Exonet N°80

Description

Propriétés	Description
Intitulé long	Analyse d'un document XML et de sa transformation XSLT vers HTML.
Date de publication	20 Septembre 2004, révisé en novembre 2006
Public concerné	BTS Services informatiques aux organisations
Matière	SI1 - Support système des accès utilisateurs
Compétences	
Savoirs	Format d'échange des données
Transversalité	
Objectifs	Analyser et modifier le code qui produit un document électronique
Outils	Éventuellement un environnement de développement XML et XSLT comme http://www.xmlcooktop.com/
Pré-requis	Le langage XML, les DTD (voir Exonet n°57)
Mots-clés	XML, XSLT
Auteur(es)	Eric Deschaintre, merci à Olivier Korn pour sa relecture

Énoncé

Un logiciel de messagerie instantanée permet à des utilisateurs présents dans le monde entier d'avoir des conversations écrites par échange synchrone de messages (*chat*). Chaque utilisateur possède une adresse électronique (*Logon name*) qui l'identifie de façon unique dans le système ainsi qu'un surnom (*Friendly name*) qui lui permet d'être reconnu des autres utilisateurs. Les conversations peuvent réunir deux interlocuteurs ou plus.

Le logiciel permet à chaque utilisateur de garder une trace de ses conversations, celles-ci sont enregistrées dans un fichier texte au format XML. Dès qu'une fenêtre de conversation est ouverte, une nouvelle session de dialogue débute.

Un même fichier regroupe toutes les sessions de dialogue (conversations) ayant eu lieu avec un interlocuteur donné. Pour un utilisateur donné, il existe donc autant de fichiers de conversation que d'interlocuteurs.

A tout moment un utilisateur peut choisir de relire les dialogues qu'il a eus avec un interlocuteur. Pour cela, il suffit qu'il sélectionne l'adresse électronique de l'interlocuteur dans une liste, le logiciel affiche alors l'intégralité des échanges avec cette personne, session par session (par défaut de la plus ancienne session à la plus récente), puis message par message.

L'affichage des dialogues est obtenu par transformation du fichier source XML vers du code HTML affichable dans un navigateur. Cette transformation est définie dans un script XSL.

L'exercice consiste à :

- Analyser la structure du fichier XML qui contient le texte des conversations (voir annexe 1)
- Étudier le script XSLT de transformation du code XML vers HTML (voir annexe 2 et 3)
- En déduire la présentation obtenue, puis, adapter cette présentation pour qu'elle réponde à de nouveaux besoins

Questions

Analyse du document XML

Après avoir analysé le code source XML (annexe 1), répondre aux questions suivantes :

1. Combien de sessions de dialogue sont contenues dans l'extrait ?
2. Quels sont les surnoms des interlocuteurs ?
3. Combien de temps au total ont duré les conversations ?
4. Représenter la structure du document sous la forme d'une arborescence.
5. Écrire une DTD (*Document Type Definition*) susceptible de définir la structure d'un document XML valide pour décrire une conversation.

Analyse du script XSLT

Après avoir analysé le code du script XSLT :

6. Rechercher sur le web la signification des balises XSLT présentes dans le tableau ci-dessous.

Balise XSLT	Signification
<xsl:stylesheet>	
<xsl:variable>	
<xsl:template match="*pattern* ">	
<xsl:value-of select="*pattern*">	
<xsl:if test="*condition*">	
<xsl:choose>	
<xsl:apply-templates>	
<xsl:call-template>	

Sources possibles :
http://www.laltruiste.com/document.php?compteur=1&page=1&rep=5
http://www.w3schools.com/xsl/xsl_w3celementref.asp

7. Indiquer quelles informations supplémentaires sont prises en charge dans le mode « Debug » (valeur 'debug = 1')

8. Identifier la partie du script qui permet de colorer de façon alternée les conversations (sessions de dialogue), modifier le code de façon à inverser le rythme des couleurs.

