Mitschrift der Vorträge:

DOM
RDF
XForms

Netzbasierte Informationssysteme
Freie Universität Berlin

Wojtek Polcwiartek
Berlin, 27.01.2005

Dom

Der Vortrag von Kolleginnen Opitz und Richter, der am 16.01. gehalten wurde, hatte das Thema „DOM (Document Object Model)“.
Der Vortrag bestand aus folgenden Teilen:

1. Einführung
2. Dom Module
3. Dom Intern
4. Dom Level 2 HTML
5. HTML - Elemente

Ad. 1 Einführung

Document Object Model ist eine Spezifikation von Schnittstellen, die das Objekt-Modell von Web-Dokumenten beschreiben. Ein objektorientiertes Model definiert die Attribute und Methoden von Dokumenten, so wie von ihren Teilen und Elementen. Ein Dokument, laut DOM, besteht aus Objekten (sog. Knoten – eng. Nodes), die in einer Baumstruktur organisiert sind. DOM erlaubt den dynamischen Zugriff und die Modifikation von Inhalt und Struktur von Dokumenten. DOM ist sprach- und plattformneutral.

Beispiel:

Die Tabelle

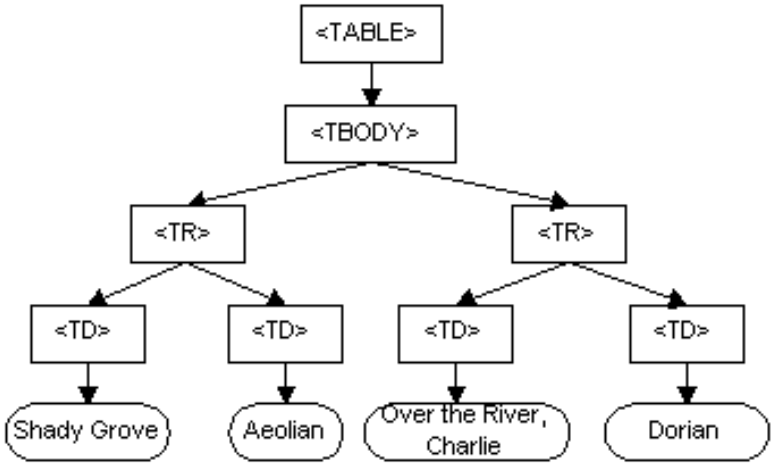
Shady Grove	Aeolian
Over the River, Charlie	Dorian

in HTML

```
<TABLE border="1">
<TBODY>
<TR>
<TD>Shady Grove</TD>
<TD>Aeolian</TD>
</TR>
<TR>
<TD>Over the River, Charlie</TD>
<TD>Dorian</TD>
```

</TR>
</TBODY>
</TABLE>

sieht in DOM (bei einer Baum-Implementierung) wie folgt aus:

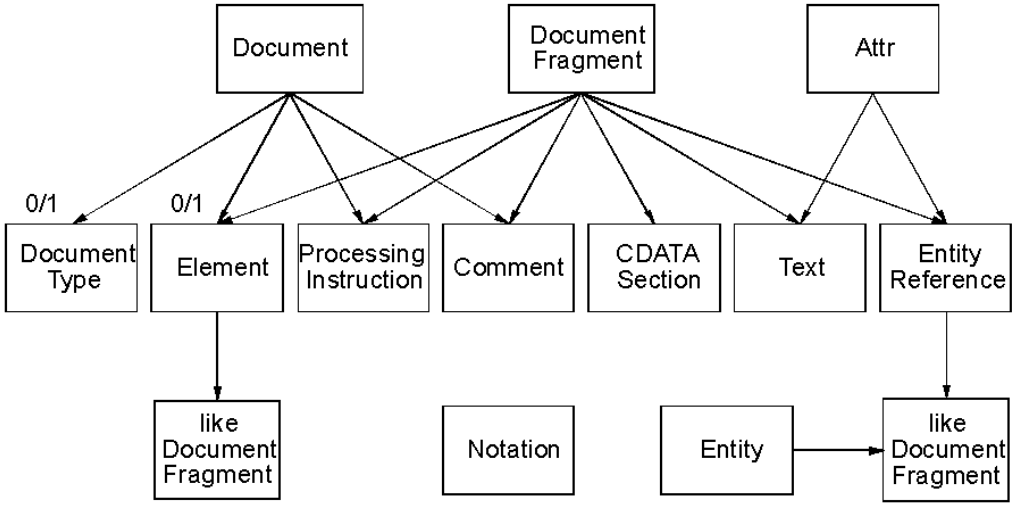


Beispiel: Quelle W3C

Ad. 2

Erste DOM Spezifikation – sog. DOM Level 0 erschien als Durchschnitt davon, was bereits Microsoft und Netscape in ihren Webbrowser implementiert hatten. Im Jahr 1998 entstand erste DOM – Spezifikation, die von W3C stammte. Das Ziel von W3C in der Zusammenarbeit mit führenden IT-Unternehmen (u.a. Miscrosoft, Sun, IBM) war ein einheitliches Interface für Behandlung von Webdokumenten auf den Markt zu bringen. Eine DOM Implementierung muss mindestens das „Core“ oder HTML Modul beinhalten. Weitere Module (z.B. Events (Ereignisse), Traversal (Dokumentbäume Traversieren)) stehen als Erweiterung zu Verfügung.
Das Core Modul beschreibt welche Knotentypen (Objekte) welche anderen Typen in einem Webdokument enthalten können:

- Document hat 0 oder 1 DocumentType. Das einzige Element ist das root-Element, welches dann das gesamte Webdokument enthält.
- DocumentFragment dient als Container, um beliebige Teile eines Dokuments zwischenzuspeichern



Im Jahr 2000 war die Spezifikation von DOM Level 2 fertig. Diese Spezifikation unterstützt Namensräume und Style Sheets.

Ad. 3
Der DOM – Parser nach dem Überprüfen von Syntax bildet das ganze Dokument als ein Baum ab, und ermöglicht einen Zugriff zu den DOM-Baumknoten. Heutige populäre Webbrowser (z.B. IE, NN) enthalten meistens einen DOM-Parser. Der ist aber auch erhältlich als eine Bibliothek für diverse Programmiersprachen (u.A. Xerces von Apache Foundation für Java).

Ad. 4

Das HTML Modul ist eine Erweiterung von Core-Modul und vererbt ganze Funktionalität von dessen Schnittstellen. Das HTML Level 2 Modul beschreibt Objekte und Methoden die für HTML 4.01 und XHTML 1.0 spezifisch sind. Für eine große Anzahl von HTML-Tags wurde eine eigene Schnittstelle definiert. Dies sollte eine einfachere Script Verarbeitung ermöglichen.

Ad. 5

Das HTML Level 2 definiert eine Menge von Schnittstellen, einige werden hier kürzlich beschrieben.
a) HTMLDocument - ist eine Spezialisierung von Document aus dem Modul Core und repräsentiert das gesamte HTML-Dokument.

```
interface HTMLDocument : Document {
        attribute DOMString title;
        readonly attribute DOMString referrer;
        readonly attribute DOMString domain;
        readonly attribute DOMString URL;
        attribute HTMLElement  body;
        readonly attribute HTMLCollection  images;
        readonly attribute HTMLCollection  applets;
```

```
            readonly attribute HTMLCollection  links;
            readonly attribute HTMLCollection  forms;
            readonly attribute HTMLCollection  anchors;
            attribute DOMString cookie;
                  // raises(DOMException) on setting
      void open();
      void close();
      void write(in DOMString text);
      void writeln(in DOMString text);
      NodeList getElementsByName(in DOMString elementName);
};
```

b) HTMLElement - ist eine Spezialisierung von Element aus dem Modul Core. Es speichert bestimmte Attribute wie class, title und id direkt als DOMString ab.

```
      interface HTMLElement : Element {
            attribute DOMString      id;
            attribute DOMString      title;
            attribute DOMString      lang;
            attribute DOMString      dir;
            attribute DOMString      className; };
```

c) HTMLCollection entspricht der NodeList des Moduls Core; es ist eine Menge von Node-Objekten. Auf einen bestimmten Knoten greift man entweder mittles Index zu oder man gibt einen Namen an, der im gesuchten Knoten als id – Attributwert gesetzt ist.

```
interface HTMLCollection {
      readonly attribute unsigned long length;
      Node item(in unsigned long index);
      Node namedItem (in DOMString name);
};
```

d) HTMLBodyElement ist eine Spezialisierung von HTMLElement und repräsentiert den Körper (Body) der HTML-Datei. Es definiert neue Attribute wie background, bgcolor und link

```
interface HTMLBodyElement : HTMLElement { attribute DOMString aLink;
      attribute DOMString background;
      attribute DOMString bgColor;
      attribute DOMString link;
      attribute DOMString text;
      attribute DOMString vLink;
};
```

