VSR | EDU

XML

XML

Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und Selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

ComplexType & ComplexContent

- ComplexType describes structures (elements) and additional information (name-value pairs / attributes)
  - Forms: Sequence, Choice, Group
  - Nesting of these types is allowed
  - PCDATA between elements must be explicitly allowed, for example: <xs:complexType name="Item" **mixed="true"**>…
- Inheritance with extension or restriction of elements or attributes
  - Homeworks: What happens if ComplexType inherits from ComlexType or from SimpleType?

```
<xs:complexType name="DateDescr">
  <xs:sequence>
    <xs:element ref="date" />
    <xs:element minOccurs="0" maxOccurs="1"
                ref="description" />
  </xs:sequence>
</xs:complexType>

<xs:element name="Item" type="DateDescr"/>


<xs:complexType name="DateDescrUrl">
  <xs:complexContent >
    <xs:extension base ="DateDescr">
      <xs:choice minOccurs ="0" maxOccurs ="1">
        <xs:element name="URL" type ="xs:anyURI"/>
      </xs:choice>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:element name="ItemMitURL" type="DateDescrUrl"/>
```

Content Compared

SimpleContent

- Focused on PCDATA (unstructured content)
- Attributes are allowed
- Contents can be strong typed, according to data types from the XML schema namespace
- SimpleTypes can be used as a base for new types

ComplexContent

- Focused on structures (elements with or without attributes)
- Contents can be mixed, structures are provided in XML schema namespace
- ComplexTypes can be used as a base for new types

Inheritance (Type Substitution)

- News (Item+)
- Item ::= Descr
- DescrUrl inherits from Descr

- Problem:
  - Both *Desc* and *DescrUrl* could be correct types, which find use with Item as a structure
  - How does one announce that?

05-01.xsd | K05-02.xsd | K05-03.xml | **K05-04.xsd**

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
        targetNamespace="http://example.org/news"
        xmlns="http://example.org/news"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="News">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Item" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Descr">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1"
                  name="descr" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DescrUrl">
    <xs:complexContent >
      <xs:extension base ="Descr">
        <xs:choice minOccurs ="0" maxOccurs ="1">
          <xs:element name="URL" type ="xs:anyURI"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="Item" type="Descr"/>
</xs:schema>
```

Inheritance (Type Substitution)

- News (Item+)
- Item ::= Descr
- DescrUrl inherits from Descr

- Problem:
  - Both *Desc* and *DescrUrl* could be correct types, which find use with Item as a structure
  - How does one announce that?

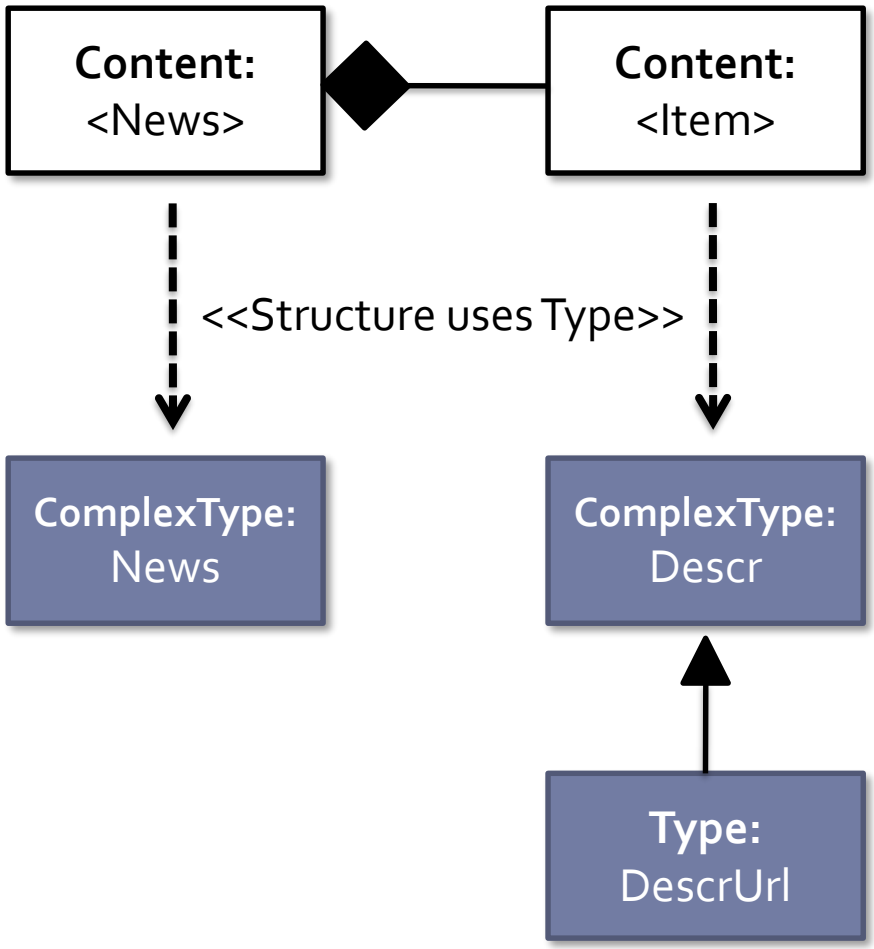
05-01.xsd | K05-02.xsd | K05-03.xml | **K05-04.xsd**

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
        targetNamespace="http://example.org/news"
        xmlns="http://example.org/news"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="News">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Item" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Descr">
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="1"
                  name="descr" type="xs:string" />
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="DescrUrl">
    <xs:complexContent >
      <xs:extension base ="Descr">
        <xs:choice minOccurs ="0" maxOccurs ="1">
          <xs:element name="URL" type ="xs:anyURI"/>
        </xs:choice>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:element name="Item" type="Descr"/>
</xs:schema>
```

