Uma Abordagem Prática sobre a Aplicação do padrão MVC com o Framework Struts

Glauber da Rocha Balthazar, Fábio Mendes Ramos Guimarães, Melise Maria Veiga de Paula, Elio Lovisi Filho

Bacharelado em Sistemas de Informação – Faculdade Metodista Granbery (FMG)
Rua Batista de Oliveira, 1145 - 36010-532 - Juiz de Fora - MG

{gbalthazar,fguimaraes}@si.granbery.edu.br, elio …., melisepaula@gmail.com,

Abstract. *The objective of this work is to present a study of the architectural pattern used in software engineering, the MVC, through of the Struts framework. The essence of this work is an analysis of the main aspects that surround this environment, considering the advantages and possible problems with this utilization.*

Resumo. *O objetivo deste trabalho é apresentar um estudo do padrão de desenvolvimento de aplicações orientadas a objetos, denominado MVC (Model View Controller), através do framework Struts. O que constituirá a essência deste trabalho é uma análise dos principais aspectos do ambiente definido, considerando as vantagens e desvantagens na adoção deste padrão.*

INTRODUÇÃO

Atualmente, a globalização e a necessidade, cada vez maior, de inovação para manter a competitividade, estabelecem um novo cenário que define a gerência das informações e conhecimento relacionados aos processos executados nas organizações como uma das principais atividades [PEREIRA & GOMES].

O conceito de CPD (Centro de Processamento de Dados), ambiente no qual se tinha um profissional de informática conhecido como operador de computador que era o responsável por inserir os dados (em lotes) nas máquinas e gerar relatórios dos resultados de forma independente do restante da empresa, sem ligações diretas entre usuário, tecnologia e informação, já se tornou obsoleta.

A idéia do ambiente único e centralizado, com funções meramente operacionais já está desconsiderada. Desta forma, surge então um novo ambiente no qual a gerência da informação deve ser realizada de forma a se estabelecer como uma parceira direta nas tomadas de decisões da empresa [REIS].

Além disso, pode-se observar também, significativas mudanças no desenvolvimento de software. De acordo com [PRESSMAN], o software se tornou a força motora de uma empresa, sendo responsável pelo gerenciamento do fluxo de informações e por isso, pode ser visto como um diferencial entre as organizações. O aumento da complexidade do processo de desenvolvimento de software é outro aspecto que se destaca no cenário atual. Atualmente, o software não é mais desenvolvido para atender exclusivamente a um problema de um departamento exclusivo e a informação é compartilhada entre todos os setores e segmentos de uma organização [PRESSMAN].

A partir destas constatações, pode-se concluir que, a organização que consegue trabalhar com sistemas de informação, baseado em um software com alta manutenibilidade, disponibilidade e segurança, tende a aumentar a sua competitividade perante seus concorrentes.

Contudo, analisando a realidade da grande maioria das organizações, é possível perceber que ainda não existe um esforço definitivo por parte dos gestores de tecnologia para definir um planejamento formal que seja documentado, discutido, aceito e exposto a todos os envolvidos em um projeto com o objetivo de estabelecer uma estratégia de implantação e atuação das tecnologias considerando as informações que fluem nos processos [REIS].

Neste contexto, alguns conceitos vêm se destacando como a orientação a objetos, desenvolvimento em camadas, a utilização de padrões de projeto e *frameworks*. Como exemplos, pode-se citar o desenvolvimento de camadas de persistência (*frameworks* de persistência) que são utilizados como alternativa para reduzir o descasamento de impedância entre aplicações orientadas a objetos que manipulam banco de dados relacionais e, os Padrões de Projeto (*Design Patterns*), que podem ser definidos como soluções para problemas que alguém um dia teve e resolveu aplicando um modelo que foi documentado e que pode ser adaptado integralmente, ou de acordo com necessidade, em outras soluções [MACORATTI].

Motivado pelas constatações apresentadas anteriormente, este artigo trás como objetivo apresentar uma alternativa para o desenvolvimento de aplicações orientadas a objetos a partir da adoção do padrão MVC (*Model, View e Controller*). A visão prática foi desenvolvida a partir da utilização do Struts, um *framework* de desenvolvimento que implementa o padrão MVC.

O artigo está organizado da seguinte forma: a Seção 2 apresenta uma visão geral do modelo MVC de desenvolvimento com suas características e formas de aplicação. Na Seção 3 é visto o *framework* Struts que disponibiliza ao desenvolvedor uma forma de aplicação do MVC. Em seguida, na Seção 4 é apresentado um breve tutorial sobre a utilização do Struts no Eclipse.

