

PRÜFUNG IN "SEMI-STRUKTURIERTE DATEN" 184.705			26. 06. 2018
Kennnr.	Matrikelnr.	Familienname	Vorname

Arbeitszeit: 100 Minuten.

Aufgaben sind auf den Angabeblättern zu lösen; Zusatzblätter werden nicht gewertet.

Bitte tragen Sie Namen, Studienkennzahl und Matrikelnummer ein. Halten Sie Ihren Studentenausweis bereit.

Aufgabe 1:

(12)

Betrachten Sie folgende XML Schema Datei **test.xsd**:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="A">
    <xsd:complexType mixed="true" >
      <xsd:all>
        <xsd:element name="B" minOccurs="0" type="xsd:boolean"/>
        <xsd:element name="C" type="typeC"/>
      </xsd:all>
    </xsd:complexType>
    <xsd:unique name="key1">
      <xsd:selector xpath="//C"/>
      <xsd:field xpath="@id"/>
    </xsd:unique>
  </xsd:element>

  <xsd:complexType name="typeC" mixed="false">
    <xsd:sequence>
      <xsd:element name="C" type="typeC" minOccurs="0" maxOccurs="2"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:integer"/>
  </xsd:complexType>
</xsd:schema>
```

Betrachten Sie weiters die acht verschiedenen XML-Dateien, die unten angeführt sind.

Sie können davon ausgehen, dass alle folgenden XML-Dateien wohlgeformt sind. Es geht also lediglich darum, ihre Gültigkeit bezüglich test.xsd zu entscheiden.

Kreuzen Sie an, welche der folgenden xml-Dateien gültig bezüglich test.xsd sind.

- 1. <A><C/> valid invalid
- 2. <A>SSD<C></C> valid invalid
- 3. <A>0 valid invalid
- 4. <C id="42"/> valid invalid
- 5. <A>SSDwahr<C></C> valid invalid
- 6. <A>SSD<C></C>1 valid invalid
- 7. <A><C><C id="12"></C></C>0 valid invalid
- 8. <A><C id="12"><C id="12"></C></C></C>0 valid invalid

(Für jede korrekte Antwort 1.5 Punkte, für jede falsche Antwort -1.5 Punkte, unbeantwortete Fragen 0 Punkt, Insgesamt nicht weniger als 0 Punkte)

Aufgabe 2:

(15)

Beantworten Sie, die folgenden Fragen kurz und bündig (Für jede korrekte Antwort 1.5 Punkte).

1. Welches Datenmodell verwenden wir für semistrukturierte Daten?
2. Geben Sie einen Formalismus (ein Format) an um semistrukturierte Daten zu speichern:
3. Wofür steht die Abkürzung DTD?
4. Wie verhält sich der Speicherbedarf von Event-based Parsern wie SAX zur Größe des geparsen XML Dokuments?
5. Welche in der Vorlesung behandelte API erlaubt wahlfreien Zugriff auf das gesamte XML Dokument?
6. Welche der in der Vorlesung behandelten Sprachen enthalten XPath?
7. Wir haben verschiedene Sprachen um Schemata zu definieren kennen gelernt. Welche hat die größte Ausdrucksstärke?
8. Zu welchem Zweck werden XML Namespaces verwendet?
9. Wieso können URIs nicht direkt als prefix für XML Elemente/Attribute verwendet werden?
10. Was macht das Default-Template in XSLT für Attribute?

Die folgenden Aufgaben 3 – 7 beziehen sich auf das XML-Dokument `travelAgency.xml`, das Sie auf der letzten Seite dieser Prüfungsangabe finden.

Aufgabe 3:

(12)

Vervollständigen Sie das DTD Dokument `travelAgency.dtd`, sodass XML-Dokumente in der Gestalt von `travelAgency.xml` (siehe Anhang) bezüglich dieser DTD gültig sind. Berücksichtigen Sie beim Erstellen der DTD folgende Punkte:

- Das `sportcamps` Element enthält beliebig viele `summercamp` und `wintercamp` Elemente.
- Das `customers` Element enthält beliebig viele `customer` Elemente.
- Das `bookings` Element enthält beliebig viele `booking` Elemente.
- `summercamp` und `wintercamp` Elemente enthalten ein `name` Element, möglicherweise ein `venue` Element, mindestens ein `sport` Element, ein `duration` Element und beliebig viele `date` Elemente (in dieser Reihenfolge).
- `summercamp`, `wintercamp` und `customer` Elemente haben als Attribut eine eindeutige ID.
- In `venue` Elementen sind möglicherweise Länder durch `country` Elemente ausgezeichnet. Jedes `venue` Element hat ein `id` Attribut.
- `booking` Elemente haben ein `customerid` Attribut das auf die ID eines `customer` Elements verweist, und ein Attribut `campid` das auf die ID eines Camps verweist. Zusätzlich gibt es ein optionales `date` Attribut.
- Wenn nicht angegeben treffen Sie plausible Annahmen über Typen von Attributen und Elementen.

File `travelAgency.dtd`:

