Faculdade de Tecnologia de Americana
Curso de Processamento de dados

INTEGRAÇÃO DE TECNOLOGIAS UTILIZANDO JAVA

LUÍS FERNANDO GRIM

Americana, SP
2010

Faculdade de Tecnologia de Americana
Curso de Processamento de dados

INTEGRAÇÃO DE TECNOLOGIAS UTILIZANDO JAVA

LUÍS FERNANDO GRIM

luisfernandogrim@yahoo.com.br

Monografia desenvolvida em cumprimento à exigência curricular do Curso de Processamento de Dados, sob orientação da Profº. Antonio Alfredo Lacerda.

Americana, SP
2010

BANCA EXAMINADORA

Profº. Ms. Antonio Alfredo Lacerda (Orientador)

Profº. Ms. José Mário Scafi

Profº. Ms. Wladimir da Costa

AGRADECIMENTOS

Ao meu orientador Antonio Alfredo Lacerda que, desde o início, acreditou em meu trabalho e me incentivou.

À minha família que me ensinou o norte da vida e está o tempo todo torcendo por mim.

À minha amiga Marciene Lopes de Sá, pela companhia, força, amizade e enorme ajuda com os trabalhos.

À minha amiga Greice Mariano, que me deu uma força para desenvolver esta monografia.

A todas as pessoas que me ajudaram direta ou indiretamente em mais este trabalho.

E à Deus pela força de superar os obstáculos e vencer os desafios.

DEDICATÓRIA

Aos professores da FATEC-AM.

A todos os meus amigos.

E à minha família.

RESUMO

A evolução tecnológica dos computadores, sistemas e o crescimento da Internet disponibilizam a cada dia novos recursos e novas técnicas, proporcionando o uso da Computação Distribuída, compartilhando informações entre usuários e sistemas que estejam conectados entre si através destes ambientes. Este trabalho tem como objetivo abordar a tecnologia Java com ênfase em alguns de seus componentes que tornam possível essa integração entre diferentes tecnologias. Será destacado principalmente o uso da Tecnologia Java para desenvolvimento Web com a especificação Java Enterprise Edition, utilizando principalmente Servlets e JSPs (Java Server Pages) para realizar as integrações. Também será abordado o uso do Apache Tomcat e do Eclipse que são algumas das ferramentas essenciais para desenvolver aplicativos em Java.

Palavras Chave: Computação distribuída, Integração e Java.

ABSTRACT

The technological evolution of computers, systems and growth of the Internet offer every day new features and new techniques, allowing the use of Distributed Computing, sharing information between users and systems that are interconnected through these environments. This study aims to address the Java technology with emphasis on some of its components that make possible the integration between different technologies. Will be highlighted mainly the use of Java Technology for Web development with Java Enterprise Edition specification, mainly using Servlets and JSPs (Java Server Pages) to perform the integrations. It also will address the use of Apache Tomcat and Eclipse that are some of the essential tools to develop applications in Java.

Keywords: Distributed Computing, Integration and Java.

SUMÁRIO

LISTA DE FIGURAS E DE TABELAS

INTRODUÇÃO

A partir do exposto a **problemática** a ser estudada consiste em compreender como é possível integrar sistemas de várias tecnologias utilizando a tecnologia Java com a sua implementação voltada para a Web.

O **objetivo geral** foi demonstrar como é possível a integração de diversos sistemas de diferentes linguagens desenvolvendo Servlets e Java Server Pages em JAVA, buscando compreender seu funcionamento.

Como **objetivos específicos**, primeiramente abordar a respeito da computação distribuída; conhecer a respeito da tecnologia Java e sua implementação para Web, e algumas ferramentas que auxiliam no seu desenvolvimento.

O **método científico** de pesquisa utilizado foi bibliográfico em sites acadêmicos, livros, monografias e revistas especializadas.

O presente estudo se **justifica** pela importância que a Tecnologia JAVA exerce hoje em dia. Devido a sua portabilidade os programas baseados em JAVA rodam em praticamente qualquer sistema operacional, sem a necessidade de alterações específicas.

O trabalho foi estruturado em cinco capítulos, sendo que o primeiro conceitua a Computação Distribuída, o segundo fala sobre a Tecnologia Java e o terceiro fala especificamente sobre Java EE e sua implementação na Web com Servlets e JSP.

No quarto capítulo foram mostradas algumas ferramentas que auxiliam no desenvolvimento de aplicações em Java mencionadas.

Com base nas informações conseguidas a partir dos estudos realizados nos capítulos anteriores, o capítulo quinto se reserva às considerações finais.

1 COMPUTAÇÃO DISTRIBUÍDA

Com o advento da computação pessoal, iniciada pela Apple em 1977, os preços dos equipamentos caíram significantemente, ao ponto de tornar vantajosa a compra para uso pessoal, que fez surgir também uma infinidade de programas e de dados funcionando de maneira isolada. Esse panorama começou a mudar quando começaram a surgir as primeiras redes locais. Esses computadores começaram a ser interligados através das redes se comunicando entre si.

A evolução das redes levou ao fenômeno da computação distribuída, em que as informações e os sistemas, em vez de ocorrer isoladamente nas instalações de algum computador central, é distribuída através das redes entre os locais em que o trabalho real da organização é realizado.

1.1 O QUE É UM COMPUTADOR?

O computador é um dispositivo capaz de realizar cálculos e de tomar decisões lógicas a velocidades de milhões e até mesmo bilhões de vezes mais rápidas do que os seres humanos. Por exemplo, muitos dos computadores pessoais de hoje podem realizar centenas de milhões, até bilhões, de somas por segundo. Uma pessoa operando uma calculadora de mesa pode precisar de décadas para completar o mesmo número de cálculos que um computador pessoal poderoso pode realizar em um segundo. (Aspectos a ponderar: como você saberia se a pessoa adicionou os números corretamente? Como você saberia se o computador somou os números corretamente?) Os supercomputadores mais rápidos atualmente podem realizar centenas de bilhões de operações por segundo - cálculos que centenas de milhares de pessoas precisariam de um ano para fazer! E computadores de um trilhão de instruções por segundo já estão funcionando em laboratórios de pesquisa! (DEITEL, 2003).

Os computadores processam dados sob o controle de conjuntos de instruções chamados programas de computador. Esses programas de computador orientam o computador por conjuntos ordenados dc ações especificadas por pessoas chamadas de programadores de computador. (DEITEL, 2003).

Os vários dispositivos que compõem um sistema de computação (como teclado, tela, discos, memória e unidades de processamento) são conhecidos como hardware. Os programas de computador que rodam em um computador são conhecidos como software. Os custos de hardware têm caído significativamente nos últimos anos, a ponto de os computadores pessoais terem-se tomado um bem de consumo popular. Infelizmente, os custos de desenvolvimento de software têm aumentado constantemente, à medida que os programadores desenvolvem aplicativos cada vez mais complexos e poderosos sem serem capazes de aprimorar significativamente a tecnologia de desenvolvimento de software (DEITEL, 2003).

1.2 EVOLUÇÃO DOS SISTEMAS OPERACIONAIS

Os computadores antigos eram capazes de realizar somente um trabalho ou uma tarefa por vez. Essa forma de operação do computador é freqüentemente chamada de processamento em lotes de um único usuário. O computador executa um único programa por vez enquanto está processando dados em grupos ou lotes. Nesses sistemas antigos, geralmente os usuários submetiam seus trabalhos ao centro de processamento de dados em maços de cartões perfurados. Os usuários freqüentemente tinham de esperar horas ou mesmo dias antes de as impressões serem retornadas para suas mesas (DEITEL, 2003).

Os sistemas de software chamados sistemas operacionais foram desenvolvidos com o intuito de ajudar a tornar mais conveniente o uso dos computadores. Os sistemas operacionais anteriores gerenciavam a transição suave entre os trabalhos. Isso minimizou o tempo necessário para que os operadores de computador alternassem entre trabalhos e, portanto, aumentou a quantidade de trabalho, ou *throughput*, que os computadores poderiam processar (DEITEL, 2003).

Quando os computadores se tornaram mais poderosos, tomou-se evidente que o processamento em lotes de um único usuário raramente utilizava os recursos de computador de maneira eficiente. Em vez disso, imaginou-se que muitos trabalhos ou tarefas poderiam compartilhar os recursos do computador para obterem melhor utilização. Isso é chamado de multiprogramação. A multiprogramação envolve a operação “simultânea” de muitos trabalhos no computador — o computador compartilha seus recursos entre os trabalhos que competem por sua

atenção. Com os sistemas operacionais anteriores de multiprogramação, os usuários ainda submetiam trabalhos em cartões perfurados e esperavam horas ou dias pelos resultados (DEITEL, 2003).

Na década de 60, vários grupos na indústria e nas universidades foram os pioneiros dos sistemas operacionais de compartilhamento de tempo. O compartilhamento de tempo é um caso especial da multiprogramação em que os usuários acessam o computador através de terminais, em geral dispositivos com teclados e telas. Em um típico sistema de computador baseado em compartilhamento de tempo, pode haver dezenas ou mesmo centenas de usuários compartilhando o computador de cada vez. O computador não executa de fato todos os trabalhos dos usuários simultaneamente. Em vez disso, ele executa uma pequena parte de trabalho de um usuário e segue para atender o próximo usuário. O computador faz isso tão rapidamente que pode fornecer um serviço para cada usuário várias vezes por segundo. Assim, os programas dos usuários parecem estar rodando simultaneamente. Uma vantagem do compartilhamento dc tempo é que o usuário recebe respostas quase imediatas às solicitações, em vez de ter de esperar muito tempo pelos resultados, como ocorria com os modos de computação anteriores. Da mesma forma, se um usuário particular estiver atualmente desocupado, o computador pode continuar a servir outros usuários em vez de esperar por um usuário específico (DEITEL, 2003).

1.3 COMPUTAÇÃO PESSOAL DISTRIBUÍDA

Em 1977, a Apple Computer popularizou o fenômeno da computação pessoal. Inicialmente, era um sonho de aficcionados. Os computadores tornaram-se suficientemente econômicos para as pessoas comprá-los para seu próprio uso pessoal. Em 1981, a IBM, o maior fornecedor de computadores do mundo, lançou o IBM Personal Computer. Quase da noite para o dia a computação pessoal tornou-se uma realidade no comércio, na indústria e nos órgãos do governo (DEITEL, 2003).

