Semi-structured Data

4 - Document Type Definitions (DTDs)
Outline

- DTDs at First Glance
- Validation
- Document Type Declaration
- Internal DTD Subsets
- Element Declarations
- Attribute Declarations
- Entity Declarations (by Example)
- Namespaces and DTDs
- Limitations of DTDs
DTDs at First Glance

• Agreement to use only certain tags - interoperability

• Such a set of tags is called XML application - application of XML on a particular domain (e.g., phonebook, real estate, etc.)

```
<person>
  <name>
    <first>Andreas</first>
    <last>Pieris</last>
  </name>
  <tel>740072</tel>
  <fax>18493</fax>
  <email>pieris@dbai.tuwien.ac.at</email>
</person>

<house>
  <address>
    <street>Bräuhausgasse</street>
    <number>49</number>
    <postcode>A-1050</postcode>
    <city>Vienna</city>
  </address>
  <rooms>3</rooms>
</house>
```
DTDs at First Glance

- **Schema** - the markup permitted in a particular application

- Many different XML schema languages available:
  - Document Type Definitions (DTDs)
  - W3C XML Schema
  - REgular LAnguage for XML Next Generation (RELAX NG)
  - Schematron
  - ...

- In the context of this course we are going to see DTDs and W3C XML Schema

  ...but for the moment let us focus on DTDs
DTDs at First Glance

- A DTD lists all the elements and attributes the document uses

```xml
<!ELEMENT person (name, tel, fax, email+)>
<!ATTLIST person id_number ID #REQUIRED>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

**ATTENTION:** The order of the declarations is not significant
Validation

• When a document matches a schema is valid; otherwise, is invalid

```
<!ELEMENT person (name, tel, fax, email+)>
<!ATTLIST person id_number ID #REQUIRED>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>           <person id_number="E832740">
<!ELEMENT last (#PCDATA)>            <name>
<!ELEMENT tel (#PCDATA)>            <first> Andreas </first>
<!ELEMENT fax (#PCDATA)>            <last> Pieris </last>
<!ELEMENT email (#PCDATA)>          </name>
                                         <tel> 740072 </tel>
                                         <fax> 18493 </fax>
                                         <email> andreas.pieris@tuwien.ac.at </email>
                                         <email> pieris@dbai.tuwien.ac.at </email>
                                         ✓
```
Validation

- When a document matches a schema is **valid**; otherwise, is **invalid**

```xml
<!ELEMENT person (name, tel, fax, email+)>
<!ATTLIST person id_number ID #REQUIRED>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>

<person id_number="E832740">
  <name>
    <first>Andreas</first>
    <last>Pieris</last>
  </name>
  <fax>18493</fax>
  <tel>740072</tel>
  <email>andreas.pieris@tuwien.ac.at</email>
  <email>pieris@dbai.tuwien.ac.at</email>
</person>
```
Validation

- Validating parsers - check both for well-formedness and validity
- Validating errors may be ignored (unlike well-formedness errors)
- Whether a validity error is serious depends on the application

**ATTENTION:** Validity errors are not necessarily fatal
Document Type Declaration

• A valid document contains a URL indicating where the DTD can be found
• This is done via the document type declaration - after the XML declaration

```xml
<!DOCTYPE person SYSTEM "http://www.mysite.com/dtds/person.dtd">
```

root element of the document

where the DTD can be found

**ATTENTION:** DTD = Document Type Definition (not Declaration)
Document Type Declaration

- **Relative URL** - if the document and the DTD reside in the same base site

  ```xml
  <!DOCTYPE person SYSTEM "/dtds/person.dtd">
  ```

- **Just the file name** - if the document and the DTD are in the same directory

  ```xml
  <!DOCTYPE person SYSTEM "person.dtd">
  ```
Document Type Declaration: Public IDs

• The keyword `SYSTEM` is used for DTDs defined by the user

• For official, publicly available DTDs, the keyword `PUBLIC` is used

```
<!DOCTYPE person SYSTEM "http://www.mysite.com/dtds/person.dtd">
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "xhtml11.dtd">
```

Public ID uniquely identifies the XML application in use

Backup URL in case the public ID is not recognizable
Document Type Declaration: Public IDs

- Anatomy of the public ID

