Урок 2

План заняття:

1. Введення в DTD (Document Type Definition)

На минулій парі ми з Вами говорили про те, що кожен XML-документ повинен бути не тільки **дійсним**, тобто бути синтаксично вірним, але й **коректним**, тобто відповідати певним правилам побудови документу (вкладеність та кількість елементів, порядок слідування, типи та кількість атрибутів тощо). Як зробити XML-документ дійсним ми вже розібрали і сьогодні перейдемо до другої частини – почнемо розглядати як зробити його коректним. І почнемо ми з того, як логічно побудувати XML-документ за допомогою **DTD (Document Type Definition - оголошення типів документу)**.

2. Опис структури XML документа за допомогою DTD. Декларація типу документа

Оголошення або декларація типу документа складається з одного або кількох правил-обмежень структури документа. Але для того, щоб ці правила можна було застосовувати в основному XML-документі, необхідно вказати декларацію DTD-документа, який описує ці правила. Така декларація має наступний вигляд:

```
<!DOCTYPE кореневий_елемент .....>
```

Ключове слово DOCTYPE вказує на те, що це DTD декларація. Після нього йде ім'я документа, якого стосуються описані правила. Не слід забувати про те, що ім'ям документа є назва його кореневого елемента. Наприклад:

```
<!DOCTYPE notebook>
```

Далі можливі три варіанти розвитку подій і, щоб добре їх зрозуміти, розглянемо кожен з них окремо.

Перший спосіб полягає в тому, що визначення правил задається в самому XML-документі, тобто **внутрішнє оголошення**.

```
<!DOCTYPE кореневий_елемент [опис_правил]>
```

Наприклад:

```
<!DOCTYPE notebook [ .......... ]>
```

Другий спосіб визначення DTD-декларації використовується при описі правил в зовнішньому файлі:

```
<!DOCTYPE кореневий_елемент
          PUBLIC | SYSTEM
          "id" | "filename">
```

Ключове слово **PUBLIC** задається, якщо кореневий елемент загальновідомий, тобто описаний в DTD-документі, який має відкриту затверджену URL-адресу. Наприклад, це може бути документ, який містить опис певного стандарту. Після вказання кореневого елемента і ключового слова PUBLIC в лапках вказується його відомий ідентифікатор. Цей ідентифікатор фактично задає інформацію про даний ресурс. Наприклад, документи на мові XHTML, повинні починатись з оголошення:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Рядок ідентифікатора розшифровується наступним чином:

- ✓ Якщо DTD являється стандартом ISO і був офіційно прийнятий групою стандартизації, тоді вказується символ (+), інакше символ (-).
- ✓ Далі вказується фірма-розробник. В даному прикладі це консорціум W3C.
- ✓ Ключове слово, яке вказує на тип інформації в файлі. Найчастіше використовуються наступні ключові слова: DTD, ELEMENT та TEXT. DTD використовується лише для файлів DTD, ELEMENT зазвичай використовується для фрагментів DTD, які містять лише об'єкти або оголошення елемента. TEXT – для вмісту SGML (текст і мітки). Оскільки ми описуємо DTD-документ, тоді вказуємо ключове слово "DTD".
- ✓ Опис вмісту файла. Як правило, це йде назва стандарту та номер версії.
- ✓ В самому кінці вказується мова документа – EN (англійська). Мова позначається двохсимвольним кодом стандарта ISO.

Після вказання такого рядка, програма, яка обробляє XHTML-документ з таким заголовком (зокрема браузер), зможе по ідентифікатору зрозуміти, що документ створений на мові XHTML і обробляти його потрібно згідно стандарту XHTML 1.0 Transitional. Якщо ж програма такого стандарту незнає, тоді вона може завантажити оголошення типів по адресі "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd". DTD-документ належить консорціуму W3C, написаний на англійській мові та не стандартизований офіційно.

