HowTo create JSP taglibs and their usage in ATG Dynamo

20.07.2001, mm

Autor	Michael Maretzke
eMail	michael@maretzke.com

Stand	24.07.2001
Status	preliminary finished released
	x intern x extern
Version	1.0
Druckdatum	17.11.03 22:27
Verwendungszweck	Knowledge Management

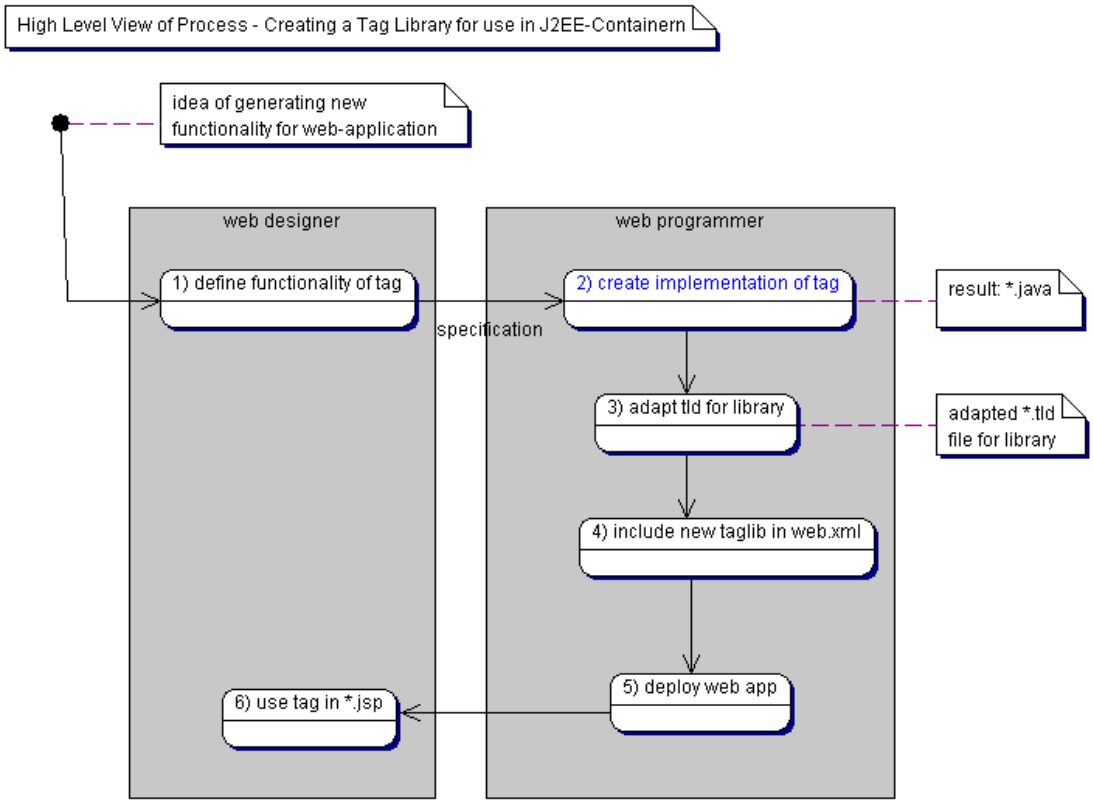
1 Overview

There are some steps to follow to create a J2EE-compliant tag library and deploy it into a running ATG Dynamo installation. These steps will be described in the following.

But first a word on why creating a tag library. A tag library simply follows the paradigma defined in the J2EE specification especially in the JSP specification (which is one of many specifications mentioned in the J2EE one). The usage of these libraries may help to separate the web design from the logic layer behind.

One way to generate JSP is to mix HTML code (design) with Java code (logic). The result is a unmaintainable mix of functionality and design. For maintainance of such a JSP the knowledge of HTML and Java is needed. If there's a way to separate these two tasks the result is a clean JSP which contains only tags. HTML tags and the custom JSP tags. Having a good documentation of these customized ones a web designer is capable of building JSPs. Only the knowledge of tag combination is needed. The result is a clear separation of design and functionality.

To get such a library there are some steps to follow.



