План лекции по XML и XPath

- Общие принципы
- Document type definition
- SAX и DOM
- XPath

У истоков

- Standard Generalized Markup Language (SGML)
  - Предназначался для описания структуры сложных документов
  - Был разработан в 1970 году
  - Основные цели:
    - Все документы должны быть выполнены в строгом соответствии с правилами
    - Уменьшение количества документации

Наследники

- Hypertext Markup Language (HTML)
  Язык разметки гипертекста (описание представления Web-страницы)

- Extensible Markup Language (XML)
  Язык для описания иерархических данных (портируемое хранение данных)
  http://www.w3.org/XML/

Отличия XML от HTML

- XML чувствителен к регистру
- В XML нужно закрывать тэги
- В XML часто встречаются тэги, одновременно открывающие и закрывающие
  `<img src="coffeecup.png" />`
- В XML значения атрибутов должны быть заключены в кавычки
- В XML все атрибуты должны иметь значения

Пример XML

```
<configuration>
  <title>
    <font> <name>Helvetica</name> <size>36</size> </font>
  </title>
  <body> <name>Times Roman</name> <size>12</size> </body>
  <window> <width>400</width> <height>200</height> </window>
  <color>
    <red>0</red>
    <green>50</green>
    <blue>100</blue>
  </color>
  <menu>
    <item>Times Roman</item>
    <item>Helvetica</item>
  </menu>
</configuration>
```

Структура XML-документа

- Заголовок

```
<?xml version="1.0"?>
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="ex01-1.xsl"?>
```

- Объявления типа документа

```
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.
//DTD Web Application 2.2//EN"
"http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
```

- Корневой элемент

```
<configuration>
</configuration>
```

Структура XML-документа

- Смешанное наполнение не рекомендуется

```
<font>
        Helvetica
        <size>36</size>
</font>
```

- Существуют атрибуты

```
<font>
        <name>Helvetica</name>
        <size unit="pt">36</size>
</font>
```

Некоторые инструкции

- Символы

```
&#233, &#x2122
```

- Стандартные символы

```
&lt; &gt; &amp; &quot; &apos;
```

- Инструкции обработки

```
<?xml version="1.0"?>
```

- Комментарии

```
<!-- This is a comment. -->
```

Правильный документ

- Начинается с объявления
- Содержит один уникальный корневой элемент
- Все открытые теги закрываются
- Учтена чувствительность к регистру
- Теги корректно вложены друг в друга
- Значения всех атрибутов заключены в кавычки
- Специальные символы задаются с помощью инструкций

Document Type Definition (DTD)

- Содержит правила, описывающие структуру документа
- Транслятор может автоматически проверять документ на соответствие этим правилам
- Описывает дочерние элементы и атрибуты для каждого элемента
- Включение в XML-документ

```
<!DOCTYPE имя [правила]>

<!DOCTYPE configuration SYSTEM "config.dtd">

<!DOCTYPE configuration SYSTEM
"http://myserver.com/config.dtd">
```

Регулярные выражения

Правило	Смысл
E*	0 или больше вхождений E
E+	1 или больше вхождений E
E?	0 или 1 вхождение E
E1|E2|…|En	Одно из E1, E2, …, En
E1, E2, …, En	Последовательность E1, E2, …, En
#PCDATA	Текст
(#PCDATA|E1|…|En)*	Смешанное наполнение
ANY	Произвольный дочерний тэг
EMPTY	Нет дочерних тэгов

Примеры выражений

```
Описание меню
<!ELEMENT menu (item)*>

Описание шрифта
<!ELEMENT font (name,size)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT size (#PCDATA)>

Описание главы в книге
<!ELEMENT chapter
(intro,(heading,(para|image|table|note)+)+)>
```

Описание атрибутов: типы

Тип	Смысл
CDATA	Произвольная строка
(A1|A2|…|An)	Один из строковых атрибутов A1, A2, …, An
NMTOKEN, NMTOKENS	Одна или более строк, записанных по правилам имен
ID	Уникальный ID
IDREF, IDREFS	Одна или более ссылка на уникальный ID
ENTITY, ENTITIES	Ссылки на внешние сущности

Описание атрибутов: значения

Значение	Смысл
#REQUIRED	Атрибут обязателен
#IMPLIED	Атрибут опционален
A	Атрибут опционален, если значение не указано, то принимается равным A
#FIXED A	Атрибут не указывается или равен A

