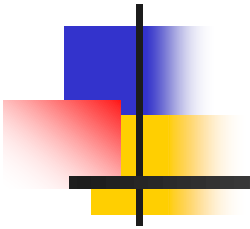


Module 3

DTDs

Module Overview

In this module, you will learn about:

- Document Type Definition
- Working with DTDs
- Valid XML Documents
- Declarations

Lesson 1 – Document Type Definition

In this first lesson, **Document Type Definition**, you will learn to:

- Define what is meant by a DTD.
- Identify the need for a DTD.

Definition of a DTD

- It is a non XML document made up of element, attribute and entity declarations.
- It helps XML parsers to validate the XML document.

Need for a DTD

- XML allows a user to define his/her own tag.
- Standardization of elements and attributes was needed.
- A DTD can define all the possible combinations and sequences for elements.

Lesson 2 – Working with DTDs

In this second lesson, **Working with DTDs**, you will learn to:

- Describe the structure of a DTD.
- Explain how to create a simple DTD.
- Describe what is meant by document type declarations.

Structure of DTD

- Element Declarations
- Attribute Declarations
- Entity Declarations

Creating Internal DTDs 1-3

- Declare all the possible elements
- Specify the permissible element children, if any
- Set the order in which elements must appear
- Declare all the possible element attributes
- Set the attribute data types and values
- Declare all the possible entities

Creating Internal DTDs 2-3

Syntax

```
<!ELEMENT element-name (element-content)>
…
<!ATTLIST element-name attribute-name attribute-
type default-value>
…
<!ENTITY entity-name “entity-value”>
…
```

