

Laborpraktikum Software

Wintersemester 2019/20

A	0. Organisatorisches 1. Refresher Java / UML 2. Vererbung, Interfaces und Polymorphie	✓
B	3. Wert-/Verweis-Typen / -Parameter 4. Exceptions 5. Streams, Dateien, Kodierungen	✓
C	6. Objektgeflechte 7. Collections 8. Binärstreams, Serialisierung	✓
D	9. XML 10. WebServices	
E	11. Callbacks 12. Events 13. Einführung in die gängigsten grafischen Dialogelemente	

Motivation

- **MashUps:** Vernetzung von öffentlichen Informations-Ressourcen in einer eigenen Anwendung
 - Geographische Informationen
 - Wikipedia-Artikel
 - Fotos / Videos / Audio
 - Schlagzeilen (RSS)
 - Bücher
 - Wissenschaftliche Artikel
 - Ebay-Angebote
 - Werbe-Anzeigen
 - ...
- Ziel: Mehrwert für den Nutzer schaffen.

Formel

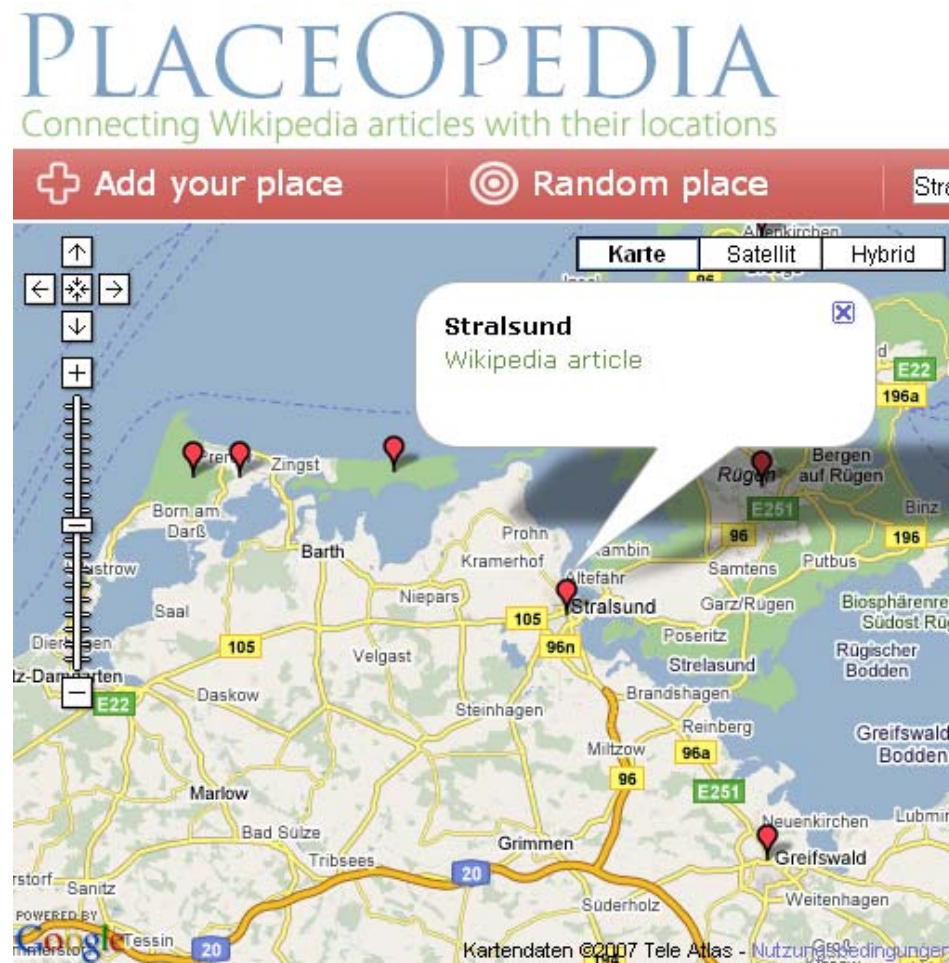
Application A +
Application B +
Application C +
.....(Mashup) =
Application X

More examples can be seen in the
following website:

<http://www.programmableweb.com/mashups>



MashUp - Beispiel



Quelle: <http://www.placeopedia.com>

MashUp - Beispiel




local.alkemis.com Combining the best of the best 'local' sites, without the ads. Click on the map to set a Temporary (Red) Pin and get more info. See a map of North America featuring [Live Traffic Cams](#) or [Google Street View](#). See [list of cities](#).



MashUp - Beispiel

Periodic Table of Heavy Metal

[Summary](#) [Comments \(0\)](#) [Track this](#)



The image shows a screenshot of a web application titled "Periodic Table of Heavy Metal". It features a periodic table layout where elements are represented by heavy metal bands. The top row includes bands like Van, Meg, T, K, and A. Below this, a large black box contains the text "(Big 4)". The main table has rows for "Black" and "Death" genres, with bands like Ba, My, Drt, R, Nd, Po, Ma, and V. To the right of the table, there is a "Description" section, "APIs" (Echo Nest + Last.fm), "Tags" (fun, music, search, steaming), and a "Mashup of the Day" badge. At the bottom left, there are five stars for voting and social media icons for RSS, Google+, and Facebook.

Description

An interactive Periodic Table of top Heavy Metal bands grouped by genre. Uses the Echonest and Last.fm APIs to display band information and images and links to watch videos, listen to songs and easily find more information on the bands.

