L'Environnement Numérique de Travail *NetoCentre*. Le marquage XiTi.

P. Legay

18 décembre 2015

Table des matières

1 Contexte.

Pour évaluer les usages et pouvoir les comparer, les administrations de tutelle souhaite l'utilisation d'indicateur commun aux différents ENT de l'enseignement secondaire. La CDC-numerique (Caisse des dépôts) coordonne au niveau national la mise en place de ces indicateurs. Elle a choisi, pour ce faire, d'utiliser la technologie des marqueurs XiTi de la société Kleegroup.

La CDC a produit le document "Guide de marquage des environnements numériques de travail de l'enseignement secondaire", il fixe les nomenclatures communes à respecter et détail les méthodes de marquage à employer. Sa "Version 2.0 validée - Avril 2012" (CDC-doc) donne donc le cadre de référence à la mise en place des marqueurs XiTi.

Le principe de base est d'insérer dans chaque page de l'ENT un marqueur fournit par Kleegroup qui transmettra au site de Kleegroup les informations sur l'utilisateur connecté, son profil, son établissement et le type de service que la page fournie. Pour les services extérieurs, ceux dont on ne maîtrise pas le codage des pages, le marquage consiste à détecter l'appelle à ces services.

Dans le contexte *NetoCentre* toutes les pages sont dynamiques, les services proposés sont fonction de l'utilisateur et de l'établissement. Pour ne pas être trop intrusif, dans le code des différentes applications de l'ENT, les marqueurs XiTi seront posés en utilisant des fonctions JavaScript.

2 Le plan de marquage.

Le plan de marquage définit quelles pages doivent être marquées et à quels services elles correspondents. La nomenclature des services est définie dans CDC-doc.

L'objectif du CDC-numérique est d'avoir des indicateurs d'usage comparables entre les ENT du secondaire. Le plan de marquage doit donc être validé par le CDC-numerique. Il sert de référence pour la calification du marquage, effectuée aussi par CDC-numérique.

3 Les outils de marquage de l'ENT *NetoCentre.*

Le marquage de l'ENT se fait par un code JavaScript réparti dans deux fichiers : esciti.js et escitiData.js et un cookie. Ces fichiers seront placés sur le frontal pour être accessibles depuis les mêmes domaines que les portails. On y ajoutera, à la même place, le fichier xtfirst_ENT.js fourni par CDC.

Le script esciti.js contient du code statique (sans évolution) tandis que escitiData.js contient les données de paramétrage et le code pouvant évoluer dans le temps. Le cookie est utilisé pour mémoriser les dernières informations de marquage dans le cas où la page suivante est incapable de les déduire, elles sont alors réutilisables.

3.1 esciti.js

Script statique contenant le code de gestion du cookie. Il fait le chargement de escitiData.js versionné à la date du jour. Ceci force à une relecture tous les jours.

3.2 escitiData.js

Créé un objet JavaScript, $InfoENT4esciti$, qui contient les données d'initialisation liées à XiTi et dix tables de correspondance ou de nomenclature ; il faut les initialiser avec les données du domaine. À $InfoENT4esciti$ est aussi associé les fonctions JavaScript utilisables dans les pages à marquer.

3.2.1 Le paramétrage.

$InfoENT4esciti.projet$: numéro du projet XiTi.

$InfoENT4esciti.plateforme$: numéro de la plate-forme XiTi ; mettre la valeur par défaut, elle sera modifiée à l'exécution.

$InfoENT4esciti.collectivites$: tableau contenant les collectivités du projet.

L'indice dans le tableau donne la référence de la collectivité utilisée dans le reste du programme. Chaque enregistrement contient le numéro XiTi de la collectivité (*ID_COLLECTIVITE*) et le numéro de plate-forme XiTi (*ID_PLATEFORME*) correspondant.

Chaque ligne du tableau sera donc de la forme :

$$[ID\_COLLECTIVITE, ID\_PLATEFORME].$$

Nous avons, pour l'instant, deux collectivités : "Collectivité de Test" et "Région Centre Val de Loire". On peut en ajouter mais pas changer l'ordre.

