

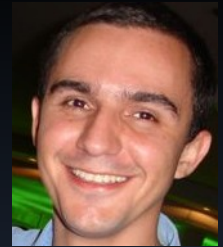


Programação Web

CURSO DE INFORMÁTICA.
4º SEMESTRE

PROF.: LUIS AUGUSTO MACHADO MORETTO. ESP.

PROFESSOR:



Formação

- Bacharel em Ciências da Computação - UNIVALI 2005;
- Especialista em Administração - UFSC 2007
- Mestrando em Engenharia e Gestão do Conhecimento UFSC 2009-2010
- Possui certificação SCJP e SCJA da Sun Microsystems

Interesses

- Engenharia de Software
- Engenharia do Conhecimento
- Análise e desenvolvimento de sistemas
- Arquitetura de Software
- Agentes Inteligentes
- Métodos Ágeis



Conteúdo


- TOMCAT
 - Conceitos do Webserver
- Servlets
 - Servlets/JSP Web Servers
 - Protocolo HTTP
 - Definição
 - Funcionamento
 - Recuperação de Parâmetros
 - Geração de Respostas
 - Recuperação de Informações
 - Uso de Cookies
 - Monitoramento de sessões
- JSP
 - Introdução
 - Elementos de Script e Diretivas
 - Java Beans e JSP
 - Tags customizadas
- JSP vs. Servlets
 - Modelagem
 - Ativação de JSP a partir de Servlets
- J2EE

Na Internet

- Tutorial de Servlets da Sun
 - <http://www.javasoft.com/docs/books/tutorial/servlets/index.html>
- Introdução a Servlets - R. G. Baldwin
 - <http://home.att.net/~baldwin.rick/Advanced/Java680.htm>
- Introdução a JSP - R. G. Baldwin
 - http://softwaredev.earthweb.com/java/sdjjavase/article/0,,12395_626381,00.html
- Faqs, Ferramentas, Exemplos
 - <http://www.jspbrasil.com.br>




Ferramentas Utilizadas

- Netbeans
 - www.netbeans.org
 - Tomcat
 - jakarta.apache.org
 - MySQL
 - Mysql.org
- 



Netbeans

- Ferramenta da SUN
 - Versão Atual: 6.5
 - Divide o trabalho em projetos, pacotes e classes
 - Divide as classes em diversas partes: definição, atributos, métodos
 - Ao salvar uma classe ela será automaticamente compilada
- 




Tecnologias Web

- Cliente
 - Javascript
 - Applets
 - Flash
- Server
 - ASP
 - JSP/Servlets
 - PHP
 - XML/XSL

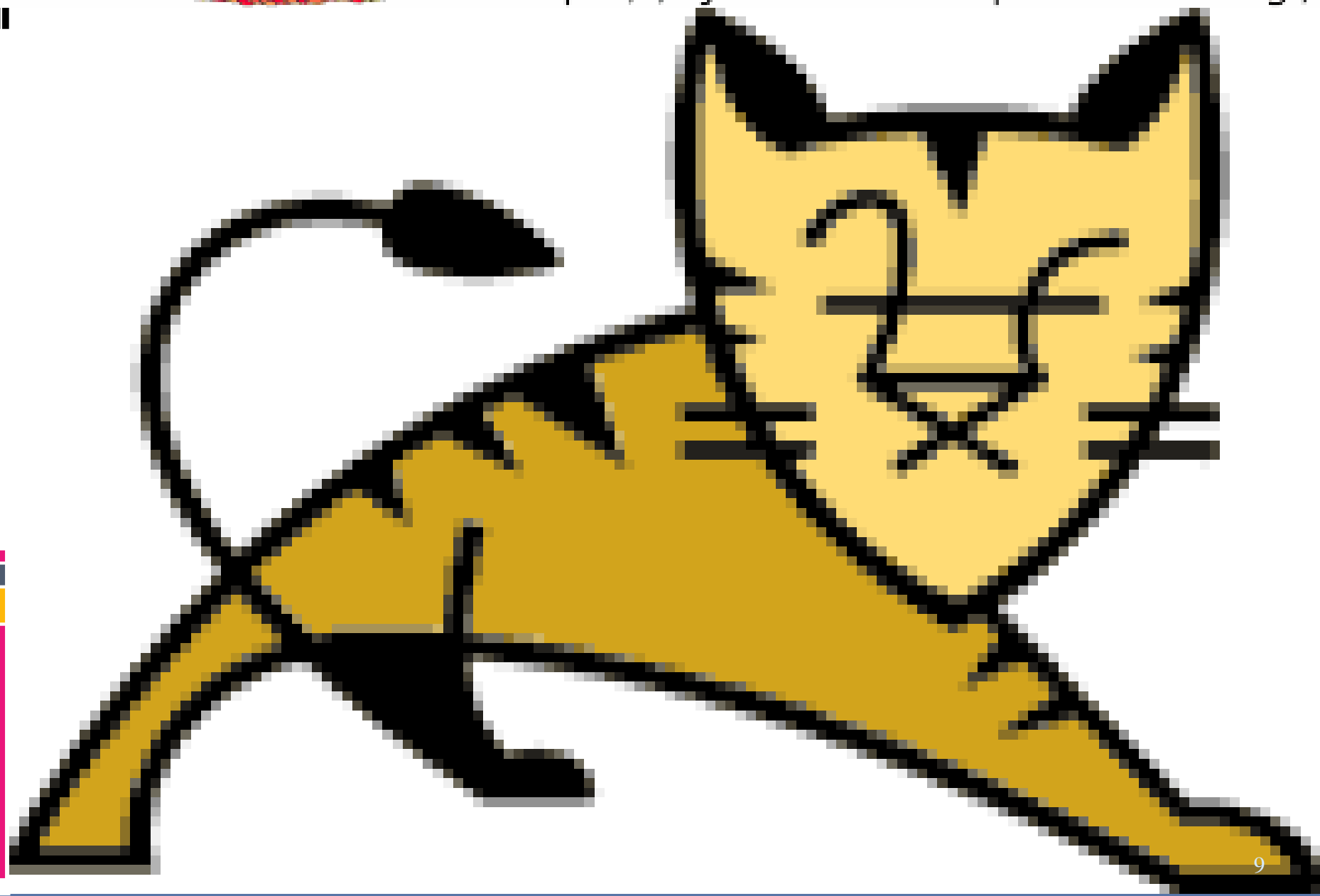


Histórico Server-Side

- HTML: sem conteúdo dinâmico
 - CGI: Common Gateway Interface
 - ASP: Active Server Pages
 - Servlets
 - JSP: Java Server Pages
 - PHP: Open source
- 

The **Apache Jakarta Project**

<http://jakarta.apache.org/>



Web Server

- **Web server** :aplicação que responde à requisições do protocolo HTTP retornando recursos 'web' como HTML arquivos, imagens, applets,na Internet
(e.g., Apache server)
- **Servlet container (ou servlet engine)**:
Middleware ou ferramenta intermediária que roda os Servlets e gerencia o ciclo de vida, (e.g., Tomcat servlet container)



■ Servidor número 1 na Internet

- Apache :65% dos websites do mundo são hospedados pelo servidor Apache

■ Porque sua popularidade??

- Livre!
- Altamente customizável
- implementa várias funcionalidades(e.g., segurança/acesso controle, etc)
- extensível possibilitando acoplar módulos(e.g., servlet engine, segurança, ...)
- Independente de plataforma (suporta várias arquiteturas de hardware e sistemas operacionais)

Windows, Unix, Linux, Solaris, Mac, etc.

Apache - Web Server

- Apache Software Foundation
 - organização não lucrativa
 - mantém e desenvolve programas open-source
- Maiores informações:
 - *www.apache.org*

Tomcat

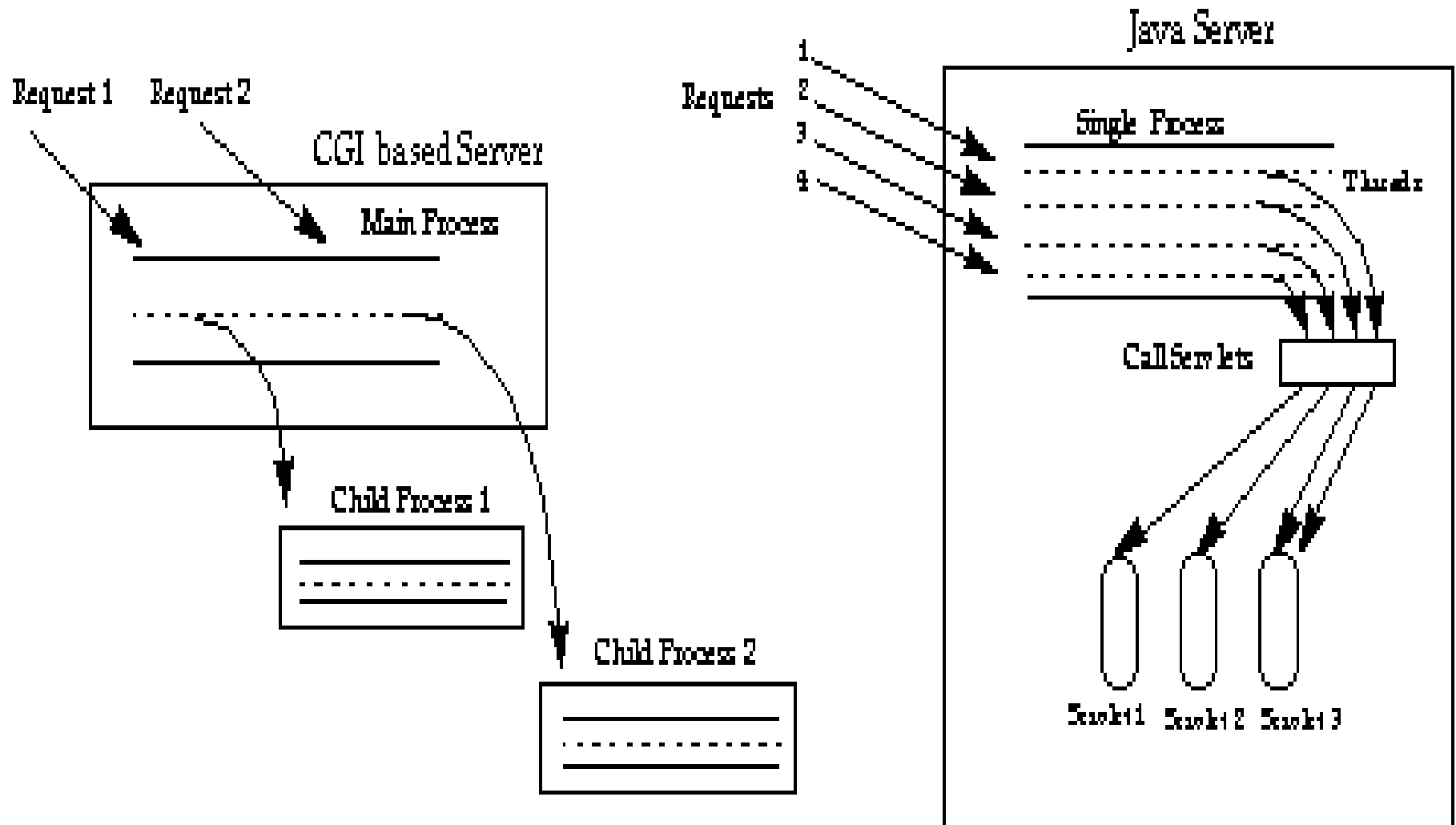
- Container **Servlet** que roda aplicações JAVA;
- Servidor de aplicação da APACHE que roda Servlets e JSP
- Website
<http://jakarta.apache.org/tomcat/>



0 que são os Java Servlets?