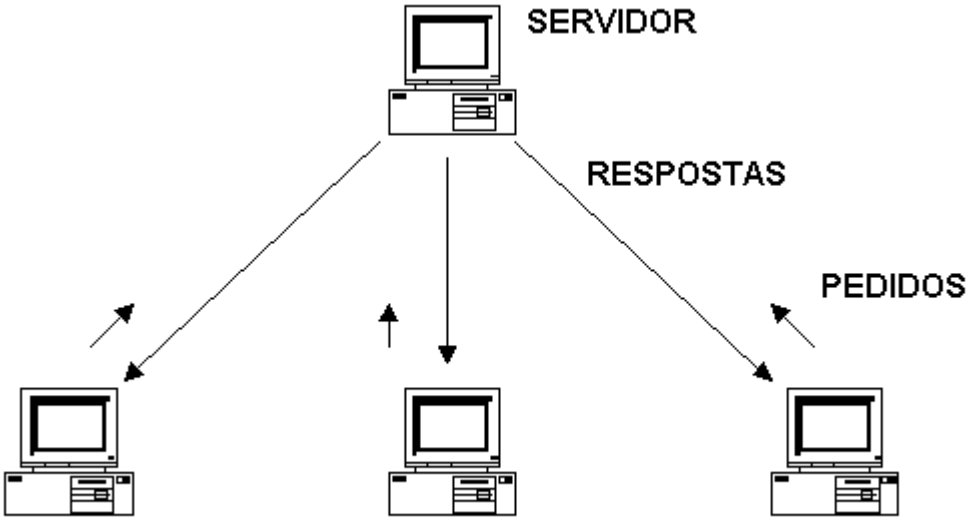
Mas esses computadores eram unidades “independentes” (stand-alone) — as pessoas faziam seus trabalhos em suas próprias máquinas e então transportavam discos de um lado para o outro para compartilhar informações. Embora os computadores pessoais antigos não fossem suficientemente poderosos para

compartilhamento de tempo de vários usuários, essas máquinas podiam ser interconectadas em redes de computadores, às vezes por linhas telefônicas e às vezes em redes locais (LAN — local area network) dentro de uma organização. Isso levou ao fenômeno da computação distribuída, em que a computação de uma organização, em vez de ocorrer estritamente nas instalações de algum computador central, é distribuída através das redes entre os locais em que o trabalho real da organização é realizado. Os computadores pessoais eram suficientemente poderosos tanto para tratar as necessidades de computação dos usuários como tratar as tarefas básicas de comunicação na transmissão de informações de um lado para o outro eletronicamente (DEITEL, 2003).

1.3.1 COMPUTAÇÃO CLIENTE/SERVIDOR

Os computadores pessoais atuais são tão poderosos quanto ás máquinas de milhões de dólares de apenas uma década atrás. As máquinas *desktop* mais poderosas — chamadas estações de trabalho — fornecem enormes capacidades aos usuários individuais. As informações são facilmente compartilhadas através das redes de computadores nas quais alguns computadores, chamados de servidores de arquivos, oferecem um armazenamento comum de programas e dados que pode ser utilizado por computadores clientes distribuídos por toda a rede (DEITEL, 2003), conforme mostra a Figura 1, daí o termo computação cliente/servidor.

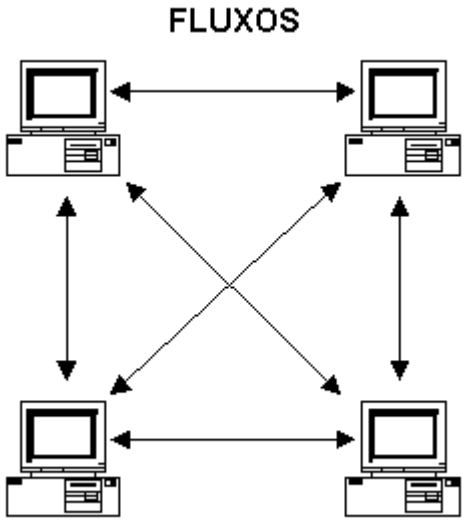
Figura1: Modelo de computação Cliente/Servidor (SAUVÉ, 2010).



1.3.2 COMPUTAÇÃO P2P E OBJETOS DISTRIBUÍDOS

Também temos outros dois modelos de computação distribuída, conhecidos como P2P e Objetos Distribuídos. Ambos é uma arquitetura caracterizada pela descentralização das funções na rede, onde cada computador realiza tanto funções de servidor quanto de cliente. A diferença entre eles é que o modelo de Objetos Distribuídos possui um Middleware (mediador) intermediando o processo de comunicação. Abaixo, na Figura 2, um exemplo de computação P2P:

Figura 2: Modelo de computação Peer to Peer (P2P) (SAUVÉ, 2010).



C e C++ tomaram-se e continuam sendo as linguagens de programação preferidas para escrever sistemas operacionais. Elas também continuam populares para escrever aplicações para redes de computadores, aplicações distribuídas cliente/servidor e aplicações para a Internet e a Web, embora Java seja agora a linguagem predominante em cada urna destas áreas. Muitos programadores descobriram que a programação em Java os ajuda a serem mais produtivos que a programação em C ou C++ (DEITEL, 2003).

1.4 ORGANIZAÇÃO DA COMPUTAÇÃO DISTRIBUÍDA

Organizar a interação entre cada computador é primordial. Visando poder usar o maior número possível de máquinas e tipos de computadores, o protocolo ou canal de comunicação não pode conter ou usar nenhuma informação que possa não ser entendida por certas máquinas. Cuidados especiais também devem ser tomados

para que as mensagens sejam entregues corretamente e que as mensagens inválidas sejam rejeitadas, caso contrário, levaria o sistema a cair ou até o resto da rede.

Outro fator de importância é a habilidade de mandar softwares para outros computadores de uma maneira portável de tal forma que ele possa executar e interagir com a rede existente. Isso pode não ser possível ou prático quando usando hardware e recursos diferentes, onde cada caso deve ser tratado separadamente com *cross-compiling* ou reescrevendo software.

1.5 SISTEMAS DISTRIBUÍDOS

A computação distribuída consiste em adicionar o poder computacional de diversos computadores interligados por uma rede de computadores ou mais de um processador trabalhando em conjunto no mesmo computador, para processar colaborativamente determinada tarefa de forma coerente e transparente, ou seja, como se apenas um único e centralizado computador estivesse executando a tarefa. A união desses diversos computadores com o objetivo de compartilhar a execução de tarefas é conhecida como sistema distribuído.

Para Tanenbaum (2007) um sistema distribuído é:

> "uma coleção de computadores independentes que se apresenta ao usuário como um sistema único e consistente"

1.6 A INTERNET E A WORLD WIDE WEB

A Internet foi desenvolvida há mais de três décadas, financiada pelo Departamento da Defesa dos EUA. Originalmente projetada para conectar os principais sistemas de computadores de cerca de uma dezena de universidades e organizações de pesquisa, a Internet atualmente é acessível a centenas de milhões de computadores no mundo (DEITEL, 2003).

Com a introdução da World Wide Web — que permite que os usuários de computador localizem e visualizem documentos baseados em multimídia sobre

quase todos os assuntos — a Internet explodiu, para se tornar um dos principais mecanismos de comunicação do mundo (DEITEL, 2003).

Nesse milênio, a Internet e a World Wide Web seguramente estarão listadas entre as mais profundas e importantes criações da raça humana. Antigamente, a maioria dos aplicativos de computador rodava em computadores que não se comunicavam entre si. Atualmente, podem-se escrever aplicativos que se comunicam com as centenas de milhões de computadores do mundo. A Internet funde tecnologias de computação e de comunicações. Ela toma nosso trabalho mais fácil. Torna informações acessíveis mundialmente de forma instantânea e conveniente. Possibilita que indivíduos e empresas locais de pequeno porte tenham uma exposição mundial. Ela está alterando a forma como o comércio é feito. As pessoas podem procurar os melhores preços de praticamente todos os produtos ou serviços. Comunidades de interesse especial podem permanecer em contato entre si. Pesquisadores podem se informar instantaneamente sobre os últimos avanços em qualquer parte do globo (DEITEL, 2003).

O próximo capítulo fala sobre a tecnologia Java que apresenta técnicas de programação que permitem que os aplicativos em Java utilizem a lnternet e a World Wide Web para interagir com outros aplicativos. Esses recursos permitem que os programadores em Java desenvolvam os tipos de aplicativos distribuídos de nível corporativo que são usados atualmente nas empresas. Pode-se escrever aplicativos Java em qualquer plataforma de computador, o que resulta em uma importante economia de tempo e custo de desenvolvimento de sistemas para as corporações.

2 A TECNOLOGIA JAVA

> “Nada se torna real até ser experimentado, mesmo um provérbio não significa nada para você até sua vida poder ilustrá-lo.”
>
> John Keats (1795 - 1821)

Conforme foi abordado no capítulo anterior, o desenvolvimento das redes locais e posteriormente da Internet fez com que os computadores, que antes funcionavam de forma isolada, passassem a compartilhar informações.

O desafio é como compartilhar e integrar essas informações já que existem muitas tecnologias diferentes e cada uma delas possuem certos requisitos para o seu funcionamento adequado. Por exemplo, uma aplicação em baseada na plataforma ASP (Active Server Pages) precisa de um ambiente Windows para ser interpretada. Para tanto, é preciso que as equipes de desenvolvimento de software decidam sobre qual plataforma e quais tecnologias serão usadas, e então definam uma arquitetura de software.

É aí que se destaca a tecnologia Java, principalmente por sua portabilidade, ou seja, você ganha independência de sistema operacional. Ou, melhor ainda, independência de plataforma em geral: não é preciso se preocupar em qual sistema operacional sua aplicação está rodando, nem em que tipo de máquina, configurações, etc.

2.1 HISTÓRIA DA LINGUAGEM JAVA

Em 1991, a empresa Sun Microsystems, prevendo um rápido desenvolvimento no mercado de dispositivos eletrônicos inteligentes, financiou um projeto de pesquisa corporativa interna com o codinome Green. Esse projeto resultou no desenvolvimento de uma linguagem baseada em C++, batizada de Oak por seu criador James Gosling, que a chamou assim em homenagem a uma árvore de carvalho que Gosling visualizava pela janela de sua sala na Sun. Entretanto, descobriu-se mais tarde que já existia uma linguagem denominada Oak. JAVA, então, foi sugerido em homenagem a uma cafeteria local que a equipe da Sun gostava muito de ir. Originalmente, JAVA é o nome de uma cidade de origem de um tipo de café importado (DEITEL, 2003).

O projeto Green passou por algumas dificuldades. O mercado de dispositivos eletrônicos inteligentes voltados para o consumo popular não desenvolveu tão rápido quanto a Sun havia previsto. Por isso, o projeto correu o risco de ser cancelado. Por uma feliz casualidade, a World Wide Web explodiu em popularidade em 1993 (HUDSON; PATEL; THOMAS, 1997) e a equipe da Sun percebeu imediatamente o potencial de utilizar o Java para adicionar conteúdo dinâmico, como interatividade e animações, às páginas da Web, que até então não tinham interação nenhuma de componentes e eram totalmente estáticas (DEITEL, 2003). Isso deu nova idéia ao projeto

Em maio de 1995, o Java foi lançado formalmente pela empresa, em uma importante conferência, e chamou a atenção da comunidade de negócios, devido ao interesse na World Wide Web (HUDSON; PATEL; THOMAS, 1997).