Примеры выражений

```
<!ATTLIST font style (plain|bold|italic|bold-italic)
plain>
<!ATTLIST size unit CDATA #IMPLIED>

<!ELEMENT gridbag (row)*>
<!ELEMENT row (cell)*>
<!ATTLIST cell gridwidth CDATA "1">
<!ATTLIST cell gridheight CDATA "1">
<!ATTLIST cell fill (NONE|BOTH|HORIZONTAL|VERTICAL)
"NONE">
<!ATTLIST cell anchor (CENTER|NORTH|NORTHEAST|EAST
|SOUTHEAST|SOUTH|SOUTHWEST|WEST|NORTHWEST)
"CENTER">
<!ATTLIST cell ipadx CDATA "0">
<!ATTLIST cell ipady CDATA "0">
```

XML Schema

- Предназначена для того же, что и DTD
- Для описания правил используется непосредственно XML
- Имеет более гибкие возможности, чем DTD
  - Расширяема;
  - Есть понятие типа данных;
  - Есть понятие пространства имен;
- Сложнее в восприятии и программировании средств, ее обрабатывающих
- www.w3.org/XML/Schema
  http://www.w3schools.com/Schema/default.asp

Поддержка типов данных

- Проще описывать допустимое содержимое документа
- Проще проверять корректность данных
- Проще накладывать ограничения на данные
- Проще определять формат данных

XML Schema описывается на XML

- Не требуется изучение еще одного языка
- Вы можете использовать свой любимый XML-редактор для работы со схемой
- Вы можете работать со схемой программно
- Вы можете изменять свою схему с помощью XSLT

Документ и тип DTD

```
<?xml version="1.0"?>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

XML Schema для документа

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com"
elementFormDefault="qualified">

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="to" type="xs:string"/>
      <xs:element name="from" type="xs:string"/>
      <xs:element name="heading" type="xs:string"/>
      <xs:element name="body" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

Указание типа документа

```
<?xml version="1.0"?>

<!DOCTYPE note SYSTEM "http://www.w3schools.com/dtd/note.dtd">

<note>
  <!-- Some content -->
</note>
```

```
<?xml version="1.0"?>

<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
 <!-- Some content -->
</note>
```

Extensible Stylesheet Language (XSL)

- Комплекс технологий, связанных с преобразованием и представлением XML-документов
- Обычно используется для преобразования документов в XML, HTML, текст и PDF (XSL-FO)
- XSL Transformations (XSLT) – язык, на котором описываются правила преобразования
- XPath – язык, позволяющий формулировать используемые в процессе преобразования выражения, использующие различные фрагменты документа
- http://www.w3.org/Style/XSL/
  http://www.w3schools.com/xsl/

Принципы XSL

- Контекстно-зависимый язык
- Основные элементы – выводимый текст и шаблоны
  - Текст просто выводится
  - Шаблоны описывают некоторые действия
    - Могут быть вызваны явно
    - Могут быть вызваны неявно, по условию совпадения шаблона
- Имеются средства управления ходом выполнения
- Позволяет создавать и вызывать бибилиотеки с помощью тега `<xsl:include href="..."/>`

Пример XML (catalog.xml) и XSL-трансформации

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="tranformation.xsl" ?>
  <catalog>
    <cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
    </cd>
    <cd>
      <title>Hide your heart</title>
      <artist>Bonnie Tyler</artist>
      <country>UK</country>
      <company>CBS Records</company>
      <price>9.90</price>
      <year>1988</year>
    </cd>
    …
  </catalog>
```

Пример XSL (tranformation.xsl)