APIs [Echo Nest](#) + [Last.fm](#)

Tags [fun](#), [music](#), [search](#), [steaming](#)

Mashup of the Day ✓

Added 26 Jun 2012

Who [psyanoid \[Profile\]](#)

URL <http://www.periodicmetal.com/>

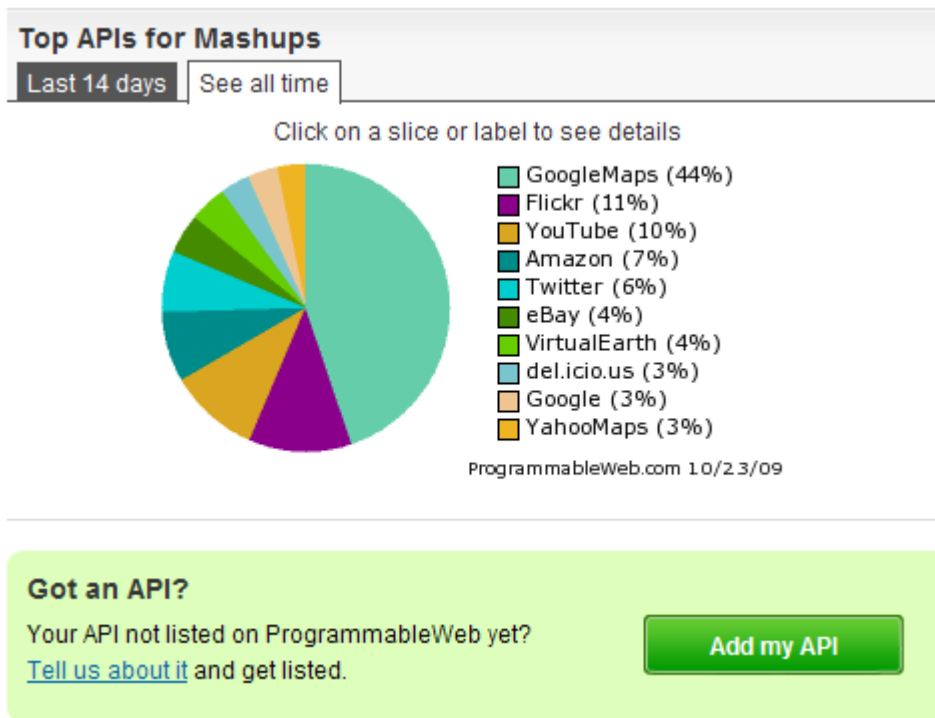
★★★★★
Click stars to vote

[RSS](#) [Google+](#) [Facebook](#) Gefällt



Quelle: <http://www.programmableweb.com/>

MashUp - Beispiel



Quelle: <http://www.programmableweb.com/apis>

MashUp – Ihre Ideen



Motivation

- Informationen im Web liegen oft in XML-Form vor.
- Wir benötigen folgendes Wissen:
 - **XML-Dokumente lesen**
 - **Webzugriff** aus dem Programm heraus

RSS – Automatisiertes „abgrasen“ von



XML



XML



XML

Newsfeeds - Opera

Gelesen L löschen Kategorie Ansicht Schnellsuche

Von	Betreff	Datum	Größe	Kategorie
Thomas	User:Thomas	Heute 09:21:35	2,0 KB	
MarkWharton	MediaWiki	Heute 08:00:11	23 KB	
Skierpage	Talk:Semantic MediaWiki development activities	Heute 07:42:11	4,2 KB	
Max Völkel	Wiki Interchange Format	Montag 21:43:...	11 KB	
Clark and Parsia	So this is cyberspace...	Montag 20:55:...	7,3 KB	
Leo Sauermann	Sauermann's home - Kanaan/Kaiserslautern/DE 20...	Montag 20:54:...	2,2 KB	
Werner Thiemann	Semweb4j/Developing	Montag 20:39:...	6,7 KB	
Leo Sauermann	typo3 board 07 - podcast	Montag 19:53:...	3,4 KB	Keine Kategorie
82.181.5.170	RandomTestPage	Montag 19:50:...	2,2 KB	
JanneJalkanen	Sandbox	Montag 19:49:...	3,1 KB	
Danny Ayers	Unanticipated use for Semantic Web technology!	Montag 18:51:...	3,1 KB	
129.13.72.145	Africa	Montag 18:34:...	6,7 KB	
129.13.72.145	Africa	Montag 18:34:...	3,6 KB	
Chrissi	Google-Orakel	Montag 18:33:...	2,3 KB	
Frederick Giasson	Revision 1.03 of the Music Ontology	Montag 18:30:...	5,7 KB	
Leo Sauermann	Sauermann's home - Kanaan/Kaiserslautern/DE 20...	Montag 18:09:...	2,2 KB	
Michael Hausenblas	Comparison of RDFa to other metadata embedding...	Montag 17:13:...	2,3 KB	
YvesPiguet	NotNew	Montag 16:42:...	2,5 KB	
YvesPiguet	ReStructuredText	Montag 16:38:...	2,0 KB	
YvesPiguet	StructuredText	Montag 16:37:...	2,0 KB	
Tinderbox Wiki	AskHere	Montag 15:38:...	1,8 KB	
Max Völkel	RDF2Go	Montag 15:38:...	23 KB	

typo3 board 07 - podcast Montag, 12. Februar 2007 19:53:00

Von Leo Sauermann

back to work, back to the web, just sitting on my couch with Rinne, a typo 3 guy who was with the T3BOARD07 - a snowboarding week of [typo3](#) geek and geekines.

watch their amazing video of a week of snow and party...

Colorful and entertaining report from the #1 TYPO3 event of the year! T3BOARD07 was held in France, Les Deux Alpes, last week in January - the 6th time! 130 participants from 10 countries in Europe. You will see snow, people, girls, football, tour de chambres and what was on the agenda for the lucky ones. And some extra gags are thrown in.



[THE VIDEO](#)

Titel, Datum, Autor, Text

Was schaffen wir in dieser Veranstaltung?

- **zum Eingewöhnen:** Nachrichten per RSS-Feed aus dem Web laden
- **zum Abschluss:** „Wikibooks“ und „Synonymwörterbuch“ vernetzen

9. XML

9.1 Einführung

- a. 10 Punkte zu XML
- b. Vorspann
- c. Elemente, Content, Attribute
- d. Kommentare
- e. Wohlgeformt und gültig
- f. Namensräume
- g. Escaping und CDATA
- h. Encoding

9.2 Dateien lesen



Quellen

- Wenn nicht gesondert angegeben:
 - Programmieren spielend gelernt mit dem Java-Hamster-Modell; D. Boles
 - „Java Developer“: <http://www.oracle.com/technetwork/java/index-jsp-135888.html>
 - www.w3c.de
 - www.w3schools.com
- „gute“ Nachschlagewerke
 - Java ist auch eine Insel von Christian Ullenboom
 - Java: How to Program; Deitel & Deitel

XML in 10 Punkten



- XML steht für **strukturierte Daten**
 - XML kann z. B. folgende Daten darstellen:
Kalkulationstabellen, Adressbücher,
Konfigurationsparameter, finanzielle Transaktionen,
technische Zeichnungen, ...
- XML = Satz an Regeln für die Erstellung von Textformaten zur Strukturierung solcher Daten.
- XML ist **keine Programmiersprache**
- XML ist **erweiterbar, plattformunabhängig** und unterstützt Internationalisierung / Lokalisierung und **Unicode**.