O Java transformou-se, assim, em uma plataforma de desenvolvimento, com importantes bibliotecas e recursos que são utilizados no desenvolvimento de aplicativos corporativos de grande porte, aprimoramento da funcionalidade Web, fornecimento de aplicativos para dispositivos voltados para o consumo popular, como telefones celulares, pagers e Personal Digital Assistants (PDAs), entre outros propósitos (MARIANO; 2008).

2.2 BIBLIOTECAS DE CLASSES JAVA

Os programas em Java consistem em partes chamadas classes. Estas consistem em partes chamadas métodos que realizam tarefas e retornam as informações ao completarem suas tarefas. Você pode programar cada pedaço que talvez você precise para formar um programa em Java. Entretanto, a maioria dos programadores em Java tira proveito de ricas coleções de classes existentes em bibliotecas de classes Java. As bibliotecas de classes são também conhecidas como Java APIs (Applications Programming Interfaces — interfaces de programas aplicativos). Portanto, há realmente duas partes para aprender o “mundo” de Java. A primeira é aprender a própria linguagem Java de modo que você possa programar suas próprias classes e a segunda é aprender como utilizar as classes e as extensas bibliotecas de classes Java. As bibliotecas de classes são fornecidas principalmente por fornecedores de compiladores, mas muitas bibliotecas de classes

são fornecidas por fornecedores independentes de software (ISV independent software vendor). Da mesma forma, muitas bibliotecas de classes estão disponíveis a partir da Internet e da World Wide Web como freeware ou shareware (DEITEL, 2003).

2.3 PRINCIPAIS CARATECRÍSTICAS

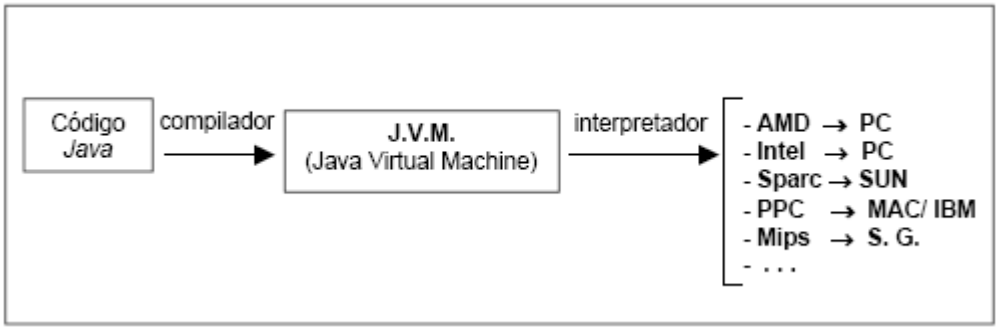
A linguagem Java e seu ambiente foram projetados para resolver problemas que ocorrem na prática de programação, para isso possui características que, em conjunto, diferenciam-na de outras linguagens. Foi projetada tendo em vista os seguintes objetivos (SILVEIRA, 2008):

• Simplicidade: a linguagem foi desenvolvida pensando no tempo de aprendizagem para construir um sistema de maneira fácil, mas que prestigiasse as práticas atuais de desenvolvimento. Essa simplicidade faz com que os softwares possam ser construídos para rodar em máquinas pequenas.

• Orientação a objeto: baseado no modelo de Smalltalk e Simula67. Essa característica é muito importante, pois um projeto orientado a objetos facilita a clara definição de interfaces e toma possível o reuso de código.

• Portabilidade: um software desenvolvido nesta linguagem possui independência de plataforma (ambientes operacionais), conforme exemplifica a Figura 3. Isso se deve, principalmente, ao fato de o código ser compilado para um processador virtual, conhecido como Java Virtual Machine (JVM) e transformado em uma seqüência de instruções chamadas bytecodes. Essas instruções são interpretadas para o processador real da máquina. Por isso essa é outra característica muito importante da linguagem.

Figura 3: Exemplo de interpretação e portabilidade de um código Java (CBPF, 2008).



• Recursos de Rede: possui extensa biblioteca de rotinas que facilitam a cooperação com protocolos TCP/IP, como HTTP e FTP, assim as aplicações Java podem abrir e acessar objetos através da rede via endereços de páginas da lnternet com as mesmas facilidades de se acessar um sistema de arquivo local

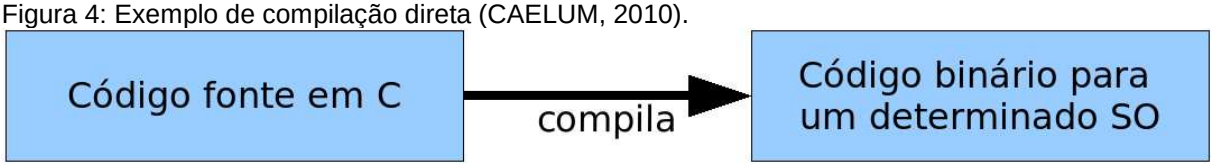
• Segurança: pode-se executar programas via rede com restrições de execução. As transferências de dados realizadas em Java incluem verificação de código de byte, o que evita acréscimo de vírus e cavalos de Tróia (essa verificação implica que, se o tamanho do pacote é alterado no caminho, a transferência será abortada). Também suporta padrões de criptografia de chave pública do algoritmo RSA, possibilitando a proteção de privacidade

Todas essas características têm feito com que o mercado use a linguagem Java nos mais diferentes tipos de soluções, ganhando-se tempo e garantindo desempenho às aplicações.

2.4 MÁQUINA VIRTUAL – JAVA VIRTUAL MACHINE

Em uma linguagem de programação como C ou Pascal, temos o seguinte quadro quando vamos compilar um programa:

Figura 4: Exemplo de compilação direta (CAELUM, 2010).



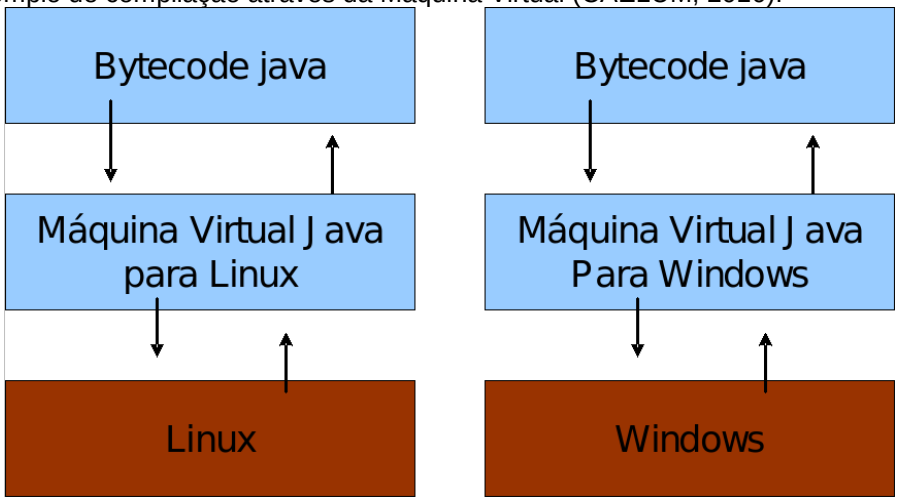
O código fonte é compilado para uma plataforma e sistema operacional específicos. Muitas vezes, o próprio código fonte é desenvolvido visando uma única plataforma. Esse código executável (binário) resultante será executado pelo sistema operacional e, por esse motivo, ele deve saber conversar com o sistema operacional em questão. Isto é, temos um código executável para cada sistema operacional. É necessário compilar uma vez para Windows, outra para o Unix, e assim por diante.

Na maioria das vezes, uma aplicação se utiliza das bibliotecas do sistema operacional, como, por exemplo, a de interface gráfica para desenhar as ‘telinhas’. A biblioteca de interface gráfica do Windows é bem diferente das do Unix. O resultado

é que se torna preciso reescrever o mesmo pedaço da aplicação para diferentes sistemas operacionais, já que eles não são compatíveis.

Já a tecnologia Java se utiliza do conceito de máquina virtual, onde existe, entre o sistema operacional e a aplicação uma camada extra, responsável por 'traduzir o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento:

Figura 5: Exemplo de compilação através da Máquina Virtual (CAELUM, 2010).



Dessa forma, a maneira com a qual se abre uma janela no Linux ou no Windows é a mesma: você ganha independência de sistema operacional. Ou, melhor ainda, independência de plataforma em geral: não é preciso se preocupar em qual sistema operacional sua aplicação está rodando, nem em que tipo de máquina, configurações etc.

Uma máquina virtual é um conceito bem mais amplo que o de um interpretador. Como o próprio nome diz, uma máquina virtual é como um computador de mentira: tem tudo que um computador tem. Em outras palavras, ela é responsável por gerenciar memória e *threads*, a pilha de execução, etc.

Sua aplicação roda sem nenhum envolvimento com o sistema operacional, sempre conversando apenas com a Java Virtual Machine (JVM). Essa característica é interessante: como tudo passa pela JVM, ela pode tirar métricas, decidir onde é melhor alocar a memória, entre outros. Uma JVM isola totalmente a aplicação do sistema operacional. Se uma JVM termina abruptamente só as aplicações que

estavam rodando nela irão terminar: isso não afetará outras JVMs que estejam rodando no mesmo computador, nem afetará o sistema operacional.

Essa camada de isolamento também é interessante quando pensamos em um servidor que não pode se sujeitar a rodar código que possa interferir na boa execução de outras aplicações. Para tudo isso precisamos de um “bytecode”. Bytecode é o termo dado ao código binário gerado pelo compilador Java (pois existem menos de 256 códigos de operação dessa linguagem, e cada “opcode” gasta um byte, dando origem ao nome bytecode). O compilador Java gera esse bytecode que, diferente das linguagens sem máquina virtual, vai servir para diferentes sistemas operacionais, já que ele vai ser traduzido” pela máquina virtual.

“Write once, run anywhere”, Esse é um slogan que a Sun usa para o Java, já que você não precisa reescrever parte da sua aplicação toda vez que quiser mudar de sistema operacional.

2.5 VANTAGENS E DESVANTAGENS DA LINGUAGEM

Para a CBPF (2008) as principais vantagens apresentadas são:

• Sintaxe similar a Linguagem C/C++, que até o lançamento de Java era a linguagem orientada a objetos mais utilizada.

• Facilidade para internacionalização do software, pois suporta nativamente caracteres Unicode.

• Simplicidade na especificação, tanto da linguagem como do “ambiente” de execução (JVM).

• Distribuição com um vasto conjunto de bibliotecas, conhecido também como conjunto de APIs.

