



ACube

Spécification Générale des évolutions de FRED 2.5



Version 1.3 du 19/04/2007

Etat : Validé



SUIVI DES MODIFICATIONS

Version	Rédaction	Description	Vérification	Date
1.0	F. RAYMONDAUD Q. DE LAPOINTE	Initialisation, rédaction composant menu	X. PRINCE	01/03/07
1.1	Q. DE LAPOINTE	Prise en compte des remarques du MAE	X. PRINCE	23/03/07
1.2	X. PRINCE	Suppression de la conversion UTF-8	X. PRINCE	18/04/07
1.3	G.PASQUEREAU	Suppression Documentation JSDoc pour projet	G.PASQUEREAU	19/04/07

LISTE DE DIFFUSION

Organisation	Nom	Info	Commentaire	Validation
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



SOMMAIRE

1	OBJECTIFS DU DOCUMENT	5
2	SPECIFICATION FONCTIONNELLE GENERALE.....	6
2.1	Composant Menu : Nouvelles fonctionnalités.....	6
2.1.1	Zones d'entête et de pied de page.....	6
2.2	Composant Tableau : Nouvelles fonctionnalités	7
2.2.1	Gestion de la marge.....	7
2.2.2	Gestion des tableaux vides	7
2.2.3	Possibilité de recharger un tableau	7
2.2.4	Outil d'export XML	8
2.2.5	Outil de gestion des colonnes.....	8
2.3	Composant Tableur : Nouvelles fonctionnalités.....	9
2.3.1	Regroupements et affichages	9
2.3.2	Un deuxième et un troisième niveaux implémentés.....	9
2.3.3	Tri par titres.....	9
2.3.4	Filtrage par titres.....	10
2.4	Composant Formulaire : Nouvelles fonctionnalités	12
2.4.1	Définition de la propriété action du formulaire	12
2.4.2	Classe ElementFormButtonLibre	12
2.5	Composant aide : lien et assistant de saisie	13
2.5.1	Un lien sur le libellé de l'élément de formulaire.....	13
2.5.2	Un assistant de saisie.....	13
2.5.3	Implémentation d'un exemple	13
2.6	Éléments de formulaire.....	15
2.6.1	Gestion du caractère « ` »	15
2.6.2	Changement de statut	15
2.6.3	Mise à jour de l'objet ElementFormCheckbox.....	15
2.6.4	Uniformisation de l'ElementFormTextArea	16
2.6.5	Refonte des classes ElementFormButton	16
2.6.6	Refonte du Flux de configuration du composant ElementFormSelectMaster	17
2.6.7	Ajout de fonctionnalités à la classe ElementFormSelect	20
2.7	Composant Calendrier	21
2.7.1	Calcul de la semaine	21
2.8	Autres évolutions.....	22
2.8.1	Valorisation des éléments de formulaire	22
2.8.2	Tâche ANT livraison DeveloppementLight.....	22
2.8.3	Gestion du format d'un numéro de téléphone	23
2.8.4	Composant Bulle.....	24
2.8.5	Composant DefaultElementForm.....	24
2.8.6	Composant DefaultXML	25
2.8.7	Amélioration de la tâche de remplacement.....	25
2.8.8	Mise à jour des illustrations de la JSDoc du framework	27
2.8.9	Récupération des paramètres dans une URL de type GET	27
2.8.10	Libellés d'erreurs en Espagnol	27
2.8.11	ComposantFile et ElementFormFile	27
2.9	Exemples supplémentaires implémentés	29
2.9.1	Initialisation à zéro	29
2.9.2	Un menu contenant des items de niveau 3	29
2.9.3	Un deuxième menu.....	29
2.9.4	Tableau à deux colonnes : String et date	31



2.9.5 ComposantRattachement : Libellés dans le constructeur, affichage et nouvelles fonctionnalités.....31

TABLEAUX

Erreur ! Aucune entrée de table d'illustration n'a été trouvée.

FIGURES

Erreur ! Aucune entrée de table d'illustration n'a été trouvée.

DOCUMENTS DE REFERENCE

Version	Titre

1 OBJECTIFS DU DOCUMENT

Il s'agit dans le cadre de la mise en place des évolutions au sein du Framework Ergonomique ACube de spécifier au travers de ce document le cadre fonctionnel des évolutions mises en place ainsi que leur mode de réalisation au sein du Framework.

Ce document sert donc à la fois de manuel utilisateur à destination des futurs utilisateurs du Framework, mais aussi de référence pour les futures maintenances sur le Framework.

2 SPECIFICATION FONCTIONNELLE GENERALE

Ce chapitre permet de définir dans les grandes lignes les nouvelles fonctionnalités mises en place dans le Framework Ergonomique. Il permet à un utilisateur de prendre rapidement connaissance de ces nouvelles fonctions et de leur contexte d'utilisation.

2.1 COMPOSANT MENU : NOUVELLES FONCTIONNALITES

2.1.1 ZONES D'ENTETE ET DE PIED DE PAGE

MAE_2094

Le composant menu dispose de deux nouvelles zones dans sa partie supérieure et inférieure: *divDebutMenu* et *divFinMenu*. Elles permettent « d'encadrer » le menu avec des informations personnalisables et indépendantes du flux XML de configuration du menu.

Ces zones sont initialisées automatiquement par le composant lors de son affichage. Leurs identifiants sont mémorisés dans les propriétés privées de l'objet *divDebutMenu* et *divFinMenu*. Quatre nouvelles méthodes publiques ont été ajoutées à la classe pour gérer cette nouvelle fonctionnalité :

- *setDebutHtml* pour définir le contenu de la zone d'entête du menu.
- *setFinHtml* pour définir le contenu de la zone de pied de page du menu.
- *getDebutId* pour obtenir l'ID de la zone *divDebutMenu*.
- *getFinId* pour obtenir l'ID de la zone *divFinMenu*.

Par défaut, les identifiants utilisés pour ces zones sont contenus dans les constantes *ID_DIV_DEBUT_MENU* et *ID_DIV_FIN_MENU*. Un exemple a été implémenté dans le menu général de la maquette du Framework, il place en fin de menu le numéro de version.

2.2 COMPOSANT TABLEAU : NOUVELLES FONCTIONNALITES

2.2.1 GESTION DE LA MARGE

MAE_228

La marge utilisée par le composant tableau lors de son affichage est désormais paramétrable. Il existe deux manières différentes pour effectuer ce paramétrage :

- Via le flux XML de configuration du tableau et son élément `<LARGEUR_MARGE>`.
- Via la méthode publique `setMarge` de l'objet tableau concerné.

Dans le cas où ces paramètres sont inexistants, les valeurs par défaut des marges sont de *0 px* pour un tableau et de *21 px* pour une liste.

Afin de réaliser cette fonctionnalité, il a été ajouté les propriétés et méthodes suivantes à la classe *ComposantTableau* :

- La propriété privée `marge` qui contient la valeur de la marge du tableau.
- La méthode publique `getMarge` pour obtenir la valeur de la marge.
- La méthode publique `setMarge` pour définir la valeur de la marge.