O MODELO MVC DE DESENVOLVIMENTO

Um dos principais objetivos do padrão MVC é a organização do código de uma aplicação em camadas, realizando assim a separação física dos componentes[1] do software. Desta forma, a organização em camadas é a chave para a independência entre os componentes, objetivando desta forma agrupar componentes por responsabilidades em comum [FRAGMENTAL].

A fundamentação da divisão das funcionalidades de um sistema em camadas surgiu como alternativa para solucionar alguns problemas existentes nas aplicações monolíticas, nas quais, dados e código eram armazenados em uma mesma máquina, na qual todas as funcionalidades eram definidas em um único módulo contendo uma grande quantidade de linhas de código e de difícil manutenção [MACORATTI].

A necessidade de compartilhar a lógica de acesso aos dados entre vários usuários, impulsionou o desenvolvimento de aplicações em duas camadas. Nesta estrutura, a base de dados é armazenada em uma máquina específica (servidor) diferente das máquinas que

[1] Componente, neste artigo, é descrito como qualquer artefato de software como classe e objeto [FRAGMENTAL].

executam as aplicações (clientes). Um problema desta abordagem é o gerenciamento de versões, pois para cada alteração de uma das regras implementadas obriga a atualização dos aplicativos em todas as máquinas clientes [MACORATTI].

Porém, a Internet trouxe para os usuários e desenvolvedores uma nova visão de aplicativos. Neste novo paradigma, a aplicação (vista na Internet como *site*) é disponibilizada por meio de um servidor *web* no qual o usuário apenas realiza requisições. As requisições são processadas no servidor *web* e as respostas são enviadas ao usuário.

Desta forma, surgiu a necessidade de responder, dinamicamente, às solicitações dos usuários, característica essa que contribuiu para os esforços dos desenvolvedores em separar a lógica de negócio da interface com o usuário, ressurgindo o modelo MVC de desenvolvimento. Essa idéia não é nova, tendo se popularizada a partir da década de 90 em um modelo na linguagem Smalltalk [FRAGMENTAL].

Este modelo consiste em uma tríade de classes freqüentemente usadas em sistemas interativos para construção de interfaces com o usuário. A implementação deste modelo mantém o núcleo funcional do sistema independente da interface. Assim, as interfaces internas podem permanecer estáveis, mesmo quando a interface necessita ser alterada para se adaptar a novas plataformas e dispositivos de interação [BUSCHMANN].

Com isso a apresentação, a lógica (negócio) e o acesso ao banco de dados estão separados em camadas específicas, tornando os sistemas mais manuteníveis e garantindo a independência entre estas camadas. Desta forma, as camadas de negócio podem ser divididas em classes podendo ser agrupadas em pacotes ou componentes reduzindo as dependências entre as mesmas. Esta divisão facilita a reutilização por diferentes partes do aplicativo e até por aplicativos diferentes. O modelo MVC se aplica como uma excelente arquitetura para o desenvolvimento de sistemas corporativos com base na *web*.

O MVC é apresentado na Figura 1 que representa as camadas de separação física e lógica proposta pelo modelo. Na camada *View*, é apresentada todas as interfaces que interagem com o usuário permitindo a realização de requisições – entrada de dados. Na camada *Controller*, é desenvolvido toda a parte que controla o comportamento da aplicação, servindo como uma intermediária entre a *View* e o *Model*. E por último, a camada *Model* é a responsável por conter o código da ação que foi requisitada como, por exemplo, se comunicar com outra camada que busque dados em um banco [BUSCHMANN].

