



Service Engineering

Technische Aspekte für (Web-) Services

Die Inhalte der Vorlesung wurden primär auf Basis der angegebenen Literatur erstellt. Darüber hinaus finden sich vielfältige Beispiele aus dem industriellen Umfeld.



Agenda



- Web APIs - Webbasierte Serviceangebote
- HTTP als zustandsloses Basisprotokoll
- XML – e**X**tensible **M**arkup **L**anguage
- JSON – JavaScript Object Notation
- XML/JSON – Werkzeugunterstützung



Service- und weborientierte Architekturen



Web Services – allgemeine Sicht

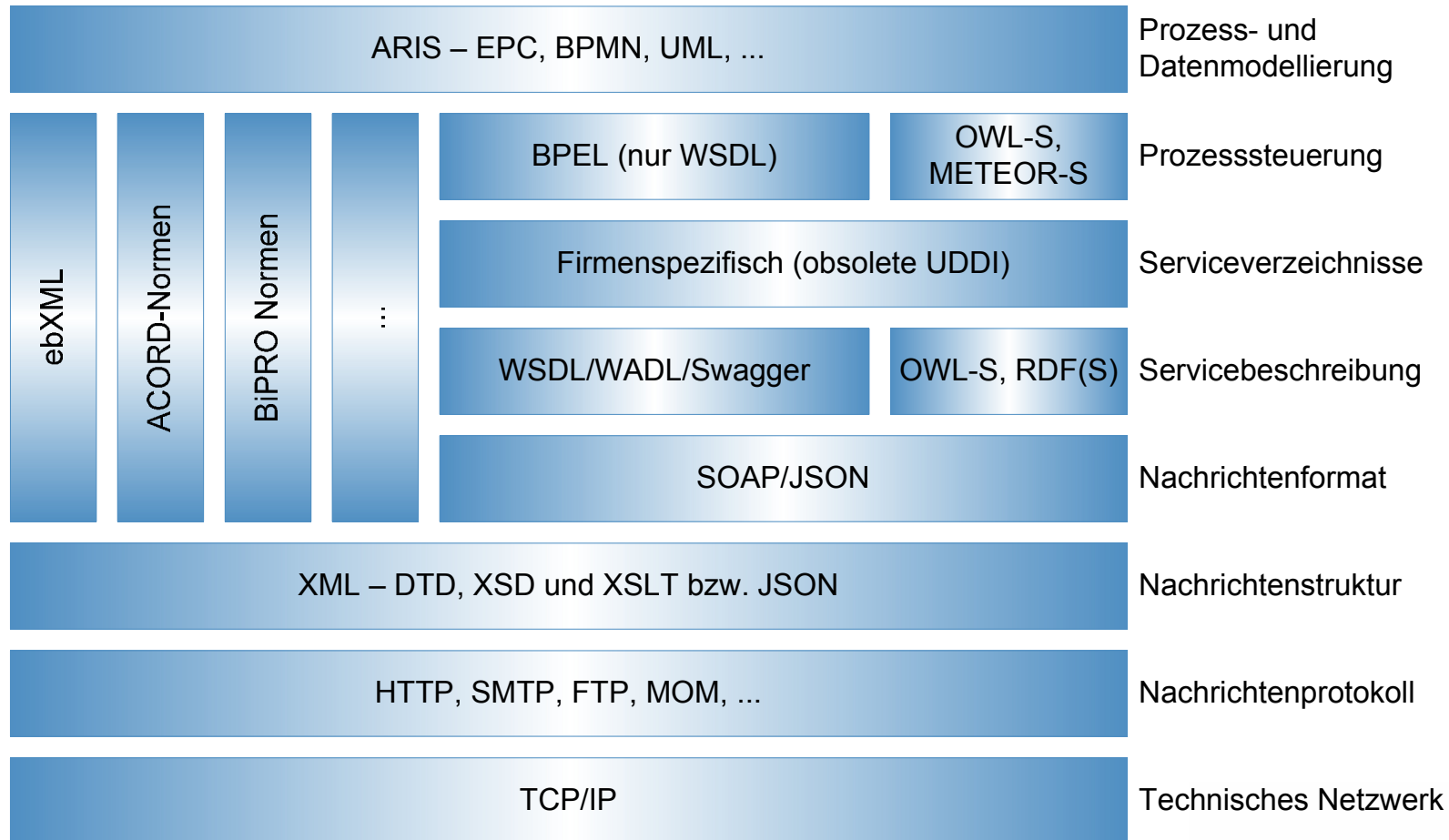


A web service is a piece of business logic, located somewhere on the Internet, that is accessible through standard-based Internet protocols such as HTTP or SMTP. Using a web service could be as simple as logging into a site or as complex as facilitating a multi-organization business negotiation.