2.2.2 GESTION DES TABLEAUX VIDES

MAE_2804

Il est désormais possible de gérer un tableau « vide » dès le chargement de la page. Pour cela, il suffit de renseigner le paramètre `urlXMLData` du constructeur de l'objet *ComposantTableau* à *null*. Ainsi, la propriété `XMLData` de l'objet, représentant les données du tableau au format XML, est initialisée avec un objet de type *XMLObject* ne contenant aucune information (son attribut `xmlDoc` est initialisé avec un objet *XmlDocument* vide). Il appartiendra à l'utilisateur de renseigner cet objet lors d'évènements de type ajout, modification ou suppression.

Un exemple d'utilisation a été réalisé dans la maquette à la page *Article>Consultation>Détail de l'article*, au niveau de l'onglet *Fournisseurs*.

2.2.3 POSSIBILITE DE RECHARGER UN TABLEAU

MAE_1816

Réalisation de la méthode `refresh()` dans `fw_tableau.js` qui permet de recharger le fichier XML contenant les données d'un tableau. Ajout d'une icône pour ce rafraîchissement dans les outils du tableau. Pour ajouter l'icône, placer un tag `ACTION_REFRESH_PRESENT` de valeur *true* dans la balise `OUTILS`. Il est également souhaitable de renseigner la balise `ALT_REFRESH`.

Dans la maquette, un exemple est accessible depuis le menu *Articles>Catalogue*. Pour tester le bon fonctionnement du chargement, aller dans le répertoire de déploiement apache (`htdocs/Maquette`) puis ouvrir et modifier le fichier `flux/protected/article/ListeArticleParCritere.xml`, les modifications doivent alors être prises en compte dans la page de catalogue.

2.2.4 OUTIL D'EXPORT XML

Dans le fichier XML de description de flux de la maquette, la balise `<OUTILS>` contenant les outils automatisés peut maintenant contenir une balise `<FONCTION_EXPORT_XML>` dans laquelle on spécifie la fonction à appeler pour réaliser l'export XML des données du tableau. La DTD associée au tableau a été mise à jour.

Afin d'ajouter le bouton pour l'export XML, la fonction `initInfosOutilsByXML()` dans le `jsclient/ergonomique/fw_tableau.js` du Framework a été modifiée. On y ajoute les variables `fonctionExportXML` et le booléen `boolExportXML` à partir desquelles on crée un `ObjectOutilTab`. Cela permet d'afficher le bouton de façon automatique et de l'associer à la fonction donnée dans le XML.

Dans la maquette, on peut trouver un exemple dans `xml/catalogue.xml` avec une fonction `doExport()` associée à l'export XML. Cette fonction est définie dans le fichier JavaScript `jsclient/article/catalogue.js`. Elle permet d'exécuter la fonction du Framework `composantTableauTest.exportData()` puis d'afficher le flux obtenu sous la forme décidée : ici, ce sera sous forme de page HTML.

2.2.5 OUTIL DE GESTION DES COLONNES

MAE_2205

Dans le fichier XML de description de flux de la maquette, la balise `<OUTILS>` contenant les outils automatisés peut maintenant contenir une balise `<ACTION_GESTION_COLONNES_PRESENT>` que l'on renseigne à OUI ou true si l'on souhaite pouvoir accéder à l'outil de gestion des colonnes du tableau. Cette balise doit obligatoirement apparaître dans le fichier XML mais peut bien évidemment être renseignée à NON ou false si l'outil n'est pas souhaité. Le fichier de description du flux XML de paramétrage du tableau (DTD) a été mise à jour pour prendre en compte cette balise.

Afin d'ajouter le bouton de gestion des colonnes, la fonction `initInfosOutilsByXML()` dans le `jsclient/ergonomique/fw_tableau.js` du Framework a été modifiée. On y ajoute le booléen `boolGestionCol` à partir duquel on crée un `ObjectOutilTab`. Cela permet d'afficher le bouton de façon automatique et de l'associer à la fonction `gestionColonne()` du tableau. Cette fonction ouvre la popup de gestion des colonnes et permet de sélectionner celles que l'on souhaite afficher ou non.

Dans la maquette, on peut trouver un exemple dans `xml/catalogue.xml`. On accède à cet outil dans `article/catalogue`.

2.3 COMPOSANT TABLEUR : NOUVELLES FONCTIONNALITES

2.3.1 REGROUPEMENTS ET AFFICHAGES

MAE_2205

Ajout d'une image permettant l'expansion ou non d'un groupement de tableur. Les groupements sont initialisés dans le fichier XML en entrée. On peut désormais les afficher ou non sur la page HTML.

Pour cela, implémentation d'une méthode `onClickGroupe` dans la classe `ComposantTableur` qui est appelée depuis la méthode HTML `onClick` de la ligne de titre. Cette méthode gère :

- Le changement de l'image d'expansion de la partie.
- Le parcours des lignes du tableur correspondantes à la partie sélectionnée pour affichage ou non-affichage.
- Les niveaux de titres sont testés pour une bonne gestion des affichages.

2.3.2 UN DEUXIEME ET UN TROISIEME NIVEAUX IMPLEMENTES

MAE_2205

Afin de mieux illustrer les différentes fonctionnalités proposées par le tableur, les fichiers xml de définition et de contenu ont été complétés :

- Dans `xml/cahier/catalogueParCahier.xml`, déclaration d'un deuxième niveau de titres contenus dans une balise `CAHIER2` et d'un troisième niveau de titres contenus dans une balise `CAHIER3`.
- Dans `flux/protected/cahier`, création de titres de niveau 2 et 3.
- Mise à jour des fichiers CSS associés pour affichage différent selon le niveau.

2.3.3 TRI PAR TITRES

MAE_2205

Implémentation dans la barre d'outils du tableur, d'une fonctionnalité de tri par titres. Le tri effectué sera alternativement croissant, décroissant ou dans l'ordre du fichier XML. L'icône de tri affichée dans la barre d'outils change à chaque tri pour proposer la possibilité suivante. Les fonctionnalités `onMouseOver` et `onMouseOut` sont existantes sur l'image.

L'ajout de l'icône a été réalisé dans la méthode `initInfosOutilsByXML` du `ComposantTableau`, dont hérite le `ComposantTableur`. Trois `ObjectUtilTab` sont créés, correspondants aux trois états possibles du tri. Ils sont stockés dans le nouvel attribut `tabObjectUtilTri` du `ComposantTableur`.

Au clic sur l'icône de tri des titres, la méthode `triTableur(ordre)` du `ComposantTableur` est lancée. Elle prend en paramètre la constante correspondant à l'ordre de tri ('CROISSANT', 'DECROISSANT', 'NON'). Cette méthode exécute les opérations suivantes :

- Modifie l'icône de tri
- Récupère les niveaux de titres à trier dans le tableur (de 0 à 3), et appelle la méthode de tri du `ComposantTableau`.
- Les Titres sont alors triés par ordre alphabétique, mais pas leur contenu, celui-ci étant triable à partir des options du tableau.

Il est à noter que l'option de tri du tableau depuis les différentes colonnes a été rajoutée en l'état dans le tableur.

