



DTDs und XML-Schema

Prof. Dr.-Ing. Robert Tolksdorf
Freie Universität Berlin
Institut für Informatik
Netzbasierende Informationssysteme
tolk@ag-nbi.de

letzte Woche

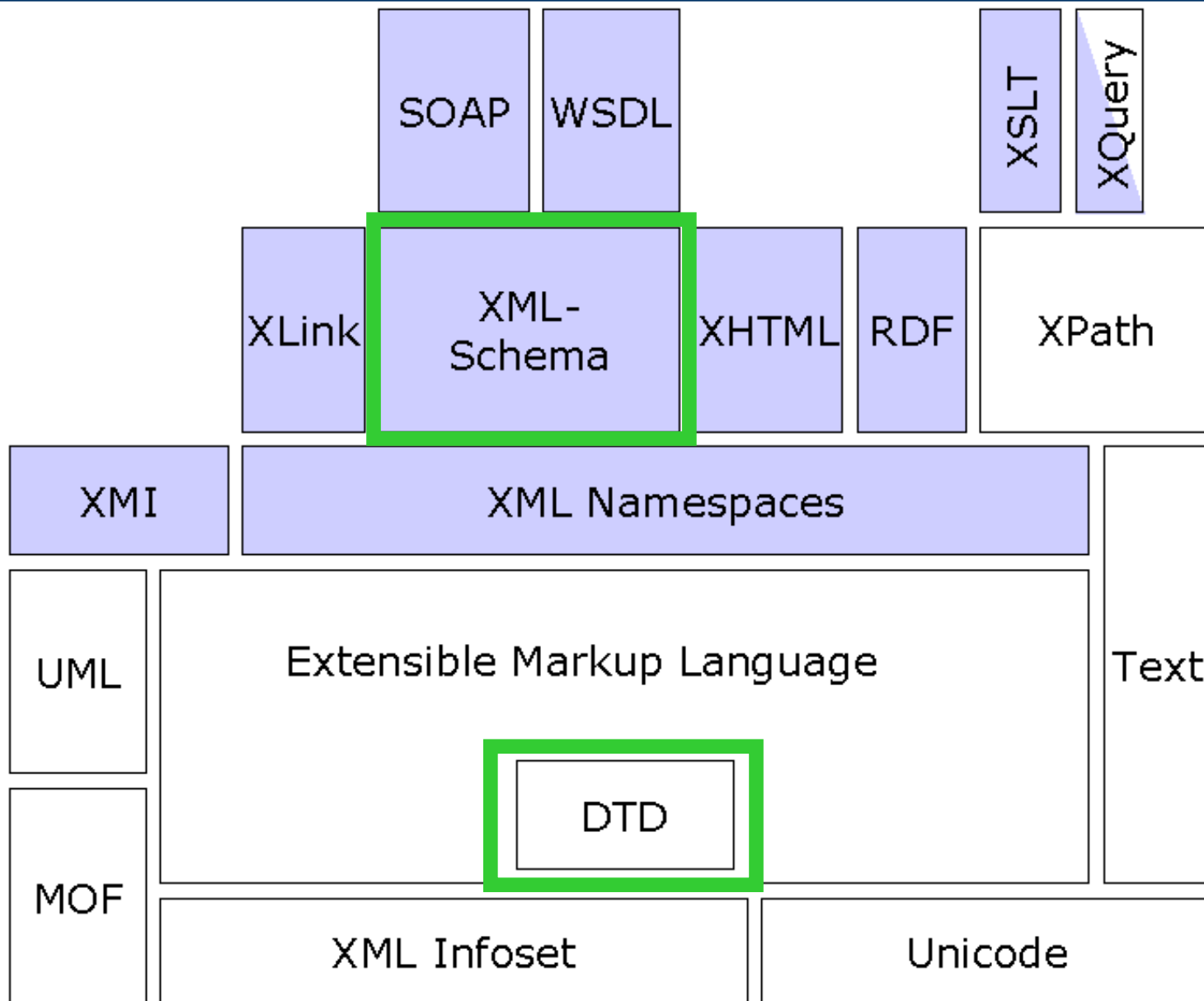
- XML-Syntax
- Namensräume

Heute

- Definition von XML-Sprachen
- DTDs und XML-Schema anhand eines einheitlichen Beispiels

nächste Woche

- XML-Schema im Detail



Quelle: <http://www.jeckle.de/images/xml/languageFamily.gif>

oder so?

```
<book>
  <title>My Life and Times</title>
  <authors>
    <author>
      <first>Paul</first>
      <last>McCartney</last>
    </author>
  </authors>
  <date>
    <year>1998</year>
    <month>July</month>
  </date>
  <isbn>94303-12021-43892</isbn>
  <publisher>McMillin
  Publishing</publisher>
</book>
```

so?

```
<Book>
  <Title>My Life and Times</Title>
  <Author>Paul McCartney</Author>
  <Date>July, 1998</Date>
  <ISBN>94303-12021-43892</ISBN>
  <Publisher>McMillinPublishing</Publisher>
</Book>
```

- ⇒ einheitliches Format nötig
- ⇒ Format sollte durch XML-Prozessor validierbar sein

Spezifische Formate

- **prinzipieller Aufbau von Dokumenten:**

Welche Elemente/Attribute dürfen wo verwendet werden?

- **Datentypen der Inhalte:** Welche Inhalte sind erlaubt?

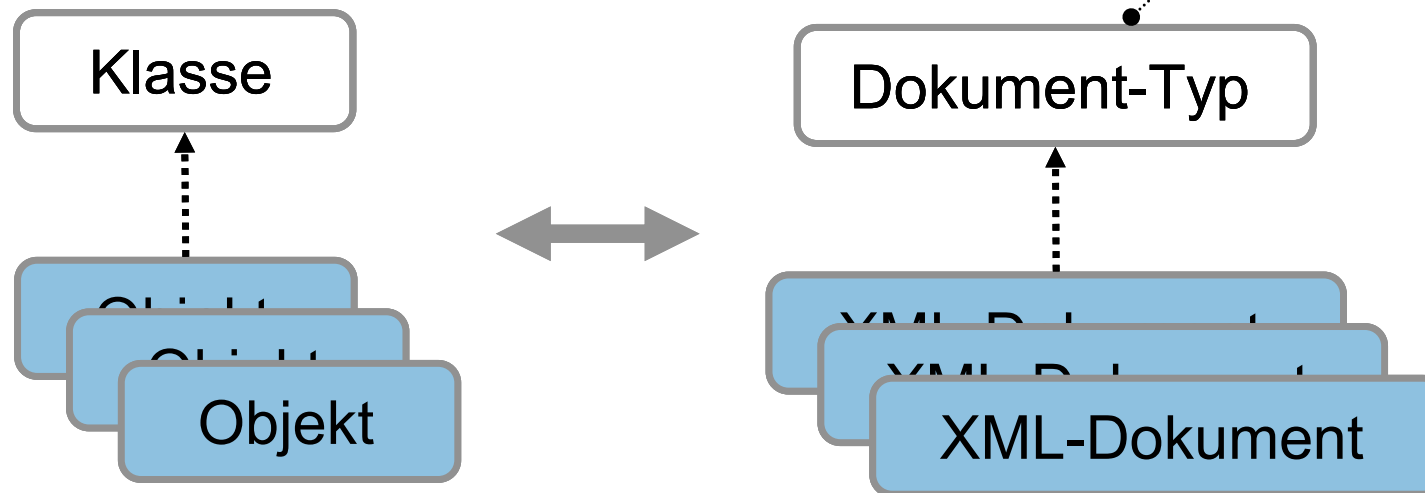
```
<Book >
  <Title> PCDATA </Title>
  <Author> PCDATA
</Author>
  <Date> PCDATA </Date>
  <ISBN> PCDATA </ISBN>
  <Publisher> PCDATA
</Publisher>
</Book >
```

- konkrete Inhalte werden nicht beschrieben

⇒ Klasse von erlaubten XML-Dokumenten

- auch **Dokument-Typ, XML-Sprache, Anwendung von XML** oder **Content Modelle** genannt

Klasse von erlaubten
XML-Dokumenten
= XML-Sprache
= Content Modell



- Dokument-Typ kann mit einer DTD, einem XML-Schema oder ähnlichen Formalismen definiert werden.