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Результат трансформации

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith

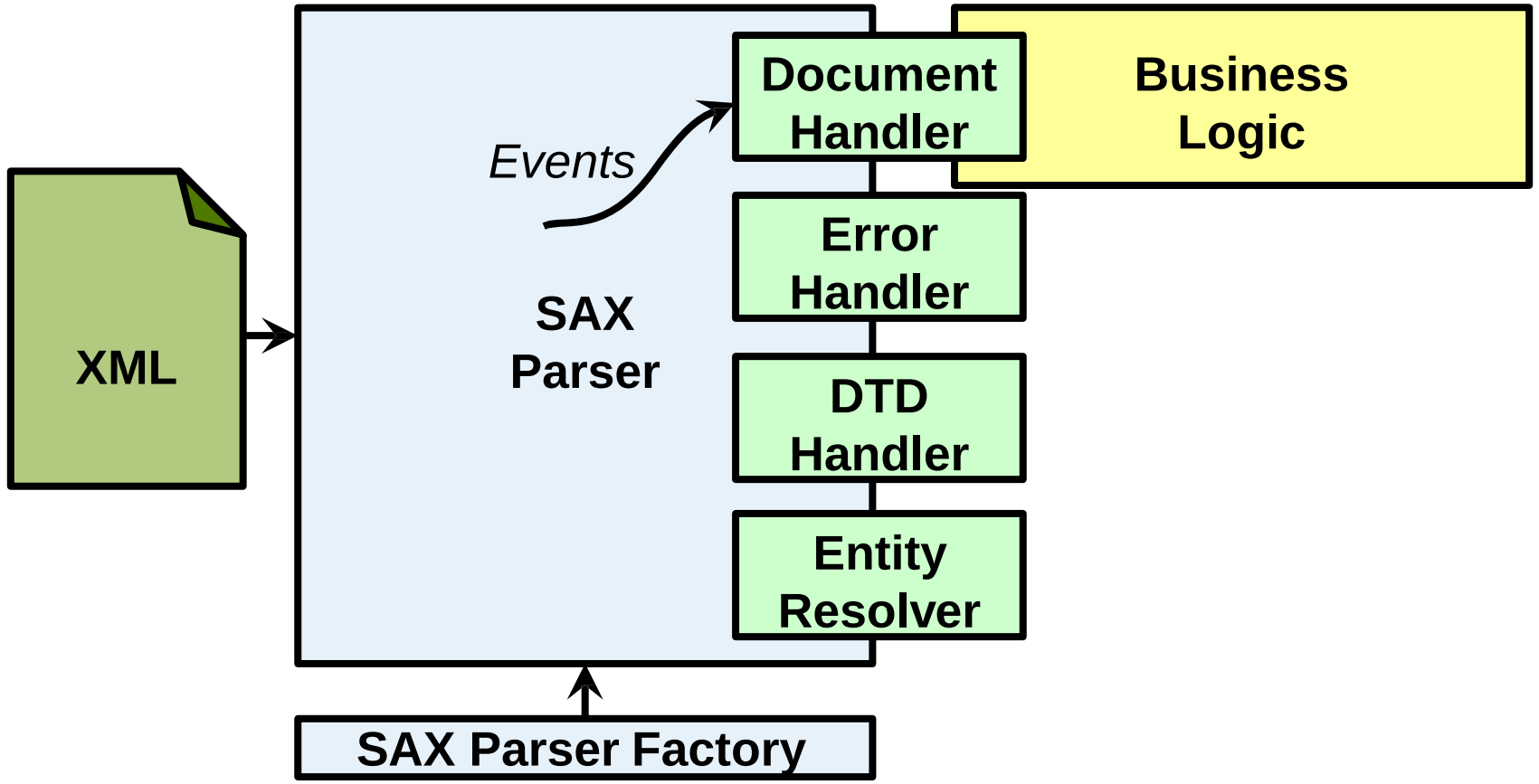
Обработка XML

Два подхода:

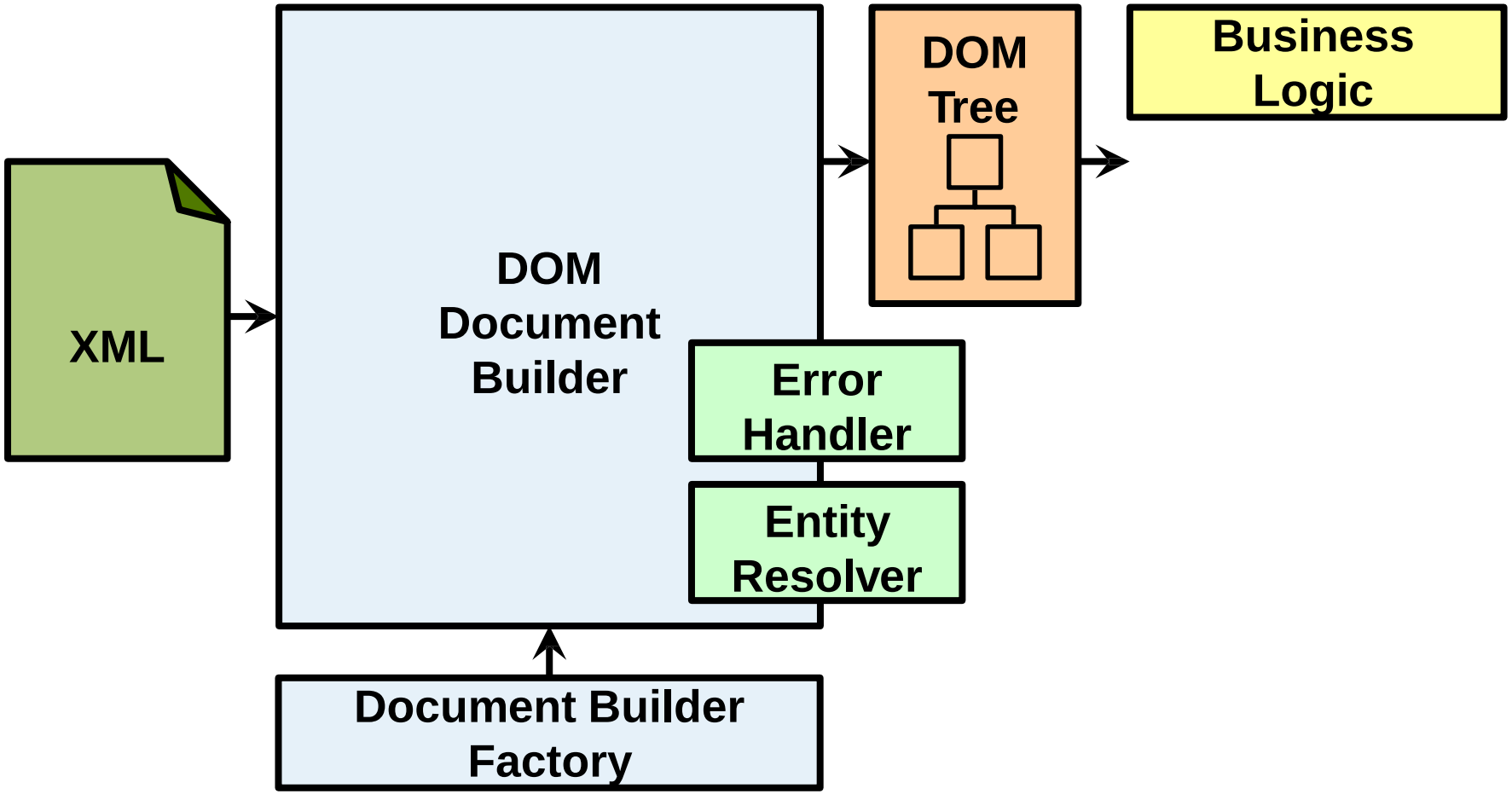
- **Simple API for XML (SAX)**
  Порождает события в процессе чтения XML документа

- **Document Object Model (DOM)**
  Представляет XML документ в форме древовидной структуры элементов

Логика SAX


Document Handler
Business Logic
Events
Error Handler
XML
SAX Parser
DTD Handler
Entity Resolver
SAX Parser Factory

Логика DOM


XML
DOM
Document
Builder
DOM
Tree
Business
Logic
Error
Handler
Entity
Resolver
Document Builder
Factory

Особенности SAX и DOM

SAX	DOM
Модель обработки событий	Древовидная структура данных
Последовательный доступ (поток событий)	Произвольный доступ (структура данных в памяти)
Используется мало памяти (порождаются только события)	Используется много памяти (документ загружен полностью)
Для обработки частей документа (обработка релевантных событий)	Для редактирования документа (обработка данных в памяти)
Для однократной обработки документа	Для многократной обработки документа

Введение в XPath.

XPath на примерах.

Мотивация для создания XPath

XML:

```
<configuration>
  . . .
  <database>
     <username>dbuser</username>
     <password>secret</password>
  </database>
  . . .
</configuration>
```

Получение значения **username** с помощью анализа DOM:

- Получение узла документа;
- Перечисление его дочерних узлов;
- Обнаружение элемента database;
- Получение первого дочернего элемента, username;
- Получение первого дочернего узла, имеющего тип Text;
- Получение данных.

