Como criar Custom Tags

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

Objetivos

- Este é um módulo opcional. Explora os fundamentos para criação de custom tags
  - Funcionamento: como fazer um custom tag
  - Exemplos de tags simples
- Não exploramos aspectos mais avançados
  - A criação de uma biblioteca de custom tags complexa não é uma tarefa comum para a maior parte dos desenvolvedores. É mais comum reutilizar tags existentes e criar alguns mais simples
  - Veja exemplos de tags mais elaborados no diretório do CD correspondente a este capítulo

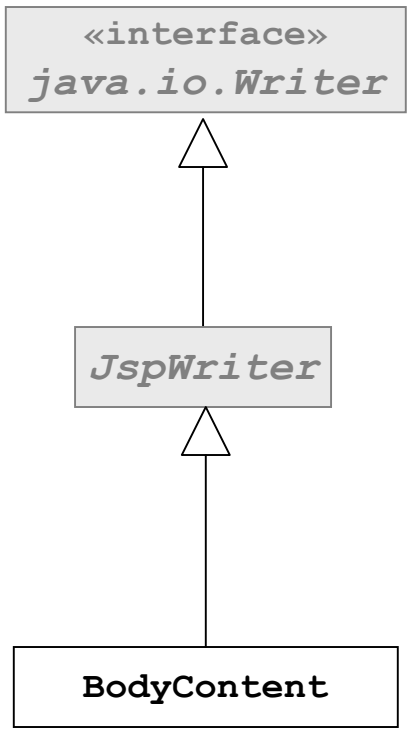
Como criar um custom Tag

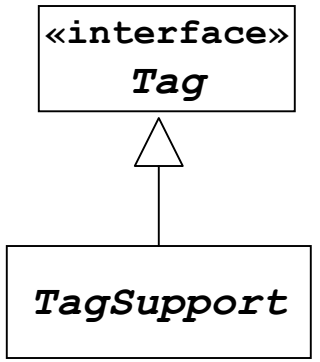
- Para criar um custom tag é preciso programar usando as APIs
  - javax.servlet.jsp e
  - javax.servlet.jsp.tagext
- Resumo dos passos típicos
  - Escreva uma classe que implemente a interface Tag ou BodyTag (ou as adaptadores TagSupport e BodyTagSupport)
  - Implemente os métodos do ciclo de vida desejados
  - Escreva um descritor TLD ou acrescente os dados de seu tag a um TLD existente
  - Empacote tudo em um JAR
  - Inclua o JAR em um WAR e use os tags em suas páginas

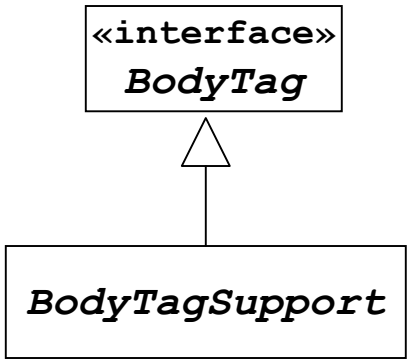
API

Pacote *javax.servlet.jsp.tagext*: principais componentes

- Interfaces
  - *Tag*
  - *BodyTag*
  - *IterationTag*
- Classes
  - *TagSupport*
  - *BodyTagSupport*
  - *BodyContent*



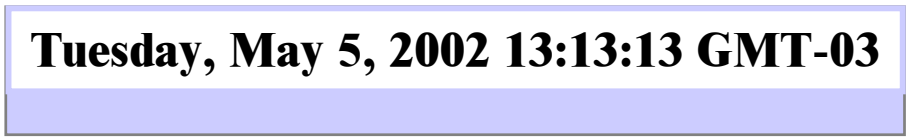




Exemplo (tag usado em capítulo anterior)

- No capítulo 11 mostramos como usar um custom tag muito simples
- `<exemplo:dataHoje />`

  que tinha como resultado



- Para construir este tag é preciso
  - 1. Escrever a classe que o implementa
  - 2. Declará-lo em seu arquivo TLD
- Para distribuí-lo, encapsule tudo em um JAR que o usuário do tag possa colocar em seu WEB-INF/lib

