ARS

Member of TIMETOACT GROUP SOFTWARE & CONSULTING

Document Object Model (DOM)

XML-Verarbeitung mit Java

ARS

Inhalte dieses Kapitels

Allgemeines

Java DOM API

Parsen und Serialisieren

Sonstige Java APIs

ARS

Allgemeines [1|3]

- Standard des W3C seit 1998 (http://www.w3.org/DOM/)
- Präsentation von Baumstrukturen
  - Plattformübergreifend
  - Sprachenübergreifend
- Keine Versionierung
- Unterteilung in *DOM Levels*
- Sprachbindungen für verschiedene Sprachen
  - Javascript: Zugriff auf HTML-Baum im Browser
  - Java: Lesen und Schreiben von XML

ARS

Allgemeines [2|3]

DOM Levels

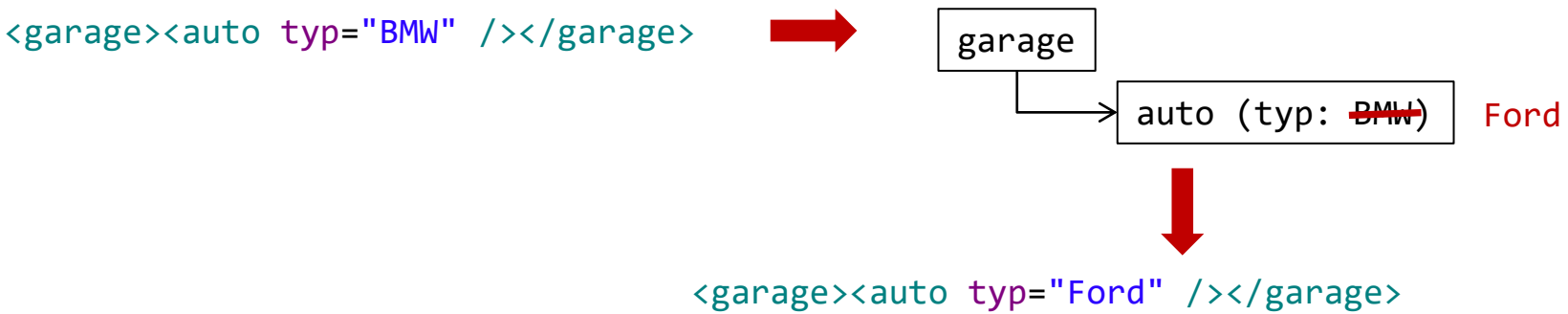
- Level 1
  - Grundlegende Funktionalität (Core)
  - Navigation innerhalb des Dokuments
- Level 2
  - Verwendung von Namensräumen und Stylesheets
  - Unterstützung für bestimmte Inhaltsmodelle (XML, HTML, usw.)
  - Events
  - Selektion und Iteration über Knotenmengen (Traversal und Range)
- Level 3
  - Standardisiertes Lesen und Speichern (Load and Save)
  - Validierung
  - XPath-Unterstützung

ARS

Allgemeines [3|3]

