SE/CS 351

XML

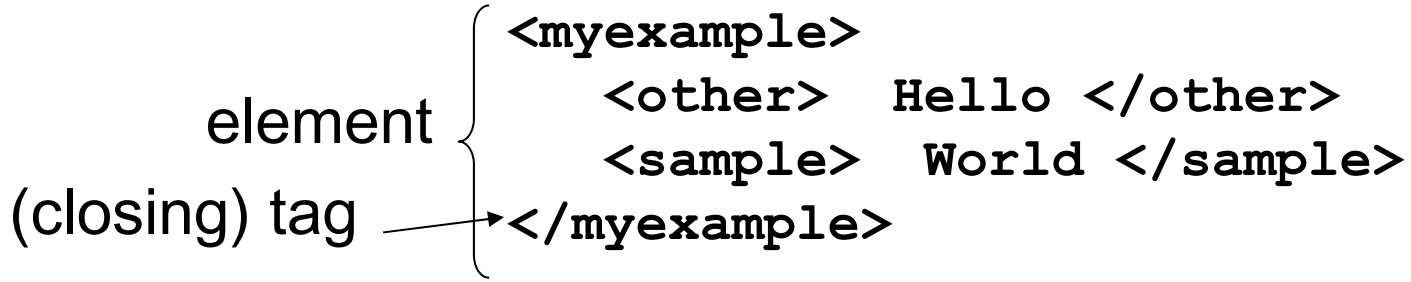
- Motivation from Databases
- well-formed XML
- XSLT

What's the language in your eclipse .project file?

```
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
    <name>pdstore</name>
    <comment></comment>
    <projects>
    </projects>
    <buildSpec>
        <buildCommand>
            <name>org.eclipse.emf.codegen.JETBuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
        <buildCommand>
            <name>org.eclipse.jdt.core.javabuilder</name>
            <arguments>
            </arguments>
        </buildCommand>
    </buildSpec>
    <natures>
        <nature>org.eclipse.emf.codegen.jet.IJETNature</nature>
        <nature>org.eclipse.jdt.core.javanature</nature>
    </natures>
</projectDescription>
```

XML introduction

- XML is an ASCII (plain text) language to represent data.
- XML is semistructured:
  - The structure of data can be defined ad hoc.
  - XML is “like HTML with arbitrary element names.”
- XML can be used to represent relational data for output.
- XML can be conveniently transformed with stylesheet language XSLT.



Well-formed XML: stack automaton

```
dbtable
dbtable row
dbtable row lastname
dbtable row √
dbtable row sample
dbtable row sample firstname
dbtable row sample √
dbtable row √
dbtable √
dbtable row
dbtable row lastname
dbtable row √
dbtable row firstname
dbtable row √
dbtable √
√
```

```
<dbtable>
  <row>
    <lastname> Grant
    </lastname>
    <sample>
      <firstname>
      </firstname>
    </sample>
  </row>
  <row>
    <lastname> Kay
    </lastname>
    <firstname> Jo
    </firstname>
  </row>
</dbtable>
```

Well-formed XML: nesting elements

- One main requirements for *well-formed* XML documents:
- elements nest properly
  - Can be defined through a **stack automaton** that goes through the document.
  - Every opening tag puts the element name onto the stack.
  - The stack content represents the currently open elements.
  - Every closing tag must match the topmost name on the stack.