```
<!ELEMENT travelAgency (sportcamps,customers,bookings)>
```

Aufgabe 4:

(10)

Betrachten Sie die folgenden XPath-Abfragen angewandt auf das Dokument **travelAgency.xml** (siehe Anhang).

- Falls der angegebene XPath Ausdruck keine Knoten selektiert, notieren Sie im entsprechenden Feld "leere Ausgabe".
- Falls als Ergebnis eine Zahl selektiert wird (count, sum, ...), geben Sie diese Zahl an.

Geben Sie nun die entsprechende Ausgaben der folgenden XPath-Abfragen an.

```
sum(//summercamp/duration)
```

```
//customer[@id=//booking/@customerid][2]
```

```
//sportcamps/*[@id=//booking/@campid][last()-1]/name/text()
```

```
//customer[last()][@id=//booking/@customerid]
```

```
//summercamp[not(@id=//booking/@campid)][last()]/name
```

Aufgabe 5:

(8)

Betrachten Sie folgende XQuery **xquery.xq**:

```
<venues>
{
  for $vid in distinct-values(//venue/@id)
  let $v := distinct-values(//venue[@id = $vid])
  where count(//sport[preceding-sibling::venue/@id = $vid])>2
  order by $v
  return
    <venue><name>{$v}</name>
    {
      for $s in //sport[preceding-sibling::venue/@id = $vid]
      order by $s
      return
        <sport>{$s/text()}</sport>
    }
  </venue>
}
</venues>
```

Geben Sie nun die Ausgabe von **xquery.xq** angewandt auf **travelAgency.xml** an.

Sie müssen sich nicht um Whitespaces kümmern.

Aufgabe 6:

(10)

Schreiben Sie ein XSLT Dokument um folgende Aufgabenstellung zu lösen:

Es soll ein HTML Dokument mit Teilnehmerlisten für alle Wintercamps ausgegeben werden.

Für jedes Wintercamp wird ein Element `h1` mit dem Namen des Camps als Überschrift erzeugt.

In einem `ul` Element werden die Campteilnehmer ausgegeben. Für jeden Campteilnehmer wird ein `li` Element angelegt. Der Inhalt des `li` Elements ist der Name des Campteilnehmers. Campteilnehmer können über den `bookings` Teilbaum bestimmt werden.

Zusätzlich soll die Anzahl der Campteilnehmer in einem `p` Element ausgegeben werden.

Ihre XSL-Transformation soll, angewandt auf `travelAgency.xml`, folgende Ausgabe liefern:

```
<h1>Advanced Skiing</h1>
<ul>
  <li>Arthur Dent</li>
</ul>
<p>Total participants: 1</p>
```

Hinweis: Sie müssen nur den Inhalt des `body` Elements ausgeben.

Geben Sie nun ihre XSL-Transformation an:

Betrachten Sie die folgende Java Klasse und geben Sie die Ausgabe der Klasse an, wenn als Input die Datei `travelAgency.xml` verwendet wird.

