

# Présentation du framework Rialto

par B. Le Roux ([b-le-roux.developpez.com](http://b-le-roux.developpez.com))

Date de publication : 25/02/2008

Dernière mise à jour :

Rialto est un framework orienté vers les applications de gestion. Cet article montre la mise en oeuvre des principaux widgets.

---

I - Généralités.....	3
II - Présentation de Rialto.....	4
III - Organisation de Rialto.....	5
IV - Installation.....	6
V - Première page avec Rialto.....	7
V-A - Les déclarations.....	7
V-B - L'écriture du code en Javascript.....	7
VI - Les widgets.....	9
VI-A - Les conteneurs.....	9
VI-A-1 - Splitter.....	9
VI-A-2 - Tableau à onglets.....	10
VI-A-3 - Form.....	10
VI-A-4 - Pop-up.....	12
VI-A-5 - Frame.....	13
VI-B - Les composants visuels.....	14
VI-B-1 - TreeView.....	14
VI-B-2 - Grid.....	15
VI-B-3 - GridTreeView.....	17
VI-C - Le modèle objet.....	17
VII - Ajax.....	19
VIII - L'apparence.....	20
IX - Gestion des langues.....	21
X - Rialto studio.....	22
XI - Les sous projets Rialto.....	23
XII - Résumé.....	24
XIII - Références.....	25

## I - Généralités

L'arrivée du Web 2,0 s'est accompagné du développement de technologies comme Ajax demandant une mise en oeuvre plus complexe que la constitution de simple page HTML. C'est ainsi que nous avons vu l'apparition de nombreux frameworks ou bibliothèques souvent écrits en Javascript facilitant l'utilisation de ces techniques. Ces différents frameworks couvrent des périmètres de fonctionnalités allant de l'ensemble des besoins à seulement un aspect. On peut en citer quelques frameworks : Rialto, ExtJS, Dojo, etc. Bien sûr, leurs mises en oeuvre sont plus ou moins complexes, Je vous propose de vous présenter le framework Rialto (**R**ich **I**nternet **A**pplication **T**oolkit).

## II - Présentation de Rialto

Rialto est un framework Javascript orienté vers les applications de gestion proposant un ensemble de composants d'IHM pouvant utiliser Ajax. Ce framework est compatible avec FireFox et Internet Explorer. Initialement, Rialto a été conçu à l'Institut Gustave Roussy (IGR) pour des besoins internes de développement. Puis, Rialto a été placé sous licence Apache et est actuellement activement soutenu principalement par Cyril Balit (Improve) et par François Lion (IGR).

### III - Organisation de Rialto

Rialto est divisé en trois parties.

La première partie contient l'ensemble des widgets permettant d'élaborer les IHM d'applications Web.

L'ensemble des techniques se rapportant à AJAX forme la deuxième partie.

La troisième rassemble un ensemble de bibliothèques facilitant l'utilisation du framework Rialto : trace, debuggage avec firebug de FireFox, internationalisation, manipulation de chaîne de caractères, de date, du DOM, le drag and drop, etc.

## IV - Installation

Étape 1 :

télécharger le framework Rialto : <http://rialto.improve-technologies.com/wiki/rialto/download>

Étape 2 :

Déposer le répertoire RialtoEngine dans l'arborescence de votre site

Étape 3 :

Configuration de config.js (cf Configuration de Rialto)

### Configuration de Rialto

```
var rialtoConfig = {version : 0.9,  
1 isDebug : false,  
2 TraceLevel : 1,  
  IsTestVersion : false,  
3 Language : 'en'  
};  
if (document.location.protocol == "http:" || document.location.protocol == "https:"){  
  rialtoConfig.isFilePath=false;  
4 rialtoConfig.pathRialtoE = '/monsite/rialtoEngine/';  
}
```

1 : paramétrage du mode debug,

2 : paramétrage du mode trace,

3 : paramétrage de la langue : en, fr, de,

4 : chemin de Rialto sur le serveur Web.