```
"-//W3C//DTD XHTML 1.1//EN"
```

- owner identifier
  - indicates unregistered IDs
  + indicates registered IDs

- text identifier
  DTD - class
  XHTML 1.1 - description
  EN - language

... but public IDs are not used very much in practice
Internal DTD Subsets

• A DTD can be directly given in the document (between [ ])

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE person [ 
  <!ELEMENT person (name, tel, fax, email+)>
  <!ATTLIST person id_number ID #REQUIRED>
  <!ELEMENT name (first, last)>
  <!ELEMENT first (#PCDATA)>
  <!ELEMENT last (#PCDATA)>
  <!ELEMENT tel (#PCDATA)>
  <!ELEMENT fax (#PCDATA)>
  <!ELEMENT email (#PCDATA)>
]>
<person id_number="E832740">
  <name>
    <first>Andreas</first>
    <last>Pieris</last>
  </name>
  <tel>740072</tel>
  <fax>18493</fax>
  <email>andreas.pieris@tuwien.ac.at</email>
  <email>pieris@dbai.tuwien.ac.at</email>
</person>
```
Internal DTD Subsets

• Only part of the DTD can be directly given in the document (between [ ])

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE  person  SYSTEM "person_text.dtd" [  
   <!ELEMENT person (name, tel, fax, email+)>
   <!ATTLIST person id_number ID #REQUIRED>
   <!ELEMENT name (first, last)>
 ]>
<person id_number="E832740">
   <name>
      <first> Andreas </first>
      <last> Pieris </last>
   </name>
   <tel> 740072 </tel>
   <fax> 18493 </fax>
   <email> andreas.pieris@tuwien.ac.at </email>
   <email> pieris@dbai.tuwien.ac.at </email>
</person>
```

person_text.dtd:

```
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT tel (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
<!ELEMENT email (#PCDATA)>
```

not a standalone document
Internal DTD Subsets

- DTD = internal DTD subset \cup external DTD subset

\[
\begin{align*}
\text{<?xml version=}&"1.0" \text{ encoding="UTF-8" standalone="no"?>} \\
\text{<!DOCTYPE } \text{ person SYSTEM "person_text.dtd" [} \\
\text{<!ELEMENT person (name, tel, fax, email+)>} \\
\text{<!ATTLIST person id_number ID #REQUIRED}> \\
\text{<!ELEMENT name (first, last)>} \\
\text{]} > \\
\text{<person id_number="E832740">} \\
\text{<name>} \\
\text{<first> Andreas </first>} \\
\text{<last> Pieris </last>} \\
\text{<name>} \\
\text{<tel> 740072 </tel>} \\
\text{<fax> 18493 </fax>} \\
\text{<email> andreas.pieris@tuwien.ac.at </email>} \\
\text{<email> pieris@dbai.tuwien.ac.at </email>} \\
\text{</person>}
\end{align*}
\]

ATTENTION: The two subsets must be compatible - no multiple declarations
Up to Now

• DTDs at First Glance
• Validation
• Document Type Declaration
• Internal DTD Subsets
• Element Declarations
• Attribute Declarations
• Entity Declarations (by Example)
• Namespaces and DTDs
• Limitations of DTDs
Element Declarations

• Every element used in a valid document must be declared

• This is done via an element declaration

```xml
<!ELEMENT element-name content-specification>
```

indicates what children the element must or may have, and in which order
Element Declarations: #PCDATA

- An element may only contain parsed character data

```xml
<!ELEMENT name (#PCDATA)>
```

**Valid:**

```xml
<name> Andreas Pieris </name>
```

**Invalid:**

```xml
<name>
  <first> Andreas </first>
  <last> Pieris </last>
</name>
```
Element Declarations: Child Elements

- An element must have one child element

```xml
<!ELEMENT person (name)>
<!ELEMENT name (#PCDATA)>
```

Invalid:

```xml
<person>
  <name> Andreas Pieris </name>
  <tel> 740072 </tel>
</person>
```

Valid:

```xml
<person>
  <name> Andreas Pieris </name>
</person>
```
Element Declarations: Sequences

- An element has multiple child elements

```xml
<!ELEMENT  name  (first, last)>
<!ELEMENT  first  (#PCDATA)>
<!ELEMENT  last  (#PCDATA)>
```

Invalid 1:
```xml
<name>
  <last> Pieris </last>
</name>
```

Valid:
```xml
<name>
  <first> Andreas </first>
  <last> Pieris </last>
</name>
```

Invalid 1:
```xml
<name>
  <last> Pieris </last>
</name>
```
Element Declarations: Sequences

• An element has **multiple child element**

