Seminar: Datenbanken - Anwendungsentwicklung

XML

Techniken und Einsatzbeispiele

17.02.2000

UNIVERSITÄT
KAISERSLAUTERN

Fachbereich Informatik
AG Datenbanken & Informationssysteme
http://www.uni-kl.de/AG-Haerder

DBIS
Datenbanken und Informationssysteme

Gliederung

1. Einleitung

2. Techniken

- XML
- XLink
- XSL
- XPath
- XPointer
- Metadaten und RDF

3. Einsatzbeispiele

- E-Commerce
- Informationssuche im Web
- Dokument- und Content-Management
- Web-Automation

4. Zusammenfassung

Einleitung

HTML (Hypertext Markup Language) basiert auf der Metasprache *SGML (Standard Generalized Markup Language).*

Das World Wide Web Consortium (W3C*)* beschloß eine Sprache zu entwickeln, die einfacher zu lernen und anzuwenden ist aber auf den gleichen Grundgedanken basiert wie SGML:

- **Trennung der Daten von ihrer Präsentation**
- **Möglichkeit zur strukturellen Validierung**
- **Erweiterbarkeit**

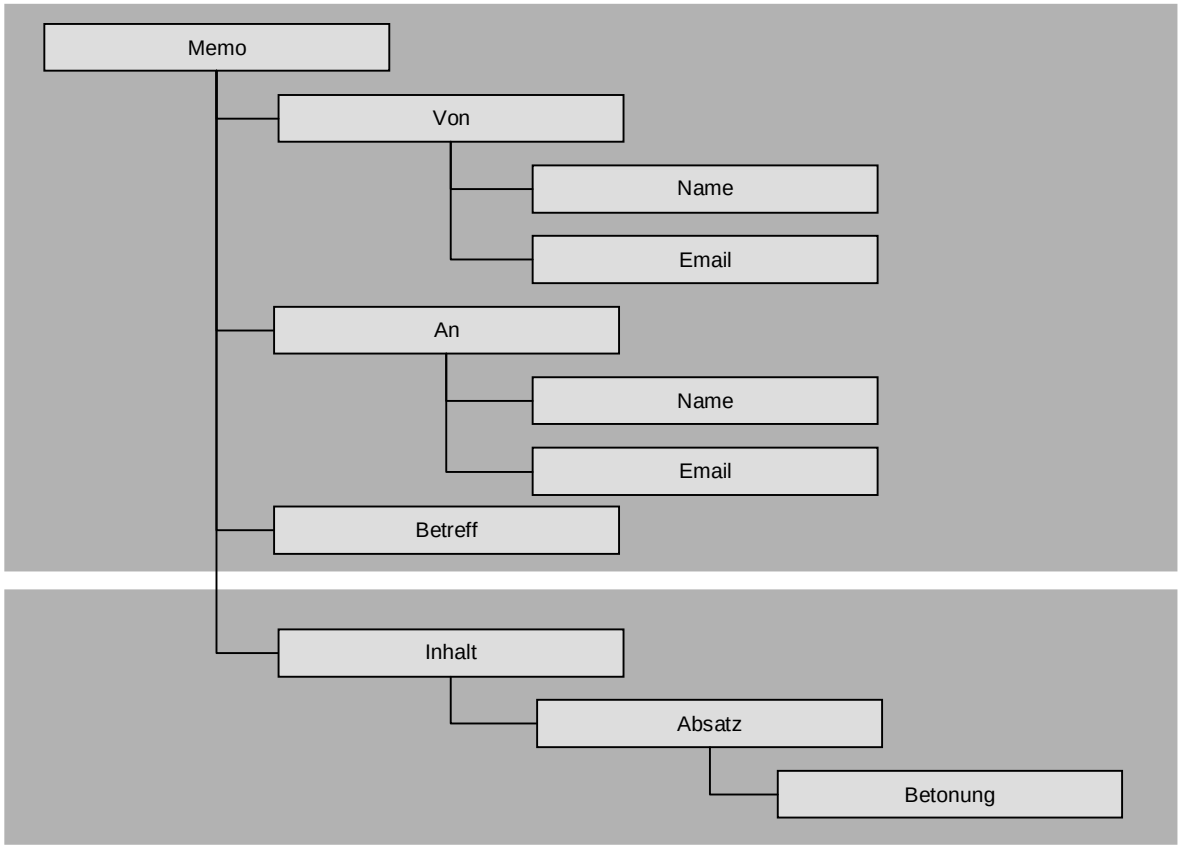
Die Extensible Markup Language (XML) ist eine Teilmenge von SGML.