Voilà, Rialto est paramétré. Nous pouvons désormais commencer à l'utiliser et créer notre première page avec Rialto.

## V - Première page avec Rialto

Je vous propose d'écrire notre première page avec Rialto. L'objectif (traditionnel) est d'afficher les mots « Hello World ». Pour cela, nous allons procéder par étapes :

### V-A - Les déclarations

Pour pouvoir utiliser le framework Rialto, il faut déclarer dans une page HTML les différentes feuilles de style, le fichier de configuration (config.js) et Rialto.js.

#### Déclarations

```
<link rel='STYLESHEET' type='text/css' href='./rialtoEngine/style/rialto.css'/>
<link id = 'standart_behavior' rel='STYLESHEET' type='text/css' href='./rialtoEngine/style/behavior.css'>
<link rel='STYLESHEET' type='text/css' href='./rialtoEngine/style/defaultSkin.css'>
<script type='text/javascript' src='./rialtoEngine/config.js'></script>
<script>rialtoConfig.pathRialtoE = "./rialtoEngine/";</script>
<script type='text/javascript' src='./rialtoEngine/javascript/rialto.js'></script>
```

### V-B - L'écriture du code en Javascript

Pour atteindre notre objectif, il faudra instancier trois classes (cf Premier exemple avec Rialto):

- SimpleWindow va créer une fenêtre,
- Frame va être le conteneur qui va accueillir le (ou les ) widget(s),
- Label va permettre d'afficher le texte « Hello World ».

Voici le code complet :

#### Premier exemple avec Rialto

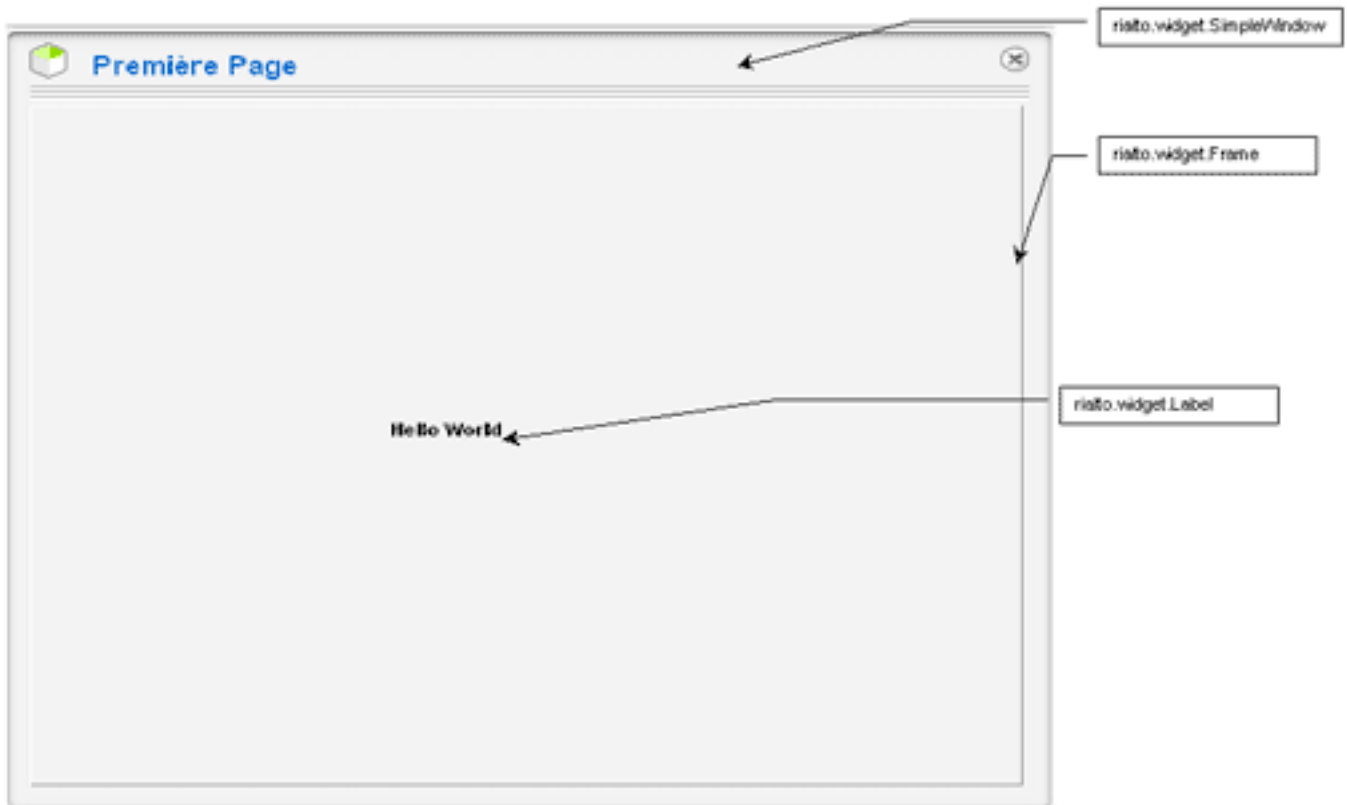
```
1 var winMain=new rialto.widget.SimpleWindow(
  {
2   title:'Première Page',
   position:"absolute",
   name:"simpleWindow",
3   parent:document.body
  });
4 var frame = new rialto.widget.Frame({
   name:'frame',
5   width:'99%',
6   height:'99%',
   title:'',
7   parent:winMain
});
8 new rialto.widget.Label(
   "hello",
9   200,
10  200,
   frame,
11  'Hello World',
12  'libellel');
```

