XML

Mark-up Languages

- Digitalizing information
  - Content
  - Format
- SGML
- HTML
- XML
- RDF
- OWL …

SGML

- SGML (Standard Generalized Markup Language)
  - Language to define content and format of online information
  - Mother of all markup languages
  - Existing earlier than the Web
- Originated from GML (IBM, 1969) which is used to solve heterogeneous problem of document formats
- Becomes ISO Standard, and renamed as SGML
- Allows user-defined markup tags
- Platform-free, structure, extendable

SGML Components

- Grammar definition
  - Defines grammer for document type and document instances
- Document type defintion (DTD)
  - Defines logical structure and item type for document instances
- Document instances
  - Contains all the instance data
  - the main part of SGML file

From SGML to HTML

- SGML is too complicate and hard to master
- Write a browser for processing SGML becomes difficult
  - W3C propose HTML

HTML

- Hyper Text Markup Language
- Subset of SGML
- No user-defined tags
- No DTD
- Easy to learn
- Easy to write browser

From HTML to XML

- No user-defined tags
- Describes only data format, no content
- Lack of compatibility with other popular browsers
- Too many incorrect HTML files (wrong HTML grammer)
  - XML

XML

- eXtensible Markup Language
- Data format and data content
- User-defined tags
- Has its own grammer
- Describe structured and unstructured data
  - Structured: database, table,
  - Unstructured: webpage, eCommerce document, etc.
- Platform for storing and sharing data (Oracle, IBM, Microsoft)

More about XML

- Simplified SGML
- describing data format and content
- Storing structured and unstructured data
- Extensible (user-defined tag)
- Platform-free
- Text-based (any text editor), Unicode-based (language free)

XML Example

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

<customertable>
        <customer>
                <company>Northeast Invention Inc.</company>
                <contact>Alice Heath</contact>
                <photo file="alice.gif"/>
                <position>Marketing Director</position>
                <address>East Avenue 52</address>
                <tel>493 972904</tel>
        </customer>

        <customer>
                <company>Insight Inc.</company>
                <contact>Tom Hepp</contact>
                <photo/>
                <position>Sales Representative</position>
                <address>Sundown Avenue 30</address>
                <tel>676 873201</tel>
        </customer>
</customertable>
```

XML Grammer

- XML file structure
- Key components
- Properties
- Namespace
- Valid XML document

XML File Structure

- Containing two main parts:
  - Prolog
    - XML declaration (XML version and the encoding used)
    - PI (Processing Instruction)

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

  - Root element

```
<customertable>
        <customer>
                <name> ....</name>
                ...
        </customer>
        ...
</customertable>
```

XML Declaration

- <?xml
  - Shows the beginning of xml document
- ?>
  - Shows the end of the declaration
- version=“1.0”
  - Shows xml version information, which states that this xml document follow W3C XML1.0 Standard.
- encoding=“UTF-8”
  - Allows to use different encodings, such as UTF-8, UTF-16, GB2312
  - By default: UTF-8
- standalone=“yes”
  - DTD is included in the xml document
  - “no” means external dtd will be referenced here.
  - Default: no

XML: Comments

- Comments are a special set of tags that start with `<!--` and end with `-->`
- All data written between these two tags is ignored by the XML processor.
- Comments are usually used to make small notes inside the XML document or to comment out entire sections of XML code

```
<!-- I HAVE TO GET GUSTAVS EMAIL
  <employee name="Gustav Sielmann" >
    <email/>
  </employee>
-->
```

XML Element

- Tag must have starting tag and ending tag.
- Attributes can be put in starting tag
- Case sensitive
- Tag name cannot have space in between

```
<tag>data</tag>
```

XML Element

- Non empty element

```
<customer>
       <company>Northeast Invention Inc.</company>
       <contact>Alice Heath</contact>
       <position>Marketing Director</position>
       <address>East Avenue 52</address>
       <tel>493 972904</tel>
</customer>
```

- Empty element

```
<resume></resume>
BR (add enter),
<resume name=”Ying Ding” gender=”female” />
```

Nested XML Element

- Only one root element
- Other elements are all nested under root element
  - parent element
  - child element
  - sub-element
- Nesting rule is strict, if it is wrong, then parser shows the wrong information (while html does not)

Strict Nesting Rule

- Right nesting

```
<customer>
        <company>Northeast Invention Inc.</company>
        <contact>Alice Heath</contact>
        <position>Marketing Director</position>
        <address>East Avenue 52</address>
        <tel>493 972904</tel>
</customer>
```

- Wrong nesting

```
<customer>
        <company>Northeast Invention Inc.</customer>
</company>
```

XML Attribute

- User can define his own attribute
- Attribute has to stay in the beginning tag
  - Non empty element

```
<tagname attributename=“value” attributename=“value”
attributename=“value” ...>data</tagname>
```

  - Empty element