XML in 10 Punkten

- XML sieht ein wenig wie HTML aus
 - XML verwendet **Tags** (durch '<' und '>' geklammerte Wörter)
 - XML verwendet **Attribute** (der Form name="value").
 - HTML legt fest, was jedes Tag und Attribut bedeutet
 - XML benutzt **Tags nur zur Abgrenzung von Daten**
 - XML überlässt die **Interpretation** der Daten allein der Anwendung
 - Bedeutung von "<p>" in einer XML-Datei:
 - Paragraph?
 - Preis?
 - Person?
 - Andere (auch nicht mit P beginnend)

XML in 10 Punkten



- **XML ist Text, aber nicht zum Lesen**
 - MS Excel speichert Kalkulationstabelle in Binärformat.
 - XML: Textformat
 - Vorteil: lesbar mit Texteditor (ohne Excel).
 - Dennoch: nur eingeschränkt verwendbar.
 - Strikte Regeln bei XML: Ein weggelassenes Tag oder ein Attribut ohne Anführungszeichen, machen eine XML Datei unbenutzbar (HTML toleriert so etwas)
 - offizielle XML Spezifikation: ist die Datei fehlerhaft, muss die Anwendung an dieser Stelle anhalten und eine Fehlermeldung ausgeben.

XML in 10 Punkten

- **XML ist vom Design her ausführlich**
 - XML-Dateien sind fast immer größer als binäre Formate.
 - Abhilfe: komprimieren (z. B. HTTP/1.1)

XML in 10 Punkten



- **XML ist eine Familie von Techniken**
 - wachsender Satz an Modulen für die Verwirklichung wichtiger und häufig angefragter Aufgaben. Z. B.:
 - **XLink** (Hyperlinks in XML Dateien hinzuzufügen)
 - **XPointer** und **XFragments** (Verweise auf Teile von XML-Dokumenten).
 - **XSL** (Erstellen von Style Sheets)
 - **XSLT** (Transformationssprache für das Umstellen, Hinzufügen und Löschen von Tags und Attributen)
 - **DOM** (Funktionsaufrufe für die Manipulation von XML)
 - **XML Schema** (zur Definition eigener XML-basierter Formate)

XML in 10 Punkten



- **XML ist neu, aber nicht so neu**
 - Entwicklung begann 1996
 - Seit Februar 1998: W3C-Standard
 - XML profitiert von den Erfahrungen mit Vorläufern:
 - SGML (seit 1986 eine ISO-Norm)
 - HTML (Entwicklung ab 1990)
 - XML enthält die besten Teile von SGML, ist aber schlanker

XML in 10 Punkten



- **XML überführt HTML in XHTML**
 - W3C's XHTML ist Nachfolger von HTML.
 - XHTML hat mit HTML viele gleiche Elemente.
 - Syntax ein wenig geändert, um mit den XML Regeln konform zu sein.

XML in 10 Punkten



- **XML ist modular**
 - Möglich: Neues Dokumentenformat definieren, indem man andere Formate kombiniert oder wiederbenutzt.
 - Problem: Namenskollisionen: `<p>=Absatz?` Oder `<p>=Preis?`
 - **Namensraummechanismus.**
 - Beispiele: XSL und RDF sind XML-basierte Formate, die Namensräume benutzen.

XML in 10 Punkten



- XML ist die **Basis für RDF und das Semantic Web**
 - RDF = Resource Description Framework
XML Textformat zur Beschreibung von Ressourcen
 - z. B. Fotoalbum im Web,
 - Jedes Bild mit computer-lesbaren Eigenschaften wie Fotograf und Ort
 - Beispielanwendung:
 - Verknüpfung einer eigenen Kontaktliste mit dem Fotoalbum.
 - Aktion: E-Mail an alle Fotografen im Album, die ich kenne.

XML in 10 Punkten

- XML ist **lizenzfrei, plattformunabhängig** und **gut unterstützt**
 - Große Auswahl an Werkzeugen
 - Lizenzfrei
 - Nicht an einen einzigen Anbieter gebunden
 - XML ist nicht immer die beste Lösung, aber es lohnt sich immer, XML in Erwägung zu ziehen.

Vorspann

Erste Zeile:

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



Elemente und Content

- Immer **genau ein Root-Element**
- Element umfasst alles vom Start- bis einschl. Ende-Tag
- Elemente besitzen **Content**:
 - „Element content“
 - „Mixed content“
 - „Simple content“
 - „Empty content“



Elemente und Content - Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>
    Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```

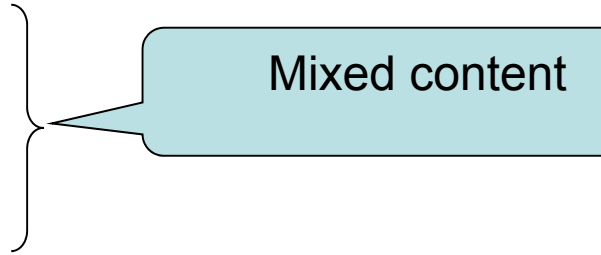

Elemente und Content - Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>
    Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```

Element content
„book“ enthält Elemente
„title“, „prod“, ...

