A Evolução da Plataforma J2EE

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Java
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Java
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

O que é a plataforma J2EE?

- Estende a **linguagem Java** através de **padrões** para aplicações distribuídas através de um modelo baseado em **componentes**.

Linguagem Java

- A linguagem Java é Orientada a Objetos
- A linguagem é totalmente portável
- A linguagem implementa desalocação automática de memória

O que é a plataforma J2EE?

- Estende a **linguagem Java** através de **padrões** para aplicações distribuídas através de um modelo baseado em **componentes**.

Padronização

- Java Community Process. www.jcp.org
- Participantes: Accenture, Adobe, **BEA**, **Borland**, British Telecom, Bull, Cisco, Compaq, DOD, EDS, Ericsson, **Fujitsu**, **HP**, **Hitachi**, **IBM**, **Macromedia**, Mitsubishi, Motorola, **NEC**, Nokia, Novell, NTT, **Oracle**, Panasonic, Philips, Rational, SAP, Sharp, Siemens, Sony, **Sun**, **Sybase**, Telefonica, Texas Instruments, Unisys, Verisign, Vignette, Vodafone, Xerox, Yamaha,...

O que é a plataforma J2EE?

- Estende a **linguagem Java** através de **padrões** para aplicações distribuídas através de um modelo baseado em **componentes**.

Componentes

- Componentes são:
  - Partes de programa independentes, dinamicamente ligáveis, acessáveis via interfaces recuperáveis em *runtime*, com dependências de ambiente explicitadas.
  - Independentes = análise, projeto, implementação, testes, **deploy** e **execução** e manutenção independentes.

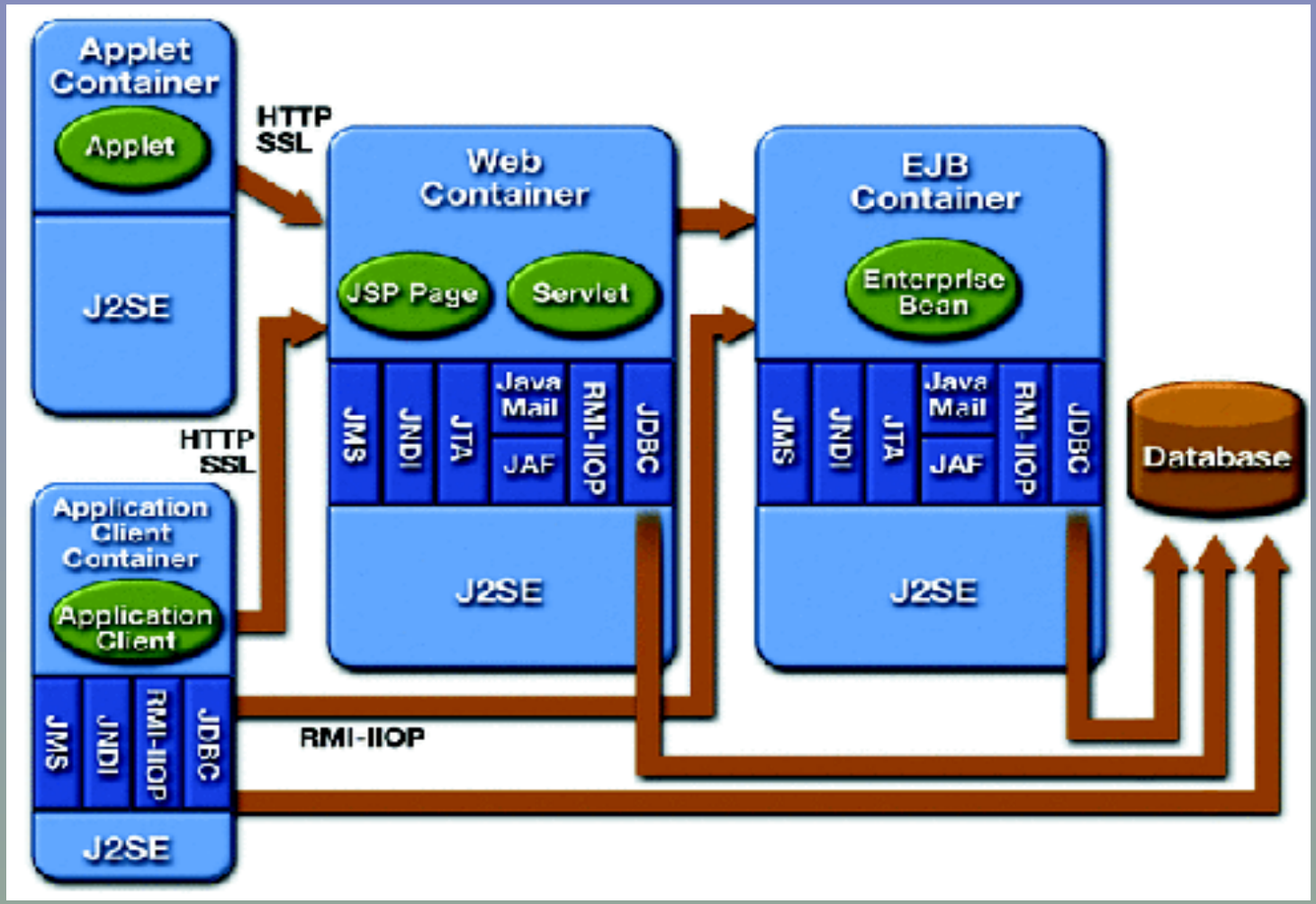
Componentes

- Formas de antender a nova realidade:
  - Usar de novas tecnologias de *deploy* de *software*
  - **Encorajar a reutilização**
  - **Flexibilizar a atualização e substituição de partes de sistemas - manutenção evolutiva**
  - Organizar as práticas organizacionais, adaptando-as às mudanças tecnológicas e mercadológicas

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Java
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

O que são Servidores de Aplicação ?



Certificação de Compatibilidade

- Em Abril 2004:
  - 18 servidores de aplicação certificados em J2EE1.2
  - 20 servidores de aplicação certificados em J2EE1.3
  - 5 servidores de aplicação certificados em J2EE1.4

Servidores de Aplicação J2EE 1.4

- **IBM WAS 6.0 (versão 5.0 homologada em 10 plataformas)**
- **Oracle Application Server Containers for J2EE (10.0.3) - Developer Preview**
- **Sun ONE Application Server 7**
- **Tmax Soft JEUS 5.0**
- **Trifork T4 Application Server**

Ambientes de Desenvolvimento

- **Eclipse 3 (open source)**
- **IBM WSAD 5.1 (Eclipse 2)**
- **Borland JBuilder 9**
- **NetBeans 3.6 (open source)**
- **Sun ONE Studio 5 (NetBeans)**
- **Sun Java Studio Creator (NetBeans)**
- **Oracle9i Jdeveloper**

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Java
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

Linguagem Java

- A linguagem Java é Orientada a Objetos
- A linguagem é totalmente portável
- A linguagem implementa desalocação automática de memória