Funktionsweise bei I/O

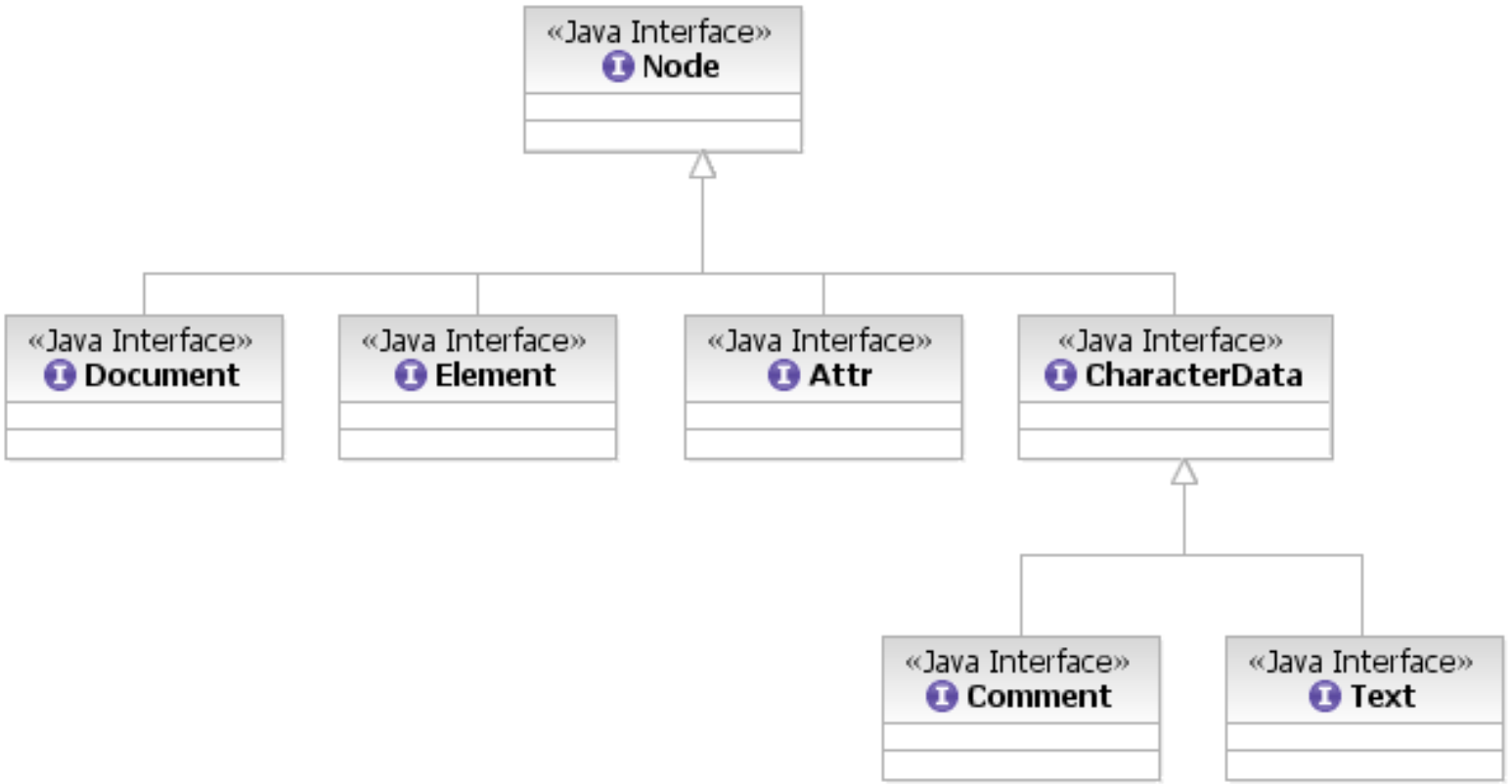
- Erzeugung eines Objektbaums auf Basis der XML-Struktur
- Nur der komplette Baum wird geliefert
  - Jederzeit Zugriff auf jeden Teil einer XML-Datei
  - Hoher Speicherverbrauch
  - Geringe Performanz und Reaktionszeit
- Manipulation des Objektmodells
  - Objekte (*Knoten*, *Nodes*) ändern, entfernen, einfügen
  - Ausgabe der Struktur im Speicher in XML-Dokument



ARS

Java DOM API [1|7]

- Java-Implementierung der W3C-standardisierten Typem
- Package `org.w3c.dom`
- Zugriff auf die geprüfte Struktur über `Document`-Objekt
- Elemente sind Knoten (`Node`-Objekte)
- Methoden zum
  - Navigieren
  - Auslesen
  - Ändern

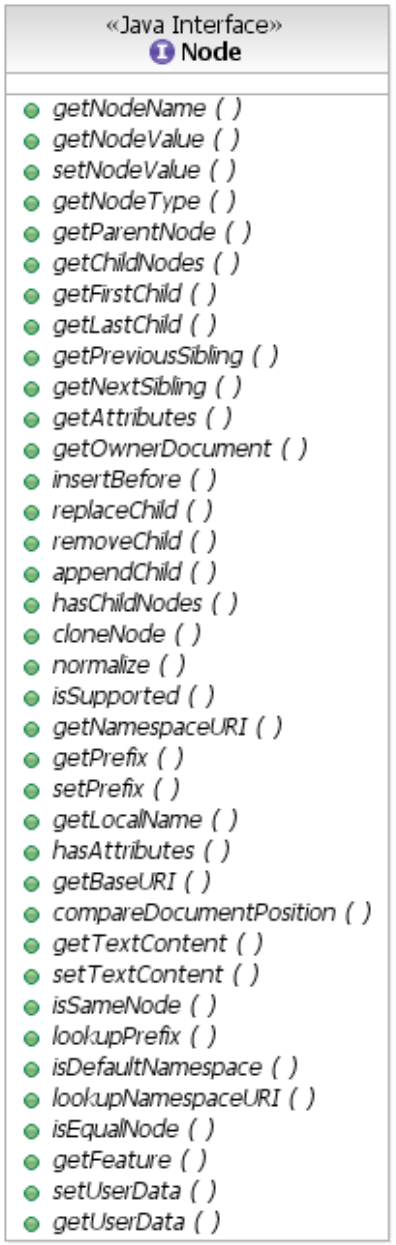


ARS

Java DOM API [2|7]