Elemente und Content - Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>
    Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```



Elemente und Content - Beispiel:

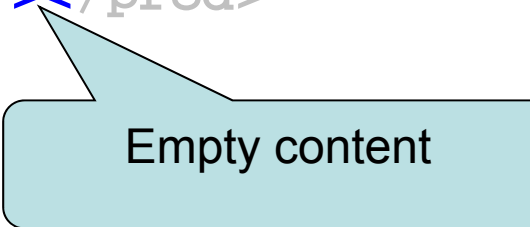
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```



Simple content

Elemente und Content - Beispiel:

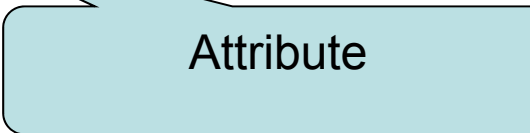
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```



Empty content

Attribute – Beispiel:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<book>
  <title>My First XML</title>
  <prod id="33-657" media="paper"></prod>
  <chapter>Introduction to XML
    <para>What is HTML</para>
    <para>What is XML</para>
  </chapter>
  <chapter>XML Syntax
    <para>Elements must have a closing tag</para>
    <para>Elements must be properly nested</para>
  </chapter>
</book>
```



Attribute

Attribute und Content

- Wann Attribute? Wann Content?
- Es gibt keine Vorschrift.
- Erfahrung: Attribute möglichst vermeiden
 - Kind-Elemente, wenn die Information sich irgendwie nach Daten anfühlt.
 - Attribute führen zu schwer(er) les- und wartbaren Dokumenten.

Probleme mit Attributen

- Attribute können **keine mehrfachen Werte** enthalten (Kind-Elemente hingegen schon)
- Attribute können **nicht leicht** für zukünftige Ergänzungen **erweitert** werden
- Attribute können **keine Strukturen** beschreiben (Kind-Elemente hingegen schon)
- Attribute sind durch Programmcode **schwerer zu handhaben**
- Attribute lassen sich nicht leicht gegen eine DTD testen

So nicht !

```
<note day="12" month="11" year="2006"  
to="Hugo" from="Egon" heading="Erinnerung"  
body="Vergiss unser Treffen nicht!"></note>
```

Kommentare

```
<!-- Dies ist ein Kommentar -->
```

Wohlgeformt / Gültig

- Ein **wohlgeformtes** XML-Dokument ist **syntaktisch korrekt**
 - Z. B. fehlendes Ende-Tag: nicht wohlgeformt.
 - Prüfung der Wohlgeformtheit: mit Firefox oder Internet-Explorer überprüfen.
- Wohlgeformt ist die **Minimalanforderung**.



Wohlgeformt / Gültig

- Beispiel:

```
<address>  
  <street>Zur Schwedenschanze 15</stret>  
  <plz>18435</plz>  
  <ort>Stralsund</ort>  
</address>
```

XML-Verarbeitungsfehler: Nicht übereinstimmendes Tag. Erwartet: </street>.
Adresse: file:///C:/address.xml
Zeile Nr. 2, Spalte 35:

```
<street>Zur Schwedenschanze 15</stret>  
-----^
```

Wohlgeformt / Gültig

- Wohlgeformt ist die Minimalanforderung.
- **Wohlgeformt \neq Gültig**
- Gültig, wenn es gewissen **Regeln** entspricht, die **für einen Anwendungsbereich festgelegt** wurden.
 - Z. B. eine Notiz muss immer die Information enthalten, für wen die Notiz bestimmt ist.
- Derartige Regeln in Form einer **DTD** oder einem **XML-Schema** festlegen.

Gültigkeit eines XML-Dokumentes

- DTD = document type definition
- Beispiel:

```
<!ELEMENT note (to,from,heading,body)>  
<!ELEMENT to      (#PCDATA)>  
<!ELEMENT from      (#PCDATA)>  
<!ELEMENT heading  (#PCDATA)>  
<!ELEMENT body      (#PCDATA)>
```

- PCDATA = parsed character data
 - Text zwischen Start- und Ende-Tag
 - sog. Entities wie „<“ werden geparsed und aufgelöst („<“).

Gültiges XML-Dokument

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
  <to>Hugo</to>
  <from>Egon</from>
  <heading>Erinnerung</heading>
  <body>Vergiss unser Treffen nicht!</body>
</note>
```


Gültigkeitsregeln

- Gültigkeitsregeln:
 - **DTD**
 - **XML-Schema** (genauere Festlegung möglich, aber komplexer)

Namensräume

- Namenskonflikte bei Elementnamen
- Dokument 1 (Tabelle):

```
<table>  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>
```

Apples	Bananas
--------	---------

- Dokument 2 (Möbel):

```
<table>  
  <name>African Coffee Table</name>  
  <width>80</width>  
  <length>120</length>  
</table>
```



Namenskonflikte

- Beispielanwendung: Shop, in dem man Bananen, Äpfel und afrikanische Tische kaufen kann.



Präfix

- Dokument 1 (Tabelle): Präfix „h“

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>
```

Präfix beliebig. Wir nehmen h als Markierung für html-Tags.

- Dokument 2 (Möbel): Präfix „f“ (furniture)

```
<f:table  
  xmlns:f="http://www.w3schools.com/furniture">  
  <f:name>African Coffee Table</f:name>  
  <f:width>80</f:width>  
  <f:length>120</f:length>  
</f:table>
```

Präfix

- `xmlns`-Attribut
 - **Nur im Start-Tag**
 - Bindet Präfix an URI
 - **URI** = Zeichenkette zur Identifizierung einer abstrakten oder physikalischen Ressource.
 - kann, muss aber keine gültige Web-Adresse sein
 - „blabla“ wäre auch in Ordnung, so lange es sich um einen eindeutigen Bezeichner für diesen Namensraum handelt.
 - Syntax:
`xmlns:Namensraumpräfix="NamensraumURI"`

Homogene Bereiche

- Präfix kann entfallen:

```
<table xmlns="http://www.w3.org/TR/html4/">  
  <tr>  
    <td>Apples</td>  
    <td>Bananas</td>  
  </tr>  
</table>
```

- Im Gegensatz zu

```
<h:table  
  xmlns:h="http://www.w3.org/TR/html4/">  
  <h:tr>  
    <h:td>Apples</h:td>  
    <h:td>Bananas</h:td>  
  </h:tr>  
</h:table>
```

Problem

- Beispiel