Існує можливість визначити свій власний набір правил. В такому разі потрібно скористатись ключовим словом **SYSTEM**, після якого в лапках необхідно вказати шлях до файла DTD.

```
<!DOCTYPE notebook SYSTEM "notebook.dtd">
```

Останній спосіб – змішаний та виглядає наступним чином:

```
<!DOCTYPE кореневий_елемент
          PUBLIC | SYSTEM
          "filename" | "id"
          [опис_правил]>
```

Такий спосіб опису правил використовується здебільшого для того, щоб розширити опис зовнішнього DTD-документа новими правилами, які притаманні лише даному XML-документу.

З оголошенням розібрались, а тепер давайте розберемось орієнтовно з синтаксисом такого DTD документу, а він виглядає досить незвично. Сам документ складається з оголошень розмітки (markup declaration), які починаються з пари символів **<!**. Після цих символів задаються правила опису та використання, які задаються за допомогою наступних **ключових слів**:

- ✓ **ELEMENT** – визначення елемента;
- ✓ **ATTLIST** – визначення списка атрибутів;
- ✓ **ENTITY** – визначення сутностей;
- ✓ **NOTATION** – визначення нотацій.

3. Опис елемента

В документі XML повинен бути описаний кожен елемент. Для його опису використовується наступний запис:

```
<!ELEMENT назва_елемента опис_елемента>
```

Як видно з синтаксису оголошення елемента починається з символів <!ELEMENT, після яких йде назва елемента та його опис. По своєму вмісту елементи XML-документа поділяються на чотири групи. В залежності від цього ви можете скористатись одним з **чотирьох варіантів опису елемента**:

1. **EMPTY** - пустий елемент, тобто елемент може мати атрибути, але не може містити текст або дочірні елементи.
2. **ANY** - будь-що в суміші, тобто елемент може мати довільний вміст.
3. **(#PCDATA) -** елемент може містити лише звичайний текст.
4. **(модель вмісту)** – елемент може містити лише дочірні елементи, які через кому перелічуються в дужках. При цьому, вказаний порядок елементів в XML-документів повинен зберігатись. Також існує можливість задати частоту появи того чи іншого елемента за допомогою наступних квантифікаторів та символів:
   - , - перераховані елементи в заданому порядку;
   - | - вибір одного варіанту з можливих;
   - ( ) – використовується для групування;
   - ? - вказаний елемент не є обов'язковим;
   - * - елемент може не зустрічатись або зустрічатись довільну кількість разів;
   - + - елемент може зустрічатись один або більше разів.

Наприклад:

```
<!ELEMENT notebook ANY>          <!-- елемент може містити довільні дані -->
<!ELEMENT person ANY>            <!-- елемент може містити довільні дані -->
<!ELEMENT name (#PCDATA)>        <!-- елемент може містити лише текст -->
<!ELEMENT имя (#PCDATA)>         <!-- елемент може містити лише текст -->
<!ELEMENT middlename (#PCDATA)><!-- елемент може містити лише текст -->
<!ELEMENT surname (#PCDATA)>     <!-- елемент може містити лише текст -->
<!ELEMENT phone (#PCDATA)>       <!-- елемент може містити лише текст -->
<!ELEMENT br EMPTY>              <!-- елемент буде пустим -->

<!-- елемент record0 може містити дочірні елементи name, surname і phone. При цьому
елемент name зустрічається мінімум один раз, surname тільки один раз, а phone може не
бути взагалі або бути кілька -->
<!ELEMENT record0 ( name+, surname, phone* ) >

<!-- елемент record1 може містити дочірні елементи name, surname або phone. При цьому
елемент name зустрічається мінімум один раз, якщо в елементі присутній surname, то він
буде лише один раз, а якщо phone, тоді їх може не бути взагалі або бути кілька -->
<!ELEMENT record1 ( name+, (surname | phone*) )>

<!-- елемент record2 може містити дочірні елементи name, surname або phone. При цьому
елемент name зустрічається мінімум один раз, а елементи surname або phone можуть бути
відсутніми або зустрічатись один раз -->
<!ELEMENT record2 ( name+, (surname | phone)? )>

<!-- елемент record3 може містити текст або дочірний елемент name. При цьому кожен з них
може бути відсутнім або зустрічатись довільну кількість разів -->
<!ELEMENT record3 (#PCDATA | name)*>

<!-- елемент fullname може містити дочірні елементи name або имя, middlename і surname.
При цьому елемент name та имя може зустрічатись лише один раз і тільки однин з них.
Елемент middlename може бути відсутнім взагалі або існувати в одному екземплярі, а
surname повинен обов'язково бути присутнім -->
<!ELEMENT fullname ((name | имя), middlename?, surname)>
```

