Web Services

XML, WSDL, SOAP und UDDI
Einblicke und Ausblicke

Beschreibung von Informationen - XML

- Inhalt
  - Instanzen und Schema
  - Verarbeitung von XML Dokumenten
  - Namensräume – Namespaces
  - Transformationen
    - XSLT
    - XPath
    - Tools
  - XML und Web Services

Beschreibung von Informationen – XML

- *Web Services basieren auf XML*
  - XML liefert
    - Beschreibung
    - Speicherung
    - Übertragungsformate
  - XML
    - Ist ähnlich wie HTML (Hypertext Markup Language)
    - Besteht aus
      - Elementen
      - Attributen
      - Werten
    - Können, falls sie wohlgeformt sind,
      - Im Browser dargestellt werden (MS IE)

Beschreibung von Informationen – XML

- *XML definiert Datentypen und Datenstrukturen*
  - XML Elemente definieren
    - Datentypen und
    - Strukturinformationen
      für die Daten, welche im XML-Dokument enthalten sind.
  - XML kann Daten und Strukturen modellieren
    - Für spezifische Anwendungsgebiete (Software-Domänen)
  - Programme, welche XML interpretieren können, sind in der Lage, XML-Dokumente auf Domänen-spezifische Formate abzubilden.
    - Ein Web Service macht im wesentlichen das selbe.

Beschreibung von Informationen – XML

- *XML definiert auch Übertragungsformate und Quality of Service*
  - Im Rahmen der Web Services wird auch definiert,
    - Wie und
    - Mit welcher Qualität
      Daten übermittelt werden.

    - Wie Services publiziert und
    - Gefunden werden können.

Beschreibung von Informationen – XML

- *Ausserhalb der Web Services wird XML überwiegend benutzt, um Daten zu formatieren.*
  - Im Rahmen der Web Services wird XML eingesetzt für:
    - Datenspeicherung und Datenformatierung
    - Beschreibung der Software (Schnittstelle), welche die XML Daten zu manipulieren hat.

Beschreibung von Informationen – XML Einführendes Beispiel

- *Web Services Daten können in XML konvertiert werden.*
  - Damit Daten in Web Services benutzt werden können, müssen sie als XML Dokumente dargestellt werden.
    - Eine Applikation kann jederzeit ASCII oder andere Anwendungsdaten in XML überführen
    - Viele Datenbanken erlauben die Ausgabe der Daten in XML
  - Tabellenbeschreibung
    - Kunde

KundenNummer	Integer
KundenName	String
KundenAdresse	String
KundenPLTZ	Integer
KundenOrt	String

Beschreibung von Informationen – XML
Einführendes Beispiel

- *Web Services Daten können in XML konvertiert werden.*
  - XML-Beschreibung
    -
```
<?xml version="1.0"?>
<Kunde>
        <KundenNummer>879012</KundenNummer>
        <KundenName>Kuhn</KundenName>
        <KundenAdresse>Dorfstrasse12</KundenAdresse>
        <KundenPLTZ>1234</KundenPLTZ>
        <KundenOrt>Rikon</KundenOrt>
</Kunde>
```

Beschreibung von Informationen – XML
Einführendes Beispiel

- *Zuerst werden die Daten,*
  Dann das dazugehörige Schema definiert.
  - <xsd:schema
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="Customer" type="CustomerType"/>
    <xsd:complexType name="CustomerType">
    <xsd:sequence>
    <xsd:element name="CustomerNumber,,
    type="xsd:integer"/>
    <xsd:element name="CustomerName,,
    type="xsd:string"/>
    <xsd:element name="CustomerAddress,,
    type="xsd:string"/>
    <xsd:element name="PostalCode,,
    type="PostalCodeType"/>
    <xsd:element name="CustomerCity,,
    type="xsd:string"/>
    </xsd:sequence>
    </xsd:complexType>

Beschreibung von Informationen – XML
Einführendes Beispiel

- *Zuerst werden die Daten,*
  Dann das dazugehörige Schema definiert.(2)
  - ...

```
        <xsd:simpleType name="PostalCodeType">
            <xsd:restriction base="xsd:string">
                <xsd:pattern value="\d{4} | CH-\d{4}"/>
            </xsd:restriction>
        </xsd:simpleType>
    </xsd:schema>
```

Beschreibung von Informationen – XML
Einführendes Beispiel

- *Schema validieren Daten*
  - Das Beispiel zeigt
    - Schema definieren Validation, Datentypen und Dokumentstruktur für unsere Kundendatensätze.
    - Die erste Zeile zeigt, dass unser XML Schema der W3C Vorgabe (gemäss URL) gehorchen muss.
      - Ein verarbeitendes Programm muss dieses Schema kennen oder eine Fehlermeldung generieren.