Ce code Javascript demande quelques explications :

- 1 : instanciation de la classe SimpleWindow,
- 2 : titre de la fenêtre,
- 3 : élément parent de cet objet. Il s'agit ici de corps de page,

- 4 : instanciation de la classe Frame,
- 5, 6 : dimension ( ici en %) du Frame par rapport à son parent,
- 7 : parent du frame,
- 8 : instanciation de la classe Label,
- 9,10 : position du texte sur l'écran,
- 11 : le texte à afficher,
- 12 : nom du style utiliser pour afficher le texte (voir feuille de style).

Nous venons d'écrire notre première page avec Rialto.  
Vous trouverez ci-dessous le résultat visuel.



*Écran de Hello World*

## VI - Les widgets

Nous venons de voir le principe de mise en oeuvre de Rialto de trois composants de Rialto. Rialto compte un grand nombre de widgets. Les widgets de Rialto peuvent être classés en deux catégories souvent rencontrées dans les interfaces graphiques : les composants conteneurs et les composants visuels.

Dans la suite de ce chapitre, je vais vous présenter quel qu'uns des widgets de Rialto. Je ne détaillerais pas non plus leurs mises en oeuvre car chacun d'entre eux peut être utiliser simplement mais ils présentent de nombreuses options afin de s'adapter à nos besoins particuliers.