Quelle: Chappell, A.; Jewell, T.: Java Web Services, O'Reilly-Verlag



Technologiestack





HTTP - HyperText Transfer Protocol



Standardverben von HTTP

- GET – über eine URL/URI adressierte Ressource laden
- HEAD – Metadaten über Ressourcen laden
- PUT – bestehende Ressource aktualisieren
- POST – Anlegen einer neuen Ressource
- DELETE – Löschen einer Ressource
- OPTIONS – Metadaten (u.a. unterstützte Methoden)
- TRACE – Diagnose von HTTP-Verbindungen
- CONNECT – SSL E2E-Verbindung via proxy



Unified Resource Identifier

Ziel: weltweit eindeutiger Namensraum für Web-Ressourcen

- PROTOKOLL://SERVER/VERZEICHNISPFAD/Ressource

- Protokoll: http(s) (file, ftp, mailto, ftp, https, ...)

- Server:

 - IP-Adresse z.B. 164.19.200.20

 - alternativ Domain Name Service (DNS-Name)

- Verzeichnispfad: Verzeichnis innerhalb des lokalen Filesystems

- Jede Ressource unterstützt die gleiche Schnittstelle

```
// Java Pseudosyntax
// HTTP-Schnittstelle je Ressource
public interface Resource {
    Resource (URI u);
    Response options();
    Response get();
    Response post (Request r);
    Response put (Request r);
    Response delete();
}
```

Pseudosyntax in Anlehnung an: Tilkov, S. et al.: REST und HTTP, dpunkt.verlag, Heidelberg 2015



REST-Stil

REpresentational State Transfer - REST

- Architekturstil REST – abstrakte Architektur des Web
- Orientiert sich an den Kernprinzipien des HTTP-Protokolls
 - Zustandslose Client/Service-Kommunikation
 - Identifizierbare Ressourcen – weltweit eindeutig!
 - Prinzip der Ressourcenrepräsentation (MIME types)
 - Gleichförmige/uniforme Schnittstelle – und daher universell einsetzbar
 - „Hypermedia as the engine of application state“
- REST ist kein Produkt und kein Standard!

Quelle: Fielding, R. T.: Architectural Styles and the Design of Network-based Software Architectures, UNIVERSITY OF CALIFORNIA, IRVINE, 2000

