1. **XML.NET**

**XML**(Extensible Markup Language)은 W3C에서 관장하는 확장 가능한 마크업 언어 이다.  
XML은 주로 다른 시스템간 데이터를 쉽게 주고 받을 수 있게 하여 데이터 표준화에 사용 할 수 있다.

**<선언부>**

<?xml로 시작하여 ?>끝남 -🡪처리 구문  
**<주석>**  
<!—로 시작하여--!>로 표현한다.

예제, 기본형)  
<?xml version=”1.0” encoding=”euc-kr” standalone=”yes”?>--🡪선언문  
<회원정보> ---🡪 요소 시작

<회원아이디>ehclub</회원아이디>  
 <회원이름>장문석</회원이름>

<회원아이디>jejutour</회원아이디>  
 <회원이름>송정수</회원이름>  
</회원정보>--🡪 요소의 끝

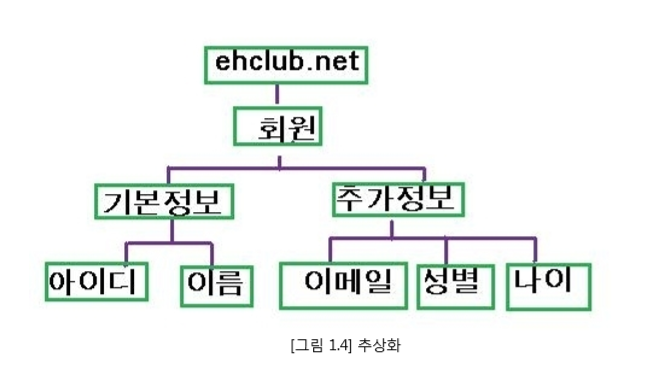
**<<Escape 문자.(이스케이프 문자)>>**

<?xml version="1.0" encoding="euc-kr" standalone="yes" ?>  
 <회원정보>  
 <회원아이디>&lt;ehclub&gt;</회원아이디> 🡪<회원아이디><ehclub></회원아이디>  
 <회원이름>&quot;장문석&quot;</회원이름>🡪<회원이름>’장문석’</회원이름>  
 <회원아이디>&lt;jejutour&gt;</회원아이디>🡪<회원아이디><jejutour></회원아이디>  
 <회원이름>&quot;송정수&quot;</회<회원원이름> 🡪<회원아이디>’송정수’</회원아이디>  
<!-- 참고로 & 표시는 &amp;로 하며 ' vytlsms &apos; 로 한다. --> </회원정보>

|  |  |  |
| --- | --- | --- |
| **문자** | **표현** |  |
| < | &lt; |  |
| >; | &gt |  |
| ‘ | &aqos; |  |
| “ | &quot; |  |
| & | &amp; |  |

escape 문자를 포함한 XML 문서

**<<추상화>>**

  
**<<<추상화한 걸 XML문서화>>>**  
└ 특성을 표현할 때는 요소 태그 내부에 특성명="특성값" 형태로 표현합니다. <회원 나이="38" 성별="남">홍길동</회원>

<?xml version="1.0" encoding="euc-kr" standalone="yes" ?>

<회원정보>   
<회원>  
<기본정보>  
<회원아이디>ehclub</회원아이디>

<회원이름>장문석</회원이름>  
</기본정보>

<추가정보>   
<이메일>jejutour@daum.net</이메일>   
<나이>38</나이>   
<성별>남</성별>   
</추가정보>

</회원>   
</회원정보>

**1.2 DTD**

특정 시스템 내부에서 자신의 시스템에서 사용할 수 있는 유효한 XML 문서 구조를 정의할 때 DTD를 사용 할 수 있습니다.  
XML 문서를 표준 문서 포멧으로 추상화하는 것을 말합니다.

**<<기본형>>**

DTD 문서는 <!DOCTYPE DTD 명 [내부 요소들]> 형태로 표시합니다.

<?xml version="1.0" encoding="euc-kr" ?>   
<!-- DTD 정의 --> 🡪주석  
<!DOCTYPE 회원 [ <!ELEMENT 회원리스트 (회원)> <!ELEMENT 회원 (아이디,이름,이메일)> <!ELEMENT 아이디 (#PCDATA)>  
 <!ELEMENT 이름 (#PCDATA)>  
 <!ELEMENT 이메일 (#PCDATA)> ]>  
 <!-- DATA 정의 -->  
 <회원리스트>  
 <회원>  
 <아이디>ehclub</아이디>  
 <이름>장문석</이름>  
 <이메일>jejutour@daum.net</이메일>  
 </회원>

└다음은 XML문서와 DTD문서를 분리한 예입니다.