Ein XML-Parser sollte so klein sein, daß er ohne Probleme in eine Anwendung, z.B. einen Web-Browser, integriert werden kann.

Logische Dokumentstruktur

Aufteilung des Dokumentinhalts in Komponenten

Zweiteilung von Dokumenten



(Baumstruktur eines Dokuments)

Prolog

Der Prolog eines XML Dokuments besteht aus zwei optionalen Teilen:

- **XML-Deklaration**
  - **Versionsangabe**
  - **Kodierungsdeklaration (optional)**
  - ***standalone*-Angabe (optional)**
- **Dokumenttypdeklaration**
  - **Angabe der verwendeten DTD**

Beispiel

PROLOG

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<!DOCTYPE Memo SYSTEM "http://www.fck.de/DTDs/memo.dtd">
```

DOKUMENTINSTANZ

```
<Memo>
     <Von>
         <Name>Otto Rehagel</Name>
         <Email>Otto_Rehagel@fck.de</Email>
     </Von>
     <An>
        <Name>Vorstand</Name>
        <Email>Vorstand@fck.de</Email>
     </An>
     <Betreff Dringlichkeit="hoch">Spielereinkauf</Betreff>
     <Inhalt>
           <Absatz>Vorstand, ich glaube wir <Betonung>muessen
                  </Betonung> unbedingt noch einige neue Spieler
                  kaufen, sonst wird das wohl nichts mit der
                  Meisterschaft.
           </Absatz>
     </Inhalt>
</Memo>
```

Elemente

Ein XML-Dokument besteht aus *Markup-Elementen*.

Ein Markup-Element kann

- **leer sein (Leeres Element).**
- **selbst wieder Elemente (Unterelemente) enthalten.**
- ***Zeichendaten* enthalten.**

Ein Markup Element besteht aus einem Tag-Paar:

- **Beginn-Tag**
- **Ende Tag**

```
<Name>Harry Koch</Name>

<LeeresElement/>
```

Die Namen von Markup-Elementen werden bzgl. Groß- und Kleinschreibung unterschieden.

Attribute (1)

Attribute beschreiben ein Element näher.

Attribute haben Namen und Werte.

Man unterscheidet drei Attributwertetypen:

- **Zeichendaten**
- **Aufzählungen**
- **Schlüsselwörter**

Innerhalb eines Elements darf der Name eines Attributs nur einmal verwendet werden.

Attribute (2)

Durch Attribute kann man ein und dasselbe auf verschiedene Art und Weise beschreiben:

(i) Version mit Unterelementen

```
<Spieler>
        <Nummer>24</Nummer>
        <Tore>5</Tore>
        <Name>Harry Koch</Name>
</Spieler>
```

(ii) Version nur mit Attributen

```
<Spieler Nummer="24" Tore="5" Name="Harry Koch"/>
```

(iii) Gemischte Version

```
<Spieler Nummer="24" Tore="5">Harry Koch</Spieler>
```

Physikalische Dokumentstruktur

Entitäten *(entities)* bilden die physikalische Dokumentstruktur.

Entitäten haben einen Namen und Inhalt.

Auf den Entitäteninhalt bezogen, unterscheidet man zwischen:

- **geparsten Entitäten**
- **ungeparsten Entitäten**

Des weiteren unterscheidet man zwischen:

- **externen Entitäten**
- **internen Entitäten**
- **allgemeinen Entitäten**
- **Parameterentitäten**
- **Dokumententität**

Über eine Entitätenreferenz verweist man auf eine Entität.