Inheritance (Type Substitution)

- News (Item+)
- Item ::= Descr
- DescrUrl inherits from Descr

- Problem:
  - Both *Desc* and *DescrUrl* could be correct types, which find use with Item as a structure
  - How does one announce that?

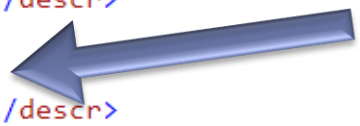


Inheritance (Type Substitution)

- Solution:
  - Instance directive via xsi:type

K05-05.xml* | K05-01.xsd | K05-02.xsd | K05-03.xml | K05-04.xsd

```
<?xml version="1.0" encoding="utf-8"?>
<News xmlns="http://example.org/news"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://example.org/news K05-04.xsd">
  <Item>
    <descr>Ein normales Item</descr>
  </Item>
  <Item xsi:type="DescrUrl">
    <descr >Item mit Subtype</descr>
    <URL >http://extension.example.org</URL>
  </Item>
</News>
```



05-01.xsd | K05-02.xsd | K05-03.xml | K05-04.xsd

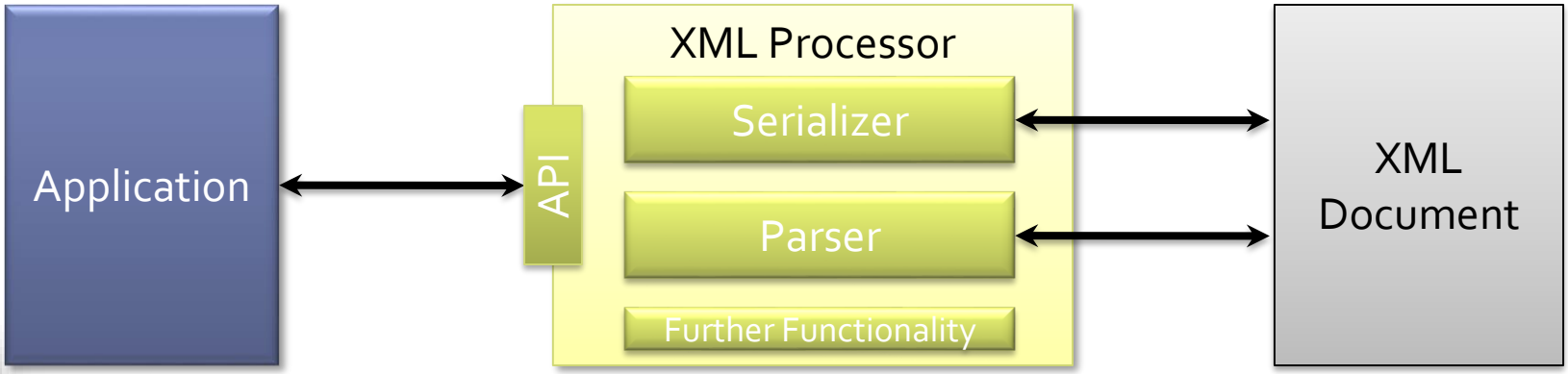
```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified"
        targetNamespace="http://example.org/news"
        xmlns="http://example.org/news"
        xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="News">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="unbounded" ref="Item" />
ence>
Type>

e name="Descr">
e>
t minOccurs="0" maxOccurs="1"
   name="descr" type="xs:string" />
e>
pe>

e name="DescrUrl">
Content >
sion base ="Descr">
ce minOccurs ="0" maxOccurs ="1">
lement name="URL" type ="xs:anyURI"/>
oice>
nsion>
Content>
pe>

  <xs:element name="Item" type="Descr"/>
</xs:schema>
```