```
<message>if salary < 1000 then</message>
```

XML-Verarbeitungsfehler: nicht wohlgeformt
Adresse: file:///C:/message.xml
Zeile Nr. 1, Spalte 21:

```
<message>if salary < 1000 then</message>  
-----^
```

Lösung: Entities

- Beispiel

```
<message>if salary &lt; 1000 then</message>
```

- Entities in XML:

- **<** <
- **>** >
- **&** &
- **'** '
- **"** "

Andere Lösung: CDATA

- Beispiel

```
<script>  
  <![CDATA[  
    function matchwo(a,b)  
    {  
      if (a < b && a < 0) then  
      {  
        return 1  
      }  
      else  
      {  
        return 0  
      }  
    }  
  ]]  
</script>
```

Wird nicht geparsed

Encoding

- Aufgabe: XML-Dateien soll fremdländisch aussehende Zeichen enthalten (z. B: œ oder ě oder noch fremder).
- Parser muss diese Zeichen verstehen
- Lösung: Dokument im Unicode-Standard speichern
- Codierung im Vorspann angeben

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



Encoding

- **Falle:**
 - XML-Datei mit einem Editor z. B. als Unicode-16bit abspeichern
 - XML-Vorspann mit anderer Kodierung (z. B. `<?xml version="1.0" encoding="UTF-8"?>`)
- **Deshalb:**
 - Editor benutzen, der Kodierung unterstützt
 - **Wissen, welche Kodierung der Editor verwendet**
 - Diese Kodierung im Vorspann angeben
- **In dieser Veranstaltung:**
 - ECLIPSE (Eclipse > Einstellungen (Preferences) > General > Content/Types ...)



Übung 9i

- Schreiben Sie ein XML-Dokument auf Papier, das die folgenden Informationen enthält:
 - Ihren Namen,
 - Studienbeginn und
 - Studiengang.
- Ergänzen Sie ein Attribut in einem geeigneten Element mit Ihrer Matrikelnummer. Wählen Sie einen geeigneten Namen für das Attribut.

Übung 9i (Lösung)

```
<?xml version="1.0" encoding="UTF-8"?>
  <Studierende>
    <Person>
      <Name>Peter Ludolf</Name>
      <Semester>1997</Semester>
      <Studiengang>Angewandtes Recycling</Studiengang>
    </Person>
    ...
  </Studierende>

  ....
  <Person Matrikel="123456789">
    <Name>Peter Ludolf</Name>
    <Semester>1997</Semester>
    <Studiengang>Angewandtes Recycling</Studiengang>
  </Person>
  ...
```

Beispiel

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<auftrag id="PO1456">
```

```
  <datum>
```

```
    <!-- Auftragsdatum -->
```

```
    <jahr>2006</jahr>
```

```
    <monat>6</monat>
```

```
    <tag>14</tag>
```

```
  </datum>
```

```
  <adresse typ="liefer">
```

```
    <name>Fritz Mendels</name>
```

```
    <strasse>Badstr. 2</strasse>
```

```
    <plz>12345</plz>
```

```
    <ort>Musterstadt</ort>
```

```
  </adresse>
```

```
  <adresse typ="rechnung">
```

```
    <name>Fritz Mendels</name>
```

```
    <strasse>Ringstr. 1</strasse>
```

```
    <plz>12346</plz>
```

```
    <ort>Musterdorf</ort>
```

```
  </adresse>
```

```
  <items>
```

```
    <item>
```

```
      <menge>1</menge>
```

```
      <artnr>R-173</artnr>
```

```
      <beschreibung>
```

```
        14,4 Volt Akkuschrauber
```

```
      </beschreibung>
```

```
      <preis>39,95</preis>
```

```
    </item>
```

```
    <item>
```

```
      <menge>1</menge>
```

```
      <artnr>1632S</artnr>
```

```
      <beschreibung>Bit-Set</beschreibung>
```

```
      <preis>9,95</preis>
```

```
    </item>
```

```
  </items>
```

```
</auftrag>
```

Aufgabe

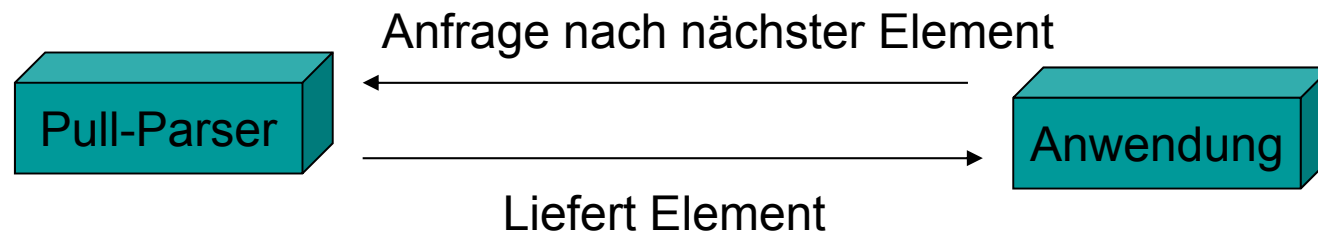
- Aufgabe: Einlesen und Inhalt ausgeben.
- „Zu Fuß“ mühsam.
- Lösung: **XML-Parser**

XML-Parser

- Varianten:
 - **Push-Parser**
 - Anwendungscode registriert Callback-Funktionen
 - Parser geht selbständig das XML-Dokument durch
 - Parser ruft bei gefundenen Elementen die registrierten Callbacks auf.
 - Kontrollfluss in der Verantwortung des Parsers
 - Unflexibel
 - Beispiel: SAX (simple API für XML).