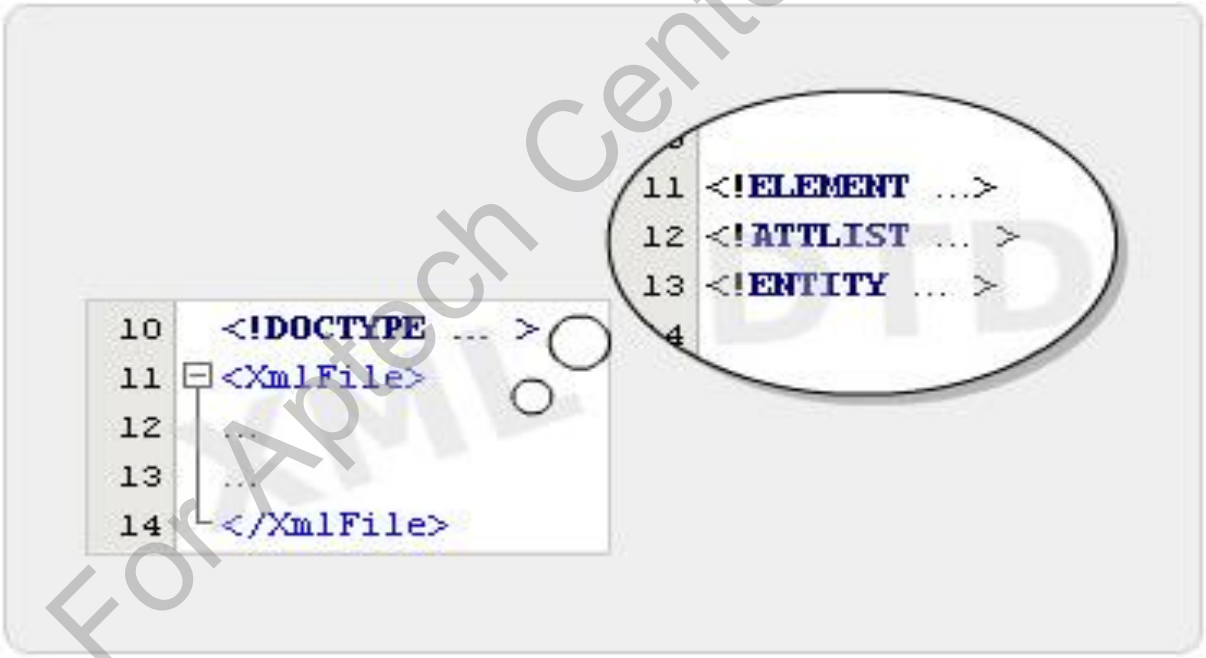
Creating Internal DTDs 3-3

Code Snippet

```
<!ELEMENT Mobile (Company, Model, Price,
Accessories)>
<!ELEMENT Company (#PCDATA)>
<!ELEMENT Model (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
<!ELEMENT Accessories (#PCDATA)>
<!ATTLIST Model Type CDATA "Camera">
<!ENTITY HP "Head Phones">
<!ENTITY CH "Charger">
<!ENTITY SK "Starters Kit">
```

DOCTYPE Declarations 1-2

- It specifies the name of the DTD and either its content or location.
- It begins with `<!DOCTYPE` and ends with a `>`.



DOCTYPE Declarations 2-2

Syntax

```
<!DOCTYPE name_of_root_element [ internal DTD
subset ]>
or
<!DOCTYPE name_of_root_element SYSTEM "URL of the
external DTD subset" >
```

Types of DTDs 1-2

- DTDs can be classified as Internal or External.
- **Internal DTDs**
  - It consists of the DTD name followed by the DTD enclosed in square brackets.

Internal DTD

```
14  ...
15  <!DOCTYPE Mobile SYSTEM "mobile.dtd">
16  ...
```

Types of DTDs 2-2

- **External DTDs**
  - It consists of the DTDs name followed by the `SYSTEM` keyword followed by the address

External DTD Reference

```
<!DOCTYPE Mobile [
   <!ELEMENT Mobile (Company, Model, Price, Accessories)>
   <!ELEMENT Company (#PCDATA)>
   <!ELEMENT Model (#PCDATA)>
   <!ELEMENT Price (#PCDATA)>
   <!ELEMENT Accessories (#PCDATA)>
   <!ATTLIST Model Type CDATA "Camera">
   <!ENTITY HP "Head Phones">
   <!ENTITY CH "Charger">
   <!ENTITY SK "Starters Kit">
]>
...
```

```
11 <!ELEMENT Mobile (Company, Model, Price, Accessories)>
12 <!ELEMENT Company (#PCDATA)>
13 <!ELEMENT Model (#PCDATA)>
14 <!ELEMENT Price (#PCDATA)>
15 <!ELEMENT Accessories (#PCDATA)>
16 <!ATTLIST Model Type CDATA "Camera">
17 <!ENTITY HP "Head Phones">
18 <!ENTITY CH "Charger">
19 <!ENTITY SK "Starters Kit">
```

Lesson 3 – Valid XML Documents

In this third lesson, **Valid XML Documents**, you will learn to:

- Define document validity.
- Describe in brief how to test for document validity.

Well-Formed XML documents 1-2

- All elements must be enclosed by the root element
- All elements must have closing tags
- All tags should be case sensitive
- All elements must be properly nested
- All attribute values must always be quoted