Chapter 6

PROCESSING WITH XML PROCESSORS

Architecture for XML Processing

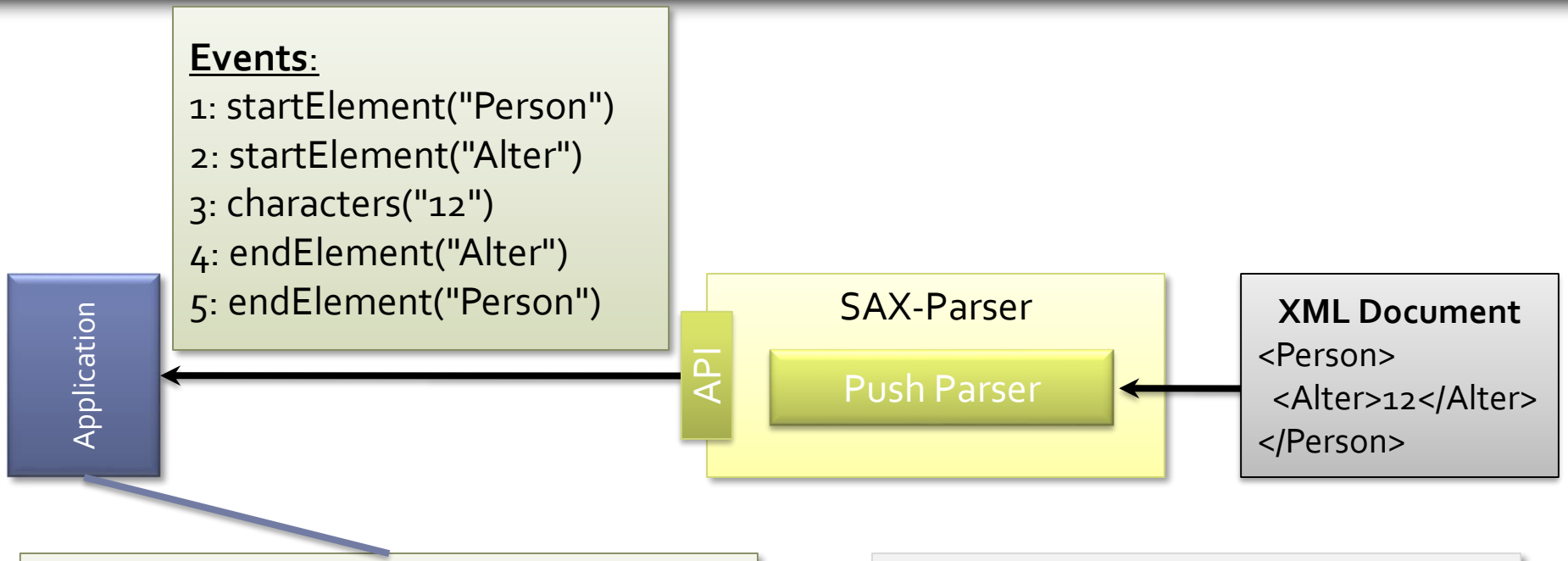
- Goal: Simplification of XML document processing by "Standardization of APIs" of XML processors
  - API (Parser) for reading the structure and content
  - API (Serializer) for writing
- Examples: SAX-Parser and DOM-Parser



SAX: Simple API for XML

- SAX: Project since the end of 1997, SAX 1.0 Parser beginning of 1998
- SAX operation
  - Event-driven processing of XML documents
  - Parser uses a Push Model (Push Parser)
  - API defines three callback methods, which are called by the parser in an event-driven manner
  - De-facto standard under the Push Parsers
- API (Pseudocode):
  - **startElement** (namespaceURI, localName, attributes):
    Is called by the parser if a Start Tag is found
  - **endElement** (namespaceURI, localName):
    Is called by the parser if a End Tag is found
  - **characters** (char[], startIndex,length)

SAX: Example



Callback Definitions:

startElement(elem):
if (elem.name = "Alter") → print: "Ihr Alter ist:"

characters(s):
print: s

endElement(elem):
//do nothing.