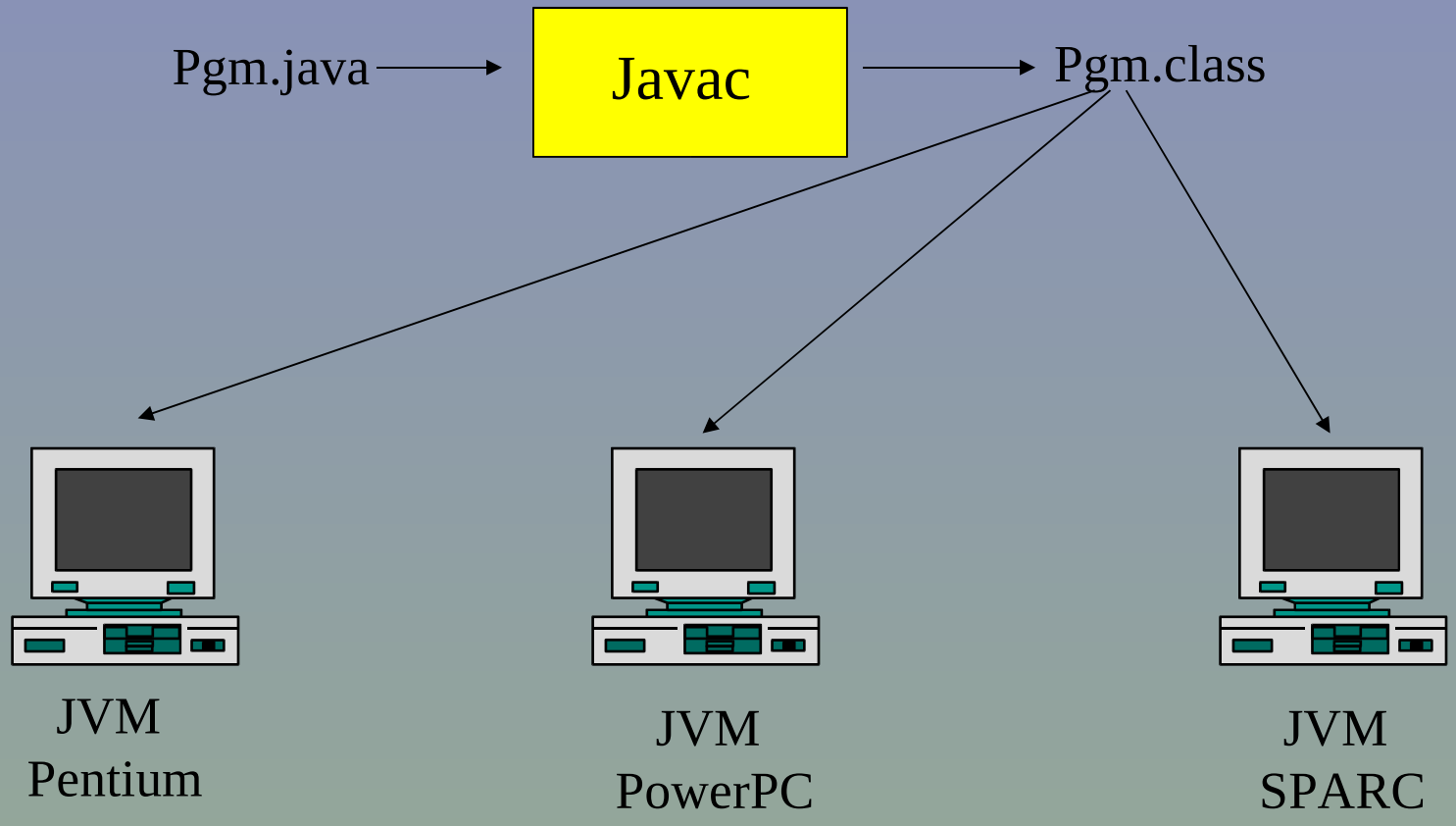
Linguagem Java

- A linguagem Java é Orientada a Objetos

Influenciada diretamente por C++ e Eiffel, a linguagem segue a grande tendência das linguagens de programação nas décadas de 80 e 90. Neste período, linguagens como Pascal, Ada, Lisp e Cobol ganharam versões Orientadas a Objetos.

Linguagem Java

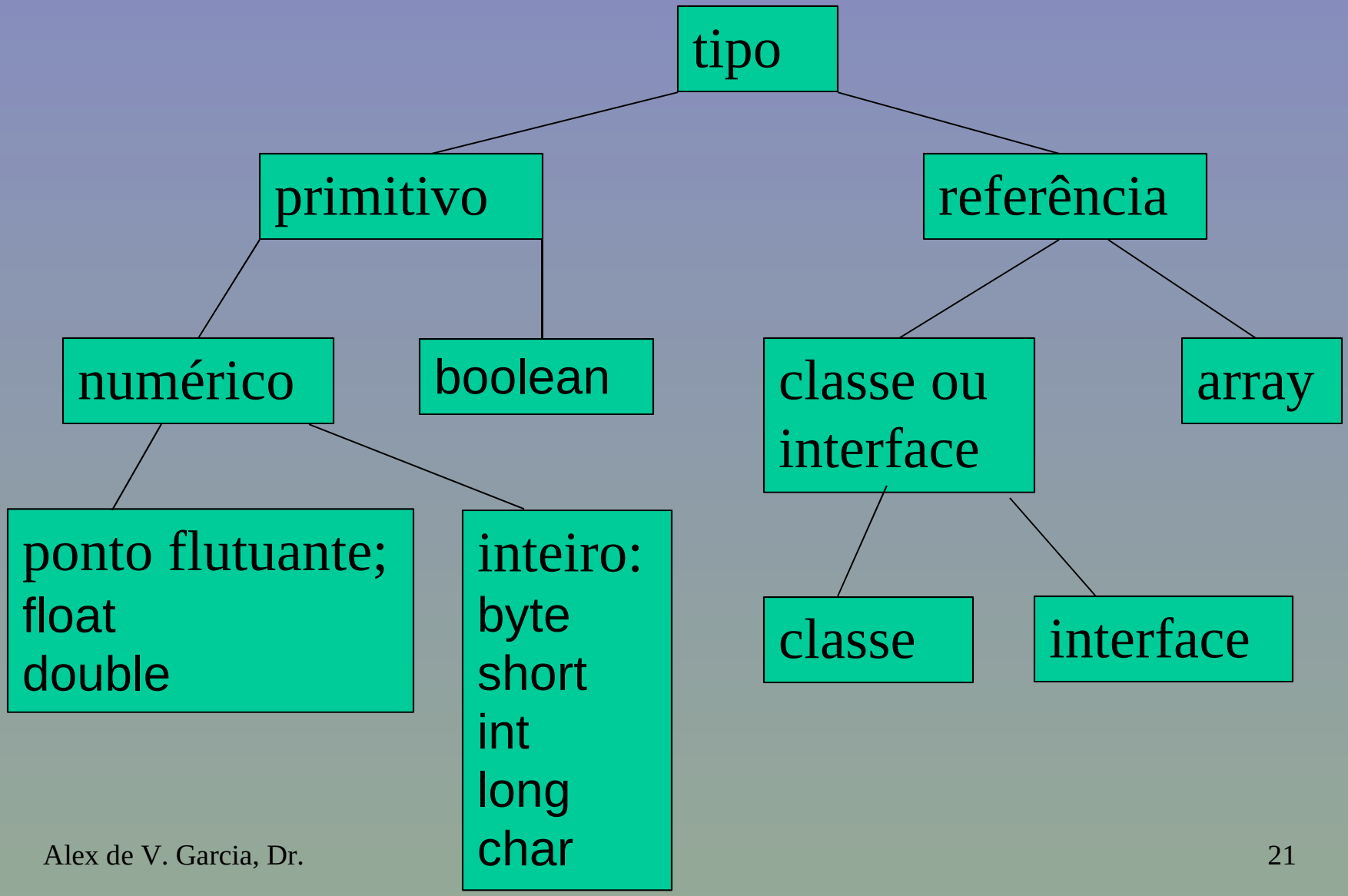
- A linguagem é totalmente portável



Linguagem Java

- A linguagem implementa desalocação automática de memória (“coleta de lixo”), e não permite a desalocação manual.
  - Produtividade do programador 
  - Testes 
  - Bugs 
  - Qualidade 