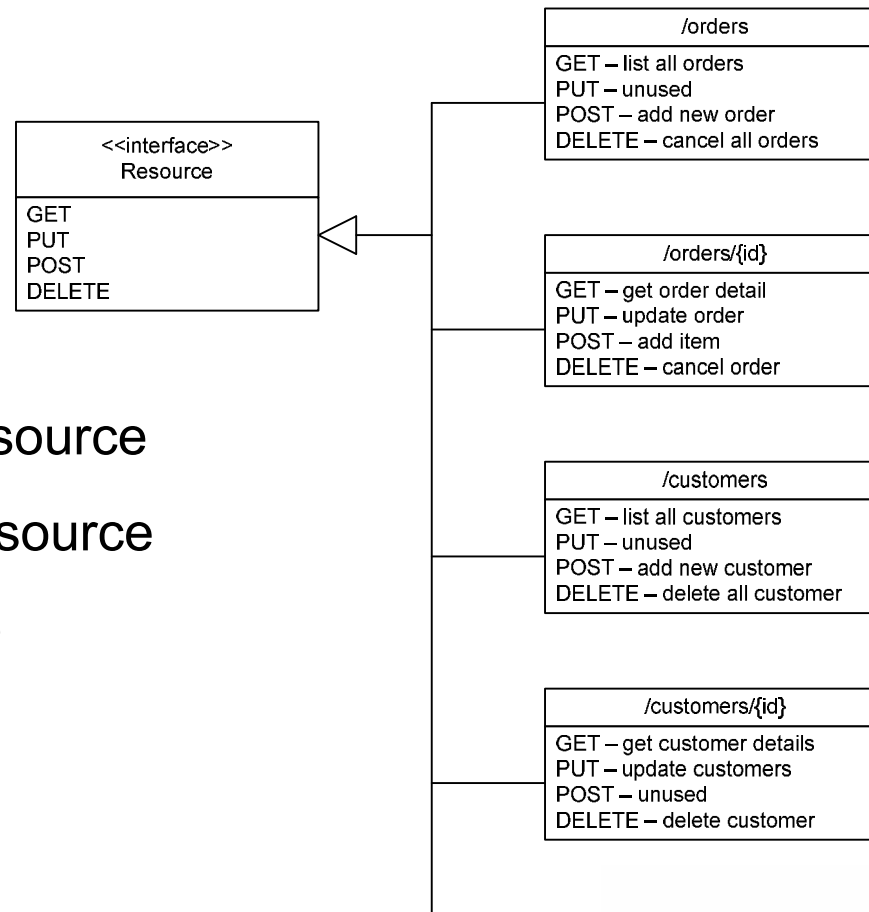


REST-Stil

Verwendung uniformer SSt.:

bei Verwendung von HTTP:

- GET – Lese eine Ressource
- PUT – Verändern einer Ressource
- DELETE – Lösche eine Ressource
- POST – Erzeuge Ressource



Quelle: Tilkov, S.: REST – eine Einführung, in Starke, G.; Tilkov, S.: SOA-Expertenwissen – Methoden, Konzepte und Praxis serviceorientierter Architekturen, dpunkt.verlag, Heidelberg 2007



XML - eXtensible Markup Language



eXtensible Markup Language



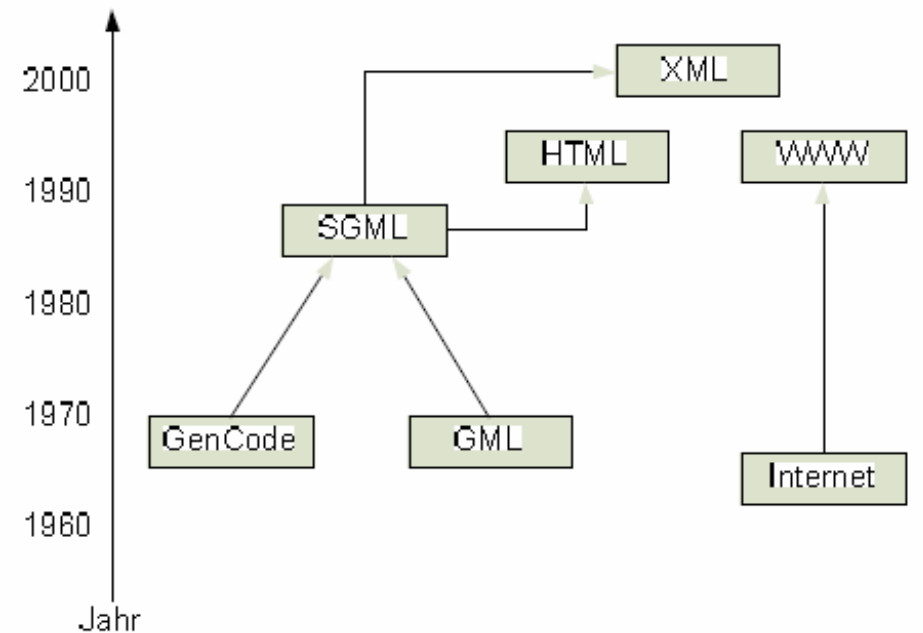
- *GML – Generalized Markup*

Language (Syntax)

- *GenCode (Semantik)*

- *SGML – Standard Generalized Markup Language – 500 Seiten*

- *HTML – Hyper Text Markup Language*





eXtensible Markup Language



- eXtensible Markup Language (XML)
- „Esperanto“ der IT Welt
- Auszeichnung und Strukturierung von Daten mittels öffnender und schließender Tags
- XML stellt unter anderem die Basis für SOAP und WSDL sowie die WS-* Technologien dar

```
<?xml version="1.0" ?>
<brief>
  <adresse typ = "an">
    <name>Harry Mustermann</name>
    <strasse>Musterstrasse</strasse>
    <nummer>1b</nummer>
    <postleitzahl>12345</postleitzahl>
    <ort>Musterhausen</ort>
  </adresse>
  <adresse typ = "von">
    <name>Klaus Gustav</name>
    <strasse>Dorfstrasse</strasse>
    <nummer>33</nummer>
    <postleitzahl>54321</postleitzahl>
    <ort>Michelbinge</ort>
  </adresse>
  <briefftext>
    Lieber Harry, wie geht es dir ... etc.
  </briefftext>
</brief>
```