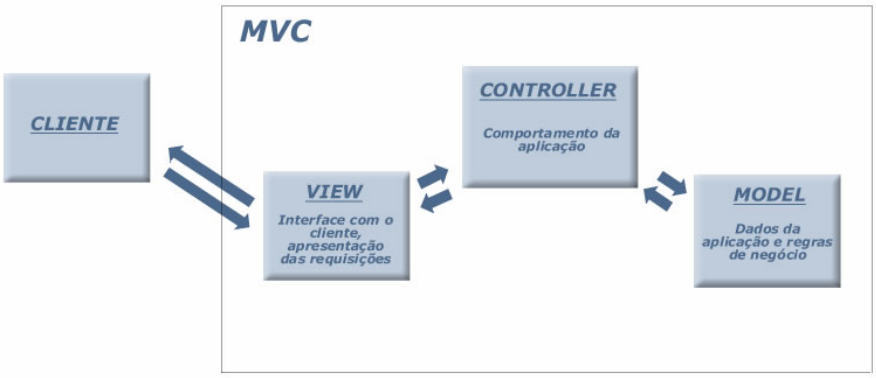

Figura 1 – O Modelo MVC

As vantagens em se utilizar o MVC no desenvolvimento de uma aplicação estão diretamente relacionadas com a manutenibilidade que se consegue atingir quando se divide um sistema em módulos específicos, que tratam de responsabilidades diferentes e únicas.

Ainda, dentro deste contexto, a reusabilidade é bem explorada, pois uma mesma funcionalidade que é desenvolvida para uma janela (ou página *web*), pode ser reutilizada em outras janelas que a requisitem.

Porém, a maior desvantagem deste modelo é o auto custo no desenvolvimento de uma aplicação, pois, o processo tende a levar mais tempo, tendo em vista que todas as classes deverão ser organizadas em pacotes – representação física das camadas - que representem a sua funcionalidade. Isso obriga aos desenvolvedores a seguir um padrão no desenvolvimento de qualquer parte da aplicação, aumentando desta forma o prazo de entrega do produto.

Ainda, uma outra desvantagem é que este padrão exige a presença de um profissional especializado que domine os conceitos apresentados, aumentando os gastos com treinamentos e conscientização da efetiva adoção do padrão.

O FRAMEWORK STRUTS

Baseado na idéia de separar a lógica de negócio da apresentação, a Apache *Software Fundation* desenvolveu o Struts que é um *MVC-style Controller open-source (framework web* para implementação do modelo MVC), construído para aplicações WEB implementadas na linguagem Java [STRUTS], ou seja, o Struts é uma coleção de funcionalidades que ajudam o desenvolvedor na construção de aplicações web, implementando o padrão MVC [HUDSON].

De acordo com [STRUTS], este *framework* oferece três conceitos fundamentais. São eles:

- funcionalidade de "*front controller*" (controlador primário) que dispara requisições para um "*action*" - uma classe que é construída pelo desenvolvedor com a funcionalidade de responder às requisições;
- "*location handler*" que transfere o controle do código para outro recurso que processe os dados;
- e permitir a construção de "*tag library's*" que ajudam os desenvolvedores a construírem bibliotecas de códigos que podem ser reaproveitadas durante toda a aplicação.

A Figura 2 apresenta uma comparação entre o modelo MVC e o funcionamento do Struts. Na *View* são construídas as páginas JSP e as *TagLibs*. No *Controller* são desenvolvidas as *ActionForms*, que são páginas responsáveis por instanciar objetos e associar, após uma validação, os dados informados na JSP com um Objeto e persistir ou recuperar dados no banco. Por último, no *Model*, têm-se a comunicação com a camada responsável pelo acesso ao banco, ou com outras que processem os dados.

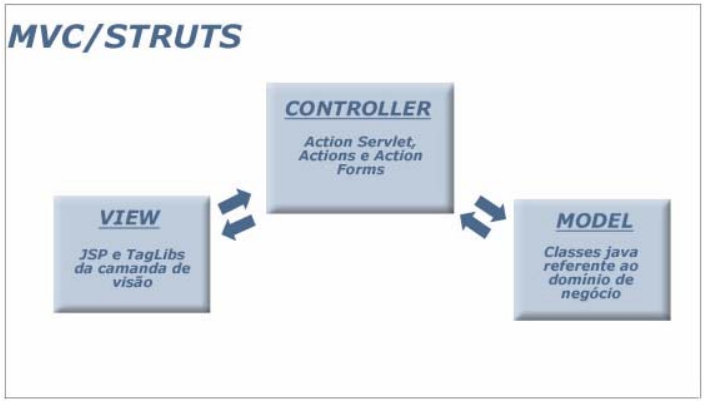

Figura 2 – Comparação entre o modelo MVC e o Struts

O Struts é constituído de componentes conhecidos como *Controller Components*, que são um conjunto de componentes programáveis que permitem aos desenvolvedores definirem exatamente como a aplicação deverá interagir com o usuário. Desta forma, o desenvolvedor interage com estes componentes programando-os para agir da melhor forma que desejam, enquanto que os usuários da aplicação interagem através dos *hyperlinks* e dos formulários em HTML. Os *hyperlinks* conduzem às páginas que indicam dados e outros elementos, tais como texto e as imagens. Os formulários submetem geralmente dados ao servidor *web* através de algum tipo de ação escrita pelo desenvolvedor [HUDSON].