Interface `Node`

- Knoten innerhalb eines Dokuments
- Zugriff auf Eigenschaften
  - Name: `getNodeName()`
  - Wert: `getNodeValue()` und `setNodeValue()`
  - Typ: `getNodeType()`
  - Attribute: `getAttributes()`
- Ermittlung von Eltern-, Geschwister und Kindknoten
  - `getParentNode()`
  - `getPreviousSibling()`, `getNextSibling()`
  - `getChildNodes()`, `getFirstChild()`, `getLastChild()`
- Einfügen/Ändern/Löschen von Kindknoten
  - `insertBefore()`, `appendChild()`
  - `replaceChild()`, `removeChild()`

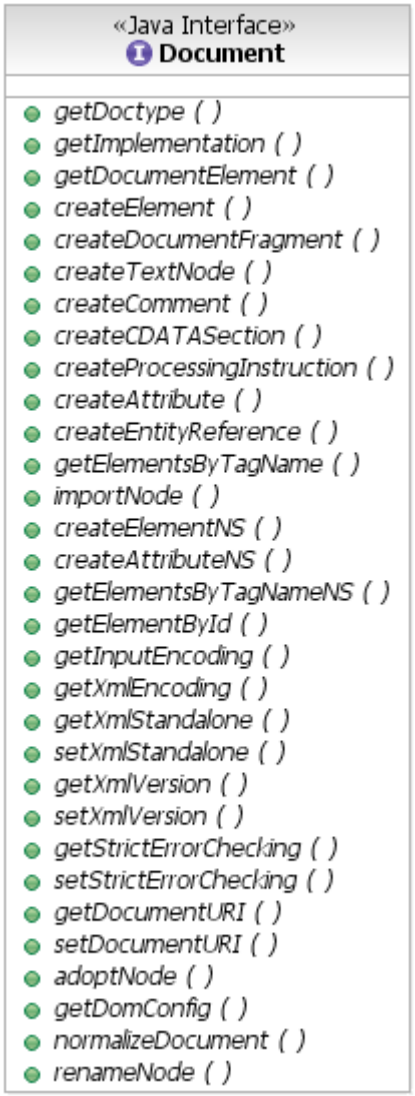


ARS

Java DOM API [3|7]

Interface `Document`

- Dokument (Wurzelelement)
- Factory-Methoden zum Erzeugen von Knoten, bspw.
  - `createElement()`
  - `createTextNode()`
  - `createAttribute()`
- Zugriff auf Wurzelelement: `getDocumentElement()`
- Zugriff auf Knoten
  - Name des Tags: `getElementsByTagName()`
  - Id: `getElementById()`
- Informationen über das Dokument
  - XML-Version: `getXmlVersion()` und `setXmlVersion()`
  - Encoding: `getXmlEncoding()`
  - Document URI: `getDocumentURI()` und `setDocumentURI()`

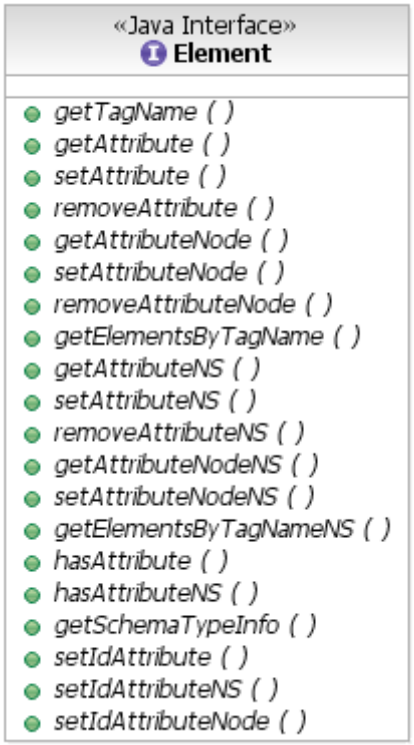


ARS

Java DOM API [4|7]

Interface `Element`

- Stellt ein Element dar
- Leitet von `Node` ab
- Erweiterter Zugriff auf Attribute
  - Werte: `getAttribute()`und `setAttribute()`
  - Knoten: `getAttributeNode()`, `setAttributeNode()`
  - Löschen: `removeAttribute()`, `removeAttributeNode()`