Beschreibung von Informationen – XML
Einführendes Beispiel

- *Schemas enthalten einfache und komplexe Datentypen.*
  - Das Beispiel zeigt
    - Wir können einfache und (selber definierte) komplexe Datentypen kombinieren.
    - Die angegebene Sequenz (**`sequence`**) muss eingehalten werden.
    - Die Datentypen der einzelnen Datenelemente der Adresse ergeben sich aus dem Schema.
    - Der Aufbau der Postleitzahl muss dem Muster in der **`restriction`** Spezifikation gehorchen (entweder 4 Zahlen oder CH- gefolgt von 4 Zahlen).

Beschreibung von Informationen – XML
Einführendes Beispiel

- *Schemas können unabhängig von den Daten definiert und ausgetauscht werden.*
  - Typische Seitenbeschreibungssprachen wie Runoff, TeX oder LaTeX betten die Formatieranweisungen in den Text ein.
  - Mit XML Schema kann man externe Definitionen festlegen und auf Dokumente anwenden.
    - Globale Änderungen lassen sich dadurch leichter durchführen als im eingebetteten Fall.

Beschreibung von Informationen – XML Instanz und Schema

- *Schemas helfen Daten ins XML Format umzuwandeln bzw. aus XML zu extrahieren.*
  - Die Standardisierung der Datentypen führt zu einer grossen Interoperabilität (HW, SW Unabhängigkeit).
  - XML Tools erlauben die Transformation der Daten und XML Dokumente in unterschiedliche XML Formate.
  - Mit XML ist der Programmierer in der Lage, ein einheitliches Austauschformat einzusetzen:
    - Er muss seine Daten nur ins XML Format bringen bzw. Daten aus dem XML Format in sein eigenes umwandeln können.

Beschreibung von Informationen – XML Instanz und Schema

- *XML-Dokumente sind Daten-unabhängig definierbar*
  - In XML werden primär die Tags definiert.
  - In den Dokument-*Instanzen* werden die Daten gespeichert:
    - **`<KundenNummer>123123</KundenNummer>`**
      - **`<KundenNummer>`** ist der beschreibende Tag
      - **`123123`** ist der Wert (innerhalb des Tags)
  - XML Elemente werden durch <, > begrenzt und haben sowohl Start- als auch End-Tag (</...>).
  - Falls ein Element nur Attribute enthält, kann die Schreibweise vereinfacht werden (<.../>):
    - **`<KundenZahlungsmoral Format=„fünf Zeichen“/>`**

Beschreibung von Informationen – XML Instanz und Schema

- *XML-Elemente können ein oder mehrere Attribute enthalten*
  - Attribute bestehen aus *Namen=Wert* Paaren
    - **<FirmenName Land=„Brazil“ Seit=„1890“>**
      Caffee do Brazil
      </FirmenName>
  - XML Schemas definieren den Datentyp, die Datenstruktur und die Semantik:
    - **<xsd:element name=„KundenNummer“**
      type=„xsd:integer“/>
  - Dabei können Informationen aus unterschiedlichen XML Dokumenten stammen:
    - **xsd** : entspricht dem W3C Schema (XML Dokument)

Beschreibung von Informationen – XML Instanz und Schema

- *XML-Prozessoren verarbeiten Schema und Daten*
  - XML Prozessoren überprüfen, ob die Elementnamen in den XML Instanzen jenen im Schema entsprechen.
    - Oft werden Parser im Fehlerfall einfach weiterfahren, also nicht abbrechen. Dies entspricht dem Verhalten der HTML Parser (unbekannte Tags werden überlesen).
    - Der Grund für dieses Verhalten liegt darin, dass ein Dokument unterschiedliche Schemata einbinden kann, also eventuell in einem späteren Schritt die Validation vervollständigt wird.