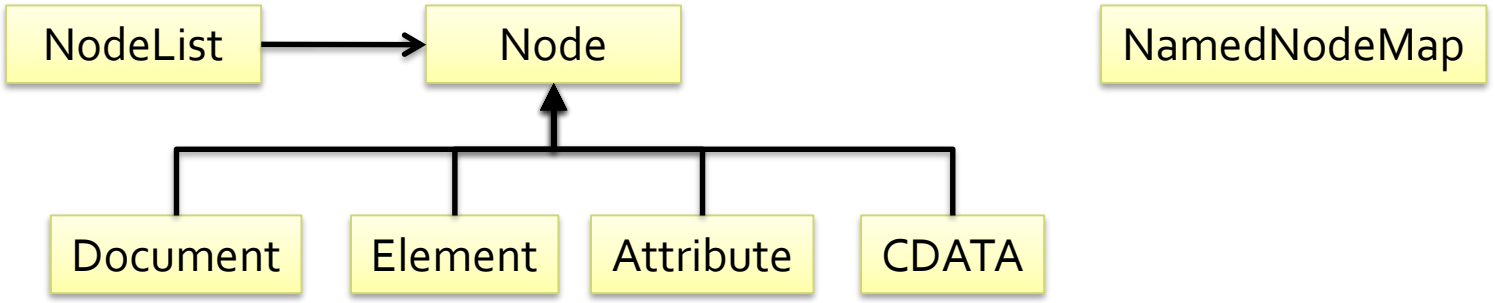
Advantage: Very fast
Problem:
Looks very simple, but SAX maintains no context (such as nesting or element dependencies)

DOM: Document Object Model

- DOM
  - W3C-Standard (recommendation)
  - Describes an API for XML- and HTML documents – Is independent of the programming language
  - Defines logical structures of documents (access and modification)
  - Is no binary code – only defines the programming model, which can then be supported by various XML processor vendors
- History & development
  - Introduced under pressure from browser developers who wanted to facilitate dynamic websites (i.e. Dynamic HTML). Problem: No unified JavaScript access to document structures in different browsers
  - DOM Level 1 (Focus: Objects), later DOM Level 2 (Focus: Events, Stylesheets), and Level 3 (Focus: XPath, loading/saving etc.)
  - DOM Level 3 Core Specification: W3C Recommendation 07 April 2004

DOM Level 1: Objects (simplified)

- Some basic classes for API description
  - XML document is then a set of instances of DOM classes, i.e. an XML document is a hierarchy of Node objects, which implement different interfaces according to the marked Node type