Для прикладу напишемо DTD-документ, який описує правила побудови записної книжки:

notebook.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT notebook (person)+>
<!ELEMENT person ANY>
<!ELEMENT fullname EMPTY>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT text (#PCDATA)>
```

notebook.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE notebook SYSTEM "notebook.dtd">
<notebook>
      <person>
            <fullname/>
            <phone>22-22-22</phone>
            <city>Zmerinka</city>
      </person>
      <person>
            <fullname />
            <phone>333-33-33</phone>
            <city>Kyev</city>
      </person>
</notebook>
```

4. Опис списку атрибутів

Після того, як оголосили елементи можна описати їх атрибути. Оскільки у елемента може бути атрибутів, то вони організовуються в список. Згідно специфікації, значення атрибутів поділяються на 3 **типи**:

1. **рядкові** – довільний рядок, тобто значення атрибута може бути довільним рядком символів;
2. **перелічувані** – список значень. До цієї групи відносяться типи атрибутів, значення яких являються значеннями з певного фіксованого набору. Прикладом такого типу атрибута може бути атрибут CSS border-width, який може приймати одне з рядкових значень: thick, thin або medium;
3. **марковані, токени** або **іменовані** – рядок фіксованого розміру. До них відносяться типи, які мають певні лексичні обмеження на значення.

Синтаксис оголошення списку отрибутів має наступний вигляд:

```
<!ATTLIST елемент
          атрибут1 тип1 характеристика1
          ............
          атрибутN типN характеристикаN>
```

Тобто список атрибутів починається з символів <!ATTLIST, після яких йде назва елемента, який буде мати атрибути. Далі йде опис атрибутів: назва, тип та їх характеристика. Кількість атрибутів, які може мати елемент необмежена.

Правила іменування атрибутів є такими ж як і для елементів. Щодо типу, то атрибут може мати один з наступних **типів**:

- ✓ CDATA – символьні дані;
- ✓ [перелік_значень] – атрибут може приймати одне з кількох значень, які розділені пропусками або символами (|);
- ✓ ID – позначає унікальний ідентифікатор елемента;
- ✓ IDREF – посилання на унікальний ідентифікатор;
- ✓ IDREFS – набір посилань через пропуск (space);
- ✓ ENTITY – сутність;
- ✓ ENTITIES – набір сутностей через пропуск (space);
- ✓ NMTOKEN – іменований токен, тобто name елемента. Фактично це аналог атрибута ID, але унікальність якого не вимагається;
- ✓ NMTOKENS – набір іменованих токенів через пропуск (space);
- ✓ NOTATION – нотація.

Всі типи, крім перших двох являються маркованими.

Характеристика визначає те, як значення може бути присвоєно атрибуту та може приймати **одне з значень**:

- #REQUIRED – обов'язковий атрибут;
- #IMPLIED – необов'язковий атрибут;
- #FIXED "значення" – атрибут може приймати тільки значення, яке вказується біля нього через пропуск;
- "значення" – задає значення по замовчуванню.