Java - Tipos de Dados



Exemplo de Programa

```
public class AloMundo2 {
        public static void main(String args[]) {
                // args é um array de Strings
                String s = " Alô,  ”;
                s = s + args[0] + “!”;
                System.out.println(s);
        }
}
```

Linguagem Java

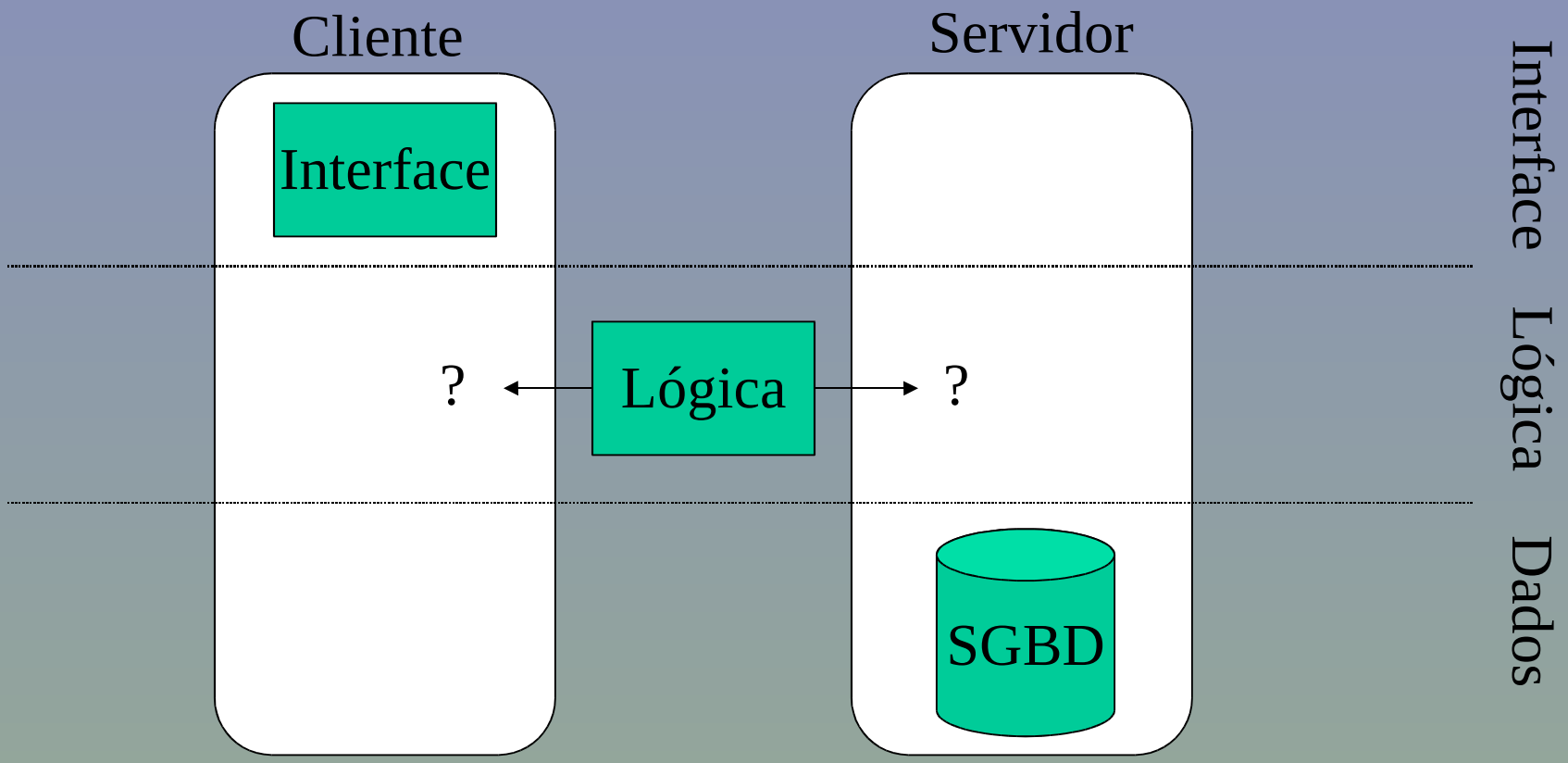
- Conclusão:
  - Java é uma das linguagens mais bem projetadas desde Ada.
  - Java não apresenta performance comparável a Fortran ou C.
  - As decisões de projeto da linguagem são ótimas para rodar em um servidor Web.
    - Portabilidade entre diferentes sistemas operacionais.
    - Facilidades de *multithreading* incluídas na linguagem.
    - Baixo custo de desenvolvimento (alta produtividade).

Sumário

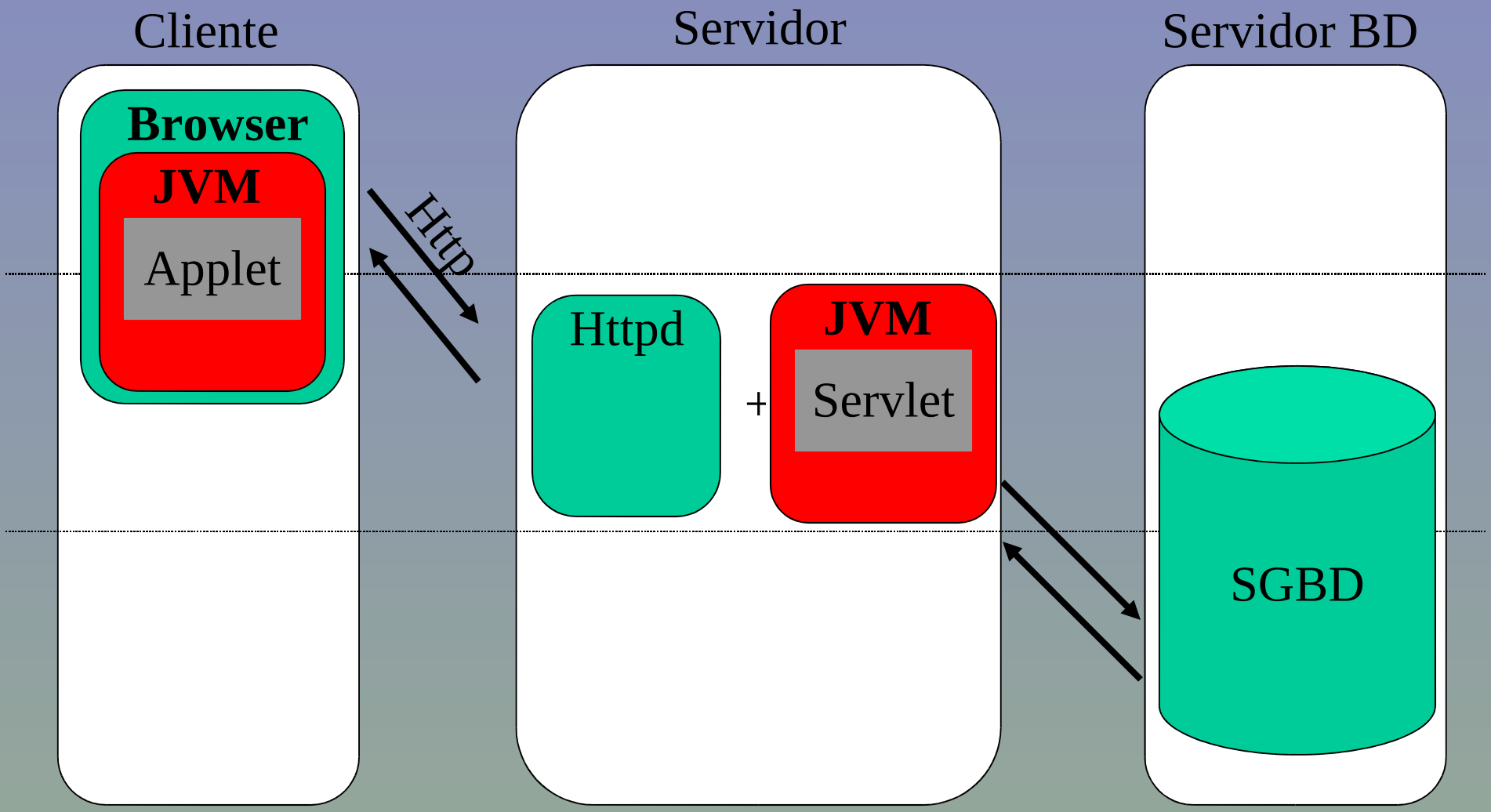
- Plataforma J2EE
- Servidores de Aplicação
- Java
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

