XSL
eXtensible Stylesheet Language

XSLT

XML dokumentuak
transformatzeko lengoaia

Xabier Arregi Iparragirre

Xabier Artola Zubillaga

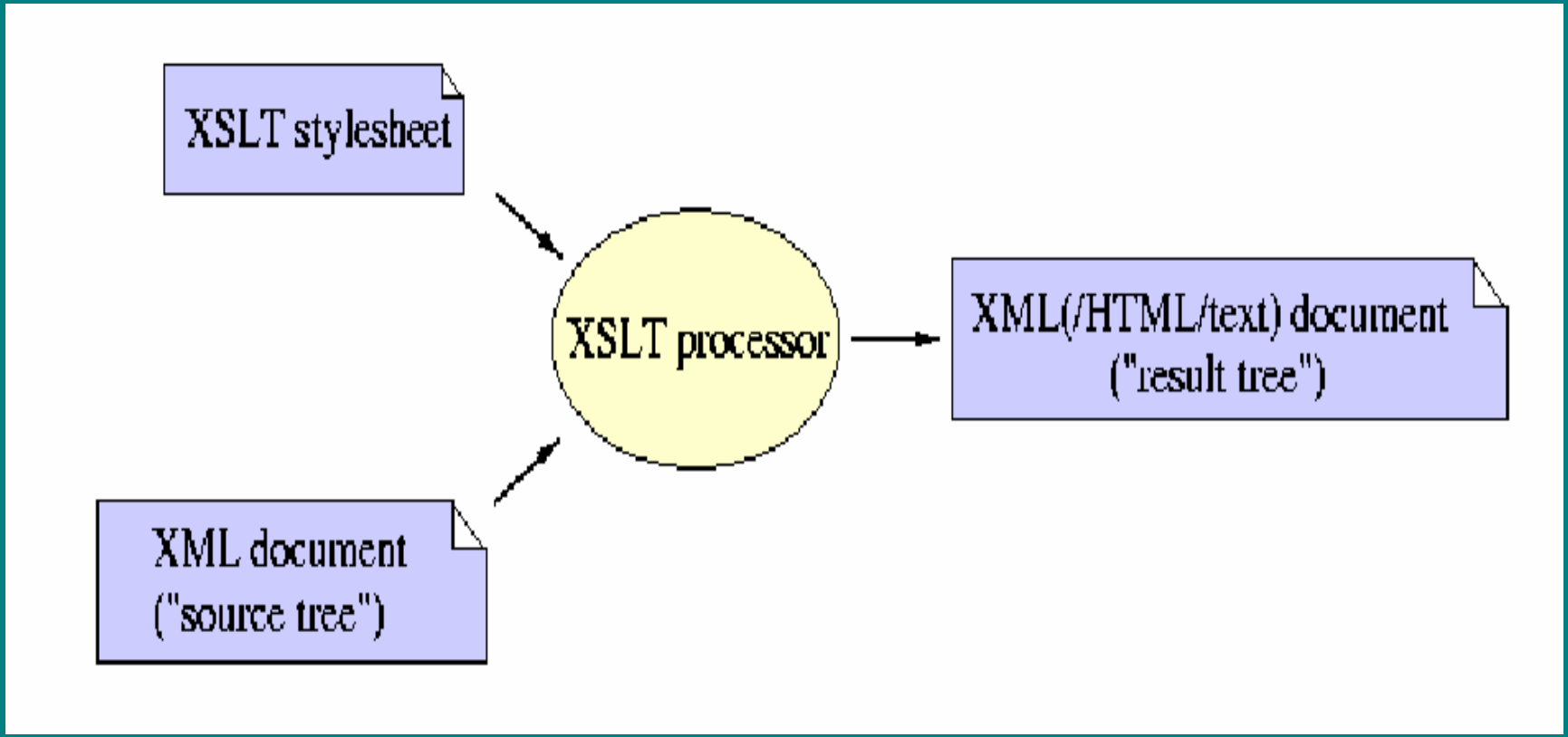
Sarrera

- **XSL** (*e**X**tensible **S**tylesheet **L**anguage*) lengoaiak bi osagai ditu:
  - *XSL Transformations* (**XSLT**)
  - *XSL Formatting Objects* (**XSL-FO**)
- Estilo-orri (*stylesheet*) baten helburua **edukia eta egitura logikoa aurkezpenetik** bereiztea da.
- **XSLT estilo-orri** (edo **script**) bat XML dokumentu bat da. Bere helburua XML dokumentu bat beste zerbait (XML desberdina, HTML, testu soila) bihurtzea da
- **XSL-FO**, berriz, XML lengoaia bat da, HTML eta CSS konbinatuz egiten ahal den baino formatu-emate zehatzago eta behe mailakoagoa adierazteko aukera ematen duena