eXtensible Markup Language



- XML ist Teilmenge von SGML (Standard Generalized Markup Language)
 - HTML kann mittels XML definiert werden
 - DTD für HTML – www.w3c.org/TR/REC-html40/strict.dtd
- Trennung von Struktur und Inhalt
- Möglichkeit selbstbeschreibender Dokumentenstrukturen
- Möglichkeit einer Validierung von XML-Dokumenten
- Leichte Erweiterbarkeit durch die Definition neuer Tags
- Geschachtelte Tags zur Beschreibung strukturierter Daten



eXtensible Markup Language



Wohlgeformtes XML:

- XML Dokumente bestehen aus Prolog und Elementen
- Im Prolog steht die XML-Declaration (XML-Version)
- Jedes öffnende Tag muss explizit geschlossen werden
 - z.B. Zeilenvorschub: `
</BR>` oder `
`
- Attribute müssen in Anführungszeichen gesetzt werden
- „<“ (<) und „&“ (&) dürfen im Text nicht vorkommen
- Standardattribute müssen vom Typ CDATA (Character Data) sein



eXtensible Markup Language

- DTD Document Type Definition - definiert die Grammatik der Dokumente

- Element Declaration

```
<!ELEMENT Elementname (Inhaltsbeschreibung)>
```

- Case sensitiv
- ? (optionales Element), + (mind. einmal), * (beliebig häufig)
- Elemente müssen immer mit Buchstaben oder Unterstrich beginnen

- Attribute Declaration

```
<!ATTLIST Elementname Attributdefinition>
```

- Attributdefinition bestehen aus:

Attributname

Attributtyp (implizit als CDATA, ID oder explizit als (Wert1 | Wert2 | ...))

Defaulttyp

- weitere Bestandteile: Kommentare, Processing Instructions, ...



eXtensible Markup Language



```
<? xml version="1.0" ?>  
  
<!DOCTYPE email SYSTEM "mail.dtd">  
  
<mail>  
  
  <empfaenger>schmiete@ivs.cs.uni-magdeburg.de</empfaenger>  
  
  <absender>dumke@ivs.cs.uni-magdeburg.de</absender>  
  
  <betreff>Vorlesung Web Services</betreff>  
  
  <nachricht>Prüfungstermin im Februar</nachricht>  
  
</mail>
```



eXtensible Markup Language



```
<!--mail DTD V.2-->
```

```
<!ELEMENT mail
```

```
    (empfaenger, absender, betreff, nachricht, termin*)>
```

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

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

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

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

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



eXtensible Markup Language



- XML Schema Language XS
- Definition einfacher und komplexer Datentypen
- Schema ist selbst XML-Dokument – d.h. validierbar
- Einfache Datentypen:
 - xs:integer, xs:decimal, xs:boolean, xs:string
- Komplexe Datentypen:
 - xs:complexType, xs:sequence, xs:group, ...
- Verwendung von Facets – Einschränken von Wertebereichen



eXtensible Markup Language



```
<? xml version="1.0" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="mail">
  ...
  <xs:complexType>
    <xs:sequence>
      <xs:element name="empfaenger" type="xs:string">
      <xs:element name="absender" type="xs:string">
      <xs:element name="betreff" type="xs:string">
      <xs:element name="nachricht" type="xs:string">
      <xs:element name="termin" type="xs:integer">
    </xs:sequence>
  </xs:complexType>
</xs:element></xs:schema>
```