Aplicações Web

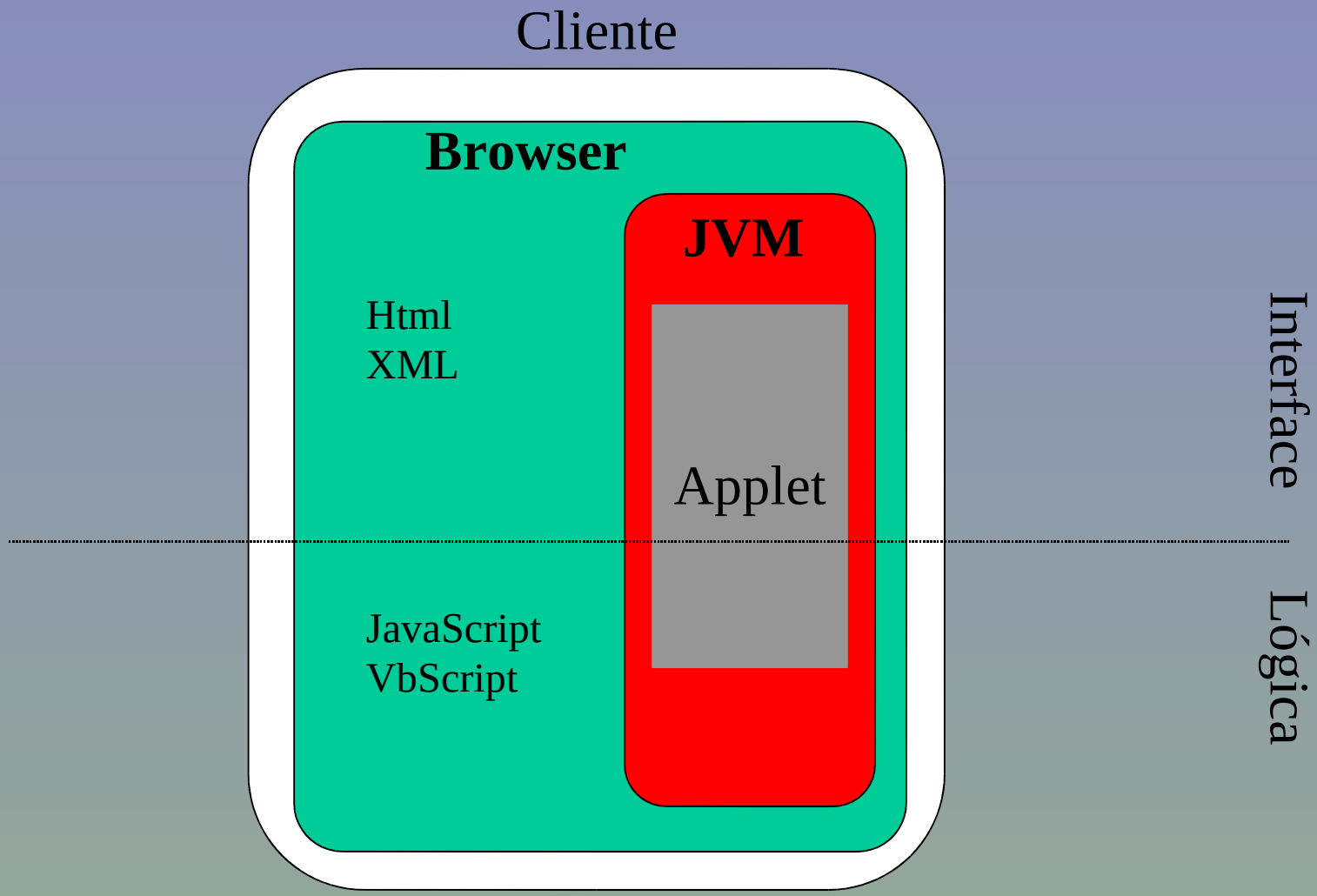
- São aplicações cliente-servidor



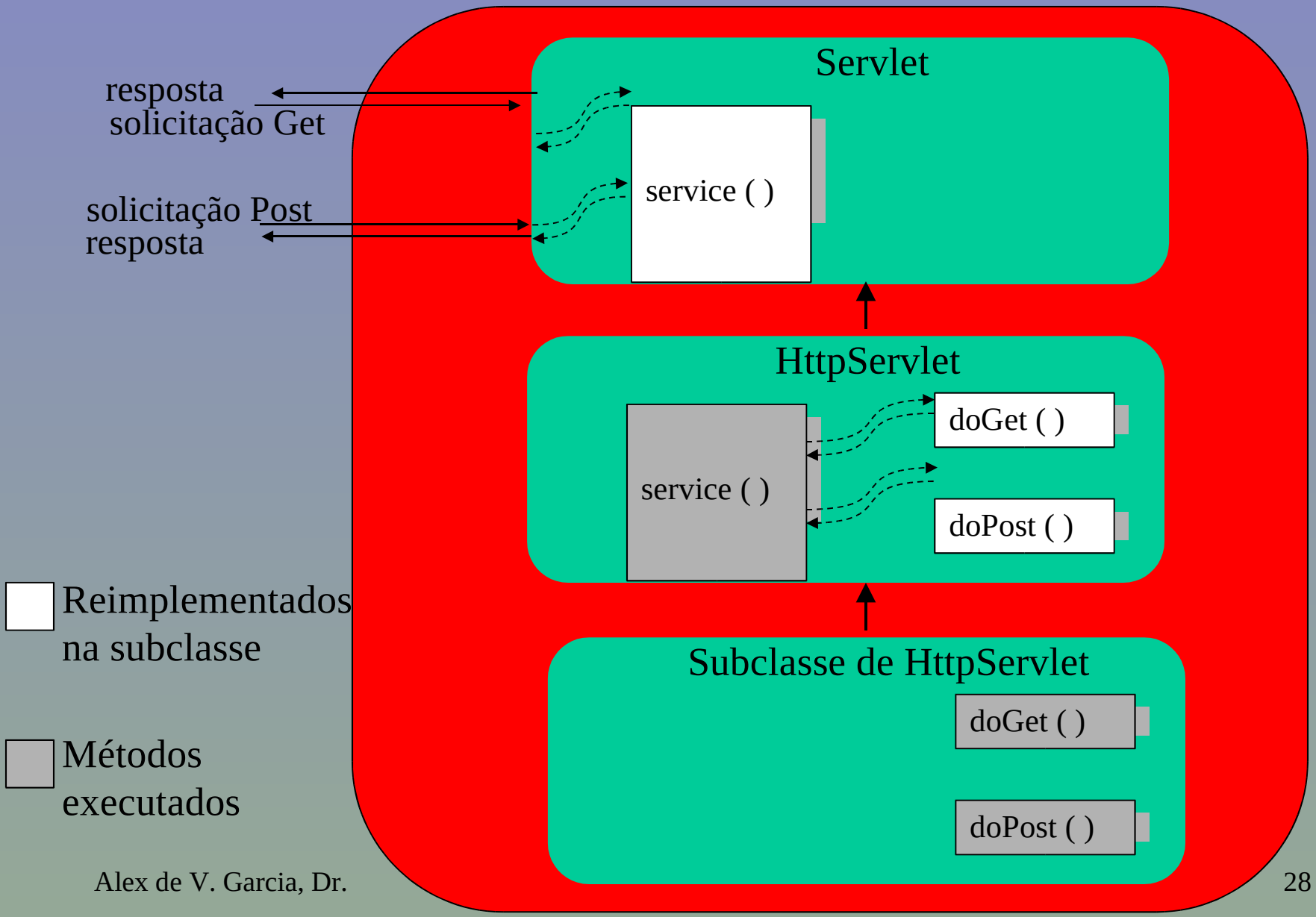
Aplicações Web



Java no Cliente



Java no Servidor



Exemplo de Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class AloMundo extends HttpServlet {

    public void doGet(HttpServletRequest req, HttpServletResponse res)
                                throws ServletException, IOException {

        res.setContentType("text/html");
        PrintWriter out = res.getWriter();

        out.println("<HTML><BODY>");
        out.println("<BIG> Alô Mundo </BIG>");
        out.println("</BODY></HTML>");
    }
}
```

Tratando o Request

```
<HTML>
<BODY>
<FORM METHOD=GET ACTION="/Olah">
Qual é o seu nome?
<INPUT TYPE = TEXT NAME="nome"><P>
<INPUT TYPE = SUBMIT>
</FORM>
</BODY>
</HTML>
```

- Cada campo em um formulário HTML representa um parâmetro para um request do tipo Get ou Post, de acordo com o especificado no atributo "METHOD".

Tratando o Request

```
public class Olah extends HttpServlet {

  public void doGet(HttpServletRequest req, HttpServletResponse res)
