

NATJXT : COMPOSANTS RICHES ET DRAG AND DROP

Les années 90 ont marqué l'avènement de la technologie Client/ Serveur (client lourd), controversée en raison de la lourdeur du déploiement, mais également de celle du navigateur Internet, élément central du Web (client léger).

Une évolution récente, appelée Client Léger Riche, permet de combiner ces 2 architectures. Il s'agit bien de tirer parti des avantages du Client/Serveur et de les associer à ceux du client Web.

Du client lourd au client léger

La « webisation » d'applications Client/ Serveur offre de réels avantages :

- Minimisation des coûts d'intégration
- Disparition de la complexité du déploiement
- Maintenance réduite au minimum
- Qualités accrues : souplesse, modularité, évolutivité et réel découplage entre le poste de travail et le serveur.

D'autre part, les applications Web sont multi-plateformes contrairement au Client/Serveur : en tant que client léger, un browser Internet ne remet pas en cause la capacité des postes utilisateurs.

La mise à jour d'une WebApp est centralisée sur le serveur uniquement, et de ce fait simplifiée, notamment pour les grands comptes.

Le Client Léger Riche

En réponse à une logique purement économique, le Client Léger Riche permet également de tirer profit du meilleur du Web et du Client/Serveur :

- Des écrans dotés de composants évolués
- L'enchaînement d'écrans complexes
- La gestion des profils utilisateurs
- La gestion des listes longues
- Des interfaces adaptées à la saisie intensive (formulaires)
- Des raccourcis clavier.

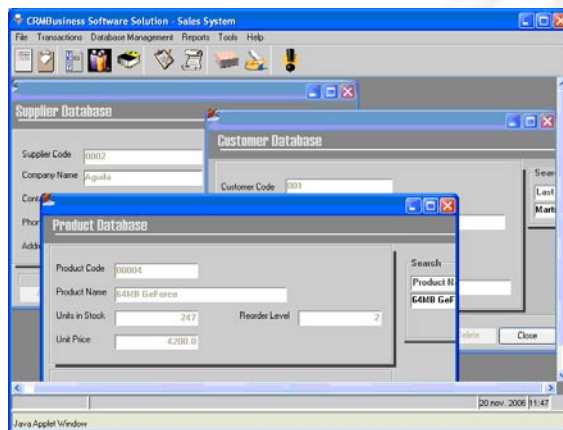


Figure 1 : Exemple de Client Léger Riche

Le GuiBuilder de NatJxt

NatJxt met à disposition du développeur Web un plugin Eclipse permettant de concevoir graphiquement les différents formulaires de son application :

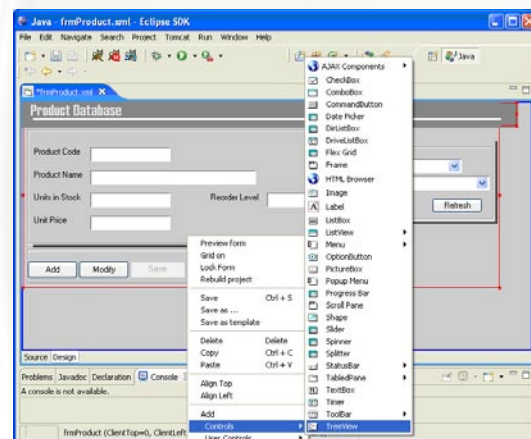


Figure 2 : Le plugin JxtGuiBuilder

Les composants d'un formulaire et leurs propriétés sont stockés dans des fichiers XML échangés entre le serveur et le client pour la construction des écrans, que ce soit pour la cible JXT (applet générique) ou bien pour la cible AJAX (JavaScript). La trentaine de composants graphiques supportés par GuiBuilder comprend des composants basiques (Label, TextBox, Combo, ListBox...) mais aussi d'autres, conformes au Web V2 :

- Menu et PopupMenu
- ToolBar
- TabbedPane
- Spinner
- Splitter
- ListView
- TreeView

- Flex Grid
- Date Picker
- Flash Container

Le contrôle ListView

Ce contrôle permet de représenter un ensemble d'items sous forme de tableau.

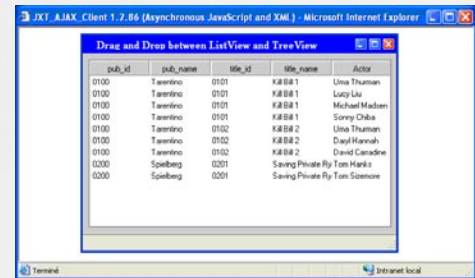


Figure 3 : Le contrôle ListView

Propriétés remarquables :

- AllowColumnReOrder : Permet le tri par colonne
- Anchor : Dimensionnement automatique
- GridLines : Contour graphique des cellules
- MultiSelect : Sélection de lignes multiples
- Opaque : Transparence avec le conteneur
- PopupMenu : Menu contextuel par ligne.