Pull-Parser

- Applikation kontrolliert den Parse-Vorgang
- Abruf einzelner Dokumentelemente durch die Applikation

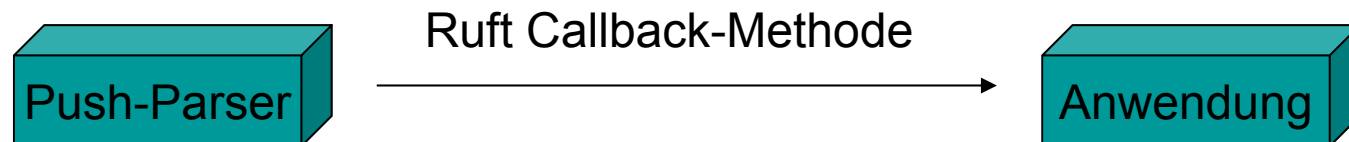


XML-Parser

- Varianten:
 - **Pull-Parser**
 - Anwendungscode holt sich selbst das nächste Ereignis durch Aufruf einer Read-Methode.
 - Beispiel: Java *PullParser*

Push-Parser

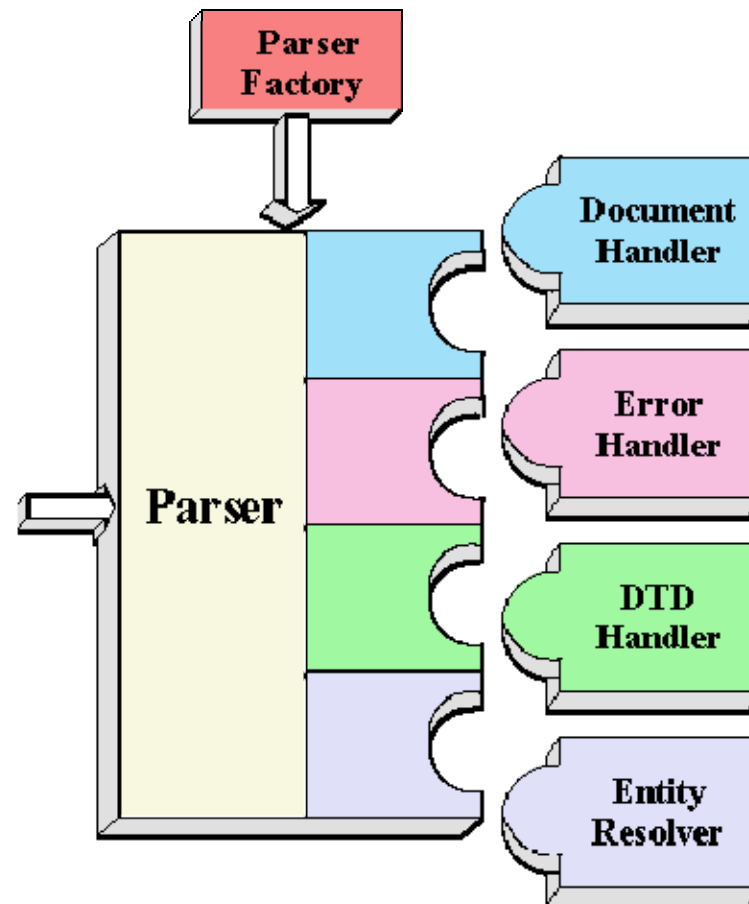
- Parser kontrolliert den Parse-Vorgang
- Parser benachrichtigt die Applikation mittels Ereignissen
- Ereignisse sind das Auftreten von Informationseinheiten im Dokument



XML-Parser

- Varianten:
 - **Forward-Only**
 - Dokument kann nur vorwärts gelesen werden.
 - reicht in vielen Fällen aus und ist Speicherplatz sparend
 - Beispiel: Java `SAXParser` (`Simple API for XML`)
 - **Wahlfreier Zugriff**
 - Z. B. DOM (document object model)
 - Dokument komplett in Hauptspeicher laden.


Java-API: SAX



Quelle: <http://www.fh-wedel.de>

SAX

SAX-API definiert folgende Handler:

- **DocumentHandler** 
Erlaubt das Lesen eines Dokuments, die Verarbeitung von Knoten und die Positionierung.
- **ErrorHandler:**
Fehlerbehandlung.
- **DTDHandler:** ,
Behandlung von nicht „parsbaren“ Elementen
z.B. Binärdaten.
- **EntityResolver:**
Referenzen auf externen Dateien.



Beispieldatei

```
<?xml version="1.0"?>
<personen>
  <person id="1">
    <name>Mustermann</name>
    <vorname>Max</vorname>
    <geburtsdatum>25.11.1983</geburtsdatum>
    <postleitzahl>54321</postleitzahl>
    <ort>Bierdorf</ort>
  </person>
  <person id="2">
    <name>Müller</name>
    <vorname>Petra</vorname>
    <geburtsdatum>13.04.1990</geburtsdatum>
    <postleitzahl>76543</postleitzahl>
    <ort>Bierdorf</ort>
  </person>
</personen>
```

Quelle: <http://blog.mynotiz.de/htmlQuellcodes/personen.xml.html>

Lesen mit SAX

- Aufgabe: Struktur verstehen.
Deshalb: alles parsen und ausgeben.

Öffnen

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
import org.xml.sax.helpers.XMLReaderFactory;
...

// XMLReader erzeugen
XMLReader xmlReader = XMLReaderFactory.createXMLReader();

// Pfad zur XML Datei
FileReader reader = new FileReader("X:\\personen.xml");
InputSource inputSource = new InputSource(reader);

// PersonenContentHandler wird übergeben
xmlReader.setContentHandler(new PersonenContentHandler());

// Parsen wird gestartet
xmlReader.parse(inputSource);
```

Quelle: <http://blog.mynotiz.de/htmlQuellcodes/personen.xml.html>

Lesen mit dem `XmlTextReader`

- Eine XML-Datei besteht aus „Knoten“ (nodes)
 - Start-Element
 - Kommentare
 - Text-Content
 - CDATA-Content
 - End-Tag
 - ...
 - sogar Whitespaces

Struktur eines XML-Leseprogramms

- Durchlaufen des Dokuments und Lesen der Informationen übernimmt der XML Reader
- Unsere Aufgabe:
Handler für Datenübernahme (Beim Parsen der XML-Datei werden die jeweiligen ContentHandler Funktionen aufgerufen).

Beispielprogramm 1

```
package parser;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
import org.xml.sax.Locator;
import org.xml.sax.SAXException;

public class PersonenContentHandler implements ContentHandler {

    private ArrayList<Person> allePersonen = new ArrayList<Person>();
    private String currentValue;
    private Person person;

