Padrões de Projeto J2EE 5

Estudo de Caso 1 (Jakarta Struts)

Helder da Rocha (helder@acm.org)

argonavis.com.br

Objetivos

- *Apresentar uma visão geral do Framework Jakarta Struts*
- *Identificar os principais padrões de projeto J2EE encontrados na implementação do Struts e nas aplicações implementadas usando Struts*

O que é Struts?

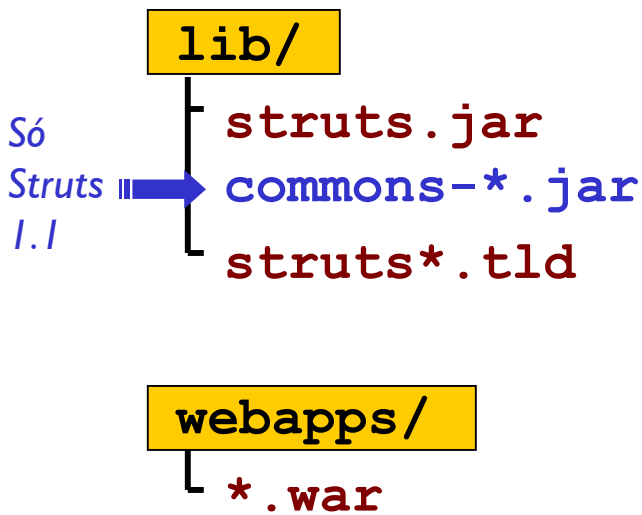
- *Framework para facilitar a implementação dos padrões da camada de apresentação em aplicações JSP*
- *Oferece*
  - *Um servlet controlador configurável através de documentos XML externos, que despacham requisições a classes Action (Command pattern) criadas pelo desenvolvedor*
  - *Uma vasta coleção de bibliotecas de tags JSP (taglibs)*
  - *Classes utilitárias que oferecem suporte a tratamento de XML, preenchimento de JavaBeans e gerenciamento externo do conteúdo de interfaces do usuário*
- *Onde obter: jakarta.apache.org/struts*

Struts como arquitetura MVC

- *Model (M)*
  - *Os Value Beans (View Helpers que têm o papel de guardar uma versão temporária dos dados) são geralmente classificados como o Model, na visão MVC do Struts*
- *View (V)*
  - *Geralmente uma página HTML ou JSP*
- *Controller (C) - Front Controller*
  - *org.apache.struts.action.ActionServlet ou subclasse*
- *Classes ajudantes (Model, View e Controller Helpers)*
  - *FormBeans: encapsula dados de forms HTML (M)*
  - *ActionErrors: encapsulam dados de erros (M)*
  - *Custom tags: encapsulam lógica para apresentação (V)*
  - *Actions: implementam lógica dos comandos (C)*
  - *ActionForward: encapsulam lógica de redirecionamento (C)*

Componentes da distribuição

- *Requisitos*
  - *J2SDK 1.4 ou J2SDK1.3 + JAXP*
  - *Servlet container, servlet.jar e Jakarta Commons (Struts 1.1)*
- *Distribuição binária (pré-compilada)*
  - *Abra o ZIP da distribuição. Conteúdo essencial:*



