Ausblick (AUS)

Lernziele
• Sie kennen die Konzepte und Entwicklungsbestrebungen der wichtigsten in Zukunft relevanten XML-Standards. • Sie wissen, welche Grundgedanken hinter XQuery stehen. • Sie erhalten einen Ausblick auf XML als Metadaten-Format (RDF) • Sie erfahren, warum Java und XML zusammenpassen. • Sie wissen, was sich hinter dem Schlagwort Enterprise Application Integration verbirgt. • Sie erhalten einen kurzen Ausblick, auf XML-Standards, die in diesem Seminar nicht behandelt wurden.

Suche in XML-Dokumenten: XQuery

Was ist XQuery?

XQuery ist eine vom W3C standardisierte Anfragesprache für XML-Dokumente.

Mittels einer (formalen) Anfragesprache stellt man Anfragen an ein Datenhaltungssystem. Die Antwort des Datenhaltungssystems auf eine Anfrage besteht in den Daten des Datenhaltungssystems, die die Anfrage erfüllen.

Die mit Abstand bekannteste standardisierte Anfragesprache ist SQL (Structured Query Language), die von IBM für relationale Datenbanksysteme entwickelt wurde.

Beispiel AUS-1: Anfrage in natürlicher Sprache und SQL

```
"Gib mir alle Titel aus der Büchertabelle, deren dazuge-
höriges Datum aus diesem Jahr ist".


SELECT titel
FROM bücher
WHERE datum="2001";
```

XQuery ermöglicht Anfragen über einzelne und mehrere Dokumente sowie die Konstruktion neuer Dokumente aus den Ergebnissen der Anfragen.

XQuery selbst folgt keiner XML-Syntax, sondern einer für Menschen leichter lesbaren Syntax. Eine XML-basierte Syntax ist jedoch in Entwicklung.

Warum XQuery?

XML-Daten lassen sich in Datenbanken abspeichern. Warum also eine eigene XML-Anfragesprache, wenn man doch SQL auf Datenbanken nutzen könnte?

- Die Anfragen lassen sich "XML-typisch" formulieren, also in XML-Termini und XML-Syntax ausdrücken und wirken auf diese Weise problemadäquater. Man muss sich nicht überlegen, wie denn die Daten in SQL anzusprechen sind, sondern kann direkt XML benutzen.
- Die Anfragen werden unabhängig von der jeweils gewählten Art der Abspeicherung formuliert und sind somit portabel und wiederverwendbar.

Die Spezifikationen

XML Query Requirements
Beschreibt die formalen Anforderungen für Datenmodell und Suchfunktionalitäten von XQuery.

XML Query Use Cases
Beschreibt einen Satz von Anwendungsfällen, die XQuery erfüllen muss.

XML Query 1.0 and XPath 2.0 Data Model
Bildet die Grundlage für XQuery. Beschreibt die Informationsobjekte, die einer Anfrage/Suche an ein XML-Dokument zugänglich sein müssen.

XQuery 1.0 and XPath 2.0 Functions and Operators
Definiert die Basisoperatoren und -funktionen der Datentypen aus XML-Schema und dem XQuery-Datenmodell zum Gebrauch in XQuery und XPath.

XML Query 1.0 Formal Semantics
Beschreibt in abstrakter Repräsentation die Kernoperationen von XQuery.

XQuery 1.0: A Query Language for XML
Eigentlich das zentrale Dokument. Während die ersten drei Spezifikationen den Focus auf das theoretische Model legen, geht es hier um den Anwendungsteil: die Beschreibung der Syntax von XQuery.

XML Syntax for XQuery 1.0 (XQueryX)
Eine alternative Syntax für XQuery in XML.

Zur Zeit liegen alle diese Spezifikationen erst als Arbeitsentwurf (*Working Draft*) vor. Es bleibt also abzuwarten, wohin die Entwicklung geht und welche

Unterstützung durch parallele Implementierungen der Standardisierungsversuch erfährt.

Welche Tools unterstützen XQuery?

