JSP

Tag Extension

JSP custom tag

Ideally, JSP pages should contain no code written in the Java programming language (that is, no expressions or scriptlets). Anything a JSP page needs to do with Java code can be done from a

custom tag

- ***Separation of form and function.***
- ***Separation of developer skill sets and activities.***
- ***Code reusability.***
- ***Clarified system design.***

a JSP custom tag

```
<%@ taglib uri="/hello" prefix="example" %>
<HTML><HEAD><TITLE>First custom tag</TITLE></HEAD>
<BODY>
This is static output
<p />
<i><example:hello>HELLO THERE</example:hello></i>
This is static output
</BODY>
</HTML>
```

hello.doStartTag()

hello.doEndTag()

a JSP custom tag

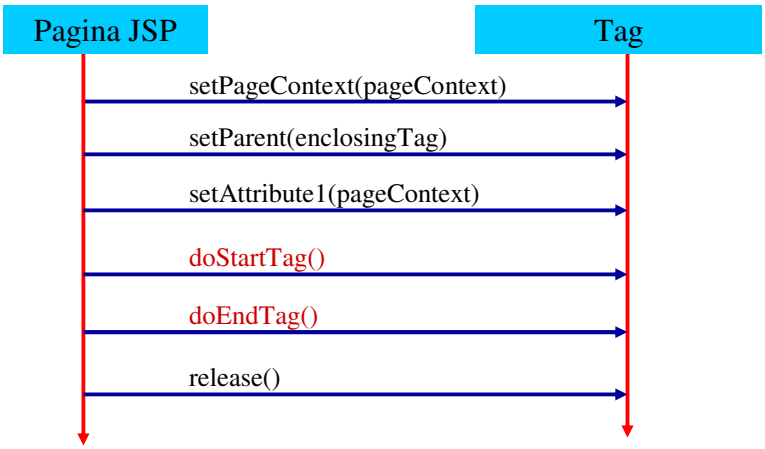
```
package jsptags;
import java.io.IOException;
import java.util.Date;
import javax.servlet.jsp.*;
import javax.servlet.jsp.tagext.*;

public class HelloTag extends TagSupport {
  public int doStartTag() throws JspTagException {
    try {
      pageContext.getOut().write("Start tag found here<BR>");
    } catch (IOException e) {
      throw new JspTagException("Fatal error: could not write to JSP out");
    }
    return EVAL_BODY_INCLUDE;  // return SKIP_BODY;
  }
```

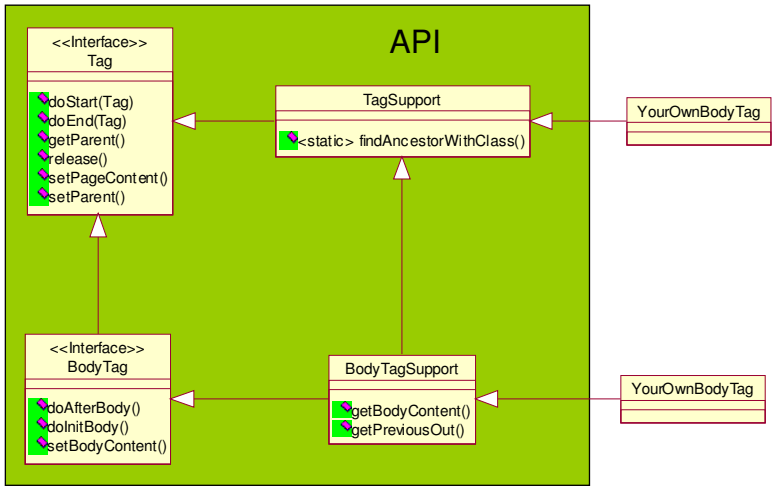
a JSP custom tag

```
...
public class HelloTag extends TagSupport {
...
 public int doEndTag() throws JspTagException {
   try {
     pageContext.getOut().write("End tag found<BR>");
   } catch (IOException e) {
     throw new JspTagException("Fatal error: could not write to JSP out");
   }
   return EVAL_PAGE;      // return SKIP_PAGE;
  }
}
```

