**Einsendeaufgaben zu XML02**

|  |  |  |
| --- | --- | --- |
| Name: Zang | Vorname:Keping | **Ihr Fernlehrer: Frau Dr. Ahlborn** |
| Postleitzahl und Ort:73760 Ostfildern | Straße:Scharnhauser straße |  |
| Studien-Nr.:793302 | Lehrgangs-Nr.:460 | Note:1 100P |
| Einsendeaufgabe: **XML02-XX** | Auflage: **3** |  |

Füllen Sie das Adressfeld (*die nicht hinterlegten Felder*) bitte sorgfältig aus.

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Aufgaben/Lösung** | **Punkte** |
| 1. | Das nachstehende XML-Dokument „Auftrag“ ist noch unvollständig: Vervollständigen Sie dieses Dokument.   |  | | --- | | <?xml version="1.0" encoding=“ISO-8859-1“ ?>  <Auftrag>     <Auftragskopf>        <Adresse>           <Name>            <Nachname>Müller</Nachname>            <Vorname>David</Vorname>           </Name>           <Strasse>Sonnenstraße</Strasse>           <Hausnummer>6</Hausnummer>           <Ort PLZ=“12345“>Berlin</Ort>           <E-Mail href=“jedermann@gmx.de“></E-Mail>        </Adresse>     </Auftragskopf>     <Positionen>        <Bestellposition>           <Bezeichnung>Lernheft XML 1</Bezeichnung>           <Artikelnummer>4711</Artikelnummer>           <Artikelanzahl>5</Artikelanzahl>        </Bestellposition>        <Bestellposition>           <Bezeichnung>Lernheft XML 2</Bezeichnung>           <Artikelnummer>4712</Artikelnummer>           <Artikelanzahl>5</Artikelanzahl>       </Bestellposition>      </Positionen>  </Auftrag> | | 12P |
| a) Ergänzen Sie den Prolog, sodass im Dokument auch Umlaute erlaubt sind.  ...  Die Aufgabe liegt auch noch einmal als separate Datei Auftrag.xml vor.  encoding=”ISO-8859-1” oder encoding=”uft-8”“ | + |
| b) Ergänzen Sie den unteren Teil „Positionen“, sodass ein wohlgeformtes Dokument entsteht.  ... siehe separate Datei Auftrag.xml | 3P |
| c) Ergänzen Sie den Teil „Adresse“: Eine Adresse soll die Elemente  „Name“ (untergliedert in „Nachname“ und „Vorname“), „Strasse“, „Hausnummer“, „Ort“ und „E-Mail“ enthalten. Die Postleitzahl („PLZ“) soll als Attribut von „Ort“ vorkommen. „E-Mail“ soll als leeres Element mit dem Attribut einer E-Mail-Adresse dargestellt werden.  ... <Name>            <Nachname>Müller</Nachname>            <Vorname>David</Vorname>           </Name>           <Strasse>Sonnenstraße</Strasse>           <Hausnummer>6</Hausnummer>           <Ort PLZ=“12345“>Berlin</Ort>           <E-Mail href=“jedermann@gmx.de“></E-Mail> | 6P |
| 2. | Ein XML-Dokument „Adresse“ hat folgendes Aussehen:   |  | | --- | | <?xml version="1.0" encoding="ISO-8859-1" ?>  <Adresse>     <Nachname Vorname="Hugo">Jedermann</Nachname>     <E-Mail>jedermann@t-online.de</E-Mail>     <PLZ>123456</PLZ>     <Strasse Hausnummer="17">Blattweg</Strasse>     <Ort>Baumstadt</Ort>     <Adresszusatz>Hinterhof</Adresszusatz>  </Adresse> | |  |
| a) Erstellen Sie zu diesem Dokument eine DTD „Adresse.dtd“, die folgen-den Anforderungen genügt:  – Die Elemente „Nachname“, „Strasse“ und „Ort“ kommen genau einmal vor. (🡪 keine Sonderzeichen )  – Das Element „E-Mail“ ist optional und kann mehrfach vorkommen. (🡪 \* Zeichen)  – Ein „Adresszusatz“ kann einmal oder gar nicht vorkommen. (🡪 ? Zeichen)  – „Vorname“ ist ein Attribut, das vorkommen muss. (🡪 #REQUIRED)  ...die anderen Tags wurden von mir auch passend in das dtd übertragen. Ich habe bei der Hausnummer angenommen, dass eine Straße ohne Nummer nicht hilfreich ist und daher das Attribut required gesetzt.  vollständige Dokument 🡪 siehe separate Datei „Adresse.dtd“ | 12P |