Document Type Definitions (DTDs)

Wie sieht eine DTD hierfür aus?

```
<BookStore>  
  <Book>  
    <Title>My Life and Times</Title>  
    <Author>Paul McCartney</Author>  
    <Date>July, 1998</Date>  
    <ISBN>94303-12021-43892</ISBN>  
    <Publisher>McMillin Publishing</Publisher>  
  </Book>  
</BookStore>
```

- BookStore soll mindestens ein Buch enthalten.
- ISBN optional
- alle anderen Kind-Elemente obligatorisch

Die DTD für das Beispiel-Dokument

```
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

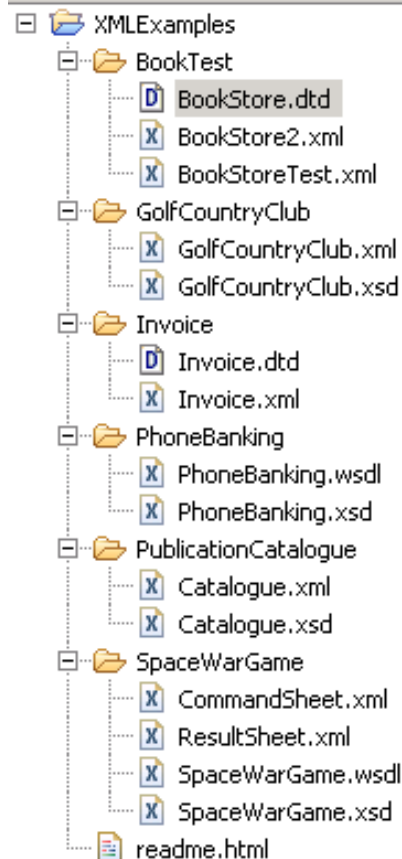
ähnelt einer
regulären
Grammatik

<!ELEMENT *Name Content-Modell*>

Element-Deklaration



Package Explorer Hierarchy



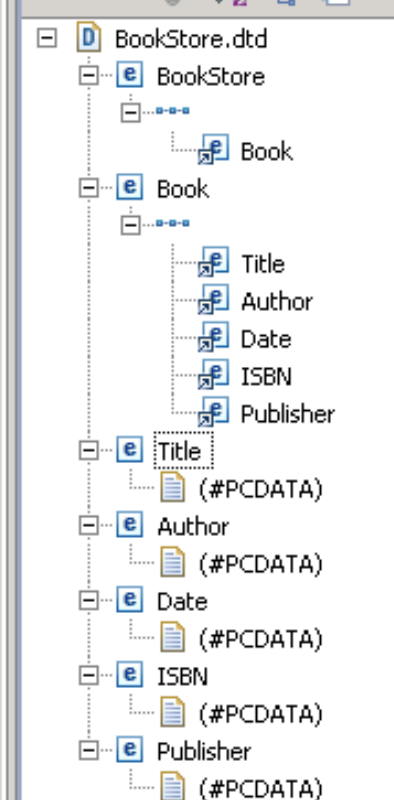
BookStore.dtd BookStoreTest.xml BookStore2.xml

```
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

Task List

Find: All

Outline



Problems Javadoc Declaration Properties





DTDs: Element-Deklaration

Datentypen für Element-Inhalte

verschiedene Datentypen:

1. **Element**: Element mit speziellen Symbolen + * | ?
2. **#PCDATA**: unstrukturierter Inhalt ohne reservierte Symbole < und &.

<!ELEMENT Title (#PCDATA)>

2. **EMPTY**: leerer Inhalt, Element kann aber Attribute haben

<!ELEMENT br EMPTY> →

3. **ANY**: beliebiger Inhalt (strukturiert, unstrukturiert, gemischt oder leer)

<!ELEMENT title ANY>

Datentypen wie **INTEGER** oder **FLOAT** stehen nicht zur Verfügung.

```
<!ELEMENT BookStore (Book+)>
```

```
<BookStore>  
  <Book>...</Book>  
  <Book>...</Book>  
</BookStore>
```

- + bezeichnet n Wiederholungen mit $n > 0$.
- * bezeichnet n Wiederholungen mit $n \geq 0$.
- BookStore hat mindestens ein Kind-Element Book
- Außer Book darf BookStore keine anderen Kind-Elemente haben.



Package Explorer

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - GolfCountryClub
 - GolfCountryClub.xml
 - GolfCountryClub.xsd
 - Invoice
 - Invoice.dtd
 - Invoice.xml
 - PhoneBanking
 - PhoneBanking.wsdl
 - PhoneBanking.xsd
 - PublicationCatalogue
 - Catalogue.xml
 - Catalogue.xsd
 - SpaceWarGame
 - CommandSheet.xml
 - ResultSheet.xml
 - SpaceWarGame.wsdl
 - SpaceWarGame.xsd
 - readme.html

readme Invoice.xml BookStore.dtd *BookStoreTest.xml

```

?? xml
DOCTYPE
BookStore
  Book
  Book
  
```

version="1.0" encoding="UTF-8"
 BookStore SYSTEM "BookStore.dtd"
 (Book+)
 (Title, Author, Date, ISBN?, Publisher)
 (Title, Author, Date, ISBN?, Publisher)

Design Source

Task List

Find: All

Uncategorized

Outline

```

?? xml
DOCTYPE:BookStore
BookStore
  Book
    Title
    Author
    Date
    ISBN
    Publisher
  Book
  
```

```
<!ELEMENT BookStore (Book | (Book, BookStore))>
```

- Bookstore besteht aus genau einer der folg. Alternativen:
 - genau ein Kind-Element Book
 - zwei Kind-Elemente: Book und BookStore
- | bezeichnet **Auswahl**: genau eine der beiden Alternativen
- , bezeichnet **Sequenz** von Elementen.
- Beachte: Rekursive Deklaration nicht äquivalent zur vorherigen, iterativen Definition!

Rekursive vs. iterative Deklaration

```
<BookStore>
  <Book>...</Book>
  <Book>...</Book>
  <Book>...</Book>
</BookStore>
```

BookStore mit 3 Büchern

```
<!ELEMENT BookStore (Book+)>
```

```
<BookStore>
  <Book>...</Book>
  <BookStore>
    <Book>...</Book>
    <BookStore>
      <Book>...</Book>
    </BookStore>
  </BookStore>
</BookStore>
```

BookStore mit 3 Büchern

```
<!ELEMENT BookStore (Book | (Book, BookStore))>
```

```
<!ELEMENT BookStore (Book+)>
```

```
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
```

```
<!ELEMENT Title (#PCDATA)>
```

```
<!ELEMENT Author (#PCDATA)>
```

```
<!ELEMENT Date (#PCDATA)>
```

```
<!ELEMENT ISBN (#PCDATA)>
```

```
<!ELEMENT Publisher (#PCDATA)>
```

```
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher) >
```

- Title, Author, Date, ISBN und Publisher (in dieser Reihenfolge) Kind-Elemente von Book.
- außer diesen keine anderen Kind-Elemente

- ? bedeutet optional

```
<Book >  
  <Title>...</Title>  
  <Author>...</Author>  
  <Date>...</Date>  
  <ISBN>...</ISBN>  
  <Publisher>...</Publisher>  
</Book >
```



Package Explorer Hierarchy

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - GolfCountryClub
 - GolfCountryClub.xml
 - GolfCountryClub.xsd
 - Invoice
 - Invoice.dtd
 - Invoice.xml
 - PhoneBanking
 - PhoneBanking.wsdl
 - PhoneBanking.xsd
 - PublicationCatalogue
 - Catalogue.xml
 - Catalogue.xsd
 - SpaceWarGame
 - CommandSheet.xml
 - ResultSheet.xml
 - SpaceWarGame.wsdl
 - SpaceWarGame.xsd
 - readme.html

BookStore.dtd BookStoreTest.xml

xml	version="1.0" encoding="UTF-8"
DOCTYPE	BookStore SYSTEM "BookStore.dtd"
BookStore	(Book+)
Book	(Title, Author, Date, ISBN?, Publisher)
Title	My Life and Times
Author	Paul McCartney
Date	July, 1998
ISBN	94303-12021-43892
Publisher	McMillin Publishing
Book	(Title, Author, Date, ISBN?, Publisher)