Наприклад:

```
<!-- елемент має обов'язковий атрибут reg-num, який виступає в якості ідентифікатора -->
<!ATTLIST name reg-num ID #REQUIRED>

<!-- елемент name має обов'язкові атрибути first та surname, та необов'язковий middle,
які можуть містити будь-які символьні дані -->
<!ATTLIST name
          first   CDATA #REQUIRED
          middle  CDATA #IMPLIED
          surname CDATA #REQUIRED>

<!-- елемент city має необов'язковий атрибут type, який може мати значення "місто",
"смт" або "село". Причому, якщо атрибут не заданий, то приймається його значення по
замовчуванню "село" -->
<!ATTLIST city type (місто | смт | село) "село">

<!-- елемент product має обов'язковий символьний атрибут title, необов'язковий атрибут
для задання унікального ідентифікатора елемента id, символьний атрибут quantity, який
може бути також відсутнім, але в такому випадку приймається його значення по
замовчуванню 1 та атрибут value, якому обов'язково необхідно надати значення "дорого"-->
<!ATTLIST product
          id ID #IMPLIED
```

```
          title CDATA #REQUIRED
          quantity CDATA "1"
          value CDATA #FIXED "дорого">
```

А тепер доповнимо наш XML-документ атрибутами.

notebook.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT notebook (person)+>
<!ELEMENT person ANY>
<!ELEMENT fullname EMPTY>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT text (#PCDATA)>

<!ATTLIST fullname
          first   CDATA #REQUIRED
          middle  CDATA #IMPLIED
          surname CDATA #REQUIRED>
<!ATTLIST city type (місто | смт | село) "село">
```

notebook.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE notebook SYSTEM "notebook.dtd">
<notebook>
      <person>
            <fullname first="Vasja" surname="Pupkin"/>
            <phone>22-22-22</phone>
            <city type="смт">Zmerinka</city>
      </person>
      <person>
            <fullname first="Dasha" surname="Ivanova"/>
            <phone>333-33-33</phone>
            <city>Kyev</city>
      </person>
</notebook>
```

5. Поняття сутностей. Опис своїх власних та використання зарезервованих

В XML існує можливість розбивати документи на окремі іменовані об'єкти, які носять назву **сутності**. По суті, сутності являють собою макропідстановку, яку спочатку потрібно оголосити, а потім викликати. В місці виклику робиться текстова заміна посилання на вміст самої сутності. Це дозволяє зекономити сили та час розробнику.

Існує **два типи сутностей, що продиктовані їх описом**:

- **внутрішні (internal entities)** – описуються в DTD-документі разом з визначенням елементів та атрибутів;
- **зовнішні (external entities)** – описуються в окремому файлі, який може мати довільне розширення;
- **переметричні -** для визначення окремого типу даних**.**

5.1. Внутрішні сутності

Синтаксис **внутрішньої сутності** має наступний вигляд:

```
<!ENTITY назва_сутності "зміст">
```

Посилання на сутність, яке використовується для її виклику, являє собою комбінацією амперсанда, після якого йде назва сутності та крапка з комою:

```
&імя;
```

Наприклад:

```
<!ENTITY author "V.Pupkin Incorporation">
<!ENTITY copyright "Copyright &author;">
```

XML-документі виклик буде наступним:

```
<text>&author;</text>
<text>&copyright;</text>
```

З результату гарно видно, що посилання на сутність &author; було замінено на її вміст навіть в сутності &copyright;. Причому, якщо в майбутньому фірма змінить свою назву, тоді можна буде не змінюючи всього XML-документа, замінити лише її сутність &author;.

Крім того, внутрішні сутності поділяють на два **види**:

a) **аналізуємі (parsed entity) –** вміст сутності аналізується, тобто в випадку виклику сутностей всередині інших, вони всі будуть заміщатись їх вмістом. Причому такі сутності можуть в тілі містити розмітку, яка також буде інтерпретуватись;
b) **неаналізуємі (unparsed entity)** – вміст сутності не аналізується та не інтерпретується. В такому випадку в якості вмісту виступають зображення, двійкові файли, звук, відео тощо.

Варто було б відмітити, що в специфікації XML визначений набір зарезервованих сутностей, які перелічені в таблиці.