9. Ajouter un paramètre *noTimeStamp* de façon à pouvoir ne pas afficher les date et heure de chaque message

10. Produire une représentation schématique de la page *web* qui doit résulter de la transformation XSLT

Annexe 1 – Extrait du code source XML d'une conversation

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type='text/xsl' href='MessageLog.xsl'?>
<Log LogonName="axel.aissete@wanadoo.fr" FirstSessionID="1" LastSessionID="2">
   <Message Date="05/09/2004" Time="18:20:16" DateTime="2004-09-
05T16:20:16.640Z" SessionID="1">
      <From>
         <User LogonName="gordon.freeman@free.fr" FriendlyName="ed" />
      </From>
      <To>
         <User LogonName="axel.aissete@wanadoo.fr" FriendlyName="viveXML" />
      </To>
      <Text Style="font-family:Microsoft Sans Serif; color:#000000; ">Hello
!</Text>
   </Message>
   <Message Date="05/09/2004" Time="18:20:43" DateTime="2004-09-
05T16:20:43.906Z" SessionID="1">
      <From>
         <User LogonName="axel.aissete@wanadoo.fr" FriendlyName="viveXML" />
      </From>
      <To>
         <User LogonName="gordon.freeman@free.fr" FriendlyName="ed" />
      </To>
      <Text Style="font-family:Viner Hand ITC; font-weight:bold; color:#800080;
">Bonjour :)</Text>
   </Message>
   <Message Date="05/09/2004" Time="18:22:45" DateTime="2004-09-
05T16:22:45.031Z" SessionID="2">
      <From>
         <User LogonName="gordon.freeman@free.fr" FriendlyName="ed" />
      </From>
      <To>
         <User LogonName="axel.aissete@wanadoo.fr" FriendlyName="viveXML" />
      </To>
      <Text Style="font-family:Microsoft Sans Serif; color:#000000; ">re-
bonjour</Text>
   </Message>
   <Message Date="05/09/2004" Time="18:22:58" DateTime="2004-09-
05T16:22:58.609Z" SessionID="2">
      <From>
         <User LogonName="axel.aissete@wanadoo.fr" FriendlyName="viveXML" />
      </From>
      <To>
         <User LogonName="gordon.freeman@free.fr" FriendlyName="ed" />
      </To>
```