- Tecnologia Java para endereçar arquitetura CGI (Common Gateway Interface)
- Util para construção de websites dinâmicos
 - Páginas baseadas em formulários
 - Quando os dados mudam frequentemente Ex: relatórios
 - Possibilita a conexão dos dados com Bancos
- Executado no Container
- Escritos em Java e seguem uma arquitetura padronizadas
- Rápido eficiente e relativamente fácil
- Geração de conteúdo dinâmico

Processes vs Threads



O que é um JSP?

- Sigla de JavaServlet Page
- Extensão da tecnologia Java Servlet que permite HTML estático e dinâmico em um só arquivo;
- JSP são documentos de texto que possuem a extensão jsp que possui uma combinação de HTML e XML como tags ou Scriptlets;
- As tags e os Scriptlets possuem a lógica para exibição do conteúdo dinâmico;
- JSP são pré-processadas e compiladas em um Java servlet que roda no container (e. g. Tomcat).

Prós e contras

- Tomcat possui vários fatores prós
- Como um container Standalone tem algumas desvantagens como:
 - Não é tão rápido como o Apache para páginas estáticas;
 - Não é tão customizável como o Apache;
 - Não é tão robusto como o Apache;
 - Não suporta funcionalidades dos módulos do Apache(e.g., Perl, PHP, security)

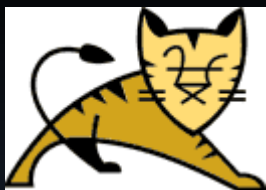
Integração Apache x Tomcat

- Apache como o webserver + Tomcat para execução de Servlet and JSP
- Funcionamento básico:
 - Permitir que o Tomcat ouça as requisições do Apache
 - Permitir que o Apache se comunique com o Tomcat
 - Apache delega o processamento de JSP e Servlets para a instância do Tomcat
- Problema
 - Como eles se comunicam?
 - > Apache é escrito em C mas o Tomcat 100% Java
 - > Precisamos de uma ponte entre eles

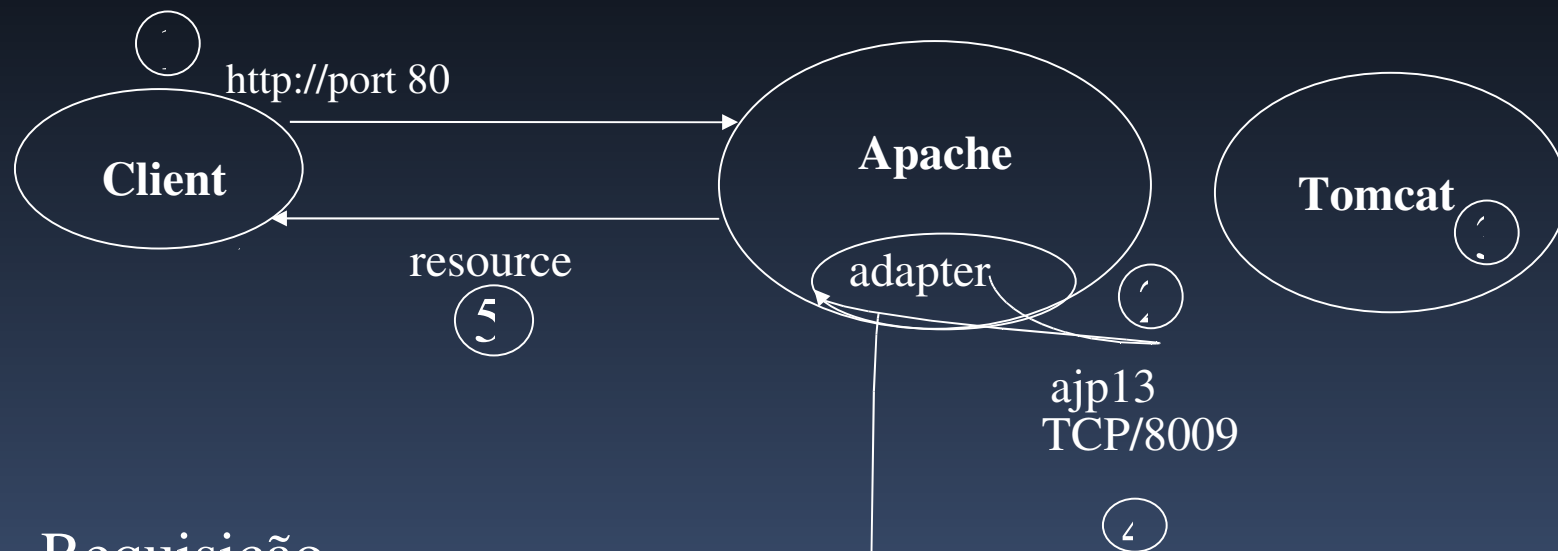
JK Connector

Mecanismo de comunicação entre o TOMCAT e o APACHE

- Denominado “web server adapter” ou “connector”
- Implementado como uma biblioteca (e.g., mod_jk.dll, mod_jk.so)
- Usa e gerencia conexões TCP/IP
- Utiliza o **ajp13** como protocolo de comunicação (definido no server.xml subdiretorio do tomcat)



E



- Requisição
1. Apache delega para o Tomcat a execução de JSPs e Servlets
 2. Tomcat executa JSPs, e Servlets
 3. Tomcat envia para o Apache os resultados
 4. Apache envia a resposta para o cliente

Passos para instalar o TOMCAT

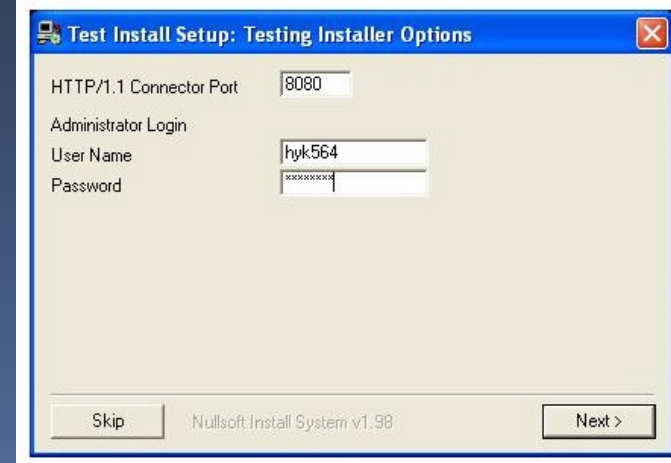
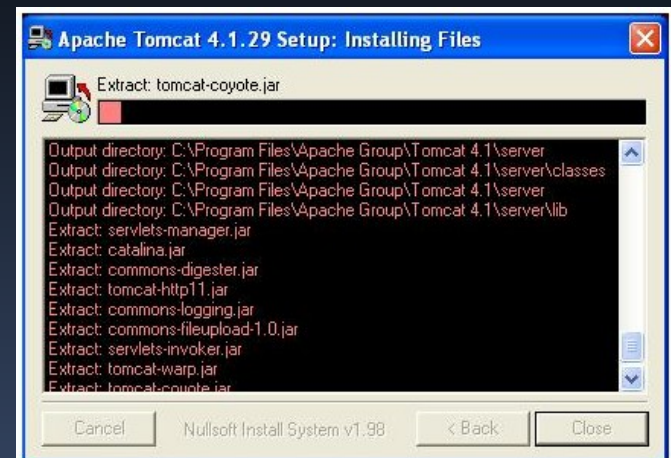
- Instalar o Java SDK
- Instalar o Tomcat
- Configurar as portas do serviço
- Testar

1. Instalar o JDK

- Download da última versão do SDK
- *<http://java.sun.com/j2se>*
- Configurar a variável de ambiente JAVA_HOME para a raíza de instalação do SDK (e.g., c:\j2sdk1.6)
- Testar no prompt de comando digitando “echo %JAVA_HOME%” e verificar que retorna o caminho configurado para a variável(e.g., j2sdk1.6)

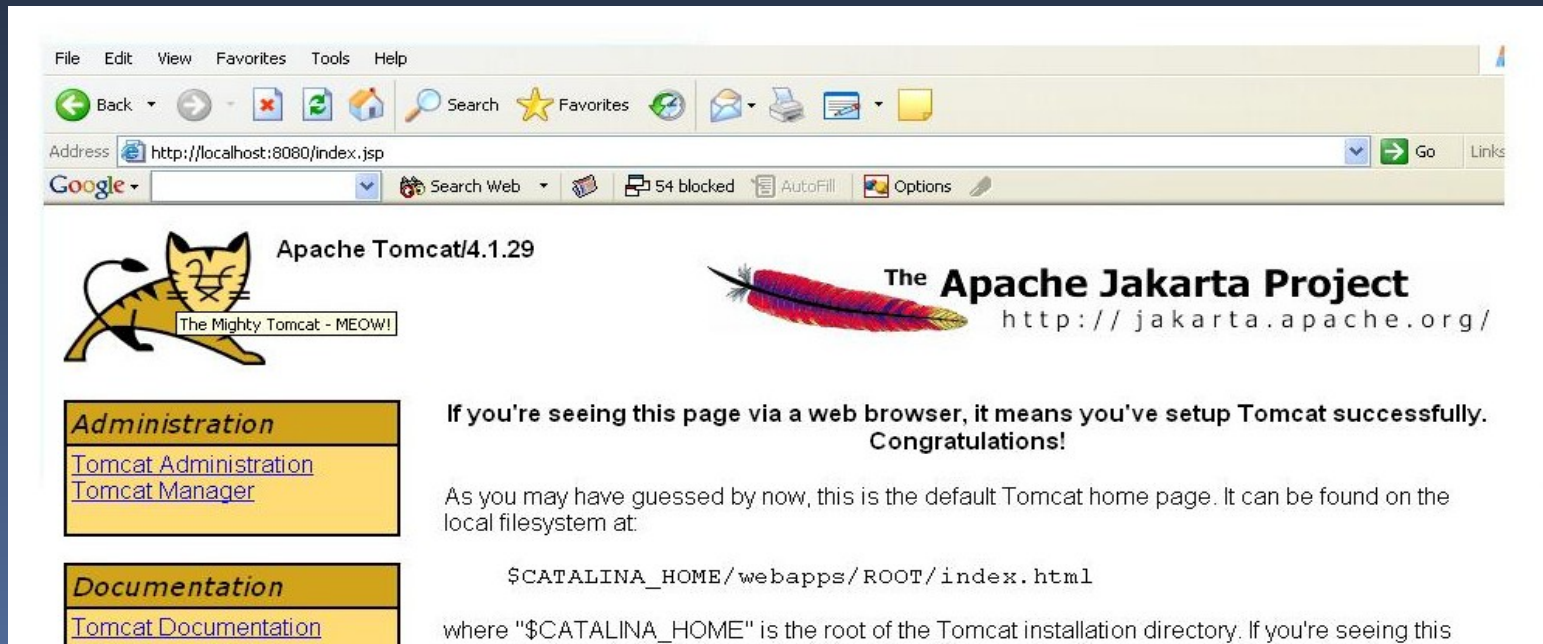
2. Instalar o Tomcat

- The latest version is Tomcat 5.5.x.
- Faça o download do .EXE instalador do Tomcat em <http://jakarta.apache.org/tomcat/> (e.g. tomcat-5.5.exe)
- Siga as instruções do instalador
- Especifique:
 - HTTP/1.1 Connector Port como **8080**
 - **User Name e Password**