Well-Formed XML documents 2-2

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <Mail>
    <To>anne@xyz.com</To>
    <From>bob@xyz.com</From>
    <Date>27th February 2007</Date>
    <Time>11:30 am</Time>
    <Cc />
    <Bcc />
    <Subject>Meeting at Main Conference Room at 4:30pm</Subject>
    <Message>Hi, Kindly request you to attend the cultural body
      general meeting in the main conference room at 4:30 pm.
      Please be present to learn about the new activities being
      planned for the employees for this year. Yours sincerely,
      Bob</Message>
    <Signature />
  </Mail>
```

Valid XML documents 1-2

- A valid XML document is a well-formed XML document that adheres to its DTD.
- The validity of an XML document is determined by checking it against its DTD.

Valid XML documents 2-2

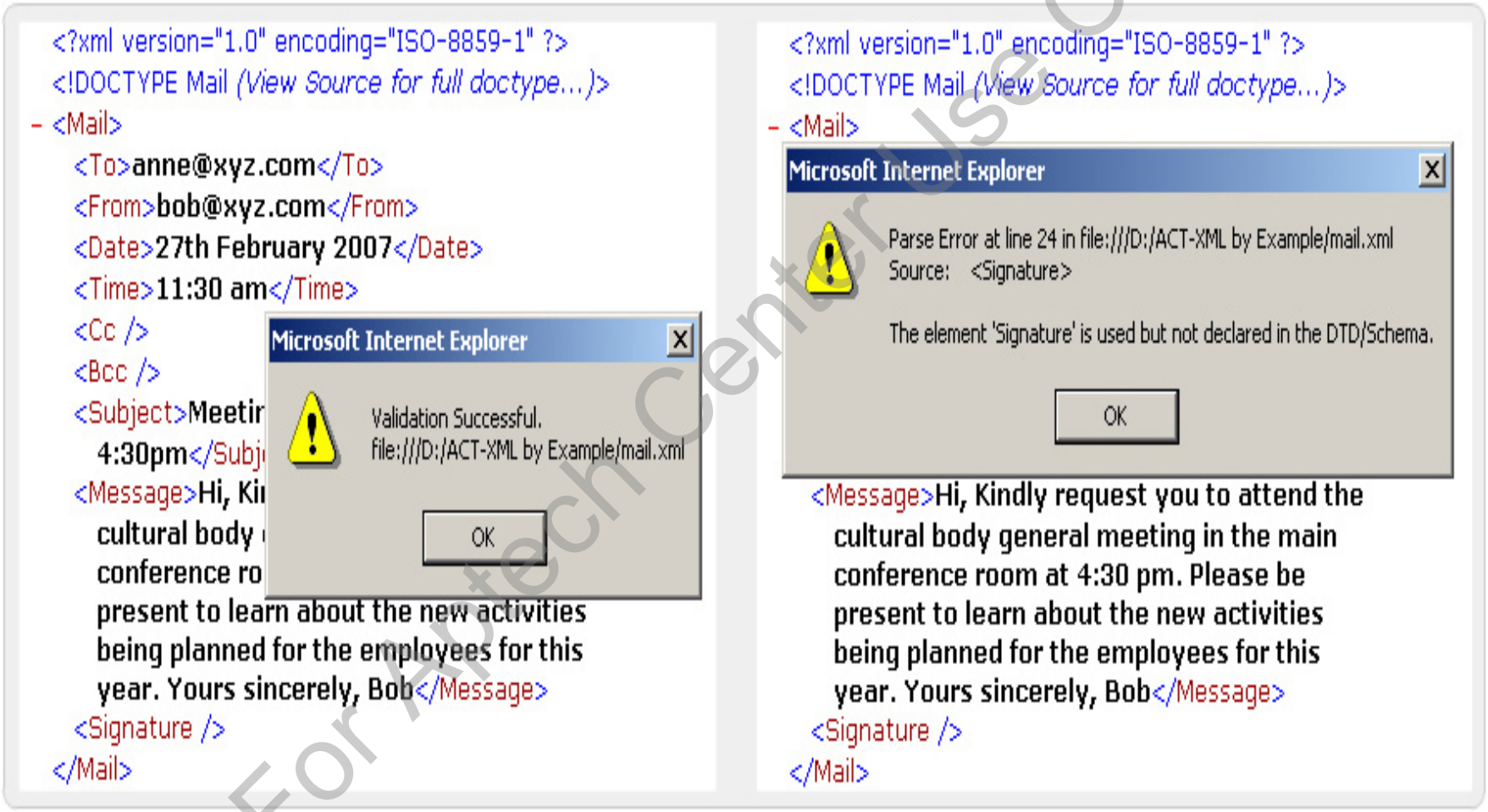
Code Snippet

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Mail [
<!ELEMENT Mail (To, From, Date, Time, Cc, Bcc, Subject, Message, Signatu
re)>
<!ELEMENT To (#PCDATA)>
<!ELEMENT From (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT Time (#PCDATA)>
<!ELEMENT Cc (#PCDATA)>
<!ELEMENT Bcc (#PCDATA)>
<!ELEMENT Subject (#PCDATA)>
<!ELEMENT Message (#PCDATA)>
<!ELEMENT Signature (#PCDATA)>
]>
<Mail>
<To> anne@xyz.com </To>
<From> bob@xyz.com </From>
<Date> 27th February 2007 </Date>
<Time> 11:30 am </Time>
<Cc> </Cc>
<Bcc> </Bcc>
<Subject> Meeting at Main Conference Room at 4:30pm </Subject>
<Message> Hi, Kindly request you to attend the cultural body general meeting in
the main conference room at 4:30 pm. Please be present to learn about the new
activities being planned for the employees for this year. Yours sincerely, Bob
</Message>
<Signature> </Signature>
</Mail>
```