```
         <Text Style="font-family:Viner Hand ITC; font-weight:bold; color:#800080;
    ">c un bon exemple ?</Text>
      </Message>
      <Message Date="05/09/2004" Time="18:23:08" DateTime="2004-09-
    05T16:23:08.109Z" SessionID="2">
         <From>
            <User LogonName="gordon.freeman@free.fr" FriendlyName="ed" />
         </From>
         <To>
            <User LogonName="axel.aissete@wanadoo.fr" FriendlyName="viveXML" />
         </To>
         <Text Style="font-family:Microsoft Sans Serif; color:#000000; ">Oui,
    merci pour ton aide</Text>
      </Message>
</Log>
```

Annexe 2 – Code source XSLT

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<!-- localized strings -->
<xsl:variable name='ColumnHeader_Date'>Date</xsl:variable>
<xsl:variable name='ColumnHeader_Time'>Heure</xsl:variable>
<xsl:variable name='ColumnHeader_From'>De</xsl:variable>
<xsl:variable name='ColumnHeader_To'>À</xsl:variable>
<xsl:variable name='ColumnHeader_Message'>Message</xsl:variable>

<!-- variables -->
<xsl:variable name='Debug'>0</xsl:variable>

<xsl:variable name='TableStyle'>font-family:Verdana; font-size:67%; text-align:left; vertical-align:top;
table-layout:fixed</xsl:variable>
<xsl:variable name='GutterStyle'>width:2ex</xsl:variable>
<xsl:variable name='HeaderStyle'>border-bottom:1 solid black</xsl:variable>

<xsl:variable name='UseZebraStripe'>1</xsl:variable>
<xsl:variable name='ZebraStripeStyle'>background-color:#e0edff</xsl:variable>

<xsl:variable name='MostRecentSessionFirst'>0</xsl:variable>

<xsl:template match="Log">

   <html dir='ltr'>
   <head>
      <title>
         Message Log for <xsl:value-of select="@LogonName"/>
         <xsl:if test="$Debug = 1"> (Debug)</xsl:if>
      </title>

      <xsl:if test="$Debug = 1">
         <span style="font-family:trebuchet ms; font-size:120%">
            Debug Version
         </span>
         <hr/>
      </xsl:if>
   </head>

   <body style='margin:0'>

      <table id='BodyTable' style="{$TableStyle}" cellspacing='0'>

         <xsl:if test="$Debug = 1">
            <col style="vertical-align:top; width:5ex;"/>
            <col style='{$GutterStyle}' />
         </xsl:if>

         <col style="width:16ex;"/>
         <col style='{$GutterStyle}' />
         <col style="width:16ex;"/>
         <col style='{$GutterStyle}' />
         <col style="width:21ex;"/>
         <col style='{$GutterStyle}' />
         <col style="width:21ex;"/>
         <col style='{$GutterStyle}' />
         <col style="width:70ex;"/>
```

```
        <thead>
          <tr>
            <xsl:if test="$Debug = 1">
              <th style="{$HeaderStyle}">SID</th>
              <th/>
            </xsl:if>
            <th style="{$HeaderStyle}">
              <xsl:value-of select="$ColumnHeader_Date"/>
            </th>
            <th/>
            <th style="{$HeaderStyle}">
              <xsl:value-of select="$ColumnHeader_Time"/>
            </th>
            <th/>
            <th style="{$HeaderStyle}">
              <xsl:value-of select="$ColumnHeader_From"/>
            </th>
            <th/>
            <th style="{$HeaderStyle}">
              <xsl:value-of select="$ColumnHeader_To"/>
            </th>
            <th/>
            <th style="{$HeaderStyle}">
              <xsl:value-of select="$ColumnHeader_Message"/>
            </th>
          </tr>
        </thead>

        <tbody style='vertical-align:top'>
          <xsl:choose>

            <!-- newest session first -->
            <xsl:when test="$MostRecentSessionFirst = 1">
              <xsl:apply-templates>
                <xsl:sort select='@SessionID' order='descending' data-type='number'/>
                <xsl:sort select='@DateTime'  order='ascending'/>
              </xsl:apply-templates>
            </xsl:when>

            <!-- oldest session first -->
            <xsl:otherwise>
              <xsl:apply-templates>
                <xsl:sort select='@SessionID' order='ascending' data-type='number'/>
                <xsl:sort select='@DateTime'  order='ascending'/>
              </xsl:apply-templates>
            </xsl:otherwise>

          </xsl:choose>
        </tbody>
      </table>
    </body>
    </html>

</xsl:template>

<xsl:template match="Message">
    <tr>
      <xsl:call-template name="CommonMessageProcessing" />

      <td> <xsl:apply-templates select="From/User"/> </td>
      <td/>
      <td> <xsl:apply-templates select="To/User"/> </td>
```

```
      <td/>
      <td>
         <span>
            <xsl:attribute name="style">
               <xsl:value-of select="Text/@Style"/>
            </xsl:attribute>
            <xsl:value-of select="Text"/>
         </span>
      </td>
   </tr>
</xsl:template>

<xsl:template match="User">
   <xsl:value-of select="@FriendlyName"/>
</xsl:template>

<xsl:template name="CommonMessageProcessing">
   <!-- zebra-stripe the sessions -->
   <xsl:if test="$UseZebraStripe = 1">
      <xsl:if test="(@SessionID mod 2) = 1">
         <xsl:attribute name="style">
            <xsl:value-of select="$ZebraStripeStyle"/>
         </xsl:attribute>
      </xsl:if>
   </xsl:if>

   <xsl:if test="$Debug = 1">
      <td> <xsl:value-of select="@SessionID"/> </td>
      <td/>
   </xsl:if>

   <td> <xsl:value-of select="@Date"/> </td>
   <td/>
   <td> <xsl:value-of select="@Time"/> </td>
   <td/>
</xsl:template>


</xsl:stylesheet>
```

Annexe 3 – Principe de la transformation XSLT sur un exemple

Principe de la transformation XSLT

Le document **XSL** permet d'associer les balises d'un document **XML** à du code **HTML** dans le but de présenter l'information contenue dans le document XML. Les balises XSL sont associées à l'espace de noms « xsl: » On y trouve aussi des éléments de langage **Xpath** qui permet de définir la position des noeuds en indiquant leur chemin dans l'arborescence XML. Ici, des modèles d'avions sont présentés dans un tableau HTML. Pour chaque modèle d'avion (**encadré jaune**) on affiche son nom en gras (balise HTML <b>). Ensuite, pour chaque caractéristique (**encadré bleu**), on ajoute une ligne (<tr>) au tableau, en affichant d'abord le nom de la caractéristique (en clair dans le document XSL) dans une première colonne, puis, pour chaque modèle, la valeur de cette caractéristique dans une nouvelle colonne (<td>). Ainsi de suite pour toutes les caractéristiques.

