

# eXtensible Markup Language (XML)

1

## XML

- [www.w3.org/XML](http://www.w3.org/XML)
- Estándar W3C, versión 1.0 en 1998
- Lenguaje de meta-marcado
- Intercambio de datos, gráficos vectoriales, serialización de objetos, XSL, etc.
- Química, música, propiedades inmobiliarias, matemáticas, etc.
- Etiquetas, elementos y atributos con estructura de árbol
- Validación: DTD para restringir sintaxis
- Parsers: SAX, DOM, JDOM

2

## Estándares relacionados

- XML procede de SGML
  - Standard Generalized Markup Language, IBM, 70's
  - Complejidad de SGML (150 páginas para su especificación)
- XHTML es una aplicación XML
  - HTML mezcla datos y formateo
  - XML se puede ver como una generalización de HTML que explicita estructuras de datos
- Aplicaciones XML: XSL, SVG, SMIL, MathML, RDF, SOAP...

3

## Sintaxis XML

- Estructura de árbol
- Elementos y atributos
- Cualquier etiqueta es válida  
(por ejemplo, en HTML se pueden usar unos 100 tags)

4

## Ejemplo

```

Raíz
  <Person birth = "1912" death = "1954" >
    <name>
      <firstName> Alan </firstName> ← Elemento
      <lastName> Turing </lastName> ← Etiqueta
    </name>
    <profession> computer scientist </profession>
    <profession> mathematician </profession>
    <profession> cryptographer </profession>
  </Person>
  
```

5

## Validación con DTD

- Document Type Definition
- No es obligatorio
- Especifican un sublenguaje de XML
  - Lista de etiquetas permitidas
  - Etiquetas y atributos permitidos dentro de cada etiqueta
- Pueden ser muy complejos
  - Para SVG (Scalable Vector Graphics) más de 1000 líneas
  - Para XHTML 1.0 más de 1,500 líneas
  - Para DocBook más de 11.000 líneas
- Alternativa: XML Schema (versión 1.0, mayo 2001)

6

### Ejemplo

```

<!ELEMENT person (name, profession*)>
<!ATTLIST person birth CDATA #IMPLIED
                death CDATA #IMPLIED>
<!ELEMENT name (first_name, last_name)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT profession (#PCDATA)>

```

### XML en la Web

- IE 4.0 incluye un parser XML para VBScript y JavaScript
- IE 5.0, Netscape 6.0 y Mozilla permiten visualizar XML y
- IE 5.0 / 5.5 soportan versión propia de XSLT incompatible con XSLT 1.0
- Mozilla y Netscape no soportan XSLT
- XML para intercambio de datos entre cliente (p.e. applet) y servidor (p.e. servlet)

### XML en Java 1.4

- Parsing de XML
  - Document Object Model: org.w3c.dom
  - Simple API for XML: org.xml.sax
  - JAXP: librerías adicionales a la SDK con extensiones a DOM y SAX
- Conversión XML ↔ JavaBean: java.beans
- Procesamiento de XSLT 1.0: javax.xml.transform
- JavaServer Pages (JSP): javax.servlet.jsp (+ servidor web), sólo en J2EE

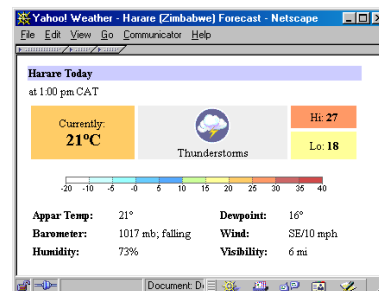
### Generación dinámica de documentos web

### Páginas web dinámicas

- En cliente
  - Tecnología soportada por navegadores web
  - JavaScript, VBScript, Java Applets, DHTML...
- En servidor
  - URL designa un programa, admite argumentos
    - `http://hostname:port?param_1=value_1&param_2=value_2&...`
  - Estándares
 

▫ Common Gateway Interface 1.1 (CGI)	- Un proceso por conexión
▫ C/C++, Fortran, PERL, TCL, Any Unix shell, Visual Basic, AppleScript	- Sin estado
▫ Java Servlets 2.3, mediados 1997	- Único proceso, un hilo por conexión
▫ JavaServer Pages 1.2 (JSP), finales 1999	- Estado: sesión, aplicación
- Fuente de datos: BD, XML, usuario, plataforma, otras URL's...

