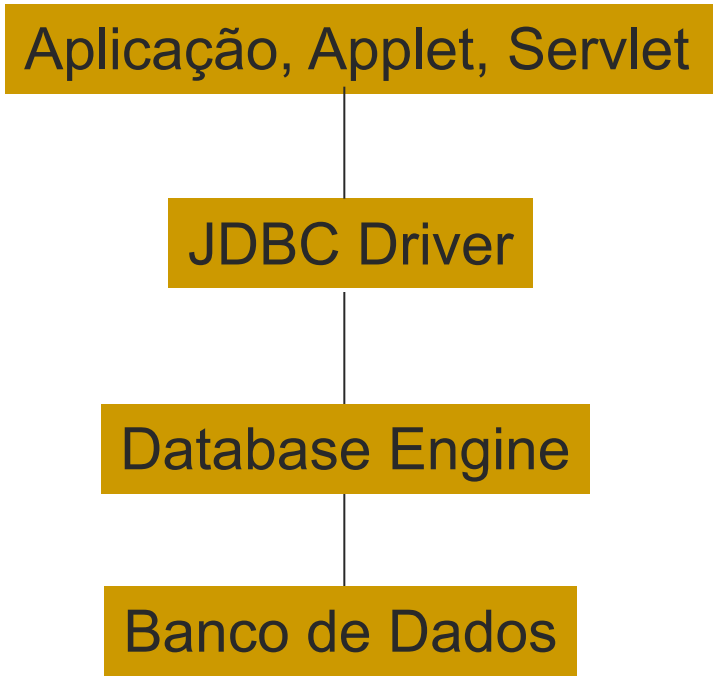
Java para Protocolo Proprietário

- São feitos puramente em Java e comunicam com outras camadas via um determinado protocolo criado pelo fabricante.



Protocolo de Banco de Dados Nativo

- São drives escritos puramente em Java que se comunicam diretamente com a engenharia do banco de dados via seu protocolo nativo;