Configuração do Tag

```
<?xml version="1.0" ?>
<!DOCTYPE taglib PUBLIC
"-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.2//EN"
"http://java.sun.com/dtd/web-jsptaglibrary_1_2.dtd">

<taglib>
    <tlib-version>1.0</tlib-version>
    <jsp-version>1.2</jsp-version>
    <short-name>exemplo</short-name>
    <uri>http://abc.com/ex</uri>

    <tag>
        <name>dataHoje</name>
        <tag-class>exemplos.DateTag</tag-class>
        <description>Data de hoje</description>
    </tag>

</taglib>
```

Sugestão de prefixo (autor de página pode escolher outro na hora)

URI identifica o prefixo. (autor de página tem que usar exatamente esta URI)

Implementação

```
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class DateTag extends TagSupport {
    /**
     * Chamado quando o tag terminar.
     */
    public int doEndTag() throws JspException {
        try {
            Writer out = pageContext.getOut();
            java.util.Date = new java.util.Date();
            out.println(hoje.toString());
        } catch (java.io.IOException e) {
            throw new JspException (e);
        }
        return Tag.EVAL_PAGE;
    }
}
```

Para tags que não precisam processar o corpo use a interface Tag ou sua implementação TagSupport

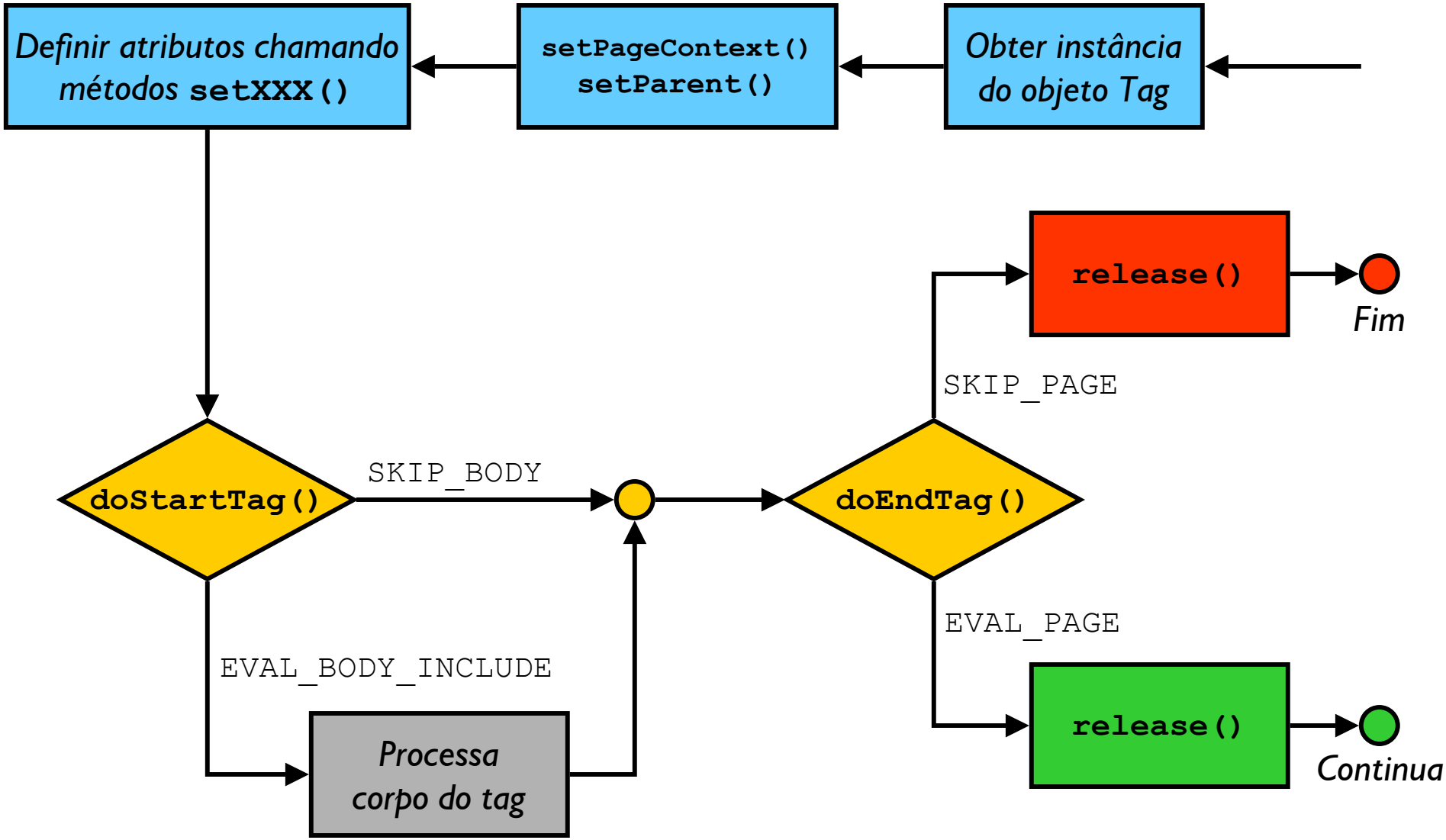
Use doStartTag() para processamento antes do tag, se necessário