DOM Level-1: Node

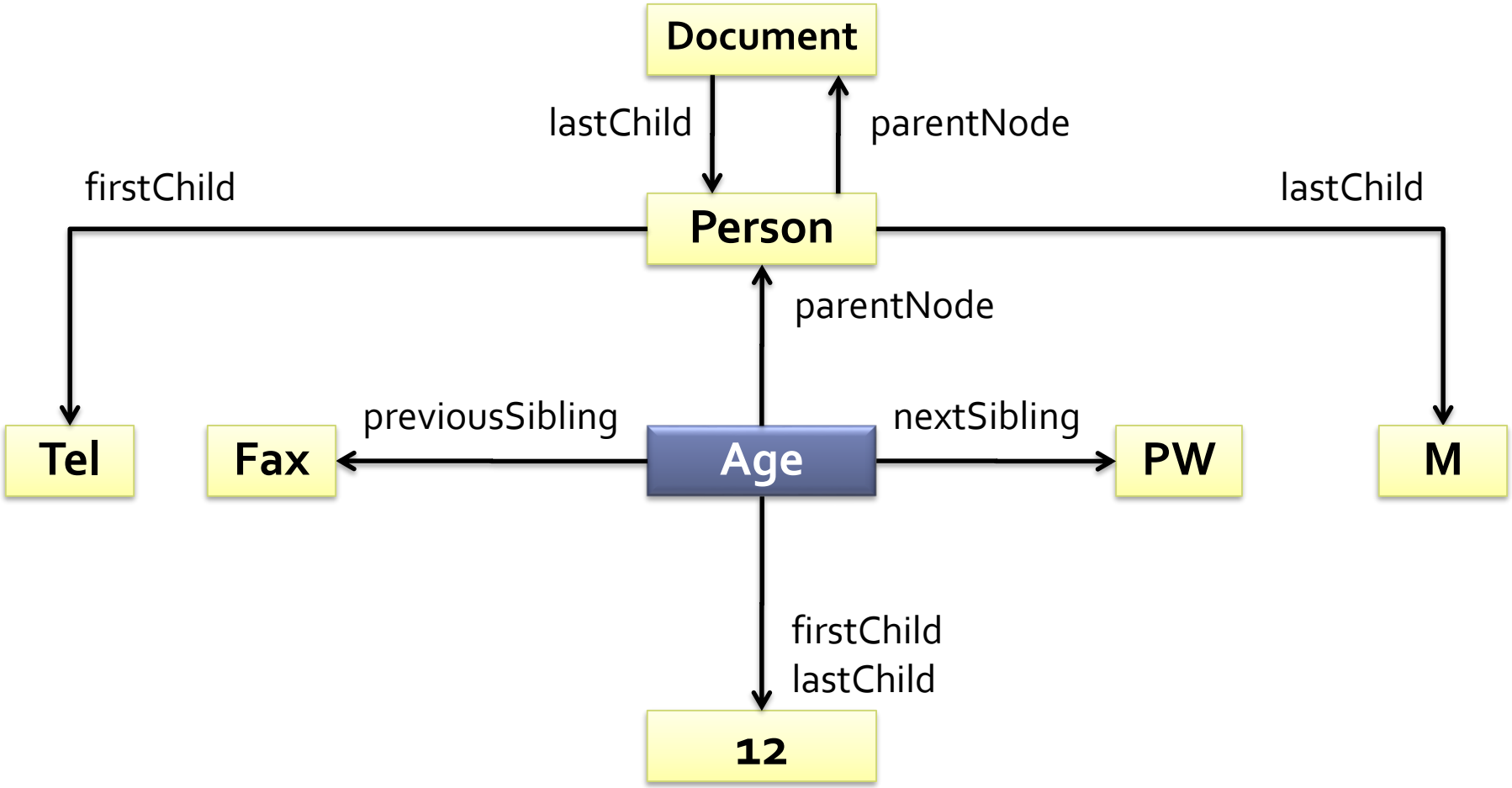
- Slice of W3C description (IDL Definition) :

```
interface Node { // NodeType

readonly attribute DOMString nodeName;
attribute DOMString nodeValue;

readonly attribute unsigned short nodeType;
readonly attribute Node parentNode;
readonly attribute NodeList childNodes;
readonly attribute Node firstChild;
readonly attribute Node lastChild;
readonly attribute Node previousSibling;
readonly attribute Node nextSibling;
readonly attribute NamedNodeMap attributes;

Node insertBefore (in Node newChild, in Node refChild) raises(DOMException);
Node replaceChild( in Node newChild, in Node oldChild) raises(DOMException);
Node removeChild (in Node oldChild) raises(DOMException);
Node appendChild( in Node newChild) raises(DOMException);
boolean hasChildNodes ();
Node cloneNode( in boolean deep);

...}
```