Grafic 1 Overview of creating a tag library

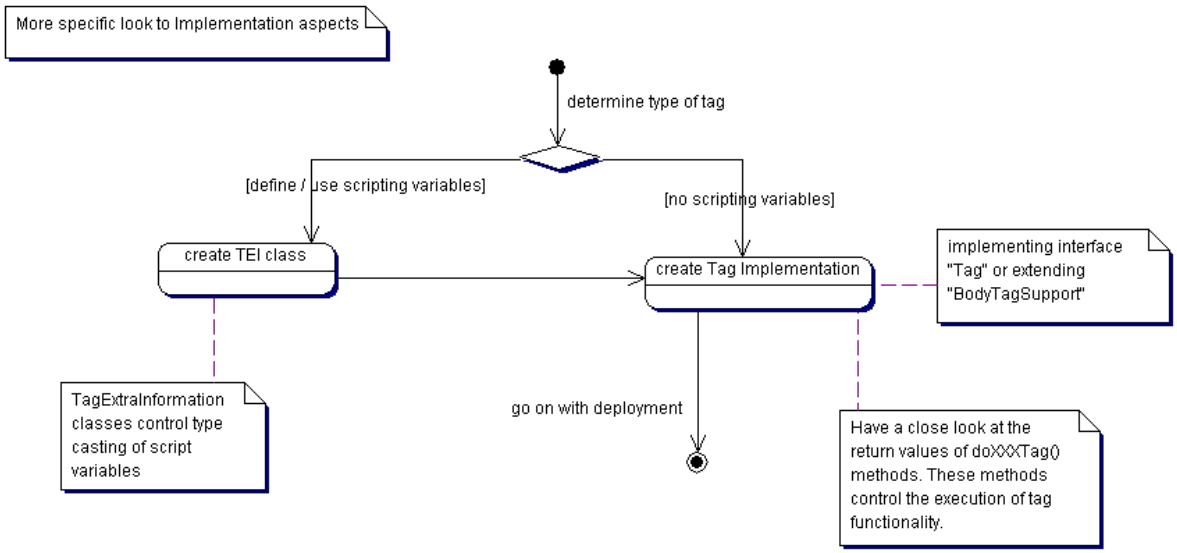
2 Single Steps in detail

2.1 Define functionality of tag

The first step is the need for new functionality which can't be produced combining existing general purpose tags. The web designer specifies his needs and gives this specification towards the web programmer.

2.2 Create implementation of tag

First decision for the programmer will be whether the wanted tag will set scripting variables or not. If so a TagExtraInformation class have to be generated. If not the programmer may start creating the implementation of tag functionality. For example code see 3. Examples.



Grafic 2 Implementation aspects

2.3 Adapt tld for library

The Tag Library Definition (tld) file can be found in „`/TagExample/j2ee-apps/web-app/Web-inf/taglibs`" directory. See a sample tag below for an example. For more information have a look at [SUN01], [SUN02], [SUN03] and [SUN04].

```
<tag>
    <name>setvar</name>
    <tagclass>de.emprise.taglib.SetScriptVarTag</tagclass>
    <teiclass>de.emprise.taglib.SetScriptVarTEI</teiclass>
    <bodycontent>empty</bodycontent>
    <attribute>
      <name>name</name>
      <required>yes</required>
      <rtexprvalue>yes</rtexprvalue>
    </attribute>
    <attribute>
      <name>value</name>
      <required>yes</required>
      <rtexprvalue>yes</rtexprvalue>
```

```
    </attribute>
    <attribute>
      <name>type</name>
      <required>no</required>
      <rtexprvalue>yes</rtexprvalue>
    </attribute>
  </tag>
```

2.4 Include new taglib in web.xml

To make the library visible to the J2EE-container registration is needed. This is made with an entry in web.xml file. See a sample entry below. Further information in [SUN01], [SUN02], [SUN03] and [SUN04].

```
<taglib>
  <taglib-uri>/empTaglib</taglib-uri>
  <taglib-location>/WEB-INF/taglibs/empTaglib1_0.tld</taglib-location>
</taglib>
```

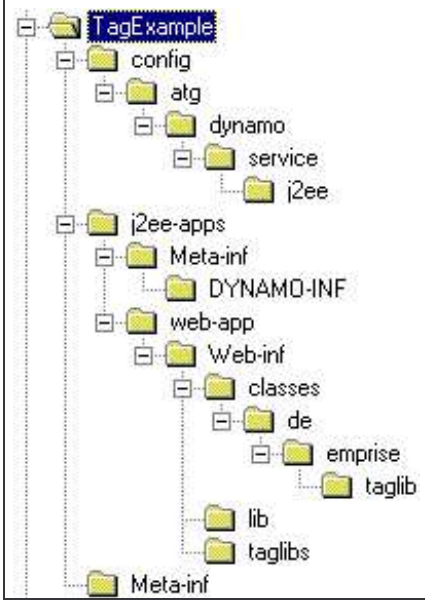
2.5 Deploy web app

The deployment depends on which J2EE container is used. In this case the ATG Dynamo application server will be used. Given the directory structure below the deployment, building, stub- & skeleton creation process may be started with the following command:

```
/opt/Dynamo5.1/TagExample# ../home/bin/runDarina ./j2ee-apps/ -o
TagExample.dar -build -overwrite-dar
```