Desta forma, o Struts implementa o padrão MVC com a utilização de ActionForwards e ActionMappings para manter o controle de fluxo de decisões fora da camada de apresentação. Isso é realizado através dos componentes principais do padrão MVC implementados no Struts, que são apresentados na Tabela 1 [HUDSON].

Componente	Descrição
ActionForward	A ação do usuário
ActionForm	Os dados da ação do usuário
ActionMapping	O responsável pelo mapeamento dos eventos da ação do usuário
ActionServlet	A parte do Controlador que recebe as ações do usuário
Action classes	A parte do controlador que interage com o modelo para executar a ação do usuário ou executar uma query, e direcionar o ActionServlet para a próxima View a ser executada

Tabela 1 – Principais componentes do padrão MVC implementados no Struts

Em adição à estes componentes, o Struts usa um número de arquivos de configuração e de ajuda para construir uma ligação entre o Controller e o Model. A Tabela 2 apresenta os arquivos de configuração do e descreve seu papel na arquitetura [HUDSON].

File	Propósito
ApplicationResources.properties	Armazena mensagens de modo que sua aplicação possa ser internacionalizada
struts-config.xml	Armazena a configuração padrão para os objetos do controlador, que inclui as ações do usuário, as mudanças de estado, e as perguntas do estado suportadas no padrão

Tabela 2 – Principais arquivos de configuração do Struts

Para exibir os dados na configuração do Struts para a utilização, o *framework* fornece um número de tag's que podem ser utilizadas no formulário JSP, conforme mostrado na Tabela 3 [HUDSON].

File	Propósito
struts-html.tld	Extensão JSP tag para os formulários HTML
struts-bean.tld	Extensão JSP tag para apresentar JavaBeans
struts-logic.tld	Extensão JSP tag para testar os valores das propriedades

Tabela 3 – Principais Tags do Struts que podem ser utilizadas nos formulários JSP

Então, todos estes arquivos, componentes e tags caracterizam o *framework* Struts permitindo o seu funcionamento em aplicações *web*. Mas não é fácil representar o verdadeiro fluxo destas características quando executadas em aplicações. Dependendo das circunstâncias, várias ações distintas podem ocorrer de formas especialmente diferentes nas aplicações *web*. Mas há ainda uma ordem geral às coisas que pode ser apresentada neste ponto [HUDSON].

Desta forma, o funcionamento interno do Struts é visualizado na Figura 3. Nessa figura é visto que (1) o usuário envia requisições para um *ActionServlet* que, (2) baseado nas configurações já previamente descritas no *struts-config*.xml, (3) realiza a validação dos dados fornecidos nas requisições através de um *ActionForm* específico. Em seguida, (4) caso a validação tenha ocorrido com sucesso, (5) os dados serão instanciados dentro de um *Action* que será o responsável por enviar o objeto para a camada (6) que irá persistir ou recuperar objetos em um banco de dados, por exemplo. Ao realizar tal ação (7) o objeto será devolvido para o *Action* que imeditamente, de acordo com uma ação definida em um *forward* (8), irá entregar os dados para a apresentação (9), sendo o usuário atendido na sua requisição inicial (10) [PORTAL JAVA].

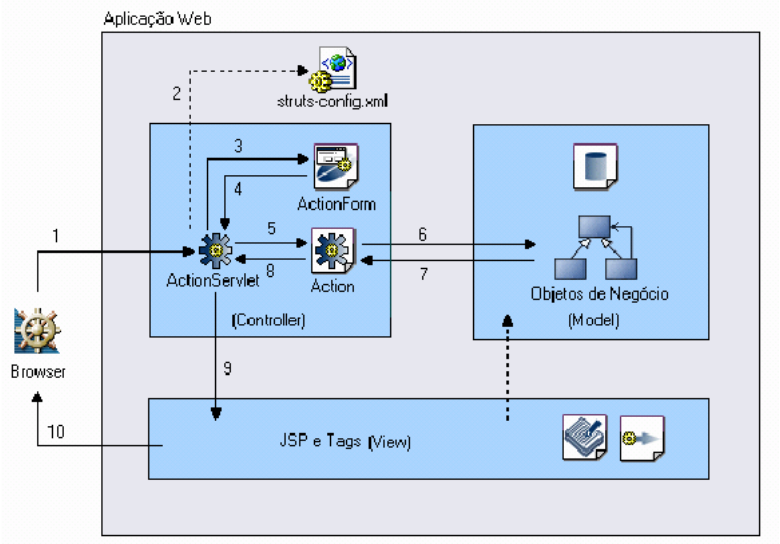

Figura 3 – Visão do funcionamento interno do Struts [PORTAL JAVA]