### Ejemplo: Yahoo! Weather Forecast



### Fuente HTML

```
<html><head><title>Yahoo! Weather - Harare (Zimbabwe) Forecast
</title></head><body><table width=100%>
<tr bgcolor=CCCCFF><td><b>Harare Today</b></td></tr>
<tr><td>at 1:00 pm CAT</td></tr>
<tr><td>
<table width=100%><tr align=center>
<td rowspan=2 bgcolor=FFCC66>Currently:
<br><b><font size=+2>21</font>C</b></td>
<td rowspan=2 bgcolor=EEEEEE><img src=thunderstorm.gif>
<br>Thunderstorms</td>
<td bgcolor=FF9966>Hi: <b>27</b></td></tr>
<tr align=center><td bgcolor=FFFF99>Lo: <b>18</b></td>
</tr></table>
...

```

13

```
...
<p><center><img src=cscale.gif></center>
<p><table width=100%>
<tr><td><b>Appar Temp:</b></td><td>21</td>
<td><b>Dewpoint:</b></td><td>16</td>
<tr><td><b>Barometer:</b></td><td>1017 mb; falling</td>
<td><b>Wind:</b></td><td>SE/10 mph</td>
<tr><td><b>Humidity:</b></td><td>73%</td>
<td><b>Visibility:</b></td><td>6 mi</td>
</tr></table>
</table>
</body></html>

```

14

Página dinámica con servlet

```
public class WeatherServlet extends HttpServlet {
    public void doGet (HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        PrintWriter out = response.getWriter ();
        out.println ( // HTML source here
            "<html><head><title>Yahoo! Weather - Harare (Zimbabwe) *
            + "Forecast</title></head><body><table width=100%>
            + "<tr bgcolor=CCCCFF><td><b>Harare Today</b></td></tr>
            + "<tr><td>at 1:00 pm CAT</td></tr><tr><td>
            + "<table width=100%><tr align=center>
            + "<td rowspan=2 bgcolor=FFCC66>Currently:"
            + "<br><b><font size=+2>21</font>C</b></td>
            + "<td rowspan=2 bgcolor=EEEEEE>
            + "<img src=thunderstorm.gif><br>Thunderstorms</td>
            + "<td bgcolor=FF9966>Hi: <b>27</b></td></tr>
            + "<tr align=center><td bgcolor=FFFF99>Lo: <b>18</b></td>
            + "</tr></table>"
            ...

```

15

```
...
+ "<p><center><img src=cscale.gif></center>"
+ "<p><table width=100%>
+ "<tr><td><b>Appar Temp:</b></td><td>21</td>
+ "<td><b>Dewpoint:</b></td><td>16</td>
+ "<tr><td><b>Barometer:</b></td><td>1017 mb; falling</td>
+ "<td><b>Wind:</b></td><td>SE/10 mph</td>
+ "<tr><td><b>Humidity:</b></td><td>73%</td>
+ "<td><b>Visibility:</b></td><td>6 mi</td>
+ "</table></tr></table></body></table>"
); // End HTML source
out.close ();
}
}

```

16

### Datos en XML

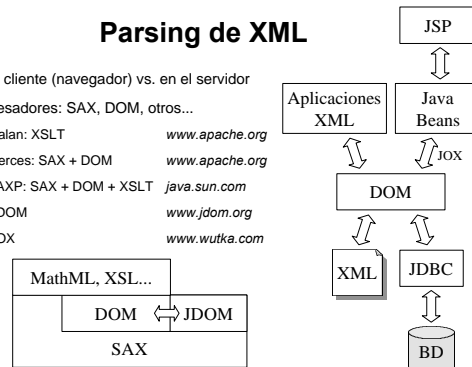
```
<!-- weather.xml -->
<weather>
  <city name = "Harare" country = "Zimbabwe">
    <curtemp> 21 </curtemp>
    <maxtemp> 27 </maxtemp>
    <mintemp> 18 </mintemp>
    <appartemp> 21 </appartemp>
    <dewpoint> 16 </dewpoint>
    <barometer> 1017 </barometer>
    <wind> 10 </wind>
    <humidity> 73 </humidity>
    <visibility> 6 </visibility>
    <sky> Thunderstorms </sky>
  </city>
  ...
</weather>