After deploying the J2EE application restart the ATG Dynamo server and access the application via browser:

```
http://host:8840/tagexample
```



TagExample/:

TagExample.dar

TagExample/config/atg/dynamo/service/j2ee:

J2EEContainer.properties

TagExample/j2ee-apps/META-INF:

application.xml

TagExample/j2ee-apps/META-INF/DYNAMO-INF:

dynamoJ2EESpecifier.xml

TagExample/j2ee-apps/web-app/WEB-INF:

web.xml

TagExample/j2ee-apps/web-app/WEB-INF/taglibs:

empTaglib1_0.tld

Grafic 3 Directory structure and configuration files

2.6 Use tag in *.jsp

After creating and deploying the tag library the web designer wants to use the library of course. Therefor the designer should be able to include the library into a JSP. The following statement may be used to include a taglib „`empTaglib`" into a JSP.

```
<%@ taglib uri="/empTaglib" prefix="emp" %>
```

After including the tag library the web designer may use a tag like this:

```
<emp:if condition="<% test %> == 123">
  <emp:then>...</emp:then>
  <emp:else>...</emp:else>
</emp:if>
```

3 Examples

1.1. Example without any scripting variables

This simple example tag will be replaced by an string expression when beeing processed. It's a good start to get some practice with the deployment process.

`SimpleTag.java:`

```
// sample usage of simple tag:
// <%@ taglib uri="/empTaglib" prefix="emp" %>
// ...
```

```
// <emp:simple />

package de.emprise.taglib;

// import needed for interface "Tag"
import javax.servlet.jsp.tagext.*;
// import needed for class PageContext, JspException, ...
import javax.servlet.jsp.*;

public class SimpleTag implements Tag {

  // Class version string
  public static final String CLASS_VERSION = "SimpleTag.java,v 1.0, 2001-07-19, Michael Maretzke";

  private PageContext _pagecontext;
  private Tag _parent;

  //
  // Tag interface methods
  //

  // Set the current page context. Called by the page implementation prior to doStartTag().
  // This value is *not* reset by doEndTag() and must be explicitly reset by a page implementation
  public void setPageContext (PageContext pc) {
    _pagecontext = pc;
  }

  // Set the current nesting Tag of this Tag. Called by the page implementation prior to doStartTag().
  // This value is *not* reset by doEndTag() and must be explicitly reset by a page implementation.
  // Code can assume that setPageContext has been called with the proper values before this point.
  public void setParent (Tag parent) {
    _parent = parent;
  }

  // Returns the parent tag of this instance.
  public Tag getParent () {
    return _parent;
  }

  // Called on a Tag handler to release state. The page compiler guarantees this method will be
  // called on all tag handlers, but there may be multiple invocations on doStartTag and doEndTag
  // in between.
  public void release () {
    _pagecontext = null;
    _parent = null;
  }

  // Process the start tag for this instance.
```

```
  public int doStartTag () throws JspException {
    try {
      _pagecontext.getOut().print("This was Michael's \"Simple\" Tag.");
      // don't evaluate the body of the tag
      return SKIP_BODY;
    }
    catch (Exception e) {
      throw new JspException ("Simple Tag: " + e.getMessage());
    }
  }

  // Process the end tag. This method will be called on all Tag objects.
  public int doEndTag () throws JspException {
    return EVAL_PAGE;
  }
}
```

1.2. Example setting a scripting variable

This tag sets a scriptvariable. The name of this variable will be specified by the „`name`" parameter of the tag. The variable is accessible using the expression „`<%= name%>`".

This tag have to implement another class called „SetScriptVarTEI". This is a class providing further information depending on how to set the scripting variable.