Beschreibung von Informationen – XML Instanz und Schema

- *Mit der Zuordnung der Tags zu Definitionen im Schema werden Datentypen und Struktur der Daten festgelegt und überprüfbar.*
  - Beispiel
    - ```
      <xsd:simpleType name="PostalCodeType">
          <xsd:restriction base="xsd:string">
           <xsd:pattern value="\d{4} | CH-\d{4}"/>
          </xsd:restriction>
      </xsd:simpleType>
      ```
    - Der selber definierte **`simpleType`** gestattet eine klare Überprüfung der Postleitzahl:
      - Grundsätzlich handelt es sich um eine Zeichenkette (**`xsd:string`**)
      - Aber zusätzlich wird für diese ein Muster definiert (**`CH-8640`**)
      ```

Beschreibung von Informationen – XML Instanz und Schema

- *Mit Hilfe komplexer Datentypen können Element-Inhalte modelliert werden.*
  - Beispiel
    - ```
      <xsd:complexType name="CustomerType">
            <xsd:sequence>
                  <xsd:element name="CustomerNumber"
                               type="xsd:integer"/>
                  <xsd:element name="CustomerName"
                               type="xsd:string"/>
                  ...
            </xsd:sequence>
      </xsd:complexType>
      ```
    - Mit **`complexType`** wird der Inhalt des XML-Dokuments modelliert.
      ```

Beschreibung von Informationen – XML
Instanz und Schema
Datentypen und Programmiersprachen

- *XML überbrückt die enge Bindung von Datentypen an **eine** Programmiersprache.*
  - Jede Programmiersprache definiert bestimmte Basisdatentypen
  - Darüber hinaus hat der Programmierer die Möglichkeit eigene Datentypen zu definieren, allerdings unterschiedlich in jeder Programmiersprache.
  - In XML werden die Datentypen universell festgelegt:
    - **`encodingStyle=„http://schemas.xmlsoap,org/soap/encoding/“`**
    - Im referenzierten Schema wird festgelegt, wie der SOAP **`encodingStyle`** für die Datenserialisierung eingesetzt werden soll.

Beschreibung von Informationen – XML
Instanz und Schema
Datentypen und Programmiersprachen

- *XML Schema sind vergleichbar mit SQL Schema.*
  - XML Instanzen sind analog zu SQL Datenbankschemas.
  - XML Schema sind universeller als SQL Schemas.
    - Die Syntax von XML ist viel flexibler und daher auch schwieriger konkret einzusetzen.

Beschreibung von Informationen – XML
Instanz und Schema
XML Schema und DTD

- *XML Schema verbessern und ersetzen DTDs.*
  - DTD‘s (Document Type Definition) war ein erster Ansatz die Syntax und Semantik von XML zu beschreiben.
    - DTD‘s sind keine XML Dokumente
    - Die Beschreibung mittels DTD‘s ist zu unflexibel und zu global.
    - DTD‘s spielen in Web Services keine Rolle.

Beschreibung von Informationen – XML
Instanz und Schema
XML Schema und DTD

- *XML Schema und DTD's können für die Validation von XML Dokumenten eingesetzt werden.*
  - Dokument-Instanz (PO_57.xml):
    -
```
<PurchaseOrder OrderDate="2002-09-15">
  <Contact Country="US">
          <Name>Coffee Shop Schipfe</Name>
          <Street>In der Schipfe 2</Street>
          <City>Zuerich</City>
          <State>Zuerich</State>
          <Zip>8022</Zip>
  </Contact>
  <LineItem>
          <SupplierName>Caffee do Brazil</SupplierName>
          <ProductName>Brazilo</ProductName>
          <Quantity>100</Quantity>
          <UnitPrice>150.00</UnitPrice>
  </LineItem>
  <LineItem>..</LineItem>
</PurchaseOrder>
```

Beschreibung von Informationen – XML
Instanz und Schema
XML Schema und DTD

- *XML Schema und DTD's können für die Validation von XML Dokumenten eingesetzt werden.*
  - Schema Definition (PO_58.xsd): 1
    - »
```
<xsd:schema
xmlns:xsd="http://wwww3.org/2001/XMLSchema"
xmlns:po="SkateboardGarage.com/order"
elementFormDefault="qualified"
targetNamespace="SkateboardGarage.com/order">
      <xsd:element name="PurchaseOrder">
            <xsd:complexType>
            <xsd:sequence>
                  <xsd:element name="Contact"
                        type="po:ContactType"/>
                  <xsd:element ref="po:LineItem"
                        maxOccurs="unbound"/>
            </xsd:sequence>
            <xsd:attribute name="OrderDate"
                        type="xsd:date"/>
            </xsd:complexType>
      </xsd:element>
```