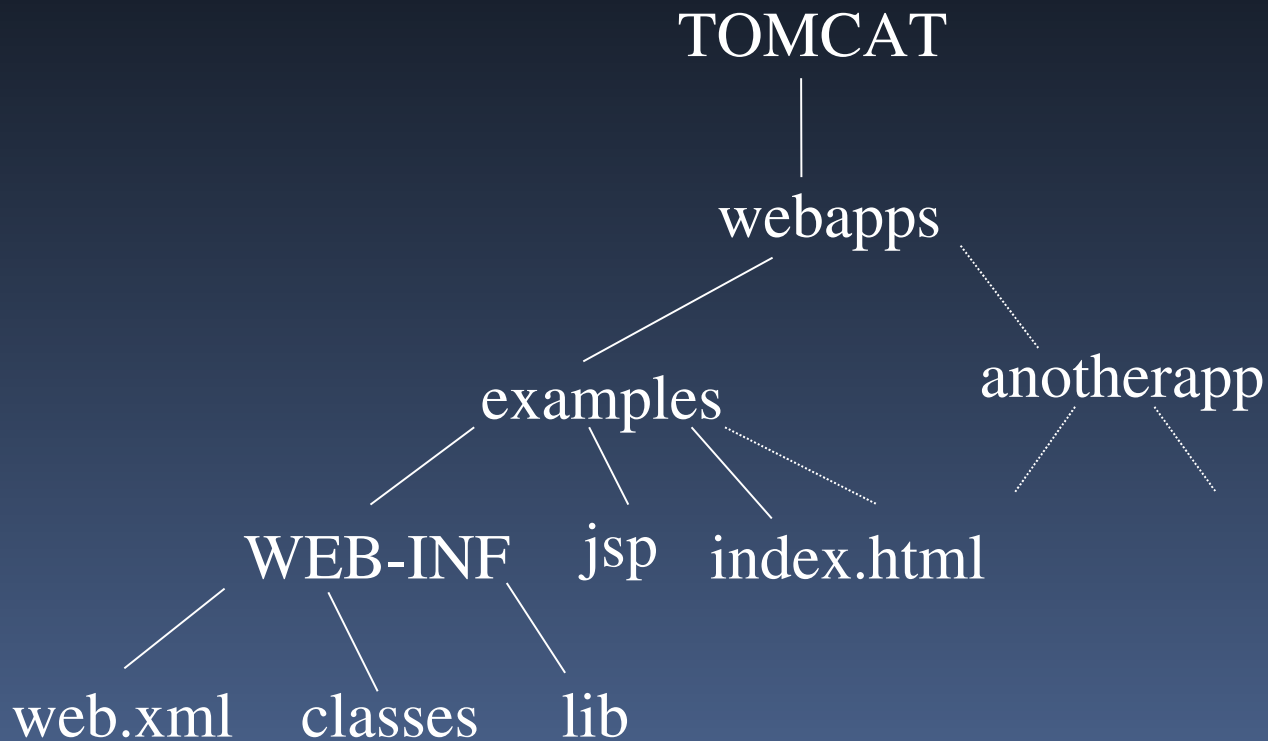


3. Testar instalação do Tomcat

- Para testar a instalação: Abra o navegador utilizando o endereço: **http://localhost:8080**
- Se você visualizar uma página similar a de baixo...BINGO!



Estrutura de arquivos do Tomcat



Referencias

- **Apache Server:** *<http://apache.org>*
- **Tomcat:** *<http://jakarta.apache.org/tomcat/>*
- **JAVA:** *<http://java.sun.com>*
- **Sublet and JSP Overview:**
<http://www.apl.jhu.edu/~hall/java/Servlet-Tutorial/Servlet-Tutorial-Overview.html>



SERVLET

Servlets - Web Servers

- Rodam no servidor com o objetivo de monitorar requisições
- Web Servers para Servlets/JSP
 - Apache Tomcat
 - IBM Websphere
 - Allaire JRun
 - BEA Weblogic
 - Mais Servidores:
 - <http://java.sun.com/products/servlet/industry.html>

Servlets - Na Internet

- Provedores gratuitos com suporte para Servlets/JSP e JDBC
 - <http://www.mycgiserver.com>
 - <http://www.webappcabaret.com>
 - <http://dev.startcom.org>
 - <http://java.isavvix.com>
 - <http://code.google.com/intl/pt-BR/appengine/>

Servlets - Protocolo HTTP

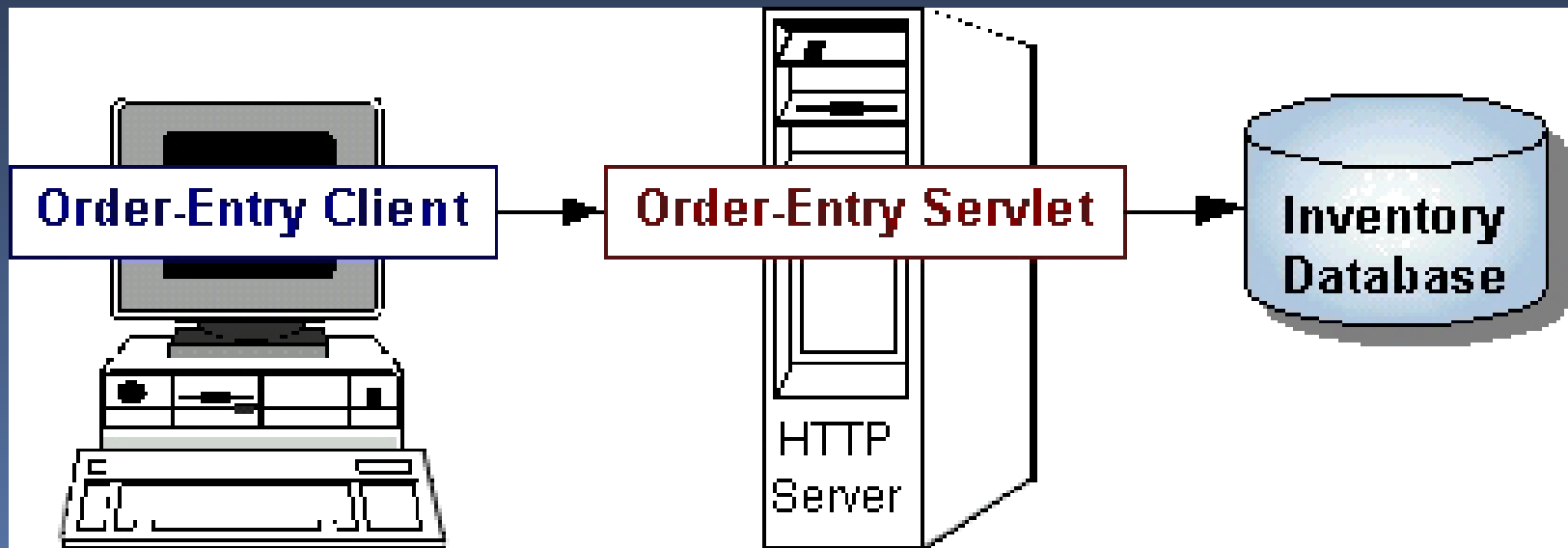
- Hyper Text Transfer Protocol
- Protocolo padrão para conexão entre browsers
- Quatro etapas: conexão, requisição, resposta e fechamento
- Utiliza o MIME para codificação dos dados

Servlets - Introdução

- Classes Java que rodam no servidor
- Applet - Browser, Servlet - Server
- Uso de Servlets
 - Geração de Conteúdo HTML Dinâmico
 - Sincronização de requisições
 - Redirecionamento de requisições

Servlets - Funcionamento

- Servlets são executados a partir de requisições de clientes
- Executam alguma tarefa (BD, Classes de negócio) e retornam uma página




Servlets - Criação

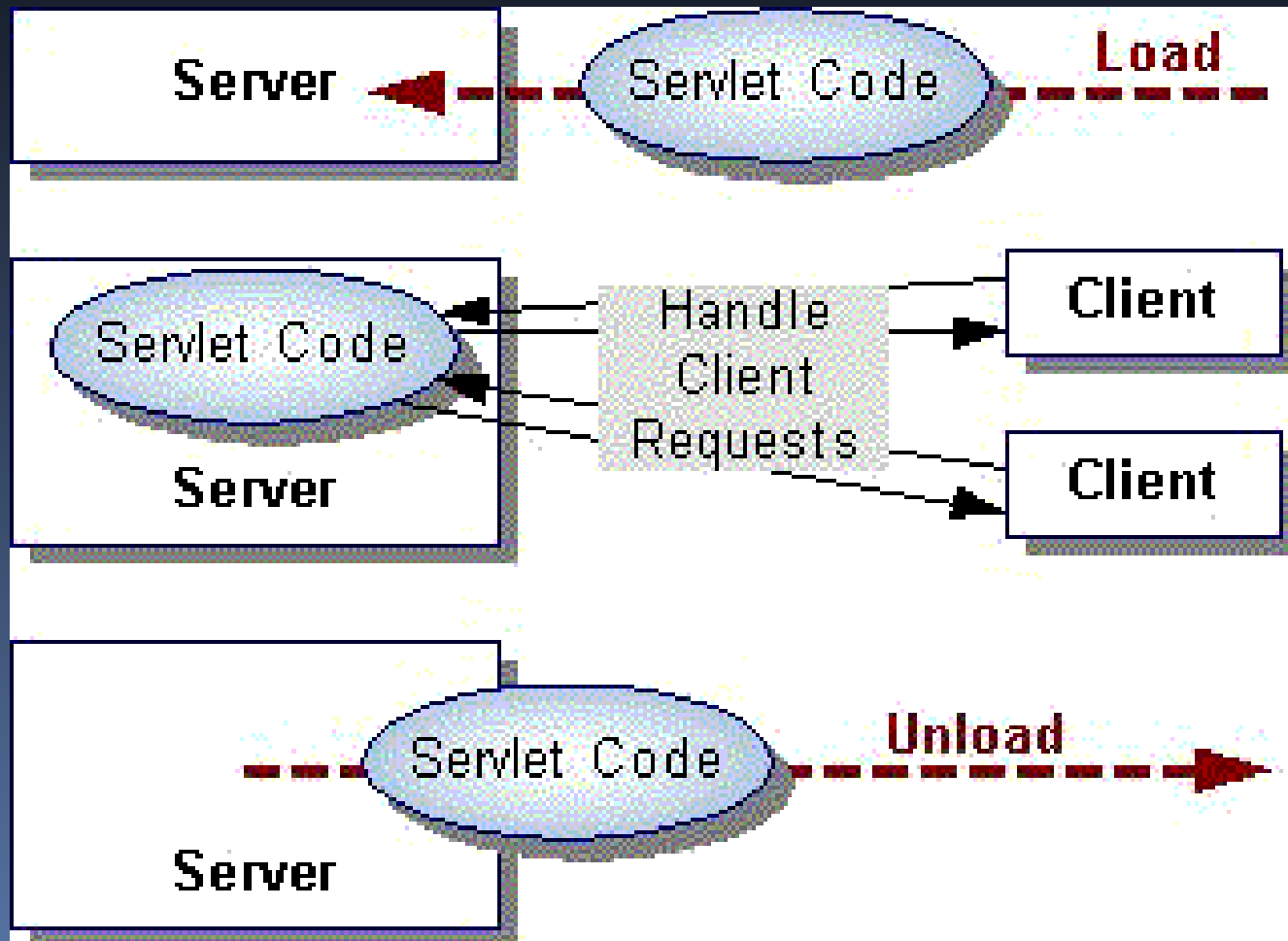
- Criar uma classe derivada de `HTTPServlet`
- Implementar os métodos `doGet` e/ou `doPost` usando os parâmetros:
 - `request` para tratamento das requisições
 - `response` para tratamento das respostas
 - `getWriter` para pegar a stream de saída
- Chamar o servlet a partir do browser