```
<tagname attributename=“value” attributename=“value”
attributename=“value” .../>
```

XML Attribute

- One element can have many attributes,
- Use space to separate each attribute
- Attribute name and value should appear in pair, using “=“ to link them
- Name convention for attribute is the same as for element
- One element cannot have more than one attribute with the same name.
- Attribute value must use ‘’ or “” (while HTML does not need that)

Balancing attribute and element

```
<customer>
        <contact gender=“female” birthday=“18/05/1978”>
        Alice Heath</contact>
</customer>
```

```
<customer>
        <contact>Alice Heath</contact>
        <gender>female</gender>
        <birthday>18/05/1978</birthday>
</customer>
```

Attribute definition

- attributename=“attributevalue”
- Attribute value must be quoted using “” or ‘’
- Attribute value can not contain: “<“, “>”, “&”, “ ’ ”, “ “ ”

```
<contact middlename=“”Y””>
<media type=“<CD>”>
<weather forecast=“cold & windy”>
```

XML: Entity References

- Entity references are used to reference data that is not directly in the structure (not available on your computer keyboards)
  - "ñ" = "&#241;" → "España" = "Espa&#241;a",
- Pre-built entity references are used to represent special characters, such as

  & `&amp;` < `&lt;` > `&gt;` " `&quot;` ' `&apos;`

  or character-References: `&#211;` (decimal), `&#xF3;` (hex)

- New entities can be declared in DTDs

e.g. the string `Peter&Tom("Don't cry for me")` would be written:

```
Peter&amp;Tom(&quot;Don&apos;t cry for me&quot;)
```

XML reserved characters

Special Character	Char code	Decimal code
&	&amp;	&#38;
<	&lt;	&#60;
>	&gt;	&#62;
“	&quot;	&#34;
‘	&apos;	&#39

XML Namespace

- Allow to reuse existing defined markup vocabulary
- Solve the problem of “same element name and same attribute name” from different software packages
  - <name> can be book name, person name, company name, etc.
- One namespace corresponds to one DTD
- Using URI (Uniform Resource Identifier) to define namespace, which can be URL (Uniform Resource Locator) or URN (Uniform Resource Name)

Namespace definition

- It is defined in the starting tag of one element.
- Namespacename should be unique, cannot be xml, html, xsl, xmlns
- Element and attribute can have namespace

```
<tagname xmlns:namespacename="URI">
```

Namespace example

```
<?xml version=“1.0” encoding=“UTF-8”?>
<!- file name: namespace1.xml -->

<customertable
xmlns:cus=“http://www.aaa.com/customer.dtd”
xmlns:emp=“http://www.aaa.com/employee.dtd”>
   <cus:customer>
       <cus:contact>Thomas Luger</cus:contact>
       <cus:tel>493 972904</cus:tel>
       <emp:company>Martin Tony Brother</emp:company>
       <emp:address>East Avenue 52</emp:address>
   </cus:customer>
</customertable>
```

Default namespace

- The same as other namespaces, but without namespacename
- Scope covers the element with the namespace defined in the starting tag and all its subelements.
- For attribute, there is no default namespace

```
<tagname xmlns="URI">
```

Default namespace example

```
<?xml version="1.0" encoding="UTF-8"?>
<!- file name: namespace1.xml -->

<customertable xmlns="http://www.aaa.com/customer.dtd"
xmlns:emp="http://www.aaa.com/employee.dtd">
   <customer>
       <contact>Thomas Luger</contact>
       <tel>493 972904</tel>
       <emp:company>Martin Tony Brother</emp:company>
       <emp:address>East Avenue 52</emp:address>
   </customer>
</customertable>
```

XML and HTML

- Using namespace, you can write html code within xml document