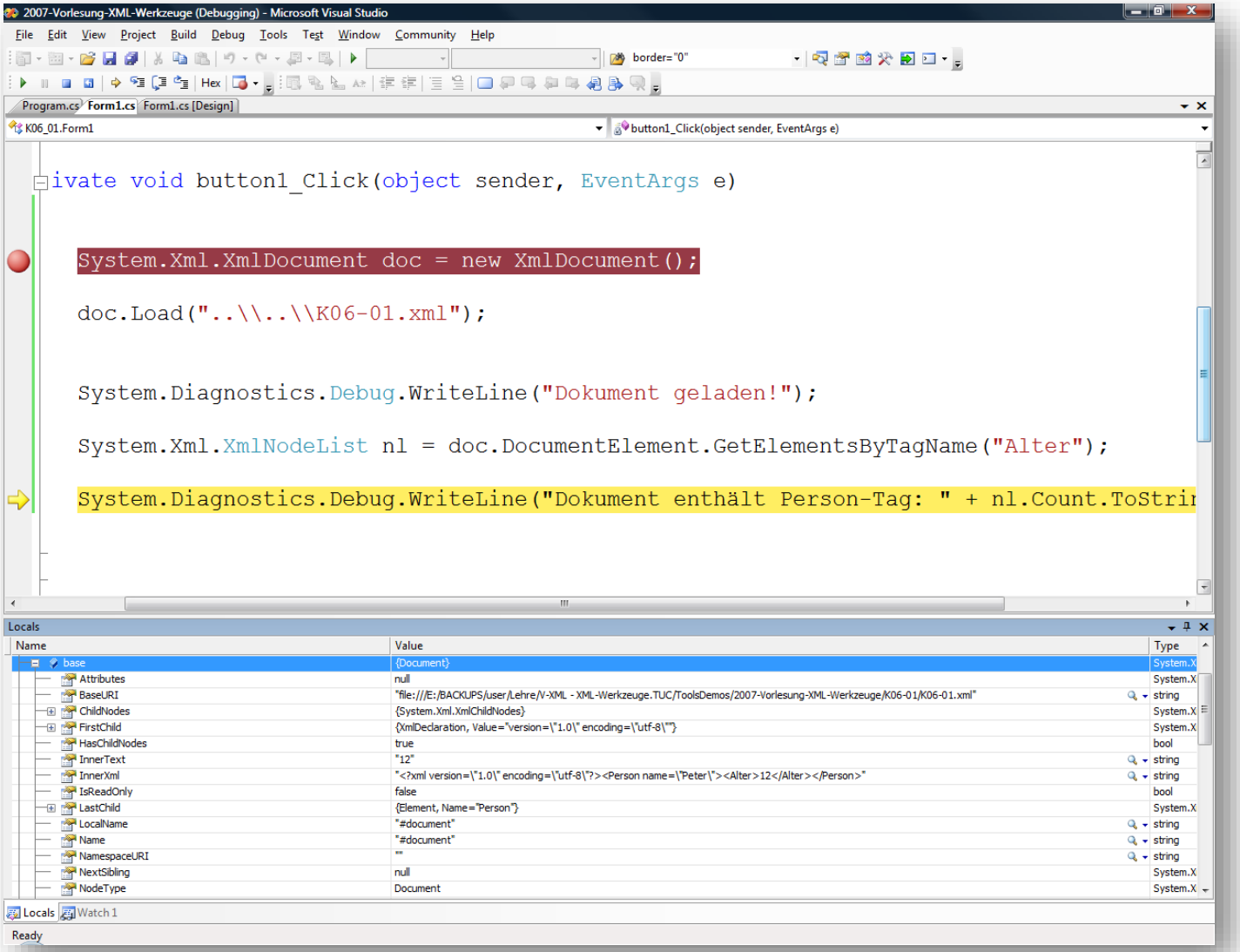
Example

```
<Person name="Peter Pater">
   <Tel>123</Tel>
   <Fax>888</Fax>
   <Age>12</Age>
   <PW>pass123</PW>
   <M>p@ter.com</M>
</Person>
```

Current Node: Age



Demo: .Net DOM-Parser



SAX vs DOM

SAX-Parser

- **Advantages:**
  - Fast to load a simple part of a document
  - Very efficient given very big data amounts (context must not be saved)
- **Disadvantages:**
  - No context
  - Only parsing is possible (modification not planned)

DOM-Parser

- **Advantages:**
  - Standardized
  - Context is maintained by the parser
  - Parsing and modification are possible
  - Is mostly combined with further aids
- **Disadvantages:**
  - Requires more storage due to the context

Conclusion: With rare exceptions, one is much better served by a DOM-Parser (especially in the context of XML document evolution)

Homework

- Which methods must be supported by a DOM Level 3 Parser for XML nodes of type Element?
- What should a DOM Level 3 Parser deliver to the following method call: getElementsByTagName("elemname")

- Relevant information can be found here: http://w3.org/TR/DOM-Level-3-Core/