Peuplement d'une ListView

```
int cpt = 0;
IVBListItem item = null;

item = this.get_ListView().
getListItems().Add(cpt++,String.
valueOf(cpt),»0100»);
item.setSubItems(0,»Tarentino»);
item.setSubItems(1,»0101»);
item.setSubItems(2,»Kill Bill 1»);
item.setSubItems(3,»Uma Thurman»);

item = this.get_ListView().get-
ListItems().Add(cpt++,String.
valueOf(cpt),»0100»);
item.setSubItems(0,»Tarentino»);
item.setSubItems(1,»0101»);
item.setSubItems(2,»Kill Bill 1»);
item.setSubItems(3,»Lucy Liu»);
```

Le contrôle TreeView

Ce contrôle permet de représenter un ensemble d'items sous forme de structure hiérarchique.

Propriétés remarquables :

- Anchor :
- Dimensionnement automatique
- DefaultLeafIcon :
- DefaultNodeIcon :
- Opaque :
- Transparente avec le conteneur
- PopupMenu :
- Menu contextuel par ligne

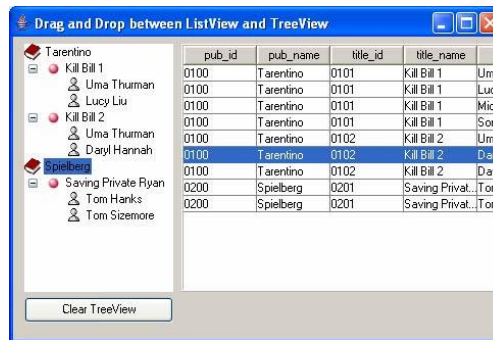


Figure 4 : Le contrôle TreeView et le Drag and Drop

Le Drag and Drop

Il permet de glisser et de déposer des éléments dans la page. Dans le cadre d'un site marchand par exemple, il permet une personnalisation simple et ludique de l'interface et des interactions plus visuelles, apport nouveau dans les applications Web professionnelles.

Mise en œuvre : Drag and Drop de la ListView vers la TreeView.

- Activer dans la fenêtre de propriétés du TreeView l'événement **Event_Drop**
- Surcharger la méthode **TreeView_Drop**

Surcharge de la méthode TreeView_Drop

```

public void TreeView_Drop(JxtComponentID compid,
IVBObject source, IVBObject target, boolean isMove) {
    boolean t1 = false, t2 = false;
    IVBNode nr = null, nt = null, na = null;

    String pub_name = ((IVBListItem) source).
getSubItems().get(0).toString();
    String title_name = ((IVBListItem) source).
getSubItems().get(2).toString();
    String actor = ((IVBListItem) source).
getSubItems().get(3).toString();

    ArrayList array = this.get_TreeView().
getNodes().getRootNodes();

    Iterator it = array.iterator();
    while(it.hasNext()){
        nr = (IVBNode) it.next();
        if(nr.getText().equals(pub_name)){
            t1 = true;
            break;
        }
    }
    if(t1){ // RootNode found
        Iterator it2 = nr.getChildren();
        t2 = false;
        while(it2.hasNext()){
            nt = (IVBNode) it2.next();
            if(nt.getText().equals(title_
name)){
                t2 = true;
                break;
            }
        }
    }
    if(t2){
        // Title found
        // Adding Actor
        addActor(nt, actor);
    }
    else{
        // Title not found
        // Adding Title
        nt = addTitle(nr, title_name);
        // Adding Actor
        addActor(nt, actor);
    }
}

public IVBNode addAuthor(String pub_name){
    IVBNode nr = null;
    nr = this.get_TreeView().getNodes().
newNode(pub_name, pub_name);
    nr.setImage(this.getApp().
loadPicture («Support/ui/images/book.gif»));
    this.get_TreeView().getNodes().
addRootNode(nr);
    return nr;
}

public IVBNode addTitle(IVBNode nr, String title_
name){
    IVBNode nt = null;
    nt = this.get_TreeView().getNodes().
newNode(title_name, title_name);
    nt.setImage(this.getApp().
loadPicture («Support/ui/images/redball.gif»));
    nr.add(nt);
    return nt;
}

public void addActor(IVBNode nt, String actor){
    IVBNode na = null;
    na = this.get_TreeView().getNodes().
newNode(actor, actor);
    na.setImage(this.getApp().
loadPicture («Support/ui/images/person.gif»));
    nt.add(na);
}
    
```