Examen Web/XML

Durée : 2 heures, Documents autorisés

QCM (2,5 points)

Une seule réponse par question. 0,5 point par bonne réponse. –0,25 par mauvaise réponse. Vous pouvez répondre directement sur la feuille.

A - Que fait un parseur lorsqu'il rencontre une déclaration d'espace de nommage ?

- ☐ Il en parle à son parfrère
- ☐ Il vérifie que l'URI existe bien
- ☑ Rien
- ☐ Il charge la DTD trouvée à l'adresse spécifiée par l'URI
- ☐ Il vérifie la syntaxe de l'URI

B - Quelle techno est la plus efficace pour transformer en HTML une toute petite partie des données d'un document XML très volumineux ?

- ☐ DOM
- ☑ SAX
- ☐ XSLT
- ☐ Regexp avec Perl

C – Que contient le paramètre test
`<xsl:variable name="test" select="/..">` ?

- ☑ Un ensemble de nœuds vide
- ☐ Tous les nœuds du document XML en entrée
- ☐ L'élément racine du document XML en entrée
- ☐ La racine du document XML en entrée

D – Dans le DOM, combien de nœuds **`Text`** consécutifs peut on trouver ?

- ☐ Aucun car ils sont tous transformés en nœuds **`CDATASection`**
- ☐ Un seul puisque les informations textuelles adjacentes sont fusionnées
- ☐ Plusieurs s'il y a des appels d'entités ou des sections CDATA ou des appels de caractères dans le texte
- ☑ Plusieurs car on peut dissocier les nœuds **`Text`** par programmation

E – Laquelle de ces grammaire XML ne se préoccupe pas de la mise en forme ?

- ☐ SVG
- ☐ XSLFO
- ☑ Docbook
- ☐ XHTML

Opérations sur les ensembles avec XPath (1,5 points)

Il n'y a pas d'opérateurs ensemblistes dans XPath, en revanche, on peut facilement les obtenir dans XSLT.
A quelle opération ensembliste correspond chacune des expressions suivantes ?

- `<xsl:copy-of select="$node-set1[count(. | $node-set2) = count($node-set2)]"/>`

L'union (ce qui est dans `$node-set1` ou dans `$node-set2`)

- `<xsl:copy-of select="$node-set1 | $node-set2"/>`

L'intersection (ce qui est à la fois dans `$node-set1` et dans `$node-set2`)

Examen Web/XML

Durée : 2 heures, Documents autorisés

- `<xsl:copy-of`
  `select="$node-set1[count(. | $node-set2) != count($node-set2)] |`
  `$node-set2[count(. | $node-set1) != count($node-set1)]"/>`

La différence symétrique (ce qui est dans `$node-set1` ou dans `$node-set2`, mais pas dans les 2 à la fois)

Problème XSLT (2 points)

La fonction XSLT **`document( )`** permet de charger un document XML dans une feuille de style XSLT. Elle retourne un **`node-set`** de ce document.
Lorsque le premier argument est une chaîne vide : **`document('')`** le document XML chargé est la feuille de style elle-même. C'est très pratique pour définir des données structurées constantes, et pour y accéder : puisqu'un document XSLT est aussi un document XML, on peut utiliser XPath pour y accéder.
A partir du code suivant, écrire les expressions qui permettent de transformer un attribut **`date`** avec le nom du mois en clair (**24 février 2003** au lieu de **24-2-2003**).
On utilisera les fonctions XPath **`substring-before( )`** et **`substring-after( )`** pour extraire le jour le mois et l'année de la date.
On pourra utiliser des variables intermédiaires pour obtenir la valeur de la variable **`nom-mois`**.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:date="*** Pour obtenir le nom des mois ***">

    <xsl:output method="html" />

    <date:noms-mois>
        <date:mois court="jan">janvier</date:mois>
        <date:mois court="fév">février</date:mois>
        <!-- etc -->
        <date:mois court="déc">décembre</date:mois>
    </date:noms-mois>

    <xsl:template match="@date">
        <!--date au format j-m-aaaa -->
        <xsl:variable name="jour" select="substring-before(., '-')"/>
        <xsl:variable name="mois"
          select="substring-before(substring-after(., '-'), '-')"/>
        <xsl:variable name="année"
          select="substring-after(substring-after(., '-'), '-')"/>
        <xsl:variable name="nom-mois"
          select="document('')/*/date:noms-mois/date:mois[number($mois)]"/>

        <xsl:text/>
        le <xsl:value-of select="$jour" />
        <xsl:text> </xsl:text>
        <xsl:value-of select="$nom-mois" />
        <xsl:text> </xsl:text>
        <xsl:value-of select="$année" />
    </xsl:template>

    <!-- etc -->

</xsl:stylesheet>
```

Examen Web/XML

Durée : 2 heures, Documents autorisés

Exemple de donnée du document XML en entrée (on considère que la règle XSLT ci-dessus est appelée) :

```
<truc date="24-2-2003"/>
```

Exemple de résultat produit :

```
Le 24 février 2003
```

Syntaxe XML (4 points)

L'auteur du document XML suivant a cru pouvoir rédiger son document en se passant d'un éditeur XML. Aidez le à corriger les **16** erreurs qu'il contient, pour qu'il puisse être parsé.

Répondez sur la feuille : entourez chaque erreur et annotez la.