                              throws ServletException, IOException {
      res.setContentType("text/html");
      PrintWriter out = res.getWriter();
      String nome = req.getParameter("nome");
      out.println("<HTML>");
      out.println("<BODY>");
      out.println(”Olá, " + nome);
      out.println("</BODY></HTML>");
  }
}
```

Java no Servidor

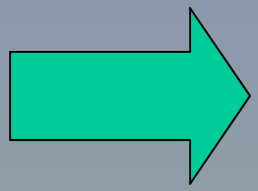
- Da mesma forma que existe lógica no cliente, existe apresentação no servidor.

- Como separar a apresentação da lógica?
  - Facilitar edição para mudança de layout
  - Facilitar mudança de padrão de saída (ex: de HTML para WML)
  - Facilitar manutenção da interface

JSP

JSP

```
---- texto html ----
---- texto html ----
---- texto html ----
<% código java %>
---- texto html ----
<% código java %>
---- texto html ----
---- texto html ----
```



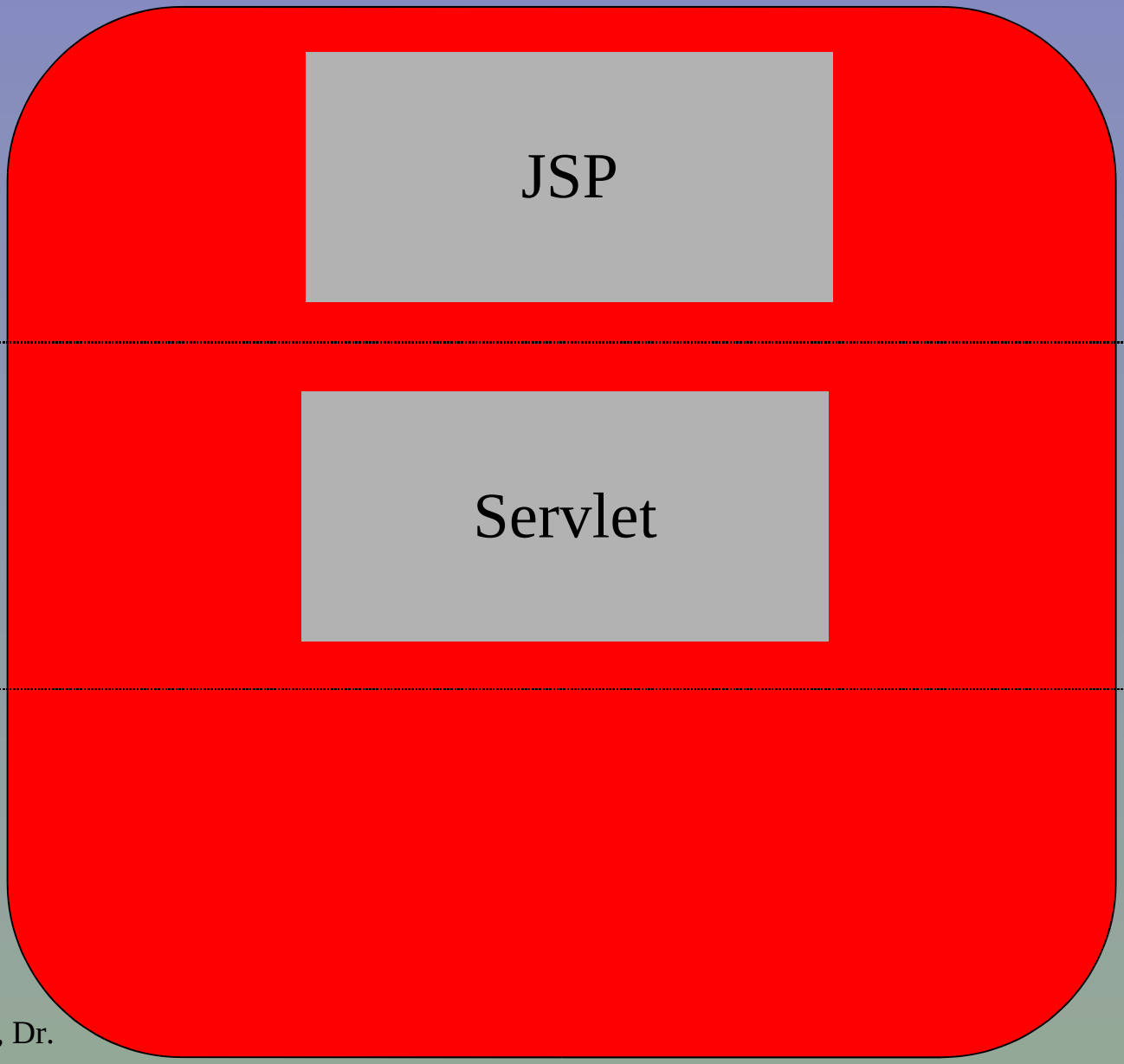
Servlet

```
out.print(texto html)
out.print(texto html)
out.print(texto html)
--   código java --
out.print(texto html)
--   código java --
out.print(texto html)
out.print(texto html)
```

Exemplo de JSP