Componentes essenciais

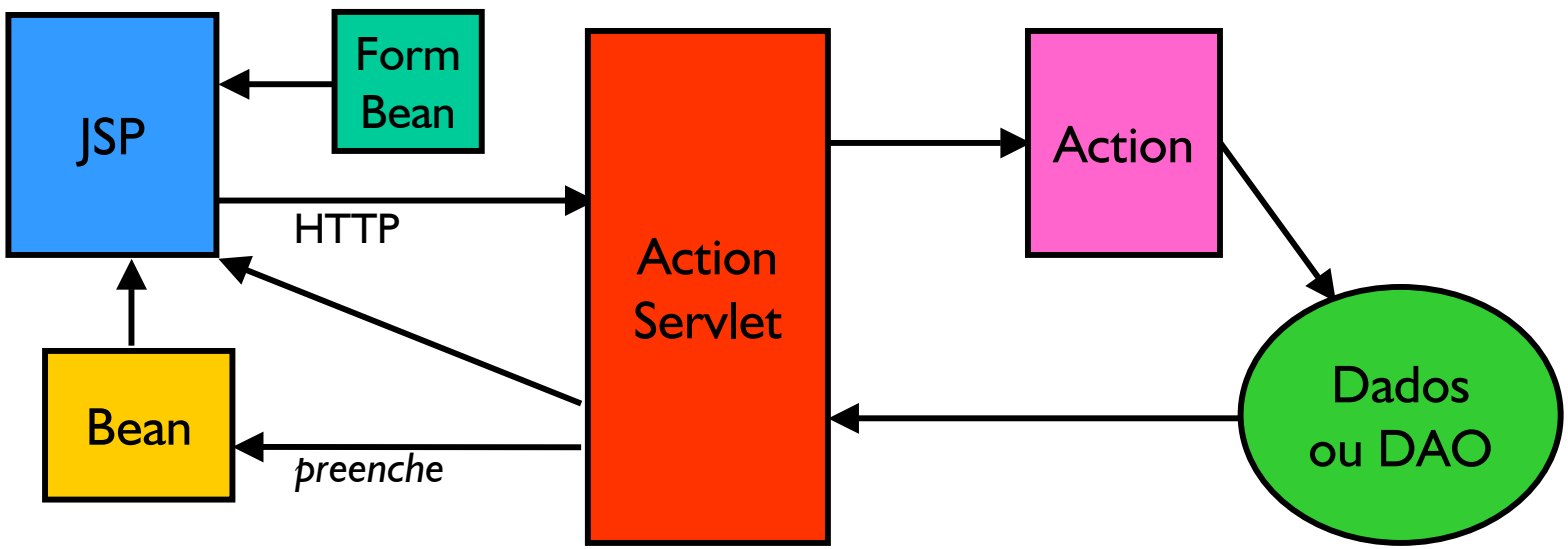
- *Framework (copie para WEB-INF/lib* ***de cada aplicação*** *Web)*
- *Descritores de taglib (copie para WEB-INF* ***de cada aplicação****)*

Aplicações Web (opcionais)

- *(jogue no webapps do Tomcat - instale pelo menos struts-documentation.war)*

Como funciona?

- *Principais componentes*
  - *ActionServlet: despachante de ações*
  - *Action: classe estendida por cada ação (comando) a ser implementada (usa Command design pattern)*
  - *struts-config.xml: arquivo onde se define mapeamentos entre ações, páginas, beans e dados*



Como instalar

- 1. Copiar os arquivos necessários para sua aplicação
  - Copie lib/struts.jar e lib/commons-*.jar para weu WEB-INF/lib (não coloque no common/lib do Tomcat ou no jre/lib/ext do JDK ou o struts não achará suas classes!)
  - Copie os TLDs das bibliotecas de tags que deseja utilizar para o WEB-INF de sua aplicação (copie todos)
- 2. Para usar o servlet controlador (MVC)
  - Defina-o como um <servlet> no seu web.xml
  - Crie um arquivo WEB-INF/struts.config.xml com mapeamentos de ações e outras as configurações
- 3. Para usar cada conjunto de taglibs
  - Defina, no seu web.xml, cada taglib a ser instalada
  - Carregue a taglib em cada página JSP que usá-la

Configuração do controlador no web.xml

- *Acrescente no seu web.xml*

```
<servlet>
   <servlet-name>action</servlet-name>
   <servlet-class>
      org.apache.struts.action.ActionServlet
   </servlet-class>
   <init-param>
      <param-name>config</param-name>
      <param-value>
         /WEB-INF/struts-config.xml
      </param-value>
   </init-param>
   ... outros init-param ...
</servlet>
```

- *Acrescente também os <servlet-mapping> necessários*
- *Crie e configure as opções de struts-config.xml*
- *Veja nos docs: /userGuide/building_controller.html*
  - *use os arquivos de struts-example.war para começar*