Javax.servlet.jsp.tagext.Tag interface



Class Diagram



a JSP custom tag

```
<%@ taglib uri="/hello" prefix="example" %>
<HTML><HEAD><TITLE>First custom tag</TITLE></HEAD>
<BODY>
This is static output
<p />
<i><example:hello>HELLO THERE</example:hello></i>
This is static output
</BODY>
</HTML>
```

hello.doStartTag()

hello.doInitBody()

hello.doAfterBody()

hello.doEndTag()

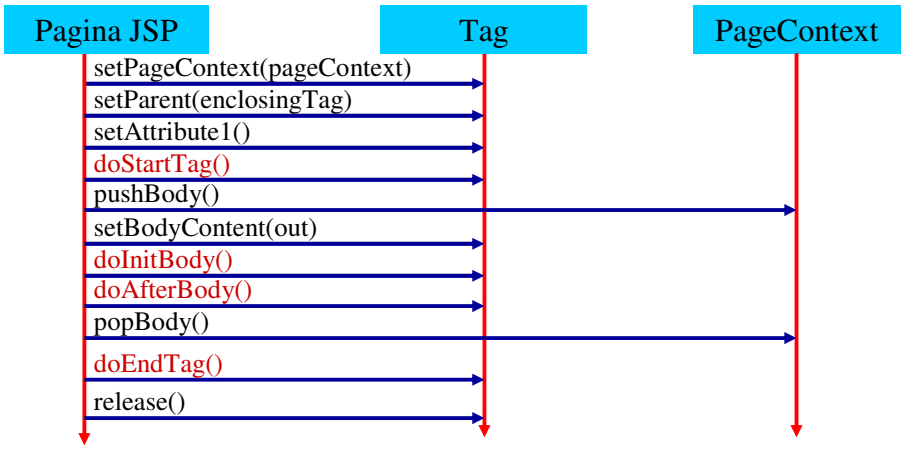
a JSP custom tag

```
package jsptags;
...
public class HelloTag extends BodyTagSupport {
  public int doStartTag() throws JspTagException {
    ...
  }
public void doInitBody() throws JspTagException {
    try {
      pageContext.getOut().write("Init Body<BR>");
    } catch (IOException e) {
      throw new JspTagException("Fatal error: could not write to JSP out");
    }
  }
```

a JSP custom tag

```
  public int doAfterBody() throws JspTagException {
    try {
      pageContext.getOut().write("After Body<BR>");
    } catch (IOException e) {
      throw new JspTagException("Fatal error: could not write to JSP out");
    }
    return EVAL_BODY_TAG; // return SKIP_BODY;
  } */
 public int doEndTag() throws JspTagException {
    ...
  }
}
```

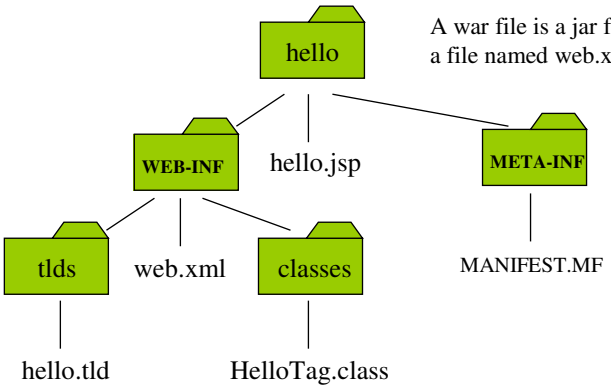
Javax.servlet.jsp.tagext.BodyTag interface



reversing body content

```
import java.io.IOException; import javax.servlet.jsp.*; import javax.servlet.jsp.tagext.*;
public class ReverseTag extends BodyTagSupport {
    public int doEndTag() throws JspTagException {
        BodyContent bodyContent = getBodyContent();
        if (bodyContent != null) {// Do nothing if there was no body content
           StringBuffer output = new StringBuffer(bodyContent.getString());
            output.reverse();
            try {
               bodyContent.getEnclosingWriter().write(output.toString());
           } catch (IOException ex) {
               throw new JspTagException("Fatal IO error");
          }
        } return EVAL_PAGE;
    }
}
```

structure of the war file



A war file is a jar file with special directories and a file named web.xml in the WEB-INF directory