Tool	Entwickler	Beschreibung
XQEngine	Fatdog (Howard Katz)	Anfragen können gegen reale Daten gestellt werden. Plattformunabhängige Implementierung in Java. ***URL:*** *http://www.fatdog.com*
QuiP	Software AG	Freier Prototyp einer XQuery-Implementierung ***URL:*** *http://developer.softwareag.com/tamino/quip/default.htm*
MS-Demo	Microsoft	Vorgefertigte Anfragen können gegen die Use-Case-Beispieldokumente aus der Spezifikation gestellt werden (Online-Demo und Download). ***URL:*** *http://xqueryservices.com*
QeXO	Per Bothner	Unabhängige Implementierung. Es gibt die Möglichkeit, über das Kawa-Framework eine Suchanfrage zu kompilieren und als Bytecode wiederzuverwenden. ***URL***: *http://www.gnu.org/software/qexo/*
XQuench	C. Wilper	Open-Source-Implementierung in Java. **URL:** *http://sourceforge.net/projects/xquench*

Ein einfaches Beispiel

XQuery ist wie SQL eine deklarative Anfragesprache, d.h. Sie sagen der Anwendung nicht, *wie* sie suchen soll, sondern *was* sie suchen soll.

XQuery erlaubt sowohl **Projektion** (Auswahl einer Knotenmenge nach bestimmten Kriterien) und **Transformation** (Erzeugung eines Ausgabedokuments, das sich vom durchsuchten Dokument unterscheidet). Beides geschieht in einer Suchanfrage.

Beachten Sie:
XQuery erlaubt auch die Suche über *mehrere* Dokumente.

In der ersten Version von XQuery ist jedoch nicht die Möglichkeit der Aktualisierung von Dokumenten vorgesehen (wie es SQL über `UPDATE` für Tabellen bietet).

Anhand eines einfachen Beispiels sei das Prinzip von XQuery-Anfragen erläutert.

Beispiel AUS-2: Einfaches XQuery-Beispiel

Folgendes Dokument soll durchsucht werden:

Ausgangsdokument

```
<?xml version="1.0" encoding="iso-8859-1"?>

<bib>
  <buch jahr="1999">
    <titel>XML kompakt</titel>
    <autor>
      <nachname>Michel</nachname>
      <vorname>Thomas</vorname>
    </autor>
    <verlag>Hanser</verlag>
    <preis>59,80</preis>
  </buch>
  <buch jahr="2000">
    <titel>
      Informationsmodellierung in XML und SGML
    </titel>
    <autor>
      <nachname>Lobin</nachname>
      <vorname>Henning</vorname>
```

```
    </autor>
    <verlag>Springer</verlag>
    <preis>69,90</preis>
  </buch>
  <buch jahr="2000">
    <titel>XML in der Praxis</titel>
    <autor>
      <nachname>Behme</nachname>
      <vorname>Henning</vorname>
    </autor>
    <autor>
      <nachname>Mintert</nachname>
      <vorname>Stefan</vorname>
    </autor>
    <verlag>Addison-Wesley</verlag>
    <preis>79,00</preis>
  </buch>
</bib>
```

Die einfachste Suchanfrage wäre nun die Ausgabe alle Bücher. Folgende Ausgabe sollte die Anfrage also erzeugen:

Ergebnisdokument

```
<buch jahr="1999">
  <titel>XML kompakt</titel>
  <autor>
    <nachname>Michel</nachname>
    <vorname>Thomas</vorname>
  </autor>
  <verlag>Hanser</verlag>
  <preis>59,80</preis>
</buch>
<buch jahr="2000">
  <titel>
    Informationsmodellierung in XML und SGML
  </titel>
  <autor>
    <nachname>Lobin</nachname>
    <vorname>Henning</vorname>
  </autor>
  <verlag>Springer</verlag>
  <preis>69,90</preis>
</buch>
....
```

Beachten Sie:
Wie bereits dieses Beispiel zeigt, muss die Ausgabe einer Suchanfrage kein wohlgeformtes XML-Dokument sein.

Die zugehörige Suchanfrage lautet:

XQuery-Suche

```
FOR $buch IN document("bib.xml")//buch
RETURN $buch
```

Der Ausdruck `FOR $buch IN document("bib.xml")//buch` stößt eine Iteration über die Menge aller Elemente an, die dem XPath-Ausdruck `document("bib.xml")` genügen. Dabei wird jedes Element der Reihe nach der Variable `$buch` zugewiesen. Die `document()`-Funktion ist eine XPath-Funktion und importiert das als Argument übergebene Dokument, wobei der Wurzelknoten als Startpunkt zurückgeliefert wird.

Der Ausdruck `RETURN $buch` erzeugt eine Sequenz (keine Knoten-Menge, wie sonst bei Ergebnissen von XPath-Knoten-Tests) von <`buch`>-Elementen, wobei der Ausdruck in Klammern evaluiert und dann ins Ergebnis kopiert wird.