Получение узла username с помощью XPath:

/configuration/database/username

XPath

- Вспомогательный язык, позволяющий обращаться к элементам документа
- Имя элемента представляется в виде пути **`/bookstore/book/title`**
- Обращение может происходить и к атрибутам
- https://www.w3.org/TR/xpath/all/ https://www.w3schools.com/xml/xpath_intro.asp

Примеры выражений XPath

Выражение	Результат
`bookstore`	Все дочерние элементы для элемента bookstore
`/bookstore`	Корневой элемент bookstore
`bookstore/book`	Все элементы book, дочерние для bookstore
`//book`	Все элементы book в документе
`bookstore//book`	Все элементы book в рамках элемента bookstore
`@lang`	Атрибуты lang
`.`	Текущий элемент
`..`	Родительский элемент

Xpath online:
https://www.freeformatter.com/xpath-tester.html

XPath

Язык XPath позволяет находить (описывать критерии поиска) набор узлов в XML-документе.

```
XPath: /AAA
```

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

XPath

XPath: /AAA/CCC

Все элементы CCC, являющиеся дочерними к корневому AAA

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

XPath

XPath: //BBB

Все элементы документа, которые соответствуют указанному шаблону:

```
<AAA>
  <BBB />
  <CCC />
    <BBB />
  </CCC>
  <BBB />
  <DDD>
    <BBB />
  </DDD>
  <CCC />
</AAA>
```

XPath

XPath: //DDD/BBB

Будут выбраны все элементы BBB, являющиеся детьми DDD.

```
<AAA>
  <BBB/>
  <CCC/>
    <BBB/>
  </CCC>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

XPath

XPath: /AAA/CCC/DDD/*

Будут выбраны все элементы, являющиеся прямыми потомками /AAA/CCC/DDD

```
<AAA>
    <CCC>
      <DDD>
        <BBB/>
        <BBB/>
        <EEE/>
        <FFF/>
      </DDD>
    </CCC>
    <CCC>
      <BBB>
        <BBB>
         <BBB/>
        </BBB>
      </BBB>
    </CCC>
</AAA>
```

XPath

`XPath: /*/*/*/BBB`

Будут выбраны все элементы BBB, имеющие трех предков.

```
<AAA>
    <CCC>
      <DDD>
        <BBB/>
        <BBB/>
        <EEE/>
        <FFF/>
      </DDD>
    </CCC>
    <CCC>
      <BBB>
        <BBB>
          <BBB/>
        </BBB>
      </BBB>
    </CCC>
</AAA>
```

XPath: //* - Все элементы

XPath

XPath: /AAA/BBB[1]
Будет выбран первый потомок BBB элемента AAA

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
</AAA>
```

XPath: /AAA/BBB[last()]
Будет выбран последний потомок BBB элемента AAA

```
<AAA>
   <BBB/>
   <BBB/>
   <BBB/>
   <BBB/>
</AAA>
```

XPath

Атрибуты определяются префиксом @.
XPath: //@id
Выбираются все атрибуты @id:

```
<AAA>
  <BBB id="b1" />
  <CCC id="b2" />
  <BBB name="bbb">
</AAA>
```

XPath: //BBB[@id]
Выбираются элементы BBB, имеющие атрибут id:

```
<AAA>
  <BBB id="b1"/>
  <BBB id="b2"/>
  <BBB name="bbb">
</AAA>
```

XPath

XPath: //BBB[@*]
Выбираются элементы BBB, имеющие любой (хоть какой-нибудь) атрибут.

```
<AAA>
   <BBB id="b1"/>
   <BBB id="b2"/>
   <BBB name="bbb">
</AAA>
```

XPath: //BBB[not(@*)]
Выбираются элементы BBB, не имеющие ни одного атрибута

```
<AAA>
   <BBB id="b1"/>
   <BBB id="b2"/>
   <BBB name="bbb">
   <BBB />
</AAA>
```

XPath

XPath: //BBB[@id='b1']
Выбираются элементы BBB, имеющие атрибут id со значением b1.

```
<AAA>
  <BBB id="b1" />
  <BBB id="b2" />
  <BBB name="bbb">
</AAA>
```

XPath: //BBB@id
Выбираются атрибуты id элемента BBB.

```
<AAA>
  <BBB id="b1" />
  <BBB id="b2" />
  <CCC id="bbb">
</AAA>
```