Kurzeinführung in XML

Was ist XML?

Well-formed u. gültiges XML

Erste Anwendung in XML

Externe DTD

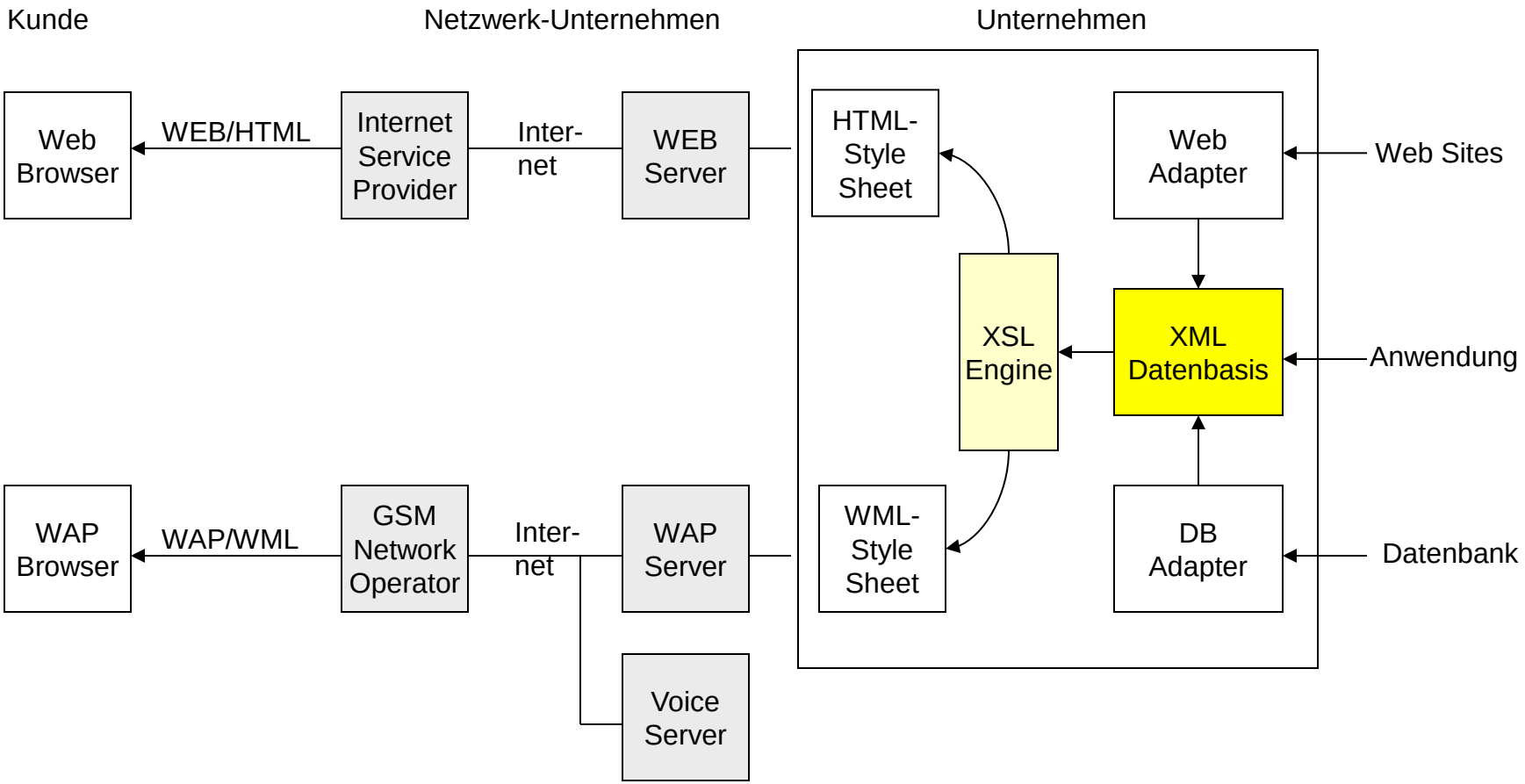
Attribute und Entities

Datenausgabe mit XSL

Was ist XML ?