• Facilidades para criação de programas distribuídos e multitarefa (múltiplas linhas de execução em um mesmo programa).

• Desalocação automática de memória por processo coletor de lixo (garbage collector)

• Coleção de classes armazenadas independentemente que podem ser carregadas no momento da utilização, dentro de um programa (Carga Dinâmica de Código).

As desvantagens propiciadas pela linguagem estão no fato de que, por Java ser uma linguagem interpretada por uma máquina virtual, a execução da aplicação fica um pouco mais lenta que em uma linguagem compilada para a arquitetura de destino. Mas isso pode ser resolvido com compiladores especiais *Just in Time*, e que infelizmente fazem com que o programa perca a habilidade de ser executado em várias plataformas. Outra desvantagem está em sistemas que utilizam a biblioteca gráfica da linguagem (AWT), pois ela reage de uma forma diferente em cada plataforma (CBPF, 2008).

Entretanto, essas desvantagens não implicam no mau funcionamento de um programa, e por isso não atrapalham a popularização da linguagem entre os programadores e as empresas de desenvolvimento tecnológico.

2.6 PLATAFORMA JAVA

Plataforma Java é o nome dado ao ambiente computacional desenvolvido pela empresa americana Sun Microsystems. Essa plataforma permite desenvolver aplicativos utilizando qualquer uma das linguagens criadas para a plataforma Java, sendo que a linguagem padrão é a que leva seu próprio nome: Linguagem Java (MARIANO, 2008). Uma grande vantagem da plataforma é a de não estar presa a um único sistema operacional ou hardware, pois seus programas rodam através de uma máquina virtual que pode ser emulada em qualquer sistema que suporte a linguagem C++.

O universo Java é um vasto conjunto de tecnologias, composto por três plataformas principais que foram criadas para segmentos específicos de aplicações. (MARIANO, 2008):

• Java SE (Java Platform Standard Edition) — edição que inclui o ambiente de execução e as bibliotecas comuns; além dessas, outras bibliotecas podem ser baixadas opcionalmente pelo site da Sun. É a tecnologia para computadores pessoais, notebooks e arquiteturas com poder de processamento e memória

consideráveis. Possui duas divisões: o Java Development Kit (JDK) ou Standard Development Kit (SDK), que é um pacote de software instalado pelos desenvolvedores para prover serviços de gravação, compilação, depuração, execução de applets e aplicativos Java; e o Java Runtime Edition (JRE) que é uma versão mais leve do JDK, preparada para executar os sistemas construídos com o SDK.

• Java EE (Java Platform Enterprise Edition) - edição voltada para o desenvolvimento de aplicações corporativas que podem estar ou não na Internet. É uma tecnologia preocupada com a segurança, por isso é utilizada para a construção de servidores de aplicação, integração de sistemas e distribuição de serviços para terceiros. Determina especificações que auxiliam no desenvolvimento dessas aplicações, facilitando a padronização e a simplificação de código.

• Java ME (Java Platform, Micro Edition) - edição para o desenvolvimento de aplicações para dispositivos móveis e embarcados. Contém bibliotecas bem simples e leves para economizar espaço, memória e processamento. É subdividida em dois grupos de bibliotecas: o Connected Limited Device Configuration (CLDC), utilizado para o desenvolvimento de aplicativos para celulares e smartphones, por serem esses mais limitados, e o Connected Device Configuration (CDC) utilizado para o desenvolvimento de aplicativos para Palmtops, Pockets Pcs e outros dispositivos mais poderosos.

Há ainda outras duas plataformas Java mais específicas, entretanto um pouco menos utilizadas:

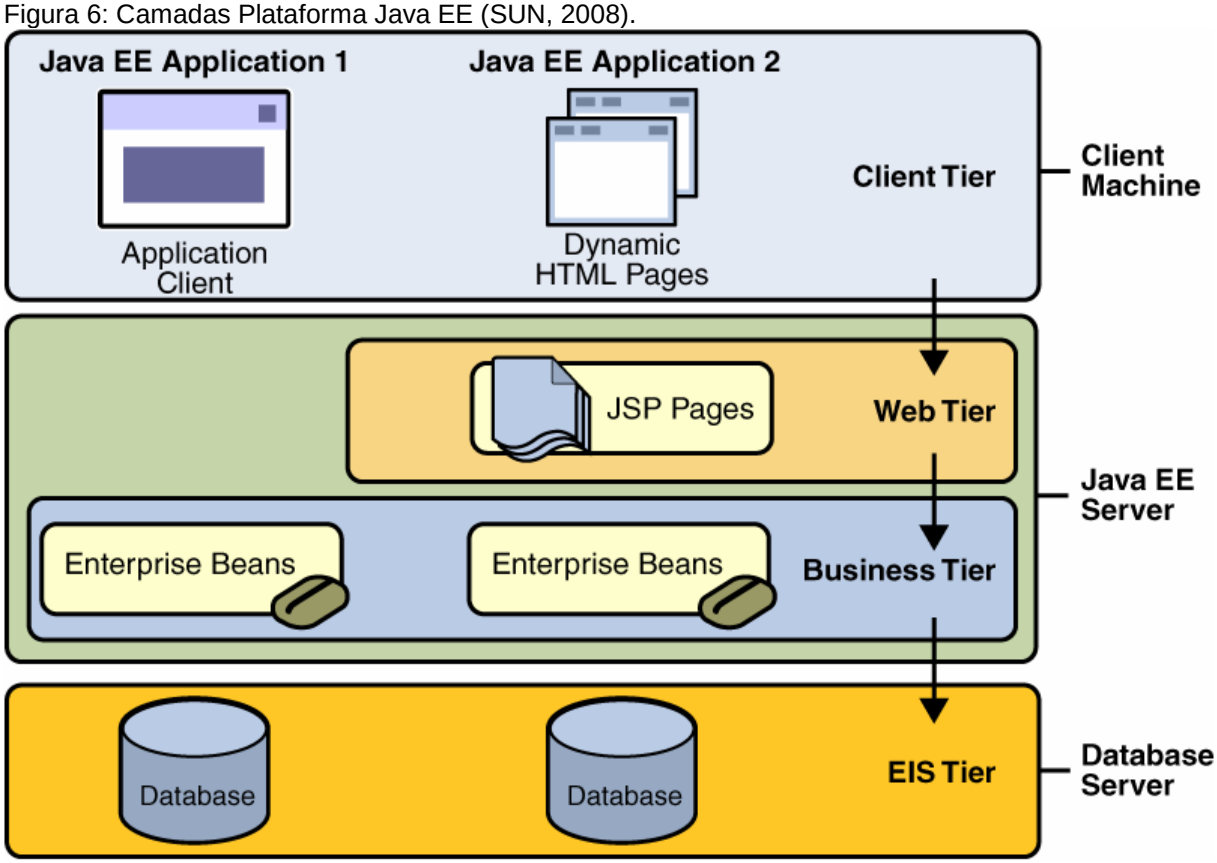
• Java Card - voltada para dispositivos embarcados com limitações de processamento e armazenamento como smart cards.

• JavaFX - plataforma para desenvolvimento de aplicações multimídia em destopM’eb (JavaFX Script) e dispositivos móveis (JavaFX Mobile)

3 JAVA EE

Java EE (Java Enterprise Edition) ou J2EE (Java 2 Enterprise Edition) é uma plataforma de programação de computadores, e que como citado, compõe a plataforma Java. Ela é direcionada para desenvolvimento de aplicações multicamadas, baseadas em componentes que são executados em um servidor de aplicações (arquitetura cliente/servidor) (MARIANO, 2008), conforme apresenta a Figura 6:

Figura 6: Camadas Plataforma Java EE (SUN, 2008).



A arquitetura, neste contexto, permite a distribuição e divisão das aplicações em camadas. Conforme apresentado pela Figura 6, as aplicações direcionadas a interface do usuário (tanto aplicações desktop, quanto páginas dinâmicas para aplicações web) ficam no lado cliente, enquanto que no lado servidor ficam os componentes e a lógica de negócios da aplicação, responsáveis pelo atendimento das solicitações de clientes, bem como acesso à base de dados.

Essa plataforma é considerada um padrão de desenvolvimento já que é preciso seguir certas regras para se declarar um software como compatível com

Java EE. Além disso, ela possui bibliotecas desenvolvidas para o acesso à base de dados. E devido a essas características é utilizada principalmente para o desenvolvimento de aplicações corporativas (MARIANO, 2008).

Java EE possui uma série de especificações, cada uma com funcionalidades distintas. Destacam-se (ALUR, CRUPI; MALKS, 2004):

• JDBC (Java Database Connectivity) — conjunto de classes utilizado para acessar e recuperar informações dos bancos de dados.

• Servlets – utilizados para o desenvolvimento de aplicações Web com conteúdo dinâmico. Possuem uma API que abstrai e disponibiliza os recursos do servidor Web de maneira simplificada para o programador.

• JSP (Java Server Pages) – uma especialização do servlet que permite que conteúdos dinâmicos sejam facilmente desenvolvidos.

• JTA (Java Transaction API) – é uma API que padroniza o tratamento de transações dentro de uma aplicação Java.

• EJBs (Enterprise Java Beans) – utilizados no desenvolvimento de componentes de software. Eles permitem que o programador se concentre nas necessidades do negócio do cliente, enquanto questões de infra-estrutura, segurança, disponibilidade e escalabilidade são responsabilidades do servidor de aplicações.

• JCA (Java Connector Architecture) – é uma API que padroniza a ligação a aplicações legadas.

• JPA (Java Persistence API) – é uma API que padroniza o acesso a banco de dados através de mapeamento Objeto/Relacional do Enterprise Java Beans.

• JAX-WS (Java API for XML Web Services), JAX-B (Java API for XML Binding) – é uma API para trabalhar com arquivos xml e webservices.

• JAAS (Java Autenthication and Authorizalion Service) – API padrão do Java para segurança.

• JMS (Java Message Service) – para troca de mensagens assíncronas.

• JNDI (Java Naming and Directory Interface) – para organizar espaço de nomes e objetos.

• JMX (Java Management Extensions) – para administração da sua aplicação e estatísticas sobre a mesma.

A Última versão disponível da especificação do Java EE è a versão 6, lançada em 10 de dezembro de 2009. É uma versão ainda muito recente, com poucas ferramentas e servidores disponíveis. A versão mais usada no mercado é a versão 5, de 2006.

