
# PROJETO FRAMEWORK - CELEPAR

# PADRÕES PARA CAMADA DE APRESENTAÇÃO


**Março – 2006**

## Sumário de Informações do Documento

**Tipo do Documento:** Definição

**Título do Documento:** Padrões para camada de apresentação

**Estado do Documento:** EB (Elaboração)

**Responsáveis**: Cleverson Budel, Diego Pozzi, Fábio Sgoda, Filipe Lautert

**Palavras-Chaves:** jstl, jsp, java, projetos web, expression language, servlet

**Resumo:** Padronização par a desenvolvimento da camada de interface

**Número de páginas**: 8

**Software utilizados:**

| Versão | Data | Mudanças |
|---|---|---|
| 1.0 | | |
| 1.1 | 07/03/06 | Alteração no tópico 1.2.2, 1.3.1 e 1.4 – por: Natasha Fortes – revisado por Elisabeth Hoffmann |

# SUMÁRIO

# 1 PADRÕES PARA CAMADA DE APRESENTAÇÃO

## 1.1 Introdução

Este documento visa indicar uma forma padrão para o desenvolvimento da camada de apresentação, a view do modelo MVC (Model View Controller).

## 1.2 Padrões

Serão utilizadas novas versões/especificações de Servlets, JSP e JSTL para compor a camada de apresentação das aplicações web da CELEPAR.

### 1.2.1 JSTL (JSP Standard Tag Library)

São fornecidas cinco bibliotecas de taglibs com o Struts: **HTML**, com tags para criação de páginas dinâmicas e formulários; **Beans** para acesso a JavaBeans e suporte a internacionalização; **Logic**, com tags para loops e execução condicional; **Tiles** para a utilização do mecanismo de templates embutido; e **Nested**, para a manipulação de estruturas hierarquizadas de beans.

As bibliotecas do Struts surgiram antes da padronização da JSTL (*JSP Standard Tag Library*). Várias tem praticamente as mesmas funcionalidades, e devem ser utilizadas as tags equivalentes da JSTL 1.1 no lugar das taglibs **Logic** e **Bean** do Struts.

### 1.2.2 Servlet

Deve-se utilizar da especificação 2.4 de servlets. Essa especificação acrescenta uma série de melhorias e novas funcionalidades.

Algumas mudanças são necessárias no arquivo *web.xml* das aplicações para utilização desta especificação. O arquivo *web.xml* serve como um descritor das funcionalidades e características da aplicação. Ele é um arquivo XML descrevendo os Servlets, JSPs , EJBs e outros componentes que fazem parte da sua aplicação

### 1.2.3 JSP

Deve-se utilizar da especificação 2.0 de JSP. Essa especificação permite a utilização de EL

(Expression Language) que facilita o desenvolvimento e se integra com as JSTLs.

## 1.3 Exemplos

Abaixo tem-se alguns exemplos da utilização de algumas dessas novas tecnologias:

### 1.3.1 Exemplo 1 (JSTL 1.1 e JSP 2.0)

Neste exemplo temos uma página JSP que utiliza a biblioteca “fmt” da implementação do JSTL 1.1. A página mostra como realizar formatação de datas e como utilizar o *resource bundle* para internacionalização ou apenas para leitura de dados de arquivo de propriedades. Caso o sistema necessite internacionalização, ele deverá ter todos os textos cadastrados no Application Resources. Caso contrário, só as mensagens precisam estar nesse arquivo.

Podemos observar nesses exemplos a utilização de EL (Expression Language) definida na especificação 2.0 de JSP.

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<fmt:setBundle
basename="gov.pr.celepar.util.properties.ApplicationResources"
var="bundle"/>
<jsp:useBean id="agora" class="java.util.Date" />

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title><fmt:message key="mensagem.app.titulo"
bundle="${bundle}"/></title>
    <meta http-equiv="pragma" content="no-cache" />
    <meta http-equiv="cache-control" content="no-cache" />
    <meta http-equiv="description" content="Exemplo de formataÃ§Ã£o de
data e leitura de arquivo de properties usando JSTL e JSP 2.0">
  </head>