La fonctionnalité est mise en place de façon automatique lors de la création d'un tableur. On peut la trouver par exemple dans la partie Cahiers Navette>Catalogue de la maquette ACube.

2.3.4 FILTRAGE PAR TITRES

MAE_2205

Ajout de l'icône de filtrage :

Implémentation dans la barre d'outils du tableur, d'une fonctionnalité de filtre par titres. Les fonctionnalités `onMouseOver` et `onMouseOut` sont existantes sur l'image.

L'ajout de l'icône a été réalisé dans la méthode `initInfosOutilsByXML` du ComposantTableau, dont hérite le ComposantTableur. Cet ajout dépend de la définition du fichier XML de configuration du Tableur. Ici, c'est le fichier `catalogueParCahier.xml` qui sert d'exemple. Il est associé à la page Cahiers Navette>Catalogue de la maquette. Dans ce fichier est définie une balise `<OUTILS><ACTION_FILTRE_TITRES_PRESENT>` que l'on renseigne à OUI si l'on souhaite activer le filtre sur les titres du tableur. L'outil est associé à la méthode `ouvrirFiltreTitres()` du tableur. Cette méthode ouvre une fenêtre popup de filtrage par titres.

La popup de filtrage par titres correspond au fichier `html/tableur/filtreTitres.html` du framework. Ce fichier HTML va lancer la méthode globale `ecrireBindPopupTableur()` en passant en paramètre le tableur auquel elle a accès grâce à la variable globale `tableurEnCours` définie dans `fw_tableur.js`. Cela va créer des éléments de formulaire de type checkbox dans la popup, grâce à

- Un parcours des titres de différents niveaux par la méthode `listerTitres` du ComposantTableur.
- Une génération automatique des éléments du tableau HTML nécessaires à l'insertion des checkbox.

On ajoute alors des boutons de validation de d'annulation du formulaire.

Gestion dynamique des cases à cocher :

Dans la pop-up, à chaque titre correspond une case à cocher. On peut choisir plusieurs titres à filtrer en même temps.

Voici les règles de gestion qui ont été implémentées :

- Quand on veut filtrer un titre de niveau 1 ou 2, par défaut tous les sous titres sont cochés. De même quand on décoche un titre de niveau 1 ou 2, tous les sous titres cochés sont décochés.
- Quand on souhaite filtrer un titre de niveau 2 ou 3, cela coche automatiquement les titres associés de niveau(x) supérieurs. Si on décoche un niveau bas, cela n'a pas d'incidence sur les titres de niveaux supérieurs.

Cette gestion est réalisée de façon automatique grâce à la redéfinition de la méthode `onChange` des checkbox. On trouve cette fonction dans `fw_tableur.js` : `onChangeCheckboxTitreFiltre()`.

Soumission du formulaire des titres à filtrer :

La méthode d'annulation permet de fermer la pop-up.



La méthode de validation est déclarée en méthode globale dans fw_Tableur.js : `validerPopupTableur`. Elle parcourt les différents titres, retrouve lesquels correspondent à des cases cochées et met à jour les attributs `filtresTitreNiveau1`, `filtresTitreNiveau2` et `filtresTitreNiveau3` du tableur. Ces tableaux contiennent chacun le libellé du titre à filtrer, ou pour les tableaux 2 et 3 un libellé vide si le tri s'effectue sur un niveau supérieur à eux.

Un booléen `boolFiltrageTitreEnCours` est mis à jour pour indiquer si le filtrage sur les titres est activé ou non. Ces variables sont réutilisées ensuite pour cocher directement les titres sélectionnés quand on rouvre la pop-up de filtrage (utilisation de la fonction `existeFiltreTitre` sur des données des 3 tableaux pour savoir si un titre est contenu dans le filtrage ou non). Et pour afficher seulement les titres filtrés lors de la réécriture du tableur demandée en fin de la méthode de soumission.

Ecriture du tableur filtré :

Lors de la réécriture du tableur, la méthode `testFiltreTitre` appelée sur un item XML permet de savoir ce qui doit être affiché ou non. Elle est utilisée, ainsi que le booléen `boolFiltrageTitreEnCours` au niveau des méthodes d'écriture des lignes du tableur.

Voir l'exemple dans la maquette :

L'exemple du tableur et du filtrage par titres est accessible dans la maquette dans le menu Cahier Navette/Catalogue. On y trouve l'icône de filtrage dans les outils.

2.4 COMPOSANT FORMULAIRE : NOUVELLES FONCTIONNALITES

2.4.1 DEFINITION DE LA PROPRIETE ACTION DU FORMULAIRE

MAE_2340

La propriété *action* du composant Formulaire, qui contient l'URL vers laquelle vont être envoyées les données saisies dans le formulaire suite à une action de type *submit*, est désormais modifiable de façon régulière via la méthode *setAction* de l'objet *ComposantFormulaire*.

Un exemple d'utilisation a été réalisé dans la maquette à la page *Article>Saisie*, avec le cas d'utilisation suivant :

- Référence de l'article à 0002.
- Cliquez sur l'icône disquette dans le menu pour soumettre le formulaire.
- Le formulaire est posté à la page *Article>Consultation*.

2.4.2 CLASSE ELEMENTFORMBUTTONLIBRE

MAE_2098

Cette classe est destinée à la création d'un bouton comportant un texte initialisé et modifiable par l'utilisateur. Ce bouton cliquable appelle la méthode *agir()* qui doit être implémentée par l'utilisateur.

La classe *ElementFormButtonLibre* est définie dans le fichier *fw_formulaire_elements.js*. Elle hérite de la classe *ElementFormButton*. Ses particularités : un attribut *libelleLibre*, modifiable par la méthode « *setLibelleLibre(nouveauLibelle)* » et accessible par la méthode « *getLibelleLibre* ». On y définit également une méthode « *agir()* » à redéfinir par l'utilisateur dans son application : dans cette méthode se trouveront toutes les actions lancées au clic du bouton.

Les méthodes *onMouseOver* et *onMouseOut* sont surchargées par rapport à un *elementFormButton* classique afin de permettre le rollover sur le bouton libre.

Un exemple d'utilisation apparaît dans la maquette à la page *Article>Saisie*, le bouton comporte le libellé « Ou suis-je ? », le clic provoque l'affichage d'une alerte et modifie le libellé du bouton :

```
var btLibre=new ElementFormButtonLibre("btLibre","Ou suis-je?");
composantFormulaire.addElement(btLibre);

//Action associée au bouton libre
composantFormulaire.agir=function(){
    alert("Vous êtes à la fin de la partie article du formulaire de saisie");
    btLibre.setLibelleLibre("Vous êtes ici");
    composantFormulaire.ecrireBind();
};
```

2.5 COMPOSANT AIDE : LIEN ET ASSISTANT DE SAISIE

2.5.1 UN LIEN SUR LE LIBELLE DE L'ÉLÉMENT DE FORMULAIRE

La page d'aide est désormais accessible depuis l'image d'aide, mais aussi depuis le libellé de l'élément de formulaire. Dans le DefaultElementForm, modification de la fonction seturlAide() pour prendre un deuxième paramètre : lien.