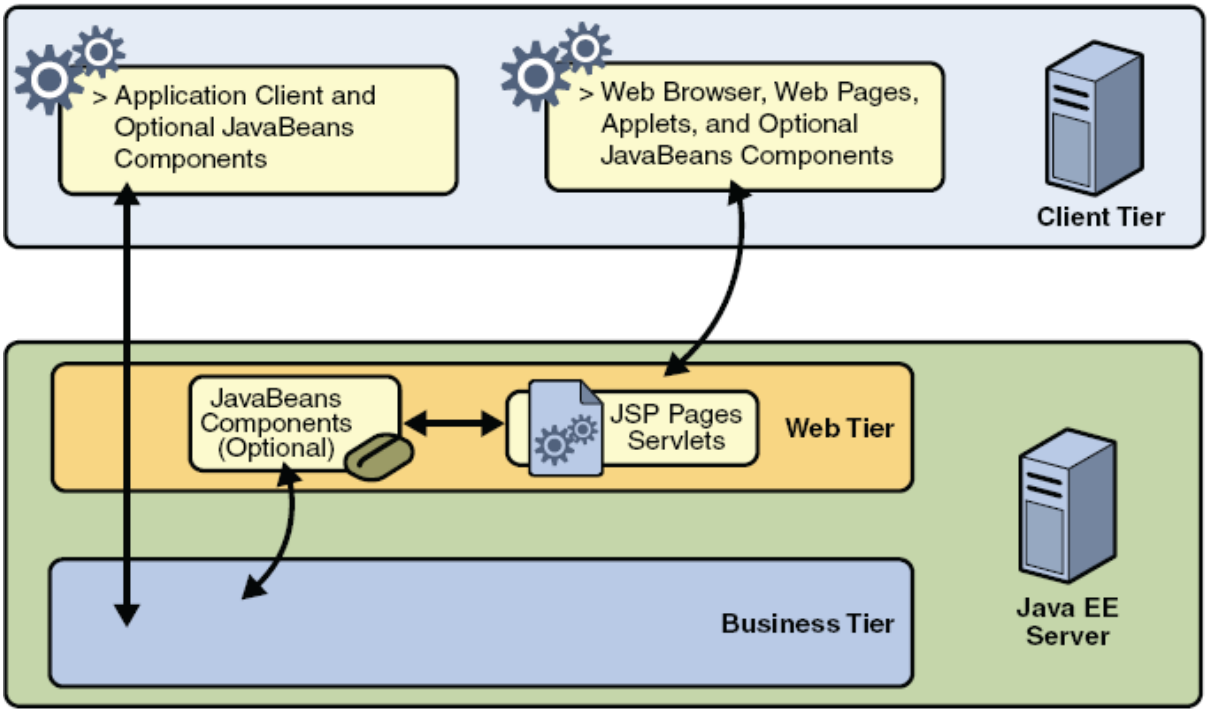
JSP e Servlets são sem dúvida as especificações essenciais que todo desenvolvedor Java vai precisar para desenvolver com a Web. Mesmo usando *frameworks* e bibliotecas que facilitam o trabalho para a Web, conhecer bem essas especificações é certamente um diferencial.

3.1 JAVA E APLICAÇÕES WEB

Uma aplicação Web consiste em um sistema de informática projetado para ser utilizado através de um navegador, na Internet ou em redes privadas (Intranet). Normalmente utiliza uma arquitetura multinível onde as funções executadas pela aplicação podem estar distribuídas em uma rede de computadores (HUDSON; PATEL; THOMAS, 1997).

A Plataforma Java Enterprise Edition, como visto, fornece um conjunto de bibliotecas e uma estrutura que dão suporte ao desenvolvimento de aplicações nos moldes da arquitetura de camadas. Desta forma, uma aplicação Web está inserida dentro da camada Web (Web Tier) da Plataforma, com seus respectivos componentes, conforme ilustra a Figura 7.

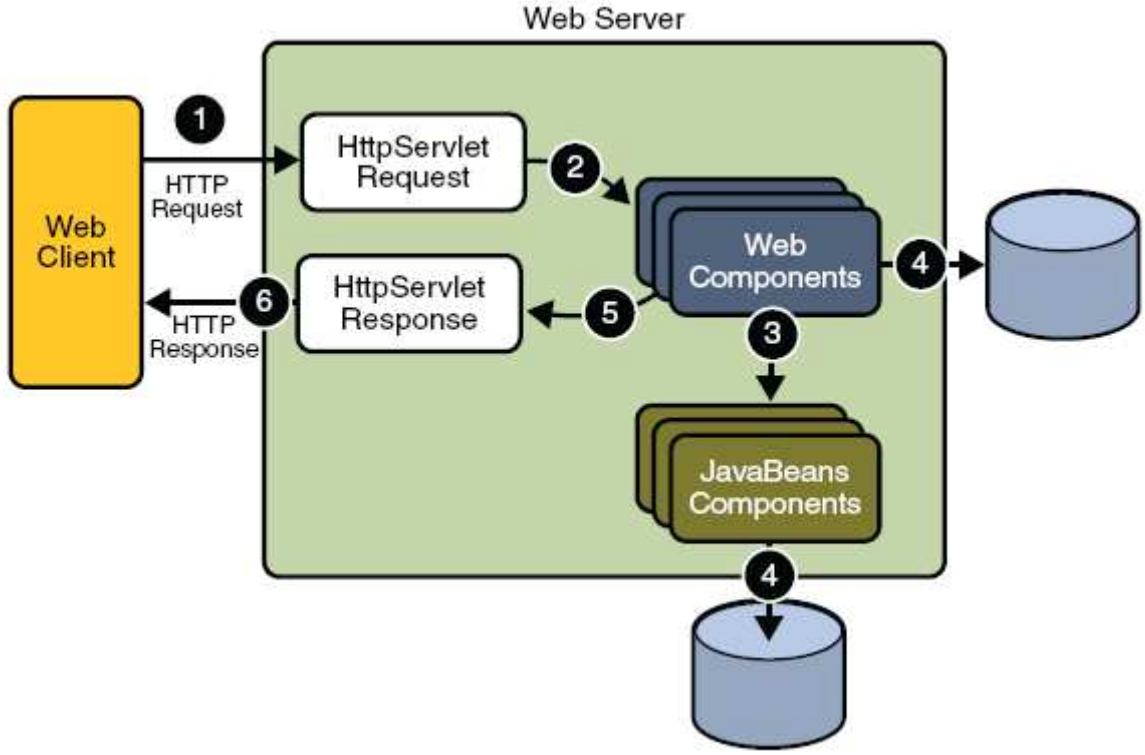
Figura 7: Componentes camada web Plataforma Java EE (SUN, 2008).



Assim os componentes principais para o desenvolvimento de uma aplicação web são Servlets e páginas criadas com a tecnologia Java Server Pages (JSP). Servlets são classes da linguagem de programação Java que processam dinamicamente requisições dos clientes e montam as respostas. Páginas JSP são documentos parecidos com arquivos textos que executam como Servlets, mas com uma abordagem mais natural para a criação de conteúdo estático É importante salientar que páginas desenvolvidas na linguagem HTML não são consideradas como componentes web pela especificação JEE (SUN, 2008)

A interação entre a camada do cliente (Web Client) e a camada de especificação web se dá conforme apresenta a Figura 8, no geral o cliente é representado pelo browser de navegação na Internet e o servidor web (Web Server) representa as tecnologias utilizadas no desenvolvimento dessas aplicações (MARIANO, 2008).

Figura 8: Funcionamento aplicação web usando a Plataforma Java Enterprise Edition (SUN, 2008).



A aplicação se inicia quando o cliente envia uma requisição HTTP para o servidor (pode ser utilizado um container Web, explicado no item 3.3 desta monografia) O servidor por sua vez, que implementa os Servlets e as páginas JSP, converte a requisição em um objeto do tipo HTTPServletRequest Este objeto é entregue para um componente web, que interage com as classes JavaBeans (representam os objetos da aplicação) ou com o banco de dados para gerar o conteúdo dinâmico O componente Web gera, então, um objeto HTTPServletResponse, que é enviado ao servidor e convertido como resposta para o cliente em HTML (MARIANO, 2008).

3.2 SERVIDOR DE APLICAÇÃO

Como já foi visto o Java EE tem um grande conjunto de especificações. Essas especificações, quando implementadas, vão auxiliar bastante o desenvolvimento da sua aplicação, pois não será necessário se preocupar com grande parte de código de infra-estrutura, que demandaria muito tempo de trabalho.

Para aplicar as especificações do Java EE pode-se utilizar um software específico e é necessário fazer o *download* de uma implementação dessas especificações.

Já que esse software tem papel de servir sua aplicação para auxiliá-la com serviços de infra-estrutura, ele ganha o nome de servidor de aplicação. A própria Sun/Oracle desenvolve uma dessas implementações, o Glassfish que é open source e gratuito, porém não é o líder de mercado apesar de ganhar força nos últimos tempos.

Existem diversos servidores de aplicação famosos compatíveis com a especificação do J2EE 1.4, Java EE 5 e alguns já do Java EE 6. O JBoss é um dos líderes do mercado e tem a vantagem de ser gratuito e open source. Alguns softwares implementam apenas uma parte dessas especificações do Java EE, como o Apache Tomcat, que só implementa JSP e Servlets (como já foi dito, duas das principais especificações), portanto não é totalmente correto chamá-lo de servidor de aplicação. A partir do Java EE 6, existe o termo “application server web profíle”, para poder se referência a servidores que não oferecem tudo, mas um grupo menor de especificações, consideradas essenciais para o desenvolvimento web.

Segue uma pequena lista de servidores de aplicação:

• RedHat, JBoss Application Server, gratuito, Java EE 5;

• Sun, GlassFish, gratuito, Java EE 6.

• Apache, Apache Geronimo, gratuito, Java EE 5;

• Oracle/BEA, WebLogic Application Server, Java EE 5;

• IBM, IBM Websphere Application Server, Java EE 5;

• Sun, Sun Java System Application Server (baseado no GlassFish), Java EE 5;

• SAP, SAP Application Server, Java EE 5;

3.3 SERVLET CONTAINER

O Java EE possui várias especificações, entre elas, algumas específicas para lidar com o desenvolvimento de uma aplicação Web:

• JSP;

• Servlets;

• JSTL;

• JSF;

Um Servlet Container é um servidor que suporta essas funcionalidades, mas não necessariamente o Java EE completo. É indicado a quem não precisa de tudo do Java EE e está interessado apenas na parte Web (boa parte das aplicações de médio porte se encaixa nessa categoria).

Há alguns servlet containers famosos no mercado, o mais famoso é o Apache Tomcat, mas há outros como o Jetty.

3.4 PÁGINAS DINÂMICAS

Quando a Web surgiu, seu objetivo era a troca de conteúdos através de páginas HTML estáticas. Eram arquivos escritos no formato HTML e disponibilizados em servidores para serem acessados nos navegadores. Imagens, animações e outros conteúdos também eram disponibilizados.

Mas logo se viu que a Web tinha um enorme potencial de comunicação e interação além da exibição de simples conteúdos. Para atingir esse novo objetivo, porém, páginas estáticas não seriam suficientes. Era preciso servir páginas HTML geradas dinamicamente baseadas nas requisições dos usuários.