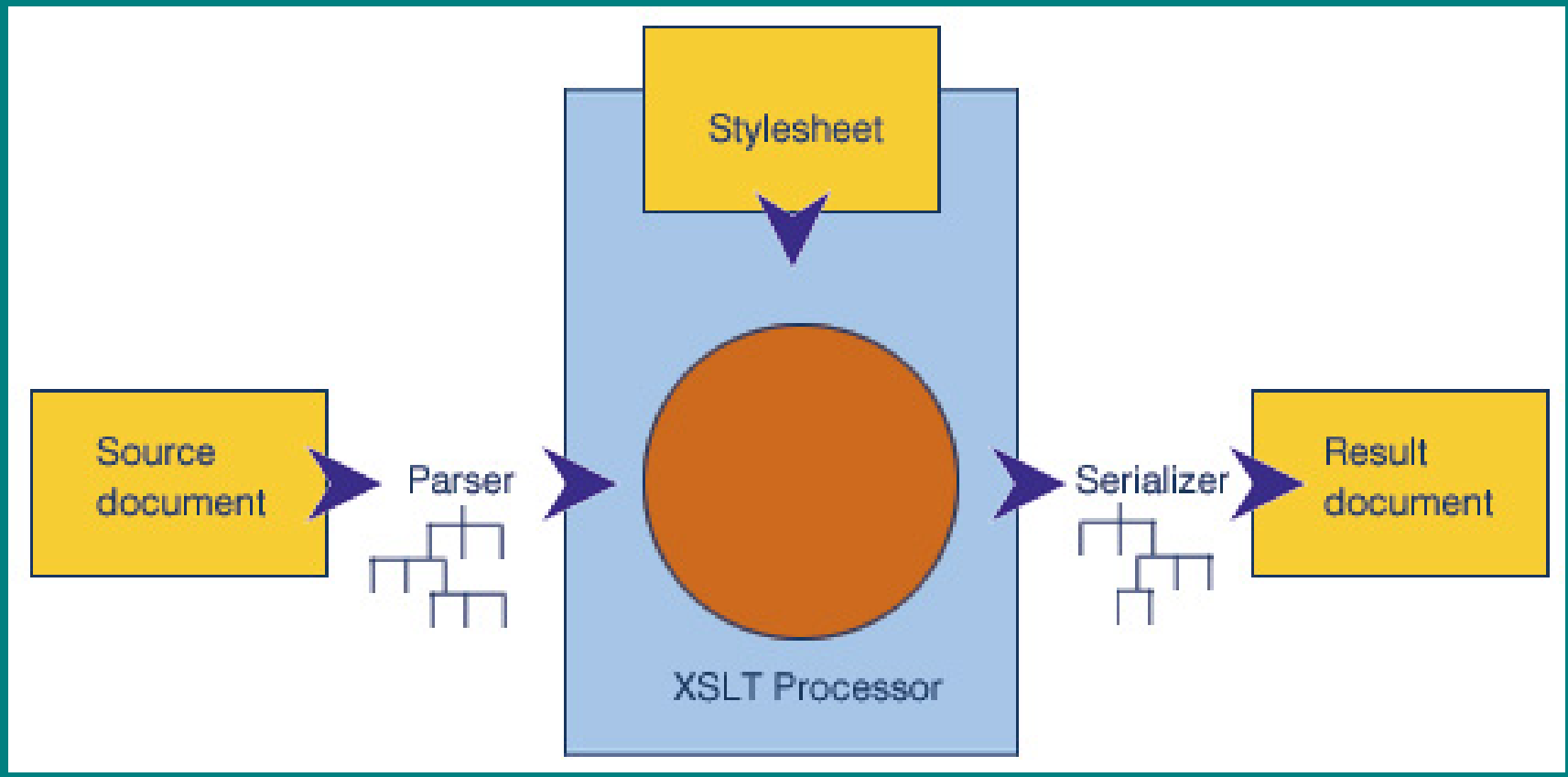
XSL Transformations

XSLT

XSLTren oinarrizko ideia



XSLTren oinarrizko ideia: beste irudi bat



XSLT script-a edo estilo-orria

- Deklaratiboa da
- Patroi-parekatzea eta erregela modukoak (*templates*) erabiltzen ditu transformazioa zehazteko
- CSS estilo-orriek baino askoz adierazpen-ahalmen handiagoa du

Tresnak

- Web-ean, XSLT transformazioa zerbitzarian (Apache Xalan) edo bezeroan (Mozilla, IExplorer...) egin daiteke
  - aurre-prozesu gisa edo *on-the-fly*
- Etorkizunean, nabigatzaileek XSLT eta XSL-FO baino ez dute "ulertu" beharko XML dokumentuak zuzenean bistaratzeko
- Egun, transformazioaren helburua XHTML izan ohi da maiz, hori ezagutzen baitute nabigatzaileek
- XSLT erabat zabaldua eta tresna askotan inplementatuta badago ere, XSL-FO ez oraindik

Informazio bila...

- **http://www.w3.org/Style/XSL/** (W3Cren XSL orri nagusia)
- **http://www.w3.org/TR/xslt** (XSLT 1.0 espezifikazioa)
- **http://www.w3.org/TR/xslt20/** (XSLT 2.0, lantzen ari)
- **http://www.w3.org/TR/xsl** (XSL 1.0, XSL-FO)
- **http://www.w3.org/TR/2006/PR-xsl11-20061006/** (XSL 1.1, XSL-FO)

- **http://www.mulberrytech.com/xsl/xsl-list/** (XSL-List)
- "The XML Bible" (chap. 17: XSLT; chap. 18: XSL-FO)
- **http://www.ibiblio.org/xml/books/bible2/chapters/ch17.html** eta **ch18.html**
- **http://xml.apache.org/xalan-j** (*Xalan*, Java/C++-ez egina)
- **http://saxon.sourceforge.net** (*SAXON*, Java-z egina)
- **http://xmlsoft.org/XSLT** (*libxslt*: C-z egina, Gnome proiektuan)
- **http://www.jclark.com/xml/xt.html** (*XT*, Java-z egina)
- **http://xml.apache.org/fop** (XSL-FO, XMLtik PDFra)

- **Liburu bat:** ***XSLT Programmer's Reference*** (Michael H. Kay)

Prozesatze-eredua

- XSLT script bat hainbat *template* erregelak osatzen dute:

  template-erregela = parekatze-patroia + template-a

- Emaitza lortzeko:
  - iturburu-zuhaitza prozesatzea adabegi erroa prozesatzea da;
  - edozein adabegi prozesatzeko, berriz:
    1. patroi egokiena duen *template* erregela aukeratu
    2. bertako *template*-a instantziatu: emaitza-zatia eratu eta jarraitu errekurtsiboki
  - adabegi-lista bat prozesatzeko, adabegiak sailean-sailean prozesatuz eta emaitzak kateatuz egiten da

Estilo-orri edo script baten egitura

```
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
   ...
   <xsl:template match="patroia">
      template-a (instrukzioak)
   </xsl:template>
   ...
   ...
   ...
</xsl:stylesheet>
```

edo `xsl:transform`

`xsl` izen-espazioa

template erregela

goi-mailako beste elementu batzuk

Estilo-orri edo script baten egitura (II)

- Script batean, elementu batzuk goi-mailakoak dira (`xsl:stylesheet` elementuaren ume):
  - **`<xsl:template...`**
  - **`<xsl:output...`**
  - **`<xsl:variable...`**
  - **`<xsl:param...`**
  - **`<xsl:import...`**
  - **`...`**
- Beste batzuk, aldiz, ez dira goi-mailakoak, eta nahitaez beste batzuen barruan etorri behar dute (*template* baten osagai gisa, maiz):
  - **`<xsl:value-of...`**
  - **`<xsl:copy...`**
  - **`<xsl:for-each...`**
  - **`<xsl:if...`**
  - **`<xsl:import...`**
  - **`...`**

XML dokumentu bat eta bere estilo-orria

- XML dokumentu batean zehazten ahal da, nahi izanez gero, zein estilo-orrirekin prozesatu nahi den:

```
<?xml-stylesheet type="text/xsl" href="itxura.xsl"?>
```

- Egungo nabigatzaileek (Firefox, IExplorer…) XSLT motorra dute.