```

17

### Parsing de XML

- En el cliente (navegador) vs. en el servidor
- Procesadores: SAX, DOM, otros...
  - Xalan: XSLT [www.apache.org](http://www.apache.org)
  - Xerces: SAX + DOM [www.apache.org](http://www.apache.org)
  - JAXP: SAX + DOM + XSLT [java.sun.com](http://java.sun.com)
  - JDOM [www.jdom.org](http://www.jdom.org)
  - JOX [www.wutka.com](http://www.wutka.com)



18

### Parsing de XML con XSLT

```
<!-- weather.xml -->

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--
<xsl:param name="cityName"/>

<xsl:template match="weather">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="city[@name='$cityName']">
-->
...

```

19

```
...
<xsl:template match="city">

<html><head><title>Yahoo! Weather -
<xsl:value-of select="name"/> (<xsl:value-of select="country"/>)
Forecast</title></head><body>

<table width="100%">
<tr bgcolor="#CCCCFF"><td><b>
<xsl:value-of select="name"/> Today</b></td></tr>
<tr><td>at 1:00 pm CAT</td></tr>
...

```

20

```
...
<tr><td><table width="100%"><tr align="center">

<td rowspan="2" bgcolor="#FFCC66">Currently:
<br/><b><font size="+2">
<xsl:value-of select="curtemp"/>
</font></b></td>

<td rowspan="2" bgcolor="EEEEEE">
<br/><xsl:value-of select="sky"/></td>

<td bgcolor="#FF9966">Hi:
<b><xsl:value-of select="maxtemp"/></b></td></tr>

<tr align="center">
<td bgcolor="#FFFF99">Lo:
<b><xsl:value-of select="mintemp"/></b></td></tr>
</table>

<p/><center></center>
...

```

21

```
...
<p/><table width="100%">

<tr><td><b>Appar Temp:</b></td><td>
<xsl:value-of select="appartemp"/></td>
<td><b>Dewpoint:</b></td><td>
<xsl:value-of select="dewpoint"/></td></tr>

<tr><td><b>Barometer:</b></td><td>
<xsl:value-of select="barometer"/> mb; falling</td>
<td><b>Wind:</b></td><td>
<xsl:value-of select="wind"/> mph</td></tr>

<tr><td><b>Humidity:</b></td><td>
<xsl:value-of select="humidity"/>%</td>
<td><b>Visibility:</b></td><td>
<xsl:value-of select="visibility"/> mi</td></tr>

</table></td></tr></table></body></html>
</xsl:template>
</xsl:stylesheet>

```

22

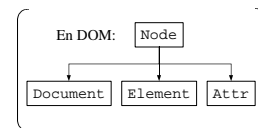
### Algunas limitaciones de XSLT

- No permite selección más elaborada de templates
  - p.e. sobre "jerarquía de tipos de nodo XML"
- Sólo tiene iteración sobre listas de nodos (aunque permite recursión)
- Acceso a BD requiere código Java
- Control de flujo va tipo por tipo de nodo, p.e. es difícil expresar:
  - condiciones generales para grupos de tipos de nodos
  - condiciones sobre otros nodos circundantes
- ...

23

### JDOM

- Similar a DOM (documento como un árbol), sintaxis más sencilla
- Fácil conversión a DOM y viceversa
- Clases principales: **Document**, **Element**, **Attribute**, **Text**, **Comment**, **Namespace**, **ProcessingInstruction**...
- Las clases no forman jerarquía como en DOM
- No es un estándar W3C
- [www.jdom.org](http://www.jdom.org)