```xml
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
```

Invalid 2:

```xml
<name>
  <last> Pieris </last>
  <first> Andreas </first>
</name>
```

Valid:

```xml
<name>
  <first> Andreas </first>
  <last> Pieris </last>
</name>
```

Invalid 2:

```xml
<name>
  <first> Andreas </first>
  <last> Pieris </last>
</name>
```
Element Declarations: Sequences

- An element has multiple child elements.

\[
\text{<!ELEMENT name (first, last)>}
\]
\[
\text{<!ELEMENT first (#PCDATA)>}
\]
\[
\text{<!ELEMENT last (#PCDATA)>}
\]

Valid:

\[
<\text{name}>
  \text{<first> Andreas </first>}
  \text{<last> Pieris </last>}
\text{</name>}
\]

Invalid 3:

\[
<\text{name}>
  \text{<first> Andreas </first>}
  \text{<middle> T. </middle>}
  \text{<last> Pieris </last>}
\text{</name>}
\]
Element Declarations: Number of Children

- Not all instances of an element have the same children

```
<name>
  <first> Andreas </first>
  <last> Pieris </last>
</name>
```

```
<name>
  <first> Andreas </first>
  <middle> T. </middle>
  <last> Pieris </last>
</name>
```

```
<name>
  <first> Andreas </first>
  <middle> T. </middle>
  <middle> A. </middle>
  <last> Pieris </last>
</name>
```

- Sequences are not enough to make all the above documents valid

- Occurrence indicators (?,*,+)

Element Declarations: Number of Children

- Occurrence indicators (?, *, +)

ATTENTION: DTDs cannot specify the exact number of occurrences, or say at most k or at least k occurrences.
Element Declarations: Number of Children

<!ELEMENT  name  (first, middle*, last)>
<!ELEMENT  first  (#PCDATA)>
<!ELEMENT  middle  (#PCDATA)>
<!ELEMENT  last  (#PCDATA)>
Element Declarations: Choices

• Exactly one child element from a predefined list of elements

<!ELEMENT day (Mon | Tue | Wed)>
<!ELEMENT Mon (#PCDATA)>
<!ELEMENT Tue (#PCDATA)>
<!ELEMENT Wed (#PCDATA)>

Valid:    <Mon> Monday </Mon>
</day>

Invalid:  <Mon> Monday </Mon>
<Wed> Wednesday </Wed>
</day>

ATTENTION: The separator | is interpreted as exclusive OR
Element Declarations: Parentheses

- Individual elements, sequences, ?, *, + and choices are rather limited

- E.g., we cannot say a name element may contain:
  - Just a first name,
  - Just a last name, or
  - A first and a last name with an arbitrary number of middle names

- Combine the above features in an arbitrary way - (nested) parentheses
Element Declarations: Parentheses

```xml
<!ELEMENT  person  (name, (tel | email))>
<!ELEMENT  name  (first, last)>
<!ELEMENT  first  (#PCDATA)>
<!ELEMENT  last  (#PCDATA)>
<!ELEMENT  tel  (#PCDATA)>
<!ELEMENT  mail  (#PCDATA)>
```

A person element contains a name element, and either a tel or an email
Element Declarations: Parentheses

```
<!ELEMENT books-catalogue ((title, author, year?)+)>  
<!ELEMENT title (#PCDATA)>                     
<!ELEMENT author (#PCDATA)>                    
<!ELEMENT year (#PCDATA)>                      
```

A books-catalogue element consists of a non-empty list of triples of the form title, author, year, with the year being optional.
Element Declarations: Parentheses

<!ELEMENT name (last
| (first, ((middle+, last) | last?))
)>=

<!ELEMENT first (#PCDATA)>

<!ELEMENT middle (#PCDATA)>

<!ELEMENT last (#PCDATA)>
Element Declarations: Parentheses

<!ELEMENT name (last
  | (first, ((middle+, last) | last?))
  )>

<!ELEMENT first (#PCDATA)>  
<!ELEMENT middle (#PCDATA)>  
<!ELEMENT last (#PCDATA)>

A name element may contain:

- Just a first name,
- Just a last name, or
- A first and a last name with an arbitrary number of middle names
Element Declarations: Mixed Content

- An element may contain **both child elements and character data**

