Übungsblatt 2 WT:III

Bis zum 14.05.2013 sind Lösungen zu folgenden Aufgaben abzugeben: 1, 3, 5, 7.

Aufgabe 1 : SGML, HTML, XML

Erläutern Sie, ausgehend von der SGML-Dokumentenverarbeitung, den Zusammenhang von SGML, HTML und XML. Welche dieser Auszeichnungssprachen sind mächtig genug, um eine der anderen genannten Sprachen nachzubilden, welche nicht? Begründen Sie Ihre Antworten.

Aufgabe 2 : Programmieraufgabe: Web-Crawler

Sie entschließen sich, einen Web-Crawler (auch Web-Spider genannt) zu implementieren. Dabei soll Ihr Web-Crawler ausgehend von einer Start-URL im Internet das zugehörige Dokument auf Ihren PC laden und in einer Datei speichern. Alle Links, die im Dokument vorkommen und auf weitere HTML-Web-Seiten verweisen, sollen in einer Queue gespeichert und nacheinander vom Web geladen werden. Es soll einstellbar sein, ob nur HTML-Dokumente der gleichen Domäne (der Startseite) geladen werden sollen, oder ob alle Links verfolgt werden sollen. Berücksichtigen Sie folgende Punkte:

- Laden Sie lediglich HTML-Dateien aus dem Web. Ignorieren Sie eingebundene Elemente wie Bilder, Videos oder Cascading Style Sheets.
- Informieren Sie sich über die Java-Klassen `URI` und `URL` sowie `HttpURLConnection`. Benutzen Sie die beiden erstgenannten, um relative Links in absolute umzuwandeln.
- Extrahieren Sie Links nicht nur aus HTML-Anchor-Elementen, sondern z.B. auch aus Frame-Sets.
- Benutzen Sie eine FIFO-Queue zur Vorhaltung von Links.
- Um die Anzahl von geladenen Web-Seiten zu beschränken, soll der Benutzer eine Maximalzahl spezifizieren können.

Aufgabe 3 : HTML-Validierung

Laden Sie die Datei `non-valid-html.html` von der Homepage der Vorlesung.

(a) Korrigieren Sie mit möglichst wenigen Änderungen diese Datei bedeutungserhaltend, so dass ein valides HTML-4.01-Transitional-Dokument entsteht.

(b) Nehmen Sie das in (a) entstandene Dokument und verwandeln Sie es in ein valides HTML-4.01-Strict-Dokument.

(c) Besuchen Sie die Internetseite `www.abipur.de` und speichern Sie den Quellcode der Startseite. Prüfen Sie dann mit dem Validator auf `http://validator.de.selfhtml.org/`, ob die Internetseite ein valides HTML-4.0-Transitional-Dokument ist. Sollte dies nicht der Fall sein, ändern Sie den Quellcode solange ab, bis es sich um ein valides HTML-4.0-Transitional-Dokument handelt und dokumentieren Sie Ihre Änderungen.

Aufgabe 4 : CSS

(a) Wie ist eine Stylesheet-Regel aufgebaut?

(b) Erläutern Sie zwei Möglichkeiten, um Stylesheet-Information in eine HTML-Seite einzubetten.

Aufgabe 5 : Multiple Choice: Spezifität in CSS

Jede richtig beantwortete Frage gibt einen Punkt. Beachten Sie, dass zu einer Frage mehrere Antworten zutreffen können. Ein Frage gilt als richtig beantwortet, falls alle zutreffenden und keine unzutreffende Antwort angekreuzt ist. Falsch beantwortete Fragen führen zu einem Minuspunkt – jedoch kann die gesamte Summe der Punkte dieser Aufgabe nicht kleiner als 0 werden. Eine nicht beantwortete Frage (= kein Kreuz) wird mit 0 Punkten gewertet.