Code XML contenant les informations sur des modèles d'avions

```
- <models>
-   <model>
-     <Name>123</Name>
-     <MaxTakeoffWeight>910000</MaxTakeoffWeight>
-     <MaxLandingWeight>652000</MaxLandingWeight>
-     <MaxZeroFuelWeight>555000</MaxZeroFuelWeight>
-     <EngineOfferings>GE, P+W, and R-R </EngineOfferings>
-     <FuelCapacity>60305</FuelCapacity>
-     <CruiseMach>0.855</CruiseMach>
-     <Passengers>416</Passengers>
-     <DesignRange>7500</DesignRange>
-     <LowerHoldVolume>4876</LowerHoldVolume>
-     <Pallets>4</Pallets>
-   </model>
-   <model>
        <Name>123xml</Name>
-     <MaxTakeoffWeight>1043000</MaxTakeoffWeight>
-     <MaxLandingWeight>685000</MaxLandingWeight>
-     ...
```

Code XSL qui présente les modèles d'avions dans un tableau

```
<xsl:template match="models">
<table width="100%">
<tr>
        <td>Principal Characteristics</td>
        <xsl:for-each select="model"><td bgcolor="808080">
            <b><xsl:value-of select="Name" /></b>
        </td>
        </xsl:for-each>
</tr>

<tr bgcolor="C0C0C0">
        <td>Max. Takeoff Weight</td>
        <xsl:for-each select="model">
            <td><xsl:value-of select="MaxTakeoffWeight" /></td>
        </xsl:for-each>
</tr>
<tr bgcolor="C0C0C0">
        <td>Max. Landing Weight</td>
...
```

Quelques balises XSLT

xsl:template : contient les transformations à appliquer quand le nœud spécifié par l'attribut match (ici "**models**") est rencontré dans le code XML.

xsl:for-each fait une boucle sur l'ensemble des nœuds "**model**" (il y en a 3) en exécutant à chaque boucle les instructions situées avant sa balise de fermeture </xsl:for-each>

xsl:value-of permet d'extraire du document XML le contenu de la balise sélectionnée par l'attribut **select** (ici "**Name**") c'est-à-dire, successivement, les noms des modèles d'avions : «123», «123xml» et «123xml Stretch».

Transformation

Code HTML résultant

```
<table width="100%">
<tr>
  <td>Principal Characteristics</td>
  <td bgcolor="808080"><b>123</b></td>
  <td bgcolor="808080"><b>123xml</b></td>
  <td bgcolor="808080"><b>123xml Stretch</b>
  </td>
</tr>

<tr bgcolor="C0C0C0">
  <td>Max. Takeoff Weight</td>
(...)
```

Affichage obtenu dans la page HTML

Principal Characteristics	123	123xml	123xml Stretch
Max. Takeoff Weight	910000	1043000	1043000
Max. Landing Weight	652000	685000	725000
Engines	GE, P+W, and R-R	EA and R-R	EA and R-R
Fuel Capacity	60305	72853	72853
Cruise Mach	0.855	0.86	0.86
Passengers	416	442	522

Corrigé

Analyse du document XML

Après avoir analysé le code source XML (annexe 1), répondre aux questions suivantes :

1. Combien de sessions de dialogue sont contenues dans l'extrait ?

L'attribut « LastSessionID » de la balise <Log> vaut 2. L'attribut « FirstSessionID » de la même balise vaut 1. On en déduit que 2 sessions sont contenues dans l'extrait.

