Apache Struts

Helder da Rocha (helder@acm.org)
www.argonavis.com.br

Sobre este módulo

- Este é um módulo opcional.
- Apresenta uma visão geral do framework Struts, usado para desenvolver aplicações Web com o padrão MVC
  - Não é exaustivo. Consulte a documentação suplementar no CD para mais informações
- A apresentação será feita através de um exemplo simples, mas que utiliza diversos recursos comuns em aplicações típicas
  - Utilize-o como base para aplicações maiores

Struts

O que é Struts?

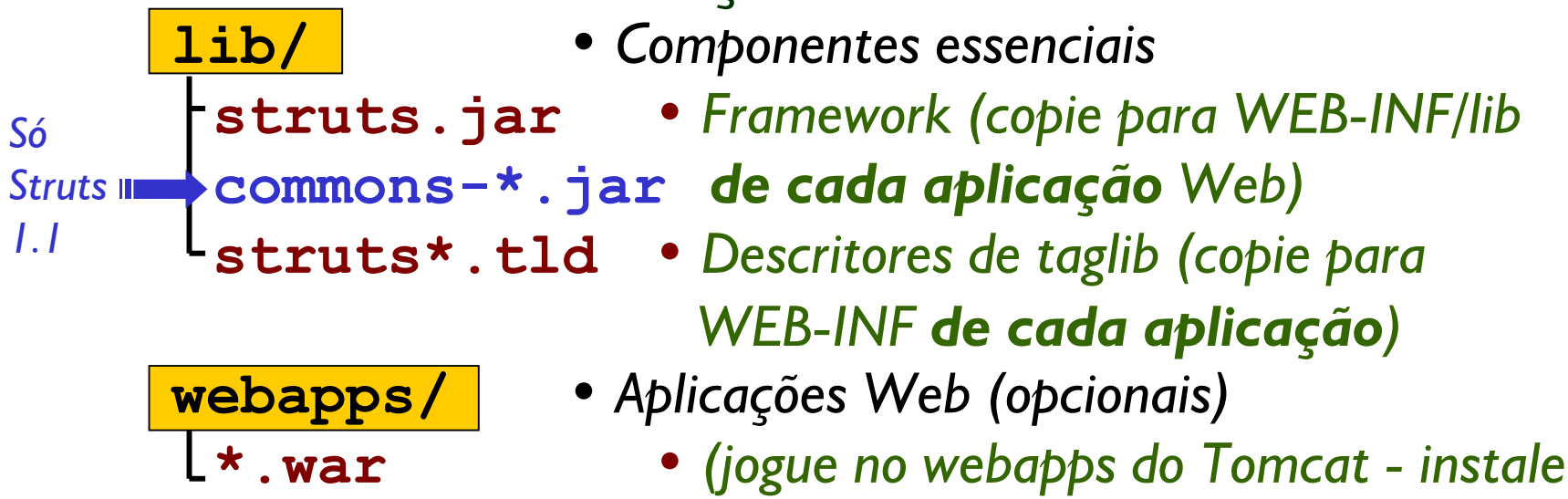
- *Framework para facilitar a implementação da arquitetura MVC em aplicações JSP*
- *Oferece*
  - *Um servlet controlador configurável (Front Controller) através de documentos XML externos, que despacham requisições a classes Action (Command) criadas pelo desenvolvedor*
  - *Uma vasta coleção de bibliotecas de tags JSP (taglibs)*
  - *Classes ajudantes 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*

Componentes MVC no Struts

- Model (M)
  - Geralmente um objeto Java (JavaBean)
- View (V)
  - Geralmente uma página HTML ou JSP
- Controller (C)
  - org.apache.struts.action.ActionServlet ou subclasse
- Classes ajudantes
  - 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)