TLD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE taglib
      PUBLIC "-//Sun Microsystems, Inc.//DTD JSP Tag Library 1.1//EN"
      "http://java.sun.com/j2ee/dtds/web-jsptaglibrary_1_1.dtd">
<taglib>
  <tlibversion>1.0</tlibversion>
  <jspversion>1.1</jspversion>
  <shortname>examples</shortname>
  <info>Simple example library.</info>
  <tag>
    <name>reverse</name>
    <tagclass>tagext.ReverseTag</tagclass>
    <bodycontent>JSP</bodycontent>
    <info>Simple example</info>
  </tag>
</taglib>
```


7

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN'
    'http://java.sun.com/j2ee/dtds/web-app_2.2.dtd'>
<web-app>
 <display-name>tagext</display-name>
 <description>Tag extensions examples</description>
 <session-config>
  <session-timeout>0</session-timeout>
 </session-config>

 <taglib>
    <taglib-uri>/hello</taglib-uri>
    <taglib-location>/WEB-INF/tlds/hello.tld</taglib-location>
 </taglib>

</web-app>
```

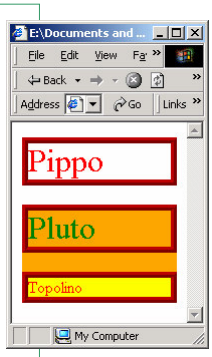
CSS

Cascading Style Sheet
CSS2 – CSS/P

Nota: tratteremo solo le proprieta' definite a livello di standard ed attualmente implementate sia da Netscape che da Explorer

Esempio di cascata di stili

```
<HTML>
<HEAD>
    <STYLE TYPE="text/css">
        p {font-size:24pt;color:green;
        border-width:thick;border-style:ridge;
        border-color:red}
        p.red {color:red}
    </STYLE>
</HEAD>
<BODY>
    <p class="red">Pippo</p>
    <div style="background:orange;border-color:green">
        <p>Pluto</p>
        <p class="red" style="font-size:12pt;
        background:yellow">Topolino</p>
    </div>
</BODY>
```



Elementi di formattazione

color:*color*
background-color:*color*
background-image:*uri*

font-family:name
font-size:xx-small|x-small|small|medium|large|x-large|xx-large|
larger|smaller|*absoluteSize*|*relativeSize*|*percentage*|*lenght*
font-style:normal|italic
font-weight:bold|bolder|lighter|normal|100|200|...|800|900

Unita' di lunghezza

LUNGHEZZE ASSOLUTE:
Sistema Internazionale
cm *centimetri*
mm *millimetri*

Unita' anglosassoni
in *inch (pollici)*
pt *point:1/72 pollice*
pc *pica:12 point=1/6 pollice*

System dependent:
px *pixel*

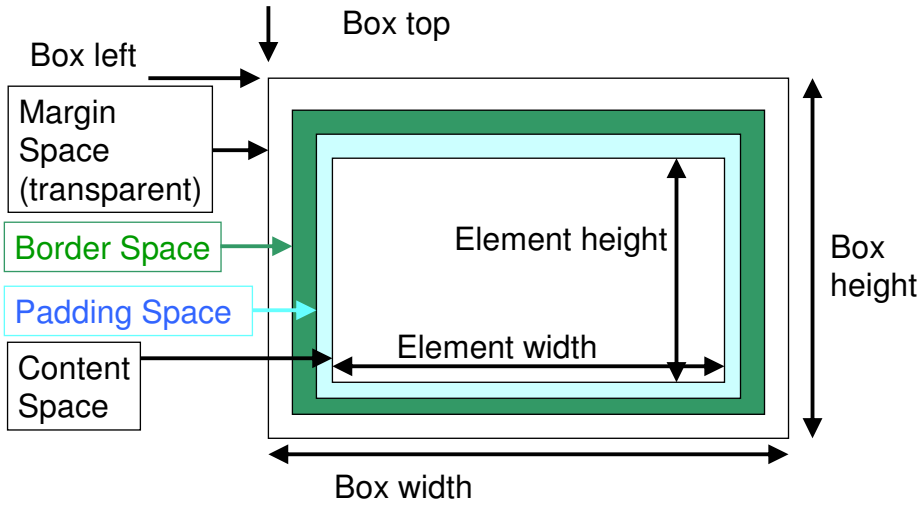
LUNGHEZZE RELATIVE:
em *altezza del font dell'elemento*
ex *altezza estesa del font dell'elemento*