Configuração das Taglibs

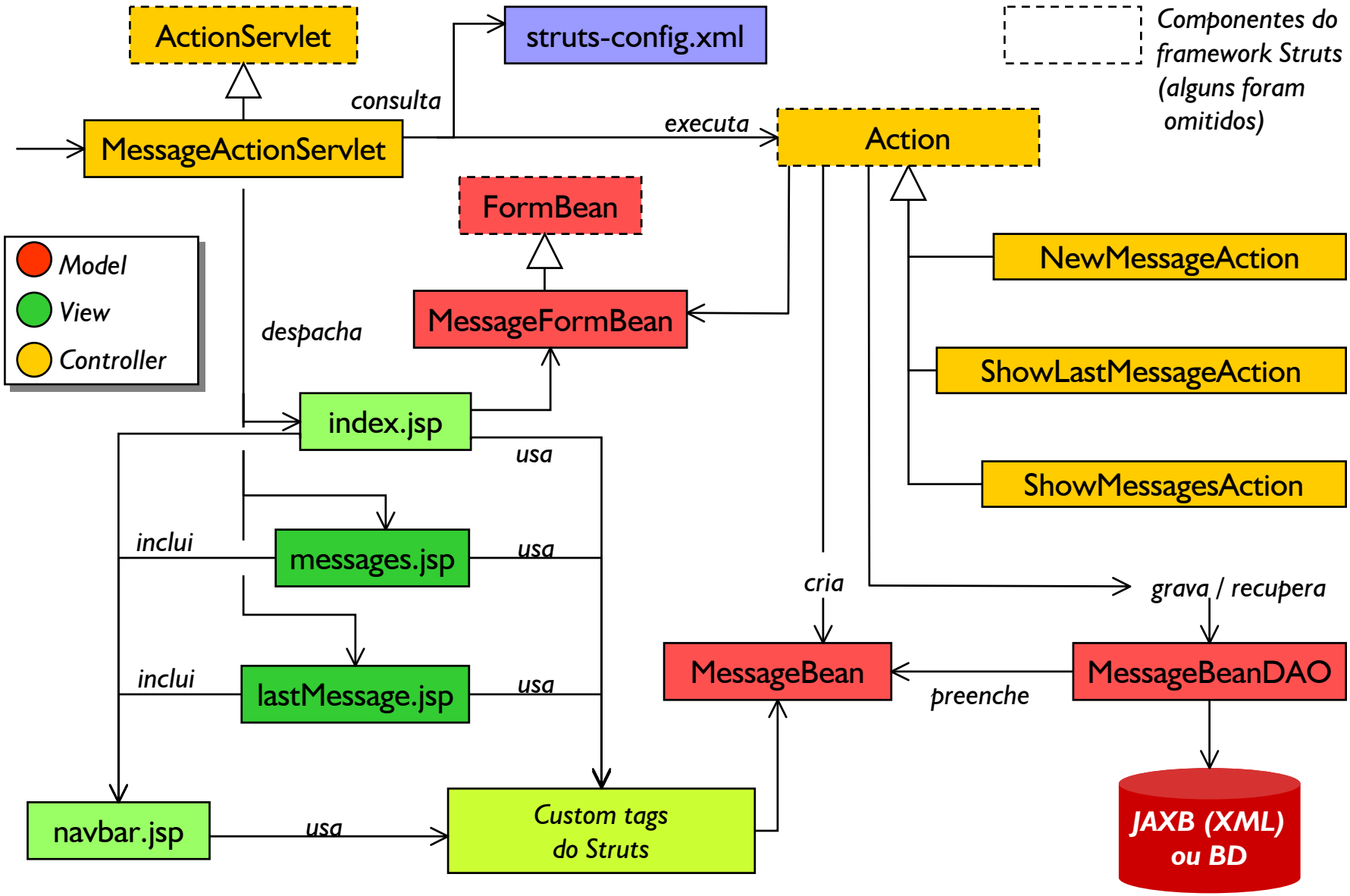
- *Acrescente em **web.xml***
- *Veja detalhes na aplicação struts-example.war ou nos docs:*
  /userGuide/building_controller.html#dd_config_taglib

```
<taglib>
   <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
   <taglib-location>/WEB-INF/struts-bean.tld
   </taglib-location>
</taglib>
<taglib>
   <taglib-uri>/WEB-INF/struts-form.tld</taglib-uri>
   <taglib-location>/WEB-INF/struts-form.tld
   </taglib-location>
... outros taglibs ...
</taglib>
```

- *Acrescente em cada página JSP*

```
<@ taglib uri="/WEB-INF/struts-bean.tld" prefix="bean" %>
<@ taglib uri="/WEB-INF/struts-logic.tld" prefix="logic" %>
   ...
```

Implementação de hellojsp com Struts



Mapeamentos (ActionMappings)

- *Veja webinf/struts-config.xml*

```
<struts-config>
  <form-beans>
    <form-bean name="newMessageForm" type="hello.jsp.NewMessageForm" />
  </form-beans>
  <global-forwards>
    <forward name="default" path="/index.jsp" />
  </global-forwards>

  <action-mappings>
    <action path="/newMessage" type="hello.jsp.NewMessageAction"
            validate="true"
            input="/index.jsp" name="newMessageForm" scope="request">
      <forward name="success" path="/showLastMessage.do" />
    </action>
    <action path="/showLastMessage"
            type="hello.jsp.ShowLastMessageAction" scope="request">
      <forward name="success" path="/lastMessage.jsp" />
    </action>
    <action path="/showAllMessages"
            type="hello.jsp.ShowMessagesAction" scope="request">
      <forward name="success" path="/messages.jsp" />
    </action>
  </action-mappings>

  <message-resources parameter="hello.jsp.ApplicationResources" />
</struts-config>
```