```
<!ENTITY FckLogo SYSTEM "http://www.fck.de/wapppen.gif" NDATA GIF>

<!ENTITY FCK "1. FC Kaiserslautern">
<Text>Otto Rehagel ist Trainer des &FCK;.</Text>
```

DTD (Dokumenttypdefiniton)

Die DTD eines XML-Dokuments dient zu seiner strukturellen Validierung.

Ein XML-Dokument, das mit dem XML-Standard konform ist nennt man wohlgeformt.

Ein XML-Dokument, das mit seiner DTD konform ist nennt man gültig.

Man kann eine DTD extern deklarieren

```
<!DOCTYPE Memo SYSTEM "http://www.fck.de/DTDs/memo.dtd">
```

oder intern.

DTD Syntax (Elemente)

Es gibt vier Arten von Inhaltsspezifikationen für Elemente.

1. Leerer Inhalt

```
<!ELEMENT Email EMPTY>
```

2. Beliebiger Inhalt

```
<!ELEMENT Email ANY>
```

3. Kombinierter Inhalt

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

<!ELEMENT Email (#PCDATA|Element1|Element2)*>
```

4. Elementeninhalt

```
<!ELEMENT Email (Element1,
                 (Element2, Element3)+,
                 Element4?)>
```

Indikatoren für das Auftreten: ?, *, +

DTD Syntax (Attribute)

Die Attribute der Elemente werden in einer Attributlistendeklaration beschrieben.

In der Attributlistendeklaration werden der Attributtyp, der Status (optional) und der Defaultwert (optional) eines Attributs abgegeben.

```
<Spieler Nummer="24" Tore="5">Harry Koch</Spieler>

<!ATTLIST Spieler Nummer CDATA #REQUIRED 99
                  Tore   CDATA #REQUIRED 0>
```

Der Status eines Attributs kann folgende Werte annehmen: `#REQUIRED`, `#IMPLIED`, `#FIXED`

Einige Typen, die ein Attribut annehmen kann: `CDATA`, `NOTATION`, `ID`, `ENTITY` **oder eine Aufzählung**

Beispiel

```
<!DOCTYPE Memo[
    <!ELEMENT Memo (Von, An, Betreff, Inhalt)>
    <!ELEMENT Von (Name, Email)>
    <!ELEMENT An (Name, Email)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT Email (#PCDATA)>
    <!ELEMENT Betreff (#PCDATA)>
    <!ATTLIST Spieler Dringlichkeit (niedrig|
                                     mittel|
                                     hoch)      #REQUIRED "niedrig"
    <!ELEMENT Inhalt (Absatz+)>
    <!ELEMENT Absatz (#PCDATA|Betonung)*>
    <!ELEMENT Betonung (#PCDATA)>
]>
```

XLink (XML Linking Language) – Elemente

Verweise in XML-Dokumenten werden mit XLink beschrieben.

XLink stellt insgesamt sechs verschiedenen Elementtypen zur Verfügung:

- **`simple`**
- **`extended`**
- **`locator`**
- **`arc`**
- **`title`**
- **`resource`**

XLink - Attribute

XLink Elemente werden durch die folgenden acht Attribute näher beschrieben:

- **`type`**
  Mögliche Werte: `simple`, `extended`, `locator`, `arc`, `resource` oder `title`.
- **`href`**
  Mögliche Werte: URI-Referenz oder ein XPointer.

Die semantischen Attribute:

- **`role`**
- **`title`**

Attribute, die das Verhalten eines Verweises beschreiben:

- **show-Attribut**
  Mögliche Werte: `new`, `replace`, `embed`, oder `undefined`.
- **`actuate`-Attribut**
  Mögliche Werte: `onLoad`, `onRequest` oder `undefined`

Attribute, die Richtung der Traversierung bestimmen:

- **`from`**
- **`to`**

XLink – Einfache Verweise

Ein einfacher Verweis ist mit dem HTML-Element A vergleichbar.

In der DTD:

```
<!ELEMENT Spieler ANY>
<!ATTLIST Spieler
    xlink:type    (simple)              #FIXED     "simple"
    xlink:href    CDATA                 #REQUIRED
    xlink:role    CDATA                 #IMPLIED
    xlink:title   CDATA                 #IMPLIED
    xlink:show    (new|embed|replace)             "replace"
    xlink:actuate (onLoad|onRequest)              "onRequest"
>
```

In der Dokumentinstanz:

```
<Spieler xlink:href="http://www.fck.de/Spielerliste.xml"
         xlink:role="playerlist"
         xlink:title="Liste aller FCK-Spieler"
         xlink:show="new"
         xlink:actuate="onRequest">
         Hier geht es zu einer Liste aller FCK-Spieler.
</Spieler>
```

