C# XML

321190
2015년 가을학기
12/4/2015
박경신

XML (eXtensible Markup Language)

- XML 기본
  - Meta Language – 원래 데이터에 대해 추가적인 정보를 표시
  - Element와 Contents로 구성
    - Element는 문서의 구조를 정의하는 요소. 시작과 끝 태그(Tag)를 사용하여 표시
    - Contents는 실제 데이터
  - XML 파서(Parser)가 필요함
    - www.w3.org/XML에서 정의한 구문규칙을 사용
    - 대표적인 파서로 ms XML Core Services(MSXML)
  - XML 문서는 선택적으로 문서의 구조를 정의하는 DTD (Document Type Definition) 또는 스키마 (Schema)를 참조가능

XML 구성요소

- XML 구성요소
  - XML 문서의 선언
  - **Element** – 마크업 태그와 그 안에 포함된 내용
  - **Root Element** – 문서 내 모든 Element와 내용을 포함하고 있는 XML 문서의 요소
  - **Attribute** – Element에 포함되어 추가적인 정보를 제공
  - **Entity** – 텍스트, Binary, 비 ASCII 문자를 저장하는 데 사용
  - **Processing Instruction** – 전체 문서나 문서의 일부를 처리하는 응용프로그램과 연결해주는 명령어
  - **Comment** (주석) – XML 프로세서가 해석하지 않는 설명문
  - **CDATA Section** – 모든 문자를 마크업이나 태그로 인식하지않고 일반 문자로 인식할 수 있는 표기법. 즉, 특수한 문자를 일반 텍스트로 인식하도록 하는 표기법
  - **DTD** 선언

XML 문서의 선언

- Parser에게 현재 XML 문서가 어떤 문서인지 알려 줄 수 있도록 몇 가지 정보를 제공
  - **<?xml version="1.0" encoding="EUC-KR"?>**
  - XML 문서의 선언은 반드시 파일의 맨 처음에 위치
  - Version 속성 필수
  - Encoding 속성 선택
    - 한글 encoding은 "EUC-KR"을 주로 사용
  - Standalone 속성 선택
    - "yes" – 다른 파일에 의존하고 있지 않음. 외부 파일의 요소, 그림, 개체 등을 참조하고 있지 않음
    - "no" – 다른 파일에 의존하고 있을 수 있음 (default)
  - Version, encoding, standalone 순서 준수

XML Element

- Root Element
  - 모든 Element를 포함하는 Element
  - 반드시 1개 존재함
- Element content
  - 시작 태그와 끝 태그 사이의 내용
  - PCDATA (Parsed Character DATA)
  - 다른 Element
- Element 작성 규칙
  - 모든 시작 태그는 반드시 끝 태그와 짝을 이루어야 하며, 겹쳐 쓸 수 없음
  - XML Document는 반드시 하나의 Root Element를 가져야 함
  - Element 이름은 반드시 XML의 이름 짓는 규칙에 따라야 함. 문자로 시작 첫 문자 뒤에는 "_", ".", 숫자 사용 가능
  - XML은 대소문자를 구별함

XML Element

- public class Person {

```
[XmlElement("Name")]
public string Name   {
   get;   set;
}
[XmlElement("ID")]
public int ID   {
   get;      set;
} ..... }
```

- **<Person>**
  <Name>Park **</Name>**
  <ID>1207**</ID>**
  <Phone> 000-111-2222**</Phone>**
  <Address>단국대학교**</Address>**
  </Person>

XML Attribute

- Attribute는 하나의 요소와 결합된 이름/값의 쌍
- **시작 태그에 추가**
- **속성은 반드시 값을 가짐**

```
<movie genre="SF">
    <name>2001: Space Odyssey</name>
    <director>Standley Kubrick</director>
    <year>1968</year>
</movie>
```

XML Attribute

- public class Person {

```
[XmlAttribute(AttributeName="Name")]
public string Name   {
   get;         set;
}
[XmlAttribute(AttributeName="ID")]
public int ID   {
   get;      set;
} ..... }
```

- **<Person Name="**Park**" ID="**1207**" Phone="**000-111-2222**" Address="**단국대학교**" />**

XML Element & Attribute

```
[XmlRoot("Person")]
public class Person {
  [XmlElement("Name")]
   public string Name {
      get;   set;
   }
   [XmlElement("ID")]
   public int ID   {
      get;      set;
   }
   [XmlElement("Phone")]
   public Phone phone = new
Phone();
   [XmlElement("Address")]
   public string Address   {
      get;      set;
   } ..... }
```

```
public class Phone {
  [XmlElement("WorkPhone")]
   public string WorkPhone {
      get;   set;
   }
   [XmlElement("HomePhone")]
   public string HomePhone   {
      get;      set;
   }
   [XmlElement("CellPhone")]
   public string CellPhone   {
      get;      set;
   }
}
```