Beachten Sie:
Die verwendeten Variablen sind keine zugewiesenen, sondern gebundene Variablen, d.h. ihr Wert ist nicht mehr änderbar. Dieser Variable kann kein anderer Wert mehr während der Anfrage zugewiesen werden.

FLWR-Ausdrücke

Die XQuery-Syntax basiert auf sogenannten FLWR-Ausdrücken (gesprochen: "flower"; *FLRW = FOR-LET-WHERE-RETURN*):

```
FLWRExpr ::= (ForClause | LetClause) + WhereClause? + return Expr
```

Beachten Sie:
XQuery-Anfragen sind verschachtelbar, da ein `RETURN`-Ausdruck wieder einen FLWR-Ausdruck enthalten kann.

Die WHERE-Klausel

Die WHERE-Klausel formuliert zusätzliche Selektionsbedingungen an die Projektion und ist optional.

Betrachten wir zunächst ein Beispiel mit WHERE-Ausdruck. Die folgende Anfrage schränkt die Ausgabe der Bücher auf ein Buch ein, dessen Titel XML kompakt lautet.

Beispiel AUS-3: WHERE-Ausdruck

```
FOR $buch IN document("bib.xml")//buch
WHERE $buch/titel="XML kompakt"
RETURN $buch
```

Beachten Sie:
Sie können die Filterung auch über XPath-Prädikate im XPath-Ausdruck nach dem IN vornehmen. Obiges Beispiel ist also äquivalent zu dem folgenden:

Beispiel AUS-4: Nutzung von XPath-Prädikaten statt WHERE-Klausel

```
FOR $buch IN document("bib.xml")//buch[titel="XML kom-
pakt"]
RETURN $buch
```

Element-Konstruktoren

Innerhalb des RETURN-Ausdrucks können Sie auch neue Tags setzen, sogenannte **Element-Konstruktoren**.

Folgende Suche gibt z.B. eine HTML-Liste aller Buchtitel aus:

Beispiel AUS-5: Element-Konstruktoren

```
FOR $titel IN document("bib.xml")//buch/titel
RETURN <LI>{string-value($titel)}</LI>
```

Einbettung einer Suchanfrage in Tags

Sie können die Suchanfrage auch in z.B. ein HTML-Dokument-Gerüst einbetten und so eine Transformation von XML nach HTML durchführen.

Folgende Suchanfrage bettet die Liste der Buchtitel gleich in ein HTML-Dokument ein:

Beispiel AUS-6: In HTML eingebettete Suchanfrage

```
<HTML>
  <HEAD>
    <TITLE>Ergebnis einer Suchanfrage</TITLE>
  </HEAD>
  <BODY>
    <H1>Alle Titel des XML-Literaturverzeichnisses</H1>
    <UL>
      {
        FOR $titel IN document("bib.xml")//buch/titel
        RETURN <LI>{string-value($titel)}</LI>
      }
    </UL>
  </BODY>
</HTML>
```

Der LET-Ausdruck

Über LET-Ausdrücke kann man einer Variablen Knotensequenzen zuweisen.

Eine Suchanfrage mit LET könnte folgendermaßen aussehen:

Beispiel AUS-7: LET-Ausdruck

```
LET $autoren :=document("bib.xml")//autor
RETURN
<AUTOREN>
{
  $autoren
}
</AUTOREN>
```

Hier werden alle <autor>-Elemente des zu durchsuchenden Elements ausgewählt und ausgegeben innerhalb eines Elements namens <AUTOREN>.

Die Variable kann auch in einem weiteren FOR-Ausdruck verwendet werden.

XML-Syntax

Nur zur Demonstration die entsprechende sehr schreibintensive XML-Syntax:

Beispiel AUS-8: Beispielanfrage in XML-Syntax

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

<q:query
  xmlns:q="http://www.w3.org/2001/06/xqueryx"  /">
```

```
  <q:flwr>
    <q:letAssignment q:variable="$autoren">
      <q:step q:axis="descendant-or-self">
        <q:function q:name="document">
          <q:constant q:datatype="xs:string">
            bib.xml
          </q:constant>
        </q:function>
        <q:identifier>autor</q:identifier>
      </q:step>
    </q:letAssignment>
    <q:return>
      <q:elementConstructor>
        <q:tagName>
          <q:constant q:datatype="xs:qname">
            AUTOREN
          </q:constant>
        </q:tagName>
        <q:constant q:datatype="xs:string"></q:constant>
        <q:variable>$autoren</q:variable>
        <q:constant q:datatype="xs:string"></q:constant>
      </q:elementConstructor>
    </q:return>
  </q:flwr>
</q:query>
```

Komplexere Suchanfragen

Folgende Suchanfrage zeigt, wie LET und FOR in Kombination genutzt werden können und verwendet darüber hinaus im RETURN-Ausdruck einen weiteren FLWR-Ausdruck.

Die Suchanfrage gibt eine Liste aller Bücher aus, sortiert nach den Autoren. Im Grunde wird also der Baum invertiert.

Beispiel AUS-9: Komplexe Suchanfrage

```
LET $autoren := distinct-
values(document("bib.xml")//autor)
LET $buecher := document("bib.xml")//buch
FOR $autor IN $autoren
RETURN
  <autor>
    <name>{string-value($autor)}</name>
    <buecher>
      {
        FOR $buch IN $buecher
        WHERE $buch/autor=$autor
        RETURN <buch>{string-value($buch/titel)}</buch>
```

```
      }
    </buecher>
  </autor>
```

Eingebaute Funktionen und Operatoren

Eingebaute Funktionen gibt es - wie in XPath und XSLT - auch in XQuery. In den obigen Beispielen haben Sie bereits die Funktionen `document()`, `string-value()` und `distinct-values()` kennen gelernt.

XQuery-Vorfahren

Die Beispiele zeigen, das XML-Query ein Mix aus verschiedenen bereits für XML-Anfragen entwickelten Sprachen ist. Ziel ist es natürlich, von jeder Sprache das Beste zu "erwischen".

Anfragesprache	**Übernahme von**
Quilt	der gesamten Grundidee. XQuery lässt sich als Weiterentwicklung von Quilt sehen.
XPath	Pfadbeschreibungs-Syntax für Navigation in hierarchischen Dokumenten.
XQL	
XML-QL	gebundenen Variablen und deren Benutzung zur Konstruktion neuer Strukturen.
SQL	der Idee einer Serie von Klauseln die auf Schlüsselwörtern basieren und ein Muster zur Restrukturierung von Daten liefern (SELECT-FROM-WHERE-Muster).
OQL	der Idee einer funktionalen Sprache, die sich aus verschiedenen Arten von Ausdrücken zusammensetzt, die miteinander verschachtelt werden können.

XML und Metadaten: RDF

Was sind Metadaten?

Metadaten = Informationen/Daten über Daten

Beispiel:
Ein Bibliothekskatalog enthält Metadaten, da er Informationen über Bücher enthält.

Metadaten sind ein Mittel, um die Masse von Informationen im World Wide Web in den Griff zu bekommen. Metadaten beschreiben Web-Seiten oder Teile von Web-Seiten. Allgemein spricht man von *Web-Ressourcen*.

Beispiel ANW-1: `<meta>`-Tag in HTML