  <body>
    <h1><fmt:message key="mensagem.app.today" bundle="${bundle}"/></h1>
    <p>
    Versão curta:
    <fmt:formatDate value="${agora}" />
    <p>
    Versão longa:
    <fmt:formatDate value="${agora}" dateStyle="full" />
  </body>
</html>
```

### 1.3.2 Exemplo 2 (JSTL 1.1 e JSP 2.0)

O segundo exemplo mostra como realizar instruções condicionais e declaração de variáveis. A página pede que seja informado um nome com três palavras e formata este nome conforme o padrão de alguns países.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>

<jsp:useBean id="agora" class="java.util.Date" />

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>JSTL Exemplos - Grupo Framework - Celepar</title>
    <meta http-equiv="pragma" content="no-cache" />
    <meta http-equiv="cache-control" content="no-cache" />
    <meta http-equiv="description" content="Exemplo de declaraÃ§Ã£o de
variÃ¡veis a partir funÃ§Ãµes e controle de fluxo">
  </head>

  <body>
    <h1>Escrevendo Nomes</h1>
    <p>
    Escreva um nome utilizando três palavras (ex: Carlos Silva Junior)
    <form>
      <input name="nome" value="${param.nome}" />
      <input type="submit" value="OK" />
    </form>
    <hr>

    <c:if test="${!empty param.nome}">
      <c:set var="nomes" value="${fn:split(param.nome, ' ')}" />
      <c:if test="${fn:length(nomes) != 3}">
        <i>Deve ser fornecido um nome formado por exatamente três
palavras.</i>
      </c:if>

      <c:if test="${fn:length(nomes) == 3}">
        <c:set var="primNome" value="${nomes[0]}" />
        <c:set var="nomeDoMeio" value="${nomes[1]}" />
        <c:set var="sobreNome" value="${nomes[2]}" />
        <p>
        Nos EUA, o nome seria escrito como:<br>
        ${sobreNome}, ${primNome} ${nomeDoMeio}
        <p>
        Na Espanha, o nome seria escrito como:<br>
        ${primNome} ${sobreNome} ${nomeDoMeio}
      </c:if>
    </c:if>

  </body>
</html>
```

### 1.3.3 Exemplo 3 (JSTL 1.1 x Struts Taglib)

Uma comparação entre a utilização das tags do struts e da jstl. Neste exemplo temos duas páginas que apresentam o mesmo resultado mas a primeira utiliza as tags do Struts e a segunda utiliza as tags da JSTL.

O resultado obtido executando as páginas abaixo será uma tela confirmação de cadastro.

```
<%@ taglib prefix="bean" uri="http://struts.apache.org/tags-bean" %>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>
<%@ taglib prefix="logic" uri="http://struts.apache.org/tags-logic" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html>
  <head>
    <title>Cliente Cadastrado</title>
    <meta http-equiv="pragma" content="no-cache" />
    <meta http-equiv="cache-control" content="no-cache" />
    <meta http-equiv="description" content="Exemplo de utilização das tags
do struts">
  </head>

  <body>
    Cadastro

    <logic:present name="nome" scope="session">
      <logic:equal name="cadastroForm" property="sexo" value="F"> da Sra.
</logic:equal>
      <logic:equal name="cadastroForm" property="sexo" value="M"> do Sr.
</logic:equal>
      <bean:write name="nome" scope="session"/>
    </logic:present>

    efetuado com sucesso!
  </body>
</html:html>
```

```
<%@ taglib prefix="c" uri="http://java.sun.com/jstl/core" %>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html:html>
  <head>
    <title>Cliente Cadastrado</title>
    <meta http-equiv="pragma" content="no-cache" />
    <meta http-equiv="cache-control" content="no-cache" />
    <meta http-equiv="description" content="Exemplo de utilização das tags
JSTL">
  </head>

  <body>
    Cadastro

    <c:set var="saudacao">
      <c:choose>
        <c:when test="${sessionScope.sexo == 'F'}"> da Sra. </c:when>
        <c:when test="${sessionScope.sexo == 'M'}"> do Sr. </c:when>
      </c:choose>
    </c:set>

    <c:out value="${saudacao} ${nome}" /> efetuado com sucesso!
  </body>
</html:html>
```

### 1.3.4 Outros Exemplos

Junto com este documento segue uma aplicação com exemplos desenvolvida pela Apache assim como a documentação oficial da JSTL.

Para rodar a aplicação basta colocar os arquivos *standard-doc.war* e *standard-examples.war* no seu servidor web e acessar seu contexto.

## 1.4 Conclusão

A forma padronizada de criar interfaces facilita de forma significativa o desenvolvimento e a manutenção de aplicações. Por isso é necessário que se utilize alguns padrões e que se tenha bom senso durante a construção e elaboração de interfaces, evitando criar interfaces complexas, pesadas e páginas com código java em seu corpo.

O Struts fornece bibliotecas de tags e um mecanismo de templates e pode trabalhar com JSTL e outras tecnologias de apresentação. É uma solução bastante completa e madura para criação da parte mais interativa das suas aplicações web. Uma outra opção que pode ser utilizada é a criação de tag próprias, criadas por projeto.