$InfoENT4esciti.siren2etab$: table associative entre le *SIREN* d'une structure et le couple, numéro d'établissement XiTi (*ID_ETAB*), collectivité (idx). Tous les établissements à marquer doivent être dans cette table.

Une entrée sera de la forme :

$$\text{"}SIREN\text{"} : [ID\_ETAB, idx],$$

où idx est l'indice dans la table $InfoENT4esciti.collectivites$.

On utilisera par défaut le *SIREN* de la structure de rattachement des utilisateurs pour déduire le numéro d'établissement XiTi. Ceci est mal adapté aux utilisateurs membres de plusieurs établissements, d'où la table suivante.

$InfoENT4esciti.uai2siren$: table associative entre l'*UAI* d'un établissement et son *SIREN*. Tous les établissements à marquer ayant un *UAI* doivent être dans cette table.

Une entrée sera de la forme :

$$\text{"}UAI\text{"} : \text{"}SIREN\text{"}.$$

On utilisera l'*UAI* de l'établissement courant de l'utilisateur pour déduire le numéro d'établissement XiTi. S'il n'existe pas, ou n'est pas disponible, on prendra le *SIREN* de sa structure de rattachement.

$InfoENT4esciti.objClass2profile$: table associative entre les *ObjectClass* LDAP des utilisateurs et les profils XiTi (*ID_PROFILE*). Les profils XiTi sont définis dans CDC-doc.

Une entré sera de la forme :

$$\text{"}ObjectClass\text{"} : \text{"}ID\_PROFILE\text{"}.$$

On déduit les profils utilisateurs à l'aide de leurs *ObjectClass* LDAP. Il ne faut mettre dans cette table que les *ObjectClass* définissant de façon unique un profil. Les *ObjectClass* d'un utilisateur n'appartenant pas à cette table seront ignorés. Si un utilisateur a plusieurs *ObjectClass* appartenant à cette table, le profil sera déterminé indépendamment de l'ordre dans la table (fonction de l'implémentation JavaScript) , ça ne devrait pas arriver. Si un utilisateur n'a aucun *ObjectClass* correspondant, son profil sera '*Autres*', renseigné comme un pseudo-*ObjectClass* dans la table.

$InfoENT4esciti.service2id$: table associative entre les noms des services XiTi (*LIB_SERVICE*) et leurs identifiants XiTi (*ID_SERVICE*). Cette table est donnée dans CDC-doc.

Une entrée sera de la forme :

$$\text{"}LIB\_SERVICE\text{"} : ID\_SERVICE.$$

Dans les pages à marquer, on utilisera donc que les *LIB_SERVICE*, les identifiants seront déduits.

$InfoENT4esciti.context2Service$: table associative entre les noms de contexte des portlets du portail et les services XiTi. Les portlets ne sont pas des pages complètes, elles sont incluses dans les pages du portail. Leurs marquages se fera à partir de la page hôte en utilisant le service associé à leurs noms dans cette table.

Une entrée sera de la forme :

$$"fname" : "\textit{LIB\_SERVICE}".$$

Les noms indiqués ici peuvent être des sous-chaînes des noms de portlet (fname). Ainsi :

$$"annonces" : "Actualites",$$

marquera la portlet "$annoncesCFA$" et aussi "$annoncesCLG37$" avec le service "$Actualites$". Les portlets dont le nom n'est pas trouvé ne sont pas marquées.

$InfoENT4esciti.href2Service$: table associative entre les ancres (liens) et les services XiTi. C'est le même principe que pour les portlets appliqué au marquage sur le clic d'un lien. Tous liens (href) du portail qui contient une clé de cette table sera marqué (onclick) avec le service associé.

Une entrée sera de la forme :

$$"motClef" : "\textit{LIB\_SERVICE}".$$

Par exemple :

$$"sympa" : "Page\_ENT"$$

Associera un événement onclick de marquage, à tous liens (<a>) dont le href contient le mot 'sympa'.