eXtensible Markup Language

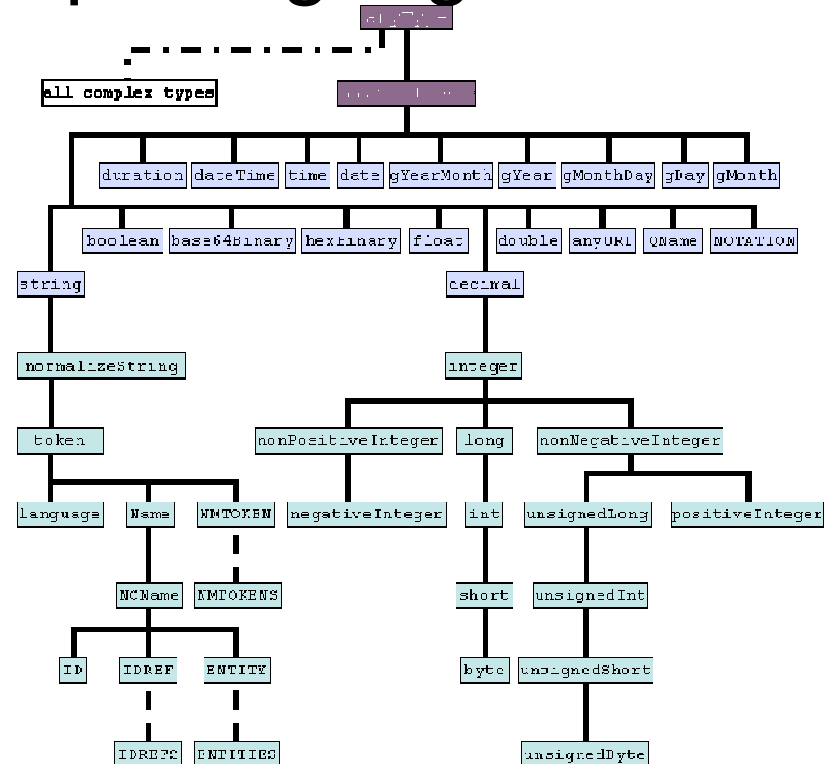


```
<xs:simpleType name="st_CallConnID">
  <xs:annotation>
    <xs:documentation>...</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:string">
    <xs:whiteSpace value="collapse" fixed="true"/>
    <xs:length value="14" fixed="false"/>
    <xs:pattern value="[A-Fa-f0-9]*"/>
  </xs:restriction>
</xs:simpleType>
```



eXtensible Markup Language

- Simple Types
 - Zeitangaben und Datum
 - Numerische Datentypen
 - Zeichenorientierte Daten
 - Complex Types
 - Gruppierung von Attributen
 - Reihenfolge & Hierarchie
- Sequence



- Urtyp
- Vordefinierter Primivtyp
- Vordefinierter abgeleiteter Typ
- Komplexer Typ
- Abgeleitet durch Einschränkung
- Abgeleitet durch Auffüstung
- Abgeleitet durch Einschränkung oder Auffüstung

Quelle der Grafik: <http://www.edition-w3c.de/TR/2001/REC-xmlschema-2-20010502/>



eXtensible Markup Language



- DOM Document Object Model
 - Plattform- und sprachunabhängiges API
 - Erlaubt externen Programmen dynam. Zugriff auf XML-Dokumente
 - Zugriff/Änderung auf Inhalte, Struktur und Layout

- SAX Single API for XML
 - Plattform- und sprachunabhängiges API
 - Erzeugt beim Parsen Events zur Steuerung externer Programme



eXtensible Markup Language



- Electronic Data Interchange EDI
 - Datenaustausch zwischen IT-Systemen – zunehmend XML-basiert
 - Primäre Verwaltung und Standardisierung durch die OASIS (600 Mitgl.)
- xCBL – XML Common Business Library – CommerceOne
 - Spezifikation für das Order Management
 - Beschreibt nicht nur die benötigten Dokumente, es werden ebenfalls Inhalte und deren Semantik festgelegt
- ebXML – electronic Business (ergänzt WS-Protokolle) - Framework
 - Komplette Spezifikation, Dokumentation und Ausführung des elektronischen Handels



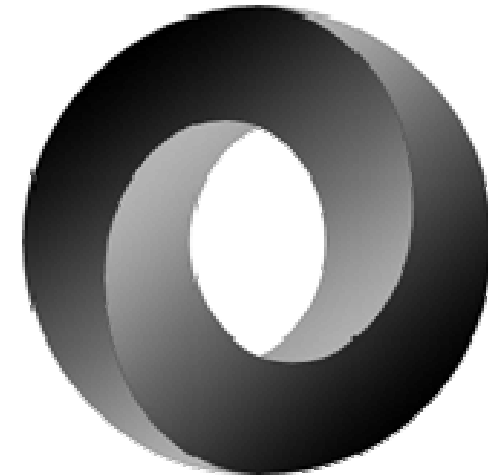
JSON – JavaScript Object Notation



JSON



- Einfacher zu verwenden als XML
- Einfaches Typsystem
- Eingeschränkte Typsicherheit
- Geringerer Overhead
- Programmiersprachenunabhängig
- Parsersysteme/Serialisierung