RDF

Der Vortrag von Michal Starosta und Tobias Escher, der am 20.01. gehalten wurde, hatte das Thema „RDF (Resource Description Framework)“.
Der Vortrag bestand aus folgenden Teilen:

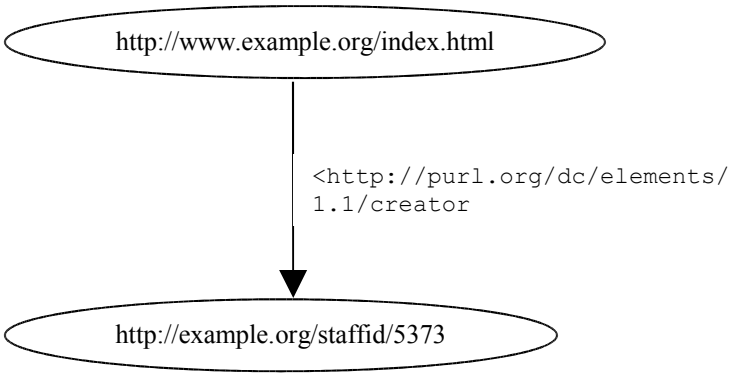
- Idee RDF/Darstellungsmöglichkeiten
- RDF/XML
- RDF Collections
- RDF Schema
- RDF Aplikationen

Ad. 1

RDF ist eine W3C Empfehlung. Es sollte die chaotisch semantische Situation im heutigen Web verbessern. RDF ist eine von mehreren Möglichkeiten den Webressourcen semantische Bedeutung mit Hilfe von sog. Metadaten zu geben. Es ermöglicht beschreibende Aussagen über Ressourcen zu erstellen.

Beispiel Aussage (aus der Vorlesung):
http://www.example.org/index.html has a creator whose name (value) is John Smith.
Diese Aussage besteht aus Subjekt, Prädikat und Objekt.
Subjekt: http://www.example.org/index.html
Prädikat: creator
Objekt: John Smith

RDF sollte eine Methode sein, die Objekte, Subjekte und Prädikate von derartigen Aussagen identifizieren zu können, um später sie bearbeiten zu können.
Unsere Beispielaussage (http://www.example.org/index.html has a creator whose name (value) is John Smith) kann man als ein solches Graph darstellen:

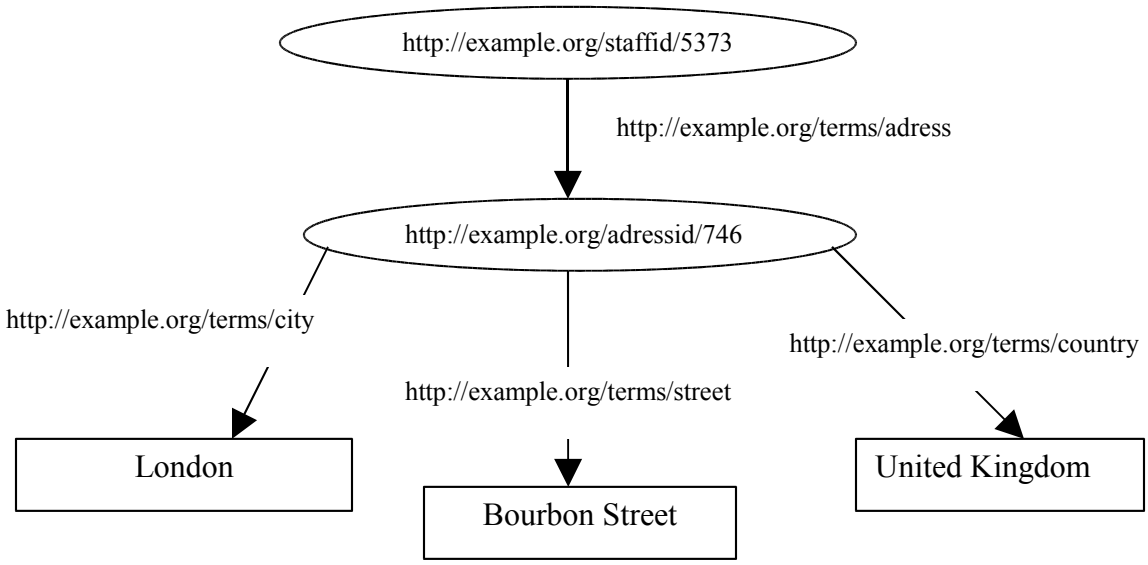


Oder als ein sog. Tripel (eine Aussage nur aus Subjekt, Prädikat und Objekt) :
<http://www.example.org/index.html >
<http://purl.org/dc/elements/1.1/creator>
<http://example.org/staffid/5373>,
oder dasselbe in XML qualified name (Qname):

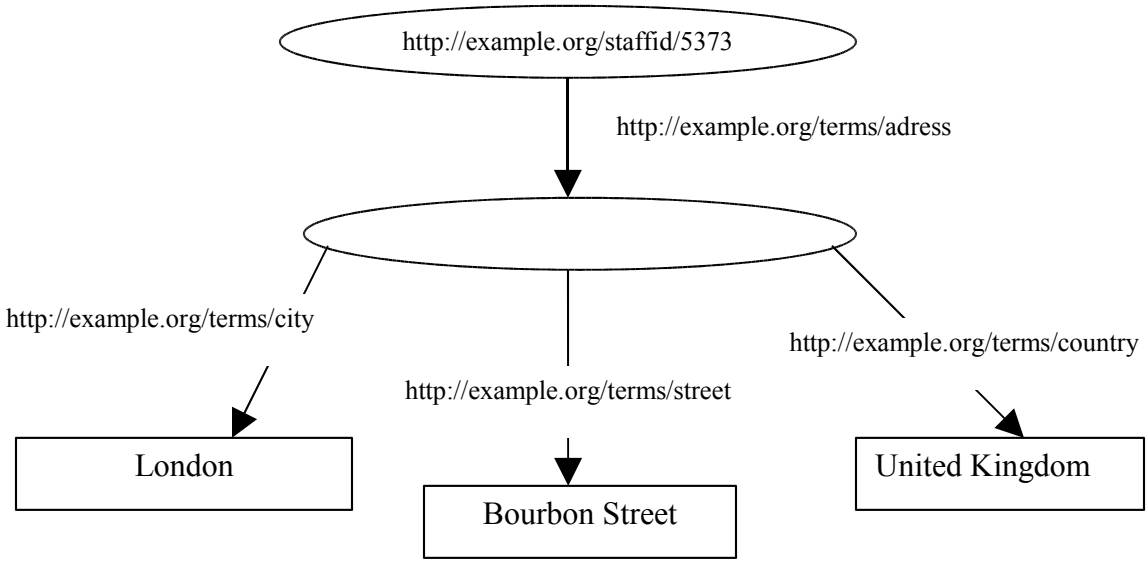
```
ex:index.html          dc:creator     exstaff:5373
```

Bei Vereinfachung von komplexen Graphen oder bei Ressourcen, die keinen URI haben, aber von anderen URIs beschrieben werden, helfen leere Knoten (blank Nodes). Die leeren Knoten können benannt werden.
Beispiel:

Ohne leere Knoten:



Mit leeren Knoten:



Ohne leeren Knoten als Tripel in Qname Format:

```
exstaff:5373    exterms:adress          exadressid:7464
exadressid:7464 exterms:city            “London”
exadressid:7464 exterms:street          “Bourbon Street”
exadressid:7464 exterms:country         “United Kingdom“
```

Mit leeren Knoten als Tripel in Qname Format:

```
exstaff:5373    exterms:adress          ???
???                     exterms:city            “London”
???                     exterms:street          “Bourbon Street”
???                     exterms:country         “United Kingdom“
```

Mit leeren Knoten als Tripel in Qname Format (nach Benennung):

```
exstaff:5373    exterms:adress          _:johnadress
_:johnadress    exterms:city            “London”
_:johnadress    exterms:street          “Bourbon Street”
_:johnadress    exterms:country         “United Kingdom“
```

