UNIVERSIDADE
ESTÁCIO DE SÁ

PÓS-GRADUAÇÃO

PATRICIA VILAS BOAS VARGAS

RIA X JAVA

Rio de Janeiro

Janeiro 2010

Patricia Vilas Boas Vargas

RIA x *Java*

Monografia apresentada à Universidade Estácio de Sá, como requisito final do curso de pós-graduação em Desenvolvimento *Java*.

Rio de Janeiro
Janeiro 2010

Agradecimentos

A Deus, pela oportunidade de reescrever a minha história.

Aos meus pais Telmo e Eliane Vargas por terem me dado a vida, o alimento (físico, mental e espiritual), o amor incondicional, os momentos bons e ruins e, por serem ontem, hoje e sempre o meu porto seguro.

Aos meus irmãos Renato e Victor Vargas por terem sempre acreditado em mim.

Ao meu companheiro Thiago Pereira, pela inspiração, força e privações para trilhar esta nova estrada.

Aos professores Adriana Aparício Sicsú, Alcebíades Lôbo, Alfredo Boente, André S. Barbosa, Carlos Alberto Alves Lemos, Carlos Sicsú, Denis Cople, João Mendes Filho, Luiz Roberto Bastos, Oswaldo Borges, Renato Côrtes, Wanelytcha Simonini pela dedicação em transmitir seus conhecimentos e pela paciência. Sei que dei trabalho!

Aos colegas de curso Alcebíades Barbosa, André Prata, Bolívar Mena, Edir Maia, Everaldo Rabelo, Fábio Bernardes, Jonath Ferreira, Juliano José Nery, Leandro A. Carneiro, Milton Brander, Rosemar Martins, Rubem Moreira, Samuel Kanashiro, Vitor Raphael Menegatti Chagas pelas alegrias, ansiedades, pelos momentos de luta e perseverança.

Epígrafe

“A primeira regra de qualquer tecnologia utilizada nos negócios é que a automação aplicada a uma operação eficiente aumentará a eficiência. A segunda é que a automação aplicada a uma operação ineficiente aumentará a ineficiência.”

Bill Gates

Sumário

INTRODUÇÃO 1
Capítulo 1 O QUE É RIA? 2
Capítulo 2 CONCEITUANDO MVC 9
Capítulo 3 FRAMEWORKS JAVA QUE IMPLEMENTAM MVC 13
CONCLUSÃO 18
REFERÊNCIAS 19
GLOSSÁRIO 21

RESUMO

A usabilidade é o objeto cobiçado na engenharia de software. Um software de fácil utilização conquista o usuário, o maior exemplo na informática é o Windows que conquistou usuários do Linux devido a sua *interface* gráfica. RIA (*Rich Internet Application* – Aplicações de Internet Rica) aparece prometendo uma revolução e fornecendo a tecnologia necessária para a construção de softwares amigáveis e de fácil operação pelo usuário. Trabalhamos a questão de como a tecnologia RIA interage com *Java* e se esta interação é aderente e simples. Nossos objetivos são conceituar RIA e MVC, apresentar os *framework*s *Java* mais conhecidos que implementam a arquitetura MVC. Utilizamos como referencial teórico a obra literária *Java and Flex Integration Bible* de Matthew Keefe e Charles Christiansen. Matthew Keefe é *design*er e desenvolvedor pleno com uma sólida experiência em desenvolvimento de aplicações para a *web*. Matt é o autor de *Flash and PHP Bible*. Charles A. Christiansen Jr. é engenheiro de software sênior, que atualmente usa *Java* e *Flex* 3. Charles tem desenvolvido aplicações que abrangem desde Aplicações *Java* centradas nos clientes que usam RMI sobre conexões *dialup* até Aplicações *web* rápidas e leves que usam *Spring* e *Hibernate*. A metodologia utilizada foi a dedutiva, analisando o aspecto fenomenológico do tema com base numa pesquisa bibliográfica com abordagem qualitativa.

Palavras-chave: RIA, MVC, *Java*, *framework*.

ABSTRACT

Usability is the coveted object in software engineering. A user-friendly software conquers the User, the greatest example is the Windows computer that has Linux users because of its graphical interface. RIA (Rich Internet Application) appears promising a revolution and providing the necessary technology to build friendly software and easy operation by the User. Work the issue of how RIA technology interacts with Java and this interaction is adherent and simple. Our objectives are to conceptualize RIA and MVC, present the most popular Java frameworks that implement the MVC architecture. We use as a literary theorist Java and Flex Integration Bible from KEEFE , Matthew, Christiansen, Charles A. Matthew Keefe is a designer and developer with a full solid experience in developing applications for the web, Matt is the author of Flash and PHP Bible. Charles A. Christiansen Jr. is a senior software engineer, which currently uses Java and Flex 3. Charles has developed applications ranging from Java-centric clients that use RMI over dialup connections to web Applications fast and light using Spring and Hibernate. The methodology used was a deductive, analyzing the phenomenological aspect of the topic based on a literature with a qualitative approach.

Tags: RIA, MVC, Java, framework.

INTRODUÇÃO

Não é de hoje que usabilidade é o objeto de estudo e a chave para o sucesso de produtos e softwares. RIA (*Rich Internet Application* – Aplicações de Internet Rica) é uma tecnologia que tem como objetivo facilitar a experiência do usuário, focado na usabilidade, facilitando a interação do usuário com o *software*. Pensando em arquitetura de software, a mais utilizada com RIA é MVC (*Model*, *View*, *Controler*). Trata-se de uma arquitetura dividida em três componentes: visão, controle e modelo. RIA é a tecnologia utilizada no componente de visão também conhecido como componente de apresentação; nos componentes de controle e modelo faremos referência ao *Java*.

Trabalhamos a questão de como a tecnologia RIA interage com *Java* e se esta interação é aderente e simples.

Nossos objetivos foram conceituar RIA, conceituar MVC e apresentar os *frameworks Java* mais conhecidos que implementam a arquitetura MVC.

As questões de estudo trabalhadas foram o que é RIA? O que é MVC? Quais *framework*s *Java* implementam MVC?

Utilizamos como referencial teórico a obra *Java and Flex Integration Bible* de Matthew Keefe e Charles A. Christiansen Jr. Matthew Keefe é *designer* e desenvolvedor pleno com uma sólida experiência em desenvolvimento de aplicações para a *web*. Matt é o autor de *Flash and PHP Bible*. Charles A. Christiansen Jr. é engenheiro de *software* sênior, que atualmente usa *Java* e *Flex* 3. Charles tem desenvolvido aplicações que abrangem desde aplicações *Java* centradas nos clientes que usam RMI sobre conexões *dialup* até aplicações *web* rápidas e leves que usam *Spring* e *Hibernate*.