Опис	Символ	Посилання на сутність	
		Звичайна сутність	Символьна сутність
Знак “більше” (less that)	>	<	&#60;
Знак “меньше” (greater that)	<	>	>
Амперсанд (ampersand)	&	&	D
Подвійні лапки (quotes)	“	"	"
Апостроф або одинарні лапки (apostrophe)	‘	'	'

Вбудовані сутності можуть бути використані для заміни деяких символів там, де вони можуть бути сприйняті як розмітка. Наприклад:

Згідно описаних правил, ми можемо написати документ наступного вигляду:

notebook.xml

```
<!-- помилковий запис -->
<text>5 > 2</text>

<!-- вірний запис -->
<text>5 &lt; 2</text>
```

5.2. Зовнішні сутності

Зовнішні сутності, як вже було сказано, описуються в окремому зовнішньому файлі. Якщо в XML-документі зустрічається посилання на зовнішню сутність, то на її місце копіюється вміст вказаного файла. Синтаксис оголошення зовнішніх сутностей наступний:

```
<!ENTITY назва_сутності
         SYSTEM | PUBLIC
         ["ідентифікатор"] "URL">
```

Аналогічно оголошенню DTD-документа, ключове слово PUBLIC при оголошенні зовнішньої сутності вказує на те, що сутність має свій публічний ідентифікатор, який задається наступним. Якщо прикладна програма, яка обробляє XML-документ незнає сутності з таким ідентифікатором, вона завантажить її по вказаній адресі.

Якщо сутність описана в Вашому власному документі, тоді слід вказати ключове слово SYSTEM та абсолютний чи відносний шлях до необхідного файла.

Виклик зовнішніх сутностей аналогічний внутрішнім. Наприклад:

advertising.txt

```
Sale of office equipment.
Christmas discounts!
```

В **notebook.dtd** такий документ можна підключити одним з наступних способів:

```
<!ENTITY adv SYSTEM "advertising.txt">        <!-- відносний шлях -->
<!ENTITY adv SYSTEM "C:\\advertising.txt">    <!-- абсолютний шлях -->
<!ENTITY adv SYSTEM "http://www.myservice.com/advertising.txt"><!-- файл на сервері -->
<!-- з використанням світового ідентифікатора  -->
<!ENTITY adv PUBLIC "-//OGO//Advertising//Description"
"http://www.myservice.com/advertising.txt">
```

notebook.xml

```
<text>Рекламне оголошення: &adv;</text>
```

Варто було б зупинитись на тому, як працюють зовнішні сутності в документах стандарту XML 1.1. Основна проблема виникає при обробці сутностей в змішаному середовищі, тобто коли сутності XML 1.0 включені в документи XML 1.1. В специфікації XML 1.1 говориться про те, що сутності обробляються згідно документа, в якому вони використовуються. На практиці це означає, що можна використовувати старі сутності XML 1.0 в нових документах XML 1.1, але перед цим вони повинні бути помічені як XML 1.1. Єдина можлива проблема полягає в тому, що якщо додати один єдиний символ XML 1.1 в сутність XML 1.0, то процесор цього не помітить і буде його обробляти як вхідні дані в XML 1.1. Але це викличе проблеми, якщо потім вам буде необхідно використовувати цю ж сутність як частину документа XML 1.0.

5.3. Параметричні сутності

Сутності можна використовувати і для визначення окремого типу даних, подібно до доменів в базах даних Interbase/Firebird, MS SQL Server тощо. Такі сутності називаються **параметричні** або **сутностями-параметрами**. Доречі, вони також являються аналізуємими.

Параметричні сутності по синтаксису дуже схожі на звичайні сутності. **Відмінностей між ними лише дві**:

1. при виклику такої сутності перед іменем посилання ставиться символ відсотка (%), а не амперсанда (&);
2. параметричні сутності описуються та використовуються всередині DTD-документів.