| b) Wie lautet die Ergänzung im XML-Dokument zur Referenzierung der DTD?  ...<!DOCTYPE Adresse SYSTEM “Adresse.dtd”> | 4P |
| 3. | Ein XML-Dokument „Positionen.xml“ hat folgendes Aussehen:   |  | | --- | | <?xml version="1.0"?>  <Positionen>    <Bestellposition>      <Bezeichnung>Lernheft XML 1</Bezeichnung>      <Artikelnummer>4711</Artikelnummer>      <Artikelanzahl>5</Artikelanzahl>    </Bestellposition>    <Bestellposition>      <Bezeichnung>Lernheft XML 2</Bezeichnung>      <Artikelnummer>4712</Artikelnummer>      <Artikelanzahl>5</Artikelanzahl>    </Bestellposition>  </Positionen> |   Erstellen Sie zu diesem Dokument eine DTD „Positionen.dtd“, die folgenden Anforderungen genügt: |  |
| a) Eine Bestellposition muss mindestens einmal vorkommen.  ... <!ELEMENT Positionen (Bestellposition+)> | 2P |
| b) Ein Element „Bezeichnung“ muss nicht, kann aber mehrere Male vorkommen.  ...  <!ELEMENT Bestellposition (Bezeichnung\*,Artikelnummer,Artikelanzahl)> | 2P |
| c) „Artikelnummer“ und „Artikelanzahl“ kommen jeweils genau einmal vor.  ...  <!ELEMENT Bestellposition (Bezeichnung\*,Artikelnummer,Artikelanzahl)> | 2P |
| 4. | a) Gestalten Sie ein CSS-Stylesheet „Positionen.css“ zum XML-Dokument „Positionen.xml“, sodass die Datei ungefähr in Form der Abb. F.1 dargestellt wird. Die Schrifthintergründe sollen blau bzw. gelb erscheinen.    **Abb. F.1:** Darstellung des XML-Dokuments „Positionen.xml“ mit einem Stylesheet  ... siehe Position.css | 17P |
| b) Ergänzen Sie das Dokument „Positionen.xml“ um die Referenzierung auf das Stylesheet „Positionen.css“ und speichern Sie das Dokument unter dem Namen „Positionen4.xml“.  ...siehe Position4.xml (getested unter Windows 10 : Firefox Browser) | 4P |
| 5. | Die Browserdarstellung der Abb. F.2 wurde mit dem XSL-Stylesheet „Positionen.xsl“ erstellt.    **Abb. F.2**: Darstellung der XML-Datei „Positionen.xml“ mit einem XSL-Stylesheet  Ergänzen Sie das Stylesheet „Positionen.xsl“ um die noch fehlenden Anweisungen, speichern Sie die Datei unter dem Namen „Positionen5.xml“ und reichen Sie diese ein.   |  | | --- | | <?xml version="1.0" ?>  <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"                  version="1.0">    <xsl:template match="/">      <html>        <head>          <title>\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_</title>        </head>        <body>          \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_        </body>      </html>    </xsl:template>    <xsl:template match="**Positionen**">      <h2>Folgende Positionen werden geliefert:</h2>      \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_      <xsl:apply-templates />    </xsl:template>    <xsl:template match="\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_">      <i>Bestellposition:</i><br />      <b><xsl:value-of select="Bezeichnung" /></b><br />      <b>Anzahl:</b>      <xsl:value-of select="\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" /><br />      <b>Artikel:</b>      <xsl:value-of select="\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" /><br />      <hr />    </xsl:template>  </xsl:stylesheet> |   Siehe Position.xsl | 15P |
| b) Wie referenzieren Sie das Stylesheet im XML-Dokument?  ... <?xml-stylesheet href="Positionen.xsl"type="text/xsl"?> | 4P |