Contudo, o Struts apresenta algumas desvantagens, podendo ser destacada como principal uma maior complexidade envolvida no processo de desenvolvimento e, além disso, qualquer camada a mais em uma aplicação gera custos no processamento das informações e a presença de um profissional treinado e capacitado, tendo em vista que as requisições não serão atendidas diretamente pelas classes responsáveis pelo processamento dos dados.

[HUDSON] cita que a aprendizagem do Struts, em um primeiro momento, é complexa. Desta forma, a empresa que pretenda utilizar este framework, tem que capacitar os seus profissionais para, além de trabalhar neste *framework*, segui-lo fielmente, evitando a construção de códigos em partes do sistema que fujam do padrão MVC, sendo escritos em camadas que não representem o local ideal para eles.

VISÃO PRÁTICA DO STRUTS

Após a apresentação do modelo MVC e como o Struts implementa este padrão no desenvolvimento de aplicações *web*, é apresentada uma visão da forma de implementação do MVC no Struts em uma aplicação Web Java [DEITEL, DEITEL] no Eclipse. Para isso, será utilizada uma aplicação que irá cadastrar um funcionário de uma empresa.

Deve-se, então, ter instalado no computador o Eclipse e ser baixado no site da Apache (http://struts.apache.org) o Struts. Em seguida, todos os arquivos que se encontram na pasta *lib* do Struts deverão ser copiadas para o projeto.

Para a parte de visualização (*View*), será construída uma página JSP que contém um formulário de cadastro de funcionário como mostrado na Figura 4.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@taglib uri="/WEB-INF/taglib.tld" prefix="minhasTags" %>
<%@taglib uri="/tags/struts-html" prefix="html" %>
<%@taglib uri="/tags/struts-logic" prefix="logic" %>
<%@taglib uri="/tags/struts-bean" prefix="bean" %>
<html>
<body>
<hr>
  <b class="titulo">
    Incluir novo Funcionário no sistema
  </b>
<hr>
<html:form action="/" method="post">
    Nome:
    <html:text property="nome"></html:text><br>
    Idade:
    <html:text property="idade"></html:text><br>
    Salário:
    <html:text property="salario"></html:text><br>
    Cargo:
    <html:text property="cargo"></html:text><br>
<html:submit property="btnOK" value="Incluir" ></html:submit>
<html:button property="btnCcancelar" onclick="history.back()"></html:button>
</html:form>
</body>
</html>
```

Figura 5 – Página JSP que está na camada de apresentação (View)

Esta página importa através das cláusulas "uri", descritas no início do código todas as TagLib's que poderão ser utilizadas na página. Neste exemplo, utilizou-se a notação do próprio Struts para a construção do formulário através da *tag* de *prefix* "html".

Após isso, deve-se definir os mapeamentos da aplicação para a validação em um *Form* e ação em um *Action* na camada de Controle. Desta forma, se fará necessário a

construção de uma classe Java *Bean* que receba todos os dados vindos do formulário como mostrado na Figura 6. Nessa figura foram omitidos os métodos de acesso (*getters* e *setters)*.

```
package br.com.sige.struts.forms;
import org.apache.struts.action.ActionForm;

public class FuncionarioForm extends ActionForm {
    private String nome;
    private Integer idade;
    private Double salario;
    private String cargo;

    public FuncionarioForm() {
        super();
    }
}
```

Figura 6 – Classe FuncionarioForm que recebe os dados vindos do formulário

Em seguida, esta classe deverá ser mapeada dentro do arquivo *struts-config*.xml na *tag Form Mapping* .

A partir do momento em que os dados estiverem validados, um *Action-Mapping* deverá ser construído para a execução da aplicação. Um *Action-Mapping* define uma classe que irá implementar uma ação nos dados passados como parâmetro, neste caso o cadastro no banco, e após esta ação, redirecionar o usuário para uma página de sucesso ou falha. Desta forma, duas páginas JSP deverão ser construídas, uma para o sucesso (sucesso.jsp) e outra para o fracasso (fracasso.jsp). Neste ponto o Struts está executando aplicação na camada de Controle, que é responsável por mapear e controlar as ações. A construção do *Action-Mapping* pode ser visto na Figura 7.