XML Element & Attribute

```
<Person>
   <Name>Park </Name>
   <ID>1207</ID>
   <Phone>
        <WorkPhone>041-550-3490</WorkPhone>
        <HomePhone>02</HomePhone>
        <CellPhone>010</CellPhone>
   </Phone>
   <Address>단국대학교</Address>
</Person>
```


XML Element & Attribute

```
[XmlRoot("Person")]
public class Person {
   [XmlElement("Name")]
   public string Name {
      get;   set;
   }
   [XmlElement("ID")]
   public int ID   {
      get;      set;
   }
   [XmlElement("Phone")]
   public Phone phone = new
Phone();
   [XmlElement("Address")]
   public string Address   {
      get;      set;
   } ..... }
```

```
public class Phone {
[XmlAttribute(AttributeName=
"WorkPhone")]
   public string WorkPhone {
      get;   set;
   }
[XmlAttribute(AttributeName=
"HomePhone")]
   public string HomePhone   {
      get;      set;
   }
[XmlAttribute(AttributeName=
"CellPhone")]
   public string CellPhone   {
      get;      set;
   }
}
```


XML Element & Attribute

```
<Person>
   <Name>Park </Name>
   <ID>1207</ID>
   <Phone WorkPhone="041-550-3490"
HomePhone="02" CellPhone="010" />
   <Address>단국대학교</Address>
</Person>
```

Blank Element

- Element의 내용이 없는 경우 끝 태그를 생략하고 빈 요소를 사용할 수 있음
- 빈 요소는 어떠한 내용도 포함할 수 없으며, **오직 속성만을 포함**
  - **<Person/> (O)**
  - **<Person /> (O)**
  - **<Person/ > (X)**
  - **<Person / > (X)**
  - **<Person Name="Park" ID="1207" Phone="000-111-2222" Address="단국대학교"/> (O)**

XML Comment

- <!- - 와 - -> 사이에 정의되고 주석처리 (Comment)로 사용
- **<!-- <Person Name="Park" ID="1207" Phone="000-111-2222" Address="단국대학교"/>-->**
- 태그 내에 주석을 쓸 수 없음
- 주석 내에 "--"를 쓸 수 없음

Processing Instruction

- 처리명령 (Processing Instruction)
  - <? 와 ?> 사이에 표현되고 특정 응용프로그램에 대해 처리할 정보나 명령을 가리키는 역할
  - **<?xml:stylesheet type="text/xsl" href="메모.xsl"?>**
    - XML 의 스타일 쉬트를 지정하고, 타입은 test/xsl이며, 소스는 메모.xsl에 있다는 의미
  - Element나 Attribute 이외에 Application에 전달하고자 하는 내용을 사용
  - XML 문서의 주석문은 프로세서에 전달되지 않기 때문에 JavaScript 코드 등을 삽입
  - Nameprocessor – Application의 이름 (PITarget)
    - "SELECT * FROM customer" : 실제 PI

PCDATA

- Element에 포함된 문자열
- 유효하지 않은 PCDATA 문자들
  - "<", "&" 등
  - 해결 방법
    - Escape 문자
    - CDATA section
- Entity Reference
  - & 문자 - &
  - < 문자 - <
  - > 문자 - >
  - ' 문자 - '
  - " 문자 - "

Entity

- Entity의 단위는 글자에서부터 문서 전체
- 외부 참조
  - 포함 시킬 내용이 외부에 존재하는 것을 참조하는 형태
- 내부 참조
  - DTD 내에 정의
- Entity에 대한 포인터는 Entity Reference
  - & 문자 - &
  - < 문자 - <
  - > 문자 - >
  - ' 문자 - '
  - " 문자 - "

CDATA Section

- Parser나 브라우저에서 유효성 검사를 하지 않는 문자열
- CDATA 섹션 (Character DATA Section)
  - <![CDATA[와]]> 사이에 정의되고 그 안의 내용을 문자열로 사용
  - **<![CDATA[<Person Name="**Park" **ID="**1207" **Phone="**000-111-2222" **Address="**단국대학교"**/>]]>**

DTD 선언

- Document Type Definition
- XML 문서의 구조를 명시적으로 정의한 것
- Valid Document (유효한 문서)
  - Well-formed document이면서 DTD에서 정의된 규칙을 따르는 XML 문서

Namespace

- 여러 개의 문서를 병합할 때 같은 이름의 Element나 Attribute을 구별하기 위한 방법
  - Schema의 재사용
  - 여러 문서와의 통합 시 이름 충돌 문제 해결
- Namespace 선언
  - **<접두어: 태그이름 xmlns:접두어="URL">**
  - **Name은 URI에 의해 구별**
  - 선언된 태그와 그 자식 태그에서 접두어를 사용할 수 있음
  - 하나의 태그에서 여러 개의 namespace를 선언할 수 있음
  - 주로 Root Element에서 선언
- 기본 Namespace 선언
  - **<태그이름 xmlns="URL">**
  - 기본 Namespace를 선연하면 접두어를 붙이지 않아도 해당 Element와 모든 자식 Element에 Namespace가 적용