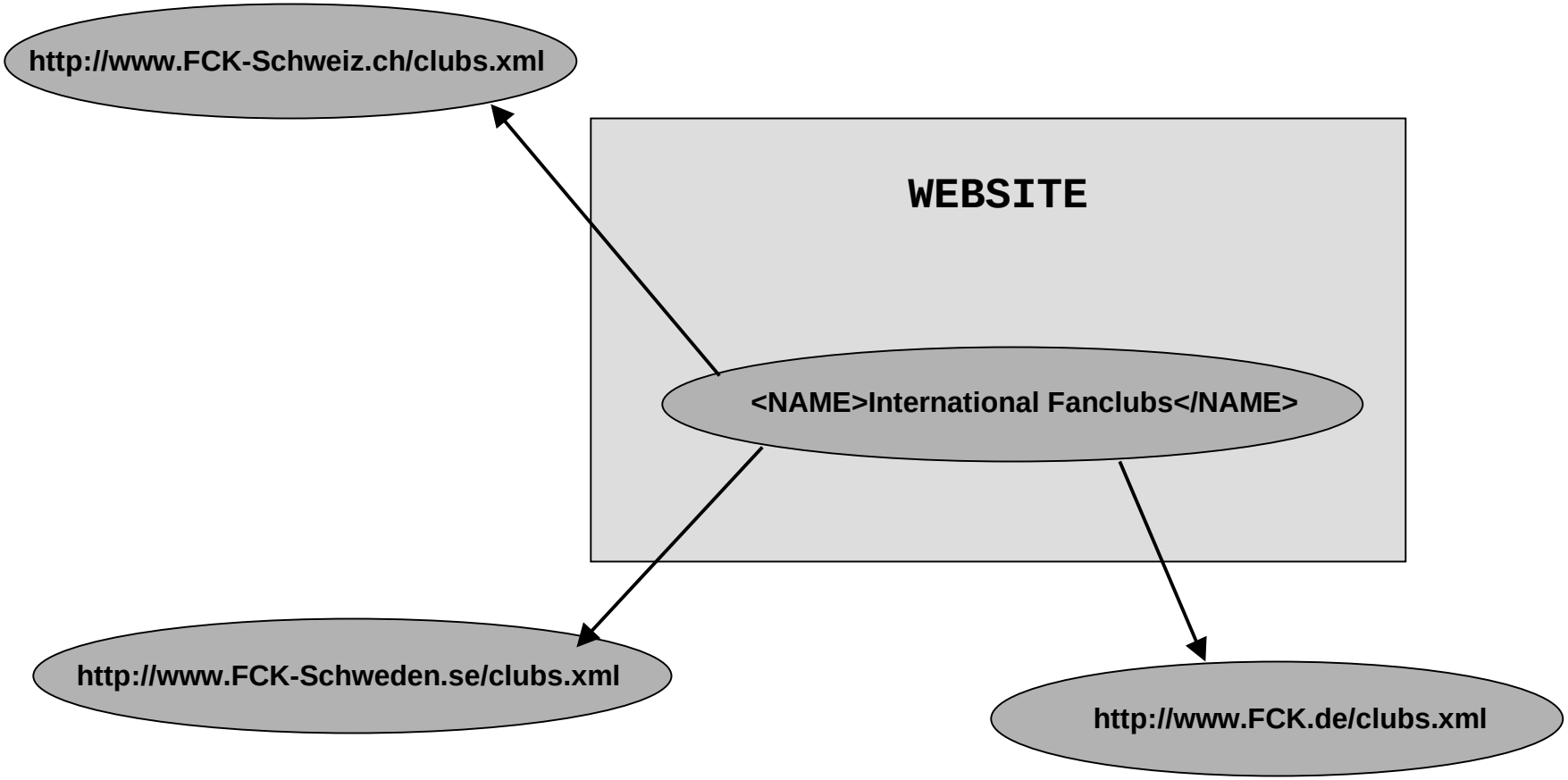
XLink – Erweiterte Verweise

Erweiterte Verweise bieten die volle Funktionalität von XLink.