Servlets - Request/Response

- Request - classe `HttpServletRequest`
 - Encapsula os dados do cliente
 - Header, cookies, parâmetros
 - Response - classe `HttpServletResponse`
 - Encapsula os dados para o cliente
 - Header, cookies, conteúdo
- 

Servlets - Ciclo de Vida



Servlets - Ciclo de Vida

- init
- init (ServletConfig)
- service
- doGet/doPost/doPut/doDelete
- getServletInfo
- destroy

Servlets - métodos init

- `init()` - Primeiro método chamado, usado para inicializações em geral como abertura de conexões para BD
- `init(ServletConfig)` - chamado após `init`, usado para leitura de parâmetros do servidor com `getInitParameter()`

Servlets - doGet e doPost

- Existem duas maneiras de se enviar dados pelo browser: Get e Post
- Como o tratamento é o mesmo chama-se um método a partir do outro
- Get:: dados vem junto com a URL
- Post:: dados vem separados da URL

Servlets - doGet e doPost


```
public void doGet (HttpServletRequest request,
                  HttpServletResponse
response)
{
    String nome =
request.getParameter("nome");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("Texto recebido: " + nome);
    out.close();
}
```

Servlets - service

- A cada nova requisição o servidor cria um novo thread e chama *service*
- O método *service* verifica a requisição e chama o método apropriado
- Se for sobreposto deverá fazer todo o tratamento de verificação e chamada dos métodos



Servlets - outros métodos


- `destroy` - chamado na destruição do servlet por inatividade ou a pedido do administrador. Usado para finalizar tarefas iniciadas por `init`
 - `getServletInfo` - deve retornar um objeto `String` contendo uma descrição do servlet
- 

Servlets - parâmetros

- Um servlet pode receber parâmetros.
- Estes parâmetros são o resultado do envio de dados de um form html.
- Recuperação de parâmetros:
 - `getParameter`: para pegar um parâmetro específico
 - `getParameterNames`: para recuperar todos os parâmetros passados



Prática

- Livraria Virtual
 - Desenvolvimento por partes durante o horário das aulas
 - Cada parte complementa a anterior
 - Conteúdo: login de clientes, catálogo de livros, carrinho de compra
- 

Prática - Parte I

- Construir uma página HTML para entrada no sistema (login)
- Construir a classe CatalogoServlet que recebe como parâmetros o nome do usuário e retorna um html em formato de tabela contendo livros e preços

Prática - Parte II

- Para melhorar a organização do sistema construir as classes:
 - Livro que contem código, título, autor, editora, categoria, quantidade e preço
 - LivroBD que retorna um array de livros. Em um sistema seria recuperado de uma tabela mas neste exemplo deixar fixos os livros.

Estrutura básica de uma classe Servlet

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

```
public class SomeServlet extends HttpServlet {
```

```
    // Handle get request
```

```
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
```

```
        // request - access incoming HTTP headers and HTML form data
```

```
        // response - specify the HTTP response line and headers
```

```
        // (e.g. specifying the content type, setting cookies).
```

```
        PrintWriter out = response.getWriter(); //out - send content to browser
```

```
    }
```

```
}
```



Um Servlet

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

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {

        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```



Gerando HTML

```
public class HelloWWW extends HttpServlet {  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response) throws  
        ServletException, IOException {
```

```
        response.setContentType("text/html");
```

```
        PrintWriter out = response.getWriter();
```

```
        out.println("<HTML>\n" +
```

```
        "<HEAD><TITLE>HelloWWW</TITLE></HEAD>\n"  
        +
```

```
                "<BODY>\n" + "<H1>Hello WWW</H1>\n"  
                "</BODY></HTML>");
```

```
    }
```

```
}
```


Formulário HTML para envio de dados

```
<FORM ACTION="/servlet/hall.ThreeParams"
      METHOD="POST">
```

```
  First Parameter: <INPUT TYPE="TEXT"
    NAME="param1"><BR>
```

```
  Second Parameter: <INPUT TYPE="TEXT"
    NAME="param2"><BR>
```

```
  Third Parameter: <INPUT TYPE="TEXT"
    NAME="param3"><BR>
```

```
<CENTER>
```

```
  <INPUT TYPE="SUBMIT">
```

```
</CENTER>
```

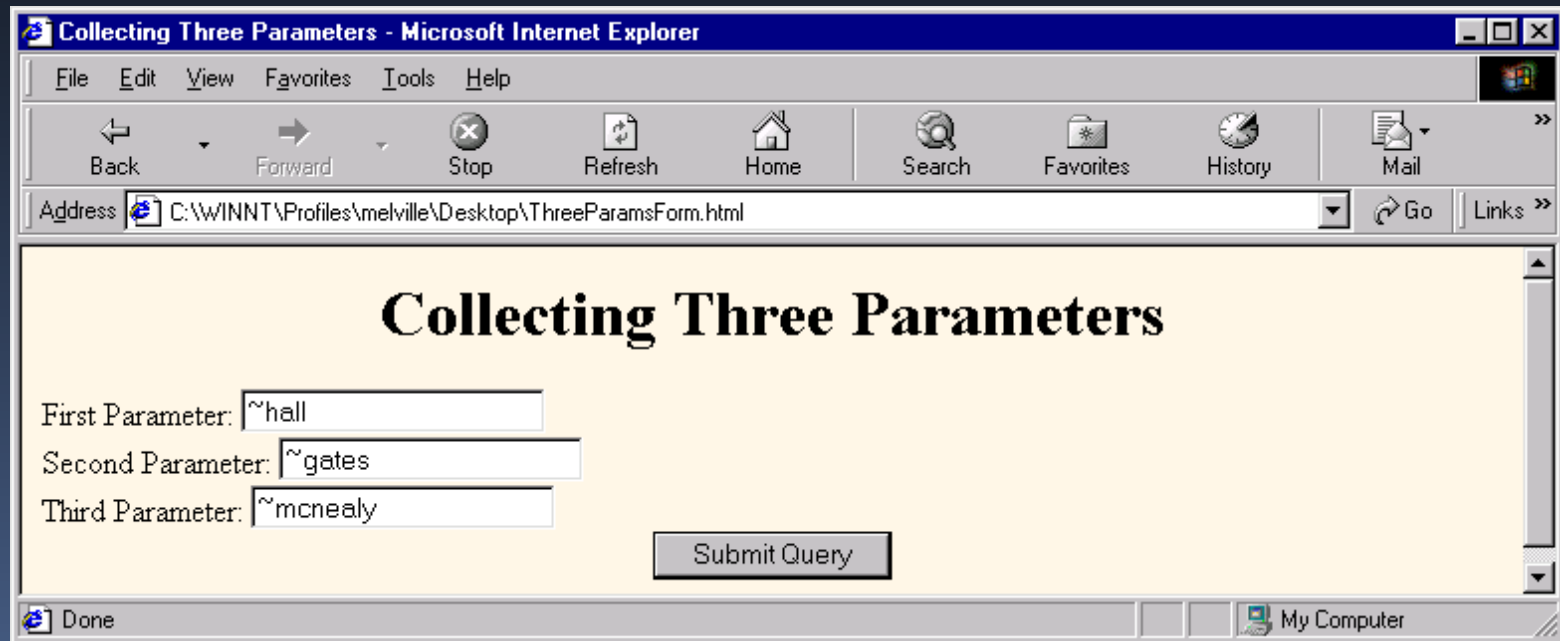
```
</FORM>
```

Lendo parâmetros

```
public class ThreeParams extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse
        response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println(... + "<UL>\n" +
            "<LI>param1: " + request.getParameter("param1") + "\n" +

            "<LI>param2: " + request.getParameter("param2") + "\n" +
            "<LI>param3: " + request.getParameter("param3") + "\n" +
            "</UL>\n" + ...);
    }
    public void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException,
        IOException {
        doGet(request, response);
    }
}
```

Exemplo de formulário



The screenshot shows a Microsoft Internet Explorer window titled "Collecting Three Parameters - Microsoft Internet Explorer". The address bar displays the URL "C:\WINNT\Profiles\melville\Desktop\ThreeParamsForm.html". The main content area has a yellow background and contains the title "Collecting Three Parameters" in a large, bold, black serif font. Below the title, there are three text input fields, each preceded by a label: "First Parameter:", "Second Parameter:", and "Third Parameter:". The first field contains the text "~hall", the second contains "~gates", and the third contains "~mcnealy". To the right of these fields is a "Submit Query" button. The status bar at the bottom shows "Done" and "My Computer".

Collecting Three Parameters

First Parameter:

Second Parameter:

Third Parameter:

Monitoramento de Session

- Cenário – compras em uma loja virtual
- Necessario porque o protocolo HTTP é "stateless" ou seja sem estado!
- API de sessions permite:
 - Acessar objetos de sessão durante uma requisição
 - Criar novos objetos de sessão quando necessário
 - Acessar informações relativas à session
 - Armazenar informação na session
 - Descartar sessão perdidas