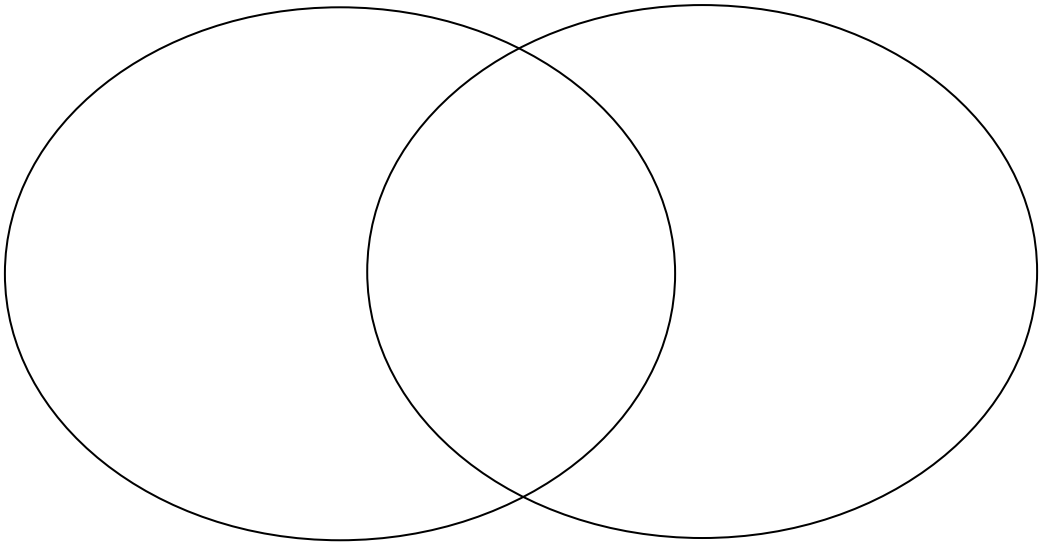
Como já mencionado, a usabilidade é o objeto cobiçado na engenharia de *software*. Um *software* de fácil utilização conquista o usuário. O maior exemplo na informática é o Windows que conquistou usuários do Linux devido a sua *interface* gráfica. RIA aparece prometendo uma revolução e fornecendo a tecnologia necessária para a construção de *softwares* amigáveis e de fácil operação pelo usuário.

A metodologia utilizada na elaboração deste trabalho foi a dedutiva, analisando o aspecto fenomenológico do tema com base numa pesquisa bibliográfica com abordagem qualitativa.

Capítulo 1
O QUE É RIA?

> Em 2001, a Macromedia (agora Adobe) usou pela primeira vez o *termo Rich Internet Applications* (RIA) para descrever um novo modelo de aplicativos para a *web*, conforme figura 1, baseado no que aplicativos tradicionais para *desktop* ofereciam (*interface* responsiva e intuitiva) com o melhor da Internet (seu alcance global e comunicação) criou uma plataforma para o desenvolvimento de RIAs, visando aprimorar a camada de apresentação das aplicações. (TERRACINI, 2006, p.52)

Figura 1 – As RIAs aproveitam o melhor dos dois mundos



(TERRACINI, 2006, p.52)

Para entender o surgimento das RIAs vamos voltar no tempo e relembrar as grandes mudanças na indústria de *softwares*. Mudamos de *mainframes* com terminais burros para a arquitetura cliente-servidor implantada em *desktops (PC - personal computer)* com poder de processamento. Com a disponibilidade da Internet surgiram as aplicações *web* e atualmente as RIAs, com a proposta de unir o que há de melhor entre os *softwares desktop* e as aplicações *web*.

> As primeiras aplicações RIA nasceram já em 1995 quando o *Java* foi criado. A popularidade inicial do *Java* manifestou-se em uma onda de pequenos programas chamados de *Applets Java* para *download*. *Applets*, criado com bibliotecas *Java* AWT (posteriormente *Swing*), foram executados por navegadores *Java Virtual Machine* (JVM). Ironicamente, a tecnologia que tornou *Java* popular não foi explorada e hoje *Java* brilha principalmente nos servidores e em dispositivos móveis. (FAIN, 2007, p.3)

Quando falamos em RIA automaticamente pensamos em usabilidade. No estudo da Interação Homem-Computador (IHC) a usabilidade se refere à facilidade com a qual o usuário opera um sistema ou interage com as interfaces do mesmo. Para a indústria, usabilidade traduz-se em retorno financeiro. Um sistema intuitivo e de fácil operação diminui o investimento em treinamento do usuário; um sistema com tratamento de falhas que possibilitam o retorno ao início do processo sem perdas, garante a fidelidade do mesmo.

Segundo a Adobe[1] (2009) os benefícios de RIAs são: Oferecer aos usuários uma experiência mais rica, mais envolvente; manter o ritmo das expectativas dos usuários em alta; aumentar a fidelidade dos clientes e gerar maiores lucros; alavancar pessoas, processos e infra-estrutura existente.

Abrangendo um pouco mais, podemos acrescentar aos benefícios e vantagens itens como: a riqueza extraída do próprio nome da tecnologia. Com a tecnologia RIA é possível trazer para a *interface* comportamentos, que com a dobradinha HTML e CSS não é possível, componentes mais elaborados, animações que conquistam o usuário; o Tempo de resposta é um benefício imensurável. Com RIA alguns processamentos podem ser feitos no cliente (*browser*) possibilitando atualizar apenas determinado componente ou áreas da interface, aumentando a produtividade do usuário uma vez que diminui a dispersão; e como fica o relacionamento entre cliente-servidor? Todos saem ganhando. O usuário ganhou tempo de resposta devido à capacidade de processamento no cliente. Com diminuição da carga de processamento nos servidores, este aumenta a capacidade de atender um número maior de solicitação de clientes; comunicação assíncrona. Devido à capacidade de atualizar componentes ou áreas da *interface* o cliente não precisa ficar parado esperando a resposta do processamento no servidor. Toda essa capacidade de processamento no cliente e de trocar apenas as informações relevantes e não mais todo o código HTML, diminui consideravelmente o fluxo na rede. E a logística de instalação e manutenção? Caiu vertiginosamente. O máximo necessário para algumas plataformas de RIA é a instalação de um *plug-in*. As versões recentes dos navegadores mais conhecidos já estão liberando o instalador com estes *plug-ins* e as versões que não instalam estes *plug-ins* no primeiro acesso, a aplicação detecta a ausência do *plug-in* e dispara a instalação do mesmo; multiplataforma. Não importa se usamos os sistemas operacionais Linux,

[1] http://www.adobe.com/resources/business/rich_internet_apps/#open

Windows... ou se usamos os equipamentos PC, dispositivo móvel.., basta ter um navegador de Internet e acesso ao servidor da aplicação via Internet ou rede local.

É claro que estamos longe do mundo perfeito; RIA também tem suas desvantagens e limitações. As RIAs são executadas dentro de *sandbox*, caixas ou áreas que não possibilitam acesso a recursos do sistema; os *browser* possibilitam desativar a opção de executar script, mas se o usuário estiver com esta opção desativada os scripts da aplicação não serão executados; para algumas plataformas RIA a velocidade de processamento de *scripts* no cliente perde performance; isso não acontece nas plataformas *Java* e *Flex*.