Design Source

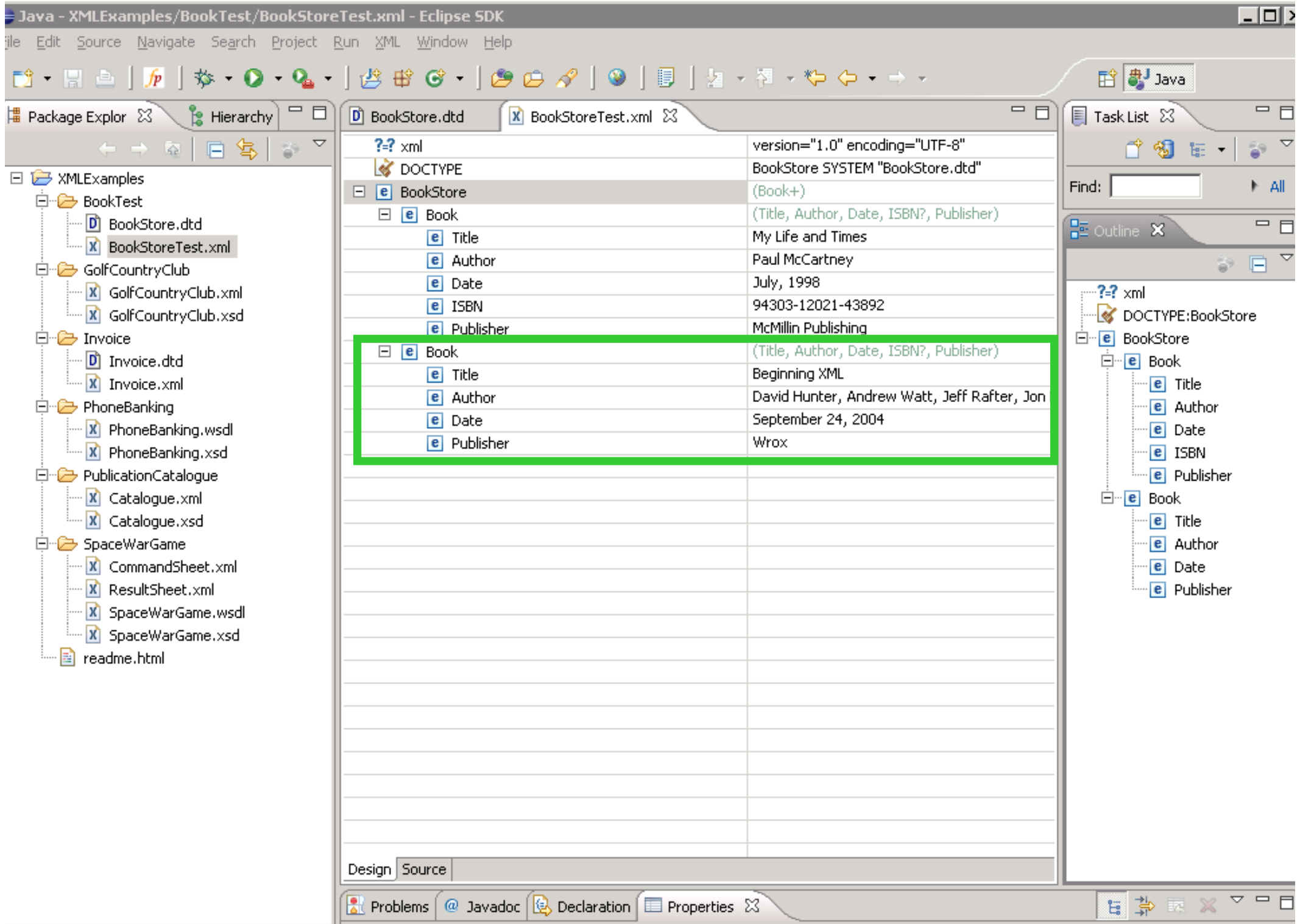
Task List

Find: All

Outline

- xml
 - DOCTYPE:BookStore
 - BookStore
 - Book
 - Title
 - Author
 - Date
 - ISBN
 - Publisher
 - Book

Problems Javadoc Declaration Properties



```
<!ELEMENT BookStore (Book+)>
```

```
<!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
```

```
<!ELEMENT Title (#PCDATA)>
```

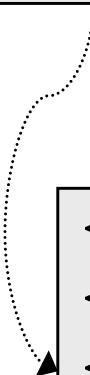
```
<!ELEMENT Author (#PCDATA)>
```

```
<!ELEMENT Date (#PCDATA)>
```

```
<!ELEMENT ISBN (#PCDATA)>
```

```
<!ELEMENT Publisher (#PCDATA)>
```

```
<!ELEMENT Title (#PCDATA) >  
<!ELEMENT Author (#PCDATA) >  
<!ELEMENT Date (#PCDATA) >  
<!ELEMENT ISBN (#PCDATA) >  
<!ELEMENT Publisher (#PCDATA) >
```



```
<Title>My Life and Times</Title>  
<Author>Paul McCartney</Author>  
<Date>July, 1998</Date>  
<ISBN>94303-12021-43892</ISBN>  
<Publisher>McMillin Publishing</Publisher>
```



Package Explorer Hierarchy

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - GolfCountryClub
 - GolfCountryClub.xml
 - GolfCountryClub.xsd
 - Invoice
 - Invoice.dtd
 - Invoice.xml
 - PhoneBanking
 - PhoneBanking.wsdl
 - PhoneBanking.xsd
 - PublicationCatalogue
 - Catalogue.xml
 - Catalogue.xsd
 - SpaceWarGame
 - CommandSheet.xml
 - ResultSheet.xml
 - SpaceWarGame.wsdl
 - SpaceWarGame.xsd
 - readme.html

BookStore.dtd BookStoreTest.xml

xml	version="1.0" encoding="UTF-8"
DOCTYPE	BookStore SYSTEM "BookStore.dtd"
BookStore	(Book+)
Book	(Title, Author, Date, ISBN?, Publisher)
Title	My Life and Times
Author	Paul McCartney
Date	July, 1998
ISBN	94303-12021-43892
Publisher	McMillin Publishing
Book	(Title, Author, Date, ISBN?, Publisher)

Design Source

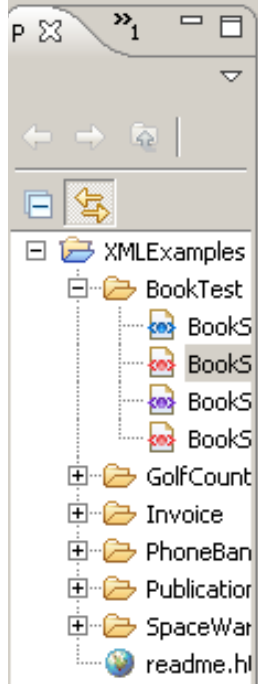
Task List

Find: All

Outline

- xml
 - DOCTYPE:BookStore
 - BookStore
 - Book
 - Title
 - Author
 - Date
 - ISBN
 - Publisher
 - Book

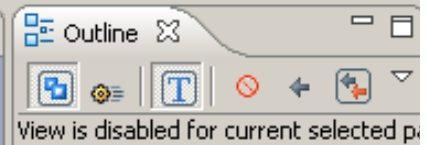
Problems Javadoc Declaration Properties



<?xml version="1.0" encoding="UTF-8"

<!DOCTYPE... ID SYSTEM "BookStore.dtd"

| BookStore | | Book (2 rows) | | | | | |
|-----------|--------|-------------------|--|---------------------|-------------------|---------------------|--|
| ID | SYSTEM | Title | Author | Date | ISBN | Publisher | |
| 1 | | My Life and Times | Paul McCartney | July, 1998 | 94303-12021-43892 | McMillin Publishing | |
| 2 | | Beginning XML | David Hunter, Andrew Watt, Jeff Rafter, Jon Duckett, Danny Ayers, Nicholas Chase, Joe Fawcett, Tom Gaven, Bill Patterson | Septem ber 24, 2004 | | Wrox | |



Text Grid Author

Verschachtelungen

- (fast) beliebige Verschachtelung von Sequenz, Auswahl |, ?, *, + und Rekursion erlaubt
- Beispiel:

```

<!ELEMENT Chap (Title, (Para | Chap)+)>
<!ELEMENT Para ANY>
<!ELEMENT Title (#PCDATA)>
    
```

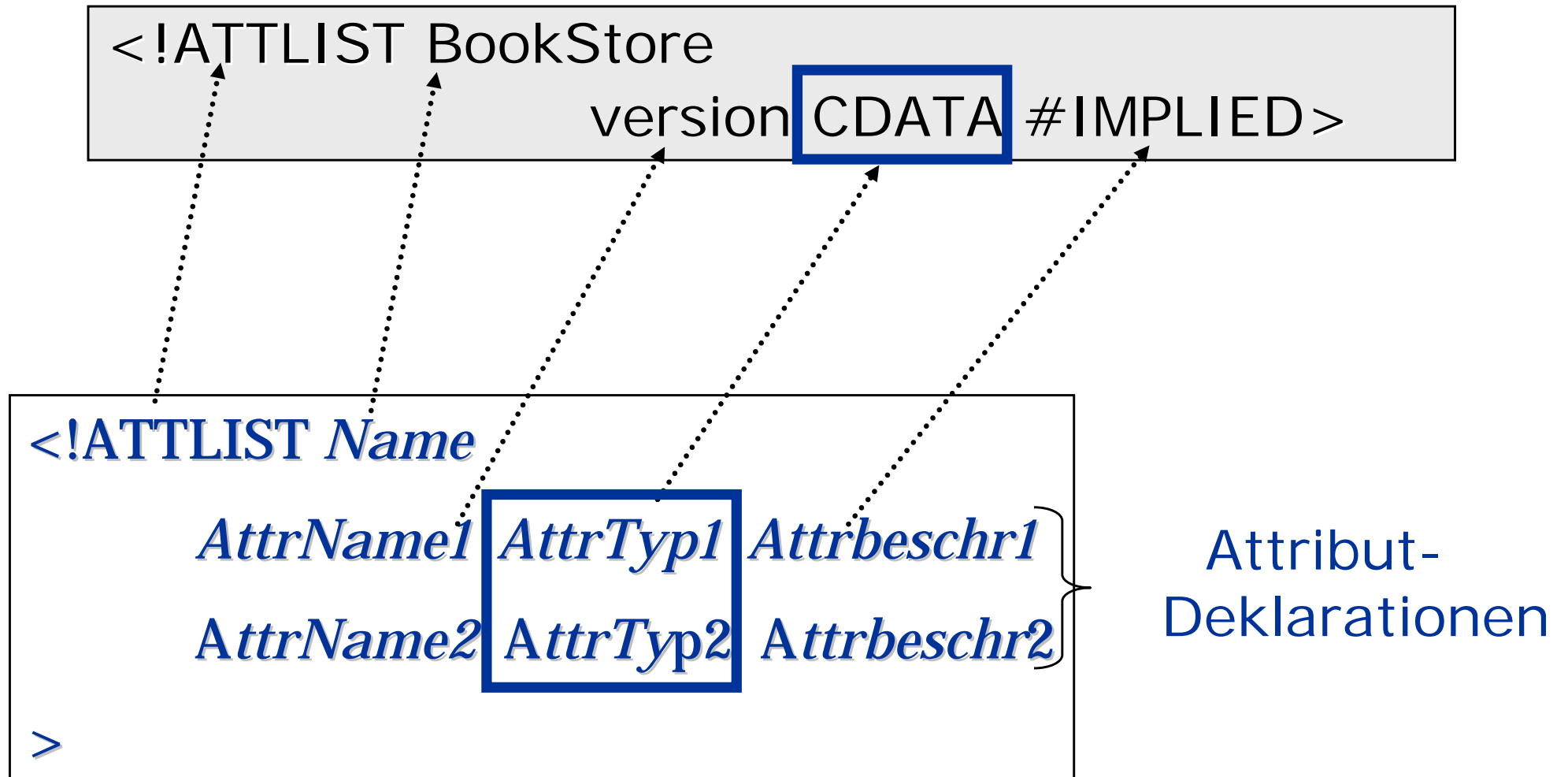
```

<Chap>
  <Title>Kap1</Title>
  <Para>Ein Absatz</Para>
  <Chap>
    <Title>Kap1.1</Title>
    <Para>...</Para>
  </Chap>
  <Para>...</Para>
  <Chap>
    <Title>Kap1.2</Title>
    <Para>...</Para>
  </Chap>
</Chap>
    
```