- ES KÖNNEN MEHR ALS ZWEI RESSOURCEN BETEILIGT SEIN.
- **Out-of-Line-Verweise**
- **Multi-direktionale Traversierung**
- **Traversierung von einer Ressource, auf die man nur Leserecht hat.**

Ein erweiterter Verweis beinhaltet ein beliebige Mischung aus `locator`-, `arc`-, `title`- und `resource`-Elementen.

XLink – Beispiel zu erweiterten Verweisen (1)



XSL (Extensible Stylesheet Language)

Durch das darstellungsunabhängige Markup von XML wird eine medienunabhängige Präsentation ermöglicht.

XSLT (XSL Transformation Language)

- **Formatvorlage *(stylesheet)***
- **Transformationsregeln**
- **Ursprungsbaum**
- **Ergebnisbaum**

Eine Regel besteht aus einem Muster und einer Schablone.

XPath (XML Path Language)

XPath ist die Sprache zum Beschreiben von Mustern in XSLT.

Mit XPath kann man bestimmte Knoten absolut oder relativ bezüglich eines Kontextknotens angeben.

- **Pfadausdruck**
  - **Anweisungen**
    - **Basis, optionale Liste von Prädikaten**
      - **Achse, Knotentest.**

```
id("Mannschaft")/descendant::Spieler/child::Name[position()=last()]
```

Die Achsen von XPath: `descendant`, `child`, `following`, `following-sibling`, `ancestor`, `parent`, `preciding`, `preciding-sibling`, `self`, `descendant-or-self`, `ancestor-or-self`

XPointer (XML Pointer Language)

XPointer erweitert die Funktionalität von XPath in zweierlei Beziehung:

1. `range`-Achse

```
id("Mannschaft")/range::Spieler[1],following-sibling::Spieler[24]
```

2. `string`-Achse

```
string::3,"Spieler",17,27
```

**Mit einem XPointer ist es somit auch möglich,
auf Regionen eines XML-Dokuments zu verweisen, die nicht wohlgeformt sind.**

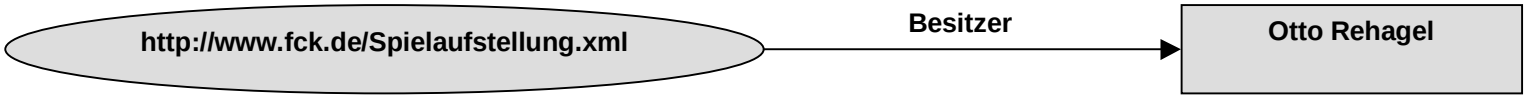
Metadaten und RDF (Resource Description Framework) (1)

RDF ermöglicht es Metadaten von Ressourcen zu beschreiben.

Metadaten sind "Daten über Daten".

Eine RDF-Anweisung ist ein Trippel aus einem Subjekt, einem Prädikat und einem Objekt.

```
(Subjekt:http://www.fck.de/Spielaufstellung.xml;
 Prädikat:Besitzer; Objekt:"Otto Rehagel")
```



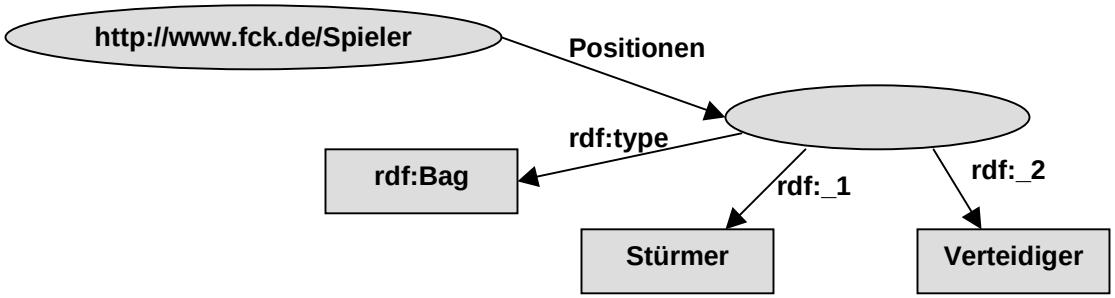

```
<rdf:Description about="http://www.fck.de/Spielaufstellung.xml">
   <bs:Besitzer>Otto Rehagel</bs:Besitzer>
</rdf:Description>
```

Metadaten und RDF (Resource Description Framework) (2)

Ein Objekt kann auch aus einer Liste von Ressourcen bestehen.