Beschreibung von Informationen – XML
Instanz und Schema
XML Schema und DTD

- *XML Schema und DTD's können für die Validation von XML Dokumenten eingesetzt werden.*
  - Schema Definition (PO_58.xsd): 2
    -
```
<xsd:complexType name="ContactType">
        <xsd:sequence>
                <xsd:element name="Name"
                        type="xsd:string"/>
                <xsd:element name="Street"
                        type="xsd:string"/>
                <xsd:element name="City"
                        type="xsd:string"/>
                <xsd:element name="State"
                        type="xsd:string"/>
                <xsd:element name="Zip"
                        type="xsd:positiveInteger"/>
        </xsd:sequence>
        <xsd:attribute fixed="CH" name="Country„
                type="xsd:NMTOKEN"/>
</xsd:complexType>
```

Beschreibung von Informationen – XML Instanz und Schema
XML Schema und DTD

- *XML Schema und DTD's können für die Validation von XML Dokumenten eingesetzt werden.*
  - Schema Definition (PO_58.xsd): 3
    -

```
<xsd:element name="LineItem">
<xsd:complexType>
        <xsd:sequence>
                <xsd:element name="SupplierName"
                        type="xsd:string"/>
                <xsd:element name="ProductName"
                        type="xsd:string"/>
                <xsd:element name="Quantity„
                        type="xsd:positiveInteger"/>
                <xsd:element name="UnitPrice"
                        type="xsd:decimal"/>
        </xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Beschreibung von Informationen – XML
Instanz und Schema
XML Schema und DTD

- *Qualifizierte Elementnamen bezeichnet man als QNames.*
  - `<xsd:schema xmlns:xsd="http://wwww3.org/2001/XMLSchema" xmlns:po="SkateboardGarage.com/order" elementFormDefault="qualified" targetNamespace="SkateboardGarage.com/order">`
  - Dieses Wurzelelement enthält ein Attribu, welches den Namensraum für das Einkaufsformular (PO:Purchase Order) beschreibt
    - `xmlns:po="SkateboardGarage.com/order„`
  - Das Formular beschreibt eine Struktur (**`complexType`**), welche eine Sequenz von **`ContactType`** und **`LineItem`** enthält (Kopf und Rumpf des Dokuments)
  - Die Elementnamen werden mit dem Präfix **`po`**: qualifiziert.

Beschreibung von Informationen – XML
Instanz und Schema
XML Schema und DTD

- *DTD's sind **einfacher für Validation**, schlechter für die Beschreibung der Semantik.*
  -
```
<!ELEMENT PurchaseOrder (Contact, LineItem+)>
<!ATTLIST PurchaseOrder
  OrderDate CDATA #IMPLIED>
<!ELEMENT Contact (Name, Street, City, State, Zip)>
<!ATTLIST Contact Country CDATA #FIXED "CH">
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Street (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Zip (#PCDATA)>
<!ELEMENT LineItem
  (SupplierName, ProductName,Quantity, UnitPrice)>
<!ELEMENT SupplierName (#PCDATA)>
<!ELEMENT ProductName (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT UnitPrice (#PCDATA)>
```

Beschreibung von Informationen – XML
Verarbeitung von XML Dokumenten

- *DOM und SAX API's ermöglichen das Parsen von XML Dokumenten.*
  - DOM (Document Object Model)
    - W3C
      - http://www.w3.org/TR/DOM-Level-2-Core/
  - SAX (Simple API for XML)
    - XML-DEV
      - http://www.saxproject.org

Beschreibung von Informationen – XML
Verarbeitung von XML Dokumenten

- *SAX list das XML Dokument einmal;*
  DOM kann das Dokument mehrfach lesen.
  - Hauptunterschied zwischen SAX und DOM
    - DOM enthält ein Objektmodell (Struktur des Dokuments), das API enthält Methoden zum Lesen und bearbeiten des Dokuments.
    - DOM liest das Dokument in den Hauptspeicher

    - SAX benötigt weniger Speicher
    - SAX generiert Callbacks (beim Einlesen der Tags)

    - SAX ist sinnvoll, wenn nur einmal gelesen und analysiert wird
    - DOM ist sinnvoll, wenn mehrfach auf Dokumentteile zugegriffen wird.

Beschreibung von Informationen – XML
Verarbeitung von XML Dokumenten

- *XML Parsers sind Commodities*.
  - Für MS und in Java existieren leistungsfähige Parser erhältlich.
    - Apache
      - http://www.apache.org
      - Xerxes
    - MS
    - IBM
    - Sun
    - ...

Beschreibung von Informationen – XML
Verarbeitung von XML Dokumenten

- *DOM Parser setzen hierarchische Strukturen voraus*.
  - DOM kennt verschiedene Interfaces
    - **`Node, Element, Document, ...`**
      - Einige Nodes sind auch Elements