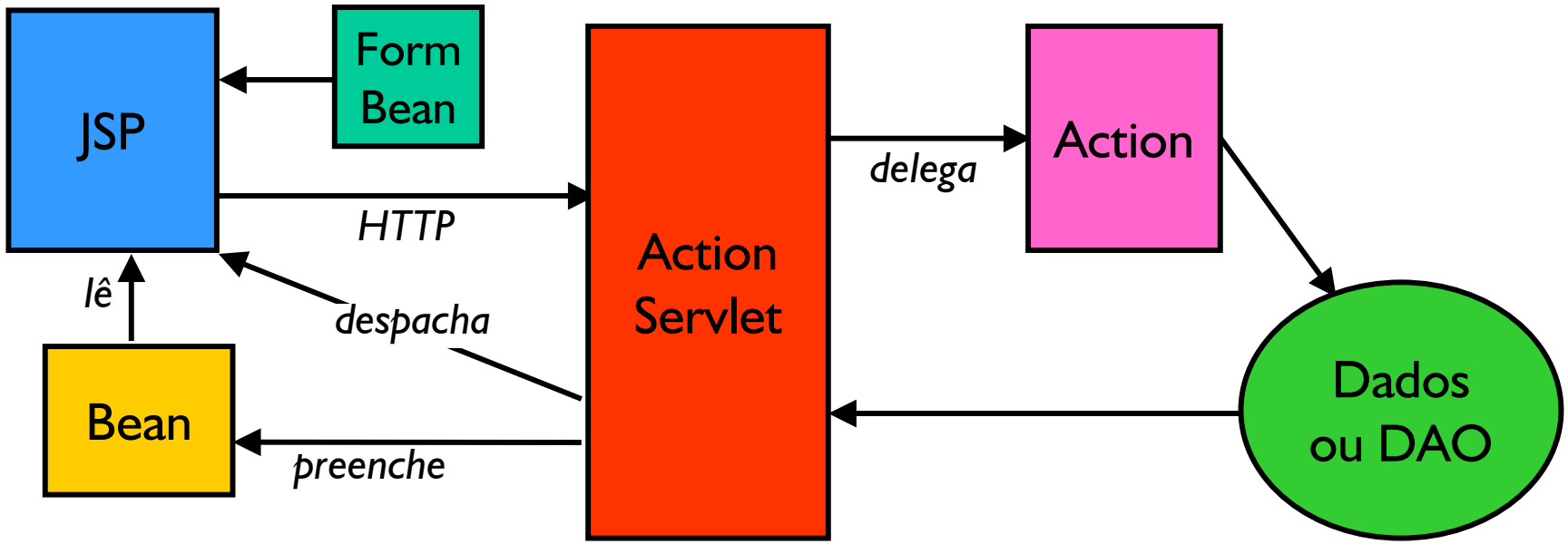
Componentes da distribuição

- 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:



Como funciona?

- Componentes-chave
  - 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 o <servlet-mapping> desejado
- 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>
```

URI é fácil de lembrar e tem o mesmo nome que a localização ideal da TLD

- *Acrescente em cada página JSP*

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

Como criar uma aplicação com o Struts?

Preparação

- 1. Defina seus comandos (Controller), escolha as URLs para chamá-los e nomes de classes Action que irão executá-los
- 2. Defina as páginas JSP que você irá precisar (Views), incluindo páginas de sucesso e de erro
- 3. Defina as páginas de entrada de dados (formulários)

Com os dados que você obteve na fase de preparação, crie

- 1. Um arquivo struts-config.xml. Use o DTD fornecido pelo Struts ou um esqueleto mínimo. Neste arquivo estão mapeadas todas as ações da sua aplicação: como serão chamadas (URLs de comando), classes Action que irão executá-las, formulários que serão usados e páginas que serão retornadas
- 2. Uma subclasse de Action para cada comando
- 3. Uma subclasse de ActionForm para cada formulário
- 4. Um resource bundle (arquivo .properties), possivelmente vazio, inicialmente, para guardar mensagens de erro.

Anatomia do struts-config.xml

- *Este é o arquivo mais importante da aplicação. Ele encapsula toda a lógica de processamento*

Bean que reflete os campos de seu formulário (criar)