- Beispiel: $((b, c) \mid (b, d))$ ist **nicht-deterministisch**
- Grund: Wenn erstes Element = b , dann kann XML-Prozessor keine der beiden Alternativen ausschließen.
- XML erlaubt nur **deterministische** Content Modelle
- jedes nicht-deterministische Content Modell kann in ein äquivalentes deterministisches umgeformt werden
- Beispiel: $((b, c) \mid (b, d)) = (b, (c \mid d))$



DTDs: Attribut-Deklaration



```
<!ATTLIST BookStore  
        version CDATA #IMPLIED>
```

```
<BookStore version="1.0">  
    ...  
</BookStore>
```

- BookStore hat Attribut version.
- Außer version hat BookStore keine weiteren Attribute.
- **CDATA**: Attribut-Wert = String ohne <, & und ' bzw. „
- Beachte: nicht verwechseln mit <![CDATA[...]]>
- daher Entity References für <, & und ' bzw. " verwenden



```
<!ATTLIST Author  
          gender (male | female) "female">
```

- hier statt CDATA **Aufzählungstyp**:
- Attribut gender hat entweder Wert male oder female.
- "female" ist Standard-Wert von gender.

Zusätzlich zu CDATA (Strings) und Aufzählungstypen:

- **NMTOKEN**: String, der Namenskonventionen von XML entspricht
- **ID**: eindeutiger Bezeichner, der Namenskonventionen von XML entspricht
- **IDREF**: Referenz auf einen eindeutigen Bezeichner

```
<!ATTLIST Author  
    key ID #IMPLIED  
    keyref IDREF #IMPLIED>
```

- Wert des Attributes `key` muss eindeutig sein:
Zwei Attribute vom Typ `ID` dürfen niemals gleichen Wert haben.
- Wert des Attributes `keyref` muss gültige Referenz sein:
Wert von `keyref` muss Wert eines Attributes vom Typ `ID` sein.

```
<!ATTLIST BookStore  
version CDATA #IMPLIED >
```

```
<!ATTLIST Name  
AttrName1 AttrTyp1 Attrbeschr1  
AttrName2 AttrTyp2 Attrbeschr2  
>
```

Attribut-
Deklarationen

```
<!ATTLIST BookStore  
        version CDATA #FIXED "1.0" >
```

- **#FIXED**: Attribut hat immer den gleichen Wert.
- **#IMPLIED**: Attribut optional
- **#REQUIRED**: Attribut obligatorisch

```
<BookStore>
  <Book>
    <Title>Text</Title>
    <Author key="k1">Text</Author>
    <Date>Text</Date>
    <Publisher pkey="p1">Text</Publisher>
  </Book>
  <Book>
    <Title>Text</Title>
    <Author keyref="k1"/>
    <Date>Text</Date>
    <Publisher pkey="p1">Text</Publisher>
  </Book>
</BookStore>
```

Wert **k1** muss eindeutig sein:
kein anderes Attribut vom Typ
ID darf diesen Wert haben.

Referenz **k1** muss existieren:
ein Attribut vom Typ ID muss
den Wert **k1** haben.



DTDs: Deklaration von Dokument-Typ

- vollständige DTD im XML-Dokument → **interne DTD**

`<!DOCTYPE Wurzel-Element [...] >`

- ein Verweis auf eine DTD im XML-Dokument → **externe DTD**

`<!DOCTYPE Wurzel-Element SYSTEM "DTD" >`

oder

`<!DOCTYPE Wurzel-Element PUBLIC "DTD" "URL" >`

- Dokument-Typ direkt nach XML-Deklaration einfügen

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BookStore [
  <!ELEMENT BookStore (Book+)>
  <!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
  <!ELEMENT Title (#PCDATA)>
  <!ELEMENT Author (#PCDATA)>
  <!ELEMENT Date (#PCDATA)>
  <!ELEMENT ISBN (#PCDATA)>
  <!ELEMENT Publisher (#PCDATA)>
]>
<BookStore>
  ...
</BookStore>
```



Package Hierarchy

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - BookStorewithDTD.xml
 - GolfCountryClub
 - GolfCountryClub.xml
 - GolfCountryClub.xsd
 - Invoice
 - Invoice.dtd
 - Invoice.xml
 - PhoneBanking
 - PhoneBanking.wsdl
 - PhoneBanking.xsd
 - PublicationCatalogue
 - Catalogue.xml
 - Catalogue.xsd
 - SpaceWarGame
 - CommandSheet.xml
 - ResultSheet.xml
 - SpaceWarGame.wsdl
 - SpaceWarGame.xsd
 - readme.html

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BookStorewithDTD [
  <!ELEMENT BookStorewithDTD (Book+)>
  <!ELEMENT Book (Title, Author, Date, ISBN?, Publisher)>
  <!ELEMENT Title (#PCDATA)>
  <!ELEMENT Author (#PCDATA)>
  <!ELEMENT Date (#PCDATA)>
  <!ELEMENT ISBN (#PCDATA)>
  <!ELEMENT Publisher (#PCDATA)>
]>
<BookStorewithDTD>
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
</BookStorewithDTD>
```

Task List

Find: All

Outline

- xml
 - DOCTYPE:BookStorewithDTD
 - BookStorewithDTD
 - Book
 - Title
 - Author
 - Date
 - ISBN
 - Publisher

`<!DOCTYPE Wurzel-Element SYSTEM " DTD" >`

```
<?xml version="1.0" encoding="UTF-8" ?>  
<!DOCTYPE BookStore SYSTEM "Bookstore.dtd" >  
<BookStore>  
...  
</BookStore>
```

**Dokument-Typ
Deklaration**

```
<?xml version="1.0" encoding="UTF-16" ?>  
<!DOCTYPE Book SYSTEM "Book.dtd" >  
<Book>  
...  
</Book>
```



Package Explorer

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - GolfCountryClub
 - GolfCountryClub.xml
 - GolfCountryClub.xsd
 - Invoice
 - Invoice.dtd
 - Invoice.xml
 - PhoneBanking
 - PhoneBanking.wsdl
 - PhoneBanking.xsd
 - PublicationCatalogue
 - Catalogue.xml
 - Catalogue.xsd
 - SpaceWarGame
 - CommandSheet.xml
 - ResultSheet.xml
 - SpaceWarGame.wsdl
 - SpaceWarGame.xsd
 - readme.html

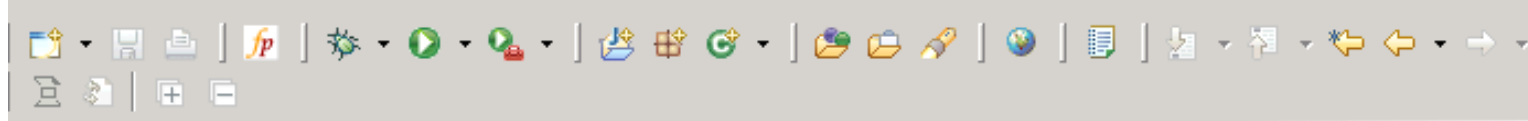
xml	version="1.0" encoding="UTF-8"
DOCTYPE	BookStore SYSTEM "BookStore.dtd"
BookStore	(Book+)
Book	(Title, Author, Date, ISBN?, Publisher)
Title	My Life and Times
Author	Paul McCartney
Date	July, 1998
ISBN	94303-12021-43892
Publisher	McMillin Publishing
Book	(Title, Author, Date, ISBN?, Publisher)
Title	Beginning XML
Author	David Hunter, Andrew Watt, Jeff Rafter, Jon D...
Date	September 24, 2004
Publisher	Wrox

Task List

Find: All

Outline

- xml
 - DOCTYPE:BookStore
 - BookStore
 - Book
 - Title
 - Author
 - Date
 - ISBN
 - Publisher
 - Book
 - Title
 - Author
 - Date
 - Publisher



Package Hierarchy Navigator

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - BookStorewithDTD.xml
 - GolfCountryClub
 - GolfCountryClub.xml
 - GolfCountryClub.xsd
 - Invoice
 - Invoice.dtd
 - Invoice.xml
 - PhoneBanking
 - PhoneBanking.wsdl
 - PhoneBanking.xsd
 - PublicationCatalogue
 - Catalogue.xml
 - Catalogue.xsd
 - SpaceWarGame
 - CommandSheet.xml
 - ResultSheet.xml
 - SpaceWarGame.wsdl
 - SpaceWarGame.xsd
 - readme.html