Eine solche Liste wird in RDF durch einen Container dargestellt.

Es gibt drei Typen von Containern: Tasche, Sequenz und Alternative.




```
<rdf:Description about="http://www.fck.de/Spieler">
  <bs:Positionen>
      <rdf:Bag>
          <rdf:li>Stürmer</rdf:li>
          <rdf:li>Verteidiger</rdf:li>
      </rdf:Bag>
  </bs:Positionen>
</rdf:Description>
```

Einsatzbeispiel: E-Commerce

Durch enorm hohe Einführungskosten und das Fehlen von Standards brachte EDI (Electronic Data Interchange) nicht den erhofften Erfolg.

Die Vorteile von XML:

- **XML ist plattform-unabhängig.**
- **Die Syntax von XML kann sowohl von Computern als auch von Menschen verstanden werden.**
- **Selbst-beschreibenden Daten.**
- **Möglichkeit zur strukturellen Validierung.**

XML ist aber nicht alleine die Lösung. Es müssen noch Standards gefunden werden.

VEO Systems - CBL *(Common Business Library)*

Informationssuche im Web (1)

Wertsteigerung durch "zusammengesetzte Informationsdienste" oder Mehrwertdienste.

Frage:

Gibt es einen Golfplatz, auf dem heute kein Regen zu erwarten ist und den man von Darmstadt aus in weniger als zwei Stunden erreichen kann?

Voraussetzung: Die Rohdaten dürfen nicht weiter in HTML "versteckt" bleiben.

Darstellung in HTML:

```
<TITLE>Laptop Computer</TITLE>
<BODY>
    <UL>
        <LI>IBM Thinkpad 560X
        <LI>233 MHz
        <LI>32 MB
        <LI>4 GB
        <LI>4.1 Pfund
        <LI>3200 $
    </UL>
</BODY>
```

Darstellung in XML:

```
<Computer Klasse="Portabel">
    <Hersteller>IBM</Hersteller>
    <Familie>Laptop</Familie>
    <Linie>Thinkpad</Linie>
    <Modell>560X</Modell>
    <Geschwindigkeit Einheit="MHz">
        233</Geschwindigkeit>
    <Festplatte Einheit="GB">
        4</Festplatte>
    <Gewicht Einheit="Pfund">
        4.1</Gewicht>
    <Preis Waehrung="US$">
        3200</Preis>
</Computer>
```

Informationssuche im Web (2)

XML-Broker

JEDI (Java based Extraction and Dissemination of Information)

- **Parser**
- **erweiterbares Objektmodell**
- **XML**

Kombination der Informationen durch eine Anfragesprache (XQL, XML-QL).

Dokument- und Content-Management

Dokument-Management

Knowledge-Management

Content-Management

Nicht nur ganze Dokumente verwalten, sondern jede einzelne Komponente.

Optimales Werkzeug zur Erstellung von Komponenten.

Vorteile von XML:

- **Trennung der physikalischen und der logischen Struktur**
- **Selbsterklärendes Markup**
- **Erleichterung der Wartung durch externe Entitäten**

Web-Automation

Was bedeutet Web-Automation?

Momentan ist es außer Web-Browsern keiner anderen Geschäftsanwendung möglich, direkt auf Webdaten zuzugreifen.

Momentane Vorgehensweise, um Web-Daten in das firmeneigene Programm zu bekommen:

1. **Browser aufrufen**
2. **Gewünschte Web-Seite aufrufen.**
3. **Eventuell einige Formulare ausfüllen.**
4. **Copy-and-Paste**

Die momentane Web-Technologie ist reich an Interaktion aber sehr arm an Automation.

WIDL (Web Interface Definition Language) **erlaubt Web-Ressourcen als funktionale Schnittstellen zu beschreiben, auf die über Standard-Web-Protokolle zugegriffen werden kann.**

Zusammenfassung

Techniken

- **XML (Elemente, Attribute, Entitäten, DTD)**
- **XLink**
- **XSL, XSLT**
- **XPath**
- **XPointer**
- **Metadaten und RDF**

Einsatzbeispiele

- **E-Commerce (CBL)**
- **Informationssuche im Web (JEDI)**
- **Dokument- und Content-Management**
- **Web-Automation (WIDL**