Entrando no mundo das plataformas RIA, além das já conhecidas *Flex*, *Java*, *Silverlight* e AJAX, Fain (2007) também nos apresenta as soluções: OpenLaszlo, GWT, *Nexaweb*, Canoo e *Backbase*. Conheceremos adiante um pouco sobre cada uma.

Flex

O *Flex*, hoje na versão 3, é uma plataforma desenvolvida pela atual Adobe que executa aplicativos multiplataforma utilizando um *plug-in*, o *Flash Player* que é uma máquina virtual leve.

> Os tipos de projeto disponíveis no *Flex Builder* são:
> Projeto *Flex*. Este tipo de projetos envolve a criação de um aplicativo principal MXML juntamente com quaisquer outros recursos, como as classes ActionScript, imagens e arquivos CSS. O código que você escreve é compilada pelo compilador *Flex* em um arquivo SWF. Um arquivo HTML para o *wrapper* (arquivo compilado) SWF também pode ser opcionalmente gerado para executar o SWF em um navegador da *web*. Projetos *Flex* permitem que você desenvolva usando um editor visual que lhe permite adicionar componentes diretamente para sua aplicação através de *drag and drop* bem como redimensionar visualmente os componentes de layout. Projetos *Flex* também fazem uso de um editor de propriedade que permite modificar as propriedades de seus componentes, como texto rótulos e estilos. Você também pode editar diretamente o código fonte MXML.
> Projeto ActionScript. Este tipo de projetos envolve a criação de um conjunto de classes ActionScript usando o Adobe *Flash* API. Este tipo de projeto não usa o *framework* do *Flex* e, como tal, não inclui um editor visual ou visualização de *design* no *Flex Builder*. Em vez de criar aplicação UI classes usando arquivos MXML e os componentes visuais, um projeto ActionScript consiste tipicamente de componentes visuais e não visuais escritos exclusivamente em código ActionScript. Compila-se estes projetos em *stand alone* gerando arquivos SWF exatamente como uma aplicação *Flex* e depois executa-os no *stand alone* do *Flash Player*. Código ActionScript podem ser utilizados em aplicações *Flex*, aplicativos em *Flash*, aplicações Adobe AIR.
> Projeto de biblioteca *Flex*. Este tipo de projetos envolve a criação de arquivos MXML, classes ActionScript, e outros recursos que são compilados e empacotados em um arquivo SWC. O arquivo SWC em uma aplicação *Flex* é muito parecido com um arquivo JAR em uma aplicação *Java*. Ele atua como uma biblioteca cujos recursos são disponibilizados para uma aplicação *Flash* ou *Flex* em tempo de execução. Uma biblioteca pode ser usada de diferentes maneiras. Primeiro, os recursos da biblioteca podem ser incluídos no arquivo SWF de um aplicativo quando o aplicativo é compilado. Neste caso, apenas os recursos específicos necessários à aplicação estão incluídos no arquivo SWF

> compilado. Segundo, a biblioteca pode ser implementada com o aplicativo e os recursos na biblioteca acessados em tempo de execução pelo aplicativo. Esta é uma maneira muito semelhante à forma do *Java* fazer as coisas. (KEEFE, 2009, p.36)

Flex é baseado em uma linguagem de programação orientada a objetos chamada ActionScript. Esta é baseada nas últimas Especificação do ECMAScript (padronização internacional para as linguagens de programação em scripts) e uma linguagem de marcação de texto MXML que na verdade é um objeto ActionScript em formato XML.

No lado servidor a plataforma *Flex* integra bem com servidores *Java*, *ColdFusion*, PHP, Ruby, ASP. Para desenvolvimento em *Flex* basta utilizar o SDK; caso esteja em busca de uma IDE existe o Adobe *Flex Builder* construída sobre a plataforma Eclipse além de *plug-in* para Netbeans.

Java

Como já mencionamos, os *Applets* foram as primeiras plataformas RIA; na época a principal razão para a não aceitação dos *Applets* foi robustes da máquina virtual *Java* (JVM). Já o *Java Swing* é uma tecnologia muito madura e robusta para criar aplicações *web* ou *desktop*. Pode-se fazer qualquer coisa com *Java Swing*, porém o ciclo de desenvolvimento é longo, são necessários profissionais especialistas; normalmente os projetos são muito caros para construir e manter.

A SUN acaba de tirar do forno o *Java*FX que promete ser a solução *Java* definitiva para tecnologia RIA. No capítulo 3 falaremos mais a fundo sobre *Swing* e *Java*FX.

Silverligth

A Microsoft vem investindo na tecnologia RIA deste o *Windows Foundation Plataform* (WPF). Seu sucessor, o *Silverlight*, encontra-se atualmente na versão 3 e segundo o *site* do fabricante: "*Silverlight* ajuda a criar aplicações *web* ricas que rodam no Mac OS, Windows e Linux". (MICROSOFT, 2009)[2]

Silverlight usa como linguagem de marcação o XAML baseado em XML; o XAML é semelhante ao MXML da Adobe; e a linguagem de programação usada pode ser C#, *Visual Basic*, *Python*, Ruby, AJAX. Para criar aplicativos, os desenvolvedores podem usar IDE *Microsoft Visual Studio* com *framework* .NET.

AJAX

Fain (2007) Nos conta que o primeiro uso conhecido do termo AJAX em público foi por Jesse James Garrett em seu artigo de fevereiro de 2005 "Ajax: *A New Approach to*

[2] http://silverlight.net/

web Applications". AJAX significa *Asynchronous Javascript And XML*. O AJAX não tem uma máquina virtual padrão, cada navegador implementa blocos de AJAX de forma diferente; freqüentemente um aplicativo AJAX implantado vai exigir alterações no código a cada nova versão do navegador e a cada navegador. Vale lembrar que AJAX não é uma plataforma de desenvolvimento e sim um conjunto de técnicas.

OpenLaszlo

OpenLaszlo[3] de Laszlo Systems é um produto de código aberto que permite criar aplicações que podem ser implantados como DHTML ou arquivos *Flash Player*. A capacidade de gerar código DHTML o tornou um bom candidato para o desenvolvimento de aplicações para dispositivos móveis, chamando a atenção da Sun Microsystems que acabou fechando parceria com Laszlo Systems para levar esta tecnologia para o JME se tornando o rival direto do *Adobe Flash Lite*.

Assim como em outras plataformas RIA o OpenLaszlo utiliza uma linguagem declarativa baseada em MXL chamada LZX e a lógica de processamento é codificada em *JavaScript*. Um dos pontos interessantes do OpenLaszlo é que além de ser uma solução de código aberto, ele permite a criação de aplicações para versões antigas do *Flash Player* como a versão 6.0.

GWT

O GWT *Google Web Toolkit*[4] permite que você escreva programas em *Java*, que são automaticamente convertidos em *JavaScript* para que eles possam ser entregues como AJAX.

Esta não é a primeira tentativa de oferecer uma ferramenta que converte *Java* para *JavaScript*. O sucesso do GWT é outro exemplo de como é importante ter o apoio de um fornecedor de software comercial, como o Google, para qualquer iniciativa AJAX. GWT não é um produto de código aberto, mas ele é livre.