```
<action
    path="/incluirFuncionario"
    type="br.com.strutsTurorial.struts.actions.FuncionarioAction"
    name="incluirFuncionarioAction"
    scope="request"
    unknown="true"
    validate="false"
    input="/Funcionario.jsp">
    <forward
        name="sucesso"
        path="/sucesso.jsp"
        redirect="false"
        contextRelative="false" />
    <forward
        name="falha"
        path="/falha.jsp"
        redirect="false"
        contextRelative="false" />
</action>
```

Figura 7 – Mapeamento do *Action-Form* no arquivo struts-config.xml

Contudo, a página JSP está com o direcionamento do *action* vazio (*html*: *form action*=”/” *method*=”*post*”>). Este *action* deverá estar direcionando para o *Action-Mapping* construído anteriormente (*html*: *form action*=” ” *method*=”*post*”>).

Agora uma classe *Action* deverá ser construída. Esta irá receber do *Form* definido anteriormente, e será possível a instância do objeto que irá ser persistido no banco. Neste ponto o *Struts* está direcionando a execução da aplicação para a camada de Modelo que é responsável por manipular dados em um banco de dados. Dentro desta camada deverá existir uma classe responsável por realizar a inclusão, alteração, consulta ou exclusão.

Esta *Action* irá devolver para o mapeamento *Action-Mapping* do *struts-config*.xml uma resultado que representa o sucesso ou o fracasso na inclusão do funcionário e o próprio *Action-Mapping* é que fará o direcionamento de acordo com o resultado.

CONSIDERAÇÕES FINAIS

Neste artigo foi apresentado o modelo MVC no desenvolvimento de aplicações orientadas a objetos, com o principal objetivo de dividir as funcionalidades de um sistema em camadas, atribuindo a cada uma responsabilidades específicas, contribuindo desta forma para a manutenibilidade futura do sistema, permitindo que outras partes venham a ser desenvolvidas apenas acoplando-se ao que já existe.

Também foi apresentada uma visão prática deste modelo de desenvolvimento através de uma descrição da utilização do *framework* Struts.

Por fim, foram apresentadas as vantagens e desvantagens da adoção deste modelo de desenvolvimento e da adoção do Struts como meio para isso.

AGRADECIMENTOS

Os autores reconhecem o apoio dado pela Faculdade Metodista Granbery ao grupo de pesquisa "J-Tec – Grupo de Estudos da Tecnologia Java"ao qual este trabalho está vinculado.

REFERÊNCIAS BIBLIOGRÁFICAS

PEREIRA, André, GOMES, Elisabeth e CAVALCANTI, Marcos. **Gestão de Empresas na Sociedade do Conhecimento: um Roteiro para Ação.** Campus: Rio de Janeiro, 2001.

REIS, Carlos. **Planejamento Estratégico de Sistemas de Informação**. Lisboa: Presença, 1993.

HUSTED, T. **Struts em ação**. 1. ed. Rio de Janeiro: Ciência Moderna, 2004.

PRESSMAN, Roger S. **Engenharia de Software**. 5. ed. Rio de Janeiro: McGraw-Hill, 2002.

DEITEL, H. M., DEITEL, P. J. **Java: Como Programar**, 4. ed. Editora Bookman, Porto Alegre, 2002.

BUSCHMANN, F. et. al. **Pattern-Oriented Software Architeture: A System of Patterns"**. Chichester, UK: John Wiley & Sons.

HUSTED, Ted**. Struts em ação**. Editora Ciência Moderna, Rio de Janeiro, 2004.

MACORATTI, J. Carlos – **Artigos de Tecnologia da Informação**, <www.macoratti.net> Acessado em 18 de setembro de 2006.

FRAGMENTAL Tecnologia – **Artigo MVC e Camadas**, <http://.fragmental.com.br/wiki/index.php?title=MVC_e_Camadas> Acessado em 2 de setembro de 2006.

STRUTS - **Apache Software Fundation,** <http://struts.apache.org/> Acessado em 26 de agosto de 2006.

PORTAL JAVA – **Portal Java – A maior comunidade Java do Brasil**, <www.portaljava.com.br> Acessado em 18 de setembro de 2006.