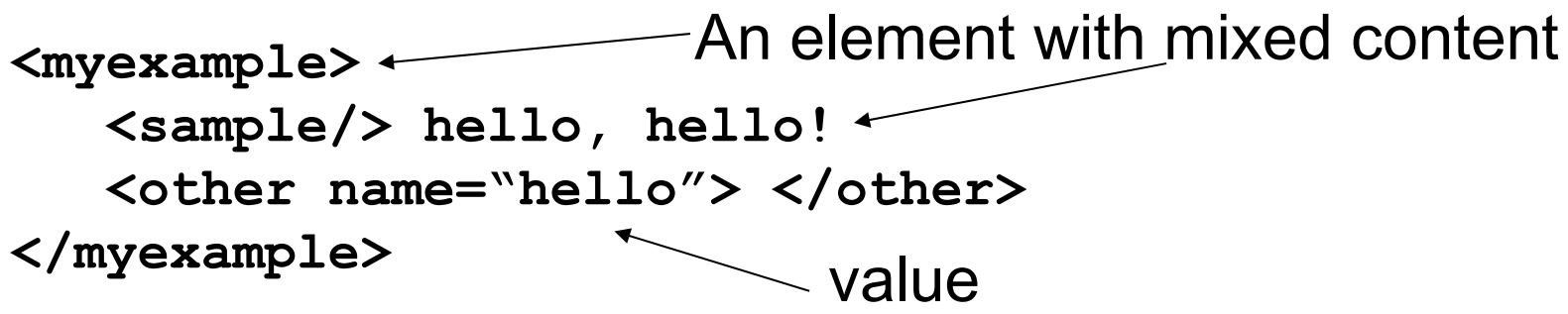
```
<dbtable>
  <row>
    <lastname> Grant
    </lastname>
    <sample>
      <firstname>
      </firstname>
    </sample>
  </row>
  <row>
    <lastname> Kay
    </lastname>
    <firstname> Jo
    </firstname>
  </row>
</dbtable>
```

Well-formed XML: example error

```
dbtable                                  <dbtable>
dbtable row                                <row>
dbtable row lastname                         <lastname> Grant
dbtable row √                                </lastname>
dbtable row sample                           <sample>
dbtable row sample firstname                   <firstname>
dbtable row sample √                           </firstname>
dbtable row √                                </sample>
dbtable √                                  </row>
dbtable row                                <row>
dbtable row lastname                         <lastname> Kay
dbtable row √                                </lastname>
dbtable row firstname                        <firstname> Jo
dbtable row firstname          ERROR         </lastnames>
dbtable (√ best-effort)                    </row>
√                                        </dbtable>
```

Technicality: XML elements and attributes

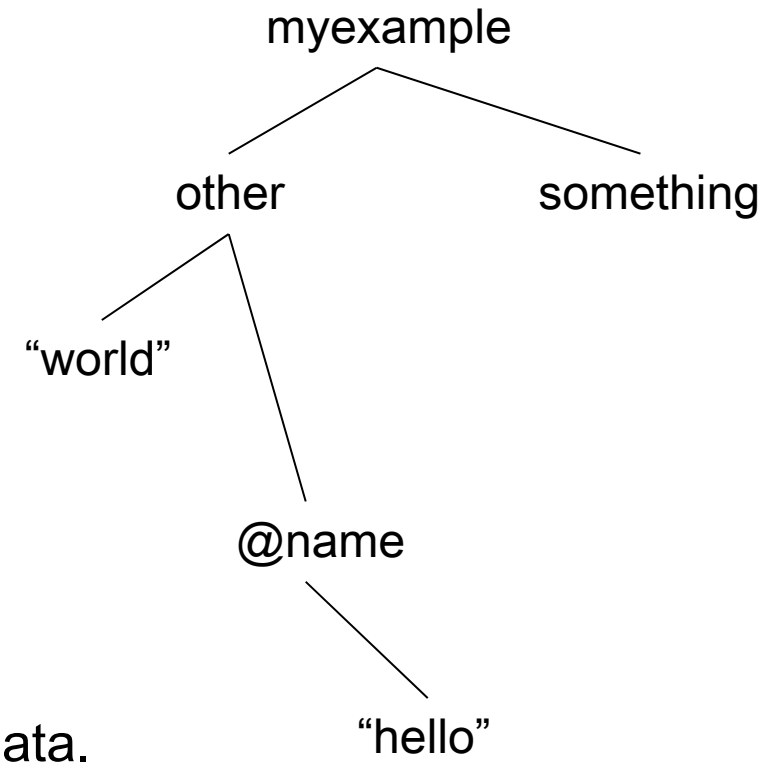
- Elements can contain nested elements: *child elements*
  - Elements can contain strings.
  - If an element contains strings and nested elements it *has mixed content*.
- Elements can have attributes within the start tag.
- One XML file has exactly one *root element*, here myexample.
- Attributes have a name and a value, separated by "="
- The value is enclosed in double quotes and is a string.