$InfoENT4esciti.txt2Service$: table associative ayant le même rôle que $context2Service$ pour ce qui n'est pas portlet. On l'utilisera, par exemple, pour moodle où les 'id' des tags <body> définissent bien les services.

$InfoENT4esciti.uidAfiltre$: table utilisée que pour la phase de recette du projet, elle définit les utilisateurs de test qui seront les seuls à déclencher le marquage des pages. Il faut commenter cette table en phase de production, car sinon, tout utilisateur dont l'*UID* ne sera pas dans cette table ne sera pas marqué.

Une entrée sera de la forme :

$$"\textit{UID}" : "\textit{login}".$$

En fait le *login* n'est pas utilisé, seul l'*UID* a de l'importance.

$InfoENT4esciti.parametre$: table associative entre un nom de paramètre et un sélecteur *CSS*. Permet de changer les sélecteurs sans avoir à redéployer les applications, le portail notamment.

Par exemple, la ligne :

"%*portletPortail*" : "*.up-portlet-wrapper*",

Permet de déclarer que les contenant les portlets du portail ont la classe *CSS* '.up-portlet-wrapper' qui suffit à les identifier.

$InfoENT4esciti.onClickSelector2service$: table associative entre un sélecteur *CSS* et un service. À tout objet correspondant au sélecteur sera associé un événement onclick de marquage pour le service donné.
Une entrée sera de la forme :

"sélecteur css" : "*LIB_SERVICE*".

Par exemple :

".createListeDialog button" : "Courrier_Electronique",

marquera 'Courrier_Electronique' tous les clics sur les boutons de classe *CSS* 'createListeDialog'.

3.2.2 Les fonctions qui préparent le marquage.

Voici les fonctions JavaScript qui fixent les données utilisateur, utilisées dans les pages du portail et autres applications pour paramétrer le marqueur correspondant à la page :

$InfoENT4esciti.setObjectClass(objectClass)$: déduit le profil de l'utilisateur (*ID_PROFILE*), à partir de l'*ObjectClass* passé en paramètre, à l'aide de la table $objClass2profile$. Met à jour le cookie avec le profil trouvé. Si l'*ObjectClass* n'est pas renseigné ou aucun profil ne peut-être déduit alors on récupère la valeur précédente du cookie. À défaut de valeur, le profil sera fixé à '*Autre*'.

$InfoENT4esciti.setSiren(siren)$: déduit le numéro d'établissement XiTi (*ID_ETAB*), à partir du *SIREN* passé en paramètre, à l'aide de la table $siren2etab$. Met à jour le cookie avec le numéro d'établissement trouvé. Si le *SIREN* n'est pas renseigné on récupère alors la valeur précédente du cookie. Si le *SIREN* ne correspond à aucun établissement de la table, le cookie est réinitialisé.

$InfoENT4esciti.setUai(UAI)$: déduit le numéro d'établissement XiTi (*ID_ETAB*), à partir de l'*UAI* passé en paramètre, à l'aide de la table $uai2siren$ et $setSiren()$.
Met à jour le cookie avec le numéro d'établissement trouvé. Si l'*UAI* n'est pas renseigné ou s'il ne correspond à aucun établissement on récupère alors la valeur précédente du cookie.
Attention, certaines structures n'ont pas d'*UAI* mais toutes ont un *SIREN*.

Certain individu travail sur plusieurs établissements, dans l'ENT *NetoCentre* c'est l'*UAI* courant qui servira pour le marquage. S'il n'est pas connu on utilisera le *SIREN* de l'établissement de rattachement.

$InfoENT4esciti.setUid(uid)$: fixe l'*UID* de la personne dans le cookie (*ID_PERSO*).
Si l'*UID* diffère de la valeur précédente du cookie, les autres valeurs du cookie (établissement et profil) sont réinitialisées. C'est pourquoi, dans le marquage d'une page, l'*UID* est la première valeur à renseigner. Sinon les autres risquent la réinitialisation.