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE BookStore SYSTEM "BookStore.dtd" >
<BookStore>
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
  <Book>
    <Title>Beginning XML</Title>
    <Author>David Hunter, Andrew Watt, Jeff Rafter, Jon Duc
    <Date>September 24, 2004</Date>
    <Publisher>Wrox</Publisher>
  </Book>
</BookStore>

```

Task List

Find: All

Outline

- xml
 - DOCTYPE:BookStore
 - BookStore
 - Book
 - Title
 - Author
 - Date
 - ISBN
 - Publisher
 - Book
 - Title
 - Author
 - Date
 - Publisher



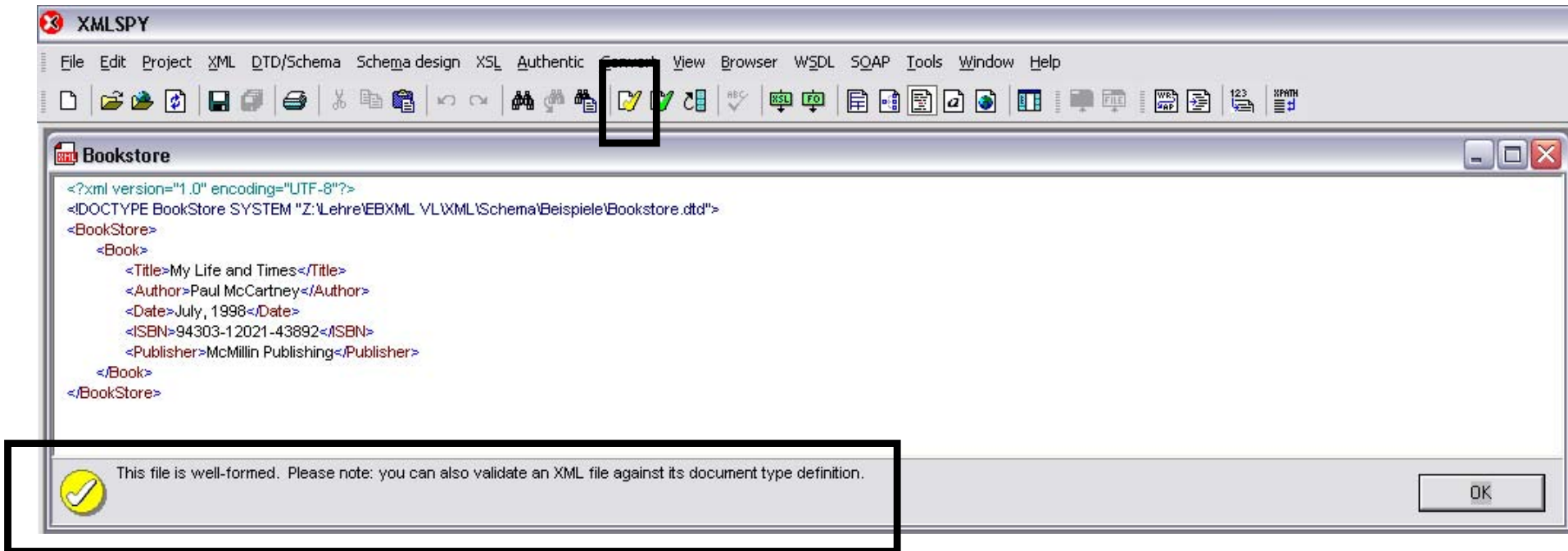
DTDs: Wohlgeformt & zulässig

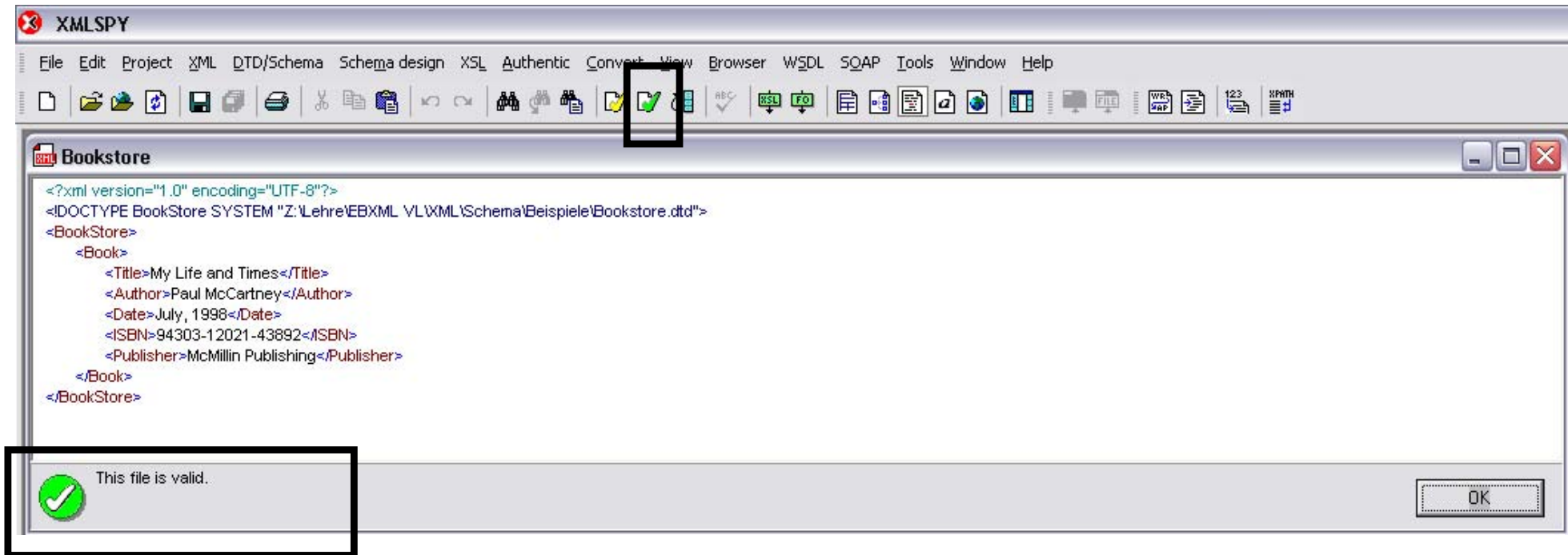
wohlgeformt (well formed)

- XML-Dokument entspricht Syntaxregeln von XML

zulässig (valid) bzgl. einer DTD

1. Wurzel-Element des XML-Dokumentes ist in der DTD deklariert und
2. Wurzel-Element hat genau die Struktur, wie sie in der DTD festgelegt ist.





Package Ex Hierarchy

- XMLExamples
 - BookTest
 - BookStore.dtd
 - BookStoreTest.xml
 - BookStoreTest.xsd
 - BookStorewithDTD.xml
 - GolfCountryClub
 - Invoice
 - PhoneBanking
 - PublicationCatalogue
 - SpaceWarGame
 - readme.html

*BookStoreTest.xml BookStoreTest.xsd

```
<?xml version="1.0" encoding="UTF-8">
```

<!DOCTYPE... ID SYSTEM "BookStore.dtd"

| BookStore | Book | Title | Author |
|-----------|----------|---------------------|---|
| | (2 rows) | 1 My Life and Times | Paul McCartney |
| | | 2 Beginning XML | David Hunter,
Andrew Watt,
Jeff Rafter,
Jon Duckett,
Danny Ayers,
Nicholas Chase,
Joe Fawcett,
Tom Gaven, Bill |

Text Grid Author

Outline

View is disabled for current selected p...

Problems Javadoc Declaration Properties Results Console

Oxygen messages

```
[17:03:17] - DTD XML Error Scanner - start scanning file:/Z:/ws_eclipse/XMLExamples/Book
[17:03:17] - Found 0 problem(s)
[17:03:26] - DTD XML Error Scanner - start scanning file:/Z:/ws_eclipse/XMLExamples/Book
[17:03:26] - Found 0 problem(s)
[17:03:31] - XMLSchema XML Error Scanner - start scanning file:/Z:/ws_eclipse/XMLExample
[17:03:31] - Found 0 problem(s)
```



Nachteile von DTDs

- keine XML-Syntax, eigener Parser nötig
- nur sehr wenige Datentypen, insbesondere für Element-Inhalte
- keine eigenen Datentypen definierbar
- keine Namensräume:
DTDs können nur dann kombiniert werden, wenn es keine Namenskonflikte gibt!
- keine Vererbungshierarchien

Und noch ein Nachteil

- Sequenzen einfach zu definieren:
<!ELEMENT Book (Title, Author) >
⇒ starre Struktur in XML-Dokumenten
- soll Reihenfolge der Kind-Elemente egal sein,
müssen alle Permutationen explizit aufgezählt
werden:
<!ELEMENT Book ((Title, Length) | (Length, Title)) >
- nicht praktikabel: bei n Kind-Elementen $n!$
Permutationen

DTD deklariert das erlaubte Vokabular

DTD definiert für jedes Element ein Content-Modell

- Content-Modell legt fest:
 - Elemente oder Daten
 - Reihenfolge & Anzahl von Elementen/Daten innerhalb eines Elements
 - Pflicht oder optionale Elemente

DTD deklariert für jedes Element eine Menge von erlaubten Attributen



XML-Schema

Warum XML-Schema?