`SetScriptVarTag.java:`

```
// sample usage of if tag:
// <%@ taglib uri="/empTaglib" prefix="emp" %>
// ...
// <emp:setvar name="variable" value="123" />

package de.emprise.taglib;

// import needed for interface "Tag"
import javax.servlet.jsp.tagext.*;
// import needed for class PageContext, JspException, ...
import javax.servlet.jsp.*;

public class SetScriptVarTag implements Tag {

  // Class version string
  public static final String CLASS_VERSION = "SetScriptVarTag.java,v 1.0, 2001-07-19, Michael Maretzke";

  private PageContext _pagecontext;
  private Tag _parent;

  // Members for attributes connected to tag
  private String _name;
  private String _value;
```

```
private String _type;

//
// Attribute getter / setter
//

// Attribute getter / setter for "name"
public void setName(String name) {
  _name = name;
}
public String getName() {
  return _name;
}

// Attribute getter / setter for "value"
public void setValue(String value) {
  _value = value;
}
public String getValue() {
  return _value;
}

// Attribute getter / setter for "type"
public void setType(String type) {
  _type = type;
}
public String getType() {
  return _type;
}

//
// Tag interface methods
//

// Set the current page context. Called by the page implementation prior to doStartTag().
// This value is *not* reset by doEndTag() and must be explicitly reset by a page implementation
public void setPageContext (PageContext pc) {
  _pagecontext = pc;
}

// Set the current nesting Tag of this Tag. Called by the page implementation prior to doStartTag().
// This value is *not* reset by doEndTag() and must be explicitly reset by a page implementation.
// Code can assume that setPageContext has been called with the proper values before this point.
public void setParent (Tag parent) {
  _parent = parent;
}

// Returns the parent tag of this instance.
```

```
  public Tag getParent () {
    return _parent;
  }

  // Called on a Tag handler to release state. The page compiler guarantees this method will be
  // called on all tag handlers, but there may be multiple invocations on doStartTag and doEndTag
  // in between.
  public void release () {
    _name = null;
    _value = null;
    _type = null;

    _pagecontext = null;
    _parent = null;
  }

  // Process the start tag for this instance.
  public int doStartTag () throws JspException {
    return SKIP_BODY;
  }

  // Process the end tag. This method will be called on all Tag objects.
  public int doEndTag () throws JspException {
    try {
      _pagecontext.setAttribute(_name, _value);
      // don't evaluate the body of the tag
      return EVAL_PAGE;
    }
    catch (Exception e) {
      throw new JspException ("SetScriptVar Tag: " + e.getMessage());
    }
  }
}
```

SetScriptVarTEI.java:

```
package de.emprise.taglib;

// import needed for class "TagExtraInfo"
import javax.servlet.jsp.tagext.*;

public class SetScriptVarTEI extends TagExtraInfo {

  // Class version string
  public static final String CLASS_VERSION = "SetScriptVarTEI.java,v 1.0, 2001-07-19, Michael Maretzke";

  public VariableInfo [] getVariableInfo (TagData data) {
    String type = (String) data.getAttribute("type");
```

```
    if (type == null) type = "java.lang.String";

    return new VariableInfo [] {
            new VariableInfo(data.getAttributeString("name"), type, true, VariableInfo.AT_END)
    };
  }

}
```

1.3. Example using some tags implementing if-functionality

After these simple examples let's look to a more sophisticated functionality. The basics of this example can be found in [SUN03].

sample usage of if tag:

```
<emp:if condition="<% test %> == 123">
  <emp:then>...</emp:then>
  <emp:else>...</emp:else>
</emp:if>
```

The <if> tag contains a condition which will be evaluated in run-time. The condition may be „true" or „false". For simplification the condition will be evaluated by the Java Runtime System. Therefor the condition is a simple Java expression (<% Java condition %>).

Inside the <if> tag there's a <then> tag and / or an <else> tag. The <then> tag will be evaluated if the condition is true. Otherwise the <else> part will be evaluated.

IfTag.java:

```
// sample usage of if tag:
// <%@ taglib uri="/empTaglib" prefix="emp" %>
// ...
// <emp:if condition="<% test %> == 123">
//   <emp:then>...</emp:then>
//   <emp:else>...</emp:else>
// </emp:if>

package de.emprise.taglib;

// import needed for interface "Tag"
import javax.servlet.jsp.tagext.*;
// import needed for class PageContext, JspException, ...
import javax.servlet.jsp.*;

public class IfTag extends BodyTagSupport {

  // Class version string
  public static final String CLASS_VERSION = "IfTag.java,v 1.0, 2001-07-19, Michael Maretzke";
```

```
  // Members for attributes connected to tag
  private boolean _condition;

  //
  // Attribute getter / setter
  //

  // Attribute getter / setter for "condition"
  public void setCondition(String strcond) {
  }
  public void setCondition(boolean condition) {
    _condition = condition;
  }
  public boolean getCondition() {
    return _condition;
  }

  // Called on a Tag handler to release state. The page compiler guarantees this method will be
  // called on all tag handlers, but there may be multiple invocations on doStartTag and doEndTag
  // in between.
  public void release () {
    _condition = false;
  }

  // Process the start tag for this instance.
  public int doStartTag () throws JspException {
    return EVAL_PAGE; // go on evaluating the page
  }

  // Process the end tag. This method will be called on all Tag objects.
  public int doEndTag () throws JspException {
    return EVAL_PAGE;
  }
}
```