Quelle der Abbildung: <http://www.json.org/json-de.html>



JSON - Typsystem

Objektart	Format	Beispiel
String	Text innerhalb von Anführungszeichen	<code>"text"</code>
Zahlenwert	Folge von 0 bis 9, optional mit Dezimalpunkt und/oder Exponent	<code>1.6</code>
boolescher Wert	true oder false	<code>true</code>
Liste/Array	Aufgelistete Werte innerhalb von eckigen Klammern, durch Komma getrennt	<code>[wert, wert2, ...]</code>
Objekt/"assoziativer Array"	Index/Wert-Zuordnung mithilfe eines Doppelpunkts innerhalb von geschweiften Klammern, durch Komma getrennt	<code>{"attname": wert, "attname2": wert2, ...}</code>

Quelle: <http://www.webmasterpro.de/coding/article/json-als-xml-alternative.html>



JSON – offizielle Webseite



Introducing JSON

العربية বাংলা العربية 中文 Český Dansk Nederlands English Esperanto Français Deutsch Ελληνικά עברית Magyar Indonesia Italiano 日本語 한국어 العربية Polski Português Română Русский Срpsko-hrvatski Slovenščina Español Svenska Türkçe Tiếng Việt

ECMA-404 The JSON Data Interchange Standard.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

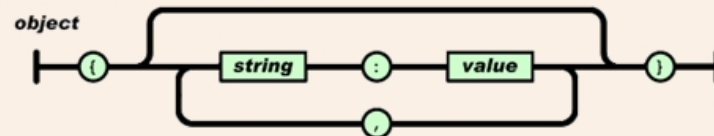
JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).



```

object
  {}
  { members }
members
  pair
  pair , members
pair
  string : value
array
  []
  [ elements ]
elements
  value
  value , elements
value
  string
  number
  object
  array
  true
  false
  null

```

Quelle: [ECMA-404 The JSON Data Interchange Standard](http://json.org/), <http://json.org/>



XML/JSON – Werkzeugunterstützung



XML-Werkzeugunterstützung



- Altnova XMLSpy
 - XML/JSON-Editor (Bearbeiten, Validieren, Dokumentieren)
 - Verwendung als Plug-In in z.B. Eclipse oder Visual Studio
 - Unterstützt die Grafische Erstellung von XML-Dokumenten→ <http://www.altova.com/de/xmlspy.html>

- Eclipse Web Tools Platform (WTP) Project
 - Entwicklung von Java EE Web-Anwendungen
 - HTML, Javascript, CSS, JSP, SQL, XML, DTD, XSD und WSDL
 - Wizard zur Erstellung von Web Services→ <http://www.eclipse.org/webtools/>



XML/JSON-Werkzeugunterstützung



The image shows a software interface for XML editing. On the left, a schema diagram shows an `OfficeType` element containing an `Address` element. The `Address` element is further detailed in two panels: `ipo:US-Address` (with fields `ipo:name`, `ipo:street`, `ipo:city`, `ipo:state`, `ipo:zip`) and `ipo:EU-Address`. Below the diagram is a toolbar with tabs for `Text`, `Grid`, `Schema`, `WSDL`, and `XBRL`. The `Schema` tab is active, showing a list of loaded schemas: `NanonullOrg`, `NanonullOrg`, and `YearlySales`.

In the center, a text editor displays a JSON array of book objects. The visible JSON data is as follows:

```

19  ], {
20  "Year": 2005,
21  "Title": "Understanding the Linux Kernel",
22  "ISBN10": "0596005652",
23  "Price": 32.97,
24  "Author": [
25    "Daniel P. Bovet", "Marco Cesati"
26  ]
27  }, {
28  "Year": 2009,
29  "Title": "XBRL For Dummies",
30  "ISBN10": "0470499796",
31  "Price": 19.79,
32  "Author": [
33    "Charles Hoffman", "Liv Watson"
34  ]
35  }, {
36  "Year": 2011,
37  "Title": "XBRL For Dummies",
38  "ISBN10": "007007007007",
39  "Price": 35.99,

```

On the right, a `JSON-Editor` window is shown with a tree view of `Elements` and an `Attributes` section. The `Elements` list includes: `Author`, `Book`, `BookList`, `Comment`, `encoding`, `ISBN10`, `Price`, `Title`, `version`, `XML`, `xmlns:xsi`, `xsi:noNamespaceSchemaL`, and `Year`.