24

**Recorrido y modificación del documento/árbol**

- **Element**
  - Element(String name)
  - getAttribute(String name) → Attribute
  - getAttributeValue(String name) → String
  - getChild(String name) → Element
  - getChildText(String name) → String
  - getChildren(String name) → List
  - getText() → String
  - getParent() → Element
  - addAttribute(Attribute attr)
  - addContent(Element elt | String text)
- **Attribute**
  - Attribute(String name, String value)
  - getValue() → String
  - setValue(String value)
  - getParent() → Element
- **Document**
  - getRootElement() → Element

25

### Parsing y conversiones

- **SAXBuilder**
  - build(URL url) → Document
- **DOMBuilder**
  - build(URL url) → Document
  - build(org.w3c.dom.Document) → Document
  - build(org.w3c.dom.Element) → Document
- **DOMOutputter**
  - output(Document) → org.w3c.dom.Document
  - output(Element) → org.w3c.dom.Element
  - output(Attribute) → org.w3c.dom.Attr
- **XMLOutputter**
  - output(Document doc, OutputStream out)
  - output(Element elt, OutputStream out)

26

### Parsing de XML con JDOM en Servlet

```

public class WeatherServlet extends HttpServlet {
    public void doGet (HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String cityName = request.getParameter ("city");
        PrintWriter out = response.getWriter ();
        Element root;
        try {
            SAXBuilder builder = new SAXBuilder();
            Document doc = builder.build (new File ("weather.xml"));
            root = doc.getRootElement ();
        } catch (JDOMException ex) { ex.printStackTrace ();
            out.println ("<html><body>Internal error</body></html>");
            out.close (); return;
        }
        ...
    }
}
    
```

27

```

...
Element city = null;
List cities = root.getChildren ("city");
for (Iterator iter = cities.iterator (); iter.hasNext ();) {
    city = (Element) iter.next ();
    if (city.getAttributeValue ("name") .equals (cityName)) break;
}
if (!city.getAttributeValue ("name") .equals (cityName)) {
    out.println ("<html><body> City does not exist </body></html>");
    out.close ();
    return;
}
...
    
```

28

```

...
String country = city.getAttributeValue ("country");
String curtemp = city.getChildText ("curtemp");
String maxtemp = city.getChildText ("maxtemp");
String mintemp = city.getChildText ("mintemp");
String appartemp = city.getChildText ("appartemp");
String dewpoint = city.getChildText ("dewpoint");
String barometer = city.getChildText ("barometer");
String wind = city.getChildText ("wind");
String humidity = city.getChildText ("humidity");
String visibility = city.getChildText ("visibility");
String sky = city.getChildText ("sky");
...
    
```

29

```

...
out.println (
    // HTML source here
    "<html><head><title>Yahoo! Weather - "
    + cityName + " (" + country + ") Forecast</title></head>"
    + "<body><table width=100%>"
    + "<tr bgcolor=CCCCFF><td><b> " + cityName + " Today</b></td></tr>"
    + "<tr><td>at 1:00 pm CAT</td></tr><tr><td>"
    + "<table width=100%><tr align=center>"
    + "<td rowspan=2 bgcolor=FFCC66>Currently:"
    + "<br><b><font size=+2> " + curtemp + "&ordm;C</font></b></td>"
    + "<td rowspan=2 bgcolor=EEEEEE>"
    + "<img src=thunderstorm.gif><br> " + sky + "</td>"
    + "<td bgcolor=FF9966>Hi: <b> " + maxtemp + "</b></td></tr>"
    + "<tr align=center><td bgcolor=FFFF99>Lo: <b> " + mintemp + "</b>"
    + "</td></tr></table>"
    ...
    );
    
```

30

```

...
+ "<p><center><img src=cscale.gif</center>"
+ "<p><table width=100%>"
+ "<tr><td><b>Appar Temp:</b></td><td>"
+ "  <b>appartemp + " &deg;</td>"
+ "<td><b>Dewpoint:</b></td><td>"
+ "  <b>dewpoint + " &deg;</td></tr>"
+ "<tr><td><b>Barometer:</b></td><td>"
+ "  <b>barometer + " mb; falling</td>"
+ "<td><b>Wind:</b></td><td>" + "  <b>wind + " mph</td></tr>"
+ "<tr><td><b>Humidity:</b></td><td>" + "  <b>humidity + "%</td>"
+ "<td><b>Visibility:</b></td><td>"
+ "  <b>visibility + " mi</td></tr>"
+ "</table></tr></table></body></html>"
); // End HTML source
out.close ();
}
}