Para este método, pode ser EVAL_PAGE ou SKIP_PAGE

Tipos de tags

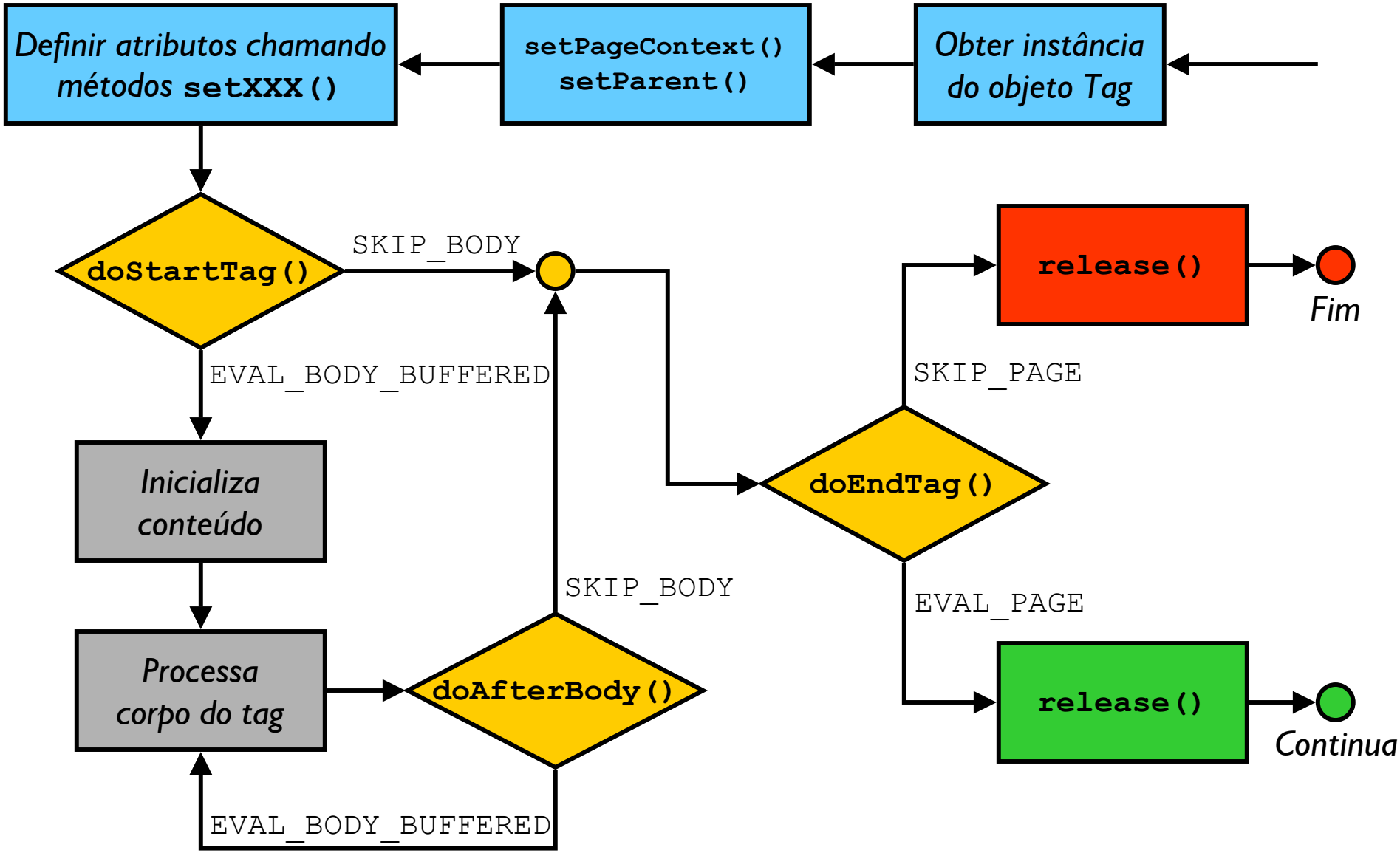
- *Há vários tipos de custom tags. Cada estratégia utiliza diferentes classes base e métodos*
- *1. Diferenciados por herança:*
  - *Tags simples: implementam a interface Tag (TagSupport é uma implementação neutra).*
  - *Tags com corpo que requer processamento: implementam BodyTag (BodyTagSupport é implementação neutra)*
- *Diferenciados por outras características*
  - *Tags que possuem atributos*
  - *Tags que definem variáveis de scripting fora do seu escopo (requerem classe extra com "Tag Extra Info")*
  - *Tags que interagem com outros tags*