XML abstract syntax

- XML files represent a piece of information that has a tree structure. Is particularly nice for non-mixed content.

```
<myexample>
   <other name="hello">
     world
   </other>
   <something />
</myexample>
```



- (For mixed content the order becomes more important.)

[Buneman, P. 1997. Semistructured data. PODS, 1997]

XML can be used to represent data

- Contents of a table or results of a query can be represented as an XML file.
- The following is an ad-hoc representation of relational data.

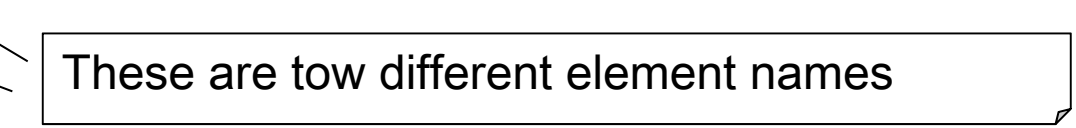
```
<dbtable>
  <row>
    <lastname> Grant </lastname>
    <firstname> Ann </firstname>
  </row>
  <row>
    <lastname> Kay </lastname>
    <firstname> Jo </firstname>
  </row>
</dbtable>
```

Technicality: XML namespaces

- Namespaces: packages for element and attribute names.
  Namespaces are identified with an URL
  - The URL serves only as an unique ID (is not accessed)
  - They are aliased with a shorthand.
- The alias is valid in the current element
- Tags with different namespaces do not clash.

```
<example
  xmlns:a=http://www.formcharts.org/ea/name1.xml>
<b:sample
xmlns:b=http://www.formcharts.org/ea/name2.xml>
    <a:hello/> <b:hello/>
</b:sample>
</example>
```

!



XSLT – a transformation language for XML

- XSLT = XML stylesheet language transformations
- An XSLT stylesheet
  - Is an XML document following the XSLT specification.
  - Represents a function:
    - Is applied to one XML document
    - Produces a new XML document.
- Can be evaluated by XSLT tools, including web browsers!
- Uses a tree-walker approach:
  - Contains translation rules, these apply to elements.
- Can be used to generate XHTML out of XML
  - Alternative approach to generators like JSP

XSLT by example

```
<bold>
  <emph>Hello</emph> World!
  <emph>hithere</emph>
</bold>
```

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

<xsl:template match="bold">
  <p><b><xsl:apply-templates/></b></p>
</xsl:template>

<xsl:template match="emph">
  <i><xsl:apply-templates/></i>,
</xsl:template>

</xsl:stylesheet>
```

matching of inner elements must be explictly triggered

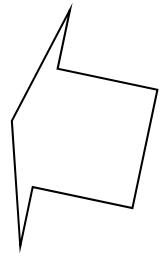
Rules

```
<p>
<b><i>Hello</i>, World!
    <i>hithere</i> , </b>
</p>
```

Can we already produce these examples?

Grant , *Ann*

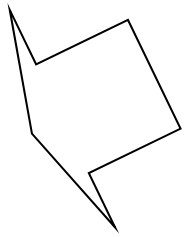
Kay , *Jo*



?

Ann **Grant**

Jo **Kay**