Elementi di formattazione

line-height:normal|*length*|*percentage*

text-align:left|center|right|justify
text-decoration:blink|line=through|overline|underline
text-indent:*lenght*|*percentage*
text-transform:none|capitalize|uppercase|lowercase

Elementi di formattazione



Elementi di formattazione

border-bottom-width,border-top-width,
border-right-width,border-left-width:thin|medium|thick|n
border-color:*color*
border-style:double|groove|none|inset|outset|ridge|solid
border-width:thin|medium|thick|*length*

margin:*thickness*
margin-bottom,margin-left,margin-right,margin-top :*thickness*

padding :*thickness*
padding-bottom,padding-left,padding-right,padding-top :*thickness*

Elementi di formattazione - liste

<LI>

<OL>
line-style:decimal|lower-alpha|lower-roman|upper-alpha|upper-roman

<UL>
line-style:circle|disc|square

Wrapping text on an element

<P> Un testo molto molto molto molto molto molto lungo.
<SPAN STYLE="float:right;width:120">
<IMG SRC="UrlOfMyImage.gif" height="90" width="120">
</SPAN>
Un testo molto molto molto molto molto molto lungo.
</P>

Caratteristiche avanzate

Selettore di adiacenza H1+H2 {color:blue}
I blocchi H1 e H2 consecutivi

Selettore Padre-figlio BODY > P{color:red}
I Paragrafi direttamente contenuti nel BODY

Selettore di attributo P[ALIGN]{color:red}
Individua solo i <P ALIGN=…>
P[ALIGN="left"]{color:red}
Individua solo i <P ALIGN="left">
*[ALIGN="left"]{color:red}
Individua solo qualunque tag con attributo ALIGN="left"

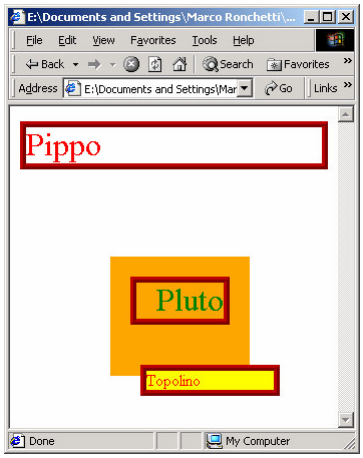
Caratteristiche avanzate

Pseudo-elementi
P:first-letter *prima lettera*
P:first-line *prima linea*

Pseudo-classi
A:link *ancore non visitate*
A:active *ancore nell'atto di essere cliccate*
A:visited *ancore visitate*

Esempio di posizionamento: CSS-P

```
<HTML>
<HEAD>
      <STYLE TYPE="text/css">
      #aposition{position:relative;left:30;top:20}
      a {color:red}
      p {font-size:24pt;color:green;border-width:thick;
            border-style=ridge;border-color=red}
      p.red {color:red}
      </STYLE>
</HEAD>
<BODY>
      <p class="red">Pippo</p>
      <div style="background:orange;border-color=green;
            position:absolute;left:100;top:150">
            <p style="padding-left:20;margin:20">Pluto</p>
            <span id="aposition">
                  <p class="red" style="font-size:12pt;
                  background-color:yellow">Topolino</p>
            </span>
      </div>
</BODY>
</HTML>
```



Elementi posizionabili – CSS/P

Specificazione del tipo di posizionamento:
position:absolute|relative

Specificazione della posizione:
top:*size*
left:*size*

Specificazione della dimensione:
width:*size*
height:*size*

Specificazione del tipo di visibilita':
visibility:hidden|inherit|show

Elementi posizionabili – CSS/P

Clip:rect(*top right bottom left*)
Taglia un elemento lasciandolo al suo posto

z-index:*n*
stabilisce l'ordine di visualizzazione degli elementi sovrapposti