```

31

## Mismo ejemplo en DOM

```

public class WeatherServletXSL extends HttpServlet {
    public void doGet (HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        String cityName = request.getParameter ("city");
        PrintWriter out = response.getWriter ();

        DOMParser parser = new DOMParser ();
        try { parser.parse ("weather.xml"); }
        catch (SAXException ex) {
            ex.printStackTrace ();
            out.println ("<html><body>Internal error</body></html>");
            out.close (); return;
        }

        Document doc = parser.getDocument ();
        ...
    }
}

```

32

```

...
Element city = null;
NodeList cities = doc.getElementsByTagName ("city");
for (int i = 0; i < cities.getLength (); i++) {
    city = (Element) cities.item (i);
    if (city.getAttribute ("name") .equals (cityName)) break;
}
if (!city.getAttribute ("name") .equals (cityName)) {
    out.println ("<html><body>City does not exist</body></html>");
    out.close ();
    return;
}
...

```

33

```

...
String country = city.getAttribute ("country");
String curtemp = city.getElementsByTagName ("curtemp") .item (0)
    .getNodeValue ();
String maxtemp = city.getElementsByTagName ("maxtemp") .item (0)
    .getNodeValue ();
String mintemp = city.getElementsByTagName ("mintemp") .item (0)
    .getNodeValue ();
String appartemp = city.getElementsByTagName ("appartemp") .item (0)
    .getNodeValue ();
String dewpoint = city.getElementsByTagName ("dewpoint") .item (0)
    .getNodeValue ();
...

```

34

```

...
String barometer = city.getElementsByTagName ("barometer") .item (0)
    .getNodeValue ();
String wind = city.getElementsByTagName ("wind") .item (0)
    .getNodeValue ();
String humidity = city.getElementsByTagName ("humidity") .item (0)
    .getNodeValue ();
String visibility = city.getElementsByTagName ("visibility") .item (0)
    .getNodeValue ();
String sky = city.getElementsByTagName ("sky") .item (0)
    .getNodeValue ();
...
// El resto del ejemplo igual que con JDOM...

```

35

## Parsing de XML con XSLT en Servlet

```

public class WeatherServletXSL extends HttpServlet {
    public void doGet (HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {

        String cityName = request.getParameter ("city");
        PrintWriter out = response.getWriter ();

        Element root;
        try {
            SAXBuilder b = new SAXBuilder ();
            Document doc = b.build (new File ("weather.xml"));
            root = doc.getRootElement ();
            ...
        }
    }
}

```

36

```

...
Element city = null;
List cities = root.getChildren ("city");
for (Iterator iter = cities.iterator (); iter.hasNext ();) {
    city = (Element) iter.next ();
    if (city.getAttributeValue ("name") .equals (cityName)) break;
}
if (!city.getAttributeValue ("name") .equals (cityName)) {
    out.println ("<html><body>City does not exist</body></html>");
    out.close ();
    return;
}
...
    
```

37

```

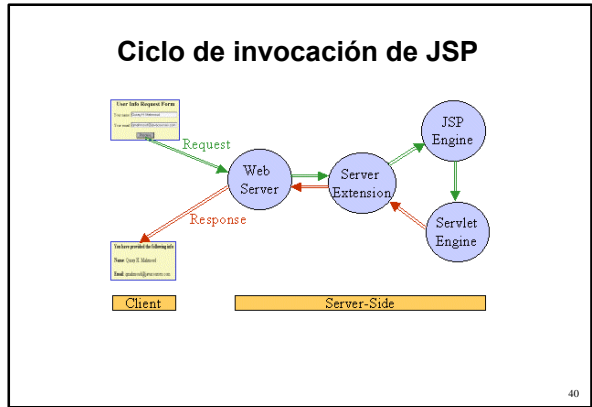
...
XSLTProcessor processor = XSLTProcessorFactory.getProcessor (
    new XercesLiaison());
XSLTInputSource xmlSource = new XSLTInputSource (
    new DOMOutputter() .output (city)); // doc
XSLTInputSource xslSheet = new XSLTInputSource ("weather.xml");
XSLTResultTarget xmlResult = new XSLTResultTarget (out);
// processor.setStylesheetParam ("cityName",
// "" + cityName + "");
processor.process (xmlSource, xslSheet, xmlResult);
} catch (SAXException ex) {
    out.println ("<html><body>Internal error</body></html>");
    ex.printStackTrace ();
} catch (JDOMException ex) {
    out.println ("<html><body>Internal error</body></html>");
    ex.printStackTrace ();
}
out.close ();
}
    
```

