DTD - ENTITIES

Entities are used to define shortcuts to special characters within the XML documents. Entities can be primarily of four types:

- Built-in entities
- Character entities
- General entities
- Parameter entities

Entity Declaration Syntax

In general, entities can be can be declared ***internally*** or ***externally***. Let us understand each of these and their syntax as follows:

Internal Entity

If an entity is declared within a DTD it is called as internal entity.

Syntax

Following is the syntax for internal entity declaration:

```
<!ENTITY entity_name "entity_value">
```

In the above syntax:

- **entity_name** is the name of entity followed by its value within the double quotes or single quote.
- **entity_value** holds the value for the entity name.
- The entity value of the Internal Entity is de-referenced by adding prefix **&** to the entity name *i.e. &entity_name.*

Example

Following is a simple example for internal entity declaration:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE address [
<!ELEMENT address (#PCDATA)>
<!ENTITY name "Tanmay patil">
<!ENTITY company "TutorialsPoint">
<!ENTITY phone_no "(011) 123-4567">
]>
<address>
   &name;
   &company;
   &phone_no;
</address>
```

In the above example, the respective entity names *name*, *company* and *phone_no* are replaced by their values in the XML document. The entity values are de-referenced by adding prefix **&** to the entity name.

Save this file as **sample.xml** and open it in any browser, you will notice that the entity values for *name*, *company*, *phone_no* are replaced respectively.

External Entity

If an entity is declared outside a DTD it is called as external entity. You can refer to an external Entity by either using system identifiers or public identifiers.

Syntax

Following is the syntax for External Entity declaration:

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

In the above syntax:

- **name** is the name of entity.
- **SYSTEM** is the keyword.
- **URI/URL** is the address of the external source enclosed within the double or single quotes.

Types

You can refer to an external DTD by either using:

- **System Identifiers -** A system identifier enables you to specify the location of an external file containing DTD declarations. Syntax is as follows:

  ```
  <!DOCTYPE name SYSTEM "address.dtd" [...]>
  ```

  As you can see it contains keyword SYSTEM and a URI reference pointing to the document's location.

- **Public Identifiers -**

  Public identifiers provide a mechanism to locate DTD resources and are written as below:

  ```
  <!DOCTYPE name PUBLIC "-//Beginning XML//DTD Address Example//EN">
  ```

  As you can see, it begins with keyword PUBLIC, followed by a specialized identifier. Public identifiers are used to identify an entry in a catalog. Public identifiers can follow any format; however, a commonly used format is called *Formal Public Identifiers, or FPIs.*

Example

Let us understand the external entity with the following example:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE address SYSTEM "address.dtd">

<address>
   <name>Tanmay Patil</name>
   <company>TutorialsPoint</company>
   <phone>(011) 123-4567</phone>
</address>
```

Below is the content of the DTD file *address.dtd*:

```
<!ELEMENT address (name, company, phone)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT company (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
```

Built-in entities

All XML parsers must support built-in entities. In general, you can use these entity references anywhere. You can also use normal text within the XML document, such as in element contents and attribute values.

There are five built-in entities that play their role in well-formed XML, they are:

- ampersand: &amp;
- Single quote: &apos;
- Greater than: &gt;
- Less than: &lt;
- Double quote: &quot;

Example

Following example demonstrates the built-in entity declaration:

```
<?xml version="1.0"?>
<note>
   <description>I'm a technical writer & programmer</description>
<note>
```

As you can see here the &amp; character is replaced by & whenever the processor encounters this.

Character entities

Character Entities are used to name some of the entities which are symbolic representation of information i.e characters that are difficult or impossible to type can be substituted by Character Entities.

Example

Following example demonstrates the character entity declaration:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE author[
<!ELEMENT author (#PCDATA)>
<!ENTITY writer "Tanmay patil">
<!ENTITY copyright "&#169;">
]>
<author>&writer;&copyright;</author>
```

You will notice here we have used **&#169;** as value for copyright character. Save this file as *sample.xml* and open it in your browser and you will see that copyright is replaced by the character ©.

General entities

General entities must be declared within the DTD before they can be used within an XML document. Instead of representing only a single character, general entities can represent characters, paragraphs, and even entire documents.

Syntax

To declare a general entity, use a declaration of this general form in your DTD:

```
<!ENTITY ename "text">
```

Example

Following example demonstrates the general entity declaration:

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ENTITY source-text "tutorialspoint">
]>

<note>
&source-text;
</note>
```

Whenever an XML parser encounters a reference to *source-text* entity, it will supply the replacement text to the application at the point of the reference.

Parameter entities

The purpose of a parameter entity is to enable you to create reusable sections of replacement text.

Syntax

Following is the syntax for parameter entity declaration:

```
<!ENTITY % ename "entity_value">
```

- *entity_value* is any character that is not an '&', '%' or ' " '.

Example

Following example demonstrates the parameter entity declaration. Suppose you have element declarations as below:

```
<!ELEMENT residence (name, street, pincode, city, phone)>
<!ELEMENT apartment (name, street, pincode, city, phone)>
<!ELEMENT office (name, street, pincode, city, phone)>
<!ELEMENT shop (name, street, pincode, city, phone)>
```

Now suppose you want to add additional eleement *country*, then then you need to add it to all four declarations. Hence we can go for a parameter entity reference. Now using parameter entity reference the above example will be :

```
<!ENTITY % area "name, street, pincode, city">
<!ENTITY % contact "phone">
```

Parameter entities are dereferenced in the same way as a general entity reference, only with a percent sign instead of an ampersand:

```
<!ELEMENT residence (%area;, %contact;)>
<!ELEMENT apartment (%area;, %contact;)>
<!ELEMENT office (%area;, %contact;)>
<!ELEMENT shop (%area;, %contact;)>
```

When the parser reads these declarations, it substitutes the entity's replacement text for the entity reference.