Gegeben ist folgendes HTML-Fragment:

```
 1   <body>
 2     <h1 class="heading">Hello, World</h1>
 3     <div>
 4       <span>Text in a span.</span>
 5     </div>
 6     <div id="div1">
 7       <p>
 8         <a href="http://1">Link 1</a>
 9       </p>
10       <a href="http://2">Link 2</a>
11       <br />
12       <a href="http://3">Link 3</a>
13     </div>
14     <h1 class="heading">Hello, Planet</h1>
15     <div>
16       More text. With <a href="http://4">Link 4</a>.
17     </div>
18   </body>
```

Gegeben ist weiterhin folgendes CSS-Stylesheet:

```
/* Regel 1: */
* { color:green; background:black; font-family:monospace; }

/* Regel 2: */
#div1 :last-child { color:yellow; }

/* Regel 3: */
p ~ a { color:red; }

/* Regel 4: */
#div1 :first-child :first-child { background:yellow; }

/* Regel 5: */
#div1 :nth-child(1) :nth-child(1) { background:gray; }

/* Regel 6: */
.heading { background:yellow; }

/* Regel 7: */
div + h1 { background:lime; }

/* Regel 8: */
body :nth-child(2) :first-child { color:cyan; }

/* Regel 9: */
body > div :first-child { color:lime; }
```

Kreuzen Sie Zutreffendes an:

(a) Die Selektoren folgender CSS-Regeln treffen auf das `<span>`-Element in Zeile 4 zu:

- ☐ Regel 1
- ☐ Regel 2
- ☐ Regel 4
- ☐ Regel 8
- ☐ Regel 9

(b) Folgende Aussagen treffen zu:

- ☐ Regel 8 ist spezifischer als Regel 9.
- ☐ Regel 9 ist spezifischer als Regel 8.
- ☐ Keine der beiden Aussagen.

(c) Der Selektor von Regel 3 erfasst:

- ☐ Das `<p>`-Element in Zeile 7.
- ☐ Das Anchor-Element in Zeile 8.
- ☐ Das Anchor-Element in Zeile 10.
- ☐ Das Anchor-Element in Zeile 12.
- ☐ Das Anchor-Element in Zeile 16.

(d) Auf das Anchor-Element in Zeile 12 werden folgende CSS-Deklarationen angewandt:

- ☐ `color:red`
- ☐ `color:yellow`
- ☐ `background:gray`
- ☐ `font-family:monospace`

(e) Folgendes trifft zu:

- ☐ Regel 4 ist spezifischer als Regel 5.
- ☐ Regel 5 ist spezifischer als Regel 4.
- ☐ Keine der beiden Aussagen trifft zu.

Aufgabe 6 : HTML

(a) Worin unterscheidet sich die Browser-Darstellung von HTML-Block-Elementen gegenüber HTML-Inline-Elementen? Geben Sie eine CSS-Regel an um diesen Unterschied im Browser sichtbar zu machen.

(b) Wofür dienen die HTML-Elemente `<div>` und `<span>`?

Aufgabe 7 : HTML + CSS: Kurstabelle

(a) Entwerfen Sie ein HTML5-Dokument, das in einer Tabelle Hyperlinks auf die Vorlesungsskripte enthält. Orientieren Sie sich bei Ihrem Entwurf an der Abbildung unten. Die URLs zu den Skripten lassen sich von der Seite „Lecture Notes“, `www.uni-weimar.de/cms/medien/webis/teaching/lecture-notes.html`, extrahieren. Verwenden Sie keine Attribute wie `border` zur Gestaltung der Tabelle, sondern ein Cascading Style Sheet.

(b) Ergänzen Sie das Cascading Style Sheet um Regeln, die die Hintergrundfarbe einer Tabellenzelle ändern, sobald sich der Mauszeiger innerhalb der Zelle befindet. Sorgen Sie weiterhin dafür, dass möglichst die gesamte Zellengröße für den enthaltenen Link genutzt wird.

(c) Die Vorlesung verwendet auch Skripte, die nicht online verfügbar sind. Tragen Sie den Titel dieser Skripte ebenfalls in die Tabelle ein. Ergänzen Sie Ihr Stylesheet so, dass die Zellen von nicht verfügbaren Skripten ihre Hintergrundfarbe beim Eintritt des Mauszeigers nicht ändern. Weiterhin soll der Titel der nicht verfügbaren Skripte grau dargestellt werden.