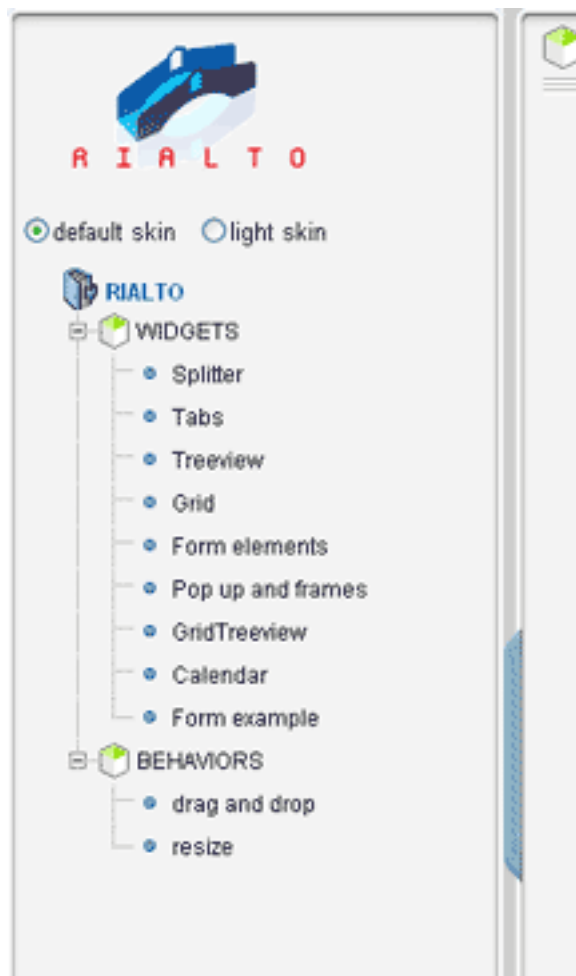
Mes présentations se contenteront de décrire les fonctionnalités les plus importantes ou les plus originales, d'une capture d'écran, et un exemple de code simple.

### VI-A - Les conteneurs

Rialto compte sept composants conteneurs. Leurs rôles est de pouvoir, comme leur nom l'indique, contenir et de positionner sur l'écran du navigateur d'autres composants, conteneurs de Rialto mais aussi d'autres widgets appartenant à d'autres bibliothèques ou des balises HTML. Je vous en présente brièvement cinq : le splitter, le tableau à onglets, le formulaire, le frame et le popup.

#### VI-A-1 - Splitter

Ce composant permet de diviser les autres conteneurs en deux parties de taille fixe ou variables. Cette division peut être horizontale ou verticale.



*Splitter vertical*

Le code Javascript ci-dessous montre la création d'un splitter avec Rialto

```
var splitPG = new rialto.widget.Splitter({
    Prop:0.20,
    orientation:'h',
    name:'splitPG',
    parent:document.body,
});
```

## VI-A-2 - Tableau à onglets

Le composant tableau à onglets est un autre conteneur de Rialto. Les onglets peuvent être situés en haut (option par défaut) , en bas , à droite, à gauche. Lorsque le nombre d'onglets devient trop important pour être affichés, le déplacement d'un onglet à un autre peut se faire par l'intermédiaire d'un gestionnaire situé à droite.



Tableaux à onglets

Le code Javascript ci-dessous montre la création d'un tableau à onglets avec Rialto.

```
1 myTabs = new rialto.widget.TabFolder({name:'myTabs',parent:fenOnglet});
2 myTabs.addTabItem("tab 1",true);
```

Ligne 1 : instantiation de la classe TabFolder

Ligne 2 : création d'un onglet

## VI-A-3 - Form

Form est un conteneur jouant un rôle similaire à la balise rencontré dans HTML. Bien sûr, ce conteneur peut contenir l'ensemble des composants présents dans Rialto et spécialement l'ensemble des composants d'affichage et de saisie : label, text (saisie de texte, numérique, date (cf Aide à la saisie d'une date), heure, mot de passe, textarea, caché, etc.), combo, checkbox, radiobutton, bouton, image, etc. De plus, le composant Form intègre les mécanismes Ajax offrant la possibilité de soumettre un formulaire par cet intermédiaire.

Formulaire contenant des simples widgets

Aide à la saisie d'une date

Le code Javascript ci-dessous montre la création d'un formulaire avec Rialto avec quelques composants de saisie :

```

1 var FORM1 = new rialto.widget.Form('FORM1','monsite',parentWin);
2 new rialto.widget.Radio(
    'T',
    7,
    10,
    FORM1,
    'INSC',
    'All',
    true,
    'libNormal');
3 new rialto.widget.Checkbox(