- XML steht für Extensible Markup Language
- XML ist eine Untermenge von SGML(Standard Generalized Markup Language)
- XML ist ein System- und Anwendungsunabhängiges Datenformat
- XML bietet dem Programmierer eine Möglichkeit, strukturierte Dokumente in lesbarer Form zu erzeugen und verwalten, die auf vielfältiger Weise dargestellt werden können.
- Man kann eigene Tags und Attribute definieren (Diese Tags haben nichts mit der Darstellung am Bildschirm zu tun, sondern sind semantische Tags)
- In XML wird Layout und Inhalt getrennt

Was ist XML ? Ein Anwendungsbeispiel



Well-formed XML (I)

Wohlgeformt ("well-formed") heißt: Das Dokument entspricht den XML-Syntax-Regeln.

Der Prolog

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

Well-formed XML (II)

Der erste Tag

```
<?xml version="1.0"?>
<AUSGABE>Hallo Thomas!</AUSGABE>
```

Fragen:

Was passiert, wenn der Abschluß-Tag fehlt?

Wie wirkt sich Groß- und Kleinschreibung aus?

Gültiges XML (I)

Gültig bzw. validiert ("validated") heißt: Das Dokument entspricht einer DTD.

Tags definieren

```
<?xml version="1.0"?>
        <!DOCTYPE hallo [
        <!ELEMENT AUSGABE (#PCDATA)>
]>
<AUSGABE>Hallo Thomas !</AUSGABE>
```

Diese Definition des Tags innerhalb des Prologs wird als Document Type Definition (DTD) bezeichnet.

Gültiges XML (II)

XML benötigt ein Wurzelelement

```
<?xml version="1.0"?>
<!DOCTYPE hallo [
   <!ELEMENT AUSGABE (ANZEIGE)>
   <!ELEMENT ANZEIGE (#PCDATA)>
]>
<AUSGABE>
   <ANZEIGE>Hallo Thomas !</ANZEIGE>
</AUSGABE>
```

Gültiges XML (III)

Kommentare in XML

```
<!--Jetzt kommt der Kommentar -->
```

Erste Anwendung in XML (I)

Bsp. Adressbuch in XML

Datensatzfelder:

Nachname
Vorname
Strasse
Postleitzahl
Ort
Telefonnummer

Aufgabe 1

Schreibe den Programmcode des Adressbuches!

Erste Anwendung in XML (II)

Zuerst der Prolog mit der DTD:

```
<?xml version="1.0"?>
    <!DOCTYPE Adresse [
        <!ELEMENT Adresse (Nachname, Vorname, Strasse, PLZ, Ort,
    Telefon)>
        <!ELEMENT Nachname (#PCDATA)>
        <!ELEMENT Vorname (#PCDATA)>
        <!ELEMENT Strasse (#PCDATA)>
        <!ELEMENT PLZ (#PCDATA)>
        <!ELEMENT Ort (#PCDATA)>
        <!ELEMENT Telefon (#PCDATA)>
    ]>
```

...

Erste Anwendung in XML (III)

Dann der erste Datensatz:

```
...

<Adresse>
    <Nachname>Kegel</Nachname>
    <Vorname>Thomas</Vorname>
    <Strasse>Eichhornstrasse 51</Strasse>
    <PLZ>78464</PLZ>
    <Ort>Konstanz</Ort>
    <Telefon>07531-958544</Telefon>
</Adresse>
```

Erste Anwendung in XML (IV)

Für weitere Datensätze kommt eine weitere Hierarchieebene hinzu:

```
<?xml version="1.0"?>
        <!DOCTYPE Adresse [
        <!ELEMENT Adresse (Datensatz)+>
                <!ELEMENT Datensatz (Nachname, Vorname, Strasse, PLZ,
Ort, Telefon)>
                <!ELEMENT Nachname (#PCDATA)>
                <!ELEMENT Vorname (#PCDATA)>
                <!ELEMENT Strasse (#PCDATA)>
                <!ELEMENT PLZ (#PCDATA)>
                <!ELEMENT Ort (#PCDATA)>
                <!ELEMENT Telefon (#PCDATA)>
]>

...
```

Erste Anwendung in XML (V)

Nun kommen weitere Datensätze hinzu:

```
...
<Adresse>
        <Datensatz>
                <Nachname>Kegel</Nachname>
                <Vorname>Thomas</Vorname>
                <Strasse>Eichhornstrasse 51</Strasse>
                <PLZ>78464</PLZ>
                <Ort>Konstanz</Ort>
                <Telefon>07531-958544</Telefon>
        </Datensatz>
        <Datensatz>
                <Nachname>Mustermann</Nachname>
                <Vorname>Hans</Vorname>
                <Strasse>Seestrasse 10</Strasse>
                <PLZ>70010</PLZ>
                <Ort>Stuttgart</Ort>
                <Telefon>0711-5623912</Telefon>
        </Datensatz>
</Adresse>
...
```