Hilfen zu den Properties, die Sie in Cascading Style Sheet für Tabellen verwenden können, finden Sie unter http://de.selfhtml.org/css/eigenschaften/tabellen.htm.

Einführung	Rechnerkommunikation und Protokolle	Dokumentsprachen	Client-Technologien	Server-Technologien	Architekturen und Middleware-Technologien
Organisation, Literatur	Kommunikation und Protokolle für Web-Systeme (1)	Auszeichnungssprachen	JavaScript	CGI, Servlets, JSP	Überblick, Ajax, Einführung Web-Services
Überblick: Web-Systeme und Web-Technologien	Kommunikation und Protokolle für Web-Systeme (2)	HTML, CSS	Java Applets	Reguläre Ausdrücke, PHP	
	Kommunikation und Protokolle für Web-Systeme (3)	XML-Grundlagen			
		XML-Schema			
		Die XSL-Familie			
		APIs für XML			

Aufgabe 8 : Programmieraufgabe: HTML-Nachrichtenliste [P]

Entwickeln Sie eine HTML-Seite zum Anzeigen von Nachrichten, die beispielsweise aus einem Newsfeed stammen könnten. Gehen Sie in folgenden Schritten vor, um die HTML-Seite zu entwickeln:

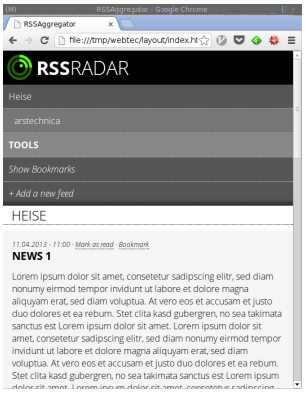
- Bauen Sie ein zweispaltiges Layout mit Kopf- und Fußbereich. Nutzen Sie dabei die mit HTML5 neu eingeführten Elemente wie `header`, `footer`, `article`, etc. Das Layout soll valides HTML5 sein. Nutzen Sie zum Prüfen z.B. den Validator des W3C.
- Informieren Sie sich über die Funktionsweise der Größeneinheit `em`. Nutzen Sie lediglich `em` als Größenangabe innerhalb ihres CSS.
- Machen Sie Ihr Layout „responsive“. Ab einer geeigneten Breite (z.B. 480px für mobile Geräte) soll das Layout nur noch einspaltig dargestellt werden. Nutzen Sie dafür `media`-Queries. Verwenden Sie die CSS3 `column`-Eigenschaft für die Darstellung von Text.

Optional Informieren Sie sich über Webfonts und das Web Open Font Format. Binden Sie in ihrem Layout einen Webfont aus der Kollektion der Google Web Fonts `www.google.com/fonts/` ein.

Hinweise:

[P] Diese Übungsaufgabe ist Teil eines Programmierprojektes, das dazu dient, das Zusammenspiel verschiedener Webtechnologien zu veranschaulichen. Zu diesem Zweck werden im Rahmen der Übungen verschiedene, aufeinander aufbauende Komponenten implementiert. Bis zum Ende des Semesters soll ein einfacher RSS-Reader mit HTML-Oberfläche entstehen.

- Orientieren Sie sich bei Bedarf an den nachfolgenden Screenshots für das zweispaltige und das einspaltige Layout.



Übungsblatt 3 WT:III

Bis zum 28.05.2013 sind Lösungen zu folgenden Aufgaben abzugeben: 4, 6, 7, 8.

Aufgabe 1 : XML

(a) Erstellen Sie für die Web-Veröffentlichung eines Artikels die Meta-Informationen nach dem Dublin-Core-Standard.

(b) Was sind Parameter-Entities? Konstruieren Sie eine Anwendung damit.

(c) Geben Sie ein Beispiel für ein wohlgeformtes, aber nicht valides XML-Dokument an.

Aufgabe 2 : XML/DTD

Gegeben sei die folgende DTD zur Repräsentation von Büchern.

```
<!ELEMENT book (bookTitle,chapter+,references)>
<!ELEMENT bookTitle (#PCDATA)>
<!ELEMENT chapter (heading,par*)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT par (#PCDATA)>
<!ELEMENT references (heading,par*)>
```

(a) Repräsentieren Sie als Baum die Elementstruktur von Dokumenten, die valide Instanzen der obigen DTD sind.