```

```

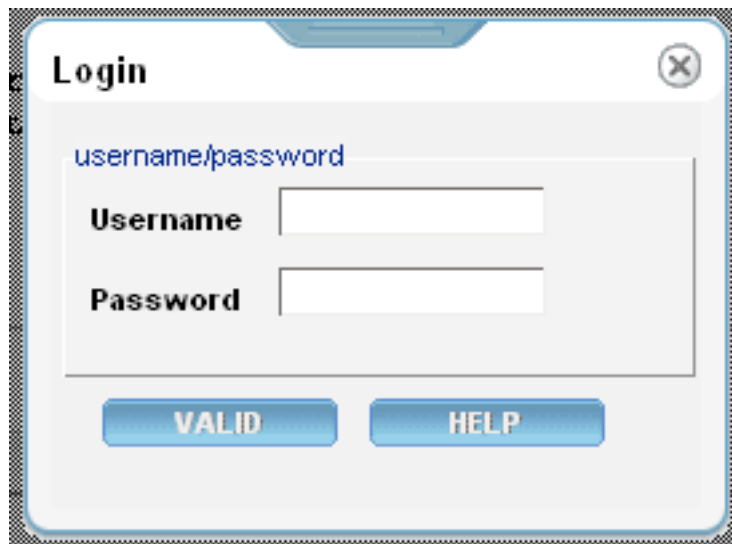
        'check1',
        5,
        7,
        FORM1,
        'check lib',
        false,
        'libelle1');
4 new rialto.widget.Combo(
    '',
    'COUNTRY',
    10,
    10,
    120,
    FORM1, {
5     suggest:true,
      enable:true,
      heightItem:0});
6 COUNTRY.addItem("En", "England");
7 COUNTRY.addItem("Fr", "France");

```

- 1 : instanciation de la classe Form,
- 2 : création d'un radio bouton,
- 3 : création d'un checkbox,
- 4 : création d'un combobox,
- 5, 6 : ajout d'items dans le combobox.

## VI-A-4 - Pop-up

Le Pop-up est un autre conteneur permettant de réaliser une fenêtre contenant des messages, une saisie (cf Pop-up de saisie) pour la saisie d'un nom d'utilisateur et un mot de passe. De nombreux autres widgets disponibles dans Rialto sont dérivées de cette classe permettant les message d'alerte, les message d'attente, fenêtre de recherche, etc.



*Pop-up de saisie*

Le code Javascript ci-dessous montre la création de ce Pop-up:

```

1 test = new rialto.widget.PopUp(
  'test',150,420,'260','150','','Login','Gris');
2 CADRE1 = new rialto.widget.Frame({
  name:'CADRE1',
  top:'20',
  left:'5',
  width:'235',

```

```

        height:'80',
        title:'username/password',
        open:true,
        position:'absolute',
        parent:test});
3 new rialto.widget.Label('lib1',15,10,CADRE1,"Username",'libelle1');
4 new rialto.widget.Text(
    'TEXT',10,80,100,'A',CADRE1,{
    rows:5});
5 new rialto.widget.Label('lib4',45,10,CADRE1,"Password",'libelle1');
6 new rialto.widget.Text(
    'PASS',40,80,100,'A',CADRE1,{
    rows:5});
7 new rialto.widget.Button(
    110,20,"VALID","Close",test,{
    enable:true,
    adaptToText:true,
    width:88,
    widthMin:88});
8 new rialto.widget.Button(
    110,120,"HELP","Show the help",test,{
    enable:true,
    adaptToText:true,
    width:88,
    widthMin:88});
    
```

- 1 : instanciation de la classe PopUp,
- 2 : création d'un Frame,
- 3, 5 : création de Label,
- 4, 6 : création de Text (zone de saisie de texte),
- 7, 8 : création de Button.

## VI-A-5 - Frame

Le conteneur Frame permet de regrouper différents composants. Il peut être statique, représenté visuellement par un simple cadre (cf la section V-B), ou bien dynamique offrant d'afficher ou de masquer les composants contenus comme ci-dessous.