ThenTag.java:

```
// sample usage of if tag:
// <%@ taglib uri="/empTaglib" prefix="emp" %>
// ...
// <emp:if condition="<% test %> == 123">
//   <emp:then>...</emp:then>
//   <emp:else>...</emp:else>
// </emp:if>

package de.emprise.taglib;

// import needed for class "BodyTagSupport"
```

```
import javax.servlet.jsp.tagext.*;
// import needed for class PageContext, JspException, ...
import javax.servlet.jsp.*;
import java.io.*;

public class ThenTag extends BodyTagSupport {

  // Class version string
  public static final String CLASS_VERSION = "ThenTag.java,v 1.0, 2001-07-19, Michael Maretzke";

  // Process the start tag for this instance.
  public int doStartTag () throws JspException {
    IfTag parent = (IfTag) findAncestorWithClass(this, IfTag.class);
    if (parent == null)
      throw new JspException("<then> tag is not inside if");

    return EVAL_BODY_TAG;
  }

  // Process the end tag. This method will be called on all Tag objects.
  public int doAfterBody () {
    IfTag parent = (IfTag) findAncestorWithClass(this, IfTag.class);
    if (parent.getCondition()) {
      try {
        BodyContent body = getBodyContent();
        JspWriter out = body.getEnclosingWriter();
        out.print(body.getString());
      }
      catch(IOException e) {
        System.out.println("Error in ThenTag: " + e);
      }
    }
    return SKIP_BODY;
  }
}
```

ElseTag.java:

```
// sample usage of if tag:
// <%@ taglib uri="/empTaglib" prefix="emp" %>
// ...
// <emp:if condition="<% test %> == 123">
//   <emp:then>...</emp:then>
//   <emp:else>...</emp:else>
// </emp:if>

package de.emprise.taglib;
```

```
// import needed for class "BodyTagSupport"
import javax.servlet.jsp.tagext.*;
// import needed for class PageContext, JspException, ...
import javax.servlet.jsp.*;
import java.io.*;

public class ElseTag extends BodyTagSupport {

  // Class version string
  public static final String CLASS_VERSION = "ElseTag.java,v 1.0, 2001-07-19, Michael Maretzke";

  // Process the start tag for this instance.
  public int doStartTag () throws JspException {
    IfTag parent = (IfTag) findAncestorWithClass(this, IfTag.class);
    if (parent == null)
      throw new JspException("<else> tag is not inside if");

    return EVAL_BODY_TAG;
  }

  // Process the end tag. This method will be called on all Tag objects.
  public int doAfterBody () {
    IfTag parent = (IfTag) findAncestorWithClass(this, IfTag.class);
    if (!parent.getCondition()) {
      try {
        BodyContent body = getBodyContent();
        JspWriter out = body.getEnclosingWriter();
        out.print(body.getString());
      }
      catch(IOException e) {
        System.out.println("Error in ElseTag: " + e);
      }
    }
    return SKIP_BODY;
  }
}
```

4 References

[SUN01] http://java.sun.com/products/jsp/taglibraries.html, more hints from SUN

[SUN02] http://java.sun.com/products/jsp/tutorial/TagLibraries.pdf, tutorial for tag libraries, also from SUN

[SUN03] http://developer.java.sun.com/developer/Books/cservletsjsp/chapter14.pdf, more and deeper information on JSPs and custom tags

[SUN04] http://java.sun.com/products/jsp/download.html, specification for JSP

[ATG01] ATG Dynamo 5, Programmers Guide, Version 5.1.1, p. 1149 ff, Usage of J2EE in Dynamo

[ATG02] ATG Dynamo 5, Programmers Guide, Version 5.1.1, p. 1405 ff, Web enclosure programming paradigm

[ATG01] ATG Dynamo 5, Programmers Guide, Version 5.1.1, p. 1443 ff, Usage of ATG Dynamo features in JSP pages – the DSP tag lib