```
<HTML>
<HEAD>
<TITLE>Olá</TITLE>
<BODY>
 Olá, <%= request.getParameter(”nome”) %>,
  a hora é <%= new java.util.Date() %>
<BR>
</BODY>
</HTML>
```

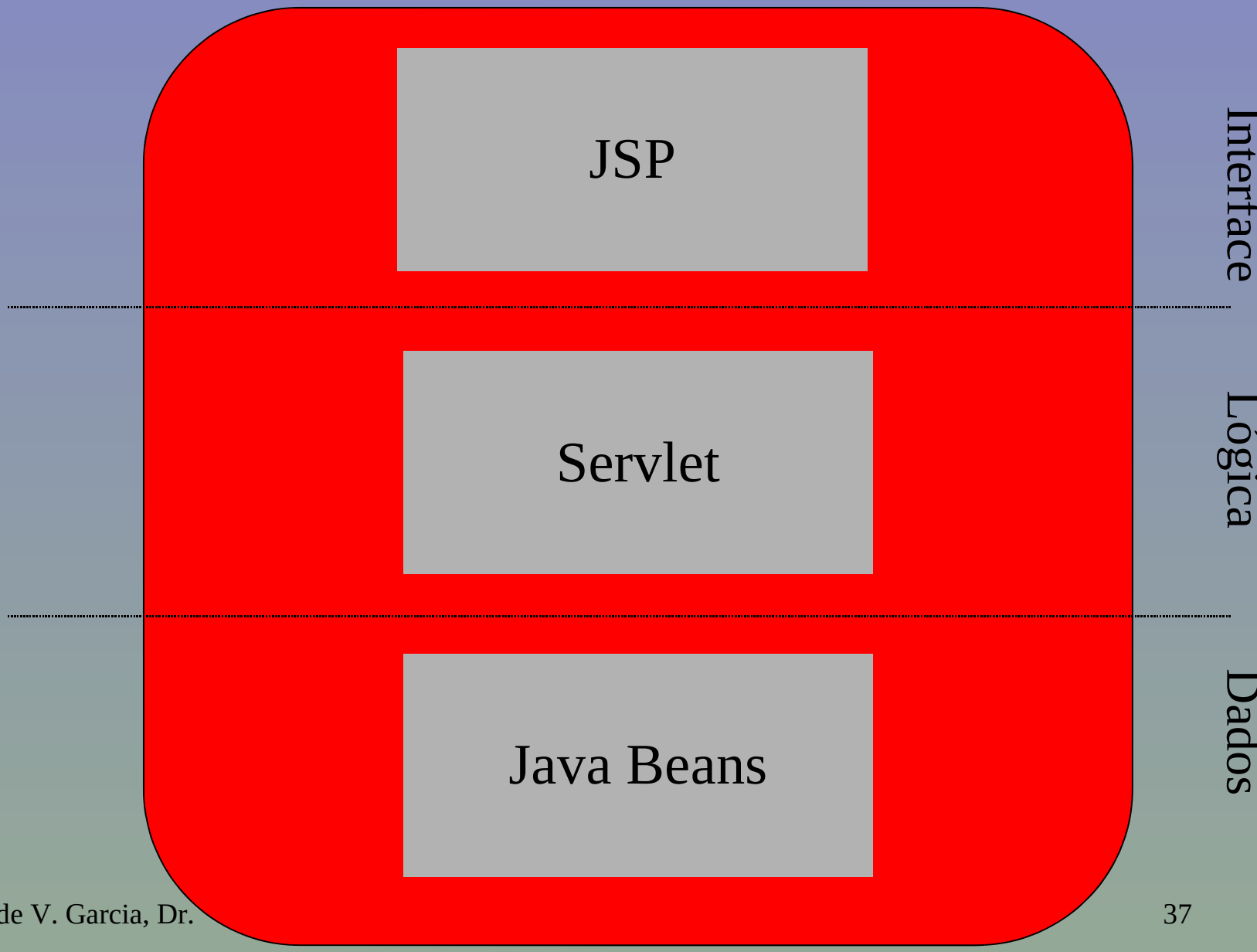
Java no Servidor



Exemplo de Java Bean

```
package br.xpto;

public class UsuarioLogado {

    private String id;
    private int nivelAcesso;