(b) Erstellen Sie ein valides Instanzdokument für die obige DTD.

Aufgabe 3 : XML/DTD

Sie haben den Auftrag, Dokumente einer bestimmten Klasse (einen Dokumenttyp) zu modellieren. Sollte Ihnen kein Dokumenttyp einfallen, dann modellieren Sie e-Learning-Dokumente.

(a) Erstellen Sie eine Auszeichnungssprache für diesen Dokumenttyp. Lassen Sie sich mindestens 5 verschiedene Auszeichnungen einfallen.

(b) Geben Sie eine DTD für Ihre Auszeichnungssprache an.

(c) Geben Sie ein XML-Dokument an, das die Benutzung der von Ihnen gewählten Dokumentklasse demonstriert.

(d) Validieren Sie das XML-Dokument aus der letzten Teilaufgabe.

Aufgabe 4 : DTDs und XML Schema

Gegeben sei die folgende Klausur-DTD:

```
<!ELEMENT written_exam (head, body)>
<!ELEMENT head (title, student)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT student (#PCDATA)>
<!ELEMENT body (exercise+)>
<!ELEMENT exercise (description, question+)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT question (#PCDATA)>
<!ATTLIST exercise number CDATA #REQUIRED
                   category (verbose | multiple_choice) "verbose">
```

(a) Erstellen Sie ein wohlgeformtes XML-Dokument, das bzgl. der Klausur-DTD *nicht* valide ist. Benutzen Sie den folgenden Rahmen.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE written_exam SYSTEM "/home/xml/written_exam.dtd">
<written_exam>









</written_exam>
```

(b) Welche Inhaltsmodelle kommen in der DTD vor? Geben Sie die entsprechen Arten mit je einem Beispiel an.

(c) Setzen Sie das `body`-Element der Klausur-DTD in XML Schema um. Hinweis: Das Attribut `category` soll vom Typ String sein und als Default den Wert „verbose“ erhalten. Benutzen Sie den folgenden Rahmen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" ...>
  ...
    <xsd:element name="body">









    </xsd:element>
  ...
```

Aufgabe 5 : XML-Schema

Peter möchte seine CDs organisieren. Er kauft sich dazu ein CD-Regalsystem, das aus 3 bis 10 Fächern besteht. Jedes Fach hat eine Nummer und kann bis zu 30 CDs aufnehmen. Die eingeordneten CDs werden durch Interpret und Titel identifiziert.

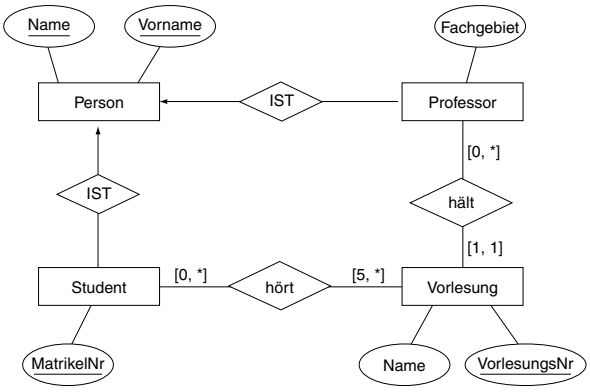
(a) Geben Sie ein XML-Schema an, das den Sachverhalt formalisiert.

(b) Geben Sie ein XML-Instanzdokument an, das gegen Ihr XML-Schema validiert.

Aufgabe 6 : XML Schema

Setzen Sie das abgebildete ER-Diagramm für Vorlesungen in XML um.

(a) Erstellen Sie eine XML-Schema-Datei zum ER-Diagramm.



(b) Erstellen Sie ein XML-Instanz-Dokument und validieren Sie es.

(c) Erläutern Sie, welches im ER-Diagramm benutzte Konzept nur unzureichend mit DTDs umsetzbar ist.