FormBeans

- *Form beans permitem simplificar a leitura e validação de dados de formulários*
  - *Devem ser usados em conjunto com custom tags da biblioteca <html:* />*

```
<html:form action="/newMessage" name="newMessageForm"
           type="hello.jsp.NewMessageForm">
    <p>Message: <html:text property="message" />
       <html:submit>Submit</html:submit>
    </p>
</html:form>
```

Configuração em struts-config.xml

```
public class NewMessageForm extends ActionForm {
    private String message = null;
    public String getMessage() { return message; }
    public void setMessage(String message) {
        this.message = message;
    }
    public void reset(...) {
        message = null;
    }
    public ActionErrors validate(...) {...}
}
```

ActionErrors

- *ActionErrors encapsulam erros de operação, validação, exceções, etc.*
  - *Facilitam a formatação e reuso de mensagens de erro.*
- *Exemplo: Método validate() do form bean:*

```
public ActionErrors validate(ActionMapping mapping,
                             HttpServletRequest request) {
  ActionErrors errors = new ActionErrors();
  if ( (message == null) || (message.trim().length() == 0) ) {
    errors.add("message",
               new ActionError("empty.message.error"));
  }
  return errors;
}
```

Nome de campo <input> ao qual o erro se aplica.

Este valor corresponde a uma chave no ResourceBundle

- *Como imprimir:*

```
<html:errors />
```

i18n (View Helper / Dispatcher View)

- *Informações localizadas podem ser facilmente extraídas de Resource Bundles através de*

```
<bean:message key="chave" />
```

  - *Locale default é usado automaticamente (pode ser reconfigurado)*
- *Exemplo de ResourceBundle*

```
empty.message.error=<tr><td>Mensagem não pode ser
vazia ou conter apenas espaços em
branco.</td></tr>
new.message.input.text=Digite a sua mensagem
message.submit.button=Enviar Mensagem
```

hello/jsp/ApplicationResources_pt.properties

- *Configuração em struts-config.xml*

```
<message-resources
      parameter="hello.jsp.ApplicationResources" />
```

- *Exemplo de uso:*

```
<p><bean:message key="new.message.input.text" />
```

Action (Controller / Service To Worker)

- *Controlador processa comandos chamando o método execute de um objeto Action*

```
public class ShowMessagesAction extends Action {

  private String successTarget = "success";
  private String failureTarget = "default";

  public ActionForward execute(ActionMapping mapping,
                               ActionForm form,
                               HttpServletRequest request,
                               HttpServletResponse response)
                            throws IOException, ServletException {
    try {
       MessageBeanDAO dao =
          (MessageBeanDAO)request.getAttribute("dao");
       MessageBean[] beanArray = dao.retrieveAll();
       request.setAttribute("messages", beanArray);
       return (mapping.findForward(successTarget));
     } catch (PersistenceException e) {
       throw new ServletException(e);
     }
  } ...
```

Como rodar o exemplo

- *1. Mude para exemplos/hellojsp_3*
- *2. Configure build.properties, depois rode*

```
> ant DEPLOY
```

- *3. Inicie o servidor (Tomcat ou JBoss)*
- *4. Rode os testes do Cactus*

```
> ant RUN-TESTS
```

- *5. Rode a aplicação, acessando a URI*

```
http://localhost:porta/hellojsp-struts/
```

- *6. Digite mensagens e veja resultados. Arquivos são gerados em /tmp/mensagens (ou c:\tmp\mensagens)*

Fontes

[Struts] Manual do Struts Copie o arquivo struts-documentation.war (se quiser, altere o nome do WAR antes) para o diretório webapps/ to Tomcat 4.0 ou deploy do JBoss.

[Core] Deepak Alur, John Crupi, Dan Malks. Core J2EE Patterns: Best Practices and Design Strategies. Prentice-Hall, 2001. http://java.sun.com/blueprints/corej2eepatterns/index.html.

[Goodwill] James Goodwill. Mastering Jakarta Struts. Wiley, 2002.

Curso J931: J2EE Design Patterns

Versão 1.0

www.argonavis.com.br

(helder@acm.org)