38

### JavaServer Pages (JSP)

- Generación dinámica de páginas web
- Permiten mezclar código HTML y código Java
  - <% expresión %>
  - <% sentencia; %>
  - Se pueden utilizar variables implícitas: request, response, y otras
- Necesita un servidor web que soporte JSP, p.e. J2EE, Apache Tomcat, Mort Bay Jetty (jetty.mortbay.com), BEA WebLogic, iPlanet, IBM WebSphere...
- Se accede igual que a una página html desde un navegador
- El servidor de JSP compila el documento JSP la primera vez que es accedido
  - Genera un servlet en un .java (en el servidor)
  - Compila el servlet, lo carga, y lo ejecuta (en el servidor)
- En realidad la salida no tiene por qué ser HTML: XML, WML, etc.

39



40

### Ejemplo

```

<% page import="org.jdom.*" %>
<% page import="java.util.*" %>
<!-- Mas imports... -->

<% String cityName = request.getParameter ("city");
    Element root;
    try {
        SAXBuilder b = new SAXBuilder();
        Document doc = b.build (new File ("weather.xml"));
        root = doc.getRootElement ();
    } catch (JDOMException ex) {
        ex.printStackTrace ();
    }
%>
<html><body> Internal error </body></html>
<%
    return;
}
...
    
```

41

```

...
Element city = null;
List cities = root.getChildren ("city");
for (Iterator iter = cities.iterator (); iter.hasNext ();) {
    city = (Element) iter.next ();
    if (city.getAttributeValue ("name") .equals (cityName)) break;
}
if (!city.getAttributeValue ("name") .equals (cityName))
%>
<html><body> City does not exist </body></html>
<%
    
```

42

```

...
else {
    String country = city.getAttributeValue ("country");
    String curtemp = city.getChildText ("curtemp");
    String maxtemp = city.getChildText ("maxtemp");
    String mintemp = city.getChildText ("mintemp");
    String appartemp = city.getChildText ("appartemp");
    String dewpoint = city.getChildText ("dewpoint");
    String barometer = city.getChildText ("barometer");
    String wind = city.getChildText ("wind");
    String humidity = city.getChildText ("humidity");
    String visibility = city.getChildText ("visibility");
    String sky = city.getChildText ("sky");
}
...

```

```

...
<html><head><title>Yahoo! Weather -
<%= cityName %> (<%= country %>) Forecast</title></head><body>

<table width=100%>
<tr bgcolor=CCCCFF><td><b> <%= cityName %> Today</b></td></tr>
<tr><td>at 1:00 pm CAT</td></tr>

<tr><td><table width=100%><tr align=center>
<td rowspan=2 bgcolor=FFCC66>Currently:
<br><b><font size=+2> <%= curtemp %> &ordm;C</font></b></td>
<td rowspan=2 bgcolor=EEEEEE><img src=/thunderstorm.gif>
<br> <%= sky %> </td>
<td bgcolor=FF9966>Hi: <b> <%= maxtemp %> </b></td></tr>
<tr align=center><td bgcolor=FFFF99>Lo: <b> <%= mintemp %> </b></td>
</tr></table>

<p><center><img src=/cscale.gif></center>
...