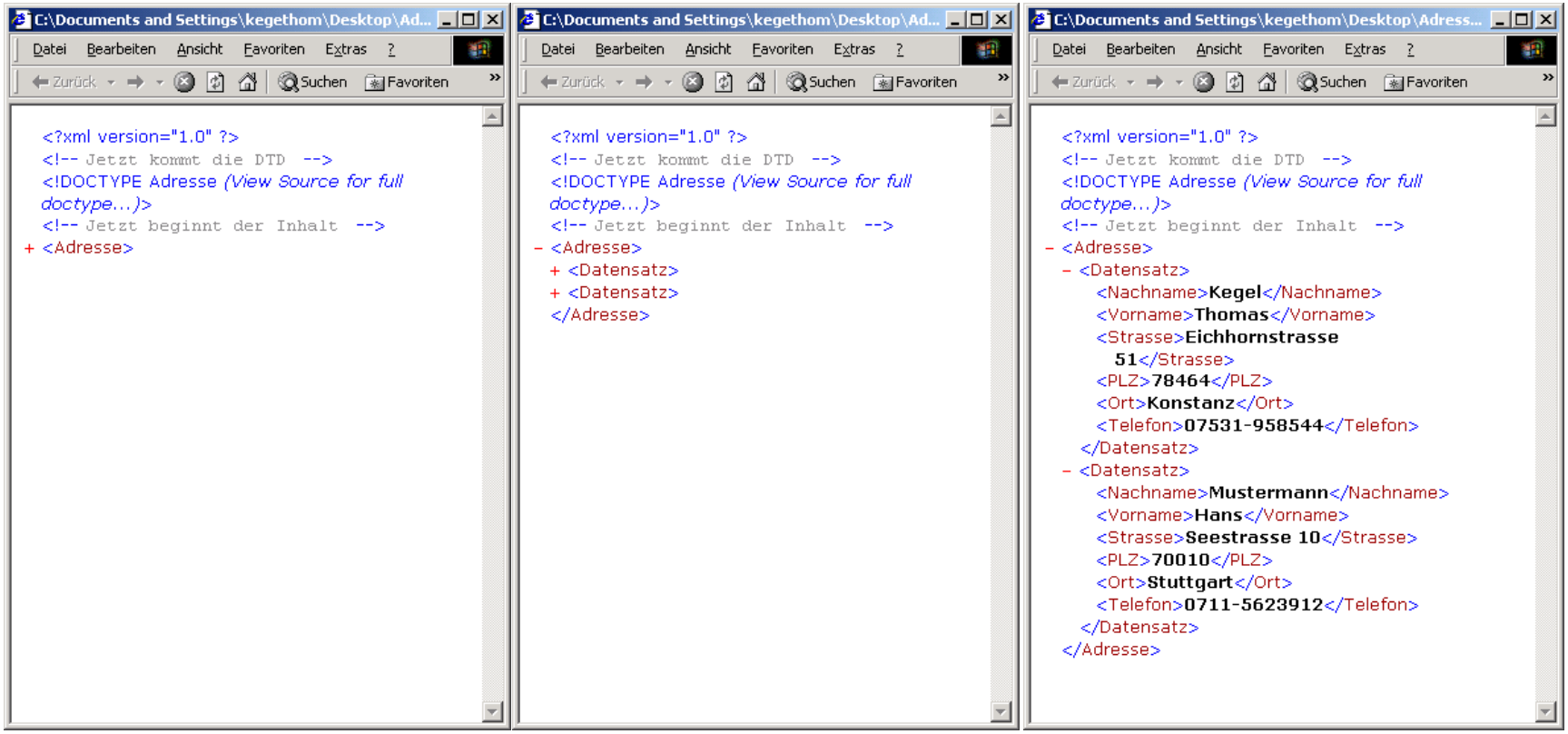
Erste Anwendung in XML (VI)