Ciclo de vida de objetos Tag



Fonte: [6]

Ciclo de vida de objetos BodyTag



Fonte: [6]

Atributos

- *Para definir atributos em um tag é preciso*
  1. *Definir método setXXX() com o nome do atributo*
  2. *Declarar atributo no descritor (TLD)*

```
<xyz:upperCase text="abcd" />
```

```
public class UpperCaseTag {
    public String text;
    public void setText(String text) {
        this.text = text;
    } (...)
```

```
<tag> <name>upperCase</name> (...)
    <attribute>
        <name>text</name>
        <required>true</required>
        <rtexprvalue>false</rtexprvalue>
    </attribute> (...) </tag>
```

- *Os atributos devem setar campos de dados no tag*
  - *Valores são manipulados dentro dos métodos doXXX():*

```
public int doStartTag() throws JspException {
    Writer out = pageContext.getOut();
    out.println(text.toUpperCase());   (...)
```

Obtenção do conteúdo do Body

- *O objeto out, do JSP, referencia a instância BodyContent de um tag enquanto processa o corpo*
  - *BodyContent é subclasse de JspWriter*
  - *Tag decide se objeto BodyContent deve ser jogado fora ou impresso (na forma atual ou modificada)*
- *Exemplo*

```
public int doAfterBody() throws JspException {
    BodyContent body = getBodyContent();
    String conteudo = body.getString();
    body.clearBody();
    (...)
    getPreviousOut().print(novoTexto);
}
```

Guarda conteúdo do tag

Apaga conteúdo

Imprime texto na página (e não no body do Tag)

Exemplos de Custom Tags

- *Veja* **`cap16/exemplos/taglibs/`**
  - *Vários diferentes exemplos de custom tags (do livro [6])*
  - *Código fonte em taglib/src/taglibdemo/*.java*
  - *Páginas exemplo em src/*Test.jsp (6 exemplos)*

1. Configure build.properties, depois, monte o WAR com:

```
> ant build
```

2. Copie o WAR para o diretório webapps do Tomcat

```
> ant deploy
```

3. Execute os tags, acessando as páginas via browser:

```
http://localhost:porta/mut/
```

- *Veja também* **`cap12/exemplos/hellojsp_2/`**
  - *Aplicação MVC que usa custom tags*

Exercício

- *1. Escreva um custom tag simples <j550:tabela> que receba um String como parâmetro (texto) e imprima o String dentro de uma tabela HTML:*
  - *O tag*

```
<j550:tabela texto="Texto Recebido"/>
```

deve produzir

```
<table border="1"><tr><td>
Texto Recebido</td></tr></table>
```

- *2. Crie mais dois atributos que definam cor-de-fundo e cor-do-texto para a tabela e o texto.*
- *3. Escreva uma segunda versão do tag acima que aceite o texto entre os tags*

```
<j550:tabela>Texto Recebido</j550:tabela>
```

helder@acm.org

argonavis.com.br