Uma característica interessante é que o GWT compila *Java* em várias versões do *JavaScript* para acomodar as necessidades específicas dos diferentes *browsers*. GWT vem com uma biblioteca de COM extensível. O GWT hospedado no navegador da *web* permite aos desenvolvedores *Java* criar e testar suas aplicações sem convertê-los a *JavaScript* até que o aplicativo esteja pronto. Assim sendo, se um desenvolvedor em *Java* por qualquer

[3] http://www.openlaszlo.org

[4] http://code.google.com/webtoolkit/

razão tiver que trabalhar em aplicações AJAX, ele deve considerar o uso do *framework* GWT.

Nexaweb

Nexaweb[5] oferece uma base em *Java thin client* que não requer qualquer instalação adicional sobre o *desktop* do usuário. O estado do aplicativo é controlado por um pequeno *Applet Java* em execução no cliente. Este *Applet* se comunica com a aplicação *Nexaweb Java* EE quando necessário. Para evitar problemas relacionados com a versão do *Java Runtime Environment* instalado com o navegador da *web*, *Nexaweb* utiliza JRE 1.1 para o *Applet*. É apoiada por todos os principais *browsers*. Este *Applet* atualiza parcialmente a página da *web* quando há mudanças de estado nas aplicações. A *interface* de usuário é definida usando XML, e a lógica da aplicação é programada em *Java*. Ele é baseado no editor Eclipse.

Canoo

Canoo[6] oferece uma chamada *UltraLightClient*, que é uma biblioteca *Java* de *server-side proxy* classes semelhante à API *Swing*, fornecendo, por exemplo, *ULCTable* em vez de *JTable*. Basicamente, cada uma dessas classes tem dois pares: um para o servidor e outro para o JRE do cliente. A biblioteca se encarrega da divisão entre o cliente e o servidor, incluindo sincronização e comunicação dessas metades usando um protocolo proprietário. Um componente de apresentação é executado no cliente, enquanto o aplicativo é executado no servidor. O componente de apresentação do cliente pode ser implantado como um *Applet Java* ou aplicação utilizando tecnologia *Java Web Start*. No lado do servidor o aplicativo pode ser implementado como um *Servlet Java* ou um *bean* de sessão *stateful*.

Backbase

Backbase[7] inclui um componente leve escrito completamente em *JavaScript*. Ele carrega sem problemas no início da sessão do cliente e traz no lado cliente *o framework* de apresentação, que inclui ferramentas para manter o estado no cliente, a sincronização cliente-servidor, e carrega os dados elementares. O *framework* de apresentação suporta *drag and drop*, *data binding*, *styling*, *skinning*, e multimídia. *Backbase* também caminha juntamente com as aplicações *web* e permite que você crie uma pseudo página de *interface*

[5] http://www.nexaweb.com

[6] http://www.canoo.com/

[7] http://www.backbase.com

única. Ele vem com uma biblioteca com mais de 50 componentes extras, além do GUI escrito em DHTML, *JavaScript*, CSS e XML.

No mercado de desenvolvimento de software que utiliza tecnologia *Java*, as plataformas RIAs utilizadas são a própria *Java* e o *Flex*.

Capítulo 2
CONCEITUANDO MVC

Para conceituar a arquitetura MVC, é fundamental começarmos antes por padrões de projeto. Padrões de projeto é a reunião de soluções bem sucedidas para problemas conhecidos; estes padrões são aplicados diariamente em diversas áreas, além da informática. O exemplo mais famoso e antigo é a área de engenharia.

> Christopher Alexander afirma: "cada padrão descreve um problema no nosso ambiente e o núcleo da sua solução, de tal forma que você possa usar esta solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira" (AIS+77, pág. x). Muito embora Alexander estivesse falando acerca de padrões, em construções e cidades, o que ele diz é verdadeiro em relação aos padrões de projeto orientados a objetos. Nossas soluções são expressas em termos de objetos e interfaces em vez de paredes e portas, mas no cerne de ambos os tipos de padrões está a solução para um problema num contexto. (GAMMA et al., 2000, p.19)

Na informática os padrões de projetos começaram a ser estudados e aplicados no desenvolvimento de *softwares* na década de 70, década marcada pelo surgimento de *softwares* multiusuários, processamento em tempo real, banco de dados.

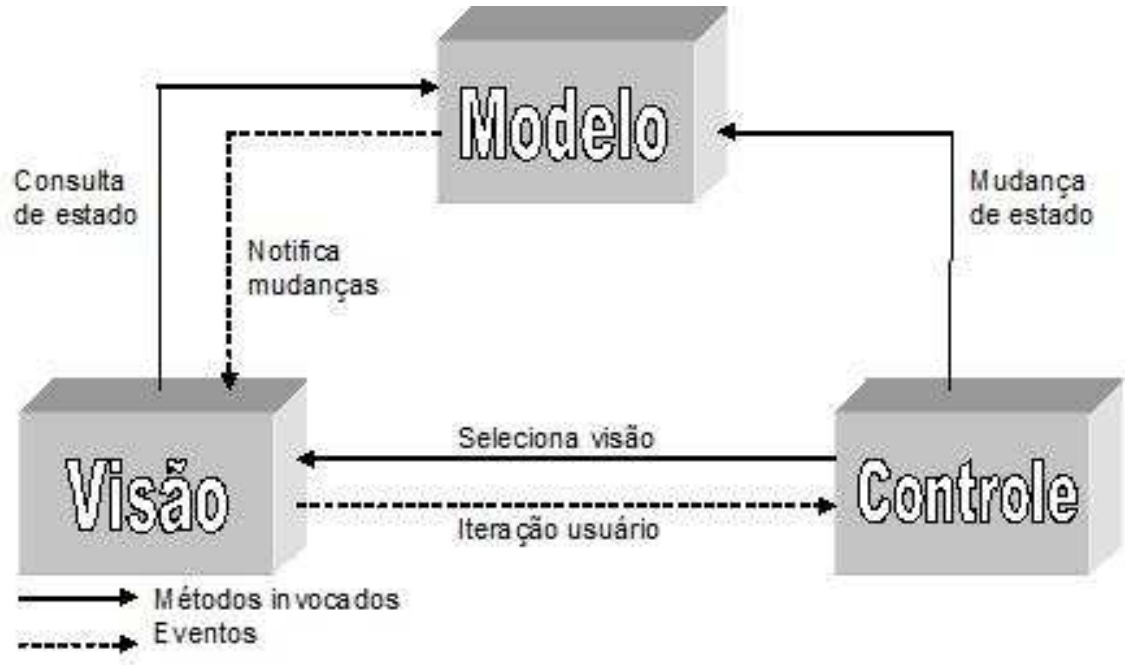
A tecnologia RIA freqüentemente utiliza em sua implementação os padrões MVC, *Value Object, Business Delegate, Command, Singleton, Observer* entre outros. Aqui vamos explorar o MVC.