    public String getId() {
            return id;
    public void setId(String id) {
            this.id = id;
    }
    public int getNivelAcesso() {
            return nivelAcesso;
    }
    public void setNivelAcesso(int nivelAcesso) {
            this.nivelAcesso = nivelAcesso;
    }
}
```

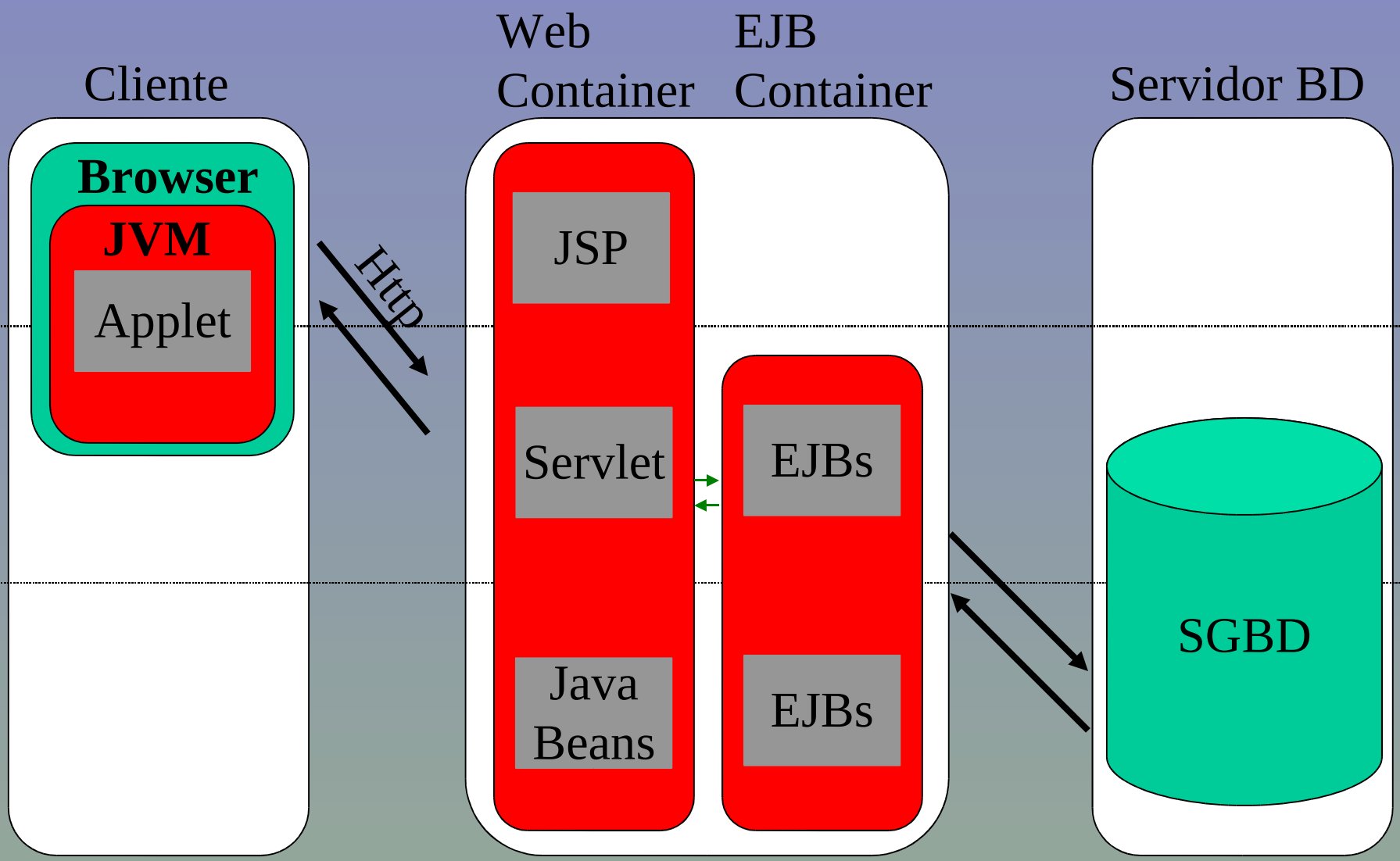
Java no Servidor



Arquitetura MVC

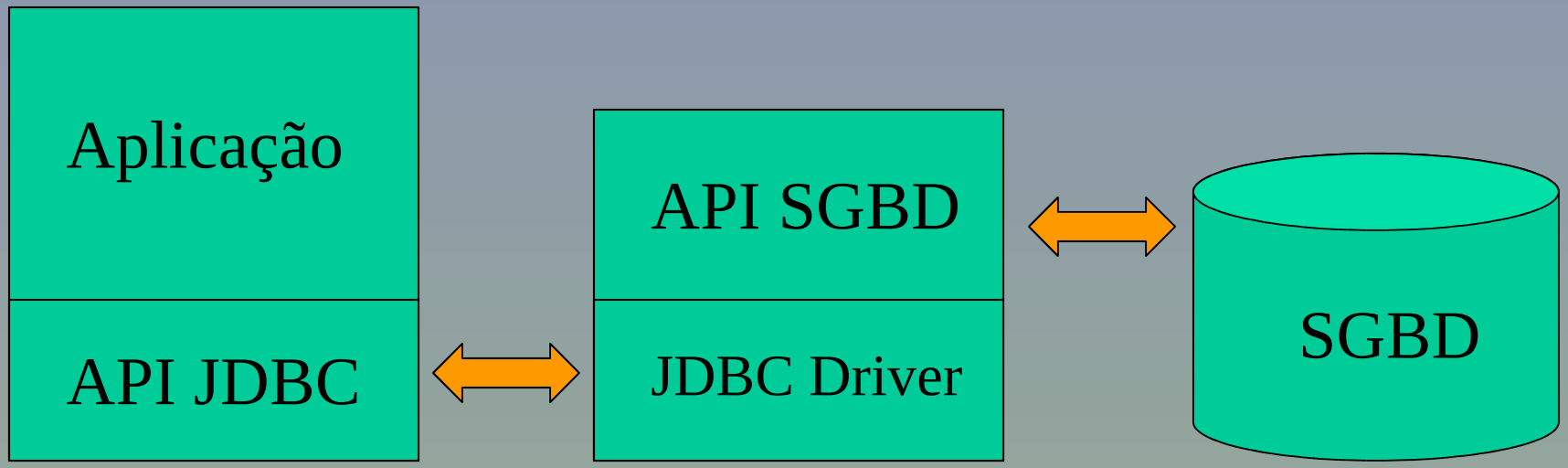
- Padrão Arquitetural da plataforma J2EE
- Model
  - Dados de negócio
- View
  - Apresentação para o cliente
- Controller
  - Lógica de controle
- Onde entra a lógica da aplicação?

EJBs


Cliente
Web Container
EJB Container
Servidor BD
Browser
JVM
Applet
Http
JSP
Servlet
Java Beans
EJBs
EJBs
SGBD

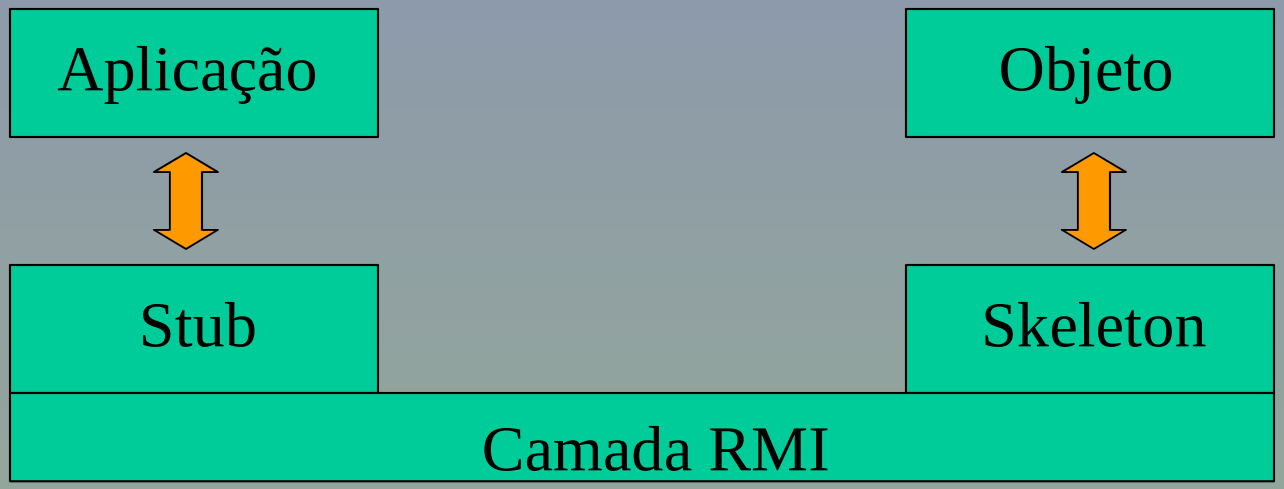
JDBC

- API padrão para acesso a base de dados heterogêneas.
- JDBC 2.0: Tipos de dados SQL 3, DataSource



RMI - IIOP

- Protocolo que permite invocar métodos remotos de objetos escritos em Java.



Exemplo RMI

```
import java.rmi.*;