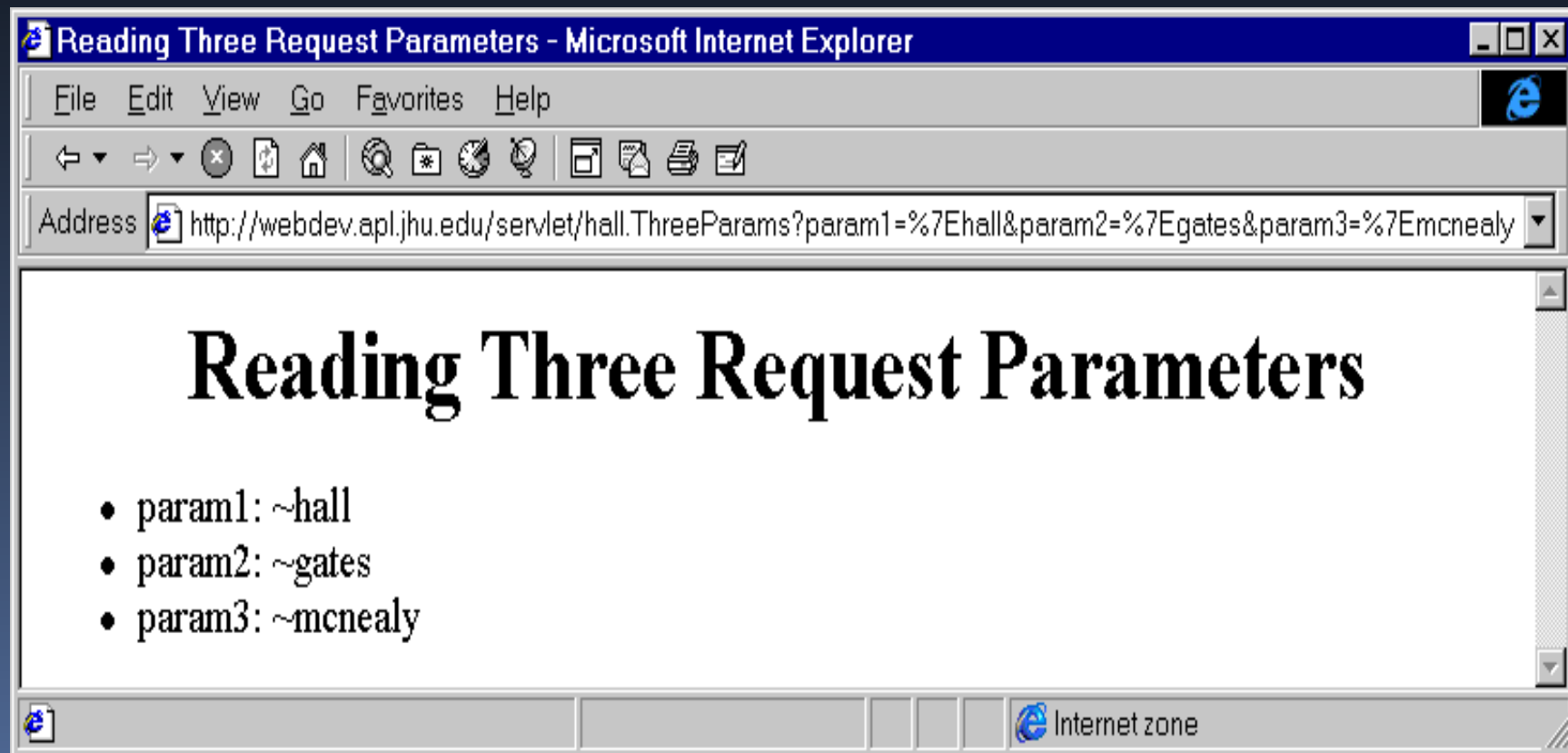
Monitoramento de Session

- Acessando o objeto de session
 - `HttpSession session = request.getSession(true);`
 - O parâmetro `true` cria uma nova session se a mesma não existir
- Gravando objetos na session
 - `session.setAttribute("user", request.getParameter("name"))`
 - Os objetos da session podem ser de qualquer tipo
- Recuperando objeto da session
 - `String name = (String) session.getAttribute("user")`

Monitoramento de Session

- **getId**
 - Identificador único da session
- **isNew**
 - true se o cliente nunca acessou a session
- **getCreationTime**
 - Tempo em milisegundos desde a criação da session
- **getLastAccessedTime**
 - Tempo em milisegundos desde que a session foi acessada
- **getMaxInactiveInterval**
 - Tempo da validade da session antes da inativação
 - Valor negativo indica que a session não expira

Saída do Servlet



Lendo todos os parâmetros

- `Enumeration paramNames = request.getParameterNames();`
 - Parâmetros não ordenados
- `String[] paramVals = request.getParameterValues(paramName);`
 - Array de parâmetros

JSP - Java Server Pages

- Mistura JAVA com HTML
 - Tem a estrutura de um HTML
 - Incorpora o JAVA com tags especiais;
- JSPs são equivalentes aos Servlets
 - Conveniente se muito HTML está envolvido
- Deve ser localizado na mesma pasta dos HTMLS

JSP Sintáxe - I

- Expressão

- `<%= expression %>`
- Current time: `<%= new java.util.Date() %>`

- Scriptlet-

- `<% code %>`
- `<% String queryData = request.getQueryString();
out.println("Attached GET data: " + queryData); %>`

- Declaração

- `<%! code %>`
- `<%! private int accessCount = 0; %>`
Number of accesses to page: `<%= ++accessCount %>`

JSP Sintáxe - II

- Diretivas de página

- `<%@ page att="val" %>`
- `<%@ page import="java.util.*" %>`
- `<%@ page extends="package.class" %>`
- `<%@ page errorPage="url" %>`

- Diretiva de inclusão

- `<%@ include file="url" %>`
- `<%@ include file="/navbar.html" %>`

- Variáveis pré-definidas

- request, response, out, session

JSP Exemplo

```
<html>
<%@ page import="exemplo.*" %>

<%@ include file="/libra/Navbar.html" %>

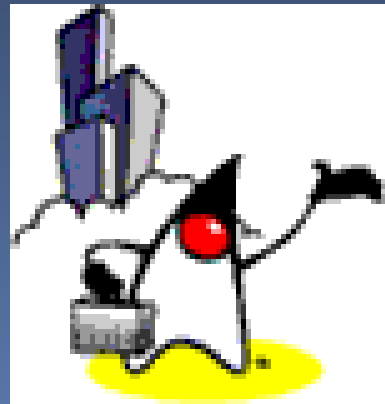
<%= (request.getParameter("username")==null ? "" :
    (request.getParameter("username")+", ")) %>Bem
vindo!

<a href="/servlet/libra.Recommend">Produz as
recomendações.</a>

Para maiores informações veja
<a href="/libra/help.html">Ajuda</a>.
</html>
```

J2EE

- Introdução ao J2EE
- Explicação das tecnologias J2EE
- J2EE aplicações
- J2EE servidores

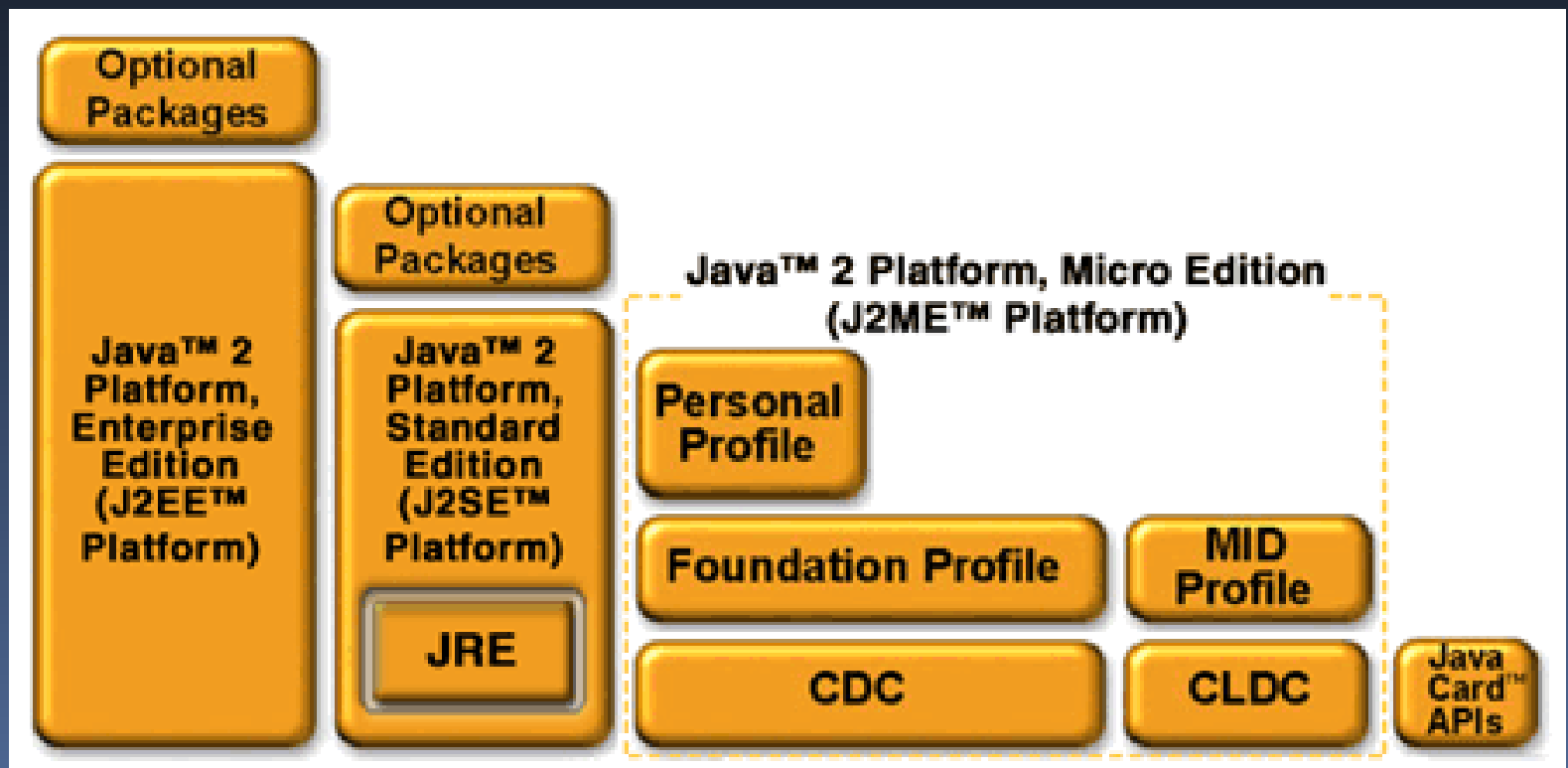




Java 2 Platform


- Criada em junho de 1999
- J2SE – Java 2 Standard Edition
 - Java para Desktop / estação de trabalho
 - <http://java.sun.com/j2se>
- J2ME – Java 2 Micro Edition
 - Java para dispositivos móveis (celular / palm)
 - <http://java.sun.com/j2me>
- J2EE – Java 2 Enterprise Edition
 - Java para o servidor
 - <http://java.sun.com/j2ee>

Java 2 Platform

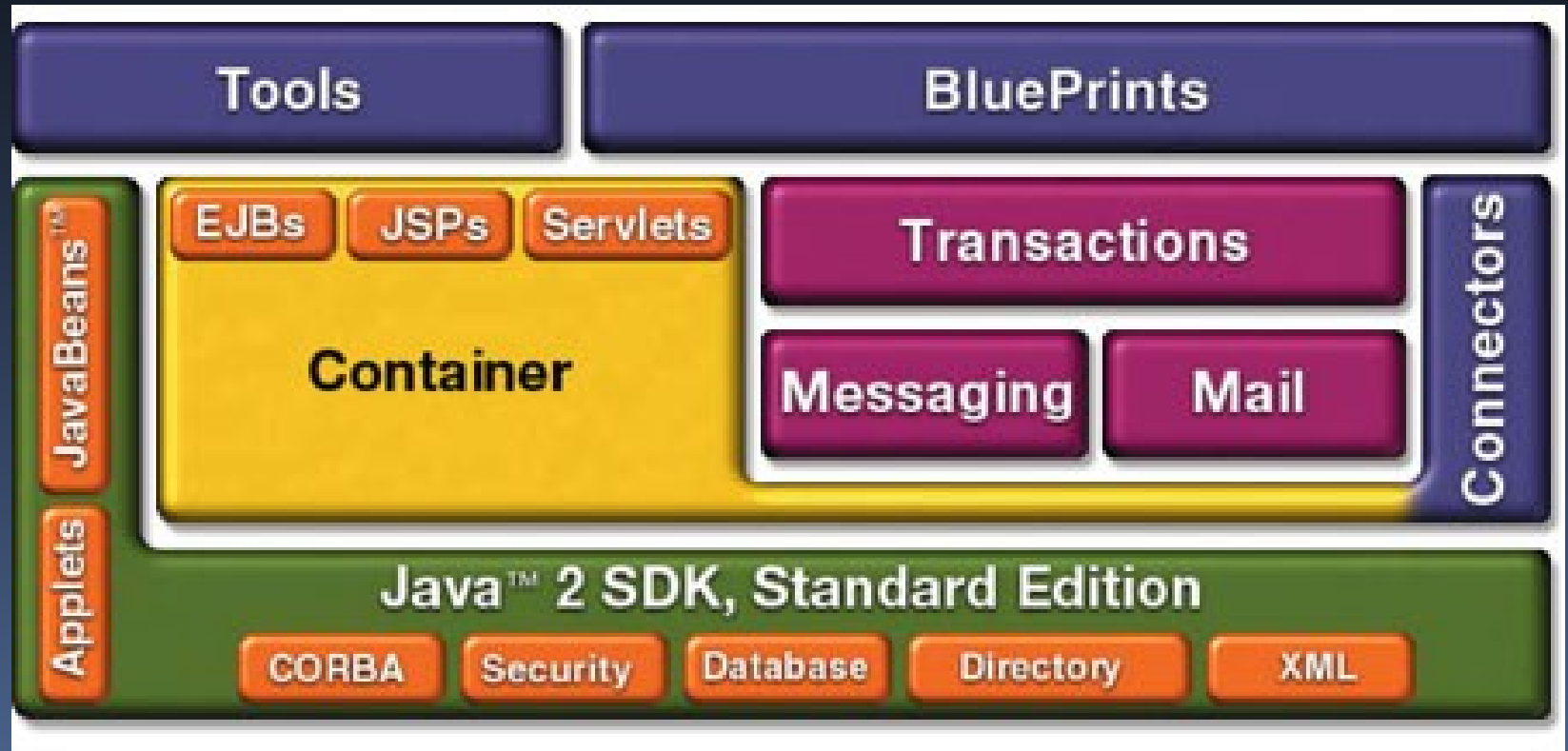




J2EE Technologies

- Java Servlets
 - JSP
 - EJB
 - JMS
 - JDBC
 - JNDI
 - JTA / JTS
 - JavaMail
 - JAAS
 - XML
 - ...
- 