```
<struts-config>
  <form-beans>
    <form-bean name="novaMsg" type="forum.EntradaDados" />
  </form-beans>
  <action-mappings>
    <action path="/nova"
            type="forum.NovaAction"
            validate="true"
            input="/index.jsp" name="novaMsg" scope="request">
      <forward name="sucesso" path="/todas.do" />
      <forward name="default" path="/index.jsp" />
    </action>
    <action path="/todas"
            type="forum.ListarAction" scope="request">
      <forward name="sucesso" path="/todas.jsp" />
      <forward name="erro" path="/index.jsp" />
    </action>
  </action-mappings>
  <message-resources parameter="forum.ApplicationResources" />
</struts-config>
```

Esta ação é iniciada por um formulário associado ao bean novaMsg implementado em index.jsp, e requer validação

Uma ação

Chama ação /todas

Resource-bundle (tem extensão .properties e está no CLASSPATH)

Ações

- *Não é preciso mexer no ActionServlet*
  - *Para grande parte das aplicações, basta escrever as classes Action e defini-las no struts-config.xml*
- *Para implementar um Action, crie uma nova classe e implemente seu método execute()*

Usado para redirecionar a uma View: "sucesso" ou "default" neste exemplo

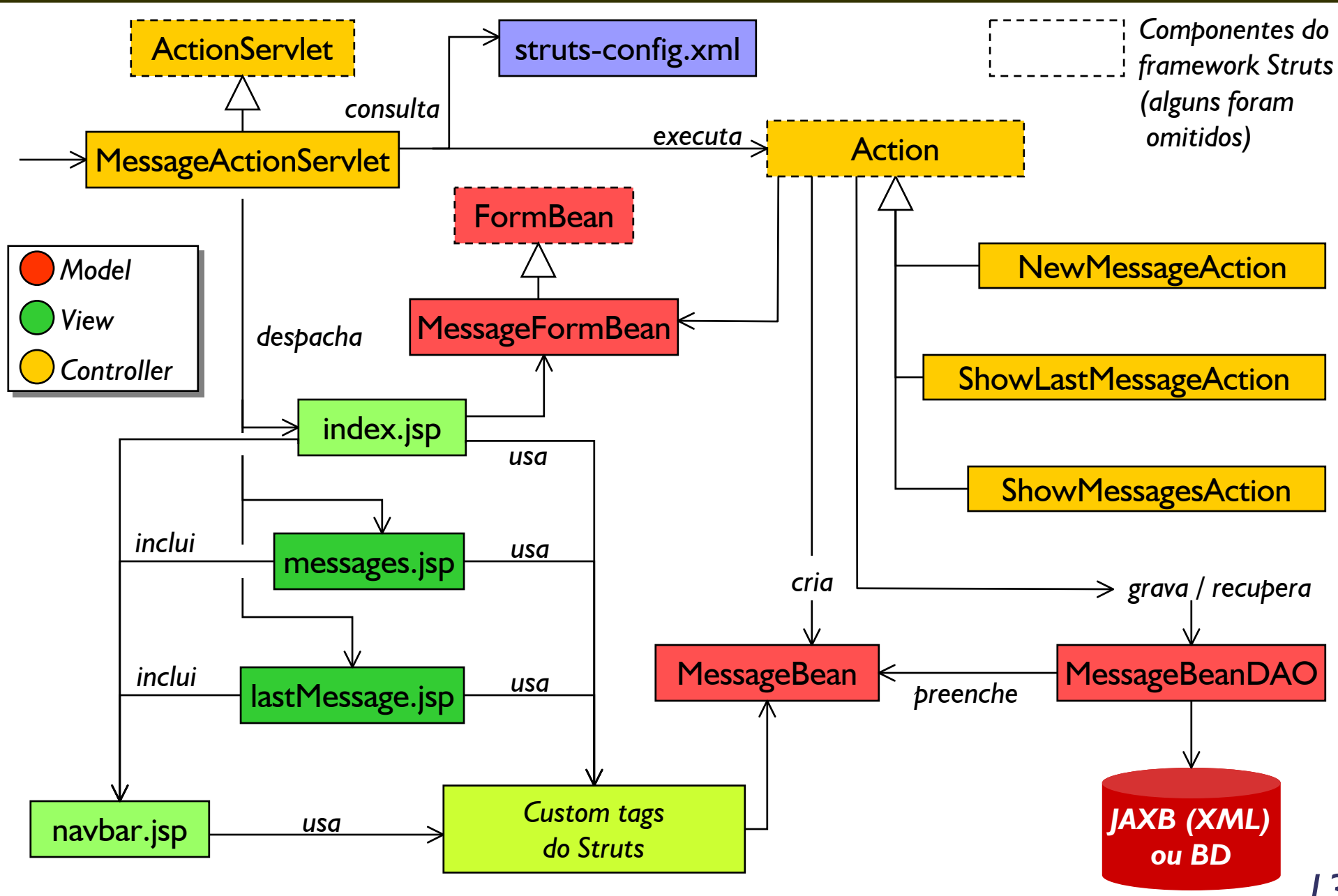
```
public class NovaAction extends Action {

   public ActionForward execute(ActionMapping mapping,
                                ActionForm form,
                                HttpServletRequest request,
                                HttpServletResponse response)
                        throws IOException, ServletException {
```

Se esta ação precisa de dados digitados no formulário, recupere-os através dos métodos get/set deste bean

```
      EntradaDados entrada = (EntradaDados)form;
      String mensagem = entrada.getMensagem();
        ...
      if (funcionou) return (mapping.findForward("sucesso"));
      else return (mapping.findForward("default"));
   }
```

Veja struts-config.xml

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

- *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 cap15/exemplos/hellojsp_3
- 2. Configure build.properties, depois rode

```
> ant DEPLOY
```

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

```
> 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)

Exercícios

- 1. Coloque o exemplo para funcionar
  - Analise o código das páginas JSP e classes Action
  - Veja o arquivo struts-config.xml
- 2. Adapte o segundo exercício do capítulo 2 para funcionar com o Struts

helder@acm.org

argonavis.com.br