> MVC é composto por três tipos de objetos. O Modelo é o objeto de aplicação, a Vista é a apresentação na tela e o Controlador define a maneira como a *interface* do usuário reage às entradas do mesmo. Antes do MVC, os projetos de *interface* para o usuário tendiam em agrupar esses objetos. MVC separa esses objetos para aumentar a flexibilidade e a reutilização. (GAMMA et al., 2000, p.20)

A sigla MVC significa *model-view-controller* que comumente traduz-se para modelo, visão e controle. MVC é um padrão de arquitetura que possibilita a utilização de vários padrões de projeto. Freeman (2007) refere-se ao MVC como um padrão composto poderoso.

A arquitetura MVC foi originalmente desenvolvida no *Smalltalk*, linguagem orientada a objeto que surgiu na década de 70 tornando-se popular na década seguinte; hoje a arquitetura MVC está difundida em várias tecnologias. Na tecnologia *Java* existem vários *frameworks* que implementam esta arquitetura, tais como: Apache *Struts*, *Spring* MVC, JSF, entre outros.

Figura 2 – Estrutura MVC[8]



(SUN, 2009 alterado pelo autor)

Modelo é o componente responsável por encapsular o estado da aplicação, por regras de negócio, por regras de acesso e segurança, por atualizações e consulta dos dados, por expor as funcionalidades da aplicação e por notificar ao componente visão as mudanças ocorridas.

Visão é o componente responsável por renderizar informações do componente modelo, formatando, organizando e exibindo o conteúdo, por requisitar atualizações ao componente modelo, por enviar as ações do usuário ao componente controle e pela usabilidade da aplicação pois é o componente de iteração com o usuário.

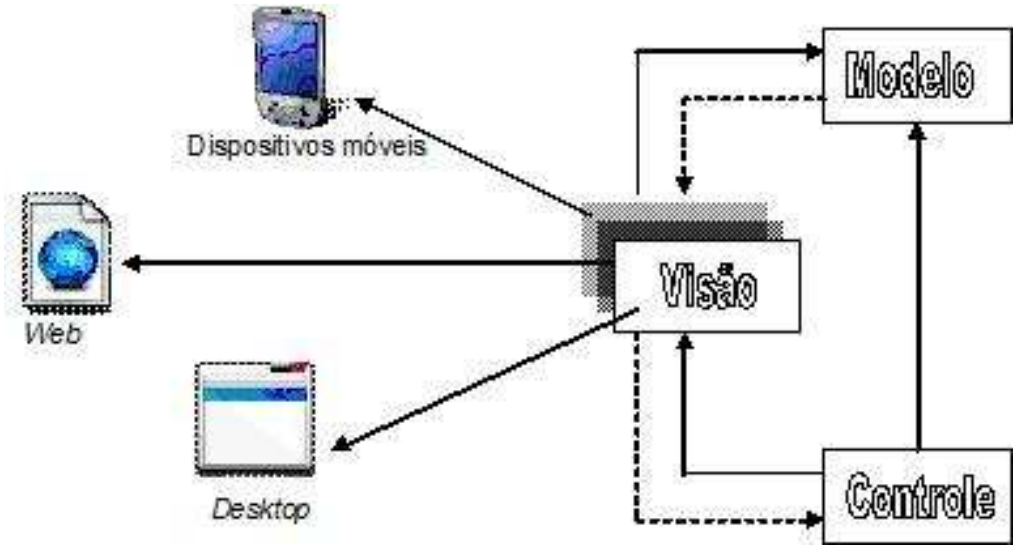
Controle é o componente responsável por definir o comportamento da aplicação, por mapear ações para atualizar o componente modelo, por selecionar o componente visão para exibição; existe um componente controle para cada funcionalidade da aplicação.

A arquitetura MVC possibilita flexibilidade para trabalhar-se em harmonia com uma equipe altamente especialista e heterogênea, permite reusabilidade de código e ganho na produtividade. Um exemplo prático se dá quando um sistema possui duas ou mais interfaces de apresentações diferentes. Adiante exemplificamos um sistema que possui uma visão para *web*, uma visão para dispositivos móveis, uma visão para *desktop*. Os

[8] Site da SUN Microsystems. http://java.sun.com/blueprints/patterns/MVC-detailed.html último acesso em: 15/10/2009.

componentes modelo e controle são reaproveitados mudando apenas os componentes de visão.

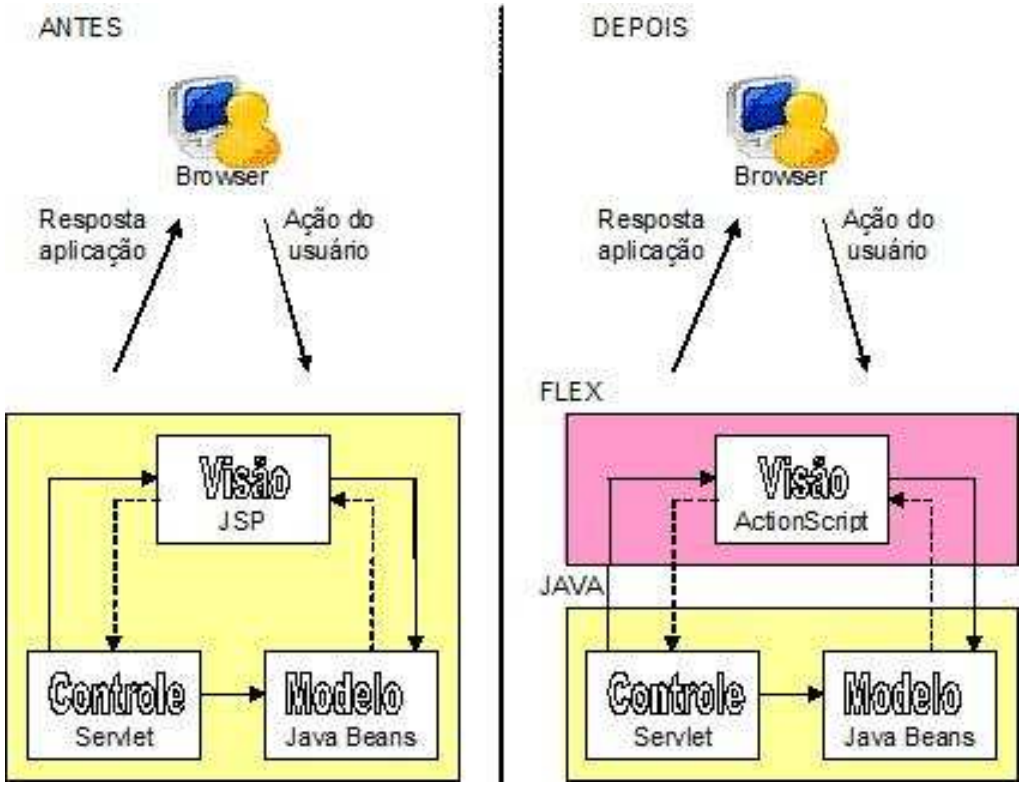
Figura 3 – Aplicação com múltiplos componentes de visão



(SUN, 2009 adaptado pelo autor)

Ainda sobre o componente visão, um exemplo real no nosso contexto é um sistema existente que sofre *upgrade* de plataforma nos componentes visão. Neste, podemos citar um sistema originalmente construído na plataforma *Java* migrando seus componentes de visão para tecnologia RIA, neste caso *Flex*.