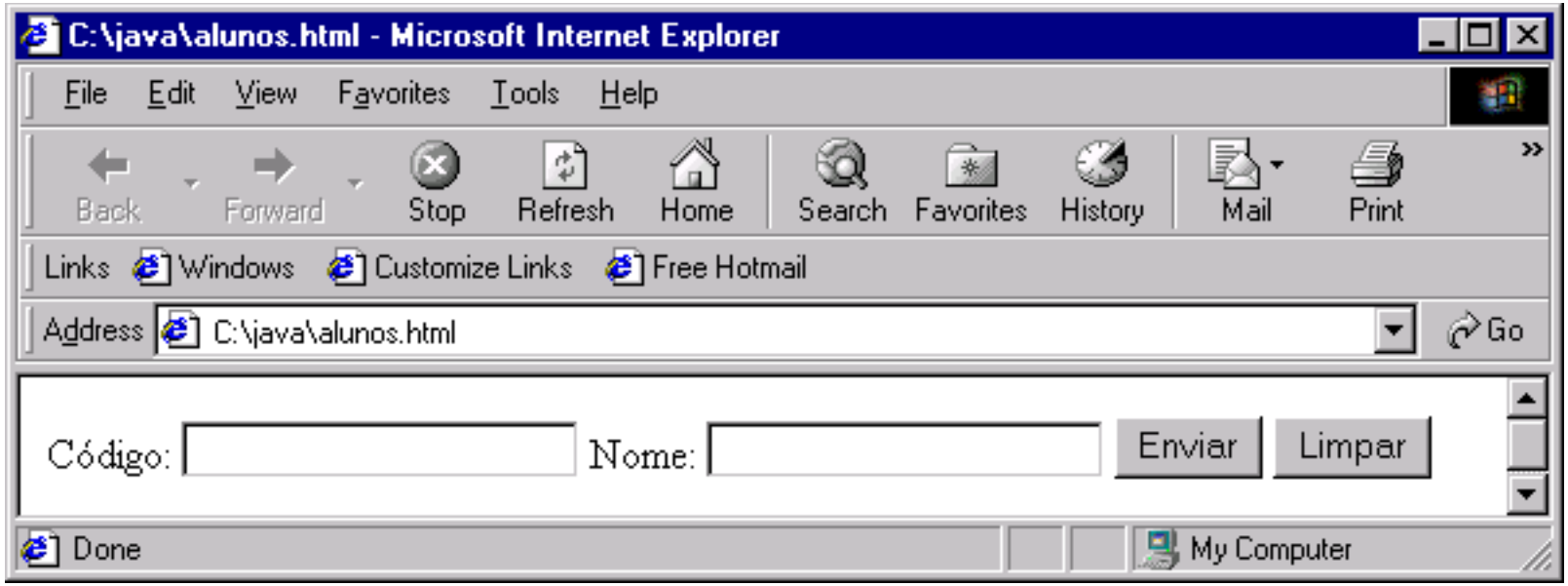
Estabelecendo a Conexão

- Registrar a o driver
  - Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
- Connection
  - con = DriverManager.getConnection ("jdbc:odbc:test", “usuario", “senha");

Acessando o Banco de Dados

- Criando o objeto “Statement”
  - Statement stmt = con.createStantement();
- Executando uma Consulta:
  - ResultSet rs = stmt.executeQuery(“SELECT * FROM alunos”);
- Atualizando Dados:
  - stmt.executeUpdate("insert into alunos values (01,’ Serra’) ”);

Enviando os Dados



- <HTML> <HEAD> Alunos </HEAD>
- <BODY TEXT="#000000" BGCOLOR="#ffffff">
- <FORM ACTION="http:\\localhost:8080\servlet\alunos">
- <P> Codigo: <INPUT TYPE="TEXT" NAME="codigo">
- Nome: <INPUT TYPE="TEXT" NAME="nome">
- <INPUT TYPE="SUBMIT" VALUE="Enviar" >
- <INPUT TYPE="RESET" VALUE="Limpar" >
- </P> </FORM> </BODY> </HTML>

Incluindo no Banco de Dados

```
public class alunos extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException   {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        }
        catch(java.lang.ClassNotFoundException e) {
            out.println("<P>Erro no registro da classe: ");
            out.println("<P>"+e.getMessage());
        }
        try {
            Connection con=DriverManager.getConnection("jdbc:odbc:alunos","", "");
            Statement stmt=con.createStatement();
            String nome=req.getParameter("nome");
            String cod=req.getParameter("codigo");
            stmt.executeUpdate("insert into alunos values("+cod+", '"+nome+"')");
                        .
                        .
                        .
```

Consultando o Banco de Dados

```
                    .
                    .
                    .
                    ResultSet rs = stmt.executeQuery("select * from alunos");
                    out.println("<P>Códigos e Alunos Cadastrados");
                    while (rs.next()) {
                            String s = rs.getString("nome");
                            int n = rs.getInt("codigo");
                            out.println("<P> Codigo: "+n + " - Nome: " + s);
                    }
                    stmt.close();
                    con.close();
            } catch(SQLException ex) {
                    out.println("<P>SQLException: " + ex.getMessage());
            }
            out.println("<P> Obrigado por participar");
            out.close();
    }
}
```

JSP = html + Java

```
<%@ page import="java.sql.*" %>
<html>
<body>
<%
Connection con = null;
Statement stmt = null;
try {
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
}
catch(java.lang.ClassNotFoundException e) {
    String mensagem = e.getMessage();
    %>
        <P> ClassNotFoundException: <%= mensagem %>
    <%
}
try {
    con = DriverManager.getConnection("jdbc:odbc:alunos","",
        "");
    stmt = con.createStatement();
    String nome = request.getParameter("nome");
    String cod = request.getParameter("codigo");


    stmt.executeUpdate("insert into alunos values("+cod+",
        '"+nome+"')");
```

```
 ResultSet rs = stmt.executeQuery("select * from alunos");
    out.println("<P>Códigos e Alunos Cadastrados");
    while (rs.next()) {
                String s = rs.getString("nome");
                int n = rs.getInt("codigo");
                %><P> Codigo: <%= n %> - Nome: <%=  s%>
<%  }
    stmt.close();
    con.close();
} catch(SQLException ex) {
    out.println("<P>SQLException: " + ex.getMessage());
}
%><P> Obrigado por participar <%
out.close();
%>
</body>
</html>
```

FIM

Contato
serra@cefetce.br