3.2.3 Les fonctions qui posent les marqueurs.

Ces fonctions placent le marqueur XiTi dans la page, elles ne peuvent servir que si les données de marquage sont renseignées dans le cookie à l'aide des fonctions précédentes (pas forcément dans la même page) :

$InfoENT4esciti.callByService(LIB\_SERVICE)$: demande de marquage pour le service *LIB_SERVICE*, l'*ID_SERVICE* sera déduit à l'aide de la table $service2id$. Le marquage sera effectué, si et seulement si, le cookie est correctement renseigné (*UID*, établissement et profil). Il y a appel du script XiTi, xtfirst_ENT.js.

$InfoENT4esciti.callByContextInCssClass(jQuery, selectorCss)$: demande de marquage pour tout tag *HTML* de la page correspondant au *selectorCss* et ayant une classe css clé de la table $context2Service$. Si *selectorCss* commence par %, alors on considère que c'est un paramètre dont la valeur (le sélecteur css) est dans la table $parametre$. Il faut passer la référence du *jQuery* déjà chargé par la page.
Par exemple, on peut marquer les portlets du portail avec l'appel suivant :

$$InfoENT4esciti.callByContextInCssClass(up.jQuery,'.up-portlet-wrapper');$$

ou si l'on garde l'exemple déjà donné :

$$InfoENT4esciti.callByContextInCssClass(up.jQuery,'\%portletPortail');$$

Les portlets ont toutes une ayant pour classe '$up-portlet-wrapper$' et une classe valant le 'fname' du service. Si l'on renseigne la table $context2Service$ avec le 'fname' associé au *LIB_SERVICE* XiTi souhaité, alors la page sera marquée avec ce *LIB_SERVICE*. Si plusieurs portlets correspondent, avec différent *LIB_SERVICE*, la page sera marquée pour chacun de ces services. Mais, chaque *LIB_SERVICE* ne devrait marquer qu'une fois.
Cette méthode retourne '*true*' si un marqueur a été posé.

$InfoENT4esciti.marqueOnClick(jQuery, cssAncreSelector)$: place tous les marqueurs onclick définis dans la table $onClickSelector2service$.
Place, en plus, des marquages onclick sur toutes les ancres sélectionnées par *cssAncreSelector* ayant dans leurs propriétés $href$, une clé de la table $href2Service$.

Si *cssAncreSelector* commence par % alors on considère que c'est un paramètre dont la valeur est dans la table *parametre*.

Pour la sélection et la pause des événements onclick la référence au *jQuery* passé en paramètre est utilisée.

Par exemple, on peut marquer les liens du portail sur les applications extérieurs avec :

$$InfoENT4esciti.marqueOnClick(up.jQuery,'a.externalLink');$$

Si la page contient le lien suivant que l'on souhaite marquer :

```
<a  class="externalLink"
    target="_blank"
    href="/portail/ExternalURLStats?
    fname=PortailArenA&service=https://extrante.ac-orleans-tours.fr/arena/"
    title="Portail ArenA">
```

Il suffit d'ajouter la clé 'arena' (mot discriminant de href) dans la table $href2Service$ avec le bon *LIB_SERVICE* associé, 'Orientation' par exemple.

$InfoENT4esciti_callById(tagName, defautService)$: recherche le premier tag de type *tagName* et utilise son identifiant (id) pour déduire à l'aide de la table $txt2Service$ le service de marquage (*LIB_SERVICE*).

Si aucun service n'est trouvé, le marquage se fait avec le *defautService*.

Cette fonction pose donc un et un seul marqueur.

4 Le marquage du portail de l'ENT *NetoCentre.*

Les pages du portail sont dynamiquement créées à partir de page `xsl`. Dans l'ENT *NetoCentre*, c'est le thème 'universality' qui est utilisé, donc les pages `xsl` à modifier sont dans l'arborescence `src/main/ressources/layout/theme/universality`. Le tag <body> de chaque page est géré dans `page.xsl`. C'est donc là, qu'il faut placer les marqueurs, avant la fermeture du <body>. Pour marquer, il faut connaître les informations sur l'utilisateur (*UID*, *ObjectClass*, *SIREN* ou *UAI*). L'*UID* et l'*UAI* courant sont déjà accessibles via des paramètres, *USER_ID* et *UAICourant*, déclarés dans `universality.xsl`. De la même manière, nous allons déclarer deux nouveaux paramètres, *OBJECTCLASS* et *SIREN*, dans `universalité.xsl`.