Hoje, boa parte do que se acessa na Web (portais, blogs, home bankings etc) é baseado em conteúdo dinâmico, O usuário requisita algo ao servidor que, por sua vez, processa essa requisição e devolve uma resposta nova para o usuários.

Uma das primeiras idéias para esses “geradores dinâmicos” de páginas HTML foi fazer o servidor Web invocar um outro programa externo em cada requisição para gerar o HTML de resposta. Era o famoso CGI que permitia escrever pequenos programas para apresentar páginas dinâmicas usando, por exemplo, Perl, PHP, ASP e até C ou C++.

Na plataforma Java, a primeira e principal tecnologia capaz de gerar páginas dinâmicas são as Servlets, que surgiram no ano de 1997. Hoje, a versão mais encontrada no mercado é baseada nas versões 2.x, mais especificamente a 2.4 (parte do J2EE 1.4) e a 2.5 (parte do Java EE 5). A última versão disponível é a versão 3.0 lançada em Dezembro de 2009 com o Java EE 6, mas que ainda não tem adoção no mercado.

3.5 SERVLETS

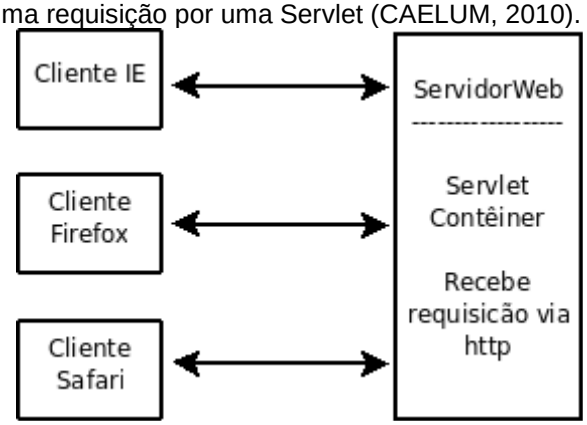
As Servlets são a primeira forma que veremos de criar páginas dinâmicas com Java. Usaremos a própria linguagem Java para isso, criando uma classe que terá capacidade de gerar conteúdo HTML. O nome “servlet” vem da idéia de um pequeno servidor (servidorzinho, em inglês) cujo objetivo é receber chamadas HTTP, processá-las e devolver uma resposta ao cliente.

Uma primeira idéia da Servlet seria que cada uma delas é responsável por uma página, sendo que ela lê dados da requisição do cliente e responde com outros dados (uma página HTML, uma imagem GIF etc). Como no Java tentamos sempre que possível trabalhar orientado a objetos, nada mais natural que uma Servlet seja representada como um objeto a partir de uma classe Java.

Cada servlet é, portanto, um objeto Java que recebe tais requisições (request) e retorna algo (response), como uma página HTML dinamicamente gerada.

A Figura 9, na página seguinte, mostra três clientes acessando o mesmo servidor através do protocolo HTTP:

Figura 9: Tratamento de uma requisição por uma Servlet (CAELUM, 2010).



O comportamento das Servlets que será visto foi definido na classe HttpServlet do pacote javax.servlet. Eles se aplicam às Servlets que trabalham através do protocolo HTTP.

A Interface Servlet é a que define exatamente como uma Servlet funciona, mas não é o que será utilizado agora, uma vez que ela possibilita o uso de qualquer protocolo baseado em requisições e respostas, e não especificamente o HTTP.

Para se escrever uma Servlet, criamos uma classe Java que estenda HttpServlet e sobrescreva um método chamado *service*. Esse método será o responsável por atender requisições e gerar as respostas adequadas. Sua assinatura:

Figura 10: Assinatura da classe Servlet (CAELUM, 2010).

```
protected void service (HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    ...
}
```

O método recebe dois objetos que representam, respectivamente, a requisição feita pelo usuário e a resposta que será exibida no final. Veremos que podemos usar esses objetos para obter informações sobre a requisição e para construir a resposta final para o usuário.

Um primeiro exemplo de implementação do método service seria aquele que não executa nada de lógica e apenas mostra uma mensagem estática de bem vindo para o usuário. Para isso, precisamos construir a resposta que a Servlet enviará para o cliente.

É possível obter um objeto que represente a saída a ser enviada ao através do método getWriter da variável response. E, a partir disso, utilizar um PrintWriter para imprimir algo na resposta do cliente:

Figura 11: Exemplo de Servlet (CAELUM, 2010).

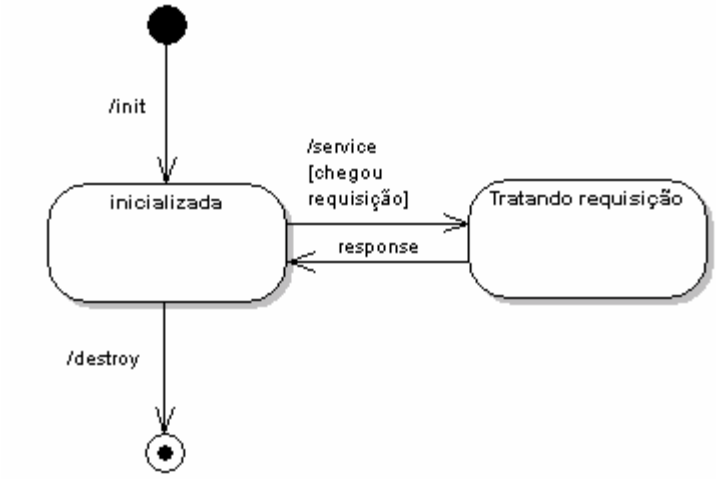
```
public class OiMundo extends HttpServlet {
    protected void service (HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
        PrintWriter out = response.getWriter();

        // escreve o texto
        out.println("<html>");
        out.println("<body>");
        out.println("Primeira servlet");
        out.println("</body>");
        out.println("</html>");
    }
}
```

O único objetivo da Servlet acima é exibir uma mensagem HTML estática para os usuários que a requisitarem. Mas note como seria muito simples escrever outros códigos Java mais poderosos para gerar as Strings do HTML baseadas em informações dinâmicas.

3.5.1 CICLO DE VIDA DE UMA SERVLET

O ciclo de vida de uma Servlet é composto por quatro transições e dois estados, conforme a figura abaixo:

Figura 12: Diagnóstico dos estados que representa o ciclo de vida de uma Servlet (VAHLDICK, 2006).



A especificação não determina quando o servidor deve instanciar uma Servlet, ou quando deve destruí-lo, mas especifica os métodos associados a cada transição (VAHLDICK, 2006):

• init(): é invocado após a instanciação do Servlet. Convencionalmente utilizado para inicializar a estrutura interna da Servlet:

• service(): é chamado para atender as solicitações do cliente. Recebe a requisição (request) e fornece a resposta (response), normalmente, uma página HTML produzida por ele:

• destroy(): é invocado antes do servidor retirar o Servlet da memória.

3.5.2 SERVLET x CGI

Diversas requisições podem ser feitas à mesma Servlet ao mesmo tempo em um único servidor. Por isso, ela é mais rápida que um programa CGI comum que não permitia isso. A especificação de Servlets cita algumas vantagens em relação ao CGI.

• Fica na memória entre requisições, não precisa ser reinstanciada;

• O nível de segurança e permissão de acesso pode ser controlado em java;

• em CGI, cada cliente é representado por um processo, enquanto que com Servlets, cada cliente é representado por uma linha de execução.

Essa monografia está focada na HttpServlet, um tipo que gera aplicações Web baseadas no protocolo HTTP, mas vale lembrar que a API não foi criada somente para este protocolo, podendo ser estendida para outros protocolos também baseados em requisições e respostas.

3.6 JAVA SERVER PAGES – JSP

As JSPs são páginas HTML dinâmicas. Nelas podem ser incorporados código Java, JavaBeans, biblioteca de *tags*, expression languages, e outras mais tecnologias J2EE (VAHLDICK, 2006).

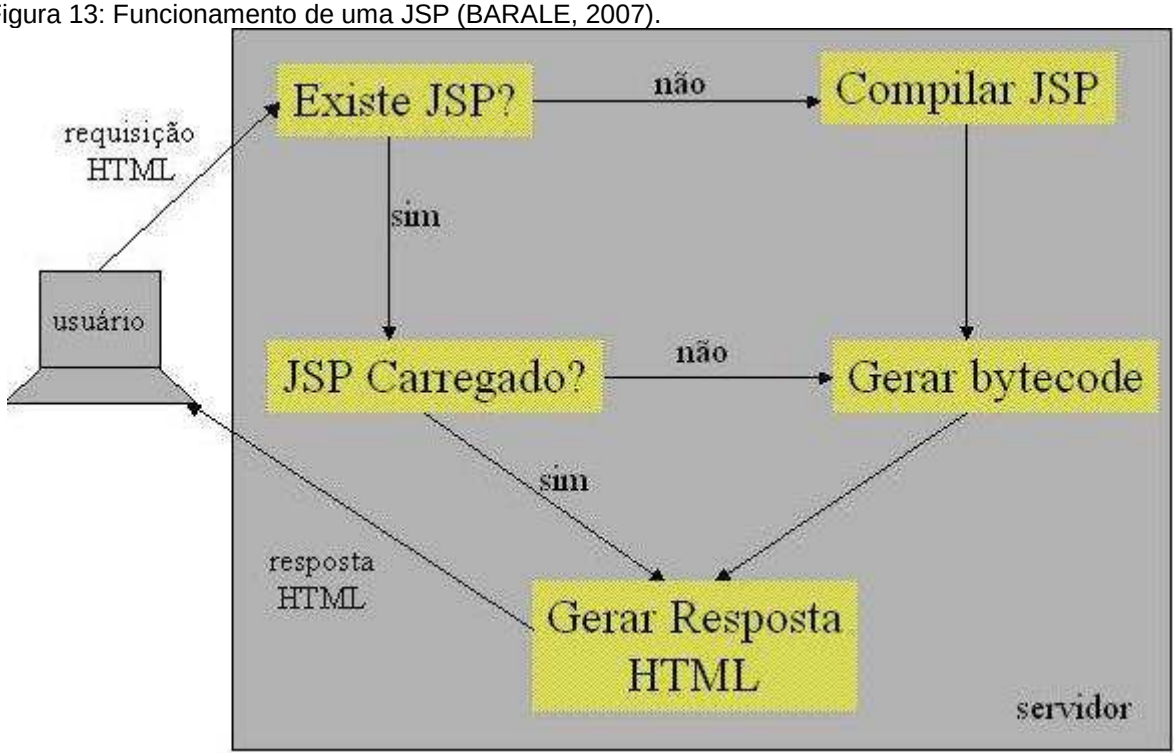
Em termos funcionais, a tecnologia JSP (Java Server Pages) não oferece nada que não possa conseguir com Servlets. O JSP, entretanto, oferece vantagem de ser facilmente codificado, facilitando assim a elaboração e manutenção da página dinâmica. Além disso, permite separar a programação lógica (parte dinâmica) da programação visual (parte estática), facilitando o desenvolvimento de aplicações mais robustas, seguras, portáteis e eficientes, em que o programador e designer podem trabalhar em um mesmo projeto de forma independente.