  ```
  <definition>
  The term <term> Semi-structured Data </term> refers to a form of structured data that does not conform with the formal structure of relational data
  </definition>
  ```

- **Mixed content** - (non-whitespace) text and elements

  ```
  <!ELEMENT definition (#PCDATA | term)*> 
  <!ELEMENT term (#PCDATA )>
  ```

**ATTENTION:** This is the only way to declare mixed content
Element Declarations: Empty Content

- **Empty elements**, i.e., without a content, are declared as

  ```xml
  <!ELEMENT  element-name  EMPTY>
  ```

  **Valid:**
  ```xml
  <element-name></element-name>
  or
  <element-name/>
  ```

  **Invalid:**
  ```xml
  <element-name>  </element-name>
  ```
Element Declarations: Any Content

- We can say that an element **simply exists**, without any restrictions

  ```xml
  <!ELEMENT element-name ANY>
  ```

- It is useful during the designing phase of a DTD

- In general, it is a bad design to use ANY in finished DTDs

**ATTENTION:** ANY does not allow undeclared child elements
Up to Now

- DTDs at First Glance
- Validation
- Document Type Declaration
- Internal DTD Subsets
- Element Declarations
- Attribute Declarations
- Entity Declarations (by Example)
- Namespaces and DTDs
- Limitations of DTDs
Attribute Declarations

• Every attribute used in a valid document must be declared

• This is done via an attribute declaration

<!ATTLIST element-name attr-name₁ attr-type₁ attr-default₁

  ...

  attr-nameₙ attr-typeₙ attr-defaultₙ>

ATTENTION: The order of the attributes is not significant
Attribute Declarations: Attribute Types

• Up to now, attribute values can be any string of text

• … except the symbols < and & - we need to use &lt; and &amp;

• DTDs can make stronger statements about the attribute values - attribute type

• There are ten attribute types in XML:
  - CDATA
  - NMTOKEN
  - NMTOKENS
  - Enumeration
  - ID
  - IDREF
  - IDREFS
  - ENTITY
  - ENTITIES
  - NOTATION

  details follow

  check out the textbook (XML in a Nutshell, Chapter 3)
Attribute Types: CDATA

• An attribute may contain any text acceptable in a well-formed document

  <!ELEMENT  book  price  CDATA  #REQUIRED>

• A price is in the form €20.00 - only CDATA allows for such values
Attribute Types: NM TOKEN

• XML name token - legal XML name, but can start with any allowed character

• Recall that XML names can start only with a letter or underscore

• NM TOKEN - an attribute can take XML name tokens

```xml
<!ATTLIST course date NM TOKEN #REQUIRED>
<!ELEMENT course (#PCDATA)>
```

Valid:  
```xml
<course date="05-03-2015"> SSD </course>
```

Invalid:  
```xml
<course date="05/03/2015"> SSD </course>
```
Attribute Types: NMTOKENS

- An attribute may contain a list of XML name tokens (separated by whitespace)

```xml
<!ATTLIST course date NMTOKENS #REQUIRED>
<!ELEMENT course (#PCDATA)>
```

Valid:  
```xml
<course date="05-03-2015 12-03-2015"> SSD </course>
```

Invalid:  
```xml
<course date="05/03/2015 12/03/2015"> SSD </course>
```
Attribute Types: Enumeration

- List of possible values (separated by |)

```xml
<!ATTLIST course day (Mon | Thu) #REQUIRED>
<!ELEMENT course (#PCDATA)>
```

Valid:       ```xml
<course day="Thu"> SSD </course>
```

Invalid:     ```xml
<course day="Sun"> SSD </course>
```

**ATTENTION:** The only attribute type that is not an XML keyword
Attribute Types: ID

- An attribute must contain an XML name (not name token) that is unique
- Each element has at most one ID attribute - ID of an element

```xml
<!ATTLIST person id_number ID #REQUIRED>
<!ELEMENT person (#PCDATA)>
```

**Valid:**

```xml
<person id_number="_832740"> Andreas Pieris </course>
```

**Invalid:**

```xml
<person id_number="832740"> Andreas Pieris </course>
```
Attribute Types: IDREF

- An attribute must contain the value of some ID type attribute in the document

```xml
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

Valid:

```xml
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="e1"> E </manager>
```
Attribute Types: IDREF