      - Das Root-Element ist ein Node und ein Document Element
    - Beispiel-Methoden
      - **`Node`** Interface
        - » **`getParentNode()`**
        - » **`getChildNodes()`**
      - **`Element`** Interface
        - » **`getAttributeNS()`**
        - » **`setAttributeNS()`**
      - **`Document`** Interface
        - » **`createElement()`**
        - » **`createAttribute()`**

Beschreibung von Informationen – XML
Verarbeitung von XML Dokumenten

- *SAX Parser lesen sequentiell das Dokument.*
  - SAX kennt verschiedene Interfaces
    - **`ContentHandler, XMLReader, Attributes, ...`**
    - Beispiel-Methoden
      - **`ContentHandler`** Interface
        - » **`startDocument()`**
        - » **`startElement()`**
      - **`XMLReader`** Interface
        - » **`getProperty()`**
        - » **`setProperty()`**
      - **`Attributes`** Interface
        - » **`getValue()`**
        - » **`getType()`**
    - **DOM und SAX können auch gemeinsam genutzt werden**

Beschreibung von Informationen – XML Namespaces

- *Namespaces definieren den Gültigkeitsbereich oder qualifizieren Elementnamen.*
  - Namespaces helfen Namenskonflikte zu vermeiden
    - Beispiel
      - Ein Web Service verwendet unterschiedliche XML Dokumente
        - » Die Instanz mit den Daten
        - » Das SOAP Envelope Schema (Message Formatbeschreibung)
        - » WSDL Instanz-Dokument (Interfacebeschreibung)
        - » WSDL Schema (Validation der Interfacebeschreibung)
  - Web Service Technologien setzen typischerweise Namenräume ein
    - **`xmlns:meinns=„http://www.meinweb.com/namespaces/WSDL“`**
  - An Stelle von URL‘s sind **URI**‘s für die Namensraum-Lokalisierung empfohlen.

Beschreibung von Informationen – XML Namespaces

- *Namespaces können auch Semantik beschreiben.*
  - Beispiel
    - In SOAP Messages kann der Codiermechanismus für die Serialisierung und Deserialisierung der Message angegeben werden.
  - URI‘s der Namespaces können auch auf Schemas verweisen
  - Namespaces können in beliebigen Elementpfaden eingesetzt werden
    - **<stkb:boot**
      xmlns:stkb=„http://www.test.com/skateboards“/>
  - Namespaces können auch auf ausführbare Programme verweisen
    - **<p:GetOrderStatus**
      xmlns:p=„http://www.test.com/Skates/Entry“>
      <OrderID>1234321</OrderID>
      </p:GetOrderStatus>

Beschreibung von Informationen – XML Transformation

- *XSLT transformiert XML Formate in andere XML Formate.*
  - Oft muss zwischen unterschiedlichen Formaten umgewandelt werden.
    - Extensible Stylesheet Language : Transformations (XSLT)
  - Verschiedene Tools gestatten die Transformation von einem Format in andere XML- oder Text-Formate.

Beschreibung von Informationen – XML Transformation - XSLT

- *XSLT ist eine Transformations-Technologie.*
  - XSLT ist Teil der XSL Spezifikation von W3C.
  - Ziel ist es, den Inhalt von der Darstellung zu trennen.
  - XSLT arbeitet mit SAX und DOM zusammen
  - XSLT Prozessoren sind frei erhältlich
    - Apache Xalan
    - Saxon von Michael Kay (Software AG)
    - MS, ...
  - Trabsformationen müssen Schemas / DTD‘s aufeinander abbilden.

Beschreibung von Informationen – XML Transformation - XPath

- *XPath stellt Suchausdrücke zum Selektieren von Dokument-Teilen zur Verfügung.*
  - XPath stellt eine Sprache zur Verfügung, um Teile eines Dokuments zu selektieren.
  - XPointer verbindet mehrere XML Dokumente und verwendet die selbe Selektionssprache.
    Identifiziert Dokumentfragmente, lokalisiert interne Strukturen.
  - XPath stellt grundlegende Suchausdrucksmöglichkeiten zur Verfügung:
    - Pattern-Matching
    - Einfache Formeln
    - Zeichenketten-Manipulation

Beschreibung von Informationen – XML
Transformation - Dokumentstruktur

- *XML Dokumente sind hierarchisch strukturiert.*
  - Parser lesen die Dokumente und kreieren neue Baumstrukturen.
  - Target-Dokumente können völlig anders strukturiert sein.
  - Mehrere Dokumente verwenden Namensräume, um Namenskonflikte zu vermeiden.
  - Mit neuen Dokumentformaten steigt der Bedarf an Transformationen.

Beschreibung von Informationen – XML Transformation – Mapping Tools

- *Verschiedene Anbieter haben Transformations-Tools entwickelt.*
  - Beispiele
    - Grafischer Editor von IONA (XML Mapping Editor)
    - Commerce One
    - ...
    - .

Beschreibung von Informationen – XML Transformation – Mapping Tools

- *XML in ASCII umwandeln.*
  - Beispiel (XML2ASCII_75.xsl)
    - <?xml version="1.0" ?>