```
<location>
  <latitude>32.904237</latitude>
  <longitude>73.620290</longitude>
  <uncertainty units="meters">2</uncertainty>
</location>
```

XML-Schema

DTD

- Ortsangabe: besteht aus Breitengrad, Längengrad und Unsicherheit.
- Breitengrad: Dezimalzahl zwischen -90 und +90
- Längengrad: Dezimalzahl zwischen -180 und +180
- Unsicherheit: nicht-negative Zahl
- Maßeinheit für Unsicherheit: Meter oder Fuß

Vorteile von XML-Schemata

- + Vielzahl von vordefinierten Datentypen
- + eigene Datentypen definierbar
- + keine eigene Syntax, sondern selbst XML-Sprache
- + Vererbungshierarchien
- + unterstützen Namensräume
- + Reihenfolgeunabhängige Strukturen einfach zu definieren

```
<!ELEMENT BookStore (Book+)>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

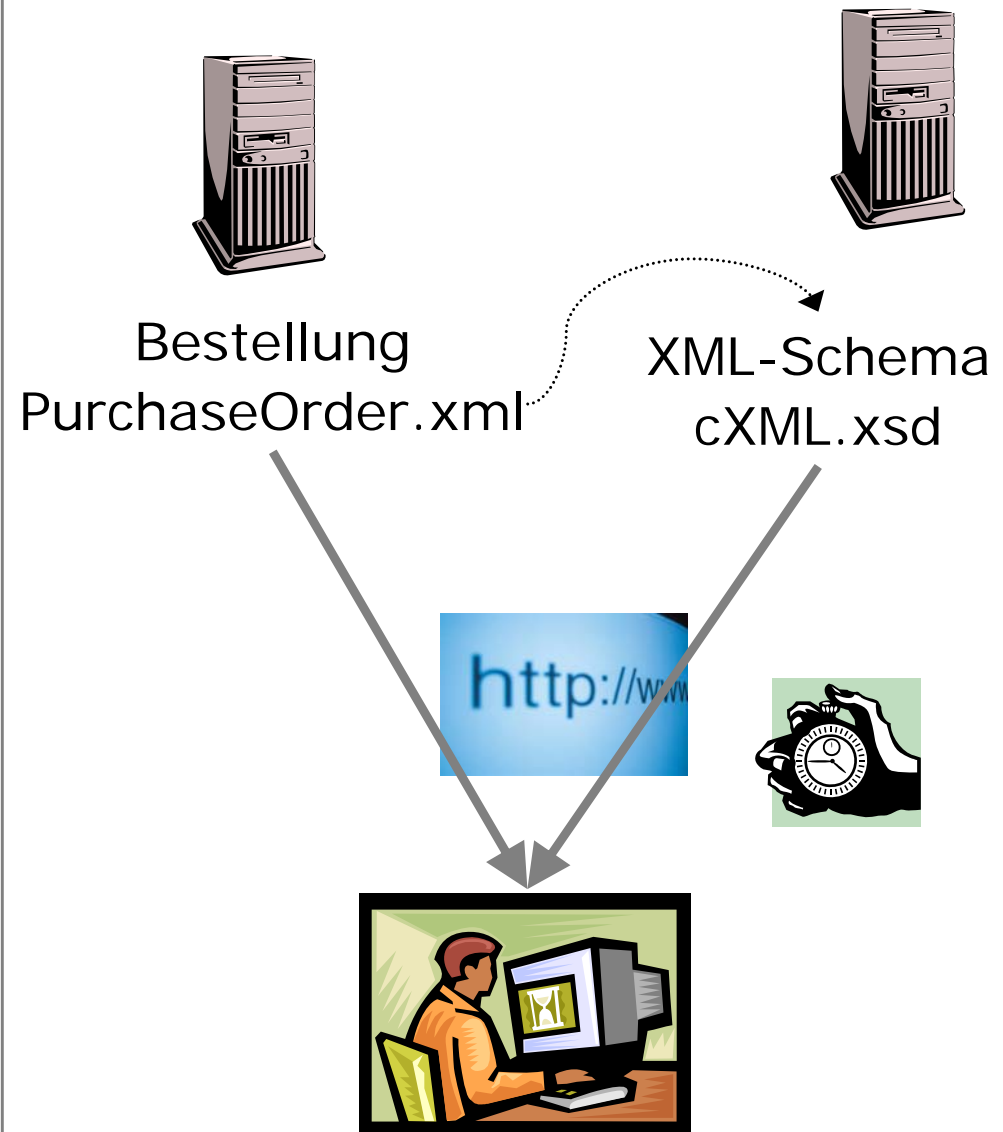
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.books.org">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="string"/>
              <xsd:element name="Author" type="string"/>
              <xsd:element name="Date" type="string"/>
              <xsd:element name="ISBN" type="string"/>
              <xsd:element name="Publisher" type="string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

- Für jede DTD gibt es ein äquivalentes XML-Schema.
 - Übersetzung z.B. mit XMLSpy
 - Umgekehrt gibt es jedoch XML-Schemata, für die es keine äquivalente DTD gibt.
- ➔ XML-Schemata ausdrucksmächtiger als DTDs

Allgemeines über XML Schema

- XML-Schemata sind wohlgeformte XML-Dokumente
 - Vorteil: kein eigener Parser nötig
 - Nachteil: geschwätzig
- Wurzelement jedes XML-Schemas ist das Element `<xsd:schema>`
- Alle Definitionen eines Schemas sind Kindelemente von `<xsd:schema>` (direkt oder tiefer geschachtelt)
- Direkte Kindelemente von `<xsd:schema>` sind *globale* Elemente.
- Jedes XML-Schema ist eine eigenständige Datei

- PurchaseOrder.xml verweist auf cXML-Schema, das Client von anderen Server laden muss.
- cXML
 - als Schema: 50KB
 - als DTD: 15KB
- XML-Schemata aber besser zu komprimieren:
 - cXML-Schema: 6KB
 - cXML-DTD: 4KB



```
<?xml version="1.0"?>  
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>  
  ...  
</xsd:schema>
```

Namensraum für
das Schema Element

- Wurzel-Element: Schema aus W3C-Namensraum <http://www.w3.org/2001/XMLSchema>
- hier XML-Schema für XML-Schema hinterlegt: Schema der Schemata

```
<?xml version="1.0"?>
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.books.org" >
...
</xsd:schema>
```

- jedes XML-Schema definiert bestimmtes Vokabular (Elemente und Attribute)
- Dieses Vokabular wird einem Namensraum zugeordnet: **Ziel-Namensraum** (target namespace).
- Ziel-Namensraum wird wie jeder Namensraum mit URI identifiziert
- Definiertes Vokabular kann über URI identifiziert werden.



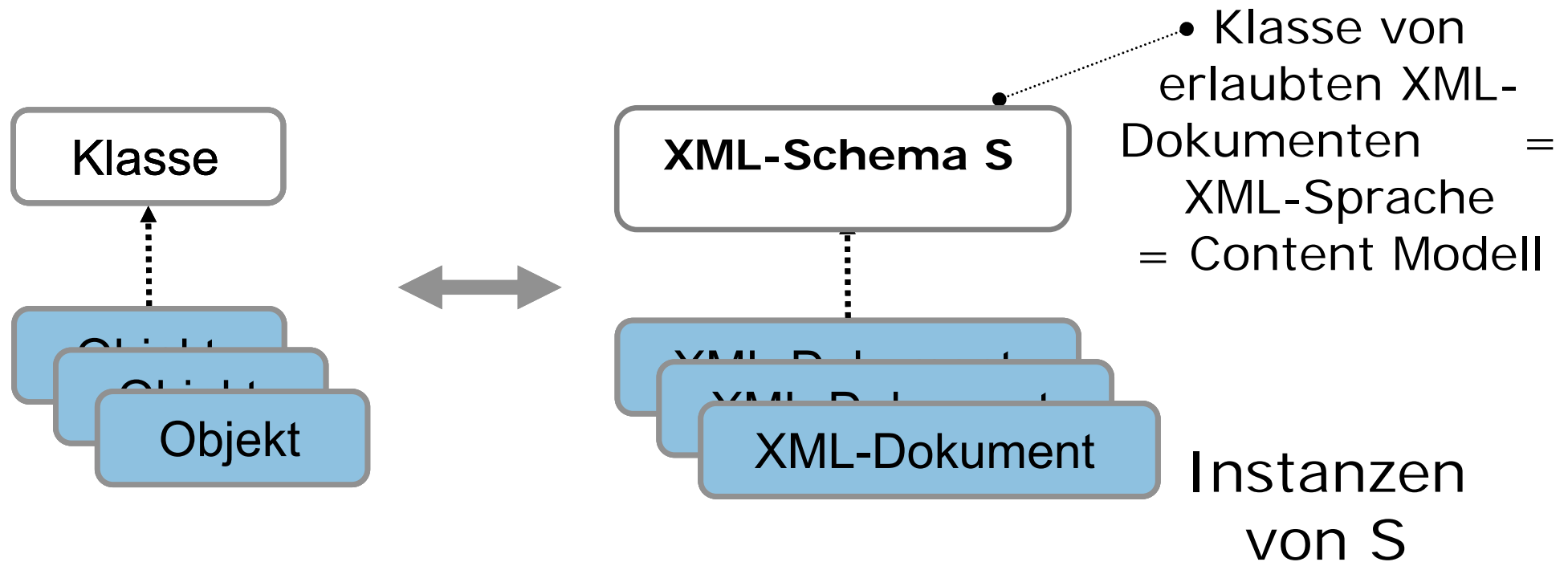
DTD definiert
keinen
Namensraum.

<http://www.books.org>



XML-Schema definiert
eigenen Namensraum, den
Ziel-Namensraum.

Instanz eines XML-Schemas



- Instanz eines XML-Schemas S** ist ein XML-Dokument, das
1. dem Ziel-Namensraum von S zugeordnet ist und
 2. gültig (valid) bzgl. S ist.

```
<?xml version="1.0"?>
<BookStore>
  <Book>
    <Title>My Life and Times</Title>
    <Author>Paul McCartney</Author>
    <Date>July, 1998</Date>
    <ISBN>94303-12021-43892</ISBN>
    <Publisher>McMillin Publishing</Publisher>
  </Book>
  ...
</BookStore>
```

Wie wird aus diesem XML-Dokument eine Instanz eines XML-Schemas?