Figura 4 - Upgrade de objeto de visão



(FREEMAN, 2007, p.438 adaptado pelo autor)

Capítulo 3
FRAMEWORKS JAVA QUE IMPLEMENTAM MVC

Quais os *framework*s *Java* encontramos disponíveis hoje no mercado para os desenvolvedores? Antes iremos definir: "*Framework* é um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação." (FAYAD, 1997, p.?)

No capítulo anterior, falamos rapidamente sobre padrões de projeto e qual a diferença entre padrões de projeto e *framework*s. Padrões de projeto são abstrações, são conceitos, *framework* inclui código, pode ser modelado utilizando vários padrões de projeto, ou seja, agregando vários conceitos. Como vimos, MVC é uma arquitetura composta por vários padrões mas, ainda sim, conceitual. A seguir comentaremos os diversos *framework*s existentes para a plataforma *Java* que implementam a arquitetura MVC e passaremos pelos preferidos do mercado de desenvolvimento de software aos menos conhecidos, terminando nos projetos brasileiros.

Algumas coisas a se considerar sobre *framework* são que seu objetivo principal sempre é simplificar a programação e aumentar a produtividade. Não existe um *framework* melhor ou pior e sim o que mais se adequa às necessidades de um projeto.

Struts

Encontra-se atualmente na versão 2.1.8[9]. Será o primeiro *framaework* MVC que iremos apresentar. O Apache *Struts* Project é uma comunidade de voluntários que criou e mantém o *framework* Apache *Struts*.

> Apache *Struts* é um *framework open-source* para criar aplicações *web* em *Java*. Diferem das aplicações *web sites* convencionais em aplicações *web* que podem criar uma resposta dinâmica. Muitos *sites* fornecem apenas páginas estáticas. Uma aplicação *web* pode interagir com as bases e os componentes de lógica de negócio para personalizar uma resposta.
> Aplicativos da *web* baseados em *JavaServer Pages*, por vezes, misturam código de banco de dados, código de *design* da página e o código de controle de fluxo. Na prática, vemos que, a menos que estas preocupações sejam separadas, aplicações maiores se tornam difíceis de manter. (EQUIPE APACHE STRUTS, 2009)

[9] http://struts.apache.org/

Spring MVC

O *Spring* surgiu com a proposta de "atender às mais diversas camadas de uma aplicação de forma simplificada, fornecendo suporte a elas e aumentando a testabilidade do software." (RATHIE, 2006). Atualmente encontra-se na versão 3.0.0 RC1[10].

> O *Spring* é bem modular, possuindo diversos serviços que uma aplicação possa necessitar. Um desses módulos, o *Spring* MVC, é um *framework* que implementa o padrão MVC para a camada de apresentação.
> O intuito de criar outro *framework* que implementa o padrão MVC, mesmo já existindo similares como o *Struts* e *Webwork*, visa prover uma solução mais elegante e limpa para a camada de apresentação. (RATHIE, 2006, p.31)

JSF

O *Java Server Faces* é mantido pela comunidade *Java Community Process under JSR - 314*, comunidade comprometida em tornar o JSF num padrão conhecido. É mantido por voluntários, profissionais altamente qualificados e fornecedores. Atualmente encontra-se na versão 2.0[11].

> Facilidade de utilização é o principal objetivo, a arquitetura do *Java*Server Faces define claramente uma separação entre a lógica da aplicação e a apresentação enquanto torna fácil para conectar-se a camada de apresentação ao código do aplicativo. Este projeto permite que cada membro de uma equipe de desenvolvimento de aplicativos *web* possa focalizar a sua parte do processo de desenvolvimento, e também oferece um modelo de programação simples para ligar os pedaços. Por exemplo, os desenvolvedores de página *web* sem conhecimentos de programação pode usar *tags* de componente UI do *Java Server Faces* para ligar o código do aplicativo a partir de uma página *web* sem escrever scripts. (SUN, 2009)

*Java*FX

O *Java*FX foi lançado em dezembro de 2008 e hoje encontra-se na versão 1.2[12]. Trata-se de um produto com pouco tempo de vida e de concepção do projeto mas, segundo especialistas, é um produto estável e promissor.

> A *JavaFX Script* é a linguagem de programação Orientada a Objetos; de "família *Java*" (sintaxe similar na medida do possível, projetada para a JVM e fácil de misturar com *Java*); estaticamente tipada (não é uma linguagem dinâmica"!); compilada para *bytecode* (não interpretada!); com sintaxe de alto nível (como funções de primeira classe).
> *Java*FX também é parcialmente declarativa, com a idéia de dizer "o que" (ex.: SQL) e não "como" (ex.: *Java*). Nesse aspecto é híbrida, pois ainda suporta o paradigma imperativo.
> Além de uma nova linguagem e *framework*, a *Java*FX inclui um completo arsenal de mídia. Existe tanto um "*framework*" multiplataforma, suportando

[10] http://www.springsource.org/

[11] http://java.sun.com/javaee/javaserverfaces/overview.html

[12] http://java.sun.com/javafx/

> formatos de mídia portáveis quanto *framework*s específicos por plataforma. O desenvolvedor de aplicações *Java*FX tem a liberdade de decidir que formato usar. Os formatos não-portáveis podem ter vantagens como maior qualidade, taxa de compressão ou facilidade de produção e de disponibilização através de servidores de *streaming*. Em compensação, podem excluir parte dos seus usuários potenciais – a não ser que você arque com o esforço de produzir e servir a mídia em diversos formatos, selecionando automaticamente conforme a plataforma de cada cliente. (DOEDERLEIN, 2009)

Com o *Java*FX pode-se construir aplicações para *desktop*, *web*, dispositivos móveis e TV digital.

Tapestry

Sua última versão foi liberada em maio deste ano: é a 5.1.0.5[13]. O Apache Tapestry tem a proposta de complementar e reforçar *Java* Servlet API.

> Tapestry divide uma aplicação *web* em um conjunto de páginas, todas construídas a partir de componentes. O desenvolvimento de aplicações Tapestry envolve a criação de *templates* HTML usando HTML puro, e combinando os modelos com pequenas quantidades de código *Java*. Em Tapestry, você cria a sua aplicação utilizando objetos, métodos e propriedades desses objetos - e não especificamente utilizando URLs e parâmetros de consulta. Tapestry traz verdadeiro desenvolvimento orientado a objetos para aplicações *web* em *Java*. (EQUIPE TAPESTRY, 2009)

Tapestry integra-se facilmente com qualquer tipo de *back-end*, incluindo JEE, *Spring* e *Hibernate*.