Testing XML for Validity 1-2

- It can be determined by using a validating parser such as Microsoft XML Core Services (MSXML) 6.0 Parser.
- MSXML enables the Internet Explorer (IE) browser to validate the code.
- The first image displays the validation result of a valid mobile.xml file.
- The second image displays the validation result after the removal of the signature element.

Testing XML for Validity 2-2



Lesson 4 - Declarations

In this last lesson, **Declarations**, you will learn to:

- Explain how to declare elements.
- Explain how to declare attributes.
- Describe entity declaration in a DTD.

Declaring Elements

- XML elements are declared with an element declaration.

Syntax

```
<!ELEMENT element-name element-rule>
```

where,

`ELEMENT` is the keyword,

`element-name` is the name of the element,

`element-rule` can be one of the following: No Content, Only Parsed Character Data, Any Contents, Children, Only One Occurrence, Minimum One Occurrence, Zero or More Occurrences, Zero or One Occurrence, Either/Or Content or Mixed Content

Testing XML for Validity

```
15 <!ELEMENT Mobile ( Company, Model, Price, Accessories)>
16   <!ELEMENT Company (#PCDATA)>
17   <!ELEMENT Model (#PCDATA)>
18   <!ELEMENT Price (#PCDATA)>
19   <!ELEMENT Accessories (#PCDATA)>
20 <!ATTLIST Model Type CDATA "Camera">
21 <!ENTITY HP "Head Phones">
22 <!ENTITY CH "Charger">
23 <!ENTITY SK "Starters Kit">
```

Declaring Attributes 1-2

Syntax

```
<!ATTLIST element-name attribute-name attribute-type default-value>
```

where,

`element-name` is the element the attribute belongs

`attribute-name` is the name of the attribute

`attribute-type` is type of data the attribute can accept

`default-value` is the default value for the attribute

Declaring Attributes 2-2

Value	Description
PCDATA	Parsed character data
CDATA	Character data
(en1|en2|..)	Enumerated list
ID	A unique id
IDREF	Id of another element
IDREFS	List of other ids
NMTOKEN	Valid XML name
NMTOKENS	List of valid XML names
ENTITY	An entity
ENTITIES	List of entities
NOTATION	Name of a notation
xml:	Predefined xml value

Specifying Attribute Values 1-3

Value	Description
`value`	Default value
`#REQUIRED`	Value must be included
`#IMPLIED`	Value does not have to be included
`#FIXED`	Value is fixed
`en1|en2|...`	Listed enumerated values

Specifying Attribute Values 2-3

Syntax

- #IMPLIED

```
<!ATTLIST element-name attribute-name attribute-type #IMPLIED>
```

- #REQUIRED

```
<!ATTLIST element-name attribute-name attribute-type #REQUIRED>
```

- #FIXED

```
<!ATTLIST element-name attribute-name attribute-type #FIXED "value">
```

- Enumerated Attribute Values

```
<!ATTLIST element-name attribute-name
(en1|en2|..) default-value>
<!ATTLIST payment type (check|cash) "cash">
```

Specifying Attribute Values 3-3

Code Snippet

- Default Value

```
<!ATTLIST Model Type CDATA "Camera">
```

- #IMPLIED

```
<!ATTLIST Model Type CDATA "Camera" #IMPLIED>
```

- #REQUIRED

```
<!ATTLIST Model Type CDATA #REQUIRED>
```

- #FIXED

```
<!ATTLIST Model Type CDATA #FIXED "Camera">
```

- Enumerated Attribute Values

```
<!ATTLIST Model Type (Camera|Bluetooth) "Camera">
```

Entities in DTD 1-2

- It is a placeholder that consists of a name and a value.
- It is declared once and then repeatedly used through out the document.