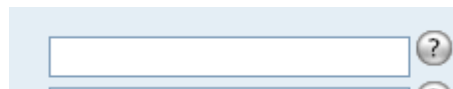
- Si le chiffre lien n'est pas renseigné ou vaut « 0 », le lien sera existant seulement sur l'image.
- Si le chiffre est à « 1 », le lien sera seulement sur le libellé de l'élément.
- Si le lien est à « 2 », l'image et le libellé seront tous les deux cliquables.

L'exemple dans Articles>Saisie.js a été complété en ajoutant ce paramètre sur l'élément référence.

2.5.2 UN ASSISTANT DE SAISIE

Utilisation d'un assistant de saisie : similarités avec le composant aide

- Ouverture d'une page d'assistant à partir d'une URL (paramètre urlAssist de la classe DefaultElementForm)
- Ouverture d'une page d'assistant à partir d'un chemin XML de type aide:tag1/tag2/.../tagN (paramètre urlAssist de la classe DefaultElementForm).
- Lors de l'ajout des éléments de formulaire, si l'attribut urlAssist est renseigné, affichage d'une icône cliquable, amenant l'utilisateur vers la page désirée.
- Affichage de la page d'assistant soit à partir de l'url donnée, soit sous forme d'arborescence ouverte à l'item désiré et à la page de droite correspondante (Création du ComposantAssistant dans Assistant.js de la maquette, fonction lierPageDroite du Framework attendu en attribut XML func de l'item).



Dans la classe DefaultElementForm, ajout des méthodes d'accès et de modification pour urlAssist et altAssist (geturlAssist, seturlAssist, setaltAssist). Ecriture de l'image et lien vers la page.

Création dans le ComposantAide d'une méthode `mettreAJour(valeur)` qui récupère l'élément de formulaire depuis lequel l'assistant est ouvert, le met à jour avec la valeur passée en paramètre. Dans le cas d'un ElementFormSelect ou ElementFormRadio, la valeur peut soit être le numéro de l'option à sélectionner, soit un tableau d'ObjectOption qui remplacera alors toutes les valeurs de l'élément de formulaire par les nouvelles options du tableau.

2.5.3 IMPLEMENTATION D'UN EXEMPLE

Un exemple d'assistant dans la page Article>Consultation Rechercher : resultat.js . Le lien se fait vers la page assistant.html. Le XML chargé est aujourd'hui le même que pour la page d'aide. Les liens proposés pour la mise à jour de l'élément de formulaire sont créés : création d'un tableau contenant les textes des liens et les valeurs associées, puis création du code html associé pour appel de la méthode javascript du framework `mettreAJour(valeur)`.

```
function creerLiens(){  
    switch(parent.window.opener.currentElementAssistant.id){  
        case('reference'):
```

```
        tabLiens[0]='1101 F/B';
        tabLiens[1]='Agrafe JAKY 6 mm';

        ...
        parent.frames['droite'].document.getElementById("droiteLiensReference").innerHTML += ajouterLiens(tabLiens);
        break;
        case('designation'):
            tabLiens[0]='Agrafe JAKY 6 mm';
            tabLiens[1]='Agrafe JAKY 6 mm';

            ...
            parent.frames['droite'].document.getElementById("droiteLiensDesignation").innerHTML += ajouterLiens(tabLiens);
            break;
        default:
            break;
    }
    return tabLiens;
}

//Fonction générant le code HTML cliquable permettant de mettre à jour l'élément
lié à la page d'aide
function ajouterLiens(tabLiens){
    var codeHTMLLiens='<br><h2>Nouvelles valeurs possibles : </h2><ul>';
    for (var i=0; i<tabLiens.length; i=i+2){
        codeHTMLLiens+="<li><a
href='javascript:parent.frames[\"gauche\"].composantAideAssistantTest.mettreAJour(
parent.tabLiens[\"+i+\"]);'>"+tabLiens[i+1];
        codeHTMLLiens+="</a></li>";
    }
    codeHTMLLiens+='</ul>';
    return codeHTMLLiens;
}
```

2.6 ELEMENTS DE FORMULAIRE

2.6.1 GESTION DU CARACTERE « ` »

MAE_2805

Certaines classes du type *ElementForm* contiennent des attributs de type chaîne de caractères. Il s'agit le plus souvent des attributs *value*, *alt* et *title*. Lors de la génération du code HTML de ces composants par la méthode *ecrireBind()*, si un de ses attributs possédait le caractère « ` », la suite de la chaîne était tronquée à l'affichage.

Désormais, une modification de la méthode *ecrireBoite()* des éléments concernés a permis de résoudre ce problème.

Un exemple est disponible sur la maquette au niveau du formulaire de modification d'un article : *article>catalogue>rechercher>article : chemise MAE bleue d'azur*.

2.6.2 CHANGEMENT DE STATUT

MAE_2392

Il a été rapporté que la méthode *setStatut()* ne fonctionnait pas sur les éléments de formulaire *ElementFormSelect*, *ElementFormCheckbox* et *ElementFormRadio*. En réalité, seul le changement de statut en *READONLY* ne fonctionne pas sur ces trois éléments de formulaire : ces tags HTML n'intègrent pas nativement la gestion de la lecture seule.

La solution choisie a été de mettre en place du code JavaScript destiné à simuler un comportement *READONLY* pour ces éléments. Ainsi les méthodes *ecrireBoite()* de ces éléments de formulaire ont été modifiées.

Les exemples dans la maquette, pour chaque élément de formulaire, sont disponibles comme suit :

- Pour l'élément *ElementFormSelect* : *articles->consultation->rechercher->onglet description->* le choix des *ElementFormRadio* change le statut de l'*ElementFormSelect* unité.
- Pour l'élément *ElementFormCheckbox* : *articles->consultation->rechercher->onglet budget->* le choix des *ElementFormRadio* change le statut de la *ElementFormCheckbox* unité.
- Pour l'élément *ElementFormRadio* : *articles->saisie->* les trois dernières unités mettent exactement dans l'ordre *enabled*, *readonly* et *disabled* le statut des *ElementFormRadio* « utilisable par ».

2.6.3 MISE A JOUR DE L'OBJET ELEMENTFORMCHECKBOX

MAE_2372

Les méthodes *getValue()* des éléments de formulaire sont chargées de lire la valeur actuelle de cet élément à l'écran (via l'objet DOM correspondant) et de mettre à jour l'objet *ElementForm* correspondant.

Dans le cas de l'élément *ElementFormCheckbox*, ce principe n'était pas respecté et a donc été corrigé.

2.6.4 UNIFORMISATION DE L'ELEMENTFORMTEXTAREA

MAE_2215

Une barre de défilement grisée s'affichait du Internet Explorer et pas sur les autres navigateurs : ajout dans le fichier globale.css d'un attribut de style dur les textarea permettant de n'afficher la scrollbar que si elle est utile (pas d'affichage grisé sur ie) :

```
textarea{
    overflow: auto;
}
```

Concernant le nombre de colonnes affichées dans les textarea, l'utilisateur doit prévoir une largeur de case de tableau suffisamment grande pour contenir le nombre passé en paramètre. Dans le cas contraire, c'est la largeur de la case du tableau qui limitera la largeur de la zone de texte.