interface HoraI extends Remote {
    long getHoraCerta() throws RemoteException;
}
```

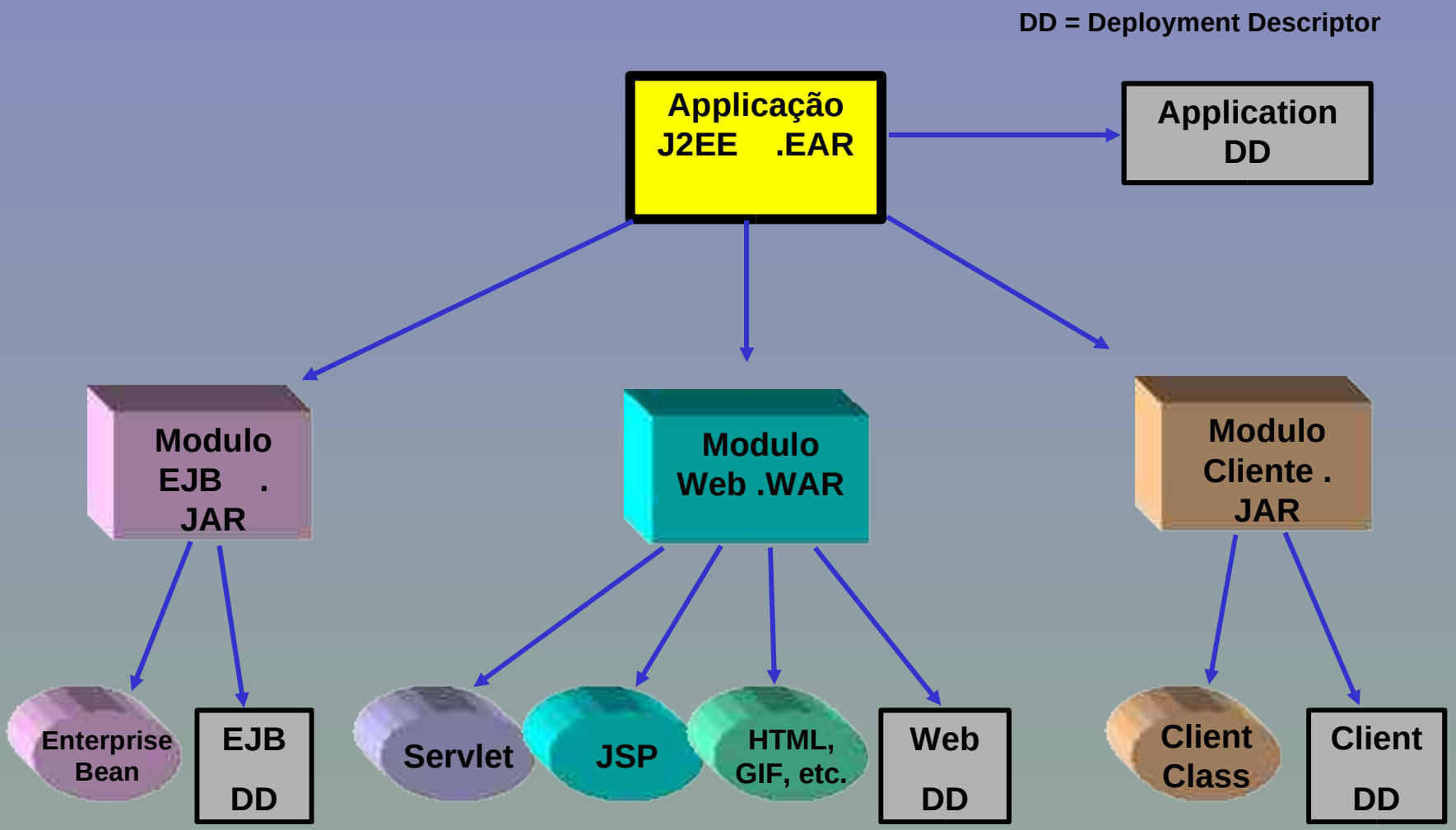
Exemplo RMI

```
public class Hora extends UnicastRemoteObject implements Horal {

    public long getHoraCerta()  throws RemoteException {
        return System.currentTimeMillis();
    }

    public Hora() throws RemoteException { }

    public static void main(String[] args)   throws Exception {
         LocateRegistry.createRegistry(2005);
         Hora h = new Hora();
         Naming.bind("//200.20.120.166:2005/hora", h);
    }
}
```

Exemplo RMI

```
public class ClienteHoraCerta {
    public static void main(String[] args)  throws Exception {
        HoraI t = (HoraI)Naming.lookup(
                "//200.20.120.166:2005/hora");
        for(int i = 0; i < 10; i++)
            System.out.println(”Hora Certa = " +  t.getHoraCerta());
  }
}
```

Deployment


DD = Deployment Descriptor
Applicação J2EE .EAR
Application DD
Modulo EJB . JAR
Modulo Web .WAR
Modulo Cliente . JAR
Enterprise Bean
EJB DD
Servlet
JSP
HTML, GIF, etc.
Web DD
Client Class
Client DD

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Java
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

Linguagem Java

- A linguagem Java é Orientada a Objetos
- A linguagem é totalmente portável
- A linguagem implementa desalocação automática de memória

Linguagem Java

- A linguagem Java é Orientada a Objetos
  - Custo de ligação dinâmica para implementar o polimorfismo.
  - Custo de indireção para a semântica por referência.
  - Alto custo do mecanismo de tratamento de exceções.

Linguagem Java

- A linguagem é totalmente portável
  - Custo de interpretação dos bytecodes.
  - Custo de interpretação dos tipos primitivos.

Linguagem Java

- A linguagem implementa desalocação automática de memória.
  - Performance. 
  - Consumo de memória. 

Performance

- Apesar dos custos de abstração da linguagem Java. A performance de servlets é equivalente a de CGI C++.
- Os principais servidores de aplicação utilizam compiladores *just in time* - JIT.
- e-bay: 320M page views/dia. 2100 transações/segundo.

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

J2EE × .NET

J2EE	.NET
Padrão Aberto	Proprietário
Multiplataformas	Windows
Treino Difícil	Treino Difícil
Diversos IDEs	IDE Microsoft
Middleware Aberto	Middleware Proprietário
Java, (Groovy)	C#, VB, C++, outras
bytecode	IL
Suporta Web Services	Suporta Web Services

Sumário

- Plataforma J2EE
- Servidores de Aplicação
- Arquitetura J2EE
- Performance
- Comparação com .NET
- Novas características

Novidades

- Autoboxing
- Comando for
- Classes Genéricas

Antes

```
public class Freq {
    private static final Integer ONE = new Integer(1);

    public static void main(String[] args) {
        // Map: String -> Integer
        Map m = new TreeMap();

        for (int i=0; i<args.length; i++) {
            Integer freq = (Integer) m.get(args[i]);
            m.put(args[i], freq==null ?
                    ONE : new Integer(freq. intValue()+1));
        }
        System.out.println(m);
    }
}
```

Depois

```
public class Freq {
    public static void main(String[] args) {
        Map<String, Integer> m =
                            new TreeMap<String, Integer>();
        for (String word : args) {
            Integer freq = m.get(word);
            m.put(word, freq == null ? 1 : freq + 1);
        }
        System.out.println(m);
    }
}
```

Problemas

- **Performance**
- **Memória**
- **Semântica dos programas**:

```
int n;
....
Map m = new TreeMap <Integer,String>;
m.put(n, “alo”);
String s = m.get(n);
```

- **Pergunta: “alo”.equals(s) ?**
  (a) Sim
  (b) Não
  (c) Depende

Novidades

- StringBuilder

Qual é o código gerado pelo comando abaixo?

String s = “a” + “b”;

(a) s = "a".concat("b");

(b) s = (new StringBuffer()).append("a").append("b").toString();

(c) s = (new StringBuilder()).append("a").append("b").toString();

Novidades

- Enum

```
public class C1 {
    public enum Dias{ SEGUNDA, TERCA, QUARTA, QUINTA,
    SEXTA, SABADO, DOMINGO }

    public static void main(String[ ] args) {
    System.out.print(“Dias Uteis: ");
    for (Dias d:EnumSet.range(Dias.SEGUNDA, Dias.SEXTA})
        System.out.print(d + " \n");
    }
}
```

Novidades

- Static imports

```
import static NomeDaClasse.Identificador;

import static NomeDaClasse.*;
```

Novidades

- Métodos de Aridade Variável

```
public static String format(String pattern, Object... nomes) {
    ...
}
    ....
format(“Nomes são {0} {1} {2}", args));
    ....
format("Nomes são {0} {1} {2}", “João", "José", "Marcos"));
```

- Overloading?

Novidades

- **Maiores novidades estão no suporte de *runtime***
- **UML/EJB Mapping Specification**
- **Concorrência**
- **Portlet Specification**
- **Java Servlet and JSP Performance Benchmark**
- **Distributed Real-Time Specification**
- **Java OLAP Interface (JOLAP)**
- **Data Mining API**
- **Multiarray package**
- **JDOM 1.0**
- **J2EE API for Continuous Availability**
- **Timer for Application Servers**

- Perguntas?
- Novas Palestras?

garcia@de9.ime.eb.br