2. Quels sont les surnoms des interlocuteurs ?

L'émetteur (balise <From>) a pour surnom (attribut FriendlyName) « ed ».
Le récepteur (balise <To>) a pour surnom « viveXML »

3. Combien de temps au total ont duré les conversations ?

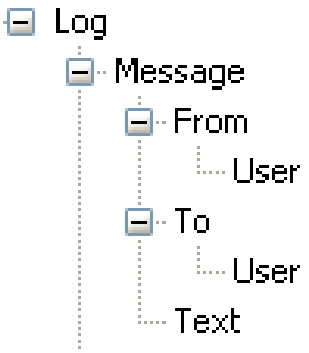
On obtient cette valeur en utilisant l'attribut Time des nœuds Message. Comme tous les messages ont été écvhangés le même jour, on peut utiliser la formule suivante :

Durée d'une conversation = [heure du dernier message] - [heure du premier message]
Il faut ensuite cumuler les temps obtenus pour toutes les conversations (attribut « SessionID" des nœuds <Message>).

Dans l'exemple on obtient :
Première session (SessionID="1") : [18:20:43] - [18:20:16] = 27 secondes
Deuxième session (SessionID="2") : [18:23:08] - [18:22:45] = 23 secondes

Soit un total de 50 secondes.

4. Représenter la structure du document sous la forme d'une arborescence.



5. Écrire une DTD susceptible de définir la structure d'un document XML valide pour décrire une conversation.

```
<!ELEMENT Log (Message*)>
  <!ATTLIST Log
    LogonName     CDATA #REQUIRED
    FirsSessionID CDATA #REQUIRED
    LastSessionID CDATA #REQUIRED
  >
<!ELEMENT Message (From, To, Text)>
  <!ATTLIST Message
    Date          CDATA #REQUIRED
    Time          CDATA #REQUIRED
    DateTime      CDATA #REQUIRED
    SessionID     CDATA #REQUIRED
  >
<!ELEMENT From    (User)>
<!ELEMENT To     (User)>
<!ELEMENT User    EMPTY>
  <!ATTLIST User
    LogonName     CDATA #REQUIRED
    FriendlyName  CDATA #IMPLIED
  >
<!ELEMENT Text    (#PCDATA)>
  <!ATTLIST Text
    Style         CDATA #IMPLIED
  >
```

Si la DTD est enregistrée sous le nom de fichier « Log.dtd », dans le même répertoire que le fichier XML, on peut l'associer à ce dernier en ajoutant une ligne de déclaration DOCTYPE sous le prologue <?xml version… ?>. Ce qui donnerait ceci :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Log SYSTEM "Log.dtd">
<?xml-stylesheet type='text/xsl' href='MessageLog.xsl'?>
```

On peut préférer une DTD intégrée au fichier XML. Dans ce cas, la déclaration DOCTYPE s'écrirait ainsi :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Log [
  <!ELEMENT Log (Message*)>
    <!ATTLIST Log
      LogonName     CDATA #REQUIRED
      FirsSessionID CDATA #REQUIRED
      LastSessionID CDATA #REQUIRED
    >
  <!ELEMENT Message (From, To, Text)>
    <!ATTLIST Message
      Date          CDATA #REQUIRED
      Time          CDATA #REQUIRED
      DateTime      CDATA #REQUIRED
      SessionID     CDATA #REQUIRED
    >
  <!ELEMENT From    (User)>
  <!ELEMENT To          (User)>
  <!ELEMENT User    EMPTY>
    <!ATTLIST User
      LogonName     CDATA #REQUIRED
      FriendlyName  CDATA #IMPLIED
    >
  <!ELEMENT Text    (#PCDATA)>
    <!ATTLIST Text
      Style         CDATA #IMPLIED
    >
]>
<?xml-stylesheet type='text/xsl' href='MessageLog.xsl'?>
```

Analyse du script XSLT

Après avoir analysé le code du script XSLT :

6. Rechercher sur le web la signification des balises XSLT présentes dans le script

Sources possibles :
http://www.laltruiste.com/document.php?compteur=1&page=1&rep=5
http://www.w3schools.com/xsl/xsl_w3celementref.asp