Des tests ont été effectués sur les différents navigateurs et le nombre de lignes et colonnes qu'ils affichent, prenons un nombre nbRows de lignes, et un nombre nbCols de colonnes passées en paramètres d'un textarea, voici ce qui sera affiché :

- Firefox affiche nbRows+1 et nbCols+2
- Internet Explorer affiche nbRows-1 et nbCols-1
- Mozilla affiche nbRows ok et nbCols+3

Il n'est pas prévu aujourd'hui dans l'application Acube de tester le navigateur pour modifier nbRows et nbCols en fonctions, car cela demanderait de mettre en dur dans le code des tests peu recommandés et de retester ce point pour chaque changement de version de navigateur.

Autre différence, aujourd'hui non corrigible en respectant la norme du W3C : sur Internet Explorer, quand on arrive à la fin d'une ligne du textArea par défaut on passe à la ligne suivante, sur Mozilla et Firefox la ligne se poursuit s'il n'y a pas de retour à la ligne.

2.6.5 REFONTE DES CLASSES ELEMENTFORMBUTTON

La relation d'héritage entre la classe *ElementFormButton* et ses classes filles *ElementFormButtonLibre*, *ElementFormButtonAnnuler* et *ElementFormButtonValider* a été repensé afin de centraliser la gestion de l'attribut *href* (lien associé à un clic sur le bouton).

Désormais, les méthodes *getHref()* et *setHref(lien)* permettront respectivement d'obtenir et de définir la valeur de l'attribut *href* d'un composant *ElementFormButton*.

Pour les boutons de type *ElementFormButtonAnnuler* et *ElementFormButtonValider*, l'attribut *href* est automatiquement renseigné pour effectuer respectivement une annulation du formulaire et une validation du formulaire.

Pour les boutons de type *ElementFormButtonLibre*, le renseignement de la propriété *href* se fait de façon directe via le constructeur de la classe.

L'exemple du composant *ElementFormButtonLibre* dans la maquette a été mis à jour en conséquence (*Articles>Saisie*).

2.6.6 REFONTE DU FLUX DE CONFIGURATION DU COMPOSANT ELEMENTFORMSELECTMASTER

MAE_2273

Lors de la mise en œuvre de listes liées grâce au composant *ElementFormSelectMaster*, dans certain cas, une grande redondance des données est présente.

Si nous prenons l'exemple d'une interface graphique avec des listes liées ZONES / PAYS / VILLES et le choix TOUS dans les deux premières listes, la redondance des données dans ce cas de figure est flagrante. Il apparaît important d'éviter cette redondance afin de ne pas obtenir des flux de taille démesurée.

C'est pourquoi le flux de données d'un composant *ElementFormSelectMaster* a été refondu pour faire apparaître la notion de « groupement de données » :

- une donnée est un flux XML représenté de la manière suivante :

```
<OPTION>
  <LIBELLE></LIBELLE>
  <VALUE></VALUE>
</OPTION>
```

- un groupement sera identifié par un élément <ID> et contiendra lui-même des données ou des groupements de données.
- Un groupement désigne par un élément <VALUE_SELECTED> la valeur de la donnée à utiliser comme donnée par défaut.
- Un groupement désigne par un élément <TYPE_TRI> le type de tri à appliquer à ses éléments :
 - AUCUN : l'ordonnement utilisé est naturel (ordre de déclaration des données)
 - LIBELLE : l'ordonnement utilisé est l'ordre lexicographique sur les éléments <LIBELLE> des données.
 - VALUE : l'ordonnement utilisé est l'ordre lexicographique sur les éléments <VALUE> des données.

Par exemple :

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<PAGE>
  <!-- DESCRIPTION DU FORMULAIRE -->
  <FORMULAIRE>
    <ELEMENT>
      <LIBELLE>Zones :</LIBELLE>
      <OPTIONS>
        <OPTIONS_GRP>ZONES</OPTIONS_GRP>
      </OPTIONS>
    </ELEMENT>
  </FORMULAIRE>

  <!-- DESCRIPTION DES GROUPEMENTS DE DONNEES -->
  <DATA>
    <OPTIONS_GRP_DEF>
      <ID>EUR</ID>
      <TYPE_TRI>LIBELLE</TYPE_TRI>
      <VALUE_SELECTED>1</VALUE_SELECTED>
      <OPTIONS>
        <OPTION>
```

```

        <LIBELLE>France</LIBELLE>
        <VALUE>1</VALUE>
        <SLAVE>
            <LIBELLE>Villes :</LIBELLE>
            <OPTIONS>
                <OPTIONS_GRP>FR_VILLES</OPTIONS_GRP>
            <OPTIONS>
        </SLAVE>
    </OPTION>
    <OPTION>
        <LIBELLE>Italie</LIBELLE>
        <VALUE>2</VALUE>
    </OPTION>
    <OPTION>
        <LIBELLE>Espagne</LIBELLE>
        <VALUE>3</VALUE>
    </OPTION>
</OPTIONS>
</OPTIONS_GRP_DEF>

<OPTIONS_GRP_DEF>
    <ID>FR_VILLES</ID>
    <TYPE_TRI>LIBELLE</TYPE_TRI>
    <VALUE_SELECTED>10</VALUE_SELECTED>
    <OPTIONS>
        <OPTION>
            <LIBELLE>Paris</LIBELLE>
            <VALUE>10</VALUE>
        </OPTION>
    </OPTIONS>
</OPTIONS_GRP_DEF>

<OPTIONS_GRP_DEF>
    <ID>AME</ID>
    <TYPE_TRI>LIBELLE</TYPE_TRI>
    <VALUE_SELECTED>4</VALUE_SELECTED>
    <OPTIONS>
        <OPTION>
            <LIBELLE>Etats-unis d'amérique</LIBELLE>
            <VALUE>4</VALUE>
        </OPTION>
        <OPTION>
            <LIBELLE>Canada</LIBELLE>
            <VALUE>5</VALUE>
        </OPTION>
        <OPTION>
            <LIBELLE>Espagne</LIBELLE>
            <VALUE>6</VALUE>
        </OPTION>
    </OPTIONS>
</OPTIONS_GRP_DEF>

<OPTIONS_GRP_DEF>
    <ID>ASIE</ID>
    <TYPE_TRI>LIBELLE</TYPE_TRI>
    <VALUE_SELECTED>8</VALUE_SELECTED>
    <OPTIONS>
        <OPTION>

```

```

        <LIBELLE>Japon</LIBELLE>
        <VALUE>7</VALUE>
    </OPTION>
    <OPTION>
        <LIBELLE>Chine</LIBELLE>
        <VALUE>8</VALUE>
    </OPTION>
    <OPTION>
        <LIBELLE>Thaïlande</LIBELLE>
        <VALUE>9</VALUE>
    </OPTION>
</OPTIONS>
</OPTIONS_GRP_DEF>