```
<?xml version="1.0" encoding="UTF-8"❶ standalone="standalone"❷?>
<?xml-stylesheet type="text/xsl" href="fg.xsl"?>
<!-- à relire -->
<?robots index="yes" follow="no"?>
<document>
    <titre style="bold" ❸style="big">Mes mémoires</titre>
    <auteur>
        <nom>Gump</nom>
        <prénom>Forrest</prénom>
    <auteur>❹
    <description xmlns="http://www.w3.org/1999/xhtml"
        style="bold" Style="big" xml:space="preserve">
        <!-- à partir d'ici, on peut utiliser
        des éléments HTML <!-- ❺ et d'autres aussi -->
        c'est pratique pour du contenu
        documentaire -->
        <p xml:space="default" align=center❻>Ma maman disait
            toujours : &#xA; "<i>la vie c'est comme une
            bo&icirc;te❼ de <b>chocolat❽</i>, on ne sait jamais
            sur quoi on va tomber</b>".</p>
        <hr width❾/>
        <a  xlink:href="bubbagump.avi"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            href="bubbagump.avi"
            xlink:role="mon film">
            <object xmlns:xlink=""❶❿
                    xlink:href="bubbagump.avi"/>
        </a>
        <script language="JavaScript">
            <![CDATA[
                function check() {
                    for (int i=10; i>0; i++) {
                        if ( a[b[i]]>❶❶5 ) break;
                    }
                }
            ]]>
            <!-- vérifier si la boucle doit s'écrire
            avec i--❶❷ et pas i++ -->
        </script>
        <p>La suite, je ne m'en souviens plus... ❶❸
    </description>
    <xml:parse❶❹ processor="JAXP"/>
```

Examen Web/XML

Durée : 2 heures, Documents autorisés

```
</document>
<remarques>❶❺
    Mon document ne parse pas
</remarques>
<!-- il y a quelques erreurs -->
fin du document❶❻
```

Une correction du document précédent :