Ad. 2

RDF Datei ist eine XML Datei. Das obige Beispiel (mit leeren Knoten) als XML-Datei:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:exterms="http://www.example.org/terms/">

<rdf:Description rdf:about=" http://example.org/staffid/5373">
<exterms:address rdf:nodeID=”johnadress” />
</rdf:Description>

<rdf:Description rdf:nodeID="johnadress">
        <exterms:city>London</exterms:city>
        <exterms:street>Bourbon Street</exterms:street>
        <exterms:country>United Kingdom</exterms:country>
</rdf:Description>

</rdf:RDF>
```

RDF ermöglicht auch Gruppen von Objekten in „Container“ zu beschreiben. Es gibt drei Arten von Containern:

- RDF:Seq – mit Berücksichtigung der Reihenfolge der Elemente
- RDF:Bag – ohne Berücksichtigung der Reihenfolge den Elemente
- RDF:Alt – für alternative Elemente

Ad. 3

Die „Container“ bilden Gruppen von Objekten, aber sie müssen nicht alle

Mitgliedobjekte nennen. Dieses Problem wird in RDF mit Hilfe von „Collections“ gelöst. Die beschreiben eine Gruppe mit einer konkreten Anzahl von Mitgliedern.

Ad. 4

Manchmal wird in RDF Anwendungen gebraucht eine Aussage mit Hilfe von einer anderen Aussage zu machen. Dabei hilft ein „reifictacion“ Mechanismus. Eine von solchen Anwendungen ist die zusätzlichen Informationen über eine Aussage beizufügen.

Beispiel:

In diesem Tripel:

```
ex:index.html          dc:creator     exstaff:5373
```

hätte man gern Informationen über dem Webservertyp.
Der Tripel wird umformiert (Objekt, Subjekt und Prädikat werden explizit aufgezählt) und die zusätzliche Information kann beigefügt werden. Es entsteht sog. „reification quad“

```
ex:comp44              rdf:type               rdf:Statement
ex:comp44              rdf:subject            ex:index.html
ex:comp44              rdf:predicate          dc:creator
ex:comp44              rdf:object             exstaff :5373
ex:comp44              exterms:server         „Apache2“
```

Ad. 4

RDF Schema stellt die Möglichkeit zu Verfügung Klassen und Eigenschaften zu beschreiben, und weist darauf hin, welche Klassen und Eigenschaften zusammen benutzt werden. Man kann sagen RDF Schema ist ein Typensystem für RDF, der den OO-Sprachen ähnlich ist.

Beispiel (Klassen Definition (Quelle W3C)):
Als Tripel:

```
ex:MotorVehicle        rdf:type           rdfs:Class .
ex:PassengerVehicle    rdf:type           rdfs:Class .
ex:Van                 rdf:type           rdfs:Class .
ex:Truck               rdf:type           rdfs:Class .
ex:MiniVan             rdf:type           rdfs:Class .

ex:PassengerVehicle    rdfs:subClassOf    ex:MotorVehicle .
ex:Van                 rdfs:subClassOf    ex:MotorVehicle .
ex:Truck               rdfs:subClassOf    ex:MotorVehicle .

ex:MiniVan             rdfs:subClassOf    ex:Van .
ex:MiniVan             rdfs:subClassOf    ex:PassengerVehicle .
```

Als RDF/XML:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>

<rdf:Description rdf:ID="PassengerVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>
```

Eigenschaften Definition (rdf:Property) kann folgende Attribute haben:

- rdfs:domain - bindet eine Eigenschaft an eine bestimmte Klasse
- rdfs:range – der Wert dieser Eigenschaft muss eine Instanz von einer bestimmten Klasse sein
- rdfs:subPropertyOf – Untereigenschaft; übernimmt alle Definitionen der übergeordneten Eigenschaft

Beispiel (Eigenschaften Beschreibung (Quelle W3C)):

```
ex:Person   rdf:type     rdfs:Class .
ex:author   rdf:type     rdf:Property .
ex:author   rdfs:range   ex:Person .
```

```
ex:Book     rdf:type      rdfs:Class .
ex:author   rdf:type      rdf:Property .
ex:author   rdfs:domain   ex:Book .

ex:driver          rdf:type             rdf:Property .
ex:primaryDriver   rdf:type             rdf:Property .
ex:primaryDriver   rdfs:subPropertyOf   ex:driver .
```