J2EE Componentes





Java Servlets

- A arquitetura de Servlets prove componentes padronizados, são independentes de plataforma para a construção de aplicações WEB sem a limitação de performance dos programas CGI.



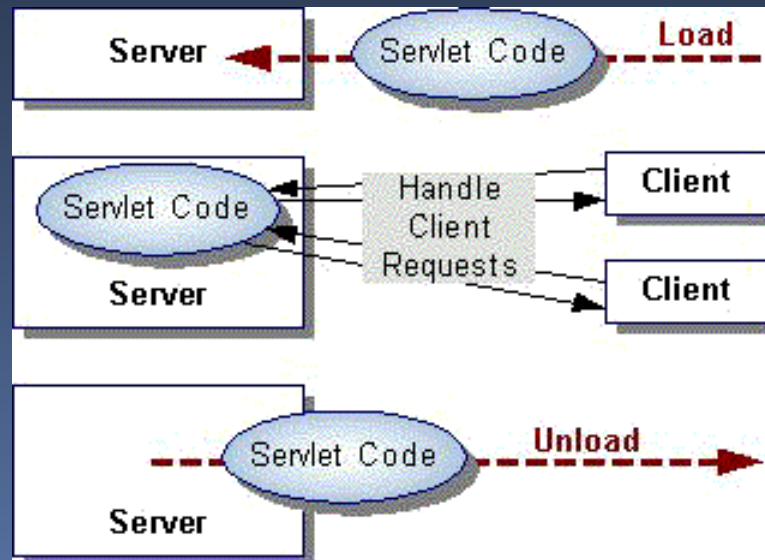
<http://java.sun.com/products/servlets/index.html>

Java Servlets

- Tem acesso a toda a família de API incluindo acesso à JDBCTM API e os bancos de dados;
- Servlets podem acessar API do pacote HTTP para chamar e receber, com todos os benefícios da madura linguagem Java incluindo portabilidade, performance, reusabilidade e robustez;

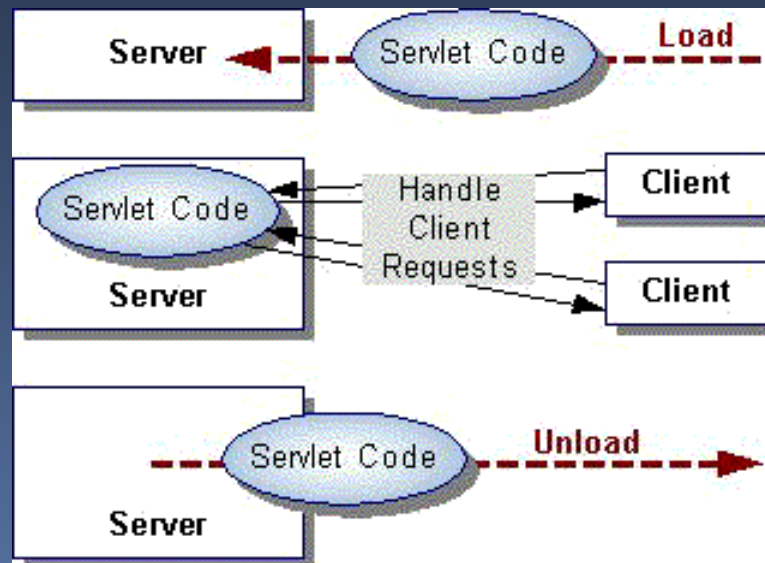
Anatomia dos Servlet

- `init()` – este método é chamado na primeira requisição do mesmo.
- `destroy()` – chamado quando o servlet está sendo destruído pelo container.




Anatomia dos Servlet

- doGet() – chamado pelo método HTTP GET.
- doPost() – chamado pelo método HTTP POST.
 - POSTs a forma ideal para enviar dados dos formulários HTML





Anatomia dos Servlet

- **HttpServletRequest**
 - Informação sobre a requisição HTTP
 - Headers
 - Parâmetros
 - Session
 - Cookies
 - **HttpServletResponse**
 - Formata a resposta HTTP
 - Headers
 - Status codes
 - Cookies
- 

Um Servlet

```
import java.io.*;                                //Apache Tomcat sample code
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter(); out.println("<html>");
        out.println("<body>");
        out.println("<head>");
        out.println("<title>Hello World!</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Hello World!</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

JSP – JavaServer Pages

- Java Server Pages usam tags no estilo do XML-e scriptlets escritos na linguagem de programação JAVA para encapsular a lógica que gera o conteúdo do Website
- Com a separação da lógica do design possibilita a criação de um componente reutilizável

<http://java.sun.com/products/jsp/index.html>

Exemplo de um JSP


```
<html>                                     <!-- Apache Tomcat Samples -->
<!-- Copyright (c) 1999 The Apache Software Foundation. All rights reserved.-->
<body bgcolor="white">
<jsp:useBean id='clock' scope='page' class='dates.JspCalendar' type="dates.JspCalendar" />

<font size=4><ul>
<li>      Day of month: is <jsp:getProperty name="clock" property="dayOfMonth"/>
<li>      Year: is <jsp:getProperty name="clock" property="year"/>
<li>      Month: is <jsp:getProperty name="clock" property="month"/>
<li>      Time: is <jsp:getProperty name="clock" property="time"/>
<li>      Date: is <jsp:getProperty name="clock" property="date"/>
<li>      Day: is <jsp:getProperty name="clock" property="day"/>
<li>      Day Of Year: is <jsp:getProperty name="clock" property="dayOfYear"/>
<li>      Week Of Year: is <jsp:getProperty name="clock" property="weekOfYear"/>
<li>      era: is <jsp:getProperty name="clock" property="era"/>
<li>      DST Offset: is <jsp:getProperty name="clock" property="DSTOffset"/>
<li>      Zone Offset: is <jsp:getProperty name="clock" property="zoneOffset"/>
</ul>
</font>


</body>
</html>
```

EJB – Enterprise Java Beans

- **Enterprise JavaBeans™** componente da arquitetura J2EE™ platform para aplicações no Servidor
- EJB™ possibilita a criação de sistemas distribuídos e robustos, transacionais.



EJB – Enterprise Java Beans

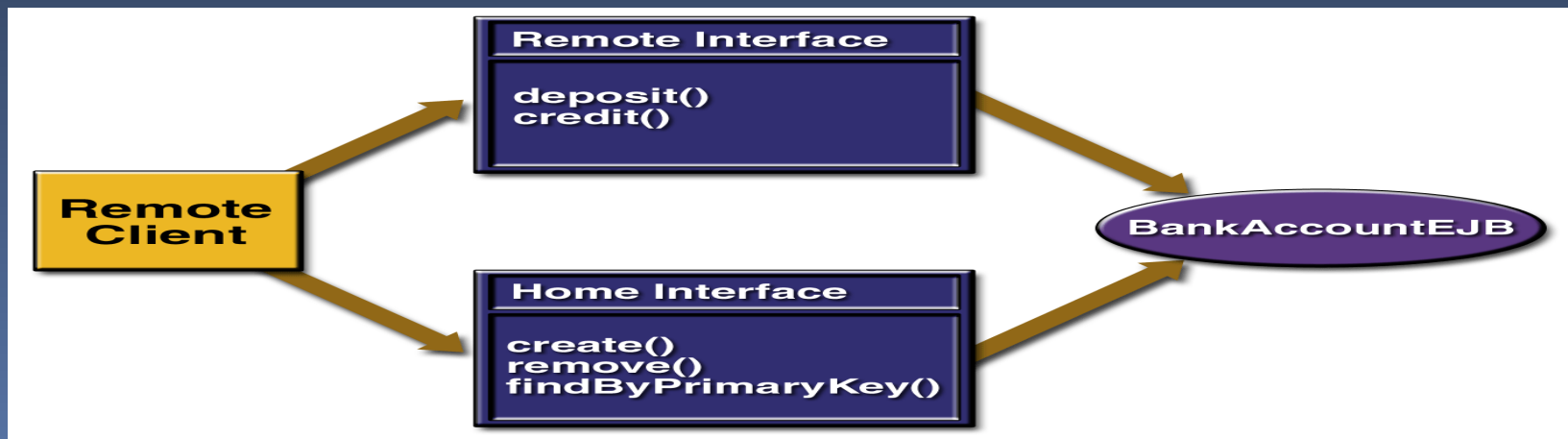
- Enterprise Java Beans são componentes que rodam em no servidor de aplicação
 - O Servidor de aplicações disponibiliza serviços como:
 - Carga / Inicialização
 - Transações
 - Persistencia
 - Comunicação com EJB clients
 - Enterprise Naming Context (JNDI name space)
- 


Anatomia dos EJB

- Interface Remota
 - Métodos que podem ser acessados remotamente.
 - Herda de `javax.ejb.EJBObject`
- Interface Home
 - Métodos do ciclo de vida (`create`, `findByPrimaryKey`)
 - Extends `javax.ejb.EJBHome` que herda de `java.rmi.Remote`
- Bean class
 - Classe que executa o processo de negócios
 - Implementa a interface do tipo do bean

Client / EJB Relacionamento

- Como uma classe cliente (Java class) utiliza EJBs?
 - Lookup - JNDI ENC
 - Protocolo de Rede - RMI
 - Container EJB container cria objetos com a interface RemoteHome e Home – estes objetos passam a chamada para o Business Object






EJB – Enterprise Java Beans

- Entity Beans
- Session Beans
- Message Beans
 - A PARTIR DO EJB 2.0





EJB – Entity Beans

- Classes de entidade representam os objetos persistidos no banco.
 - ▣ Um objeto de entidade representa uma linha da base de dados
 - Possibilitam trabalhar com abstração.
- 

Entity Beans - Persistence

- Container Managed Persistence (CMP)
 - O container automaticamente persiste o objeto de Entidade. Maior produtividade.
- Bean Managed Persistence (BMP)
 - Os objetos de entidade são persistidos pelo desenvolvedor que escolhe a melhor forma de persistencia. Possibilita flexibilidade

EJB – Session Beans

- Session beans executam o trabalho de uma estação de trabalho
 - Ex:Um session bean faz o débito crédito de uma transação de cartão

Session Beans – State

- Stateful – Um bean stateful mantém o seu estado durante transação com um determinado cliente;
- Stateless – Um bean stateless não possui estado interno de uma transação com um determinado cliente. Ex: Servlets

EJB – Session Bean