Отже, синтаксис оголошення параметризованої сутності має наступний вигляд:

```
<!ENTITY % назва_сутності "вміст">
```

При цьому пропуск між знаком відсотка та назвою параметризованої сутності при її оголошенні являється ОБОВ'ЯЗВОВИМ. Наприклад:

```
<!-- оголошення сутності %coords та її застосування для елемента sphere -->
<!ENTITY % coords "x,y,z">
<!ELEMENT sphere (%coords;, radius)>  <!-- аналог <!ELEMENT sphere (x,y,z, radius)> -->

<!-- оголошення сутності %contentType та її застосування для елемента style -->
<!ENTITY % contentType "CDATA">
<!ATTLIST style
          id ID #IMPLIED
          type %contentType; #REQUIRED
          title %contentType; #REQUIRED > <!-- аналог <!ATTLIST style
                                                    id ID #IMPLIED
                                                    type CDATA #REQUIRED
                                                    title CDATA #REQUIRED > -->

<!-- оголошення сутності %params та її застосування для елемента people  -->
<!ENTITY % params "name CDATA #REQUIRED age CDATA #IMPLIED country CDATA #REQUIRED">
<!ATTLIST people %params;>  <!-- аналог <!ATTLIST people
                                        name CDATA #REQUIRED
                                        age CDATA #IMPLIED
                                        country CDATA #REQUIRED > -->
```

Як видно з прикладів, параметричні сутності заміняють деякий код створення частини елементів або цілком всіх елементів, атрибутів тощо. Це призводить до того, що у випадку необхідності зміни, наприклад, регістру символів при вказанні координат (% coords), вмісту атрибутів елемента style, складу або імен атрибутів елемента people достатньо буде зробити зміни в одному місці, - при оголошенні параметризованих сутностей.

Варто відмітити, що параметричні сутності широко використовуються в специфікаціях консорціума W3C і тому їх розуміння є обов'язковим.

Отже, підведемо підсумок коли **необхідно використовувати сутності**:

a) заміна часто використовуватих частин документа;
b) розбиття одного XML-документа на окремі частини;
c) заміна деяких символів, які можуть бути сприйняті як розмітка;
d) використання певних символів Unicode, які можуть бути замінені відповідними кодами.

6. Нотації. Їх опис та використання

По своїй суті XML-документи являються не більше, ніж текстом, а їх вміст має виключно текстовий вигляд. Але в багатьох випадках документи повинні включати в себе дані інших форматів, наприклад, графічні зображення або двійкові файли. Несумісність фізичної реалізації XML та зовнішніх даних такого типу не дозволяє включати їх в документ шляхом звичайних сутностей – для цих цілей використовуються неаналізуємі сутності та нотації.

Нотації (NOTATION) описують формат неаналізуємої сутності XML. Фактично вони використовуються для того, щоб оголосити певний тип даних та зв'язати його з зовнішньою програмою, яка дозволить його прочитати та відобразити.

Наприклад, ми можемо в якості даних вказати зображення в форматі .jpg. В такому разі Вам необхідно буде вказати яка програма дозволить візуалізувати такі дані, наприклад Picters Viewer або Adobe Photoshop.

Нотацію оголошувати можна одним з двох способів:

1) При використанні системного ідентифікатора, тобто програми, яка розміщується локально на комп'ютері:

```
<!NOTATION назва_нотації SYSTEM "шлях_до_програми_яка_читає_дану_нотацію">
```

При цьому, якщо назва виконуючої програми не є обов'язковою, тобто визначається системними налештуваннями по замовчуванню, тоді лапки можна залишити пустими.

2) При використанні світового ідентифікатора:

```
<!NOTATION назва_нотації PUBLIC "світовий_ідентифікатор">
<!NOTATION назва_нотації PUBLIC "світовий_ідентифікатор" "локальний_шлях">
```

Наприклад:

```
<!NOTATION txt  SYSTEM "notepad.exe">
<!NOTATION jpg  SYSTEM "C:\Program Files\ACDSee\acdsee.exe">
```

В даному випадку ми оголосили дві нотації:

- перша з іменем txt, яка буде служити для перегляду текстових документів за допомогою тестового редактора notepad;
- друга має назву jpg та дозволяє переглядати зображення з розширенням .jpg за допомогою програми перегляду зображень, яка знаходиться по адресі C:\Program Files\ACDSee\acdsee.exe.