```
public class RunSAX extends DefaultHandler {
    private String eleText, id; private int total = 0;
    private HashMap<String, List<String>> s = new HashMap<String, List<String>>();
    private HashMap<String, List<String>> c = new HashMap<String, List<String>>();

    public void characters(char[] text, int start, int length) throws SAXException {
        eleText = new String(text, start, length);
    }

    public void startElement(String namespaceURI, String localName, String qName, Attributes atts)
        throws SAXException {
        if ("summercamp".equals(localName) || "wintercamp".equals(localName)) {
            id = atts.getValue("id");
            s.put(id, new LinkedList<String>());
        }

        if ("booking".equals(localName)) {
            id = atts.getValue("customerid");
            if (!c.containsKey(id))
                c.put(id, new LinkedList<String>());
            c.get(id).add(atts.getValue("campid"));
        }
    }

    public void endElement(String namespaceURI, String localName, String qName) throws SAXException {
        if ("sport".equals(localName)) s.get(id).add(eleText);
    }

    public void endDocument() throws SAXException {
        for (String key : c.keySet()) {
            System.out.print(key + ": ");
            String out = "";
            for (String cid : c.get(key))
                for (String sp : s.get(cid))
                    out += sp + ", ";
            System.out.println(out.substring(0,out.length()-2));
        }
    }

    public static void main(String[] args) throws Exception {
        if (args.length != 1) {
            System.err.println("Usage: java RunSAX <input.xml>");
            System.exit(1);
        }

        String input = args[0];
        InputSource source = new InputSource(new FileInputStream(input));

        XMLReader xr = XMLReaderFactory.createXMLReader();
        RunSAX rs = new RunSAX();
        xr.setContentHandler(rs);

        xr.parse(source);
    }
}
```

Geben Sie hier die Ausgabe an:

Total points: 75

```

<travelAgency>
  <sportcamps>
    <summercamp id="c1">
      <name>Fun Diving</name>
      <venue id="v1">Krak, <country>Croatia</country></venue>
      <sport>Scuba Diving</sport>
      <duration>7</duration>
      <date>5.7.2018</date>
      <date>5.8.2018</date>
    </summercamp>
    <summercamp id="c2">
      <name>Hiking & Biking with kids</name>
      <venue id="v2">Hintertal, Austria</venue>
      <sport>Mountain Biking</sport>
      <sport>Hiking</sport>
      <duration>6</duration>
      <date>15.7.2018</date>
    </summercamp>
    <summercamp id="c3">
      <name>Big Waves Tour</name>
      <venue id="v3">Honolulu, Hawaii</venue>
      <sport>Surfing</sport>
      <sport>Sailing</sport>
      <duration>14</duration>
      <date>1.8.2018</date>
    </summercamp>
    <wintercamp id="c4">
      <name>Advanced Skiing</name>
      <venue id="v2">Hintertal, Austria</venue>
      <sport>Skiing</sport>
      <sport>Snow Board</sport>
      <duration>7</duration>
    </wintercamp>
    <summercamp id="c5">
      <name>Sailing for beginners</name>
      <venue id="v4">Rust, <country>Austria</country></venue>
      <sport>Sailing</sport>
      <duration>5</duration>
      <date>15.8.2018</date>
    </summercamp>
  </sportcamps>
  <customers>
    <customer id="p1">Sophie Haas</customer>
    <customer id="p2">Samuel Mumm</customer>
    <customer id="p3">Arthur Dent</customer>
    <customer id="p4">Harvey Dent</customer>
    <customer id="p5">Miles O'Brien </customer>
  </customers>
  <bookings>
    <booking customerid="p2" campid="c2" date="15.7.2018"/>
    <booking customerid="p3" campid="c4" date="6.1.2017"/>
    <booking customerid="p1" campid="c1" date="5.7.2018"/>
    <booking customerid="p1" campid="c3" date="5.8.2016"/>
    <booking customerid="p4" campid="c5" />
  </bookings>
</travelAgency>

```