```
<?xml version="1.0"
encoding="utf-8"?>
<dbtable>
  <row>
    <lastname> Grant
    </lastname>
    <firstname> Ann
    </firstname>
  </row>
  <row>
    <lastname> Kay
    </lastname>
    <firstname> Jo
    </firstname>
  </row>
</dbtable>
```

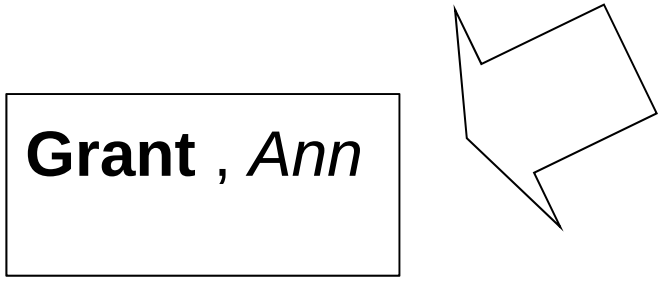
```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="dbtable">
  <html><head></head>
  <body><xsl:apply-templates/></body></html>
</xsl:template>
<xsl:template match="row">
  <p> <xsl:apply-templates
          select="firstname"/>
      <xsl:apply-templates
           select="lastname"/> </p>
</xsl:template>
<xsl:template match="firstname">
  <i><xsl:apply-templates/></i>
</xsl:template>
<xsl:template match="lastname">
  <b><xsl:apply-templates/></b>
</xsl:template>
</xsl:stylesheet>
```

```
<dbtable>
  <row>
    <lastname> Grant
    </lastname>
    <firstname> Ann
    </firstname>
  </row>
  <row>
    <lastname> Kay
    </lastname>
    <firstname> Jo
    </firstname>
  </row>
</dbtable>
```

Technicality: Invocation of stylesheets

- The XSLT stylesheet can be named in the XML file.
- Current web browsers understand XSLT
  - apply it to the XML file
  - They render the result:

```
<?xml version="1.0"
encoding="utf-8"?>
<?xml-stylesheet
    type="text/xsl"
    href="bold_last.xsl"?>
<dbtable>
  <row>
    <lastname> Grant
    </lastname>
    <firstname> Ann
    </firstname>
  </row>
</dbtable>
```



Select attributes use XPath

- Path expressions relative to current position:
  - “row”
    - Matches all <row> children in the current context
  - “row/name”
    - Matches all <name> elements that have a parent of <row>
- Path expressions can be also relative to the document root:
  - “/”
    - Matches the root of the document
  - “/dbtable” :
    - Matches all <dbtable> children of the root.

Wildcards in XPath

- Path expressions with wildcards:
  - “list/*/name ”
- The XPath expression "list/*/name“ matches all elements <name> that are in an arbitrary element that in turn is in an element <list>.
- One can use the wildcard “*” like an ordinary element name (But one cannot do ”firstn*”)

Example with wildcards

```
<?xml version="1.0" encoding="utf-8"?>
<list>
<?xml-stylesheet type="text/xsl" href="wildcard.xsl"?>
  <ship>
    <name> Iolanthe II</name>
  </ship>
</list>
```

wildcard.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="list/*/name">
  <i><xsl:apply-templates/></i>
</xsl:template>
</xsl:stylesheet>
```

Iolanthe II

Common sample uses of XSLT elements

- Templates with `match="mytag"` can be used to hierarchically build an output page.
- `apply-templates` with `select="myelem"` can be used to cherry-pick content.
- apply-templates with `select="/…"` can be used to access crosscutting concerns of the data.
- Wildcards can be used to treat different content types in a unified manner.

Technicality: XML files, concrete syntax

- Some minor details of well-formed XML.
- XML files should begin with a prolog that
  - contains an XML declaration, typically:
    <?xml version="1.0" encoding="utf-8"?>
  - May contain a type declaration: <!DOCTYPE HTML ….>
- Most XML processors will be tolerant, e.g. accept missing prolog, but not e.g. missing namespace declaration.

```
<?xml version="1.0" encoding="utf-8"?>
<myexample name="hello">
   <sample/>
</myexample>
```