1. Schritt

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.books.org
  BookStore.xsd" ...
</BookStore>
```

Ziel-Namensraum
des XML-Schemas

- Wurzel-Element und Namensraum legen zusammen Dokument-Typ fest.
- Wurzel-Element muss in XML-Schema global deklariert sein.
- Für bekannte Namensräume wie <http://www.w3.org/1999/xhtml> keine weiteren Schritte nötig.

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.books.org
  http://www.books.org/BookStore.xsd">
  ...
</BookStore>
```

kann auch lokale
Datei wie z.B.
BookStore.xsd sein

- Attribut `schemaLocation` gibt Hinweis, wo das XML-Schema zu finden ist.
- Beachte: XML-Prozessor darf diese Information ignorieren und anderes XML-Schema berücksichtigen!

3. Schritt

```
<?xml version="1.0"?>
<BookStore xmlns="http://www.books.org"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.books.org
                               http://www.books.org/BookStore.xsd">
    ...
</BookStore>
```

- Attribut `schemaLocation` stammt aus dem W3C-Namensraum für Schema-Instanzen.

- weiteres Beispiel für Erweiterbarkeit von XML!
- XML durch Attribut `schemaLocation` erweitert

Instanz

XML-Schema

schemaLocation="http://www.books.org
BookStore.xsd"

targetNamespace="http://www.books.org"

BookStore.xml

BookStore.xsd

benutzt Namensraum
http://www.books.org

definiert Namensraum
http://www.books.org

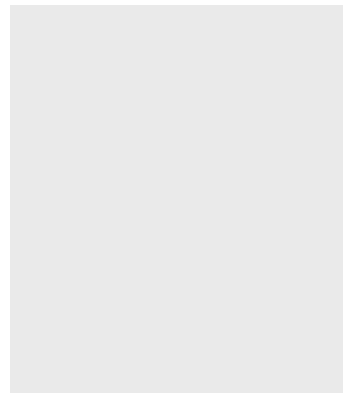
Software zum Validieren von Instanzen

- xerces by Apache (API)
 - <http://xml.apache.org/>
- MSXML (API)
 - <http://www.microsoft.com>
- XMLSpy (GUI)
 - <http://www.altova.com/>
- weitere: <http://www.w3.org/XML/Schema#Tools>

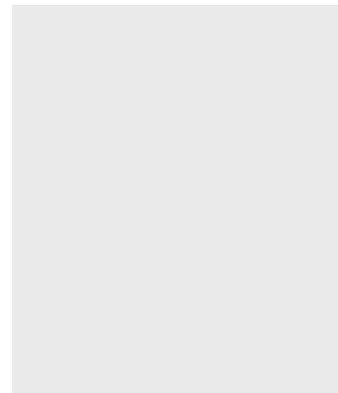
Validierung auf mehreren Ebenen

Instanz
= XML-Dokument

XML-Schema



BookStore.xml



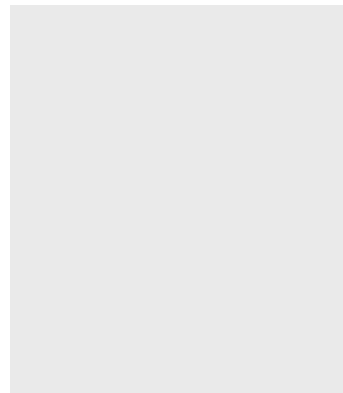
BookStore.xsd



zulässiges BookStore-
Dokument?

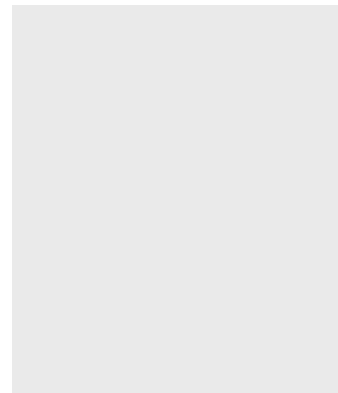
Validierung auf mehreren Ebenen

Instanz
= XML-Dokument



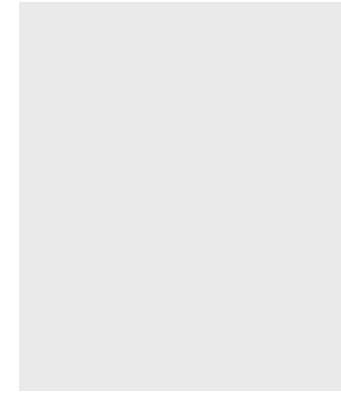
BookStore.xml

XML-Schema
= **XML-Dokument**



BookStore.xsd

Schema der
Schemata



XMLSchema.xsd

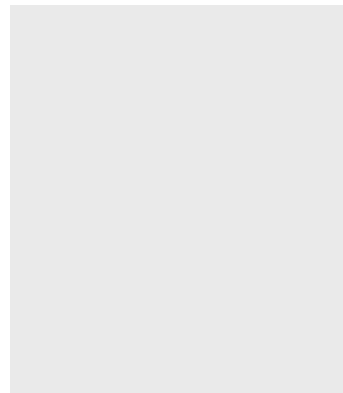


zulässiges BookStore-Dokument?

zulässiges XML-Schema?

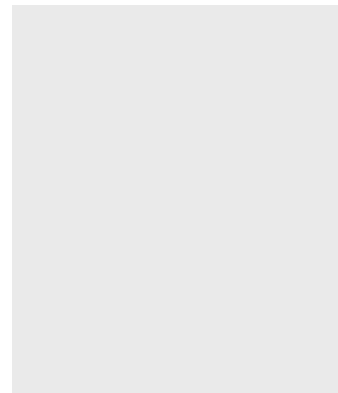
Validierung auf mehreren Ebenen

Instanz
= XML-Dokument



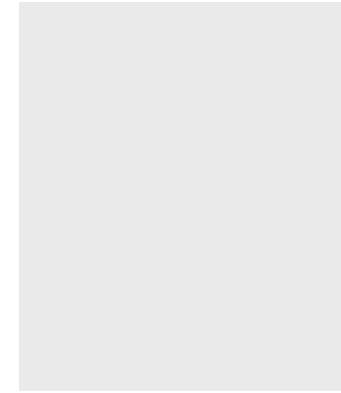
BookStore.xml

XML-Schema
= XML-Dokument



BookStore.xsd

Schema der
Schemata =
XML-Dokument



XMLSchema.xsd



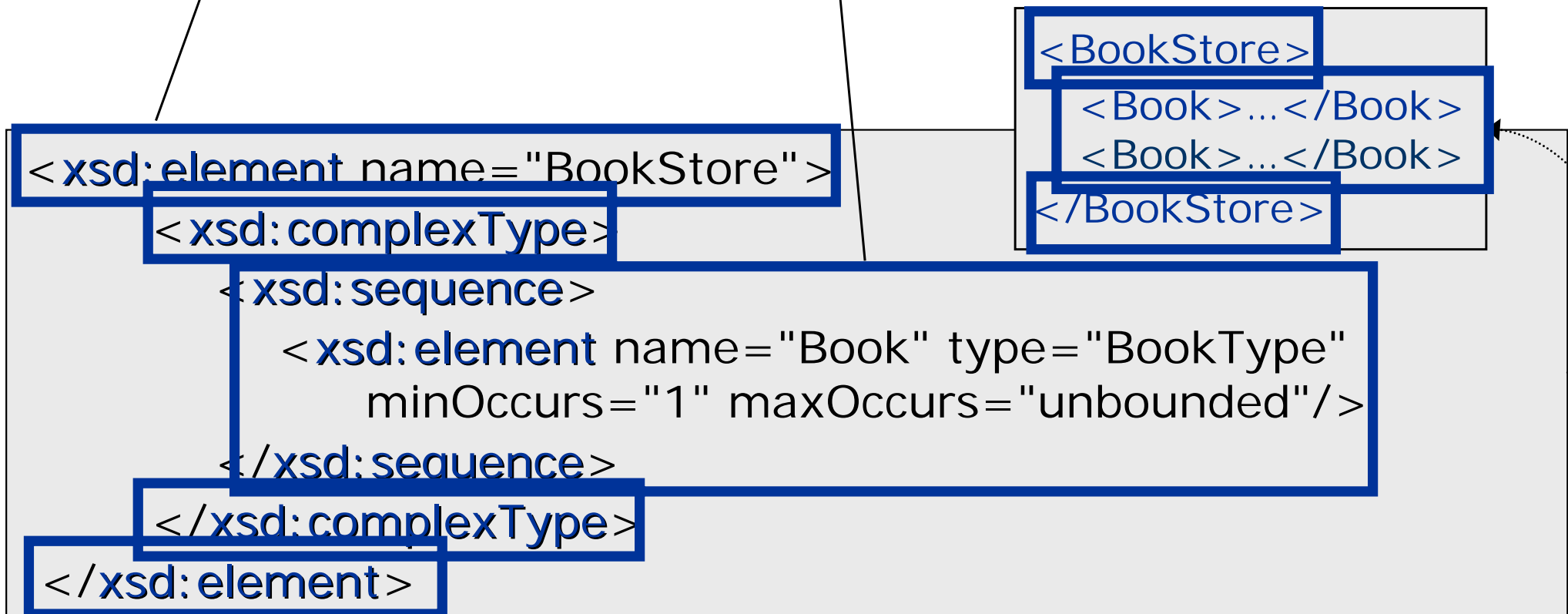
zulässiges BookStore-Dokument?

zulässiges XML-Schema?

```
<!ELEMENT BookStore (Book+)>  
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>  
<!ELEMENT Title (#PCDATA)>  
<!ELEMENT Author (#PCDATA)>  
<!ELEMENT Date (#PCDATA)>  
<!ELEMENT ISBN (#PCDATA)>  
<!ELEMENT Publisher (#PCDATA)>
```

- Wie werden diese Element-Deklarationen mit einem XML-Schema ausgedrückt?