```
<head>
  <title>Willkommen auf unserer Homepage</title>
  <meta name="author" content="Peter Hornfeld">
  <meta name="keywords" content="XML und RDF">
  <meta name="date" content="2001-05-21">
  <meta name="powered-by" content="http://www.w3.org">
  <meta name="resource-type" content="WWW-Portal">
</head>
```

Was ist RDF?

RDF = Resource Description Framework

RDF stellt den technischen Rahmen (Framework) zur Verfügung, um beliebige Datenobjekte (Resource) zu beschreiben (Description).

RDF ist keine XML-DTD, sondern eine Konvention, die festlegt, wie XML-Dokumente aussehen können, die Metadaten darstellen.

Ziel:
Entwicklung einer Sprache für den Austausch maschinen-*verständlicher* Beschreibungen von Web-Ressourcen. RDF soll die Grundlage für ein Web schaffen, in dem Daten über Metadaten so beschrieben sind, dass Program-

me eine intelligentere und bequemere Nutzung dieser Daten erlauben (**semantisches Web**).

Wozu RDF, wenn XML selbstbeschreibend ist?

Beipielszenario:
Ein Agent, der z.B. Anschriften von Professoren anhand der WWW-Seiten von Universitäten im Web recherchieren soll.

Problem:
Verschiedene Universitäten benutzen verschiedene DTDs zur Auszeichnung von Adressinformationen. So könnte z.B. der Wohnort mittels folgender Attribute oder auch Elemente beschrieben werden: `ort`, `wohnort`, `stadt`, `adressfeld3` usw.

Folge:
Der konkret gewählte Name des Attributs ist also nur einer von vielen einleuchtenden Alternativen und kann durch eine Maschine nicht verlässlich gedeutet werden.

Lösungsmöglichkeiten:

- Harte Codierung aller Möglichkeiten in den Agenten
- Standardisierung
- Anreicherung der XML-Dokumente mit Metadaten, die eine automatische Bedeutungs-Verarbeitung durch den Agenten ermöglichen.

RDF realisiert letztere Möglichkeit, indem es standardisierte Elemente und Attribute definiert, die zur Formulierung der Semantik von Dokumenten benutzt werden können.

=> Unabhängigkeit der Metadaten von der verarbeitenden Anwendung

RDF-Spezifikationen:

- RDF Primer
- RDF Semantics
- RDF/XML Syntax Specification
- RDF Schema
- RDF Test Cases

RDF Model

Das RDF Model legt die allgemeine Syntax und das zugrundeliegende (Daten-)Model zur Beschreibung von Web-Ressourcen fest.

Das RDF-Model beruht auf folgenden Regeln:

- Eine **Ressource (*Resource*)** ist alles, was durch einen URI eindeutig identifizierbar ist, insbesondere also Web-Seiten, aber auch individuelle Elemente eines XML-Dokuments oder ganze Kollektionen von Web-Seiten. Es können aber auch ganz andere Objekte wie Bücher beschrieben werden.

Beispiel ANW-2: Die XML-Seminardisposition als Ressource

```
http://www.ges-training.de/xml
```

- Eine **Eigenschaft (*property*)** ist eine Charakteristik (Attribut oder Relation), die geeignet ist, eine Ressource zu beschreiben, z.B. `Autor` oder `Titel.` Eine Eigenschaft kann ebenfalls Ressource sein und selbst wieder Eigenschaften haben.
- Eine **Aussage (*statement*)** ist eine Kombination aus einer Ressource, einer Eigenschaft und einem Wert (vergleichbar mit Subjekt, Prädikat und Objekt einer Aussage). Eine RDF-Aussage besteht immer aus diesen drei Bestandteilen. In einer Aussage wird der Wert einer Eigenschaft einer Ressource definiert.

Beispie -ANW-3: Aussage

```
"Franz Josef Herpers ist der Autor der Ressource
http://www.ges-training.de/xml"
```

Grafisch ließe sich die Aussage folgendermaßen visualisieren:

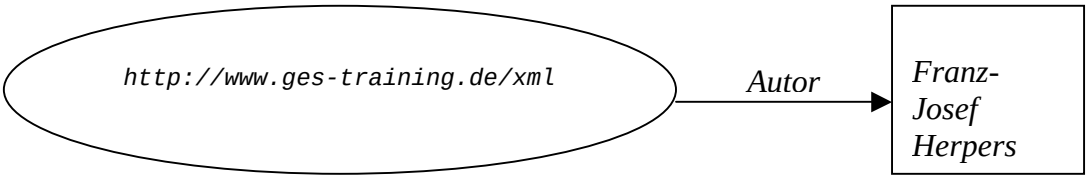

Abbildung AUS-1: RDF-Graph für eine Aussage

Die obige Aussage hat folgende Teile:

Subjekt (Ressource)	`http://www.ges-training.de/xml`
Prädikat (Eigenschaft)	Autor
Objekt (Wert)	"Franz-Josef Herpers"

Dieses Model lässt sich relativ einfach mit einer XML-Syntax ausdrücken. Aber: XML-Syntax ist nur eine mögliche Syntax! Sie ist jedoch die geeignetste, um RDF-Statements leicht austauschbar zu machen.

Beispiel ANW-4: RDF-Dokument in XML-Syntax

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

<rdf:RDF
    xmlns:rdf="http//www.w3.org/1999/02/22-rdf-syntax-
ns#"
    xmlns:dc="http://purl.org/metadata/dublin_core#">
  <rdf:Description about="http://www.ges-
training.de/xml">
    <dc:Title>Detaildispo: XML-Technologien</dc:Title>
    <dc:Publisher>GES</dc:Publisher>
    <dc:Creator>Franz-Josef Herpers</dc:Creator>
  </rdf:Description>
</rdf:RDF>
```