Adibidea: XML iturburua

```
<card>
   <name>John Doe</name>
   <title>CEO, Widget Inc.</title>
   <email>john.doe@widget.com</email>
   <phone>(202) 555-1414</phone>
   <logo url="widget.gif"/>
</card>
```

Adibidea: estilo-orria, XML dokumentua HTML bihurtzeko

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0"
  xmlns="http://www.w3.org/1999/xhtml">
   <xsl:template match="card">
      <html>
         <head>
            <title>
               <xsl:value-of select="name/text()"/>
            </title>
         </head>
         <body bgcolor="#ffffff">
            <table border="3">
              <tr>
                 <td>
                    <xsl:apply-templates select="name"/><br/>
                    <xsl:apply-templates select="title"/><p/>
                    <tt><xsl:apply-templates select="email"/></tt><br/>
                    <xsl:if test="phone">
                       Phone: <xsl:apply-templates select="phone"/><br/>
                    </xsl:if>
                 </td>
                 <td>
                    <xsl:if test="logo">
                       <img src="{logo/@url}"/>
                    </xsl:if>
                 </td>
              </tr>
            </table>
         </body>
      </html>
   </xsl:template>
   ...
```

Adibidea: estilo-orria (II)

```
   ...
   <xsl:template match="name">
      <xsl:value-of select="text()"/>
   </xsl:template>
   <xsl:template match="title">
      <xsl:value-of select="text()"/>
   </xsl:template>
   <xsl:template match="email">
      <xsl:value-of select="text()"/>
   </xsl:template>
   <xsl:template match="phone">
      <xsl:value-of select="text()"/>
   </xsl:template>
</xsl:stylesheet>
```

Emaitza: HTML dokumentua: iturburua eta nabigatzailean ikusita

```
<?xml version="1.0" encoding="UTF-8"?>
<html
  xmlns="http://www.w3.org/1999/xhtml">
  <head><title>John Doe</title></head>
  <body bgcolor="#ffffff">
    <table border="3">
      <tr>
        <td>John Doe<br/>
          CEO, Widget Inc.
          <p/>
          <tt>john.doe@widget.com</tt><br/>
          Phone: (202) 555-1414<br/>
        </td>
        <td>
          <img src="widget.gif"/>
        </td>
      </tr>
    </table>
  </body>
</html>
```

xsl:value-of

- Instrukzio honek bere `select` atributuko XPath espresioa ebaluatzen du string gisa, eta emaitza irteerara kopiatzen du.
- Funtsezko instrukzioa da edozein script-etan.
- Adibideak:

```
<xsl:value-of select="text()"/>
<xsl:value-of select="."/>
<xsl:value-of select="name/text()"/>
<xsl:value-of select="name"/>
```

CSSren bidez ere adieraz daiteke nola bistaratu dokumentu bat, baina…

- XSLT baino mugatuagoa da
- Informazioa ezin da beste era batera antolatu
- Ezin da inolako konputaziorik gauzatu iturburu-dokumentuaren gainean
- Helburu-dokumentua ezin da XML izan

- Hala ere, gaur egun, eta XSL-FOren erabilera hedatzen ez den bitartean, **XSLT eta CSS estilo-orrien konbinazioa** erabiltzen da erruz **XML dokumentuak web-ean bistaratzeko**

Parekatze-patroiak: XPath

- Patroiak: emaitzatzat adabegi-multzoak hautatzen dituzten XPath espresioak
- Adabegi bat patroi batekin parekatuko da, baldin eta adabegi hori patroia testuinguru jakin batean ebaluatzearen emaitzaren barne badago
- Gogoratu:

  patroia: kokapen-bidea | ... | kokapen-bidea

  kokapen-bidea: /urratsa/ ... // ... /urratsa

  urratsa: ardatza adabegi-egiaztapena predikatua

  Oharra: parekatze-patroion ardatzetan, `child` eta `attribute` ardatzak baino ez dira onartuko

Template-ak

- *Template* erregeletan hainbat motatako *template*-ak adierazten ahal dira:
  - Emaitza-zati literalak
  - Prozesatze errekurtsiboa
  - Emaitza-zati konputatuak
  - Baldintzazko prozesatzea
  - Ordenazioa eta zenbakitzea
  - Aldagaiak eta parametroak
  - Gakoak

Emaitza-zati literalak

- Testu-zati bat (karaktere-kateak)
- XSL izen-espaziokoa ez den XML elementu bat
- **`<xsl:text ...> ... </...>`** (testu soila bezala, baina zuriuneekin eta karaktereen kode-aldaketarako (*escape*) kontrolarekin
- **`<xsl:comment> ... </...>`** (XMLzko iruzkin bat: **`<!--...-->`**)
- Literalak estilo-orriaren parte direnez, eta estilo-orria XML dokumentu bat, orduan ongi eratutako dokumentuak baino ez dira sortuko
- Adibidea:

```
<xsl:template match="...">
   Testu hau bere horretan idatziko da
   emaitzan, template erregela hau instantziatutakoan
</xsl:template>
```

Prozesatze errekurtsiboa

- Prozesatze errekurtsiboa asko erabiltzen da XSLTn. Hona hemen horretarako zenbait XSL elementu:
- **`<xsl:apply-templates select="`***adabegi-multzoko espr.***`" .../>`**

  Patroi-parekatzea eta *template* instantziazioa aplikatu aukeratutako adabegietan (besterik ezean, hau da, `select` atributua ez bada zehazten: ume guztietan).
- **`<xsl:call-template name="`...`"/>`**

  Template-a deitu izenaz (`xsl:template` elementuak `name="..."` atributua izango du, noski): azpiprogramak!!!
- **`<xsl:for-each select="`***adabegi-multzoko espr.***`">`***template-a***`</...>`**

  Instantziatu barruko *template*-a adabegi-multzoko adabegi bakoitzeko (besterik ezean, dokumentu-ordenaren arabera).

  Oharra: Elementu honek prozesatze iteratiboa bultzatzen du, jakina, eta batzuetan oso egoki eta erosoa da (dokumentu bateko maila bereko elementuak, senideak, banan-banan eta sailean tratatzeko, esate baterako).

Prozesatze errekurtsiboa (II)

- **<xsl:copy>***template-a***</...>**

  Kopiatu uneko adabegia emaitzan, eta aplikatu barruko *template*-a (azaleko kopia).
- **<xsl:copy-of select="**...**"/>**

  Kopiatu aukeratutako adabegiak emaitzan (sakoneko kopia: ondorengoak ere kopiatu egiten ditu).
- `select` atributuak XPath espresio bat hartzen du baliotzat; espresio hori uneko testuinguruan ebaluatzen da.
- Adibidea:

```
<xsl:template match="article">
   <h1><xsl:apply-templates select="title"/></h1>
</xsl:template>
```

Adibideak: CDen katalogoa berriro ere!

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
   <cd country="USA">
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <price>10.90</price>
   </cd>
   <cd country="UK">
      <title>Hide your heart</title>
      <artist>Bonnie Tyler</artist>
      <price>9.90</price>
   </cd>
   <cd country="USA">
      <title>Greatest Hits</title>
      <artist>Dolly Parton</artist>
      <price>9.90</price>
   </cd>
</catalog>
```

1. adibidea

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
    <xsl:output method="xml" version="1.0"
       encoding="ISO-8859-1" indent="yes"/>
    <xsl:template match="/">
        <xsl:element name="tituluak">
            <xsl:for-each select="catalog/cd">
                <titulua>
                    <xsl:value-of select="title"/>
                </titulua>
            </xsl:for-each>
        </xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

1. adibidea: emaitza

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tituluak>
   <titulua>Empire Burlesque</titulua>
   <titulua>Hide your heart</titulua>
   <titulua>Greatest Hits</titulua>
</tituluak>
```

1. adibidea: galderak

- Zer gertatuko da beheko aldaketa hauek egiten baditugu script-ean?
- Bi hauetako zein da aurreko soluzioko `xsl:for-each` elementuaren baliokidea?