<OPTIONS_GRP_DEF>
    <ID>MONDE</ID>
    <TYPE_TRI>LIBELLE</TYPE_TRI>
    <VALUE_SELECTED>11</VALUE_SELECTED>
    <OPTIONS>
        <OPTION>
            <LIBELLE>Amérique du nord</LIBELLE>
            <VALUE>11</VALUE>
            <SLAVE>
                <LIBELLE>Pays :</LIBELLE>
                <OPTIONS>
                    <OPTIONS_GRP>AME</OPTIONS_GRP>
                <OPTIONS>
            </SLAVE>
        </OPTION>
        <OPTION>
            <LIBELLE>Asie</LIBELLE>
            <VALUE>12</VALUE>
            <SLAVE>
                <LIBELLE>Pays :</LIBELLE>
                <OPTIONS>
                    <OPTIONS_GRP>ASIE</OPTIONS_GRP>
                <OPTIONS>
            </SLAVE>
        </OPTION>
        <OPTION>
            <LIBELLE>Europe</LIBELLE>
            <VALUE>13</VALUE>
            <SLAVE>
                <LIBELLE>Pays :</LIBELLE>
                <OPTIONS>
                    <OPTIONS_GRP>EUR</OPTIONS_GRP>
                <OPTIONS>
            </SLAVE>
        </OPTION>
    </OPTIONS>
</OPTIONS_GRP_DEF>

<OPTIONS_GRP_DEF>
    <ID>ZONES</ID>
    <TYPE_TRI>LIBELLE</TYPE_TRI>
    <VALUE_SELECTED/>
    <OPTIONS>
        <OPTIONS_GRP>MONDE</OPTIONS_GRP>
    </OPTIONS>

```

```
</OPTIONS>  
  </OPTIONS_GRP_DEF>  
</DATA>  
</PAGE>
```

Un exemple a été implanté au niveau du menu *cahiers navettes* > *affectation*. Le nouveau flux XML se situe dans le fichier *affectationVide2.xml*.

2.6.7 AJOUT DE FONCTIONNALITES A LA CLASSE ELEMENTFORMSELECT

MAE_3042

Il est désormais possible de définir la largeur d'une liste (classe ElementFormSelect et classe fille) via la méthode setLargeur(). La méthode getLargeur() retourne la largeur actuelle de la liste.



2.7 COMPOSANT CALENDRIER

2.7.1 CALCUL DE LA SEMAINE

MAE_2532

La première semaine de l'année est celle contenant le premier jeudi de l'année et non pas le lundi comme implémenté précédemment. Pour assurer cette modification, la méthode `calculerNumeroSemaine(myDate)` du fichier `jsclient/ergonomique/fw_calendrier.js` calcule désormais le numéro du premier jeudi de la semaine, puis affecte le numéro de semaine `n` aux jours du jeudi au dimanche et `n+1` de lundi au mercredi.

La vérification de bon fonctionnement peut se faire dans la maquette dans Articles/Saisie, en cliquant sur l'icône du calendrier, on peut comparer par exemple les mois de janvier 2009 et 2010, en 2009 la semaine 1 commence dès le 01/01, en 2010, elle débute le 04/01.

2.8 AUTRES EVOLUTIONS

2.8.1 VALORISATION DES ELEMENTS DE FORMULAIRE

Les classes JavaScript représentant des éléments de formulaire liés à une source XML héritent tous de la classe *DefaultXml* et *DefaultElementForm*. La méthode de valorisation des éléments de formulaire simple *setXmlDataMono* (de la classe *DefaultXml*) a été mise à jour pour utiliser la méthode *setValue* (de la classe *DefaultElementForm*) plutôt que de valoriser la propriété *value* des éléments de formulaires.

Cette évolution permet de profiter pleinement du modèle objet pour que chaque élément de formulaire implante son propre modèle de représentation de sa valeur.

2.8.2 TACHE ANT LIVRAISON DEVELOPPEMENTLIGHT

MAE_2869

Un nouvelle tâche ANT a été ajoutée au fichier build.xml de la maquette et du template : *LivraisonDeveloppementLight*. Comme son nom l'indique, c'est une version allégée de la tâche ANT *livraisonDeveloppement*. Elle effectue seulement le remplacement des fichiers XML, JS, CSS et HTML propre à l'appliication sur le serveur de développement. Les tâches d'optimisation de la taille des scripts (suppression des commentaires, etc....) sont également écartées lors du lancement de cette tâche.

2.8.3 GESTION DU FORMAT D'UN NUMERO DE TELEPHONE

MAE_2244

Le format d'un nombre géré par la classe « NumTelFormate » se décompose en cinq sous parties délimitées chacune par un jeu de crochets [] :

[Prefixe] [Motif du numéro] [Post fixe] [Options]

Caractères proscrits: []

Format du préfixe:

Toute chaîne de caractères ne comportant pas les caractères proscrits.

Exemple : (+33)

Format du motif du numéro:

A la manière d'Excel, une chaîne de caractères composée de 0 représentant le motif périodique d'affichage du numéro:

Exemple : 0.00 pour un numéro de téléphone au format français.

Format du postfixe:

Toute chaîne de caractères ne comportant pas les caractères proscrits.

Exemple : (0.5€/appel)

Format des options :

[nbCarOmis]

- nbCarOmis : Nombre de caractères(ou chiffres) à omettre à partir du début du numéro.

Exemples:

- Pour obtenir le résultat 05.61.62.63.64 à partir de la chaîne source : 0561626364

[] [0.00] [] []

- Pour obtenir le résultat +335.61.62.63.64 à partir de la chaîne source : 05 61 62 63 64

[+33] [0.00.00.00.00] [] [1]

- Pour obtenir le résultat +33 833 444 555 (45 ct€/min) à partir de la chaîne source : 08/33/44/45/55

[+33] [0 000] [(45 ct€/min)] [1]

- Pour obtenir le résultat 05/61/62/63/64 à partir de la chaîne source : +33561626364

[0] [0/00/00/00/00] [] [3]

Un exemple d'utilisation a été implémenté dans la page de la maquette *Articles>Saisie*.

2.8.4 COMPOSANT BULLE

MAE_3371

Le composant *Bulle* permet d'associer à un composant de type « *ElementForm* » une bulle d'aide réagissant au survol de la souris sur l'élément du formulaire. Des exemples d'utilisation sont présents dans la maquette au niveau de l'onglet « *Prix Unitaire* » de la consultation des articles (champs des montants), de l'onglet « *description* » (champs *référence*, *désignation*, *unité* et *caractéristiques*) et de la page saisie des articles, sur les boutons d'aide des champs *référence* et *unité*.

Le contenu associé à un *ComposantBulle* est au format HTML, spécifié par l'utilisateur du composant.

Il existe sept styles de bulles :

- Une bulle suivant le curseur de la souris lors du survol de l'*elementForm* (TYPE_BULLE_FLOTTANTE)
- Une bulle s'affichant dans le coin supérieur gauche de l'*elementForm* lorsque le curseur de la souris le survol (TYPE_BULLE_HAUT_GAUCHE)
- Une bulle s'affichant dans le coin supérieur droite de l'*elementForm* lorsque le curseur de la souris le survol (TYPE_BULLE_HAUT_DROITE)
- Une bulle s'affichant dans le coin inférieur droite de l'*elementForm* lorsque le curseur de la souris le survol (TYPE_BULLE_BAS_DROITE)
- Une bulle s'affichant dans le coin inférieur gauche de l'*elementForm* lorsque le curseur de la souris le survol (TYPE_BULLE_BAS_GAUCHE)
- Une bulle dirigée vers la droite s'affichant au milieu de l'*elementForm* lorsque le curseur de la souris le survol (TYPE_BULLE_MILIEU_DROITE)
- Une bulle dirigée vers la gauche s'affichant au milieu de l'*elementForm* lorsque le curseur de la souris le survol (TYPE_BULLE_MILIEU_GAUCHE)

Une bulle pourra être directement associée à un *ElementForm* via la méthode *setBulleInfo* de la classe *DefaultElementForm* (voir les évolutions sur le composant *DefaultElementForm*). Il est également possible de définir dans quelle partie du composant la bulle devra intervenir (la boîte ou le bouton d'aide de l'élément de formulaire).

L'implémentation de ce composant est située dans le fichier *technique/fw_bulle.js* du Framework.

2.8.5 COMPOSANT DEFAULTELEMENTFORM

MAE_3371

La classe *DefaultElementForm* a été modifiée afin de pouvoir y rattacher un *ComposantBulle* (méthode *setBulleInfo*). De plus, il est possible de spécifier la partie du composant qui sera concernée par cette bulle (via la méthode *setPositionBulleInfo*):

- Le composant *ElementForm* lui-même (sa boîte) : POS_BULLE_BOITE
- Le bouton d'aide associé à l'*ElementForm* : POS_BULLE_AIDE

Un exemple d'utilisation est présent dans la maquette au niveau de la page « Saisie -> Article » de la maquette (voir les éléments *référence* et *unité*).

2.8.6 COMPOSANT DEFAULTXML

MAE_2301

La classe *DefaultXML* a été modifiée afin de pouvoir y rattacher une donnée nulle à partir d'un flux XML. En effet, lorsqu'une donnée avait une valeur nulle (0 ou 000000), ce composant l'interprétait comme inexistante. Désormais, la valeur est vu comme étant la chaîne « 0 » ou « 000000 ».

2.8.7 AMELIORATION DE LA TACHE DE REMPLACEMENT

MAE_2782

La tâche ANT « remplacement » (ou target) des fichiers *build.xml* de la maquette et du projet *template* a été revue afin d'améliorer son utilisation et ses performances. Cette tâche a pour objectif de remplacer tous les tags `@UN_TAG_XXX@` utilisés dans les fichiers du Framework et du projet *ACube*. L'équivalence entre un tag et sa valeur est donnée dans le fichier *build.properties* sous la forme :