```
package org.jboss.docs.interest;

import javax.ejb.EJBObject;
import java.rmi.RemoteException;

/** This interface defines the `Remote' interface for the `Interest' EJB. Its single
    method is the only method exposed to the outside world. The class InterestBean
    implements the method. */

public interface Interest extends EJBObject
{
    /** Calculates the compound interest on the sum `principle', with interest rate
    per period `rate' over `periods' time periods. This method also prints a message
    to standard output; this is picked up by the EJB server and logged. In this way
    we can demonstrate that the method is actually being executed on the server,
    rather than the client. */

    public double calculateCompoundInterest(double principle, double rate, double
    periods) throws RemoteException;
}
```

EJB – Session Bean

```
package org.jboss.docs.interest;
import java.io.Serializable;
import java.rmi.RemoteException;
import javax.ejb.CreateException;
import javax.ejb.EJBHome;

/** This interface defines the 'home' interface for the 'Interest' EJB. */
public interface InterestHome extends EJBHome
{
    /** Creates an instance of the `InterestBean' class on the server, and returns a
    remote reference to an Interest interface on the client. */

    Interest create() throws RemoteException, CreateException;
}
```

EJB – Session Bean

```
package org.jboss.docs.interest;
import java.rmi.RemoteException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;
```

```
/** This class contains the implementation for the 'calculateCompoundInterest' method
    exposed by this Bean. It includes empty method bodies for the methods prescribe
    by the SessionBean interface; these don't need to do anything in this simple
    example. */
```

```
public class InterestBean implements SessionBean
{
    public double calculateCompoundInterest(double principle, double rate, double
    periods)
    {
        System.out.println("Someone called `calculateCompoundInterest!\"");
        return principle * Math.pow(1+rate, periods) - principle;
    }
}
```

```
public void ejbCreate() {}
public void ejbPostCreate() {}
public void ejbRemove() {}
public void ejbActivate() {}
public void ejbPassivate() {}
public void setSessionContext(SessionContext sc) {}
```

EJB – Session Bean

```
<?xml version="1.0" encoding="UTF-8"?>

<ejb-jar>
  <description>JBoss Interest Sample Application</description>
  <display-name>Interest EJB</display-name>
  <enterprise-beans>
    <session>
      <ejb-name>Interest</ejb-name>
      <home>org.jboss.docs.interest.InterestHome</home>
      <remote>org.jboss.docs.interest.Interest</remote>
      <ejb-class>org.jboss.docs.interest.InterestBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Bean</transaction-type>
    </session>
  </enterprise-beans>
</ejb-jar>
```

EJB – Session Bean

```
package org.jboss.docs.interest;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;

class InterestClient
{
    /** This method does all the work. It creates an instance of the Interest EJB on the EJB
    server, and calls its `calculateCompoundInterest()' method, then prints the result of
    the calculation. */

    public static void main(String[] args)
    {
        try {
            InitialContext jndiContext = new InitialContext();
            ref = jndiContext.lookup("interest/Interest");
            InterestHome home = (InterestHome) PortableRemoteObject.narrow(ref,
            InterestHome.class);

            Interest interest = home.create(); //Create an Interest object from the Home
            interface

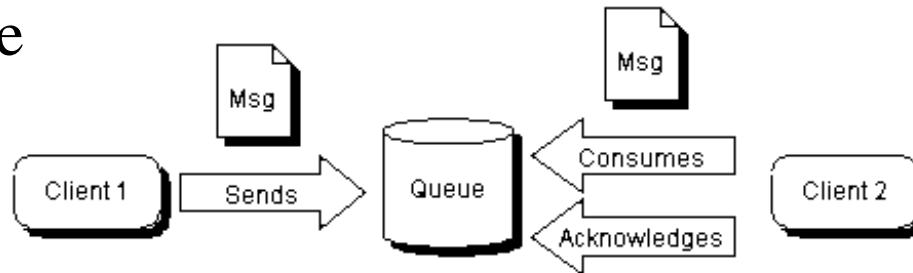
            System.out.println(interest.calculateCompoundInterest(1000, 0.10, 2));
        }
        catch(Exception e)
        {
            System.out.println(e.toString());
        }
    }
}
```

EJB – Message Beans

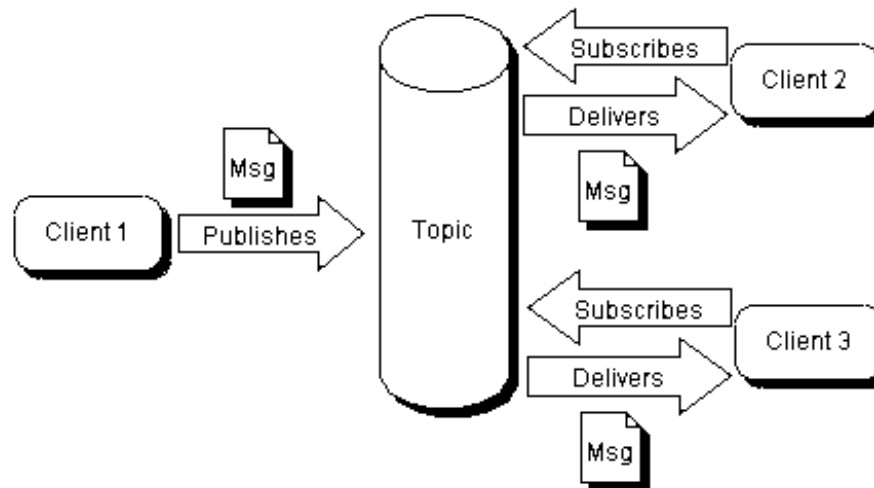
- Message beans são classes que fazem notificações de forma assíncrona utilizando o Java Message Service server

JMS – Java Message Service

JMS Queue



JMS Topic



JMS – Java Message Service

- Porque utilizar o JMS?
 - Sistemas baixamente acoplados;
 - Removes dependence on client and server platform / programming language / version
 - Remove a dependência do cliente e da plataforma do servidor
 - Publish / Subscribe metaphor
 - Send / receive information with many, unknown clients
 - Integration with other messaging systems
 - IBM MQ-Series
 - Microsoft Message Queue

<http://java.sun.com/products/jms/index.html>



JDBC – Data Access API

- JDBC™ technology is an API that lets you access virtually any tabular data source from the Java™ programming language.
 - Cross-DBMS connectivity to a wide range of SQL databases
 - Access to other tabular data sources, such as spreadsheets or flat files.



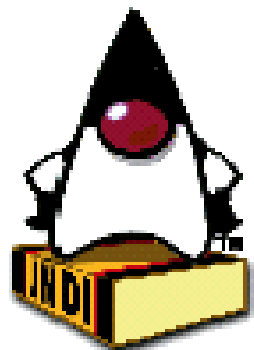
<http://java.sun.com/products/jdbc/index.html>

JDBC – Driver Types

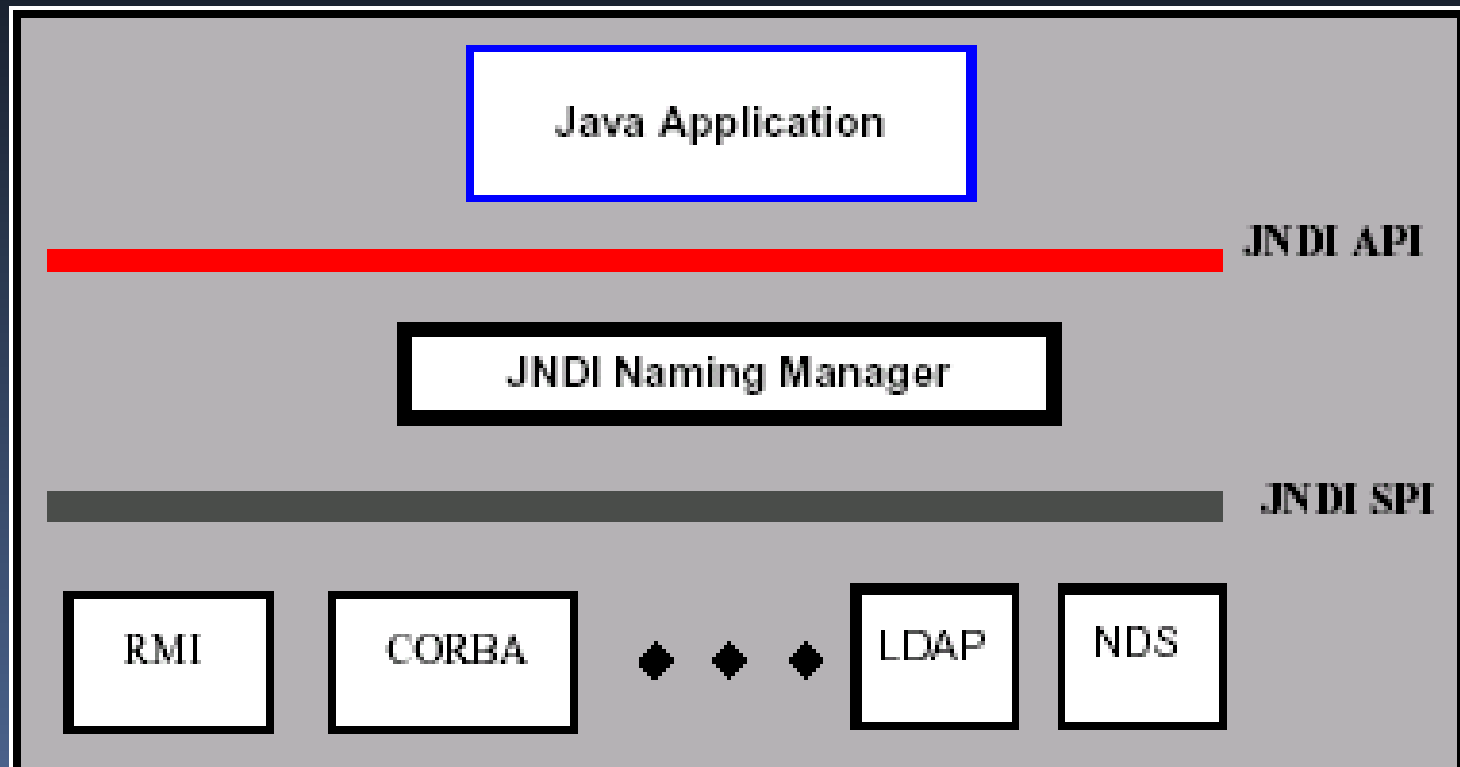
- Level 1 - A *JDBC-ODBC bridge* provides JDBC API access via one or more ODBC drivers.
- Level 2 - A *native-API partly Java technology-enabled driver* converts JDBC calls into calls on the client API for Oracle, Sybase, Informix, DB2, or other DBMS.
- Level 3 - A *net-protocol fully Java technology-enabled driver* translates JDBC API calls into a DBMS-independent net protocol which is then translated to a DBMS protocol by a server.
- Level 4 - A *native-protocol fully Java technology-enabled driver* converts JDBC technology calls into the network protocol used by DBMSs directly.

JNDI – Java Naming and Directory Interface

- JNDI is an API specified in Java™ that provides naming and directory functionality to applications written in Java. It is designed especially for Java by using Java's object model.
- Using JNDI, Java applications can store and retrieve named Java objects of any type.
- JNDI provides methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes.
- JNDI allows Java applications to take advantage of information in a variety of existing naming and directory services, such as LDAP, NDS, DNS, and NIS(YP), and allows Java applications to coexist with legacy applications and systems.



JNDI - Layers



JNDI – Usos comuns

- JNDI ENC – “enterprise naming context”
 - EJB lookup within a J2EE app server
- LDAP integration
- Dynamic registration of services and clients
- Peer to Peer computing

JNDI — Session Bean