Balise XSLT	Signification
<xsl:stylesheet>	est l'élément racine des feuilles de style.
<xsl:variable>	permet de déclarer une variable dans une feuille de style.
<xsl:template match="pattern">	traite les noeuds sélectionnés par le modèle (*pattern*) spécifié par l'attribut « match »
<xsl:value-of select="pattern"/>	Permet d'extraire la valeur d'un nœud. Quand le modèle (*pattern*) est précédé du caractère « @ », il désigne un attribut du nœud ; précédé du caractère « $ » il désigne une variable.
<xsl:if test="condition">	Cette instruction évalue une condition. Si elle est vérifiée, le processeur XSL exécutera les instructions contenues à l'intérieur des marqueurs, sinon il les ignorera et passera aux instructions suivantes.
<xsl:choose>	L'élément <xsl:choose> combiné avec <xsl:when> et <xsl:otherwise>, permet de construire des tests conditionnels à l'instar des commandes *switch* de Java ou Javascript.
<xsl:apply-templates>	Permet d'appliquer les *templates* d'une feuille de style sur les fils du nœud courant et les nœuds textuels.
<xsl:call-template>	Permet d'appeler un *template* par son nom (attribut « name »).

7. Indiquer quelles informations supplémentaires sont prises en charge dans le mode Debug (valeur 'debug = 1')

Le texte « (debug) » n'est pas affiché dans le corps du document, il est simplement ajouté au titre du document HTML (ce que de nombreux navigateurs afficheront dans leur barre de titre).

Affichage du texte « (« Debug Version») par le script suivant :

```
<xsl:if test="$Debug = 1">
   <span style="font-family:trebuchet ms; font-size:120%">
      Debug Version
   </span>
   <hr/>
</xsl:if>
```

Ajout d'une colonne « SID » (numéro de session) par le script suivant :

```
<xsl:if test="$Debug = 1">
  <th style="{$HeaderStyle}">SID</th>
  <th/>
</xsl:if>
<xsl:if test="$Debug = 1">
   <td> <xsl:value-of select="@SessionID"/> </td>
   <td/>
</xsl:if>
```

8. **Identifier la partie du script qui permet de colorer de façon alternée les conversations (sessions de dialogue), modifier le code de façon à inverser le rythme des couleurs.**

```
<xsl:if test="$UseZebraStripe = 1">
   <xsl:if test="(@SessionID mod 2) = 1"> /* N°de session modulo 2 = alternativement 0 et 1*/
      <xsl:attribute name="style">
         <xsl:value-of select="$ZebraStripeStyle"/>
      </xsl:attribute>
   </xsl:if>
</xsl:if>
```

Il suffit de remplacer 1 par 0 dans le test : « ="(@SessionID mod 2) = 1"> »

9. **Ajouter un paramètre *noTimeStamp* de façon à pouvoir ne pas afficher les date et heure de chaque message**

Dans les lignes de déclaration de variables du script, ajouter la ligne (valeur = 1) :
<xsl:variable name='noTimeStamp'>1</xsl:variable>

Puis d'entourer d'une balise <xsl :if test> le code existant destiné à afficher les entêtes de colonnes et les lignes qui contiennent la date et l'heure des messages.

***Structure de la table** :*

```
<xsl:if test="$noTimeStamp = 0">
        <col style="width:16ex;"/>
        <col style='{$GutterStyle}' />
        <col style="width:16ex;"/>
        <col style='{$GutterStyle}' />
</xsl:if>
```

***Nommage des colonnes** :*

```
<xsl:if test="$noTimeStamp = 0">
            <th style="{$HeaderStyle}">
               <xsl:value-of select="$ColumnHeader_Date"/>
            </th>
            <th/>
            <th style="{$HeaderStyle}">
               <xsl:value-of select="$ColumnHeader_Time"/>
            </th>
</xsl:if>
```

***Valeurs** :*

```
   <xsl:if test="$noTimeStamp = 0">
            <td> <xsl:value-of select="@Date"/> </td>
            <td/>
            <td> <xsl:value-of select="@Time"/> </td>
            <td/>
   </xsl:if>
```

10. Produire une représentation schématique du document qui doit résulter de la transformation XSLT

Date	Heure	De	À	Message
05/09/2004	18:20:16	ed	viveXML	Hello !
05/09/2004	18:20:43	viveXML	ed	*Bonjour :)*
05/09/2004	18:22:45	ed	viveXML	re-bonjour
05/09/2004	18:22:58	viveXML	ed	*c un bon exemple ?*
05/09/2004	18:23:08	ed	viveXML	Oui, merci pour ton aide

À noter le changement de couleur de fond entre deux sessions.