<!ELEMENT BookStore (Book+)>



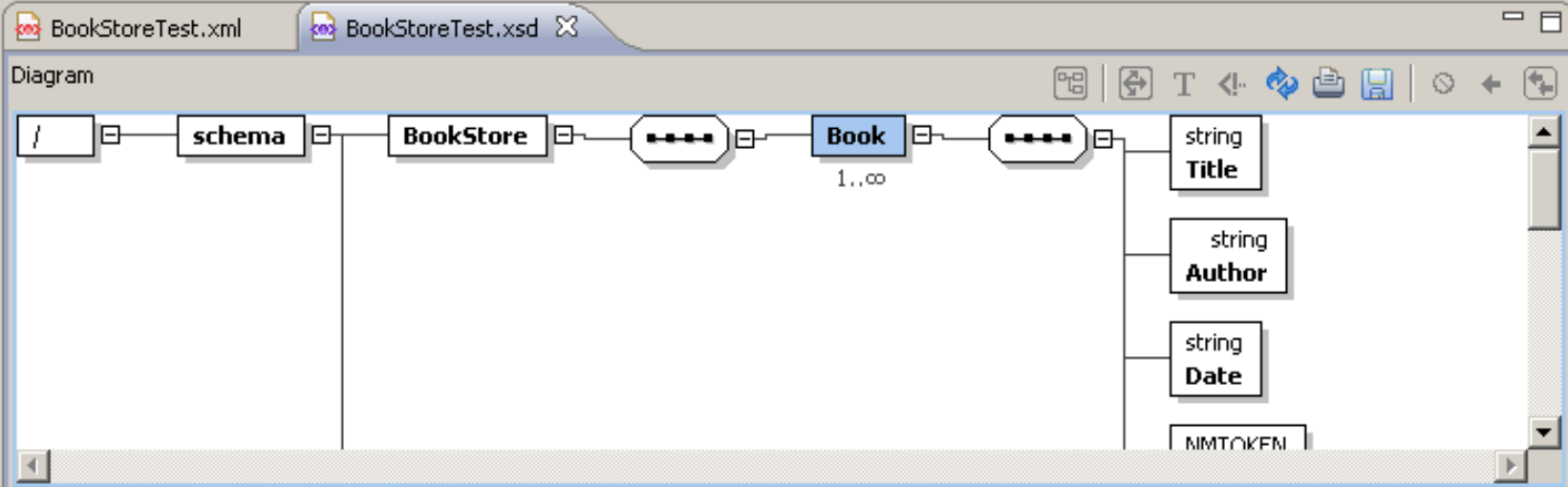
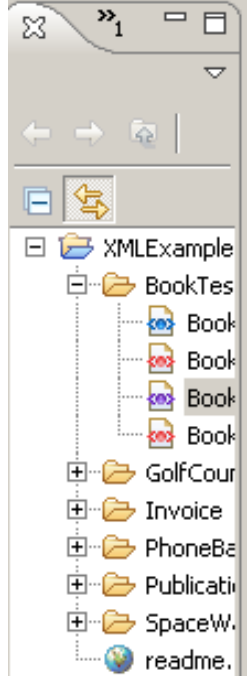
- `xsd:element`: Element wird deklariert
- `xsd:complexType`: strukturierter Inhalt
- `xsd:sequence`: Sequenz von Elementen

<!ELEMENT BookStore (Book₊)>

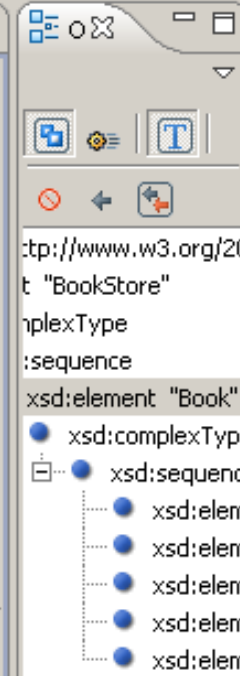
```
<xsd:element name="BookStore">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Book" type="BookType"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<BookStore>
  <Book>...</Book>
  <Book>...</Book>
</BookStore>
```

- **minOccurs**: minimale Anzahl der Wiederholungen
- **maxOccurs**: maximale Anzahl der Wiederholungen
- Beachte: Standard-Werte für minOccurs und maxOccurs jeweils 1



```
<xsd:element name="BookStore">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Book" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:string name="Title" type="string"/>
            <xsd:string name="Author" type="string"/>
            <xsd:string name="Date" type="string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
</xsd:schema>
```



```
<xsd:complexType name="BookType">  
  <xsd:sequence>  
    <xsd:element name="Title" type="xsd:string"/>  
    <xsd:element name="Author" type="xsd:string"/>  
    <xsd:element name="Date" type="xsd:string"/>  
    <xsd:element name="ISBN" type="xsd:string"/>  
    <xsd:element name="Publisher" type="xsd:string"/>  
  </xsd:sequence>  
</xsd:complexType>
```

- Kind-Elemente: Title, Author, Date, ISBN und Publisher
- wegen `xsd:sequence`: feste Reihenfolge
- jeweils genau einmal

```
<Book>  
  <Title>...</Title>  
  <Author>...</Author>  
  <Date>...</Date>  
  <ISBN>...</ISBN>  
  <Publisher>...</Publisher>  
</Book>
```

```
<xsd:complexType name="BookType">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string"/>
    <xsd:element name="Date" type="xsd:string"/>
    <xsd:element name="ISBN" type="xsd:string"/>
    <xsd:element name="Publisher" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

- **xsd:string**: vordefinierter Datentyp
- 43 weitere vordefinierte Datentypen

```
<Book>
  <Title> PCDATA </Title>
  <Author>...</Author>
  <Date>...</Date>
  <ISBN>...</ISBN>
  <Publisher>...</Publisher>
</Book>
```

```
<xsd:complexType name="BookType">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string"/>
    <xsd:element name="Date" type="xsd:date"/>
    <xsd:element name="ISBN" type="xsd:string"/>
    <xsd:element name="Publisher" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

- **xsd:date: vordefinierter Datentyp**

```
<Book>
  <Title> PCDATA </Title>
  <Author> </Author>
  <Date> Kalenderdatum </Date>
  <ISBN>...</ISBN>
  <Publisher>...</Publisher>
</Book>
```

```
<xsd:element name="Book" type="BookType"
  minOccurs="1" maxOccurs="unbounded"/>
```

```
<xsd:complexType name="BookType" >
  <xsd:sequence >
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string"/>
    <xsd:element name="Date" type="xsd:string"/>
    <xsd:element name="ISBN" type="xsd:string"/>
    <xsd:element name="Publisher" type="xsd:string"/>
  </xsd:sequence >
</xsd:complexType >
```

- BookType hier benannter Datentyp (named type)
- auch globale Typ-Definition genannt

```
<xsd:element name="Book" minOccurs="1" maxOccurs="unbounded" >
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

- äquivalente Formulierung mit anonymen Datentyp
- auch als lokale Typ-Definition bezeichnet
- Nachteil: kann an anderer Stelle nicht wieder verwendet werden

<!ELEMENT Book (Title, Author+, Date, ISBN?, Publisher)>

```
<xsd:complexType name="BookType">
  <xsd:sequence>
    <xsd:element name="Title" type="xsd:string"/>
    <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded" />
    <xsd:element name="Date" type="xsd:string"/>
    <xsd:element name="ISBN" type="xsd:string" minOccurs="0" />
    <xsd:element name="Publisher" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

- Jedes Elemente erscheint so häufig, wie mit minOccurs und maxOccurs festgelegt.

Edit Navigate Search Project Run XSD Window Help

BookStoreTest.xml BookStoreTest.xsd

Diagram

```

    graph LR
      BookStore[BookStore 1..∞] --- S1(( ))
      S1 --- Book[Book 1..∞]
      Book --- S2(( ))
      S2 --- Title[Title 1..∞]
      S2 --- Author[Author 1..∞]
      S2 --- Date[Date 0..1]
      S2 --- ISBN[ISBN 0..1]
      S2 --- Publisher[Publisher]
  
```

XMLExample

- BookTes
 - Book
 - Book
 - Book
 - Book
- GolfCour
- Invoice
- PhoneBa
- Publicati
- SpaceW.
- readme.

Full Model View Logical Model View

```

<xsd:element name="BookStore">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Book" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="Title" type="xsd:string"/>
            <xsd:element name="Author" type="xsd:string" maxOccurs="unbounded"/>
            <xsd:element name="Date" type="xsd:string"/>
            <xsd:element name="ISBN" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Publisher" type="xsd:string"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
  
```

Text Grid

Problems Javadoc Declaration Properties Results Console

0 errors, 0 warnings, 0 infos

tion

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.books.org">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Lernziel: Dieses Schema und die entsprechende DTD verstehen!

DTD's	XML-Schema
vereinfachte SGML-DTD, Teil von XML 1.0/1.1	eigener W3C-Standard
eigene Syntax	XML-Schemata = XML-Sprache
kompakter und lesbarer	ausdrucksstärker
zur Beschreibung von Text- Dokumenten ausreichend	zur Beschreibung von Daten besser geeignet.

heute

- ☑ Definition von XML-Sprachen
- ☑ DTDs und XML-Schema anhand eines einheitlichen Beispiels

nächste Woche

- XML-Schema im Detail