Dadurch dass Ressourcen eindeutig identifizierbar sind, wird es möglich, ein ganzes Netz von Beziehungen aufzubauen. So können auch komplexere Sachverhalte ausgedrückt werden:

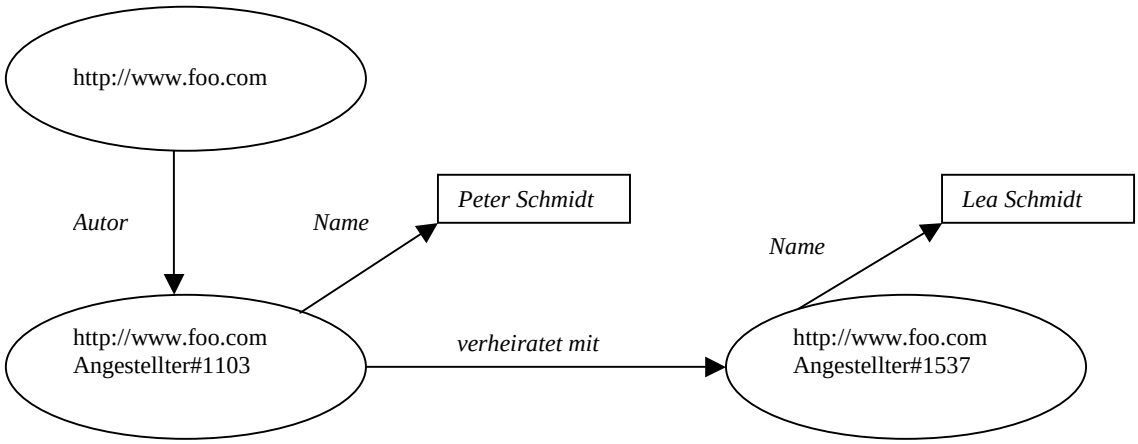

Abbildung AUS-2:Komplexerer RDF-Graph

RDF-Schema

RDF-Schema standardisiert eine Sprache zur Definition von Metadaten-Vokabularen. RDF definiert selbst keine Eigenschaften, stellt aber eine Syntax zur Deklaration solcher Eigenschaften zur Verfügung.

=> RDF-Schema kommt bei der Definition von RDF-Anwendungen die gleiche Aufgabe zu wie DTDs bei XML-Anwendungen.

Designziele von RDF

Unabhängigkeit

Da eine Eigenschaft selbst wieder eine Ressource ist, kann jede unabhängige Organisation Eigenschaften definieren. Verschiedene Organisationen können auf diese Weise ihre spezifischen Vokabulare für Metadaten festlegen. RDF legt keine spezifischen Eigenschaften fest.

Austauschbarkeit

RDF-Aussagen können in XML ausgedrückt werden und sind somit leicht austauschbar.

Erweiterbarkeit

RDF-Aussagen sind einfach und deswegen auch in großer Anzahl leicht zu verarbeiten und zu durchsuchen. Im Web wird es Billionen solcher Metadaten-Aussagen geben, die leicht durchsuchbar und verarbeitbar sein müssen. Erweiterbarkeit ist vor diesem Hintergrund besonders wichtig.

Eigenschaften können Ressourcen sein

Eigenschaften können wieder Eigenschaften besitzen. Dies ist wichtig, da es viele Eigenschaften im Web geben wird, zu viele, um sie der Reihe nach durchzugehen. Wenn ich z.B. wissen will, ob irgendjemand eine Eigenschaft definiert hat, die Filme beschreibt, mit Werten wie Komödie, Horror, Romanze, Thriller usw., dann kann ich diese finden über die Eigenschaften dieser Eigenschaften.

Werte können Ressourcen sein

Dies ist z.B. notwendig um auszudrücken, dass eine Website (Ressource) eine Homepage (Startseite) `http://www.foo.de` hat.

Aussagen können Ressourcen sein (Reification)

Auch Aussagen können Eigenschaften haben. Es sollte also möglich sein bei einer Aussage wie "Das Thema dieser Seite ist der Wetterbericht" zu fragen, "Wer sagt das?" Wann wurde das gesagt" Auf diese Weise lässt sich etwa die Verlässlichkeit und die Aktualität von Aussagen prüfen (*Web of Trust*).