Syntax

➢Entity declaration:

```
<!ENTITY entity-name "entity-value">
```

➢Entity Reference:

```
&entity-name;
```

Entities in DTD 2-2

Code Snippet

```
<!DOCTYPE Mobile [
   <!ELEMENT Mobile (Company, Model, Price, Accessories)>
     <!ELEMENT Company (#PCDATA)>
     <!ELEMENT Model (#PCDATA)>
     <!ELEMENT Price (#PCDATA)>
     <!ELEMENT Accessories (#PCDATA)>
   <!ATTLIST Model Type CDATA "Camera">
   <!ENTITY HP "Head Phones">
   <!ENTITY CH "Charger">
   <!ENTITY SK "Starters Kit">
]>
<Mobile>
   <Company> Nokia </Company>
   <Model Type="Camera"> 6600 </Model>
   <Price> 9999 </Price>
   <Accessories> &HP;, &CH; and a &SK; </Accessories>
</Mobile>
```

Kinds of Entity Declarations 1-4

Internal Entity Declaration

- The entity value is explicitly mentioned in the entity declaration.

Syntax

```
<!ENTITY entity-name "entity-value">
```

Kinds of Entity Declarations 2-4

Code Snippet

```
<!DOCTYPE Mobile [
<!ELEMENT Mobile (Company, Model, Price, Accessories)>
<!ELEMENT Company (#PCDATA)>
<!ELEMENT Model (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
<!ELEMENT Accessories (#PCDATA)>
<!ATTLIST Model Type CDATA “Camera”>
<!ENTITY HP “Head Phones”>
<!ENTITY CH “Charger”>
<!ENTITY SK “Starters Kit”>
]>
<Mobile>
<Company> Nokia </Company>
<Model Type=”Camera”> 6600 </Model>
<Price> 9999 </Price>
<Accessories> &HP;, &CH; and a &SK; </Accessories>
</Mobile>
```

Kinds of Entity Declarations 3-4

External Entity Declaration

- A link or path to the entity value is mentioned in place of the entity value

Kinds of Entity Declarations 4-4

Syntax

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

```
<!DOCTYPE Mobile [
   <!ELEMENT Mobile (Company, Model, Price, Accessories)>
   <!ELEMENT Company (#PCDATA)>
   <!ELEMENT Model (#PCDATA)>
   <!ELEMENT Price (#PCDATA)>
   <!ELEMENT Accessories (#PCDATA)>
   <!ATTLIST Model Type CDATA "Camera">
   <!ENTITY HP SYSTEM "hp.txt">
   <!ENTITY CH SYSTEM "ch.txt">
   <!ENTITY SK SYSTEM "sk.txt">
]>
<Mobile>
<Company> Nokia </Company>
<Model Type="Camera"> 6600 </Model>
<Price> 9999 </Price>
<Accessories> &HP;, &CH; and a &SK; </Accessories>
</Mobile>
hp.txt
Head Phones
ch.txt
Charger
sk.txt
Starters Kit
```

Summary 1-2

- **Document Type Definition**
  - A DTD is a non XML document made up of element, attribute and entity declarations.
- **Working with DTDs**
  - The DTD structure is composed of element declarations, attribute declarations, and entity declarations.
  - A document type declaration declares that the XML file in which it is present adheres to a certain DTD.

Summary 2-2

- **Valid XML Documents**
  - A well-formed XML document adheres to the basic XML syntax rules.
  - A valid XML document is a well-formed XML document that adheres to its DTD.
- **Declarations**
  - XML elements are declared with an element declaration in the DTD.
  - An entity is a placeholder that consists of a name and a value.