    // Aktuelle Zeichen die gelesen werden, werden in eine Zwischenvariable
    // gespeichert
    public void characters(char[] ch, int start, int length)
        throws SAXException {
        currentValue = new String(ch, start, length);
    }
}
```

... → Fortsetzung folgt

Quelle: <http://blog.mynotiz.de/programmieren/java-sax-parser-tutorial-773/>

Beispielprogramm 1

```
// Methode wird aufgerufen wenn der Parser zu einem Start-Tag kommt
public void startElement(String uri, String localName, String qName,
    Attributes atts) throws SAXException {
    if (localName.equals("person")) {
        // Neue Person erzeugen
        person = new Person();

        // Attribut id wird in einen Integer umgewandelt und dann zu der
        // jeweiligen Person gesetzt
        person.setId(Integer.parseInt(atts.getValue("id")));
    }
}

// Methode wird aufgerufen wenn der Parser zu einem End-Tag kommt
public void endElement(String uri, String localName, String qName)
    throws SAXException {

    // Name setzen
    if (localName.equals("name")) {
        person.setName(currentValue);
    }

    // Vorname setzen
    if (localName.equals("vorname")) {
        person.setVorname(currentValue);
    }
}
```

... → Fortsetzung folgt

Quelle: <http://blog.mynotiz.de/programmieren/java-sax-parser-tutorial-773/>

Beispielprogramm 1

```
// Datum parsen und setzen
if (localName.equals("geburtsdatum")) {
    SimpleDateFormat datumsformat = new SimpleDateFormat("dd.MM.yyyy");
    try {
        Date date = datumsformat.parse(currentValue);
        person.setGeburtsdatum(date);
    } catch (ParseException e) {
        e.printStackTrace();
    }
}

// Postleitzahl setzen
if (localName.equals("postleitzahl")) {
    person.setPostleitzahl(currentValue);
}

// Ort setzen
if (localName.equals("ort")) {
    person.setOrt(currentValue);
}

// Person in Personenliste abspeichern falls Person End-Tag erreicht
// wurde.
if (localName.equals("person")) {
    allePersonen.add(person);
    System.out.println(person);
}
}
```

... → Fortsetzung folgt

Quelle: <http://blog.mynotiz.de/programmieren/java-sax-parser-tutorial-773/>

Beispielprogramm 1

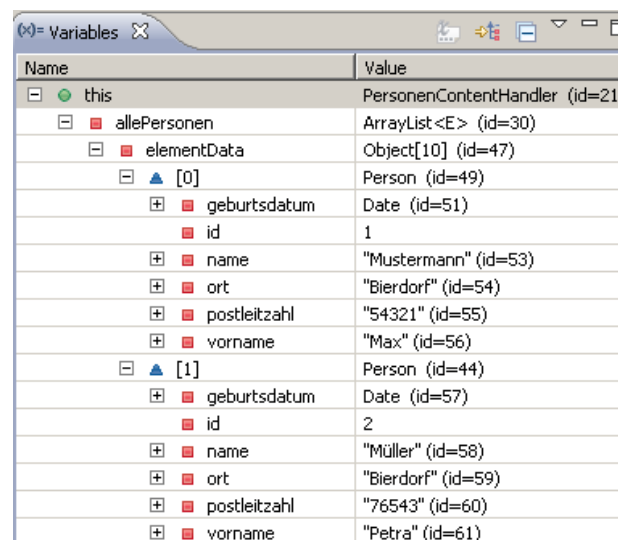
```
public void endDocument() throws SAXException {}  
public void endPrefixMapping(String prefix) throws SAXException {}  
public void ignorableWhitespace(char[] ch, int start, int length)  
    throws SAXException {}  
public void processingInstruction(String target, String data)  
    throws SAXException {}  
public void setDocumentLocator(Locator locator) { }  
public void skippedEntity(String name) throws SAXException {}  
public void startDocument() throws SAXException {}  
public void startPrefixMapping(String prefix, String uri)  
    throws SAXException {}  
}
```

Quelle: <http://blog.mynotiz.de/programmieren/java-sax-parser-tutorial-773/>

Beispielprogramm 1 - Ausgabe

- Nach dem durchlaufen des SAX Parsers befinden sich nun alle Personen in der **ArrayList allePersonen**.