Les paramètres *USER_ID* et *UAICourant* sont mis à jour par un bean de la classe java `org.jasig.portal.rendering.xslt.StaticTransformerConfigurationSource`, Ils sont tirés directement, d'attributs du compte de la personne connectée. L'injections de ce bean, dans la configuration du portail, se fait dans le fichier `renderingPipelineContext.xml`.

Pour le paramètre *SIREN*, il n'existe pas d'attribut qui donne directement sa valeur ; l'attribut, dont on dispose, est le *DN* de la structure de rattachement. Ce *DN* contient le *SIREN*, mais ce n'est pas directe, il faut le déduire. Une nouvelle classe, `org.esco.portal.rendering.xslt.-CustomizeStaticTransformerConfigurationSource`, a donc été créée, sur le modèle de

`StaticTransformerConfigurationSource`. Elle permet, en appliquant une regex, de déduire une information contenue dans un attribut multi ou mono-valué.

Le paramètre *OBJECTCLASS* sera calculé à partir des attributs utilisateur filtrés par `CustomizeStaticTransformerConfigurationSource`, pour obtenir un *ObjectClass* unique définissant le profil de l'utilisateur.

4.1 Modification de `renderingPipelineContext.xml`.

Le chemin du fichier relatif au projet est : `src/main/resources/properties/contexts/-renderingPipelineContext.xml`. Dans la déclaration du bean *themeTransformComponent* : pour la propriété *xsltParameterSource* ; il faut ajouter dans la liste des sources notre bean (`CustomizeStaticTransformerConfigurationSource`). Ce qui donne, si on le place après le bean de classe `StaticTransformerConfigurationSource` :

```
...
<bean id="themeTransformComponent"
      class="org.jasig.portal.rendering.xslt.XSLTComponent">
  ...
  <property name="xsltParameterSource">
    <bean class="org.jasig.portal.rendering.xslt.MergingTransformerConfigurationSource">
      <property name="sources">
        <list>
          ...
              <!-- le bean deja defini pour verifier les entrees USER_ID et UAICourant --
          <bean class="org.jasig.portal.rendering.xslt.
                                    StaticTransformerConfigurationSource">
            <property name="parameters">
            ...
            </property>
            <property name="parameterExpressions">
              <map>
                ...
                <entry key="USER_ID" value="person.userName" />
                <entry key="UAICourant" value="person.getAttribute('ESCOUAICourant')" />
                ...
              </map>
            </property>
              ...
          </bean>
                <!-- declaration du nouveau bean -->
          <bean class="org.esco.portal.rendering.xslt.
                                        CustomizeStaticTransformerConfigurationSource">
            <property name="regexMap">
                  <!-- definit les regex pour le filtrage -->
              <map>
                <entry key="OBJECTCLASS"
                       value=".*(    ENTEleve
                                    |ENTAuxEnseignant
                                    |ENTAuxPersRelEleve
```

```
                                        |ENTAuxNonEnsEtab
                                        |ENTAuxNonEnsCollLoc).*"
                    />
                  <entry key="SIREN"
                         value="ENTStructureSIREN=([^,]+),
                                  ou=structures,dc=esco-centre,dc=fr" />
                </map>
              </property>
              <property name="parameterExpressions">

                <map>
                  <entry key="OBJECTCLASS"
                         value="person.getAttributeValues('\objectclass\')" />
                  <entry key="SIREN"
                         value="person.getAttribute('ENTPersonStructRattach')" />
                </map>
              </property>
            </bean>
            ...
          </list>
        </property>
      </bean>
    </property>
</bean>
...
```