```
<xsl:for-each select="/catalog/cd/title">
   <titulua>
      <xsl:value-of select="title"/>
   </titulua>
</xsl:for-each>

<xsl:for-each select="/catalog/cd/title">
   <titulua>
      <xsl:value-of select="."/>
   </titulua>
</xsl:for-each>
```

2. adibidea: zer egingo du script honek?

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">
   <xsl:output method="xml" version="1.0"
     encoding="ISO-8859-1"/>
   <xsl:template match="*|@*|comment()|
   processing-instruction()|text()">
      <xsl:copy>
         <xsl:apply-templates
         select="*|@*|comment()|
         processing-instruction()|text()"/>
      </xsl:copy>
   </xsl:template>
</xsl:stylesheet>
```

2. adibidea: are errazago, `xsl:copy-of` erabiliz

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   version="1.0">
    <xsl:output method="xml" version="1.0" encoding="ISO-
   8859-1"/>
    <xsl:template match="/">
       <xsl:copy-of select="."/>
       <!-- edo <xsl:copy-of select="self::node()"/>-->
       <!-- edo <xsl:copy-of select="catalog"/>-->
       <!-- edo <xsl:copy-of select="node()"/>-->
    </xsl:template>
</xsl:stylesheet>
```

xsl:apply-templates

- XSLTren prozesatze errekurtsiboaren muina
- Ikas dezagun hori ondo!:

..\XSLT_eskulana\XSLT_eskulana.ppt

Emaitza-zati konputatuak

- Emaitza-zatiak eratzen ahal dira, XPath espresioak erabiliz:
  - **`<xsl:element name="..." namespace="..."> ... </xsl:element>`**

    elementu bat sortzen du, emandako izen, atributu eta edukiekin
  - **`<xsl:attribute name="..." namespace="..."> ... </xsl:attribute>`**

    atributu bat sortzen du (`xsl:element`-en barruan behar du)
  - **`<xsl:value-of select="..."/>`**

    karaktere-kate (testua) edo atributu-balio bat eratzen du (string bihurtutako espresio bat)
  - **`<xsl:processing-instruction name="..."> ... </xsl:processing-instruction>`**

    prozesatze-agindu bat eratzen du

Emaitza-zati konputatuak (II)

- Atributuetan *{espresioa}* idatz daiteke: instantziatzean ebaluatu eta string bihurtuko den XPath espresioa.
- Adibidea:

```
<xsl:template match="atala">
   <xsl:element name="kap{@maila}">
      <xsl:attribute name="mota">
         <xsl:value-of select="mota"/>
      </xsl:attribute>
   </xsl:element>
</xsl:template>
```

```
<atala maila="3">
  <mota>sarrera</mota>
</atala>
```

```
<kap3 mota="sarrera"/>
```

Baldintzazko prozesatzea

- **<xsl:if test="***espresioa***"> ...**
  </xsl:if>
  aplikatu *template*-a, espresioa *true* bada
- **<xsl:choose>**
  <xsl:when test="*espresioa***"> ...**
  </xsl:when>
  <xsl:when test="*espresioa***"> ...**
  </xsl:when>
  ...
  <xsl:otherwise> ...
  </xsl:otherwise>
  </xsl:choose>
  baldintzak sailean aztertu, eta aplikatu *true* den aurrenekoari dagokion *template*-a

Baldintzazko prozesatzea: adibidea

```
<xsl:template match="atala">
   <xsl:if test="@maila=3">
      <xsl:element name="azpiatala">
         <xsl:attribute name="mota">
            <xsl:value-of select="mota"/>
         </xsl:attribute>
      </xsl:element>
   </xsl:if>
</xsl:template>
```

```
<atala maila="3">
  <mota>sarrera</mota>
</atala>
```

```
<azpiatala mota="sarrera"/>
```

Aldagaiak eta parametroak

- Konputazioen emaitzak berrerabili ahal izateko
- *Template*-ak zein script osoak parametrizatzeko
- XPath balioak eduki ditzakete (*string, number, boolean, node-set*) edo emaitza-zuhaitz zati osoak (*result tree fragment*)
- Erazagupeneko balioa ezin da gero aldatu!
- **Globalak** (script osoa esparru) zein **lokalak** (*template* baten barrukoak)

Aldagaiak eta parametroak (II)

- Erazagupena:
  - **`<xsl:variable name="..." select="`*espresioa*`"/>`**
    espresioa (XPath): aldagaiaren hasierako balioa
  - **`<xsl:variable name="...">`*template-a*`</..>`**
    template-a instantziatzen da, eta emaitzatzat lortutako zuhaitz zatia da aldagaiaren hasierako balioa
- **`xsl:param`**: aldagaien antzera erazagutzen dira, baina hasierako balioak baino besterik ezeko balioak zehazten dira.

Aldagaiak eta parametroak (III)

- Erabilera:
  - **$*aldagai-izena***: XPath balioa, espresioetan-eta erabiltzen ahal dena
  - **`xsl:with-param`**: parametroak `xsl:call-template`-ri eta `xsl:apply-templates`-i pasatzeko erabiltzen da

Aldagaien erabilera: adibidea

```
<xsl:template match="edozer">
   <xsl:variable name="X" select="0"/>
   <xsl:variable name="Y">
      <etiketa>
         <xsl:value-of select="@atrib"/>
      </etiketa>
   </xsl:variable>
   <lehena>
      <xsl:value-of select="$X"/>
      <xsl:copy-of select="$Y"/>
   </lehena>
   <bigarrena>
      <xsl:value-of select="$X"/>
      <xsl:copy-of select="$Y"/>
   </bigarrena>
</xsl:template>
```

Hasierako balioa: espresioa

Hasierako balioa: template oso bat

```
<edozer atrib="???"/>
```

```
<lehena>0
  <etiketa>???</etiketa>
</lehena>
<bigarrena>0
  <etiketa>???</etiketa>
</bigarrena>
```

Parametroen erabilera: adibidea

```
<xsl:template match="/dok">
   <dok>
      <xsl:for-each select="letraMoldea">
         <xsl:choose>
            <xsl:when test="@mota='1'">
               <xsl:apply-templates select=".">
                  <xsl:with-param name="kolorea" select="'blue'"/>
               </xsl:apply-templates>
            </xsl:when>
            <xsl:otherwise>
               <xsl:apply-templates select="."/>
            </xsl:otherwise>
         </xsl:choose>
      </xsl:for-each>
   </dok>
</xsl:template>
<xsl:template match="letraMoldea">
   <xsl:param name="kolorea" select="'black'"/>
   <font color="{$kolorea}">
      <!--<xsl:apply-templates/> ez da behar-->
   </font>
</xsl:template>
```

Parametro erreala deian

Besterik ezeko balioa

```
<dok>
   <letraMoldea mota="1"/>
   <letraMoldea mota="2"/>
   <letraMoldea mota="1"/>
</dok>
```