```
build.HtmlConnexion=URL_HTML_HtmlConnexion  
build.HtmlConnexion_url=/html/connexion/connexion.html
```

Ici, le tag `@URL_HTML_HtmlConnexion@` sera remplacé par la valeur `/html/connexion/connexion.html` grâce à l'instruction

```
<filter token="${build.HtmlConnexion}" value="${build.HtmlConnexion_url}" />
```

dans la tâche de remplacement du fichier *build.xml*.

L'amélioration de ce fonctionnement entraîne deux modifications profondes :

- La création des fichiers *projet.xml* et *projet.properties*

L'objectif ici est d'éviter le mélange entre les correspondances tag/valeur du *Framework* et celles du projet *ACube* client en cours. Ainsi, les correspondances tag/valeur propres au projet *ACube* (*URLs* des fichiers *HTML* et de leur *script JS* rattaché...) devront désormais être présentes dans le fichier *projet.properties*. L'instruction de remplacement correspondante (*filter token*) devra apparaître dans la tâche *remplacementProjet* du fichier *projet.xml*.

De plus, les tâches *initLivraisonDeveloppement*, *initMajDemo* et *initLivvable* sont déplacées dans le fichier *projet.xml* afin que l'utilisateur puisse définir ses propriétés liées aux modes de déploiement (livraison développement sur serveur apache, livraison sur le serveur de démo ou constitution d'une archive de livraison au format .zip).

- L'amélioration de l'utilisation des *filters token*

L'essentiel des *filter token* des projets *ACube* concerne des couples tag/valeur de la forme :

```
build.ABCDEF=URL_ABCDEF  
build.ABCDEF_url=/html/connexion/connexion.html
```

et dont le *filter token* correspondant ressemblait à :

```
<filter token="${build.ABCDEF}" value="${build.ABCDEF_url}" />
```

Désormais, la tâche de remplacement est capable de repérer les couples tag/valeur de type *URL* (dans les fichiers *build.properties* et *projet.properties*) et d'effectuer les remplacements qui s'imposent. Ainsi, il devient inutile d'ajouter un *filter token* dans la tâche *remplacementProjet* du fichier *projet.xml* lorsqu'il s'agit d'un couple tag/valeur de type *URL*.



2.8.8 MISE A JOUR DES ILLUSTRATIONS DE LA JSDOC DU FRAMEWORK

MAE_2774

Les tags de la *JSDoc* du Framework ont été repris pour tous les composants graphiques afin de vérifier qu'ils étaient correctement illustrés de la manière suivante :

- Diagramme UML.
- Impressions d'écran (rendu IHM).
- Exemple de flux XML (de données ou de configuration).
- Exemple complet et illustré en popup pour les composants complexes.

2.8.9 RECUPERATION DES PARAMETRES DANS UNE URL DE TYPE GET

MAE_2617

Il a été demandé de fournir une classe utilitaire permettant de fournir des fonctionnalités quant à la manipulation des *URL* de type *GET* : *http://mon.url.com ?unParam=xxxx&unAutreParam=aaaa*.

La classe *RequeteGet* a été créée dans le fichier *fw_reqtest.js* du Framework technique. Un exemple de manipulation de cette classe est présent dans la *JSDoc* associée ainsi qu'une explication de toutes ces méthodes. Une application directe est également disponible dans la maquette : après avoir choisi le menu développement, *Explications* -> *Requete GET*.

L'outil *RequeteGet* permet, à partir d'une *URL* de type *GET*:

- De tester l'existence d'un paramètre : *estParam()*.
- De retrouver la valeur d'un paramètre : *getParam()*, *getTabParams()*.
- De modifier les paramètres : *setParam()*, *setTabParams()*, *supprimerParam()*, *viderParams()*.
- De répercuter ces modifications sur l'*URL* : *getUrl()*.

2.8.10 LIBELLES D'ERREURS EN ESPAGNOL

MAE_3079

Les codes d'erreur espagnols ont été rajoutés dans le fichier *fw_xml.js* du framework. Ils sont nommés *_ES, ils correspondent à ceux français ou anglais avec à la fin du libellé la mention « (à traduire en espagnol) ».

Ces codes ont également été ajouté dans la page *IncludesJSDocA3/constant-values_A3.tmpl* pour apparaître dans les constantes définies dans la *JSDoc* du Framework Ergo.

2.8.11 COMPOSANT FILE ET ELEMENT FORM FILE

MAE_2973

La classe Javascript *ComposantFile*, représentant un composant graphique *Acube* pour la personnalisation du bouton de téléchargement d'un champ *input file*, a été modifiée afin de fonctionner malgré les problèmes de sécurité avec les navigateurs de type *Gecko*.



De plus, la balise <FORM> correspondante au formulaire en cours est automatiquement mise à jour (*enctype=multipart/form-data*).

Il en va de même pour l'élément de formulaire *ElementFormFile*.

2.9 EXEMPLES SUPPLEMENTAIRES IMPLEMENTES

2.9.1 INITIALISATION A ZERO

Dans la page Articles>Saisie, la zone de texte du prix unitaire a été initialisée à zéro pour preuve de bon fonctionnement.