```
[[1] [Mustermann] [Max] [Bierdorf] [54321] [Fri Nov 25 00:00:00 CET 1983 ]]  
[[2] [Müller] [Petra] [Bierdorf] [76543] [Fri Apr 13 00:00:00 CEST 1990 ]]
```



Name	Value
this	PersonenContentHandler (id=21)
allePersonen	ArrayList<E> (id=30)
elementData	Object[10] (id=47)
[0]	Person (id=49)
geburtsdatum	Date (id=51)
id	1
name	"Mustermann" (id=53)
ort	"Bierdorf" (id=54)
postleitzahl	"54321" (id=55)
vorname	"Max" (id=56)
[1]	Person (id=44)
geburtsdatum	Date (id=57)
id	2
name	"Müller" (id=58)
ort	"Bierdorf" (id=59)
postleitzahl	"76543" (id=60)
vorname	"Petra" (id=61)

Quelle: <http://blog.mynotiz.de/programmieren/java-sax-parser-tutorial-773/>

Wdhlg. Streams

- Java bietet eine umfangreiche Bibliothek
 - zum sequentiellen Zugriff auf Dateien,
 - für wahlfreie Dateioperationen und
 - zur Verwaltung von Verzeichnissen.
- Alle Klassen zur (Datei-)Ein- und Ausgabe befinden sich im Paket java.io
 - `import java.io.*;`

•

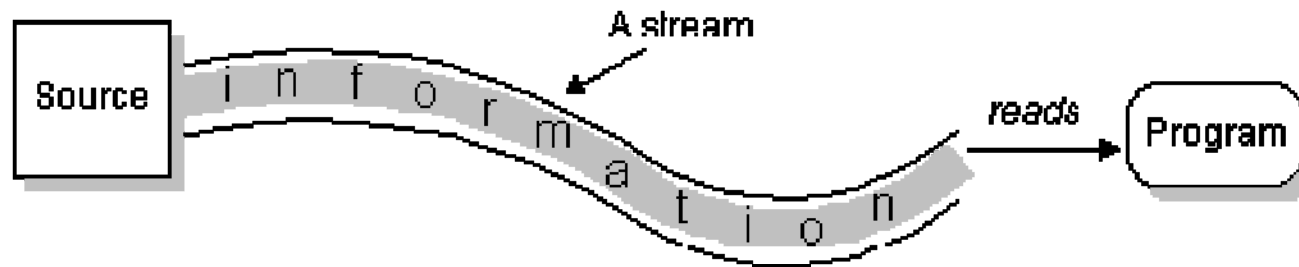
Sequentieller Zugriff auf Dateien erfolgt über das Konzept der Streams.

- Streams:
- **... abstraktes Konstrukt, dessen Fähigkeit darin besteht, Zeichen auf ein imaginäres Ausgabegerät zu schreiben oder von diesem zu lesen. Erst konkrete Unterklassen binden die Zugriffsroutinen an echte Ein- und Ausgabegeräte (Dateien, Strings oder Netzwerk-Kommunikationskanäle).**

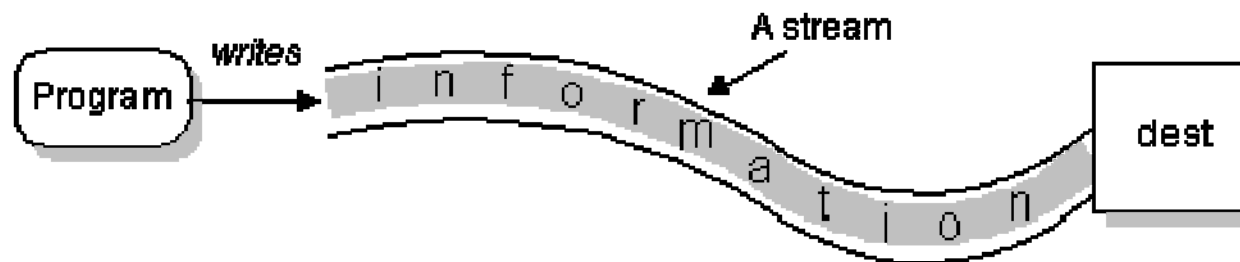
Quelle: www.iks.hs-merseburg.de/~uschroet/java_neu/ppt/j_10_einf.ppt

Wdhlg. Streams

Eingabestream



Ausgabestream



Quelle: www.iks.hs-merseburg.de/~uschroet/java_neu/ppt/j_10_einf.ppt

Zugriff auf Webressourcen

Mittels Stream

```
import java.net.*;
import java.io.*;

public void openURLConnection()
{
    try
    {
        URL url = new URL("http://www.mycompany.com");
        URLConnection connection = url.openConnection();
    }

    catch (Exception e)
    {
        System.out.println(e.toString());
    }
}
```

Quelle: <http://www.techrepublic.com/article/using-java-to-access-web-resources/>

Zugriff auf Webressourcen

Lesen

```
import java.net.*;  
import java.io.*;
```

```
public void readFromURL()  
{  
    try  
    {  
        URL url = new URL("http://www.mycompany.com");  
        URLConnection connection = url.openConnection();  
        connection.setDoInput(true);  
        InputStream inStream = connection.getInputStream();  
        BufferedReader input = new BufferedReader(new InputStreamReader(inStream));  
  
        String line = "";  
        while ((line = input.readLine()) != null)  
            System.out.println(line);  
    }  
    catch (Exception e)  
    {  
        System.out.println(e.toString());  
    }  
}
```

Quelle: <http://www.techrepublic.com/article/using-java-to-access-web-resources/>

Zugriff auf Webressourcen

Schreiben

```
import java.net.*;
import java.io.*;

public void writeToURL()
{
    try
    {
        URL url = new URL("http://www.mycompany.com");
        URLConnection connection = url.openConnection();
        connection.setDoOutput(true);
        OutputStream outputStream = connection.getOutputStream();
        ObjectOutputStream objectStream = new ObjectOutputStream(outputStream);
        objectStream.writeInt(54367);
        objectStream.writeObject("Hello there");
        objectStream.writeObject(new Date());
        objectStream.flush();
    }

    catch (Exception e)
    {
        System.out.println(e.toString());
    }
}
```

Quelle: <http://www.techrepublic.com/article/using-java-to-access-web-resources/>

Fehlerursachen

- Falsche **Protokollversion**
 - der Request erfordert http 1.1, der Server versteht aber nur 1.0
- **Kein UserAgent** gesetzt (User Agent ist der anfragende Client).
- Der Server möchte automatisch **weiterleiten**, der Request verbietet dies jedoch
- Die **Anfrage-Methode** stimmt nicht (GET / POST)
- **Timeout** ist zu klein
- Proxy falsch gesetzt

Einschub: Setzen des User-Agent

```
import java.io.IOException;
import java.net.URL;
import java.net.URLConnection;

public class TestUrlOpener {

    public static void main(String[] args) throws IOException {
        URL url = new URL("http://localhost:8080/foobar");
        URLConnection hc = url.openConnection();
        hc.setRequestProperty("User-Agent", "Mozilla/5.0 (Macintosh; U;
Intel Mac OS X 10.4; en-US; rv:1.9.2.2) Gecko/20100316 Firefox/3.6.2");

        System.out.println(hc.getContentType());
    }
}
```

Quelle: <http://stackoverflow.com/questions/2529682/setting-user-agent-of-a-java-urlconnection>

Laborpraktikum Software

Wintersemester 2017/18

A	0. Organisatorisches 1. Refresher C# / UML 2. Vererbung, Interfaces und Polymorphie	✓
B	3. Wert-/Verweis-Typen / -Parameter 4. Exceptions 5. Streams, Dateien, Kodierungen	✓
C	6. Objektgeflechte 7. Collections 8. Binärstreams, Serialisierung	✓
D	9. XML 10. WebRequest / -Response	✓
E	11. Callbacks 12. Events 13. Einführung in die gängigsten grafischen Dialogelemente	