```
<dok>
  <font color="blue"/>
  <font color="black"/>
  <font color="blue"/>
</dok>
```

Parametroen erabilera: adibidea (II)

- Beste modu bat gauza bera lortzeko:

```
<xsl:template match="/dok">
   <dok>
      <xsl:apply-templates/>
   </dok>
</xsl:template>
<xsl:template match="letraMoldea[@mota='1']">
   <font color="blue">
   </font>
</xsl:template>
<xsl:template match="letraMoldea[@mota='2']">
   <font color="black">
   </font>
</xsl:template>
```

Aldagaiak eta parametroak (II)

- Tamalez, aldagaietan gordetako emaitza-zuhaitz zatiak ezin dira gero erabili patroi-parekatzean edo *template*-en instantziazioan (XSLT 1.0-n behintzat)
- Goi-mailako parametroei (globalei) balioa pasatzeko modua inplementazioaren araberakoa da: komando-lerrotik, APIaren bidez, interfaze grafikotik…

Gakoak (*keys*)

- Gakoa: *(adabegia, deitura, balioa)* hirukotea, deitura-balio bikote bat adabegi bati egokitzeko.

  <xsl:key match="*patroia***" name="...**
  use="*adabegi-multzoko espresioa***"/>**

  xsl:key elementuak gako multzoak erazagutzen ditu: **patroiarekin parekatzen den adabegi bakoitzeko** eta **use espresioan dagoen adabegi bakoitzeko bat**.

  match: zein adabegiri aplikatzen zaion gakoa

  use: gakoaren balioa(k) zehazten ditu
- Bilaketa-espresioak erraztu egiten ditu.
- Inplementazio gehienetan, eraginkorragoa da (indizeak eratzen baitira gakootan oinarrituz).
- XSLT funtzio hauekin batera erabili ohi da:

  key(*deitura, balioa***)**

  emandako gako-deitura eta -balioa duten adabegiak itzultzen ditu

  generate-id(*adabegia***)**

  emandako adabegia unibokoki identifikatzen duen string bat itzultzen du

Gakoen erabilera : adibidea

```
<catalog>
   <authors>
      <author id="a1">
         <name>Bob Dylan</name>
         <data>Bob Dylan-en datuak</data>
      </author>
      <author id="a2">
         <name>Bonnie Tyler</name>
         <data>Bonnie Tyler-en datuak</data>
      </author>
      ...
   </authors>
   <cds>
      <cd country="USA">
         <title>Empire Burlesque</title>
         <artist><ref author="a1"/></artist>
         <price>10.90</price>
      </cd>
      ...
      <cd country="USA">
         <title>Dylan-en eta beste baten disko bat</title>
         <artist>
            <ref author="a1"/>
            <ref author="a4"/>
         </artist>
         <price>7.90</price>
      </cd>
   </cds>
</catalog>
```

Gakoen erabilera : adibidea (II)

```
<!--Gakoak definitu egileen id-aren arabera -->
<xsl:key name="egileGakoak" match="author" use="@id"/>
...

<!--Egileak prozesatu, aingurak sortuz-->
<xsl:template match="author">
   <h2>
      <a name="{generate-id()}">
         <!—zenbakitu-->
         <xsl:number/>

         <!--zuriunea utzi zenbakia eta izena bereizteko-->
         <xsl:text> </xsl:text>

         <xsl:apply-templates select="name"/>
      </a>
   </h2>
   <xsl:apply-templates select="data"/>
</xsl:template>
...
```

Gakoen erabilera : adibidea (III)

```
...
<!--CDak prozesatu, ref elementuetan hiperestekak sortuz-->
<xsl:template match="cd">
   <h2>
      <xsl:apply-templates/>
   </h2>
   <xsl:apply-templates select="data"/>
</xsl:template>
<xsl:template match="ref[@author]">
   <xsl:variable name="autorea"
                 select="key('egileGakoak',@author)"/>
   <a href="#{generate-id($autorea)}">
      <xsl:text>Egilea: </xsl:text>
      <xsl:value-of select="$autorea/name"/><br/>
   </a>
</xsl:template>
```

Gakoak eta DTDetako *id*-ak

- Gakoak XSLT script-ean erazagutzen dira (ez DTDan)
- Gakoak ez dira definitu behar nahitaez atributuen gainean beti
- Adabegi bati hainbat gako egokitu dakizkioke
- Gakoek ez dute zertan unibokoak izan

Besterik?

- XSLT askoz konplexuagoa (eta aberatsagoa) da!
- Ikusi ez ditugun zenbait gauza:
  - berezko edo besterik ezeko (*built-in*) *template* erregelak: zer gertatzen da prozesatzen ari den adabegiarekin egokitzen den *template* erregelarik ez dagoenean?
  - lehentasun-kontuak (`priority`): nola aukeratzen da *template* erregela bat patroi bat baino gehiago parekatzen direnean?
  - irteera-moduak: `xml`, `html`, `text`
  - zuriuneen maneiua (`strip-space, preserve-space`) eta eskape kontuak irteera eratzerakoan (`disable-output-escaping`)
  - XSLTk baditu, XPath-ekoez gain, beste funtzio batzuk: `document()`, `key()`, `generate-id()`, `format-number()`, `current()`...
  - script-en modulartasuna: `import`, `include`

Berezko edo besterik ezeko (*built-in*) *template* erregelak

- Adabegi erroa: **`<xsl:apply-templates/>`** deitu, umeak prozesatzeko
- Elementu-adabegia: **`<xsl:apply-templates/>`** deitu, umeak prozesatzeko
- Atributu-adabegia: atributuaren balioa testu gisa kopiatu
- Testu-adabegia: balioa (karaktere-katea) irteerara kopiatu
- Prozesatze-aginduak, iruzkinak eta izen-espazioko adabegiak: ez ezer egin

Berezko edo besterik ezeko (*built-in*) *template* erregelak (II)

```
<xsl:template match="/ | *">
  <xsl:apply-templates/>
</xsl:template>
<xsl:template match="text()">
  <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="@">
  <xsl:value-of select="."/>
</xsl:template>
<xsl:template match="comment()"/>
<xsl:template match="processing-instruction()"/>
```

Besterik ezeko *template* erregelak: adibidea

- Zer egingo du, orduan, XSLT script hutsak CDen dokumentuarekin?

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
</xsl:stylesheet>
```

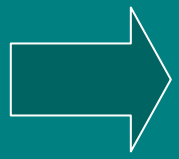
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<catalog>
   <cd country="USA">
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <price>10.90</price>
   </cd>
   ...
</catalog>
```

Besterik ezeko *template* erregelak: adibidea (II)

- Hauxe!:

```
<?xml version="1.0" encoding="UTF-8"?>

      Empire Burlesque
      Bob Dylan
      10.90
```



```
      Hide your heart
      Bonnie Tyler
      9.90


      Greatest Hits
      Dolly Parton
      9.90
```

XSL Formatting Objects

XSL-FO

XSL Formatting Objects (XSL-FO)

- XSL-FO erabiliz **dokumentuaren formatua xehetasun osoz zehazten ahal** da.
- **XML dokumentua + XSL-FO estilo-orria:** *LaTex* moduko zerbait, baina XMLn oinarritua.
- Gogoratu, HTML edo XHTML erabiltzen denean, itxura zehatza ez dugula guk erabakitzen, nabigatzaileak baizik!
- **Ongi definitua** (W3C), **baina inplementazio gutxi** oraindik:
  - *Apache FOP* (FO Processor, http://xml.apache.org/fop/); *PDF*, *PS* eta *text* irteera-metodoak ditu inplementatuta

XSL Formatting Objects: adibide xume bat

```
<?xml version="1.0" encoding="UTF-8"?>
<dok>
   <titulua>Proba</titulua>
   <gorputza>
      Hauxe proba bat besterik ez da
   </gorputza>
</dok>
```

XSL Formatting Objects: adibide xume bat (II)

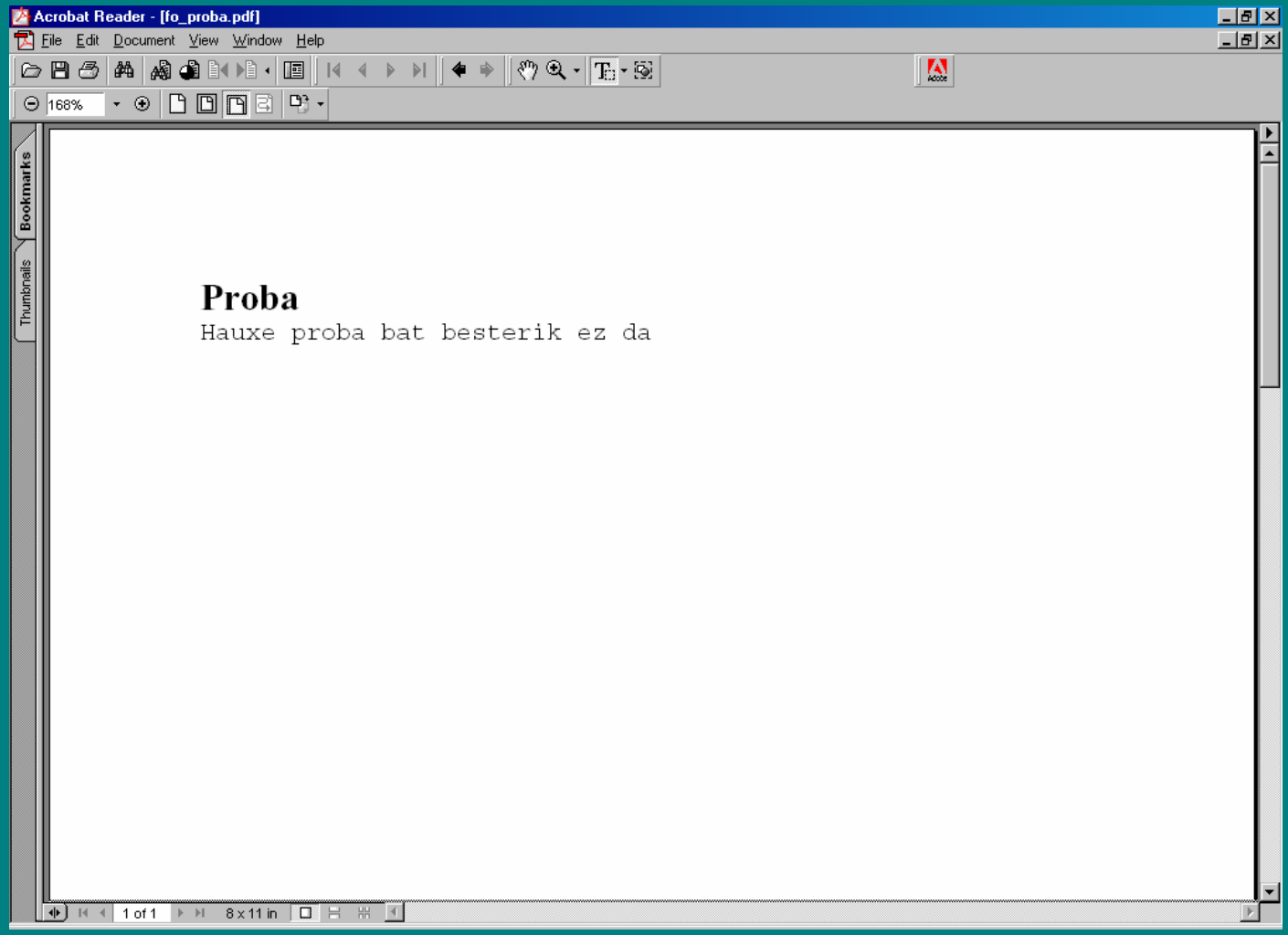
```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  version="1.0">
   <xsl:template match="/dok">
      <fo:root>
         <fo:layout-master-set>
            <fo:simple-page-master master-name="orria">
               <fo:region-body margin="1in"/>
            </fo:simple-page-master>
         </fo:layout-master-set>
         <fo:page-sequence master-reference="orria">
            <fo:flow flow-name="xsl-region-body">
               <fo:block font-family="Times" font-size="18pt">
                  <fo:inline font-weight="bold">
                     <xsl:value-of select="titulua"/>
                  </fo:inline>
               </fo:block>
               ...
```

XSL Formatting Objects: adibide xume bat (III)

```
            ...
            <fo:block font-family="Courier" font-size="12pt">
              <fo:inline>
                <xsl:value-of select="gorputza"/>
              </fo:inline>
            </fo:block>
          </fo:flow>
        </fo:page-sequence>
      </fo:root>
    </xsl:template>
</xsl:stylesheet>
```

- Orriaren egitura `fo:layout-master-set` delakoen bidez zehazten da
- Orriak orri-sekuentziatan (`fo:page-sequence`) antolatzen dira
- Edukia orrialdeei `fo:flow` objektuen bidez egokitzen zaie
- Edukia bera bloketan (`fo:block`) eratzen da

XSL Formatting Objects: adibidearen emaitza



Adibidea: XSLTren erabilera web aplikazio batean

Diccionario Básico del Español

Centro de Lingüística Aplicada
(Santiago de Cuba, Kuba)

IXA taldea (UPV/EHU)

DBE: eskolarako hiztegia

- Bertsio elektronikoa, XMLn oinarritua
  - CD bertsioa (*Ada Web Server*)
  - Web bertsioa (AWS, *Apache*-ren gainean)
- Bi bertsiootan, arkitektura bera:
  - Bezeroa (interfazea): web nabigatzailea
  - Zerbitzaria: hiztegia bera eta indizeak (XML dokumentuak)
- CD bertsioa kalean (eskoletan!), eta web bertsioa ere linean dago dagoeneko.

Hiztegia XMLz: letra bakoitzeko, sarreren dokumentua eta indize-dokumentua