```
var tmpPrix=getDefaultElementById("prix");  
tmpPrix.setValue(0);
```

2.9.2 UN MENU CONTENANT DES ITEMS DE NIVEAU 3

MAE_2272

Le menu a été modifié dans la partie « intervention du ministre » afin que les allocutions apparaissent désormais en niveau 3 de menu. Cela a permis de voir le bon fonctionnement des items de niveau 3.

Actions : modification du flux XML de la maquette : flux/protected/frameset/menu.xml pour ajout d'items de niveau 3. Création de la page html de niveau 2.

2.9.3 UN DEUXIEME MENU

MAE_2272

Il a été créé un « menu de développement » accessible depuis une zone de sélection définie au début de la page de menu.

Ce deuxième menu a permis de mettre en valeur l'erreur de fonctionnement du à la recherche par numéro d'un item de menu stocké dans le cookie et non répertorié dans le menu proposé (ce cas pourrait par exemple se trouver dans le cas de connexions avec différents droits d'accès sur un même poste : des items de menu ne sont plus proposés).

Actions :

- Dans le fichier du Framework jsclient/ergonomique/fw_navigation.js, modification de la méthode restoreFromCookie de la classe ComposantMenu : quand on récupère les numéros stockés dans le cookie, on teste s'ils existent dans le menu courant. Si c'est le cas, on les affichera, sinon, on affichera soit l'arborescence ouverte au niveau inférieur trouvé, soit la page d'accueil (dans ce dernier cas, appel de la méthode `this.SelectAccueil(false)`).
- Dans le fichier de la maquette jsclient/frameset/framesetter.js, import du nouveau menu, et affichage de l'un ou l'autre par implémentation de la fonction `onChangeMenu()` ouverte depuis la selectBox en début de page menu.
- Création d'un fichier XML de la maquette : flux/protected/frameset/menuDeveloppement.xml pour le nouveau menu.
- Création des pages html associées à ce menu (pages génériques, seule différence : les titres des pages de niveau 1 commencent par « 1 », les autres titres ne comportent pas de numéro).



2.9.4 TABLEAU A DEUX COLONNES : STRING ET DATE

MAE_2806

La page Articles/Catalogue Simplifié a été créée dans la maquette. Elle comprend des données de la page de catalogue, avec seulement des désignations et des dates. Pour cette réalisation, un item de menu a été ajouté dans flux/protected/frameset/menu.xml. Cet item a été associé aux pages html/article/catalogueSimplifie.html et jsclient/article/catalogueSimplifie.js. Le flux de données est implémenté dans flux/protected/article/ListeArticleParCritereSimplifie.xml associé au flux de description xml/article/catalogueSimplifie.xml.

Dans le fichier javascript, la fonction de modification a été réalisée de façon à pouvoir modifier un élément du catalogue dans le ComposantTableau qui lui est associé.

Lorsque nous plaçons la colonne des désignations en non trié, puis que nous trions la colonne de date (par exemple par ordre décroissant), nous pouvons ensuite réaliser une modification sans conséquence sur les tris demandés précédemment sur les colonnes.

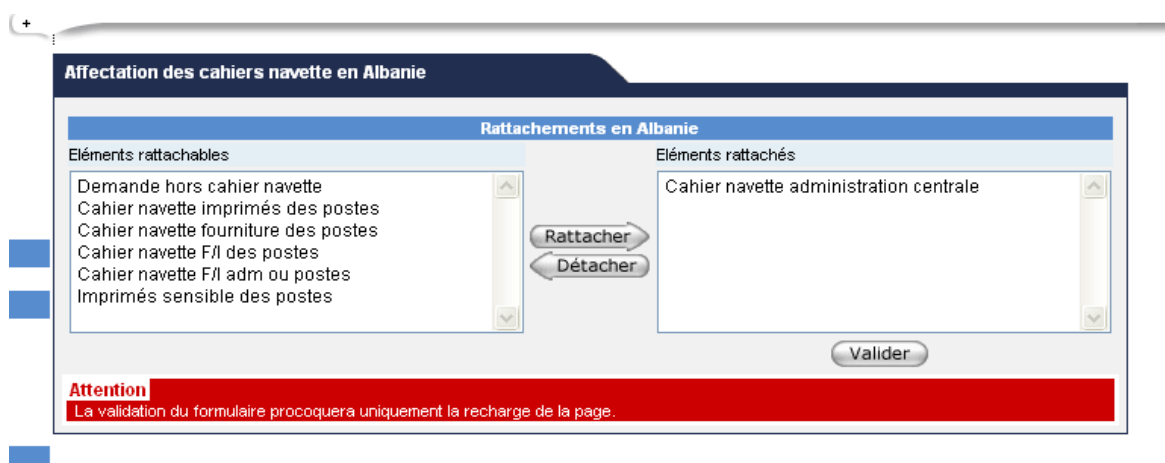
2.9.5 COMPOSANT RATTACHEMENT : LIBELLES DANS LE CONSTRUCTEUR, AFFICHAGE ET NOUVELLES FONCTIONNALITES

MAE_3041

Implémentation dans la maquette d'un nouvel exemple de composant rattachement accessible dans le menu Cahier Navette/Affectation Albanie. Cet item de menu a été ajouté dans menu.xml. Il donne accès à la page affectationAlbanie.html elle-même liée à affectationAlbanie.js (mise à jour du fichier projet.properties).

Cette page crée un ComposantRattachement à partir d'un tableau d'ObjectOptions et non d'un fichier xml. Les libellés sont entrés directement en paramètre du constructeur. Le composant rattachement s'affiche et fonctionne correctement :

```
var cahier=new ComposantRattachement("cahier",null,tabOption,"Rattachements en Albanie","Eléments rattachables","Eléments rattachés");
```



Un nouvel exemple de ComposantRattachement est accessible depuis la jsdoc. Il reprend l'implémentation décrite ci dessus (page rattach2.html, accessible depuis le lien dans la documentation du ComposantRattachement).

MAE_3042



Grâce aux méthodes `setLargeurListeGauche()` et `setLargeurListeDroite()`, il est désormais possible de définir la largeur des listes de gauche et de droite du composant rattachement.