Apache Click

Atualmente, a versão 2.1.0-rc1[14] é uma moderna estrutura de aplicações JEE *web*, com recursos ricos para a plataforma cliente. Sua proposta de aprendizado é de uma dia. Em sua lista de destaques vale mencionar que suporta Velocity, JSP, FreeMarker e *Java* 1.5 ou superior. “A filosofia de projeto por trás do Apache Click é melhor resumido na *lagom* palavra sueca que se traduz em não muito pouco ou muito, mas o ideal.” (EQUIPE CLICK, 2009)

WebWork

Encontramos um *site* onde a última notícia postada data de 2007 quando foi liberada a versão 2.2.6[15].

> WebWork é um *web Java-framework* de desenvolvimento de aplicativos. É construído especificamente com a produtividade do desenvolvedor e simplicidade de código em mente, fornecendo suporte robusto para a construção de templates de UI reutilizáveis, como controles de formulário, temas IU,

[13] Site Apache Software Foundation http://tapestry.apache.org/tapestry5 ultimo acesso 22/10/2009

[14] Site Apache Incubator http://incubator.apache.org/click/ último acesso 22/10/2009.

[15] Site Open Symphony http://www.opensymphony.com/webwork/ último acesso 30/10/2009

> internacionalização, mapeamento dinâmico de parâmetros de formulário para *Java*Beans, cliente robusto e validação do lado do servidor, e muito mais. (OPEN SYMPHONY, 2009)

O Brasil também produz *framework*. Conheceremos estes a seguir.

Mentawai

O Mentawai encontra-se hoje na versão 1.15[16] liberada em outubro do corrente ano. Pelo movimento no *site* e depoimentos encontrados, este *framework* demonstra ser bem aceito, segundo apresentação no *site*.

> O Mentawai foi o primeiro *framework web* MVC em *Java* a adotar, implementar, documentar e incentivar todo e qualquer tipo de configuração (*actions*, filtros, validação, listas, *connection pooling*, ioc, di, etc.) única e exclusivamente através de configuração programática (100% *Java*), abolindo por completo o uso de XML e *Annotations* para as configurações. O *framework* nasceu em 08/Jun/2005 e logo depois em 18/Jul/2005 publicamos um artigo no *site JavaWorld* enfatizando o uso de configuração programática para o controlador MVC (*actions* / resultados / conseqüências) assim como para validação. Nascia aí o ApplicationManager: configuração em código *Java* independente do restante da sua aplicação e centralizada numa única classe. (EQUIPE MENTAWAI, 2007)

VRaptor

O VRaptor encontra-se hoje na versão 3[17], criado e mantido pela Caelum que mantém uma organizada estrutura de suporte, treinamentos e documentação do produto. Seus benefícios são listados pelo fabricante como:

> Alta Produtividade: Usar o VRaptor 3 é simples e intuitivo. Você atingirá níveis altíssimos de produtividade com *Java* para *web*.
> Curva de Aprendizado: Em pouco tempo você conseguirá aprender tudo o que é necessário para desenvolver suas aplicações com o VRaptor.
> Testabilidade: Escreva código modularizado e desacoplado do VRaptor. Sua aplicação fica altamente testável e de fácil manutenção.
> Economia: Economize muitas horas de trabalho com a alta produtividade do VRaptor, a facilidade em treinar a sua equipe e a qualidade final do seu projeto.
> *Flex*ibilidade: Integre o seu projeto com qualquer *framework* de sua preferência. Você não estará preso a nenhuma tecnologia específica.
> *Soa e Rest – Ready*: Faça aplicações *RESTful* ou orientadas a serviço sem complicações, como se estivesse fazendo aplicações *web* normais.
> Melhores práticas de desenvolvimento: Utilizando os conceitos de Injeção de Dependência, Inversão de Controle e POJOs, seu código fica simples e testável.
> Documentação em Português: Aprenda tudo sobre VRaptor 3 contando com uma ampla documentação, fóruns e listas de discussão em português. (CAELUM, 2009)

[16] http://www.mentaframework.org/

[17] http://www.vraptor.com.br/

NEO Framework

O NEO, hoje na versão 3.3.17[18], foi criado por Rógel Garcia em 2007 que justifica dizendo: "O NEO não propõe mais um *framework*, mas sim o próximo passo dos *framework*s. Tanto que o NEO utiliza como base os *framework*s *Spring* e *Hibernate*. O que o NEO propõe está além do que esses *framework* já propuseram."

O NEO não utiliza XML, os famosos arquivos de configuração; os códigos comuns já vem implementados. NEO fornece DAOs implementados, componentes controles específicos para as situações mais comuns como CRUDs e relatórios; a validação é feita através de *annotations*.

[18] http://www.neoframework.org

CONCLUSÃO

Após vasta pesquisa teórica nos vimos ainda sem condições de responder a todas as questões propostas pos entendermos que na informática muitas das vezes a prática é que nos fornece convicção para fechar um parecer.

Montamos dois laboratórios; um com base em KEEFER (2009) utilizando Eclipse e SDK do FLEX e outro com base em DOEDERLEIN (2009) utilizando NetBeans e plug-ins do JavaFX.

Em ambos os casos, existe um tempo de aprendizado das linguagens de script. Uma vez transcorrido este tempo de aprendizagem, o desenvolvimento se torna rápido e aderente. As IDE's escolhidas para o desenvolvimento se mostraram estáveis, possibilitando um implementação sem transtornos.

Com o tema abordado, as pesquisas teóricas e os cenários testados, chegamos à conclusão que é fácil, rica e aderente a experiência com tecnologia RIA.

Estamos convencidos que a tecnologia RIA chegou para revolucionar e quebrar paradigmas do ramo da usabilidade, de interfaces, de integração homem x computador.

REFERÊNCIAS

ADOBE SYSTEMS INCORPORATED. *Apresentação técnica.* Disponível em: http://www.adobe.com/resources/business/rich_internet_apps/#open Acesso em: 20/10/2009.

APACHE STRUCT. *Apresentação técnica.* Disponível em: http://struts.apache.org/ Acesso em: 27/10/2009.

CAELUM ENSINO E INOVAÇÃO. *Apresentação técnica.* Disponível em: http://www.vraptor.com.br/ Acesso em: 22/10/2009.

DOEDERLEIN, Osvaldo Pinali. *Guerra do RIA. Java* Magazine, Rio de Janeiro, DevMedia Group, ano 6, n. 63, dez. 2008.

DOEDERLEIN, Osvaldo Pinali. *JavaFX: RIA a todo vapor*. *Java* Magazine, Rio de Janeiro, DevMedia Group, ano 7, n. 67, abr. 2009.

DOEDERLEIN, Osvaldo Pinali. *JavaFX: Tutorial. Começando a programar em JavaFX*. *Java* Magazine, Rio de Janeiro, DevMedia Group, ano 7, n. 68, mai. 2009.

FAIN, Yakov; RASPUTNIS, Dr. Victor; TARTAKOVSK, Anatole. *Rich Internet Applications with Adobe Flex & Java (Secrets of the Masters)*. United States of America: SYS-CON Books, 2007.

FAYAD, Mohamed E.; SCHMIDT, Douglas C. *Object-Oriented Application Frameworks*. United States of America: Communications of ACM, vol. 40, n.10, 1997.