```
...
<entry id="g_d0e124">
  <form>
    <orth>gafas</orth>
    <syll>ga|fas</syll>
  </form>
  <gramGrp>
    <pos>sf. pl.</pos>
  </gramGrp>
  <sense n="1">
    <def>Objeto formado por dos
cristales o lentes, colocados en una
armadura. Se apoya en la nariz y se
sujeta en las orejas por medio de
patas. Las personas usan gafas cuando
no ven bien o para protegerse del sol.
    </def>
    <eg>
      <q>Ayer me compré unas <oRef/>
de cristales oscuros.</q>
    </eg>
    <xr>
      <lbl>Sin.</lbl>
      <ref>espejuelos</ref>
      <ref>lentes</ref>
    </xr>
  </sense>
  <form type="infl">
    <orth>gafitas</orth>
    <gram>(dim.)</gram>
  </form>
</entry>
...
```

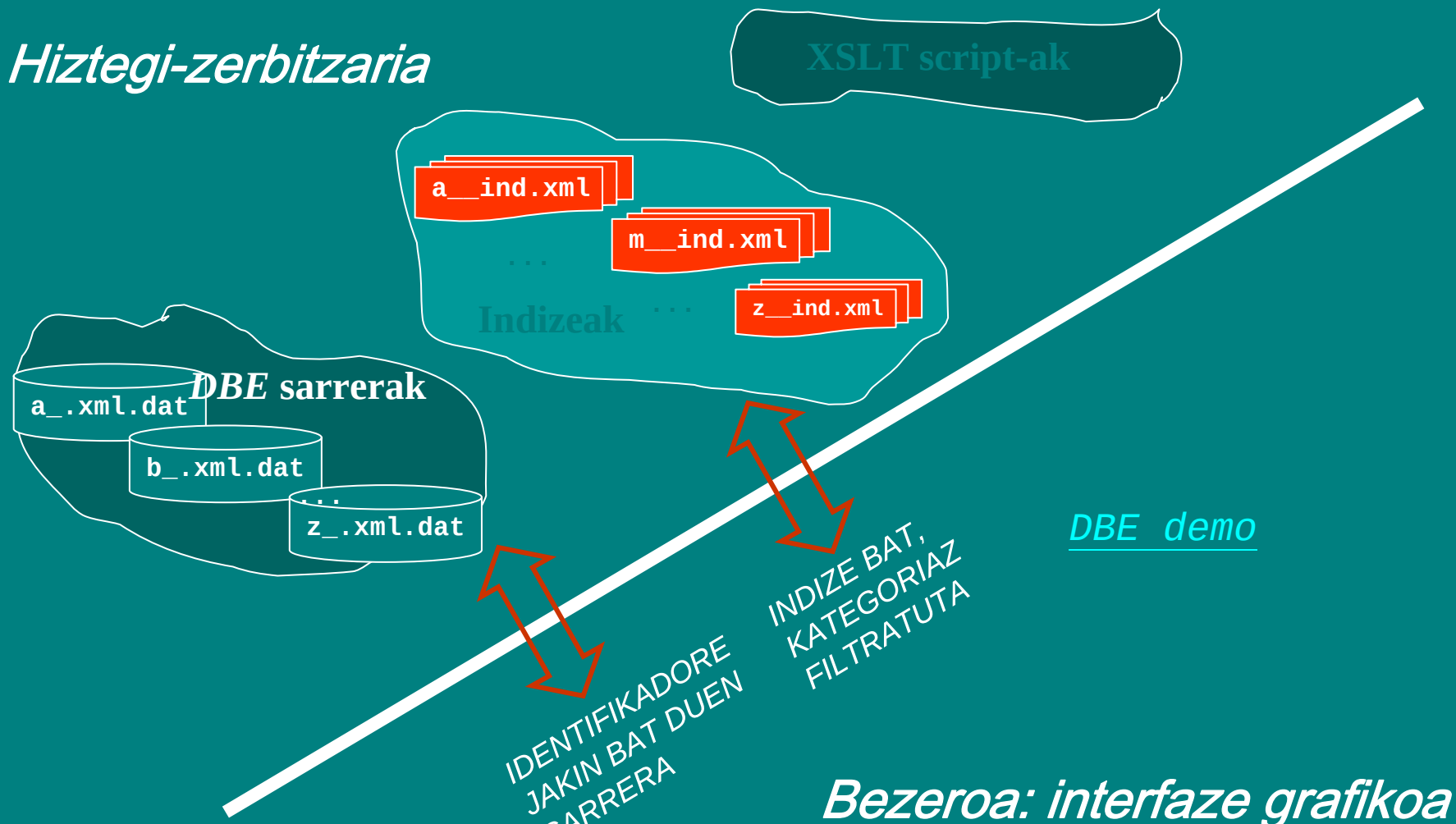
```
<entryIndex>
  <entry id="g_d0e85">
    <orth>g</orth>
    <pos>sf.</pos>
  </entry>
  <entry id="g_d0e124">
    <orth>gafas</orth>
    <pos>sf. pl.</pos>
  </entry>
  <entry id="g_d0e176">
    <orth>gajo</orth>
    <pos>sm.</pos>
  </entry>
  <entry id="g_d0e254">
    <orth>galán</orth>
    <pos>sm.</pos>
  </entry>
  <entry id="g_d0e308">
    <orth>galardón</orth>
    <pos>sm.</pos>
  </entry>
  ...
</entryIndex>
```

XSLTren erabilera

- Zerbitzarian prozesatu, nabigatzailearen ahalmenaz ezin baikaitezke ziur egon: bezeroari beti XHTML bidaltzen zaio
- *Libxslt* XSLT liburutegia erabiltzen da hiztegiaren CD bertsioan: C-z inplementatua da, eta Ada web zerbitzaritik dei egiten zaio.

- Hiru XSLT script (sarrerak lortzeko, eta sarrerak eta indizeak bistaratzeko):
  - *sarrera_lortu.xslt*
    - identifikadore bat emanda, identifikadore hori duen sarrera itzultzen du XML dokumentu batean
  - *sarrera_bistaratu.xsl*
    - *sarrera_lortu.xsl*-ren irteera XHTML bihurtzen du, nabigatzailean bistaratzeko (CSS estilo-orriak ere erabiltzen dira dokumentuari formatua emateko)
  - *indizea_bistaratu.xsl*
    - kategoria zerrenda bat emanda, kategoria hori duten sarreren zerrenda bat itzultzen du HTML dokumentu batean, non sarrera bakoitza HTMLko `<select>` baten barruko `<option>` gisa kodetzen baita
- Beste XSL estilo-orri bat aditz-paradigmak bistaratzeko.

Aplikazioaren arkitektura, pixka bat sinplifikatuta



sarrera_lortu.xsl: emandako identifikadorea duen sarrera eskuratu letra-dokumentutik

```
<xsl:output method="xml" encoding="utf-8"/>
<xsl:strip-space elements="entry"/>

<xsl:param name="idEntrada"/> <!-- parametro globala -->

<xsl:key name="identEntrada"
         match="entry | superentry"
         use="@id"/>