A tecnologia JSP é usada para servir conteúdo dinâmico para o usuário, usando lógica de dados no lado do servidor. JSP possui a vantagem da portabilidade de plataforma, e da compilação das páginas permitindo que elas rodem mais rapidamente. Possibilita também que o desenvolvedor de sites possa produzir aplicações que acessa aos mais diversos bancos de dados utilizando a tecnologia JDBC (Java Database Connectivity), acesso a arquivos texto, captação de informações sobre visitante e sobre o servidor, entre diversas outras (BARALE, 2007).

3.6.1 FUNCIONAMENTO DE UMA JSP

Inicialmente o usuário faz urna requisição de urna página JSP, por meio de um browser. Esta página, então, será processada pelo servidor: se for a primeira vez, a página JSP é transformada em um Servlet, que é compilado, e gerando um bytecode (conhecido por .class), a partir da qual é gerada urna página HTML (HyperText Markup Language) que é enviada ao browser do cliente. A partir da segunda vez que esta página for acessada é verificado apenas se ocorreram ou não quaisquer mudanças; caso negativo, apenas o bytecode é chamado para gerar a página HTML. A figura 13 ilustra esse funcionamento.

Figura 13: Funcionamento de uma JSP (BARALE, 2007).



Vale lembrar que existe uma pequena lentidão no primeiro acesso a página JSP devido à compilação do JSP em Servlets (bytecode .class). A partir do segundo acesso, as páginas JSP executam com maior rapidez.

JSP usa a linguagem Java como base para a sua linguagem de *Scripts*, aproveitando todo seu potencial, motivo pelo qual JSP apresenta-se muito mais flexível e robusto (BARALE, 2007).

3.7 MODELOS DE ARQUITETURA PARA AS APLICAÇÕES WEB J2EE

A tecnologia JSP criou a especificação informal de dois modelos de arquitetura para as aplicações WEB J2EE (KURNIAWAN, 2002): Modelo 1 e Modelo 2.

Na arquitetura de modelo 1 toda a lógica de negócios e acesso ao banco de dados está embutida no próprio JSP. No modelo 2. a Servlet é responsável pelo controle das regras de negócio, e o JSP pela apresentação.

Essa última arquitetura respeita os princípios do padrão MVC (Model-View-Controller) (LOZANO, 2005). Nesse padrão a aplicação é desenvolvida em três camadas com finalidades distintas:

• Modelo: contém a lógica do negócio, e as entidades mínimas que representam os conceitos envolvidos com essas regras;

• Visão: camada responsável pela apresentação de informações e entrada de dados. Corresponde a interface entre o sistema e o usuário. Esse pode ser humano, máquina ou outros sistemas;

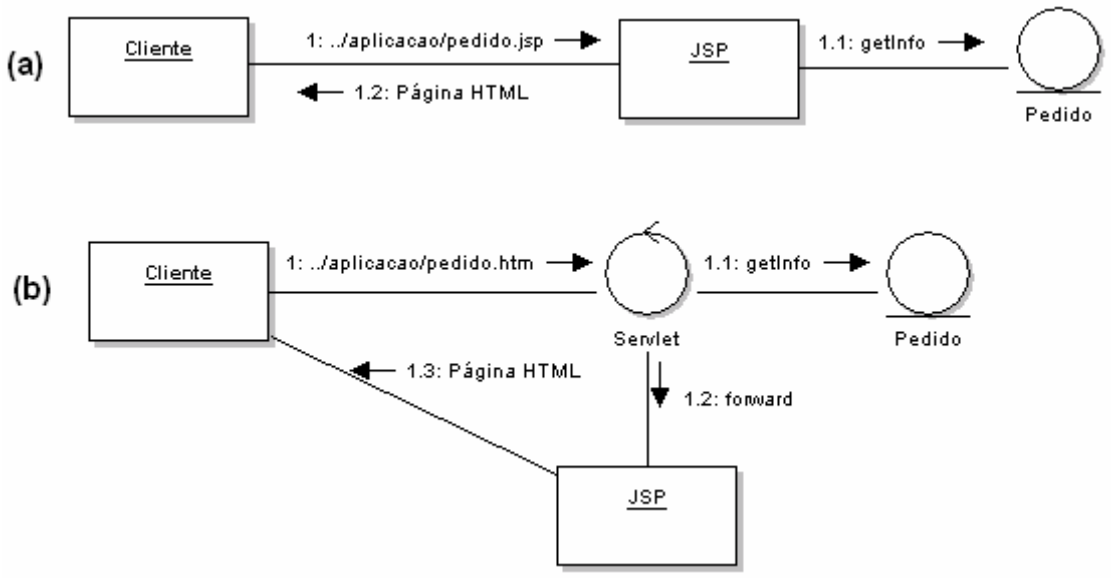
• Controle: coordena a interação com o usuário recebendo e retornando informações, e faz consultas e alterações no modelo.

Essa divisão traz muitas vantagens, como por exemplo, a possibilidade de reutilizar parte do sistema para diferentes meios de apresentação, como WEB ou GUI (interfaces gráficas). Também permite a distribuição das camadas em equipamentos diferentes e interligados. Outro ganho está no próprio desenvolvimento que é facilitado, pois permite mais programadores trabalhando ao mesmo tempo sobre o mesmo projeto, já que a própria divisão das tarefas é forçada e centrada nas funcionalidades de cada parte (VAHLDICK, 2006).

No modelo 2, também chamado de MVC2 (LOZANO, 2005), o servlet recebe as solicitações, desempenhando o papel de controle, e redireciona o resultado para uma página JSP, fazendo o papel de visão. A camada de modelo corresponde aos objetos de negócio, que podem ser persistidos e buscados de uma base de dados.

Os dois modelos de arquitetura são ilustrados na figura 14.

Figura 14: Modelos de Arquitetura: (a) modelo 1 (b) MVC2 (VAHLDICK, 2006).



4 FERRAMENTAS DE DESENVOLVIMENTO

Para implementação de sistemas corporativos Web com a plataforma Java EE, é possível encontrar na internet muitas ferramentas open source e privadas que podem ser usadas para compor o ambiente de desenvolvimento de uma equipe. Para aplicativos web com a linguagem Java podemos utilizar diversas ferramentas, entre elas iremos abordar o Apache Tomcat e o Eclipse.

4.1 APACHE TOMCAT

O Tomcat é um contêiner de servlets, desenvolvido pela Apache Software Foundation, cujo objetivo é realizar a execução de aplicações web (APACHETOMCAT, 2008) desenvolvidas na linguagem Java.

Um contêiner web, dentro da especificação JEE, deve prover serviços que redirecionem requisições, garantam a segurança, concorrência e gerenciamento do ciclo de vida de uma aplicação. Também deve possibilitar que componentes web tenham acesso às bibliotecas da plataforma de funções específicas, tais como de transações e email (SUN, 2008).

Para que possa executar e reconhecer uma aplicação Web, o Tomcat é dividido em diretórios que especificam os arquivos que tratam da execução e interpretação das aplicações. Assim, contém as seguintes pastas (MARIANO, 2008):

• bin: pasta onde se encontram os arquivos que iniciam e terminam a execução do container: Startup e ShutDown, nos formatos para os sistemas operacionais Unix (.sh) e Windows (.bat), além de outros *scripts*.

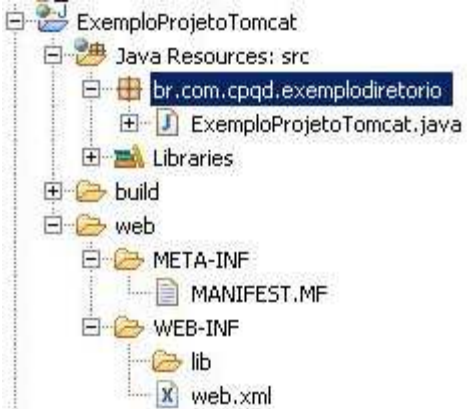
• conf: contém os arquivos de configuração do container. O arquivo mais importante desta pasta é o server.xml, pois este é o responsável pela configuração principal do container.

• logs: pasta que se encontram os logs de execução do container.

• webapps: pasta onde as aplicações são implantadas para que sejam executadas pelo Tomcat.

Uma aplicação Web é definida como uma hierarquia de diretórios e arquivos em um Ieiaute padrão, que podem ser acessados individualmente quando no ambiente de desenvolvimento, ou em um pacote de arquivos denominado de WAR (Web Archive), quando no ambiente de execução, no caso o container Tomcat. O pacote contém todos os arquivos necessários para que o Tomcat possa reconhecer e executar a aplicação. Para facilitar a criação desse arquivo a partir do ambiente de desenvolvimento é preciso que os códigos estejam distribuídos conforme a figura abaixo:

Figura15: Exemplo diretórios para criação de aplicação no padrão Tomcat (MARIANO, 2008).



Nessa figura o diretório mais importante é a pasta Web, pois contém os arquivos que deverão compor o WAR e devem ser distribuídos da seguinte forma:

• * html, *jsp: páginas (interfaces do usuário), devem ser armazenados direto na pasta web do projeto, ou em pasta específica criada dentro dessa pasta;

• /WEB-INF/webxml: arquivo descritor da aplicação, responsável pelo contexto da aplicação, contém as configurações de servlets, de filtros, mapeamentos, classes e regras de controle de acesso a recursos;

• /WEB-INF/classes: contém as classes (servlets ou não) compiladas;

• /WEB-INF/lib: contém todas as bibliotecas necessárias para a execução da aplicação;

O arquivo nesse formato deve ser implantado no diretório webapps do container, e ter um context path associado. Um context path é um nome que será usado para referenciar a aplicação no navegador web do cliente (MARIANO, 2008).