```

```

...
<p><center><img src=/cscale.gif></center>

<p><table width=100%>

<tr><td><b>Appar Temp:</b></td><td> <%= appartemp %>&deg; </td>
<td><b>Dewpoint:</b></td><td> <%= dewpoint %>&deg; </td></tr>

<tr><td><b>Barometer:</b></td><td> <%= barometer %> mb; falling</td>
<td><b>Wind:</b></td><td> <%= wind %> mph</td></tr>

<tr><td><b>Humidity:</b></td><td> <%= humidity %> %</td>
<td><b>Visibility:</b></td><td> <%= visibility %> mi</td></tr>

</table></td></tr></table></body></html>

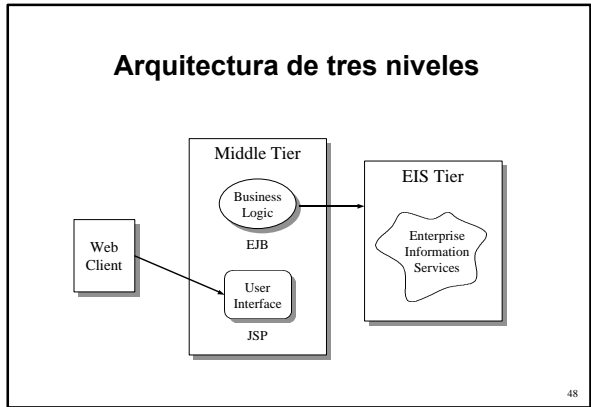
<% } %>

```

### Sintaxis JSP

- Directivas
  - <%@ page import="java.util.\*" %>
  - <%@ include file="abc.html" %>
  - <%@ taglib uri="org.apache.jakarta.\*" prefix="" %>
- Acciones: beans (use, get, set), include, forward, otras...
  - <jsp:forward page="nextPage.jsp" %>
  - <jsp:include page="nextPage.jsp" %>
- Scriptlets
  - <% sentencias... %>
  - <%= expresión %>
- JSP en formato XML

- ### Scope
- Page
  - Request
  - Session
    - Session ID se genera en el servidor
    - Se intercambia con el cliente por medio de Cookies | SSL | URL rewrite (no todos los navegadores lo soportan)
    - encodeURL (String url) en JSP hace URL rewrite (ya que el cliente puede no hacerlo)
    - No funciona fuera de conexiones a Servlets
  - Application
  - Los cuatro scopes son accesibles como variables implícitas en JSP



### Interfaz con Java Beans

- `<jsp:useBean id="x" class="C" scope="..." />`
  - Busca un bean de la clase C asociado al nombre x
  - Si no existe crea uno
  - scope indica si el bean es visible en la página, petición, sesión o aplicación
  - En adelante x es también una variable Java que se puede utilizar en scriptlets
- `<jsp:getProperty name="x" property="p" />`
  - Devuelve `x.getP()`
- `<jsp:setProperty name="x" property="p" value="valor" />`
  - Ejecuta `x.setP("valor")`

49

### Ejemplo

```
// Definir clase de bean
public class CityBean {
    String name, country, curtemp, maxtemp, mintemp, appartemp,
        dewpoint, barometer, wind, humidity, visibility, sky;
    public String getName () { return name; }
    public void setName (String str) { name = str; }
    public String getCountry () { return country; }
    public void setCountry (String str) { country = str; }
    public String getCurtemp () { return curtemp; }
    public void setCurtemp (String str) { curtemp = str; }
    ... // Etc.
}
```

50

```
<!-- Usar el bean en JSP -->
<%@ page import="org.jdom.*" %>
...
// Obtener DOM Element 'city' para la ciudad pedida
// (igual que en ejemplo anterior)
...

// Declarar cityBean
<jsp:useBean id="cityBean" class="CityBean" scope="session"/>

<% // Meter los datos de city en cityBean (usando JOX)
try { new JOXBeanBuilder (city) .readObject (cityBean); }
catch (Exception ex) { ex.printStackTrace(); }
%>
...

```

51

```
... // Usar las propiedades del bean
<html><head><title>Yahoo! Weather -
<jsp:getProperty name="cityBean" property="name"/>
(<jsp:getProperty name="cityBean" property="country"/>)
Forecast</title></head><body>
<table width=100%>
<tr bgcolor=CCCCFF><td><b>
<jsp:getProperty name="cityBean" property="name"/>
Today</b></td></tr>
<tr><td>at 1:00 pm CAT</td></tr>

<tr><td><table width=100%><tr align=center>
<td rowspan=2 bgcolor=FFCC66>Currently:
<br><b><font size=+2>
<jsp:getProperty name="cityBean" property="curtemp"/> &ordm;C
</font></b></td>
<td rowspan=2 bgcolor=EEEEEE><img src=/thunderstorm.gif>
... // Etc.