Interface `Attr`

- Repräsentiert ein einzelnes Attribut
- Leitet von `Node` ab
- Ermitteln und Setzen des Wertes
- Element ermitteln, zu dem das Attribut gehört: `getOwnerElement()`

ARS

Java DOM API [5|7]

Beispiel: Baumstruktur erzeugen

```
// Wurzelelement ermitteln
Element root = (Element) document.getDocumentElement();

// Neues Element mit einem Textknoten erstellen
Element trainingType = document.createElement("training");
Element shortName = document.createElement("title");
Text shortNameText = document.createTextNode("XML4JAVA");

// Neues Element dem Root-Element zuordnen
trainingType.appendChild(shortName);
shortName.appendChild(shortNameText);
root.appendChild(trainingType);
```

```
<training>
  <title>XML4JAVA</title>
</training>
```

ARS

Java DOM API [6|7]

Beispiel: Baumstruktur auslesen

```
// Attribute eines Elements auslesen
NamedNodeMap attributes = element.getAttributes();
for (int i = 0; i < attributes.getLength(); i++) {
  Attr attribute = (Attr) attributes.item(i);
  System.out.println(attribute.getValue());
}

// Zugriff auf die Unterelemente
NodeList children = root.getChildNodes();
NodeList shortNames = root.getElementsByTagName("short-name");
Node shortName0 = shortNames.item(0);
Node shortName1 = shortNames.item(1);
```

ARS

Java DOM API [7|7]

Weitere Beispiele

```
// Zugriff auf Vater-Element
Node parent = rootElement.getParentNode();

// Löschen eines Elements
rootElement.removeChild(shortName0);

// Inhalte eines Elements überschreiben
Text training0Text = (Text) shortName1.getFirstChild();
training0Text.setData("XML4JAVA");
```

ARS

Hinweise

Verwendung der DOM API nicht immer intuitiv!

- Fast alle Elemente sind von `Node` abgeleitet
- Verhalten einiger geerbten Methoden verwirrend bzw. nicht intuitiv
  - Methode `getAttributes()` liefert für Textelemente **`null`**
  - Methode `getChildNodes()` liefert für Elemente nur die Kindknoten, nicht die Attribute (Attribute sind laut API aber Knoten)
- Methode `getNodeType()` liefert als Ergebnis Wert vom Typ `short`
  - Prüfung auf in `Node` definierte Konstanten
- Keine Verwendung von Java-Spezifika
  - ~~Collections API~~
  - ~~Generics~~

ARS

Parsen

- Standardisierung über `javax.xml.parsers.*`
- Verwendung von SAX
  - Ankündigung einer `SAXException`
  - Verwendung von `ErrorHandler`, `EntityResolver` möglich

```
// Aufbau der Factory und des Parsers
DocumentBuilderFactory factory =
                    DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
// Parsen
Document doc = builder.parse("file://...");
// Nutzen des DOM
NodeList elementsByTagName = doc.getElementsByTagName("auto");
…
```

ARS

Serialisieren

- Standardisierung über `javax.xml.transform.*`

```
// Aufbau der Factory und des Transformers

TransformerFactory transformerFactory =
                                TransformerFactory.newInstance();

Transformer transformer = transformerFactory.newTransformer();

// Aufbau des Ausgabeziels

DOMSource source = new DOMSource(doc);

StreamResult consoleResult = new StreamResult(…);

// Serialisierung

transformer.transform(source, consoleResult);
```

ARS

Externe Java APIs (Auszug) [1|2]

JDOM

- Open-Source-Projekt
- Lizenz ähnlich zu Apache
- API Java-optimiert (v.a. Collections)
- Package `org.jdom2.*`

```
SAXBuilder saxBuilder = new SAXBuilder();

Document document = saxBuilder.build(inputFile);

List<Element> studentList = document.getRootElement().getChildren();
```

ARS

Externe Java APIs (Auszug) [2|2]

DOM4J

- Open-Source-Projekt
- Host: GitHub (https://dom4j.github.io/)
- API Java-optimiert (v.a. Collections)
- Optimiert für geringen Speicherverbrauch (großes XML)
- Package `org.dom4j.*`

```
SAXReader reader = new SAXReader();
Document document = reader.read(url);
for (Iterator<Element> it = document.getRootElement().elementIterator();
        it.hasNext();) {
    Element element = it.next();
    // do something
}
```