```
<?xml version="1.0" encoding="iso-8859-1"❶ standalone="yes"❷?>
<?xml-stylesheet type="text/xsl" href="fg.xsl"?>
<!-- à relire -->
<?robots index="yes" follow="no"?>
<document>
    <titre style="bold" ❸Style="big">Mes mémoires</titre>
    <auteur>
        <nom>Gump</nom>
        <prénom>Forrest</prénom>
    </auteur>❹
    <description xmlns="http://www.w3.org/1999/xhtml"
        style="bold" Style="big" xml:space="preserve">
        <!-- à partir d'ici, on peut utiliser
        des éléments HTML -❺ et d'autres aussi -
        c'est pratique pour du contenu
        documentaire -->
        <p xml:space="default" align='center'❻>Ma maman disait
            toujours : &#xA; "<i>la vie c'est comme une
            boîte❼ de <b>chocolat</b>❽, on ne sait jamais
            sur quoi on va tomber</i>".</p>
        <hr width="1"❾/>
        <a  xlink:href="bubbagump.avi"
            xmlns:xlink="http://www.w3.org/1999/xlink"
            href="bubbagump.avi"
            xlink:role="mon film">
            <object ❶❿
                    xlink:href="bubbagump.avi"/>
        </a>
        <script language="JavaScript">
        <![CDATA[
            function check() {
                for (int i=10; i>0; i++) {
                    if ( a[b[i] ]>❶❶5 ) break;
                }
            }
        ]]>
        <!-- vérifier si la boucle doit s'écrire
        avec i- -❶❷ et pas i++ -->
        </script>
        <p>La suite, je ne m'en souviens plus...</p>❶❸
    </description>
    <parse❶❹ processor="JAXP"/>
    <remarques>❶❺
        Mon document ne parse pas
    </remarques>
</document>
<!-- il y a quelques erreurs -->
<!-- fin du document -->❶❻
```

Examen Web/XML

Durée : 2 heures, Documents autorisés

❶ L'encodage spécifié ne correspond pas au jeu de caractère utilisé : les accents feront échouer l'analyse. Utiliser iso-8859-1 au lieu de UTF-8.
❷ Mauvaise valeur pour standalone. On corrige avec « yes ».
❸ Attributs répétés dans l'élément titre. Il faut en changer un, par exemple en mettant une majuscule.
❹ <auteur> au lieu de </auteur>.
❺ Des commentaires apparaissent dans des commentaires, ce qui est interdit.
❻ Attribut avec valeur non quotée <p align=center>. On ajoute les quotes.
❼ Entité non définie î On utilise le caractère î ou on peut définir l'entité.
❽ Balises b et i non rapprochées. On interverti.
❾ Attribut sans valeur <hr width/>. On met une valeur.
❶❿ Préfixe d'espace de nom assigné à l'espace de noms par défaut xmlns:xlink="". On supprime la déclaration puisqu'elle n'est pas utilisée.
❶❶ Il y a]]> dans CDATA, ce qui peut être interprété comme la fin de section CDATA. On ajoute un blanc.
❶❷ Il y a -- dans un commentaire, ce qui est interdit. On ajoute un blanc.
❶❸ Balise non fermée <p>. On ferme.
❶❹ Préfixe d'élément qui commence par xml : xml:parse. On renomme l'élément.
❶❺ Plusieurs éléments racine <document> et <remarques>. On déplace les remarques dans le corps du document.
❶❻ Texte hors contenu "fin du document". On met en commentaire, ou on déplace le texte dans le corps du document.

Remarques

Voici quelques remarques complémentaires :

QCM

Le QCM est destiné à valider votre compréhension et vos connaissances générales.
A – Vu dans le cours :

> « Les URI sont purement déclaratives
> Il n'y a pas nécessairement quelque chose à l'adresse indiquée
> Les parseurs et les applications ne sont pas tenus d'aller y chercher quoi que ce soit :
> - il n'y a rien qui puisse les intéresser
> - la cible n'est pas nécessairement accessible par votre réseau »

D'ailleurs, la question sur XSLT aurait pu vous aider : la déclaration d'espace de nommage `xmlns:date="*** Pour obtenir le nom des mois ***"` est parfaitement valide, bien qu'inhabituelle.

B – Vu dans le cours :

> «DOM : Toutes les données doivent être utilisées
> SAX : Seule une partie des données doivent être utilisées »

Les expressions régulières de Perl ne permettent pas d'accéder au modèle de données de XML. Par exemple comment trouver « toto » dans « `to<![CDATA[to]]>` » ?

C – Petit exercice de déduction :

> / permet d'accéder à la racine
> .. permet de remonter d'un niveau
> Il n'y a rien au dessus de la racine
> Donc /.. ne donne rien

Examen Web/XML

Durée : 2 heures, Documents autorisés

D – Vu dans le cours :

« `Text.splitText(offset);` »

Lorsqu'un document est analysé, on ne trouve pas 2 nœuds textes consécutifs dans le DOM, mais rien n'empêche d'en créer, par scission ou ajout.

E – Par élimination :

SVG permet de faire des dessins
XSLFO permet de faire de la mise en page (c'est de la PAO)
XHTML permet de formater des pages HTML

XPath

La question sur XPath permet d'évaluer votre assimilation de XPath et du modèle de données.
Rappel : comment interpréter une expression XPath ?

- on évalue chaque étape de localisation
- une étape de localisation s'évalue dans le contexte du résultat de la précédente
- un prédicat s'évalue pour chaque nœud obtenu dans l'étape de localisation concernée

Pour évaluer :

```
<xsl:copy-of select="$node-set1[count(. | $node-set2) =
                                count($node-set2)]"/>
```

(Vu dans le cours :

« Dans un node-set, les nœuds sont uniques »)

Donc :

le nombre de nœuds donnés par `count(. | $node-set2)` est égal au nombre de nœuds donnés par `count($node-set2)` seulement pour les nœuds de `$node-set1` qui sont aussi dans `$node-set2`. On ne retient dans `$node-set1` que ces nœuds là.

XSLT

La question sur XSLT permet d'évaluer votre compréhension de XSLT sur des aspects à priori déroutants (non vus en cours) mais avec tous les éléments pour le réussir.
Vous deviez avoir avec vous la documentation de `substring-before()` et `substring-after()`, juste pour les aspects syntaxiques (qui ne prennent aucune part dans la notation, ça n'a rien d'intéressant). La documentation utile de la fonction `document()` était incluse dans l'énoncé.

```
<xsl:variable name="jour" select="substring-before(., '-')"/>
<xsl:variable name="mois"
  select="substring-before(substring-after(., '-'), '-')"/>
<xsl:variable name="année"
  select="substring-after(substring-after(., '-'), '-')"/>
<xsl:variable name="nom-mois"
  select="document('')/*/date:noms-mois/date:mois[number($mois)]"/>
```

Sur le nœud contextuel `@date="24-2-2003"`, `substring-before(., '-')` retourne « `24` ».
Pour la suite, `substring-after(., '-')` retourne « `2-2003` », dont on extrait le mois « `2` », et l'année « `2003` ».
Ensuite, on récupère le document XSLT en tant que document XML avec `document('')`
L'élément racine de ce document est `xsl:stylesheet`
Dans lequel on trouve un élément `date:noms-mois`
Dans lequel on trouve les éléments `date:mois`

Examen Web/XML

Durée : 2 heures, Documents autorisés

Celui qui nous intéresse est celui dont la position correspond au numéro du mois.

Syntaxe XML

Le but de cet exercice est de trouver les erreurs, la difficulté étant d'en trouver là où il n'y en a pas, et de passer à côté des vraies erreurs.