<xsl:template match="/">
   <body> <!-- XML irteeraren dokumentu-elementua -->
      <xsl:copy-of select="key ('identEntrada', $idEntrada)"/>
   </body>
</xsl:template>
```

indizea_bistaratu.xsl: emandako kategoria duten sarreren zerrenda eratu

```
<xsl:output method="html" omit-xml-declaration="yes" encoding="ISO-8859-1"
   doctype-system="HTML" media-type="text/html"/>

<xsl:param name="seleccionPOS"/> <!-- param. globala: "adj.|vtr." -->
<xsl:template match="/entryIndex">
  <html>
    <head>...</head>
    <body background="images/fig12.gif">
      <form method="post" enctype="application/x-www-form-urlencoded"
            name="selectEntry">
        <select class="entries" onclick="javascript:setEntry('entries')"
                name="entries" size="20">
          <xsl:choose>
            <xsl:when test="$seleccionPOS!=''"> <!-- kateg.-lista ez da hutsa -->
              <xsl:variable name="indicePOS">
                <xsl:call-template name="seleccionarEntradasPOS">
                  <xsl:with-param name="lista"
                                  select="concat ($seleccionPOS, '|')"/>
                </xsl:call-template>
              </xsl:variable>
              <xsl:for-each select="exsl:node-set($indicePOS)/entry">
                <xsl:sort select="orth" lang="es"/>
                <xsl:call-template name="escribirOption"/>
              </xsl:for-each>
            </xsl:when>
            ...
```

indizea_bistaratu.xsl: emandako kategoria duten sarreren zerrenda eratu (II)

```
            ...
            <xsl:otherwise> <!-- kateg.-lista hutsa da: kategoria
                                 guztietakoak hautatu behar dira-->
              <xsl:for-each select="entry">
                <xsl:call-template name="escribirOption"/>
              </xsl:for-each>
            </xsl:otherwise>
          </xsl:choose>
        </select>
      </form>
      <script type="text/javascript">
        selectFirst (document.forms["selectEntry"].entries);
      </script>
    </body>
  </html>
</xsl:template>
```

indizea_bistaratu.xsl: emandako kategoria duten sarreren zerrenda eratu (III)

```
<xsl:template name="escribirOption" match="entry">
  <!-- para evitar elementos repetidos en el caso de las
       superentries y en entries con más de una POS-->
  <xsl:if test="not(./orth = preceding-sibling::*/orth)">
    <!--El id de la entrada se pasa al documento HTML en el
        atributo value del elemento option-->
    <option value="{./@id}">
      <xsl:value-of select="orth"/>
    </option><br/>
  </xsl:if>
</xsl:template>
```

Script honek `<option>` bat idazten du irteeran

indizea_bistaratu.xsl: emandako kategoria duten sarreren zerrenda eratu (IV)

```
<xsl:template name="seleccionarEntradasPOS">
  <!--template recursivo para procesar la lista de POS contenida en el
      parámetro string seleccionPOS (lista)-->
  <xsl:param name="lista"/>
  <xsl:variable name="listaPOS" select="$lista"/>
  <xsl:choose>
    <!--caso trivial: $listaPOS='', el string restante ha quedado
        vacío-->
    <xsl:when test="$listaPOS!=''">
      <xsl:variable name="primera"
                    select="substring-before($listaPOS, '|')"/>
      <xsl:variable name="resto"
                    select="substring-after($listaPOS, '|')"/>
      <xsl:for-each select="entry[pos][contains(., $primera)]">
        <xsl:copy-of select="."/>
      </xsl:for-each>
      <!—llamada recursiva: -->
      <xsl:call-template name="seleccionarEntradasPOS">
        <xsl:with-param name="lista" select="$resto"/>
      </xsl:call-template>
    </xsl:when>
  </xsl:choose>
</xsl:template>
```

Script errekurtsiboak maiz erabili behar dira, aldagaien balioa aldatu ezin delako!: honako honek kategoria-zerrenda prozesatzen du

sarrera_bistaratu.xsl: XMLz emandako sarrera XHTML bihurtu (formatua eman, hiperestekak ezarri)

- Script luzeena: ~1100 kode-lerro
- CSS estilo-orri batekin konbinatua
- Hona hemen adibideen bistaratzeari dagokion zati bat:

```
...
<xsl:template match="eg">
  <tr>
    <td class="c1"></td>
    <td class="c2">
      <xsl:apply-templates select="q"/> 
      <xsl:apply-templates select="lbl"/>
    </td>
  </tr>
</xsl:template>

<xsl:template match="q">
  <xsl:if test="preceding-sibling::q">
    <br/>
  </xsl:if>
  <i>
    <xsl:apply-templates select="emph|oRef|oVar|text()"/>
  </i>
  <xsl:apply-templates select="note"/>
</xsl:template>
...
```

Aditz-paradigmak: aditz sarreren `itype` elementutik atzitzen dira

```
<?xml version="1.0" encoding="utf-8"?>
<!--Generado por vb110_compose.xslt; 2005-03-09T13:09:44+01:00-->

<conjverbales>
   <modelo id="1">
      <modo id="infinitivo">estar</modo>
      <modo id="infinitivocomp">haber estado</modo>
      <modo id="gerundio">estando</modo>
      <modo id="gerundiocomp">habiendo estado</modo>
      <modo id="participio">estado</modo>
      <modo id="indicativo">
         <tiempo id="pres">
            <conj p="1s" esp="5">estoy</conj>
            <conj p="2s" esp="4">estás</conj>
            <conj p="2s'" esp="4">está</conj>
            <conj p="3s" esp="4">está</conj>
            <conj p="1p">estamos</conj>
            <conj p="2p">estáis</conj>
            <conj p="3p" esp="4">están</conj>
         </tiempo>
         <tiempo id="antepres">
            <conj p="1s" esp="2">he estado</conj>
            <conj p="2s" esp="2">has estado</conj>
            <conj p="2s'" esp="2">ha estado</conj>
            <conj p="3s" esp="2">ha estado</conj>
            <conj p="1p" esp="2">hemos estado</conj>
            <conj p="2p">habéis estado</conj>
            <conj p="3p" esp="2">han estado</conj>
         </tiempo>
```

Ortografia-laguntza

- Bezeroan: sartutako hitza indizeetan aurkitzen ez denean, espresio erregular bat eratu, eta zerbitzarira bidaltzen da (hitz zuzenaren hasierako letrak izan daitezkeenak adieraziz)
- Zerbitzarian: hasierako letra horietan oinarrituz, dagozkien indizeak arakatzen dira (SAX moduan), eta espresio erregularrarekin parekatzen diren sarrerak hautatu, eta bezerora bidaltzen dira

vurro	***[bv]h?[uúüw]s?(l|r|rr)h?[oó]d?***	***bv***	***buró, burro***
uebo	***h?[uúüw]h?[eé]s?[bv]h?[oó]d?***	***huw***	***huevo***
siudá	***[csxz]h?[iíy]h?[uúüw]s?dh?[aá]d?***	***csxz***	***ciudad***