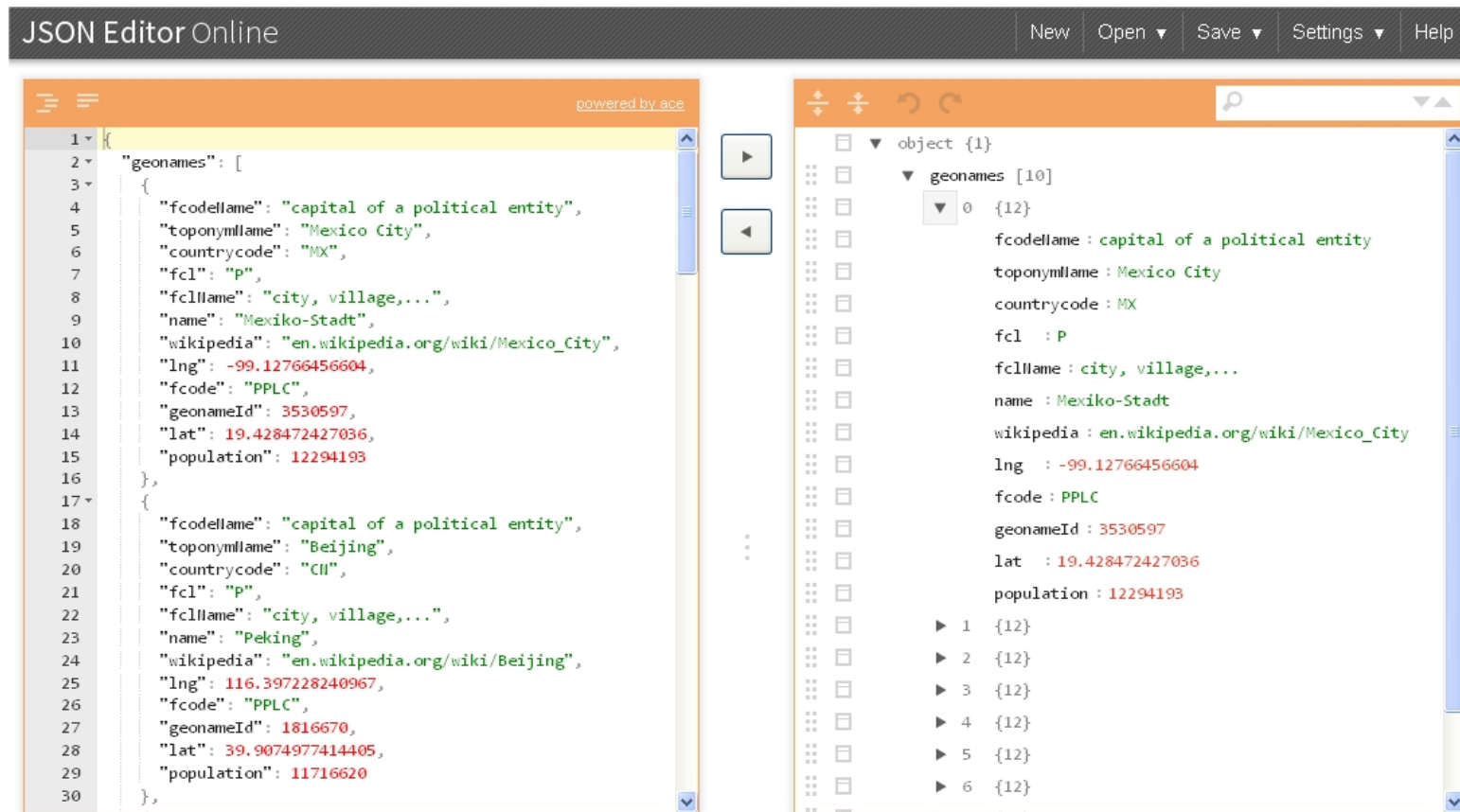
An arrow labeled `JSON-Editor` points from the JSON data area to the `JSON-Editor` window.

XML-Editor

Quelle: www.altova.com



Werkzeugunterstützung JSON



Quelle der Abbildung: <http://www.jsoneditoronline.org/> (letzte Verwendung: Oktober 2016)