| c) Erläutern Sie an den Beispielen „Positionen.css“ und „Positionen.xsl“ die unterschiedlichen Darstellungsmöglichkeiten mit Cascading Style-sheets (CSS) und mit der Stylesheet-Sprache „XSL“.  ...XSL ist eine regelbasierte Sprache, die ein XML-Dokument in ein anderes Dokument transformiert.  CSS ist eine einfache Sprache ohne Regeln, mit der wir Layoutinformationen angeben können. XML-Dokument wird nicht durch CSS transformiert.  Dabei ist CSS nicht so mächtig, aber dafür einfacher.  Es gilt dabei allgemein:  Mit CSS kann man keine HTML-Dateien erzeugen. Mit XSL kann man HTML-Dateien erzeugen.  In CSS werden die Elemente durch Selektoren identifiziert und dann mit zugeordneten Eigenschaften verarbeitet.  In XSL werden die Elements(Knoten) durch einen Mustervergleich identifiziert und mit einer formulierter Verarbeitungsregel verarbeitet.  Man erkenn deutlich, dass in Position.css den Eigenschaften der Tags Bezeichnung, Artikelnummer, … usw. Werte zugewiesen werden. Die Tags werden anhand des Namens identifiziert und ihre Properties (z.B. Farbe ) werden verändert. Es erfolgt dabei aber weder eine Transformation noch eine regelbasierte Abfrage nach Tags. Es ist eine simple „Selector – key – value“ Zuweisung (welcher Tag, welche Property, welcher Wert). Auch ist es aufgrund fehlender Transformation nicht möglich, gezielt Elemente auszublenden (also nur einen Teil des Dokumentenbaums) darszustellen.  Auf der anderen Seite besitzt Position.xsl einen komplexeren Aufbau. Anhand von Mustern können Elemente identifiziert werden (z.B. Element muss diesen Namen haben) und anschließend gewissen Anweisungen auf sie angewandt werden. So wird in <xsl:template match=“Position“> nach Elementen gesucht mit Position als Name und anschließen an diese Stelle der HTML Header „Folgende Positionen werden geliefert: „ eingefügt. Auch kann man erkennen, dass der Dokumentenbaum verändert werden könnte. So wird in <xsl:template match = “Bestellposition“> erst nach den Bestellpositionen gesucht wird und danach die Kind-Elemente für jedes gefundene Element angezeigt werden (<xsl:value-of select=“Bezeichnung“ />). Hier nimmt man zwar die Kindelemente könnte aber sicherlich auch andere Elemente innerhalb des Baumes anzeigen (z.B. Kindes Kinder – so dass bei einer Ansicht die Kinderebene weggelassen wird / das würde bei unserem Beispiel aber keinen Sinn machen, da wir keine Kindes Kinder haben). | 9P |
| 6. | a) In welchem Zusammenhang wird folgender Namensraum verwendet?   |  | | --- | | <xsl:stylesheet  xmlns:xsl=**"http://www.w3.org/1999/XSL/Transform"**  version="1.0"> |   ...Im Zusammenhang mit der Definition eines XSL-Stylesheets wird ein solcher Namensraum verwendet. | 4P |
| b) Was bewirkt die Angabe dieses Namensraumes?  ...Mit dieser Angabe sind die Ausdrücke, die mit <xsl:… beginnen , nicht als Tags, sondern als Verarbeitungsanweisungen des XSL-Stylesheets interpretiert werden. Der angegebene Namensraum ist in dem Style-Sheet-Prozessor bekannt. Nur so kann man erkennen, was Transformationsanweisungen sind und was nicht. | 4P |
| 7. | In welchem Zusammenhang werden XPath-Ausdrücke verwendet?  ...XPath-Ausdrücke werden im Zusammenhang mit XSL-Stylesheets in unterschiedlichen Kontext verwendet. Z.B:   1. Ausdruck zur Identifizierung von Knoten. 2. Ausdruck zur gezielten Auswahl von Elementen. 3. Ausdruck zur gezielten Weiterverarbeitung von Knoten. | 9P |