```
package org.jboss.docs.interest;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;

class InterestClient
{
    /** This method does all the work. It creates an instance of the Interest EJB on the EJB
    server, and calls its `calculateCompoundInterest()' method, then prints the result of
    the calculation. */

    public static void main(String[] args)
    {
        try {
            InitialContext jndiContext = new InitialContext();
            ref = jndiContext.lookup("interest/Interest");
            InterestHome home = (InterestHome) PortableRemoteObject.narrow(ref,
            InterestHome.class);

            Interest interest = home.create(); //Create an Interest object from the Home
            interface

            System.out.println(interest.calculateCompoundInterest(1000, 0.10, 2));
        }
        catch(Exception e)
        {
            System.out.println(e.toString());
        }
    }
}
```


JTA / JTS – Transactions

- The Java Transaction API (JTA) and the Java Transaction Service (JTS) allow J2EE application servers to take the burden of transaction management off of the component developer.
- Developers can define the transactional properties of Enterprise JavaBeans™ technology based components during design or deployment using declarative statements in the deployment descriptor.
- The application server takes over the transaction management responsibilities.



JavaMail

- The JavaMail™ 1.2 API provides a set of abstract classes that model a mail system.
- The API provides a platform independent and protocol independent framework to build Java technology-based mail and messaging applications.
- J2EE contains JAF – JavaBeans Activation Framework since it is required by JavaMail
- Supports common mail protocols
 - IMAP
 - POP
 - SMTP
 - MIME

JAAS – Java

Authentication and

Authorization Service

Authentication of users, to reliably and securely determine who is currently executing Java code, regardless of whether the code is running as an application, an applet, a bean, or a servlet; and

- *Authorization* of users to ensure they have the access control rights (permissions) required to do the actions performed.
- Sample authentication modules using:
 - Java™ Naming and Directory Interface (JNDI)
 - Unix Operating Environment
 - Windows NT
 - Kerberos
 - Keystore

<http://java.sun.com/products/jaas/index.html>



Java™ Security

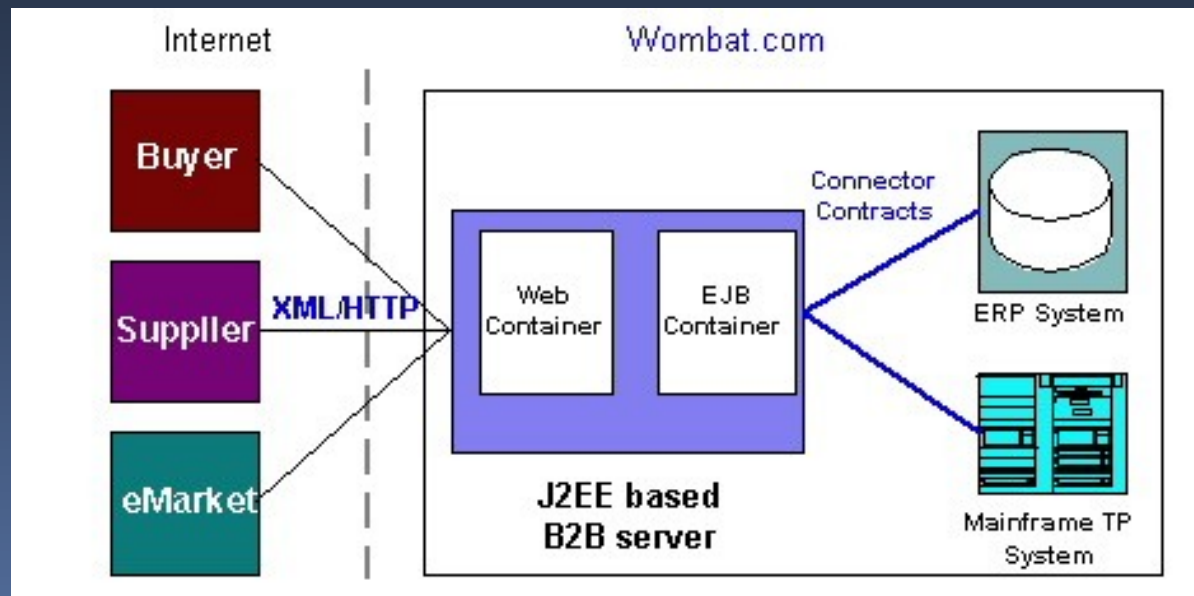


XML

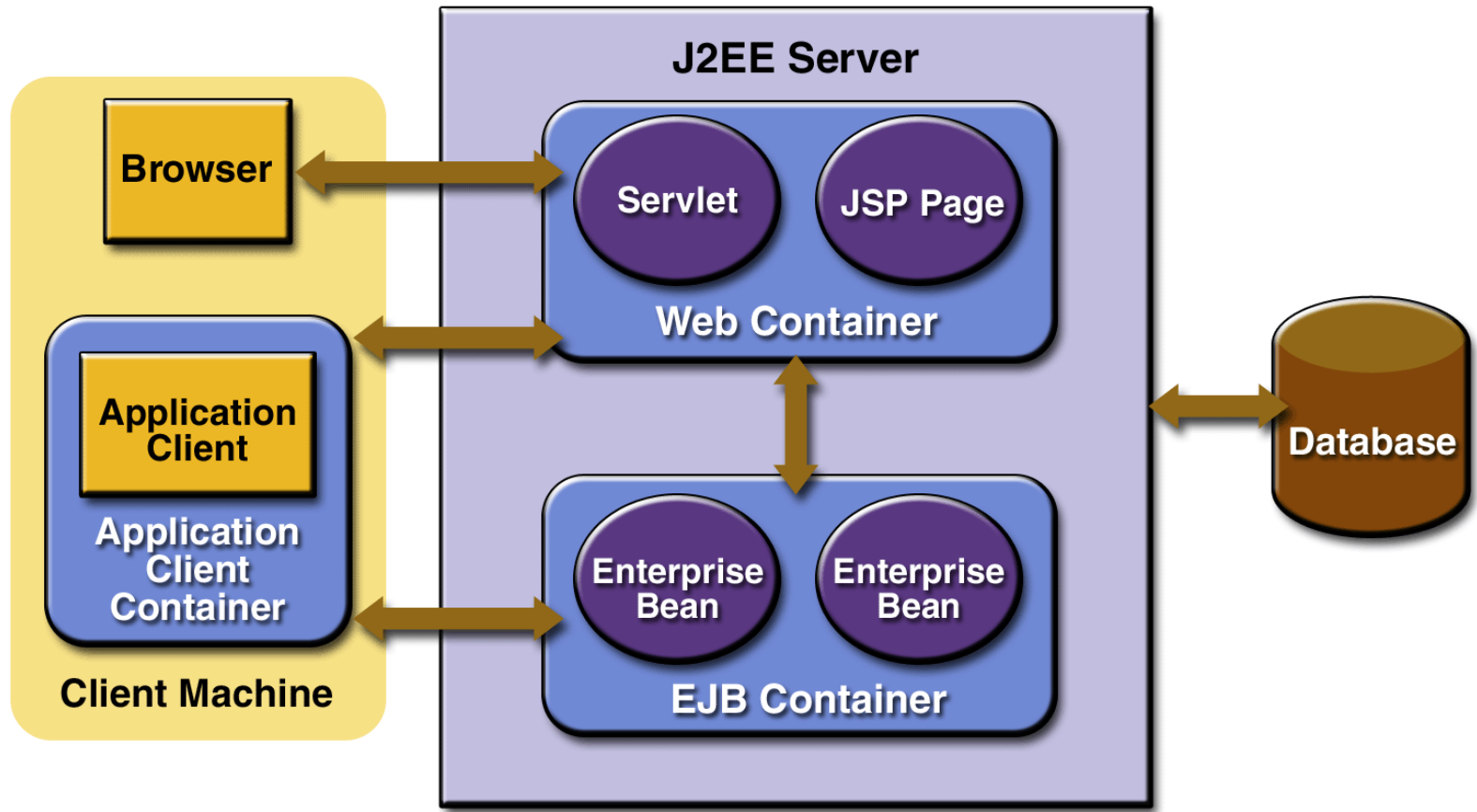
- J2EE 1.3 includes JAXP 1.1 support, as well as Servlet Filters and XML JSP™ documents.
- The Java™ API for XML Processing ("JAXP") supports processing of XML documents using DOM, SAX, and XSLT.
- The portability and extensibility of both XML and Java make them the ideal choice for the flexibility and wide availability requirements of this new web.

J2EE Connectors

- The J2EE Connector architecture defines a standard architecture for connecting the J2EE platform to heterogeneous EISs (Enterprise Information Systems).
- Examples of EISs include ERP, mainframe transaction processing, database systems, and legacy applications not written in the Java programming language.



J2EE – Arquitetura do servidor



J2EE Deployment (implantação)

- JAR – Java ARchive
 - Java class file
 - EJBs
- WAR – Web ARchive
 - Servlets
 - JSPs
- EAR – Enterprise ARchive
 - Possui todos os JARS e WARS de uma aplicação
- Deployment descriptors
 - XML
 - Requerido para EJB JARs, WARs, EARs

J2EE Servers

- Application Server

- BEA Weblogic - 37%
- IBM Websphere - 22%
- Oracle - 11%
- Iplanet - 5%
- Other- 12%

- Open-source

- Jboss - www.jboss.org
- Sun's listing of J2EE compatible servers - <http://java.sun.com/j2ee/compatibility.html>



J2EE Servers

- Servlet / JSP Servers
 - A maioria das aplicações comerciais utilizam Servlets / JSP
 - Open-Source
 - Apache Tomcat
 - Jetty





Conhecendo mais...

- Enterprise JavaBeans – 3rd Edition
 - Richard Monson-Haefel
 - O'Reilly © 2001
 - JBoss documentation
 - <http://www.jboss.org/online-manual/HTML/index.html>
 - Designing Enterprise Applications with the Java 2 Platform, Enterprise Edition
 - Nicholas Kassem and the Enterprise Team
 - Addison Wesley © 2000
 - Core Servlets and JavaServer Pages (JSP)
 - Marty Hall
 - Prentice Hall © 2000
- 

Conhecendo mais...

- J2EE Tutorial - http://java.sun.com/j2ee/tutorial/1_3-fcs
- J2EE Developers Guide -
http://java.sun.com/j2ee/sdk_1.2.1/techdocs/guides/ejb/
- JNDI - <http://java.sun.com/products/jndi/tutorial/>
- JMS - <http://java.sun.com/products/jms/tutorial/>
- JDBC - <http://java.sun.com/docs/books/tutorial/jdbc>
- Servlets -
<http://java.sun.com/docs/books/tutorial/servlets>
- JSP - <http://java.sun.com/products/jsp/docs.html>
- JAXP -
<http://java.sun.com/xml/jaxp/dist/1.1/docs/tutorial>