FRAGA, Rodrigo Pereira. *Interfaces de qualidade com Adobe Flex*. *Java* Magazine, Rio de Janeiro, DevMedia Group, ano 7, n. 68, abr. 2009.

FREEMAN, Eric; FREEMAN, Elizabeth. *Padrões de Projetos: seu cérebro em padrões de projetos.* Rio de Janeiro: Alta Books, 2007.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos*. Porto Alegre: Bookman, 2000.

KEEFE, Matthew; CHRISTIANSEN, Charles A. *Java and Flex Integration Bible*. Indiana - USA: Wiley Publishing, 2009.

MENTAWAI. *Apresentação técnica.* Disponível em: http://www.mentaframework.org/ Acesso em: 22/10/2009.

MICROSOFT CORPORATION. *Apresentação técnica.* Disponível em: http://silverlight.net/ Acesso em: 21/10/2009.

NEO FRAMEWORK. *Apresentação técnica.* Disponível em: http://www.neoframework.org Acesso em: 22/10/2009.

RATIE, Aman. *Spring MVC*. Mundo *Java*, Curitiba, Mundo, ano 3, n. 20, nov. 2006.

SPRING SOURCE. *Apresentação técnica.* Disponível em: http://www.springsource.org/ Acesso em: 23/10/2009.

SUN MICROSYSTEMS. *Apresentação técnica.* Disponível em: http://java.sun.com/javaee/javaserverfaces/overview.html Acesso em: 27/10/2009.

SUN MICROSYSTEMS. *Apresentação técnica.* Disponível em: http://java.sun.com/javafx/ Acesso em: 23/10/2009.

SUN SYSTEMS. *Java BluePrints Model-View-Controller*. Disponível em: http://java.sun.com/blueprints/patterns/MVC-detailed.html Acessado em: 15/10/2009.

TERRACINI, Fabio; MARTINELLI, Rafael. *Adobe Flex 2*. Mundo *Java*, Curitiba, Mundo, ano 3, n. 20, nov. 2006.

GLOSSÁRIO

Adobe AIR – AIR (*Adobe Integrated Runtime*) é um *framework* de desenvolvimendo para criação de aplicativos desktop.

Adobe Flash Lite - é um aplicativo *Flash* especialmente desenvolvido para telefones celulares e dispositivos portáteis.

API – (*Application Programming Interface* ou Interface de Programação de Aplicativos) é um conjunto de rotinas e padrões estabelecidos por um *software* para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços.

Applet - Componente *Java* independente e compilado, que é executado no contexto de um navegador.

Arquitetura cliente-servidor - é toda arquitetura de rede onde estações (microcomputadores) executam aplicações clientes que se utilizam de programas servidores para transferência de dados do próprio servidor ou comunicação com outras estações e suas aplicações clientes.

Back-end - nas aplicações cliente-servidor, *back-end* refere-se à parte do programa relativa ao servidor, sendo o *front-end* à parte do programa relativa ao cliente.

Browser – é o nome genérico do programa que nos permite navegar na Internet.

Bytecode - código gerado pelo compilador de *Java* e executado pelo interpretador de *Java*. O código é independente da máquina.

CRUD - é o acrônimo da expressão em inglês de *Create, Retrieve, Update* e *Delete*, usada para definir quatro operações básicas usadas em banco de dados ou em interface para usuários para criação, consulta, atualização e destruição de dados.

Drag and drop – nas *interfaces* gráficas e componentes, *drag-and-drop* (arrastar e largar) é a ação de clicar em um objeto virtual e "arrastá-lo" a uma posição diferente ou sobre um outro objeto virtual.

GUI – (*Graphic User Interface*) interface gráfica que procura facilitar a comunicação homem / máquina, lançando mão de recursos visuais a leitura, como ícones, menus interativos, etc.

Java Web Start - também conhecido como Javaws ou como javaws, é um *framework* desenvolvido pela Sun Microsystems.

JavaWorld – revista norte-americana especializada em tecnologia Java.

JEE - *Java* EE ou J2EE, ou *Java 2 Enterprise Edition* (*Java* Edição Empresarial) é uma plataforma de programação para servidores na linguagem de programação *Java*.

JME - *Java Plataform Micro Edition, Java* ME, ou ainda J2ME, é uma tecnologia que possibilita o desenvolvimento de *software* para sistemas e aplicações embarcadas, ou seja, toda aquela que roda em um dispositivo de propósito específico, desempenhando alguma tarefa que seja útil para o dispositivo.

JRE - *Java Runtime Environment* significa Ambiente de Tempo de Execução Java, e é utilizado para executar as aplicações da plataforma *Java*. É composto por bibliotecas (APIs) e pela máquina virtual *Java* (JVM).

JTable - a classe JTable é utilizada para vizualizar dados em grid no Swing e é um dos componentes mais complexos desse pacote (javax.swing.table).

Mainframe - designação dada aos antigos computadores de grandes porte e desempenho.

Open source - significa código fonte aberto. Programas *open source* divulgam seus códigos fonte para que qualquer pessoa com conhecimento de programação possa modificá-lo, ampliá-lo e melhorá-lo.

SDK - é a sigla de Software Development Kit, ou seja, Kit de Desenvolvimento de Software, pacote que inclui ferramentas (APIs, linguagens de scripting e interface gráfica de usuário).

SOA - *Service Oriented Srchitecture* (Arquitetura Orientada a Serviços)

Stand alone - que funciona sozinho, se refere a um dispositivo qualquer que não precisa de outros para funcionar. Por exemplo, um Palm ou um notebook, são dispositivos *stand alone*, que são funcionais sozinhos. Já um monitor ou uma impressora não podem ser considerados dispositivos *stand alone*, pois só funcionam conectados ao host.

Stateful session bean - é uma enterprise bean (componente EJB) que atua como um servidor de extensão do lado do cliente. O *bean* de sessão *stateful* é criado por um cliente e vai trabalhar para que apenas o cliente até que, a conexão do cliente seja derrubada ou o bean seja explicitamente removido.

Template – gabarito ou modelo. Página usada como base do desenho de todas as páginas num *site*.

Thin client - ("cliente magro") é um computador cliente em uma rede de modelo cliente - servidor de duas camadas o qual tem poucos ou nenhum aplicativo instalado, de modo que depende primariamente de um servidor central para o processamento de atividades.

UI - *interface* de usuário (também conhecido como *interface* homem-computador ou *interface* homem-máquina) é o conjunto de meios pelos quais as pessoas (usuários) interagem com o sistema.

ULCTable - componente exibe uma lista de objetos na forma de uma tabela de rolagem contendo várias colunas (com.ulcjava.base.application.ULCTable).

UltraLightClient - é um *widget toolkit* e *framework* para a criação de Rich Internet Applications em Java.

Upgrade - é um jargão utilizado em computação com significado de atualizar, modernizar

URL - (*Uniform Resource Locator* ou *Universal Resource Locator*), em português Localizador de Recursos Universal, é o endereço de um recurso disponível em uma rede; seja a Internet ou uma rede corporativa.