RDF Schema enthält auch zusätzliche Eigenschaften, die z.B. bei Erstellen der Dokumentation hilfreich sein können. Das sind unter anderen:

- rdfs:comment – erklärender, an Menschen gezielte Kommentar o. Erklärung
- rdfs:label – erklärender, an Menschen gezielte Ressourcenname
- rdfs:seeAlso – zusätzliche Informationen über Subjektressource

Ad. 5

Es existieren bereits funktionierende Systeme, die RDF Technologie anwenden. Das wohl bekannteste und populärste (weltweit 90000 sog. RSS Feeds) ist RSS (RDF Site Summary) . RSS wurde von Netscape entwickelt, um Inhalte auf seinem Portal zu sammeln und zu Synchronisieren.
Ein anderes System ist Dublin Core Metadata. DC ist das populärste Set von Elementen zur Beschreibung von Dokumenten. Es wurde im 1995 während Metadata Workshop in Dublin, Ohio entwickelt. Das Ziel von DC ist ein minimales Set von beschreibenden Elementen bereit zu stellen, die eine Beschreibung und Indexieren von Dokumenten/Objekten im Netz ermöglichen. Die Beschreibung ist einheitlich und verbindlich. Zu den Elementen von DC gehören u.a. Title, Creator, Subject, Description usw.

Xforms

Der Vortrag von Matthias Keck, der am 20.01. gehalten wurde, hatte das Thema „Xforms“.
Der Vortrag bestand aus folgenden Teilen:

1. Motivation
2. Xforms Aufbau
3. Form-Controls und Binden
4. Datentypen und Funktionen
5. Actions und Events
6. Ausblick

Ad. 1

XForms 1.0 wurde als „Working Draft“ in 2001 von W3C verabschiedet. Inzwischen wurde XForms zu W3C Empfehlung. XForms ist eine XML Sprache, die HTML Formulare in XHTML 2.0 ablösen sollte. XForms bietet eine Reihe von Mechanismen, die das Erstellen von interaktiven Webseiten und den Datenaustausch zwischen Formularen und Applikationen erleichtern. Die Erleichterungen sollten in 80% der Fälle die Notwendigkeit von Scripten unterbinden (z.B. Überprüfen der Korrektheit von Datum, Berechnen einfacher Funktionen). Die Formular Daten werden in XML gesammelt und genauso gesendet. Andere Vorteile von XForms ist besserer Umgang mit internationalen Zeichencodes und Unicode.

Ad. 2

Ein Formular in Xforms, anders als in HTML, fokussiert nicht nur auf Form-Controls (Buttons, Input-Felder usw.), sondern beinhaltet auch, in <model> Tags die Logik vom Formular. So werden Form-Controls von der Logik getrennt und erst später ans Model angebunden.

Beispiel (Quelle: Vortrag):

```
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns="http://www.w3.org/2002/xforms">
<h:head>
  <h:title>Search</h:title>
   <model>
     <submission action="http://example.com/search" method="get"
     id="s"/>
   </model>
</h:head>
<h:body>
  <h:p>
    <input ref="q"> <label>Find</label> </input>
```

```
    <submit submission="s">
      <label>Go</label>
    </submit>
  </h:p>
</h:body>
</h:html>
```

In einem Dokument kann man mehrere Modele einsetzen, die man durch ID Attribut unterscheiden kann. In einem Formular soll auch bestimmt werden, wie und wohin die Daten versendet werden. Das beschreibt *submission* Element mit mindestens 2 notwendigen Attributen, nämlich: „action“ (URI vom Ziel von Datenübertragung), und „method“ (Protokoll von Datenübertragung).