<?xml version="1.0" encoding="euc-kr" ?>  
 <!DOCTYPE 회원 SYSTEM "회원.dtd">  
 <회원리스트>  
 <회원>  
 <아이디>ehclub</아이디>  
 <이름>장문석</이름>  
 <이메일>jejutour@daum.net</이메일>  
 </회원>

**<<규격에 유효한 XML 문서 >>**

<?xml version="1.0" encoding="euc-kr" ?>  
 <!ELEMENT 회원리스트 (회원)>  
 <!ELEMENT 회원 (아이디,이름,이메일)>  
 <!ELEMENT 아이디 (#PCDATA)>  
 <!ELEMENT 이름 (#PCDATA)>  
 <!ELEMENT 이메일 (#PCDATA)>

**<<XML 문서 외부에 DTD 표현>>**

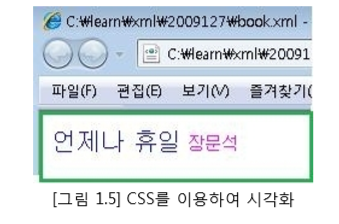
XML 문서는 외형적인 모습은 표현하지 않고 내용만을 표현합니다. 만약 외형적인 모습을 표현할 때는 CSS 를 이용하세요.

<?xml version="1.0" encoding="euc-kr" standalone="yes"?>  
 <?xml:stylesheet type="text/css" href="book.css"?>  
 <book>  
 <bookname>언제나 휴일</bookname>  
 <writer>장문석</writer>  
 </book>

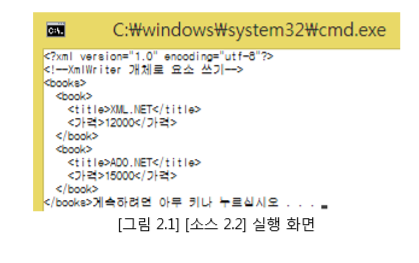
Bookname{  
 font-family:  
 "돋음체";  
 font-size:20px;  
 color:#0000ff;   
}

writer{  
 font-family:  
 "궁서체";  
 font-size:14px;  
 color:#ff00ff;  
 }  
 .codetype{  
 font-family:  
 "궁서체";  
 font-size:14px;  
 color:#00ff00   
}

**<<Book.css 파일>>**



.

****

static void Main(string[] args)   
{

XmlWriterSettings settings = new XmlWriterSettings();  
settings.Indent = true;  
XmlWriter writer = XmlWriter.Create("data.xml", settings);

writer.WriteComment("XmlWriter 개체로 요소 쓰기");  
 writer.WriteStartElement("books"); //루트 요소 쓰기

writer.WriteStartElement("book"); 🡪 book 요소 쓰기  
 writer.WriteStartElement("title"); 🡪 title 요소 쓰기

writer.WriteName("XML.NET");   
writer.WriteEndElement(); 🡪 title 요소 닫기  
writer.WriteStartElement("가격"); 🡪 가격 요소 쓰기   
writer.WriteValue(12000);   
writer.WriteEndElement(); 🡪 가격 요소 닫기  
writer.WriteEndElement();//book 요소 닫기

writer.WriteStartElement("book"); 🡪 book 요소 쓰기  
writer.WriteElementString("title","ADO.NET"); 🡪 title 요소와 값 쓰기  
writer.WriteStartElement("가격"); 🡪 가격 요소 쓰기

writer.WriteValue(15000);  
writer.WriteEndElement(); 🡪 가격 요소 닫기  
writer.WriteEndElement(); 🡪 book 요소 닫기   
writer.WriteEndElement(); 🡪 루트 요소 닫기  
writer.Close();

XmlReader xreader = XmlReader.Create("data.xml"); 🡪 XmlReader 개체 생성  
XmlWriter xwriter = XmlWriter.Create(Console.Out); 🡪 콘솔 출력 스트림으로 XmlWriter 개체 생성  
xwriter.WriteNode(xreader, false); 🡪 xreader 개체로 읽어온 데이터를 xwriter 개체에 복사  
xwriter.Close();  
xreader.Close();

}