Und das Ganze mit Kommentaren:

```
<?xml version="1.0"?>
<!--Jetzt kommt die DTD-->
        <!DOCTYPE Adresse [
        <!--Jetzt werden die Tags definiert-->
        <!ELEMENT Adresse (Datensatz)+>
                <!ELEMENT Datensatz (Nachname, Vorname, Strasse, PLZ,
Ort, Telefon)>
                <!ELEMENT Nachname (#PCDATA)>
                <!ELEMENT Vorname (#PCDATA)>
                <!ELEMENT Strasse (#PCDATA)>
                <!ELEMENT PLZ (#PCDATA)>
                <!ELEMENT Ort (#PCDATA)>
                <!ELEMENT Telefon (#PCDATA)>
]>
<!--Jetzt beginnt der Inhalt-->
<Adresse>
<!--Jetzt folgen die Daten-->
<!--Datensatz Nummer 1 -->
<Datensatz>
...
```

Erste Anwendung in XML (VII)

Der Dokumentenbaum



Erste Anwendung in XML (VIII)

Der Zeichensatz in XML

Standard ist UTF-8 (die ersten 128 Zeichen des ASCII-Codes). Er beinhaltet keine Sonderzeichen und Umlaute.

Für Westeuropa und Latainamerika gilt „ISO-8859-1“.

Erste Anwendung in XML (IX)

Der Zeichensatz wird im Prolog definiert:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
        <!DOCTYPE Adresse [
        <!ELEMENT Adresse (Datensatz)+>
                <!ELEMENT Datensatz (Nachname, Vorname, Strasse, PLZ,
Ort, Telefon)>
                <!ELEMENT Nachname (#PCDATA)>
                <!ELEMENT Vorname (#PCDATA)>
                <!ELEMENT Strasse (#PCDATA)>
                <!ELEMENT PLZ (#PCDATA)>
                <!ELEMENT Ort (#PCDATA)>
                <!ELEMENT Telefon (#PCDATA)>
]>

...
```

Aufgabe 2

Finden und korrigieren Sie die fünf Fehler im XML-Code:

```
<?xml version="1.0"?>
            <!DOCTYPE CDSAMMLUNG [
            <!ELEMENT CDSAMMLUNG (Datensatz)+>
            <!ELEMENT Datensatz (Kuenstler, Titel, Jahr)>
            <!ELEMENT Kuenstler (#PCDATA)>
            <!ELEMENT Titel (#PCDATA)>
            <!ELEMENT Jahr (#PCDATA)>
            ]>
<cdsammlung>
            <Datensatz>
                        <Kuenstler>Dido</Kuenstler>
                        <Titel>No Angel</Titel>
                        <Jahr>2000</Jahr>
            </Datensatz>
            <Datensatz>
                        <Kuenstler>Robbie Williams</Kuenstler>
                        <Titel>Swing When You're Winning</Titel>
                        <Jahr>2001<Jahr>
            </Datensatz>
            <Datensatz>
                        <Kuenstler>Herbert Grönemeyer</Künstler>
                        <Titel>Männer</Titel>
                        <Jahr>1990</Jahr>
</CDSAMMLUNG>
```

Externe DTD (I)

Der Verweis:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
        <!DOCTYPE Adresse SYSTEM "daten.dtd">
<Adresse>
        <Datensatz>
                <Nachname>Kegel</Nachname>
                <Vorname>Thomas</Vorname>
                <Strasse>Eichhornstrasse 51</Strasse>
                <PLZ>78464</PLZ>
                <Ort>Konstanz</Ort>
                <Telefon>07531-958544</Telefon>
        </Datensatz>
 </Adresse>
```

Hinweis: Beachten Sie die relativen und absoluten Pfade!

Externe DTD (II)

Die DTD "daten.dtd":

```
<!ELEMENT Adresse (Datensatz)+>
        <!ELEMENT Datensatz (Nachname, Vorname, Strasse, PLZ,
Ort, Telefon)>
                <!ELEMENT Nachname (#PCDATA)>
                <!ELEMENT Vorname (#PCDATA)>
                <!ELEMENT Strasse (#PCDATA)>
                <!ELEMENT PLZ (#PCDATA)>
                <!ELEMENT Ort (#PCDATA)>
                <!ELEMENT Telefon (#PCDATA)>
```

Aufgabe 3

Schreiben Sie eine DTD und zwei XML-Dokumente, welche dieselbe DTD verwenden!

Attribute

Zu jedem Tag können Attribute definiert werden. Dadurch kann der Inhalt des Tags genauer spezifiziert werden.