Anwendungen

- Ressourcenentdeckung (z.B.: Smart Browsing im Netscape Navigator, das zu einer angewählten Webseite andere Webseiten mit ähnlichem Inhalt bietet.)
- Inhalts- und Strukturbeschreibung einer Web-Site (Site-Map)
- Intelligente Softwareagenten
- Inhaltsbewertung (*Content Rating*)
- Beschreibung von einzelnen im Web verteilten Seiten, die zusammen ein logisches Dokument bilden.
- Beschreibung von Copyright-Rechten
- Beschreibung der Praktiken, die Web-Site-Betreiber beim Umgang mit Benutzerdaten verwenden. (P3P)
- Filterung nicht kindergerechter Inhalte (PICS).
- Beschreibung von Channels (RSS)

XML-Programmierung mit Java

XML gives Java something to do.

John Bosak

Warum XML?

Das sollten sie nun wissen :-)

Warum Java?

Java ist eine ausgereifte, objektorientierte Programmiersprache, die zunehmend für (verteilte) Internetapplikationen zum Einsatz kommt.

In diesem Zusammenhang sind die wichtigsten Eigenschaften von Java:

- Portabilität
- Sicherheit
- Verteilung
- Multithreadingfähigkeit

Die Symbiose: XML und Java

Java und XML ergänzen sich auf perfekte Weise: XML steuert plattformunabhängige Daten bei und Java bringt plattformunabhängige Verarbeitung mit ein.

Sowohl Java als auch XML sind auf die Bedürfnisse von Internetapplikationen zugeschnitten.

<table>
<tr><th>Java</th><th>XML</th></tr>
<tr><td>von Beginn an Unterstützung von Sockets, HTTP, HTML und Servern</td><td></td></tr>
<tr><td colspan="2">Unicode-Unterstützung für internationalisierte Anwendungen</td></tr>
<tr><td>bietet Möglichkeiten zur Repräsentation komplexer Datenstrukturen und objektorientierter Modelle</td><td>ideal für komplexe, hierarchische Daten</td></tr>
<tr><td>die Mehrheit der XML-Tools sind in Java geschrieben</td><td></td></tr>
<tr><td colspan="2">Die DOM und SAX-Spezifikation beinhalten ein Java-Language-Binding</td></tr>
</table>

Java APIs für XML

Dokumentenorientiert

Java API for XML Processing (JAXP)

Das Java API for XML Parsing (JAXP) ermöglicht den Zugriff auf beliebige XML-Dokumente mittels DOM, SAX und XSLT. Anwendungen können die beliebigsten XML-Inhalte lesen, verändern und transformieren und im Anschluss weiterleiten oder abspeichern.

Ein großer Vorteil des API besteht in der Möglichkeit, den XML-Prozessor dynamisch zu wechseln. Entwickler haben somit die Möglichkeit, sich aus der Menge der vorhandenen XML-Prozessor-Implementierungen den jeweils geeigneten heraussuchen zu können (Geschwindigkeit, Speicherverbrauch,...) ohne den Anwendungscode zu ändern.

Das aktuelle Release JAXP 1.2 enthält die aktuelle SAX-2 und DOM Level-2 Technologie sowie ein XSLT-Framework, basierend auf TrAX (Transformation API for XML). XML-Namensräume und XML-Schemata werden unterstützt.

URL:
`http://java.sun.com/xml/xml_jaxp.html`

JAXP 1.1 (ohne XML-Schema-Unterstützung) ist im JDK 1.4 enthalten.

Daten- bzw. funktionsorientiert

Java Architecture for XML Binding (JAXB)

Die Java Architecture for XML-Binding (JAXP) besteht aus einem API und Werkzeugen zum automatischen Mappen von XML-Dokumenten und Java-Objekten. Aus einem XML-Schema heraus können eine oder mehrerer Java-Klassen erzeugt werden. Die generierten Klassen übernehmen dabei alle notwendigen Arbeiten zum Lesen, Parsen und Schreiben von Dokumenten. JAXB bedient sich eigener Mechanismen zum Parsen der XML-Dateien und dem Aufbau der Java-Objekte. Hierbei wird meist eine höhere Geschwindigkeit als über den "normalen" JAXP-Weg erreicht.