```
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform
        version="1.0">
        <xsl:output method="text" indent="yes"/>
        <xsl:template match="*">
                <xsl:apply-templates />
        </xsl:template>
        <xsl:template match="CustomerNumber">
                CustomerNumber:
                <xsl:value-of select="." />
        </xsl:template>
        <xsl:template match="CustomerName">
                CustomerName:
                <xsl:value-of select="." />
        </xsl:template>
        <xsl:template match="CustomerAddress">...
</xsl:stylesheet>
```

Beschreibung von Informationen – XML Transformation – Mapping Tools

- *XSLT*
  - Stylesheets werden für die Transformation von XML Dokumenten eingesetzt.
  - Das Beispiel kann zusammen mit einem XML/XSL Prozessor eingesetzt werden, um die Daten aus einem XML Dokument (mit entsprechender Struktur) zu extrahieren.

Beschreibung von Informationen – XML
Referenzen - Links

- *Bücher*
  - XML
    - *XML for the World Wide Web*
      2000 Elizabeth Castro, Peachpit Press
      ISBN 0-201-71098-6
  - XSL
    - *XSLT Programmer's Reference*
      2001 Michael Kay, Wrox Press
      ISBN 1861005067

Beschreibung von Informationen – XML
Referenzen - Links

- *W3C*
  - XML
    - http://www.w3.org/TR/REC-xml
    - XML Base (Anhang, beschreibt u.a. URI Einsatz)
      - http://www.w3.org/TR/xmlbase
    - XML Names
      - http://www.w3.org/TR/REC-xml-names/
  - DOM
    - http://www.w3.org/TR/DOM-Level-2-Core
  - XML Schema
    - http://www.w3.org/TR/xmlschema-1
    - http://www.w3.org/TR/xmlschema-2
  - XML Path Language
    - http://www.w3.org/TR/xpath
  - XML Pointer Language
    - http://www.w3.org/TR/WD-xptr

Beschreibung von Informationen – XML
Referenzen - Links

- *W3C*
  - XML Linking Language
    - http://www.w3.org/TR/xlink
    - Verbindet verschiedene XML Dokumente, auch Fragmente
  - XSL Transformation
    - http://www.w3.org/TR/xslt
  - Canonical XML
    - http://www.w3.org/TR/xml-c14n
    - Generieren einer kanonischen Darstellung des Dokuments, damit es mit anderen Dokumenten verglichen werden kann.
  - XML Information Set
    - http://www.w3.org/TR/xml-infoset
    - Informations-Dokument mit Links auf relevante Spezifikationen
  - XML Inclusions
    - http://www.w3.org/TR/xinclude
  - **XML in 10 Points**
    - http://www.w3.org/XML/1999/XML-IN-10-POINTS