Bsp.:

In der DTD:

```
<!ELEMENT Strasse (#PCDATA)>
                    <!ATTLIST Strasse art (privat|buero) #REQUIRED>
<!ELEMENT Ort (#PCDATA)>
                    <!ATTLIST Ort art (privat|buero) #IMPLIED>
<!ELEMENT Telefon (#PCDATA)>
                    <!ATTLIST Telefon art (privat|buero) "buero">
```

Im XML-Dokument:

```
<Strasse art="privat">Seestraße 1</Strasse>
<Ort art="buero">Konstanz</Ort>
<Telefon>07531 958544</Telefon>
```

Aufgabe 4

Erweitern Sie die Adress-Datenbank (aus Aufgabe 3) bei den Elementen Ort, PLZ, Strasse und Telefon durch ein Attribut, welches die Geschäfts- bzw. Privatadresse spezifiziert!

Entities

Mit Entities können häufig wiederkehrende Texte abgekürzt werden.

Die Definition wird in der DTD aufgenommen.

Bsp.:

```
<!ENTITY text "Konstanz">
```

Und im XML-Dokument verwendet:

```
<Ort>&text</Ort>
```

Datenausgabe mit XSL (I)

XSL steht für Extensible Stylesheet Language.

XSL ähnelt CSS (Cascading Style Sheets) – ist aber viel flexibler.

Im Prolog des XML-Dokuments erfolgt die Zuweisung auf eine XSL-Datei.

Bsp.:

```
<?xml–stylesheet href="style.xsl" type="text/xsl" ?>
```

Datenausgabe mit XSL (II)

In der XSL-Datei stehen die Formatierungen.

Bsp.:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
        <xsl:template>
                <xsl:for-each select="Ausgabe/Datensatz">
                        <H1>
                                <xsl:value-of select="Vorname"/>
                        </H1>
                </xsl:for-each>
        </xsl:template>
</xsl:stylesheet>
```

Durchsucht das Dokument nach allen Tags auf der Ebene "Datensatz"

Liest und gibt alle Inhalte des Tag "Vorname" aus.

Datenausgabe mit XSL (III)

Bsp.: Datenausgabe in Tabellenform (Teil 1 Stil-Definition und Tabellenkopf)

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
    <xsl:template>
        <STYLE>
            <![CDATA[
                .design
                    {
                        color:blue;
                        font-size:10pt;
                        font-family:Arial;
                    }
            ]]>
        </STYLE>
        <TABLE BORDER="2">
        <TR STYLE="font-weight:bold">
            <TD>Vorname</TD>
            <TD>Nachname</TD>
            <TD>Strasse</TD>
            <TD>PLZ</TD>
            <TD>Ort</TD>
            <TD>Telefon</TD>
        </TR>
```

Definition der globalen Stilklasse ".design"

Tabellenkopf

Datenausgabe mit XSL (IV)

Bsp.: Datenausgabe in Tabellenform (Teil 2 – Die Daten-Zuordnung)

```
    <xsl:for-each select="Adresse/Datensatz">
    <TR>
    <TD>
        <DIV CLASS="design">
            <xsl:value-of select="Vorname"/>
        </DIV>
    </TD>
    <TD>
        <DIV CLASS="design">
            <xsl:value-of select="Nachname"/>
        </DIV>
    </TD>

    ...

  </TR>
  </xsl:for-each>
  </TABLE>
  </xsl:template>
</xsl:stylesheet>
```

Definition der globalen Stilklasse ".design"

Ordnet dem Abschnitt den Style "design" zu.

Liest und gibt alle Inhalte des Tag "Vorname" aus.

Datenausgabe mit XSL (V)

Bsp.: Die Ausgabe automatisch von XSL sortieren lassen

```
        ...

        <xsl:for-each select="Adresse/Datensatz" order-by="+ Nachname">
        <TR>
        <TD>
            <DIV CLASS="design">
                <xsl:value-of select="Vorname"/>
            </DIV>
        </TD>
        <TD>
            <DIV CLASS="design">
                <xsl:value-of select="Nachname"/>
            </DIV>
        </TD>

        ...

    </TR>
    </xsl:for-each>
    </TABLE>
    </xsl:template>
</xsl:stylesheet>
```

Attribut "order-by" mit Wahl auf Sortierung in aufsteigender Richtung (+)

Zusammenfassung

Sie wissen nun,

- Wie die die Grundstruktur eines XML-Dokuments aussieht
- Was ein Prolog ist
- Wie XML-Tags definiert werden
- Was eine DTD ist und wie sie verwendet wird
- Wie Kommentare in den XML-Code eingefügt werden
- Für was Attribute und Entities verwendet werden können
- Wie XML-Daten durch XSL ausgegeben werden.

Ende des Abschnitts