```
<?xml version=“1.0” encoding=“UTF-8”?>
<?xml-stylesheet href=“first.css” type=“text/css” ?>
<!- file name: namespace2.xml -->
<data xmlns:HTML=“http://www.w3.org/TR/XHTML1”>
  <book>
        <title>XML Introduction</title>
        <HTML:a href=“mailto:aa@yahoo.com”>
               <author>Tom</author>
        </HTML:a>
        <picture>
        <HTML:img src=“xml.gif” width=“80” height=“80”>
        </HTML:img>
        </picture>
  </book>
</data>
```

namespace2.xml

XML document Validation

- User defined tag and attribute must follow the regulations.
- If an XML document fulfills the tag and attribute definition regulations, and is without using corresponding DTD, then this XML document is well-formed
- If a well-formed XML document uses a corresponding DTD and passes DTD validation, then it is a valid XML document

Well-formed XML document

- In more details:
  - First comes the xml declaration (xml version)
  - Only one root element allowed per XML file, other elements are all the sub-element (daughters) of this one root element
  - Tags must be correctly closed.
  - Correct nesting must be obeyed
  - Attributes have to use single or double quotation marks
  - Case sensitive tags
  - Character entity reference (if necessary)

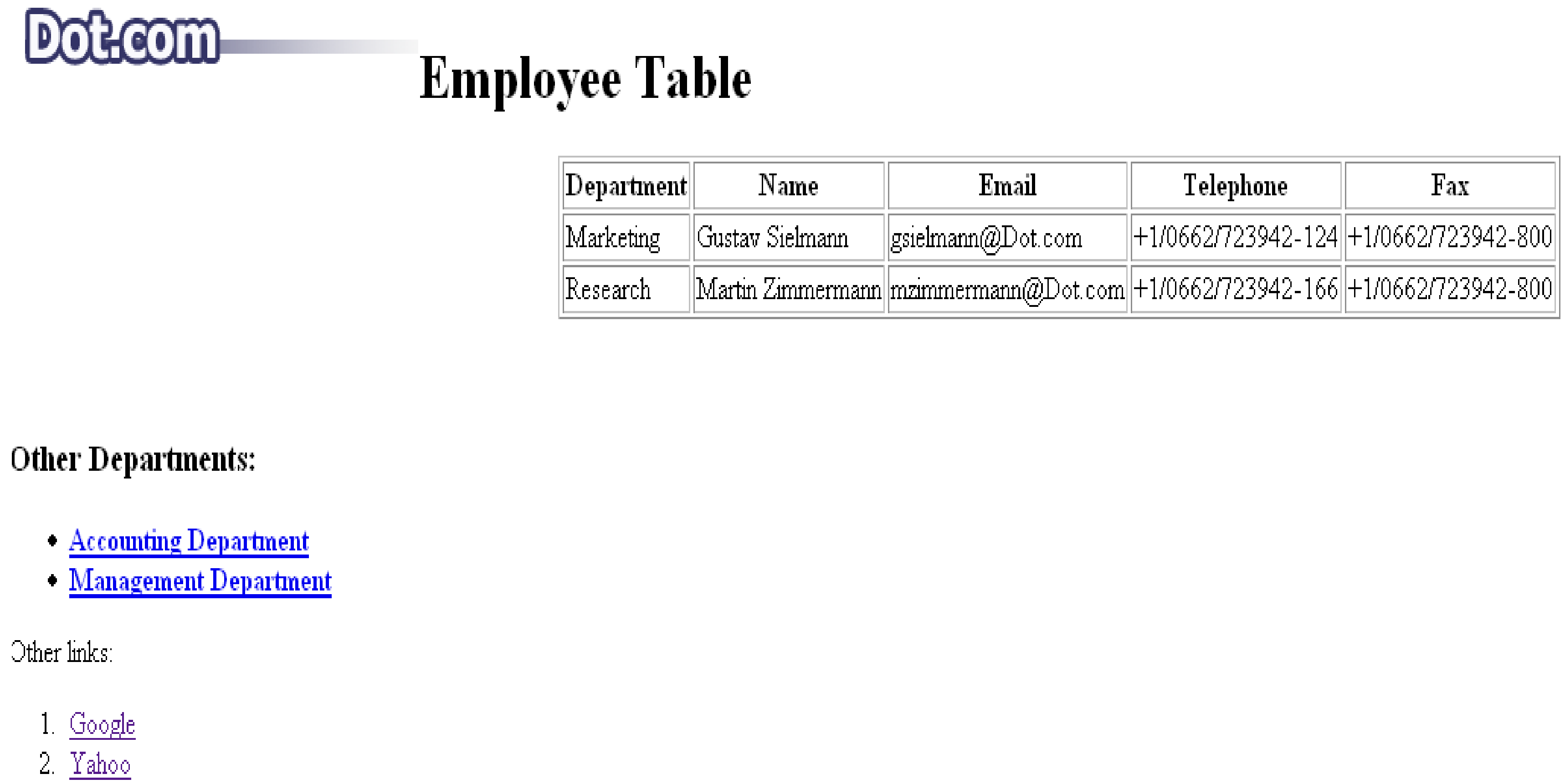
Summary of XML Syntax rule

- All XML elements must have a closing tag
- XML tags are case sensitive
- XML elements must be properly nested
- XML documents must have one and only one root element
- XML attribute values must be quoted (single quotation mark or double qutoation mark)

Reference

- W3C Schools
  http://www.w3schools.com/xml/default.asp
- W3C XML
  http://www.w3.org/TR/2006/REC-xml-20060816/
- XML.com
  A Technical Introduction to XML
  http://www.xml.com/pub/a/98/10/guide0.html

XML Exercise



Dot.com

Employee Table

Department	Name	Email	Telephone	Fax
Marketing	Gustav Sielmann	gsielmann@Dot.com	+1/0662/723942-124	+1/0662/723942-800
Research	Martin Zimmermann	mzimmermann@Dot.com	+1/0662/723942-166	+1/0662/723942-800

Other Departments:

- Accounting Department
- Management Department

Other links:

1. Google
2. Yahoo

XML Exercise

- Write the HTML file of above webpage
- Use XML to represent the data in the table

Answer:

dot.html

dot.xml