Aufgabe 7 : XML-Schema

Gegeben sei ein XML-Dokument, das die Leistungen einer Werkstatt zu einem PKW für einen Kunden zusammenfasst:

```
<ServiceListe FahrgestellNR="1234554356-34" KundenNR="M-1998-023">
  <Service Auftragsnummer="2005-1567">
    <Datum>2005-04-10</Datum>
    <KM>9808</KM>
    <Anlass>Inspektion</Anlass>
    <Arbeiten>Kontrolle und Ölwechsel durchgeführt</Arbeiten>
  </Service>

  <Service Auftragsnummer="2006-2745">
    <Datum>2006-05-06</Datum>
    <KM>25823</KM>
    <Anlass>Inspektion</Anlass>
    <Arbeiten>Kontrolle; defektes Bremslicht (HL)</Arbeiten>
  </Service>

  <Service Auftragsnummer="2008-0641">
    <Datum>2008-02-03</Datum>
    <KM>43076</KM>
    <Anlass>Rückruf</Anlass>
    <Arbeiten>Bremsschläuche ausgetauscht</Arbeiten>
  </Service>
</ServiceListe>
```

Erstellen Sie ein XML-Schema zur Beschreibung von Dokumenten dieser Art.

(a) Fügen Sie einen Zielnamensraum ein und fügen Sie in Ihr Instanzdokument die entsprechenden Zeilen zur Verwendung dieses Namensraumes und eines Ihrer Schema-Dokumente ein.

(b) Verwenden Sie zunächst das Russian Doll Prinzip für die Modellierung.

(c) Verwenden Sie in einer zweiten Version globale Elementdefinitionen. Welche Auswirkungen hat dies auf die Attribute `elementFormDefault="qualified"` und `attributeFormDefault="qualified"` bzw. die Instanzdokumente?

(d) Verwenden Sie benannte Datentypen.

Validieren Sie die Instanzdokumente gegen Ihre XML-Schema Versionen.

Aufgabe 8 : XML-Schema Namespace

Gegeben sei ein XML-Schema für Notizen:

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.upb.de/cs/webis2009"
            elementFormDefault="qualified">
  <xsd:element name="note">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="to" type="xsd:string"/>
        <xsd:element name="from" type="xsd:string" />
        <xsd:element name="heading" type="xsd:string"/>
        <xsd:element name="body" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Das folgende XML-Dokument beschreibt eine Notiz gemäß des obigen XML-Schemas.

```
<?xml version="1.0"?>
<a:note xmlns:a="http://www.upb.de/cs/webis2009"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.upb.de/cs/webis2009 note.xsd">
  <a:to>Tove</a:to>
  <a:from>Jani</a:from>
  <a:heading>Reminder</a:heading>
  <a:body>Do not forget me this weekend!</a:body>
</a:note>
```

(a) Wie ist das Instanzdokument zu ändern, wenn es mit den nachfolgenden Zeilen beginnt?

```
<?xml version="1.0"?>
<note xmlns="http://www.upb.de/cs/webis2009"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.upb.de/cs/webis2009 note.xsd">
...
```

(b) Wie ist das Instanzdokument zu ändern, wenn das XML-Schema mit den nachfolgenden Zeilen beginnt?

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.upb.de/cs/webis2009"
            elementFormDefault="unqualified">
...
```

Funktioniert in diesem Fall noch das Überschreiben des Default-Namespace?

Hinweis: Auf http://tools.decisionsoft.com/schemaValidate/ können Sie Instanzdokumente gegen ein XML-Schema validieren.

Aufgabe 9 : MathML

Die Markup-Sprache MathML dient der Darstellung mathematischer Ausdrücke in Web-Dokumenten.

(a) Bei den in MathML definierten Elementen werden zwei Kategorien unterschieden: Präsentationselemente und Inhaltselemente. Worin besteht der Unterschied?

(b) Entwerfen Sie eine XHTML-Seite mit dem Titel „Kosinusähnlichkeit“ und betten Sie die unten dargestellte Formel zur Berechnung der Kosinusähnlichkeit mit Hilfe von MathML in das XHTML-Dokument ein.

$$\cos(\varphi) = \frac{\mathsf{a}^T\mathsf{b}}{||\mathsf{a}|| \cdot ||\mathsf{b}||}$$

Validieren sie Ihre Lösung mit einem der in der Vorlesung genannten Werkzeuge.