4.2 Modification de `universality.xsl`.

Le chemin du fichier relatif au projet est : `src/main/ressources/layout/theme/universality/-universality.xsl`. Ici, il faut déclarer les nouveaux attributs, *OBJECTCLASS* et *SIREN*, sans valeur par défaut. Si, on les place avant les déclarations de *USER_NAME* et *UAICourant*, cela donne, par exemple :

```
    ...
<!-- GRENN
   xiti Setting -->
  <xsl:param name="OBJECTCLASS"></xsl:param>
  <xsl:param name="SIREN"></xsl:param>

<!-- ****** PORTAL SETTINGS ****** -->
<!--
   | YELLOW
   | Portal Settings should generally not be (and not need to be) modified.
    -->
    <xsl:param name="USER_ID">guest</xsl:param>
    <xsl:param name="userName">Guest User</xsl:param>
    <xsl:param name="USER_NAME"><xsl:value-of select="$userName"/></xsl:param>
  <xsl:param name="UAICourant"></xsl:param>
 ...
```

4.3 Modification de `page.xsl`.

Le chemin du fichier relatif au projet est : `src/main/ressources/layout/theme/universality/-page.xsl`. Il faut placer, dans la déclaration <body> de chaque page, plusieurs tags <script>. Le premier, pour faire le chargement du code JavaScript, esciti.js. Le deuxième, pour initialiser les données utilisateur. Le dernier, pour marquer la page.

Les deux premiers seront placés, dès l'ouverture du <body>, pour que l'initialisation soit disponible, si besoin, dans les portlets. Le dernier sera placé à la fermeture du <body>. Les marqueurs XiTi seront donc les derniers tags dans le <body>.

Ce qui donne dans `page.xsl` :

```
...
<body id="portal" class="up {$FLUID_THEME_CLASS}">
    <!-- Ajout de script pour XITI -->
      <!-- chargement esciti.js -->
  <script type="text/javascript" src="/esciti/js/esciti.js"></script>

      <!-- initialisation info utilisateur -->
  <script type="text/javascript">
    var X=InfoENT4esciti;
    X.setUid('<xsl:value-of select="$USER_ID" />');
    X.setSiren('<xsl:value-of select="$SIREN" />');
    X.setUai('<xsl:value-of select="$UAICourant" />');
    X.setObjectClass('<xsl:value-of select="$OBJECTCLASS" />');
  </script>
  <!-- Fin init XITI -->
...
...
    <!-- Ajout script pour XITI -->
  <script type="text/javascript">

      // marquage des portlets
    if (! InfoENT4esciti.callByContextInCssClass(up.jQuery, '%portletPortail')) {
        // si aucune portlet ne marque alors marquage page par defaut.
      InfoENT4esciti.callByService("Page_ENT");
    }
      // pose de tous les evenements onclick
    InfoENT4esciti.marqueOnClick(up.jQuery, '%ancrePortail');
  </script>
  <!-- Fin XITI -->
</body>
...
```

Attention :

- esciti.js doit être accessible dans le même domaine que le portail. Les fichiers esciti-Data.js et xtfirst_ENT.js, fourni par XiTi, doivent être dans le même répertoire que esciti.js. Dans notre exemple, ils sont accéssibles avec l'url `/esciti/js/` relative au domaine du portail.
- L'initialisation des données utilisateur doit impérativement commencer par l'*UID* ; de plus, le *SIREN* doit être instancié avant l'*UAI*.

– Pour le dernier tag <script>.
  – Le *callByService*("*Page_ENT*") fait le marquage par défaut de toutes les pages qui n'ont pas de portlet marquante.
  – Le *callByContextInCssClass*(*up.jQuery*,'*%portletPortail*') ajoute les marques des services rendus par les portlets.
  up.jQuery est la référence du jQuery utilisé par le portail.
  %portletPortail est le paramètre défini dans *InfoENT4esciti.parametre* qui doit donner les contenant les portlets. Le service de chaque portlet sera déduit des classes css de ces et de la table *context2Service* (voir escitiData.js).
  – Le *marqueOnClick*(*up.jQuery*,'*%ancrePortail*') place les evénements 'onclick' sur les liens de classes *css* définies par le paramètre *%ancrePortail*. Les services associés à ces marquages 'onclick' sont paramètrés dans *href2Service*. Cette méthode place aussi tous les marquages 'onclick' définis dans *InfoENT4esciti*.