Beispiel – Model (Quelle: Vortrag)

```
<model>
<instance>
    <person>
       <fname>Ein_Vorname</fname>
       <lname/>
    </person>
</instance>
<submission id="form1" action="submit.asp" method="get"/>
</model>
```

Ad. 3

Form Control	XForms	HTML Äquivalent

Input	`<input ref="firstname">`
	`<label>First name:</label>`
	`</input>`
		`<input type="text" name="firstname">`

Textarea	`<textarea ref="message">`
	`<label>Message:</label>`
	`</textarea>`
		`Message: <textarea name="message" rows="20" cols="80"></textarea>`

Radio-Button	`<select1 ref="sex">`
	`<label>Gender:</label>`
	`<item>`
	`<label>Male</label>`
	`<value>M</value>`
	`</item>`
	`<item>`
	`<label>Female</label>`
	`<value>F</value>`
	`</item>`
	`</select1>`
		`Gender:`
		`<input type="radio" name="sex" value="M"> Male`
		`<input type="radio" name="sex" value="F">`
		`Female`

Checkbox	<select ref="flavors" appearance="full"> <label>Flavors:</label> <item> <label>Vanilla</label> <value>v</value> </item> <item> <label>Strawberry</label> <value>s</value> </item> <item> <label>Chocolate</label> <value>c</value> </item> </select>	Flavors: <input type="checkbox" name="flavors" value="v"> Vanilla <input type="checkbox" name="flavors" value="s"> Strawberry <input type="checkbox" name="flavors" value="c"> Chocolate
Menu	<select ref="spring" appearance="minimal"> <label>Month:</label> <item> <label>March</label> <value>Mar</value> </item> <item> <label>April</label> <value>Apr</value> </item> <item> <label>May</label> <value>May</value> </item> </select>	Month: <select multiple name="spring"> <option value="Mar">March</option> <option value="Apr">April</option> <option value="May">May</option> </select>
Dateiauswahl	<submission method="form-data-post" .../> ... <upload ref="attachment"> <label>File:</label> </upload>	<form method="post" enctype="multipart/form-data" ...> ... File: <input type="file" name="attachment">
Passwörter	<secret ref="pw"> <label>Password:</label> </secret>	Password: <input type="password" name="pw">
Button	<trigger ev:event="DOMActivate" ev:handler="#show"> <label>Show</label> </trigger>	<input type="button" value="Show" onclick="show()">

Um ein Formular ans Model anzubinden, gibt es zwei Verfahren:

- mit “bind” Attribut

Beispiel (Quelle: Vortrag)

In <model>:

```
<model>
  <bind nodeset="/person/name/fname" id="firstname"/>
  <bind nodeset= "/person/name/lname" id="lastname"/>
 </model>
```

In Interface:

```
<input bind="firstname">
  <label>First Name</label>
</input>
<input bind="lastname">
  <label>Last Name</label>
</input>
```

- mit “ref” Attribut

  Beispiel (Quelle: Vortrag)

  In <model>:

```
<instance>
 <person>
  <name>
   <fname/>
   <lname/>
  </name>
 </person>
</instance>
```

  In Interface:

```
<input ref="name/fname">
  <label>First Name </label>
</input>
<input ref="name/lname">
  <label>Last Name </label>
</input>
```

Ad. 4

Xforms unterstützt Datentypen von XML Schema. Beim Eintragen der Daten ins Formular wird automatisch ihre korrektheit überprüft.

Beispiel (Quelle: Vortrag):

```
<html xmlns:xf= "http://www.w3.org/2002/xforms"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<xf:instance>
<person xmlns="">
   <fname xsi:type="xsd:string"/>
   <lname xsi:type="xsd:string"/>
   <born xsi:type="xsd:date>"/>
   <size xsi:type="xsd:integer"/>
</person>
</xf:instance>
```

Xform kann auch einfache Berechnungen durchführen. Xform beinhaltet ganze Xpath Funktionen Bibliothek (Operationen mit Knoten, Strings, Zahlen und “Bool”-Werten).

Beispiel (Quelle: Vortrag):

```
<werte>
   <a>2</a>
   <b>1</b>
   <c>10</c>
   <d>5</d>
</werte>

max(werte) = 10

Oder auch:

a+b, a-b, a/b, a*b, a or b, a div b…
```

Ad. 5

Xforms unterstützt auch XML Events. Mehr in Thema XML Events.

Ad. 6

Xforms ist ein sehr praktischer Werkzueug. Bis jetzt gibt es keine weiter verbreiteten Browser die diese XML Sprache unterstützen. Während des Vortrages wurde eine Bemerkung gemacht, was die Sicherheit von den Daten angeht: den Funktionen in Xforms sollte man nicht trauen. Sie können nicht bei wichtigen Operationen (z.B. in Internetshops) die entscheidende Instanz sein.