- An attribute must contain the value of some ID type attribute in the document

```xml
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

```
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="p1"> E </manager>
```
Attribute Types: IDREF

- An attribute must contain the value of some ID type attribute in the document

```xml
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

Valid: 
```xml
<employee emp_id="e1"> E </employee>
<project proj_id="p1"> P </project>
<manager mgr_id="p1"> E </manager>
```

although conceptually wrong
(manager is a project)
Attribute Types: IDREF

- An attribute must contain the value of some ID type attribute in the document

```xml
<!ATTLIST employee emp_id ID #REQUIRED>
<!ATTLIST project proj_id ID #REQUIRED>
<!ATTLIST manager mgr_id IDREF #REQUIRED>
<!ELEMENT employee (#PCDATA)>
<!ELEMENT project (#PCDATA)>
<!ELEMENT manager (#PCDATA)>
```

Invalid:

```xml
<project proj_id="p1"> P </project>
```

Invalid: <manager mgr_id="m1"> E </manager>

m1 is not the value of an ID type attribute
Other Attribute Types

- **IDREFS** - list of IDs occurring in the document
- **ENTITY** - entity declared in the DTD (an example is given later)
- **ENTITIES** - list of entities declared in the document
- **NOTATION** - name of a notation declared in the DTD

... for more details, check out the textbook (XML in a Nutshell, Chapter 3)
Attribute Declarations: Attribute Defaults

• Recall how an attribute declaration looks like

```xml
<!ATTLIST element-name attr-name1 attr-type1 attr-default1
  ...
  attr-name_n attr-type_n attr-default_n>
```

- **#IMPLIED** optional, no default name
- **#REQUIRED** required, no default name
- **#FIXED** attribute value is constant and immutable
- **Default Name** the actual default value is given
Attribute Defaults: #FIXED

<!ATTLIST tuwien website CDATA #FIXED “http://www.tuwien.ac.at”>

Valid:

```
<tuwien website="http://www.tuwien.ac.at"> ... </tuwien>
```

or

```
<tuwien> ... </tuwien>
```

even if the attribute is not explicitly stated, it has the specified value

Invalid:

```
<tuwien website="www.tuwien.ac.at"> ... </tuwien>
```
Attribute Defaults: Default Value

<!ATTLIST course elective (yes | no) “no”>

<course elective=“yes”> ... </course>

or

Valid:  <course elective=“no”> ... </course>

or

<course> ... </course> - the value of elective is no

Invalid: <course elective=“true”> ... </course>
Entity Declarations: Example

• Recall that XML predefines five entities (lt, gt, amp, quot, apos)

• DTDs can define more entities via an entity declaration

• The following defines the entity ssd:

  ```xml
  <!ENTITY ssd “Semi-structured Data” >
  ```

• We can use &ssd; anywhere we need to type “Semi-structured Data”

… check out the textbook (XML in a Nutshell, Chapter 3)
Namespaces in DTDs

<!-- Students’ and University’s Evaluation -->
<course
  xmlns="http://www.oeh.ac.at"
  xmlns:univ="http://www.tuwien.ac.at">
  <title>SSD</title>
  <assessment>Fair</assessment>
  <univ:assessment>Elective</univ:assessment>
</course>

<!ELEMENT course (title, assessment, univ:assessment)>
<!ATTLIST course xmlns CDATA #FIXED "http://www.oeh.ac.at">
<!ATTLIST course xmlns:univ CDATA #REQUIRED>
<!ELEMENT title (#PCDATA)>
<!ELEMENT assessment (#PCDATA)>
<!ELEMENT univ:assessment (#PCDATA)>

ATTENTION: The validator does not care about namespaces - some element and attribute names happen to contain colons (:)
Check for Validity

- Easy way: online validator - http://www.xmlvalidation.com/

- Recommended: xmllint - http://xmlsoft.org/
  - Portable C library for Linux, Unix, MacOS, Windows, ...
  - Command line call: `xmllint --valid <xml-file-name>`
  - Check out http://www.dbai.tuwien.ac.at/education/ssd/current/uebung.html
Limitations of DTDs

- Not in XML syntax
  - Different parsers for the document and the DTD

- A weak specification language
  - No control on the exact number of child elements
  - Limited selection of data types
  - The notion of inheritance does not exist

- No explicit support of namespaces
  - The validator is completely unaware of the existence of namespaces

... W3C XML Schema