```

52

### Custom action tags

- Tags definidos por el programador
- Se utilizan en ficheros JSP como los action tags predefinidos
- Se ejecuta código Java cuando se procesan
- Se agrupan por librerías de Tags
- Jakarta Taglibs ([jakarta.apache.org/taglibs](http://jakarta.apache.org/taglibs))
  - Dates
  - E-mail
  - XSL
  - SQL
  - String manipulation
  - Otras...
- Comunicación entre tags anidados
- Iteración

53

### Ejemplo: jakarta XSL taglib

```
<%@ taglib uri="http://jakarta.apache.org/taglibs/xsl-1.0"
    prefix="xsl" %>

<xsl:apply xml="documento.xml" xsl="plantilla.xsl"/>

<xsl:apply xsl="plantilla.xsl">
    ... (Código XML aquí) ...
</xsl:apply>

```

54

### Creación de custom tags

- Para cada tag se debe definir una clase java asociada, subclase de TagSupport o BodyTagSupport
- La clase java hereda y/o sobrescribe métodos de TagSupport que se ejecutan en distintos momentos del procesamiento del tag
- Los body tags (BodyTagSupport) permiten ejecutar acciones también al procesar el cuerpo del tag
- Para cada librería de tags (conjunto de tags) se crea un fichero TLD donde se describe para cada tag:
  - Nombre
  - Clase asociada
  - Atributos que tiene

55

### Ejemplo de creación de custom tag

```

<!-- 1. Crear fichero TLD con descripción del tag -->
<!-- mylib.tld -->
...
<taglib>
  <shortname> test </shortname>
  <tag>
    <name> hello </name>
    <tagclass> HelloTag </tagclass>
    <attribute> name </attribute>
  </tag>
</taglib>
    
```

Annotations in the image:

- Nombre de la librería: points to <shortname> test </shortname>
- Nombre del tag: points to <name> hello </name>
- Clase Java asociada (a definir después): points to <tagclass> HelloTag </tagclass>
- Atributos del tag: points to <attribute> name </attribute>
- Bloque de descripción de tag: points to the entire <tag>...</tag> block.

56

```

// 2. Definir subclase de TagSupport o BodyTagSupport
public class HelloTag extends TagSupport {
  private String name = "World";

  // Un método set por cada atributo
  public void setName (String str) { name = str; }

  // Se sobrescriben métodos de TagSupport, como doEndTag ()
  public int doEndTag () {
    try { pageContext.getOut () .print ("Hello " + name); }
    catch (IOException ex) { ex.printStackTrace (); }
    return EVAL_PAGE;
  }
}
    
```

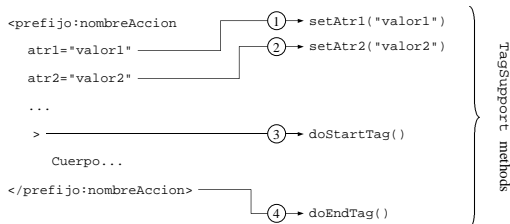
57

```

<!-- 3. Usar tag en fichero jsp -->
<%@ taglib uri="mylib.tld" prefix="test" %>
<html>
  <body>
    <test:hello name="Pepe"/>
  </body>
</html>
    
```

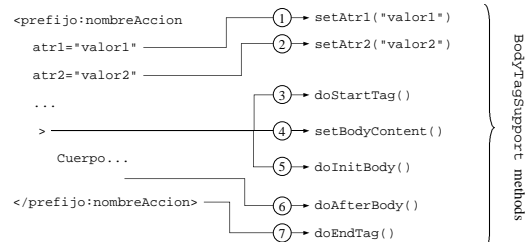
58

### Procesamiento de custom tags



59

### Custom body tags



60