Існує **<u>два основних способи використання нотацій</u>**:

1. Для визначення неаналізуємих сутностей та використання їх імен в якості типів атрибутів ENTITY або ENTITIES. На те, що сутність являється неаналізуємою вказує ключове слово NDATA (notation data - дані нотації), за яким йде назва раніше оголошеної нотації. Наприклад:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE menu [
      <!ELEMENT menu (menuitem*)>
      <!ELEMENT menuitem EMPTY>
      <!ATTLIST menuitem
                  img   ENTITY      #REQUIRED
                  title CDATA       #REQUIRED
                  href  CDATA       #IMPLIED>
      <!NOTATION png  SYSTEM "jpg-viewer.exe">
      <!ENTITY home SYSTEM "images/home.png" NDATA png>
      <!ENTITY about SYSTEM "images/about.png" NDATA png>
]>
<menu>
      <menuitem img="home" title="Home page" href="home.html" />
      <menuitem img="about" title="About as" href="about.html" />
</menu>
```

2. Для вказання імені нотації в якості типів атрибутів NOTATION для того, щоб задати формат даних, який містить даний елемент. Наприклад:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE document [
      <!ELEMENT document (#PCDATA)>
      <!-- за допомогою атрибута type буде визначатись формат даних, які будуть
           міститись в документі-->
      <!ATTLIST document
                  type  NOTATION (txt|html)     #REQUIRED>
      <!NOTATION txt    SYSTEM "notepad.exe"> <!-- для текстових документів - блокнот-->
      <!NOTATION html  SYSTEM "iexplore.exe"> <!-- для Web-сторінок-Internet Explorer-->
]>
<document type="html">
      <![CDATA[
            <!DOCTYPE HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN">
            <html>
            <head>
                  <title>Test Page</title>
```

```
                  <meta http-equiv="content-type" content="text-html; charset=windows-1251" />
            </head>
            <body>
                  <h1>Hello</h1>
            </body>
            </html>
      ]]>
</document>
```

Не дивлячись на те, що нотації являються доволі потужним механізмом, але також і доволі складним, вони на сьогоднішній день майже не використовуються. Аналогічного результату можна досягнути і іншими, більш простими способами, наприклад, використовуючи в елементах додаткові атрибути, в яких просто задається назва або шлях до програми, яка буде здійснювати відображення необхідних даних.

7. Область CDATA

Щоб задати область документу, яку при перегляді аналізатор буде розглядати як звичайний текст, ігноруючи будь-які інструкції і спеціальні символи, необхідно використовувати секцію **CDATA (character data – символьні дані)**.

Синтаксис оголошення даної обрласті наступний:

```
<![CDATA[ вміст ]]>
```

На відміну від коментарів, дані, розміщені в області CDATA, будуть використовуватись в програмі. Всередині цього блоку можна поміщати будь-яку інформацію, яка може знадобитись програмі-клієнту для виконання певних дій (в область CDATA можна поміщати, наприклад, інструкції JavaScript). Звичайно, потрібно слідкувати за тим, щоб в області, яка обмежена цими тегами не було послідовності символів **]]**.

Наприклад:

```
<![CDATA[ "Hello, World!!! :-)" ]]>
<![CDATA[ <b>Lalala</b> ]]>
```

8. Домашнє завдання

1. До створеного XML документа з попереднього домашнього завдання (книга рецептів) напишіть DTD-документ. Крім того, необхідно додати можливість виведення в кінці XML-документа інформації про авторські права на розробляєму книгу рецептів за допомогою сутності.

2. Створити параметричну сутність, яка описує використання елемента **FullAddress**, що може мати наступні атрибути:
   - **Country** – обов'язковий атрибут для опису інформації про країну, що має фіксований набір значень (наприклад, Україна, Росія, Білорусія та США);
   - **Town** – обов'язковий атрибут для збереження інформації про місто;
   - **Street** – необов'язковий атрибут для збереження даних про вулицю, будинок та квартиру (якщо існує).

   В XML документі такий елемент матиме наступний вигляд:

```
<FullAddress Country="Росія"   Town="Воронеж" Street="Лизюкова, д.12, кв.100" />
<FullAddress Country="Україна" Town="Львів" />
```