5 Le marquage de la version mobile du portail *Neto-Centre.*

Ce qui change, dans la version mobile, c'est le thème, et certaines classes *css*. Le thème utilisé est 'muniversality' et toutes les adaptations se font dans
`src/main/ressources/layout/theme/muniversality/muniversality.xsl`.

5.1 Modification de `muniversality.xsl`.

Comme pour la version non-mobile, il faut déclarer les paramètres supplémentaires : *OBJECTCLASS*, *SIREN* et *UAICourant*. La pause des marqueurs se fait aussi dans ce fichier au début et à la fin de la création du tag <body>. Ce qui donne pour les paramètres dans la section "VARIABLES and PARAMETERS" :

```
...
<!-- ****** XSL UTILITY PARAMETERS ****** -->
...
<!--
    | GREEN
    | Add-on for XiTi
 -->
<xsl:param name="OBJECTCLASS"></xsl:param>
<xsl:param name="SIREN"></xsl:param>
<xsl:param name="UAICourant"></xsl:param>
...
```

Les tags <script> d'initialisation des données utilisateur au début du <body> :

```
...
<body class="up {$FLUID_THEME_CLASS} dashboard-{$VIEW}">
    <!-- Ajout de script pour XITI -->
  <script type="text/javascript" src="/esciti/js/esciti.js"></script>
```

```
  <script type="text/javascript">
    var X=InfoENT4esciti;
    X.setUid('<xsl:value-of select="$USER_ID" />');
    X.setSiren('<xsl:value-of select="$SIREN" />');
    X.setUai('<xsl:value-of select="$UAICourant" />');
    X.setObjectClass('<xsl:value-of select="$OBJECTCLASS" />');
  </script>
    <!-- Fin init XITI -->
```

Les tags pour poser les marqueurs à la fin du <body> :

```
...
    <!-- Ajout script pour XITI -->
  <script type="text/javascript">
    if (! InfoENT4esciti.callByContextInCssClass(up.jQuery, '%portletPortailMobile')) {
      InfoENT4esciti.callByService("Page_ENT");
    }
    InfoENT4esciti.marqueOnClick(up.jQuery, '%ancrePortailMobile');
  </script>
    <!-- Fin XITI -->
</body>
...
```

6 Le marquage de moodle.

Dans l'ENT *NetoCentre* on force, dès la connexion, les utilisateurs à passer par le portail. Donc, quand ils accèdent à moodle, ils ont forcément lu une page du portail. Cette page a créé le cookie contenant les informations utilisateur pour le marquage (*ID_PERSO*, *ID_PROFILE*, *ID_ETAB*...). On n'a donc plus qu'à pauser les marqueurs.

Dans moodle, les services sont facilement déductibles de l'identifiant du tag <body> de chaque page. On utilisera donc la fonction *callById*() pour marquer.

Les pages à marquer dépendent du thème. Dans l'ENT *NetoCentre*, ce sont les pages `theme/netocentrerwd/layout/columns?.php` (où ? prend les valeurs de 1 à 3) pour le thème `netocnetrerwd` et, les pages `theme/tourainerwd/layout/columns?.php` pour le thème `tourainerwd`.

6.1 Modification des fichiers de type `columns?.php`

Par défaut les pages seront marquées *'Parcours_Pedagogiques'*. Pour celles qui doivent être marquées différemment, il faut bien renseigné la table *txt2Service* avec l'identifiant de leurs tags <body>, associé au service souhaité.

Le marquage se fait en plaçant le code suivant avant la fermeture du <body> :

```
  ...
  <body ...>
  ...
```

```
  <script  type="text/javascript" src="/esciti/js/esciti.js"></script>
  <script  type="text/javascript">
    if (typeof  InfoENT4esciti != 'undefined') {
      InfoENT4esciti.callById('body', 'Parcours_Pedagogiques');
    }
  </script>
</body>
...
```

Le script perl script/marqueMoodlPage.pl, pose le code ci-dessus sur toutes les pages passées en paramètre. Par exemple, pour le thème `netocentrerwd`, il suffit d'exécuter :

```
script/marqueMoodlPage.pl theme/netocentrerwd/layout/columns?.php
```

Ceci marquera toutes les pages du thème `netocentrerwd`.