URL:
`http://java.sun.com/xml/jaxb/index.html`

Java API for XML Messaging (JAXM)

Das Java API for Messaging (JAXM) definiert einen Standard für Nachrichten und deren Austausch zwischen verschiedenen Systemen auf Basis von XML. Dabei wird das genaue Format der Nachricht und die möglichen Protokolle

offen gelassen. Mögliche Formate sind zum Beispiel SOAP oder auch ebXML. Folgende Ziele werden mit JAXM verfolgt:

- Unterstützung für Standard-Nachrichtenformate und Adressen.
- Sichere Auslieferung von Nachrichten, Verfolgung des Weges einer Nachricht.
- Unterstützung für Nachrichtentemplates.
- Definition der zu nutzenden Datentypen und Formate.
- Definition von Zugriffskontrolle, Sicherheitsmechanismen sowie Authentifikation und Autorisation.

URL:
`http://java.sun.com/jaxm/index.html`

Java API for XML Remote Procedure Call (JAX-RPC)

Das Java API for Remote Procedure Call wird ein Framework zur Verfügung stellen, welches die Kommunikation zwischen zwei Systemen (zum Beispiel zweier Anwendungen auf verschiedenen Computern) ermöglicht. Ein Methodenaufruf wird in ein XML-Paket umgewandelt und zum Empfänger geleitet. JAX-RPC wird den Export von WSDL-basierten Dokumenten unterstützen. Obwohl die Nähe zu XML vermuten lässt, das auch andere Sprachen teilnehmen können, ist jedoch vorerst die Unterstützung von Java geplant. Genau genommen werden mit diesem API die folgenden Ziele verfolgt:

- Definition eines API für Marshalling und Unmarshalling von Argumenten.
- Definition des Mappings von Java-Methoden zu XML-basierten RPC-Aufrufen und deren Ergebnissen.

URL:
`http://java.sun.com/xml/xml_jaxrpc.html`

Java API for XML Registries (JAXR)

Das Java API for XML Registries (JAXR) definiert einen Standard für den Zugriff auf XML-Registries. Der anfängliche Support sieht die Unterstützung von ebXML-Registries und UDDI vor. Das JAXR API ermöglicht somit den Zugriff auf die verschiedensten Informationen rund um Dokumentstandards, Firmen und Organisationen, aber auch die gemeinsame Nutzung von Web-Services. Der Zugriff auf die Registry kann dabei sowohl lokal als auch über das Internet erfolgen.

URL:
`http://java.sun.com/xml/xml_jaxr.html`

TOOL-TIPP

Einige der obigen APIs sind von Sun zusammen mit einigen nützlichen Tools im sogenannten Java Web Services Developer Pack (Java WSDP) zusammengefasst worden.

URL:
`http://java.sun.com/webservices/webservicespack.html`

JDOM

Die JDOM-API besteht aus einer Spezifikation, die von Brett McLaughlin und Jason Hunter mit der Unterstützung von James Duncan Davidson (dem Autor der JAXP-Spezifikation) geschrieben wurden. Sie definiert das Verhalten einer sehr leichtgewichtigen Sicht auf das XML-Dokument. JDOM versucht, eine hochleistungsfähige Alternative zu SAX und DOM für die meisten Anwendungsfälle bereitzustellen (80/20-Regel). D.h. JDOM unterstützt *nur* Java und unterhält auch nicht so eine strenge Baum-Repräsentation wie DOM.

URL:
`http://www.jdom.org/`

Enterprise Application Integration (EAI)

Was ist EAI?

EAI = Enterprise Application Integration

Das Ziel der EAI ist es, verschiedene Systeme so zusammenarbeiten zu lassen, dass sie wie ein System erscheinen.

Zur EAI gehört sowohl die nahtlose Integration von Geschäftsprozessen, um E-Business zu ermöglichen als auch der Datenaustausch bzw. das Nutzen einer gemeinsamen Datenbasis.

- Bis Mitte der 90er Jahre monolithische Systeme für eine Aufgabe (Vertrieb, Einkauf, Human Resources)
- danach Paradigmenwechsel: modulare Komponenten, die sich aber integrieren lassen (intelligentere Systeme)
- Verbindung von A nach B, ohne dass A von B genaueres weiß! (Webservices)
- Früher auf Seiten von A und B je ein Entwicklerteam => hohe Kosten (s. EDI!), heute: Many-To-Many-Integration
- EAI ist das neue Marketingwort für Middleware, umfasst aber mehr!

Integrationsebenen

- Daten
- Funktionen
- Prozesse

Sonstige XML-Technologien

Spezifikationen

- XLink (erweitert)
- XPointer
- XInclude
- XForms
- XBase
- XML Infoset
- Canonical XML
- XSLT 2.0 / XPath 2.0

Schlagwörter

- Data Binding
- Content Syndication