4.2 AMBIENTE INTEGRADO DE DESENVOLVIMENTO (IDE) ECLIPSE

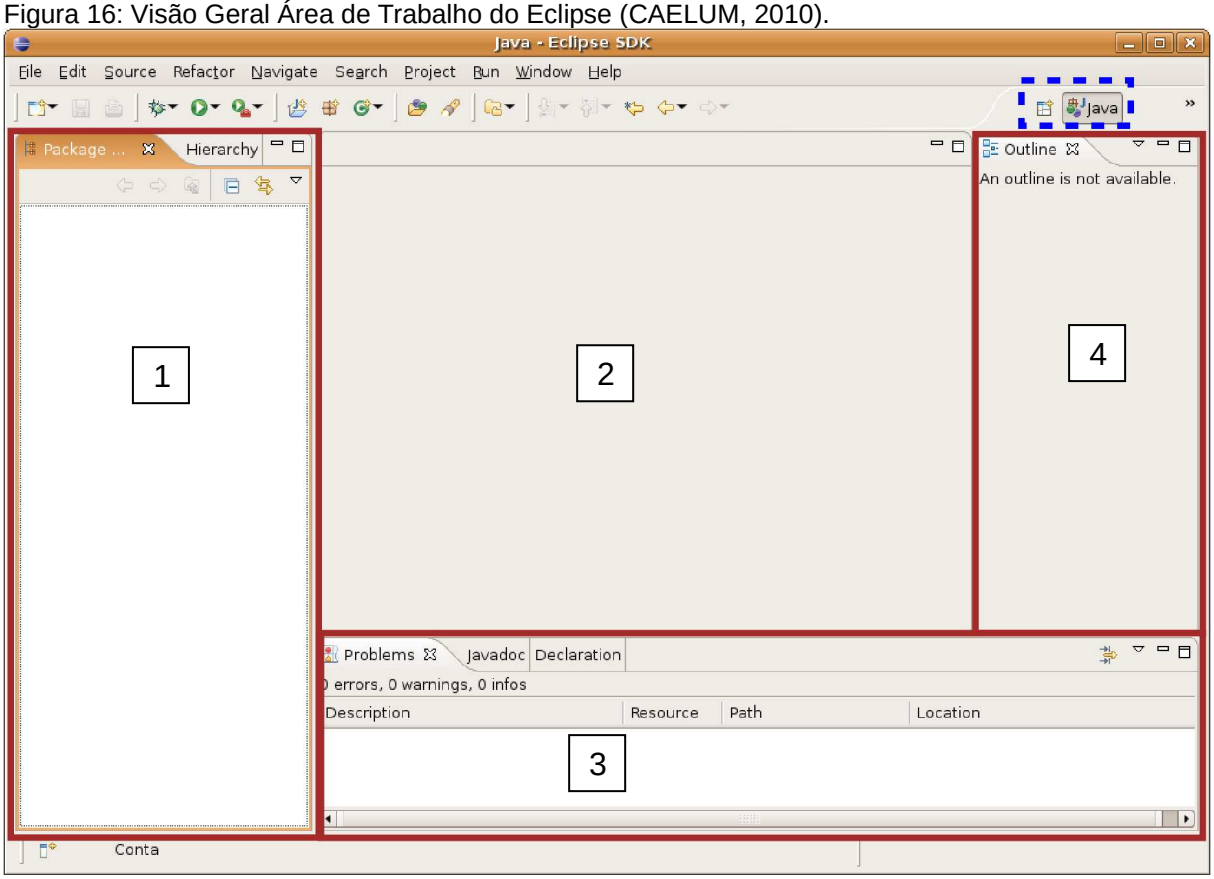
O Eclipse (HTTP://www.eclipse.org) é uma IDE (Integrated Development Environment) que contém as ferramentas necessárias para desenvolver diversos programas e aplicativos de uma maneira otimizada e padronizada. Diferente de uma RAD, onde o objetivo é desenvolver o mais rápido possível através do arrastar e soltar do mouse, onde montanhas de código são gerados em background, uma IDE auxilia no desenvolvimento, evitando se intrometer e fazer muita mágica.

Para tanto, é composto por ferramentas de criação, implantação e gestão de todo ciclo de vida do software, o que lhe confere uma estrutura flexível para o desenvolvimento de sistemas e aplicativos. Possui suporte à linguagem Java, exemplos de construção de código e forte orientação a desenvolvimento baseado em plugins. É de fácil integração com outras tecnologias, e portanto, é um ambiente adaptável á necessidade dos programadores (MARIANO, 2008).

O Eclipse é a IDE líder de mercado. Formada por um consórcio liderado pela IBM, possui seu código livre. A última versão é a 3.4. Precisamos do Eclipse 3.1 ou posterior, pois a partir dessa versão é que a plataforma dá suporte ao Java 5.0. Você precisa ter apenas a Java RE instalada.

A Figura 16 apresenta uma visão geral da área de trabalho dessa ferramenta, também conhecida como workbench.

Figura 16: Visão Geral Área de Trabalho do Eclipse (CAELUM, 2010).



Normalmente, os elementos exibidos na inicialização do Eclipse, conforme a numeração apresentada pela Figura 16, são:

1. Package Explorer é um visualizador simples e elegante dos projetos; todos os arquivos-fonte (classes - .java), os bytecodes (os arquivos classes compilados - .class), as configurações gerais, eventuais arquivos de backup, arquivos de imagens e outros arquivos inerentes ao escopo do programa podem ser visualizados nesta área. Por ter aparência semelhante ao Windows Explorer, toma familiar a manipulação desses arquivos.

2. Editor de Texto: é um editor de textos parecido com o Bloco de Notas; entretanto, ressalta as palavras da linguagem Java com diferentes cores para facilitar o entendimento do código, diferencia também as declarações de variáveis. Além disso, possui o recurso de auto-sugestão de elementos da linguagem e detecção de erros em tempo de implementação, destacando-os em vermelho, ou amarelos, dependendo do tipo do erro. Todos esses recursos auxiliam o programador durante o processo de implementação das aplicações.

3. Console: terminal de entrada e saída de dados por *streams*. Nesta parte do eclipse, pode-se verificar erros gerados em tempo de compilação e execução da aplicação que o editor não consegue detectar.

4. Outline: muito útil para verificar a estrutura interna de um arquivo, pois funciona como o Package Explorer, entretanto exibe os elementos que compõem as classes do projeto (arquivos com a extensão .java).

Todo o desenvolvimento de um projeto ocorre nessa área de trabalho. Entretanto, algumas especificações de projetos exigem que esse ambiente seja personalizado, integrado às ferramentas necessárias.

5 CONSIDERAÇÕES FINAIS

Hoje em dia, com a constante evolução Internet, sistemas Web são fundamentais para as empresas, pois agregam valor, e auxiliam no suporte e agilidade no atendimento das necessidades dos clientes.

Como a Web está em constante evolução, cada dia novas tecnologias surgem no mercado para agilizar o desenvolvimento de aplicações para esse ambiente. É preciso analisar, conhecer e desenvolver padrões fundamentados em experiências de sucesso, pois a escolha das tecnologias corretas implica o bom funcionamento de toda a aplicação.

É nesse contexto que destacam-se o uso da Linguagem Java e da Plataforma Java Enterprise Edition, que possibilitam desenvolver essas aplicações de maneira padronizada e simplificada. Mas a plataforma JEE sozinha não garante um bom resultado, é por isso que ela possibilita ser integrada com outras tecnologias. É nessa questão que, percebe-se o quanto o processo de implementação pode ser facilitado e agilizado.

Essas integrações são facilitadas principalmente devido à sua portabilidade, que permite que os sistemas nela baseados rodem em qualquer sistema operacional. Isso é possível através do conceito de máquina virtual, onde existe entre o sistema operacional e a aplicação uma camada extra, responsável por ‘traduzir o que sua aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento.

A utilização da tecnologia JAVA aponta ganhos como: manutenibilidade do código e reuso da aplicação, aliados a uma melhor organização da equipe na distribuição de tarefas e ganho no tempo de desenvolvimento.

Esta monografia apresenta, portanto, conceitos fundamentais para o desenvolvimento de aplicações Web com o uso da Plataforma Java Enterprise Edition de forma que sejam possibilitadas integrações que satisfaçam as necessidades das empresas com as tecnologias mais populares disponíveis para o desenvolvimento dessas aplicações.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Citação**: NBR-10520/ago - 2002. Rio de Janeiro: ABNT, 2002.

_____. **Referências**: NBR-6023/ago. 2002. Rio de Janeiro: ABNT, 2002.

ALUR, D.; CRUPI, J.; MALKS, D. **Core J2EE Patterns. As melhores práticas e estratégias de Design**. Campus, 2004.

APACHE-TOMCAT. **Apache Tomcat 6.0 Documentation Index.** Disponível em: <http://tomcat.apache.org/tomcat-6.0-doc/index.html>. Acesso em: 10 de maio 2010.

BARALE, R. F. **Desenvolvimento de um Sistema de Vendas na Web utilizando JSP.** 2007. 77f. Monografia (Conclusão do Curso de Bacharelado em Sistemas de Informação). UNIMINAS/Uberlândia. 2007.

CAELUM. **Java e Orientação a Objetos**. Disponível em: <http://downloads.caelum.com.br/apostila/caelum-java-objetos-fj11.pdf>. Acesso em 12 de mar. 2010.

_____. **Java para Desenvolvimento Web**. Disponível em: <http://downloads.caelum.com.br/apostila/caelum-java-web-fj21.pdf>. Acesso em 12 de mar. 2010.

CBPF from RJ: **Banco de dados.** Disponível em: <http://www.rederio.br/downloads/pdf/NT006_00.pdf>. Acesso em: 28 de abr. 2010.

DEITEL, H. M. **Java, Como Programar**; trad. Carlos Arthur Lang Lisbôa. – 4 ed. Porto Alegre: Bookman, 2003.

HUDSON, A., et al. **Programando em Java para a Internet**. Makron, 1997.

KURNIAWAN, B. **Java para a Web com Servlets, JSP e EJB**. Rio de Janeiro: Ciência Moderna, 2002.

LOZANO, F. **Servlets no Tomcat 5: Aplicações WEB MVC em Java.** Java Magazine, n. 20, 2005.

MARIANO, G. C. **Desenvolvimento de Aplicações Web com a Plataforma Java EE.** 2008. 59f. Monografia (Conclusão do Curso de Tecnologia em Informática). UNICAMP/Limeira. 2008.

TANENBAUM, A. S., VAN STEEN, M. **Distributed Systems: Principles and Paradigms** – 2 ed. Hardcover, 2007.

SAUVÉ, J. P. **Caracterização do tráfego de Rede.** Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/pr/html/req/req4.htm>. Acesso em 03 de maio de 2010.

SIERRA, K.; BATES, B. **Use a Cabeça! JSP & Servlets.** Rio de Janeiro: Atla Books, 2005.

SILVEIRA, I. F. **Java, das torradeiras à Internet**. Disponível em: <http://www.infowester.com/lingjava.php>. Acesso em: 28 de abr. 2010.

SUN. **The Java EE 5 Tutorial For Sun Java System Application Server 9.1.** Sun Microsystems, 2008.

VAHLDICK, A. **Um Componente de Groupware para Aplicações Web J2EE usando Sistemas Multi-agentes.** 2006. 91f. Monografia (Conclusão do Curso de Pós-Graduação em Tecnologias para o Desenvolvimento de Aplicações Web). Universidade Regional de Blumenau/Blumenau. 2006.