7 Le marquage du cahier de texte.

Le cahier de texte est une application *JEE* utilisant le framework *JSF* (voir sur git). Deux classes java ont été modifiées pour permettre la récupération des données utilisateur pour le marquage. Les autres modifications sont dans les pages xhtml de présentation.

7.1 Les modifications du code java.

`Profile.java` est l'énumération des profils utilisateur du cahier de texte ; on a simplement ajouté à chaque profil l'*ObjectClass* correspondant. On a déclaré, dans la classe `MenuControl`, une sous classe `Esciti` pour transmettre aux pages de présentation les données utilisateur via la méthode `getEsciti()`.

7.2 Les Modifications des pages.xhtml.

Chaque page <HTML> du cahier de texte est créé avec le tag <cr:page> défini par l'application. Le tag <cr:page> a été modifié pour pouvoir passer en paramètre (*libXiti*) le *LIB_SERVICE* correspondant à la page. Ce nouveau tag <cr:page> pose le marqueur XiTi, juste avant le </body> en utilisant la fonction JavaScript *callByservice* définie dans escitiData.js.

Si *libXiti* n'est pas défini alors le *LIB_SERVICE* utilisé est 'Cahier_Textes'. Ainsi, il n'est nécessaire de modifier que les pages qui ne doivent pas être marquées 'Cahier_Textes'. Par exemple, la page `planningMensuel.xhtml` doit être marquée 'Gestion_Temps' de la façon suivante :

```
<cr:page  titrePage="Cahier de texte mensuel"
          libXiti="Gestion_Temps"
>
...
</cr:page>
```

8 Quelques outils.

Dans le répertoire `script`, on dispose de quelques outils pour faciliter le travail.

8.1 js_compress.sh

Le script shell sciptjs_compress.sh permet la minimisation des fichiers JavaScript dont le nom commence par `esciti`. Il lit les fichiers `src/esciti*.js`, en enlève les lignes contenant des *console.log* et minimise le JavaScript à l'aide de la librairie java yuicompressor. Le résultat est placé dans le répertoire `js`. Il faut donc que java soit installé sur la machine et qu'une et une seule version de `script/yuicompressor*.jar` existe. Pour le débogage on peut copier directement le fichier `escitiData.js` dans `js` sans le minimisé. Attention, les lignes contenant des *console.log* étant supprimées il faut quelles ne contiennent que du code utile au débogage.

8.2 code_etab2js.pl

Le script perl script/code_etab2js.pl permet de construire les tables JavaScript *uai*2*siren* et *siren*2*etab* à partir des données fournit par Kleegroup (association des numéros d'établissement XiTi et leurs *UAI*) et des données de nos bases associant les *SIREN* et les *UAI*.

Par exemple, si le fichier `codeEtab.csv` contient les données fournit par Kleegroup sous ce format :

```
,CFA.BTP 28.Chartres,0280904F,2,,
```

si le fichier `uai2siren.txt` contient nos association *UAI SIREN* sous ce format :

```
0280904F        77534903800064
```

alors après avoir exécute la commande :

```
script/code_etab2js.pl codeEtab.csv uai2siren.txt > table.js
```

il suffit de copier le contenu de `table.js` dans `src/escitiData.js`.

8.3 marqueMoodlPage.pl

Le script perl script/marqueMoodlPage.pl, pose les tags définis pour le marquage moodle, avant le balise </body>, sur toutes les pages passées en paramètre. Par exemple, pour le thème `netocentrerwd`, il suffit d'exécuter :

```
script/marqueMoodlPage.pl theme/netocentrerwd/layout/columns?.php
```

Ceci marquera toutes les pages du thème `netocentrerwd`.