The screenshot shows a dynamic frame titled "Search criteria" containing a search form. The form is organized into several sections:

- Patient id:** Includes fields for "Folder from" to "to", "Name" (with a "Phonetics" checkbox), "First name", "Sex" (dropdown), "Maiden name", "Born on" (with "Age" and "to" fields), and "Committee".
- Medical inscription:** Includes radio buttons for "All", "Provisional", "Final", and "Opinion".
- Appointment:** Includes "date of" (with "to" field) and "Act" (with a dropdown).
- Hospitalization:** Includes a checkbox for "In-patients to date", "date of" (with "to" field), and "Committed" (with a dropdown).
- Therapeutic tests:** Includes a checkbox for "Patients currently included in a test" and "N° CET" (with a dropdown).

At the bottom of the form, there is a "Numbers maximum posted patients" field set to "100", and "Reset" and "Find" buttons. Below the form is a "Results of research" section with a table header containing "Folder", "Name-First name", "Date of birth", and "Committé".

Frame dynamique ouvert contenant un formulaire



Même Frame que ci-dessus mais fermer

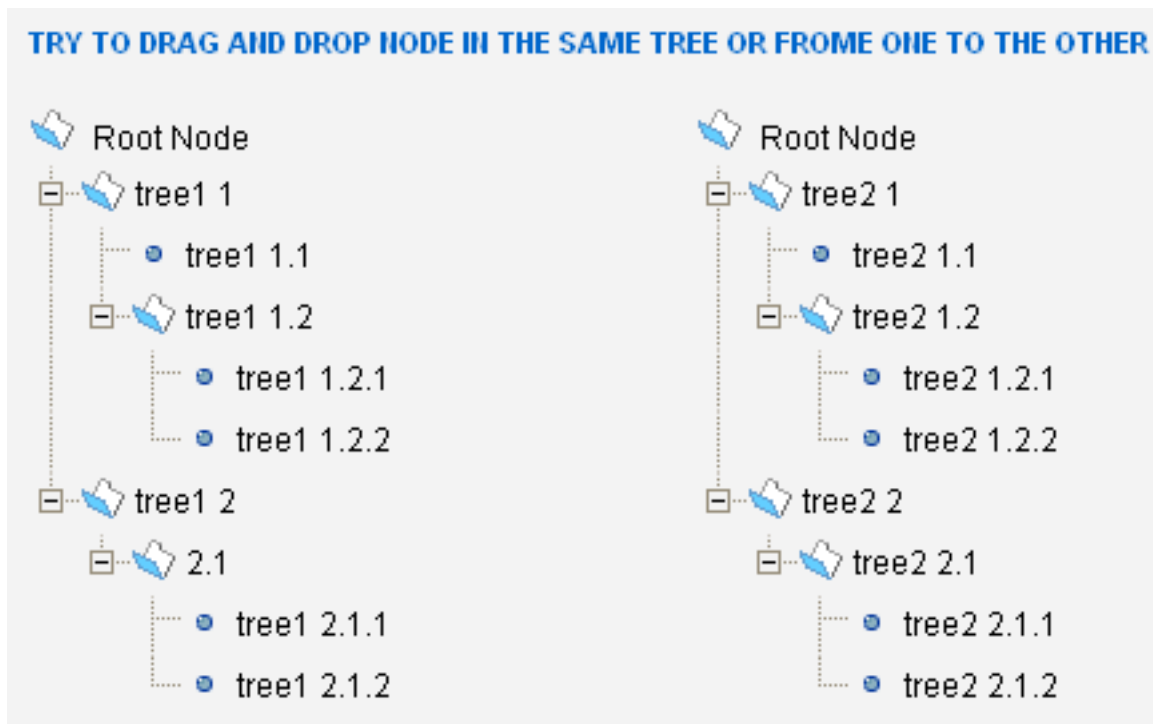
Pour rendre un Frame dynamique, il suffit d'ajouter un paramètre à l'instanciation (cf la section V-B) : `dynamic:true`

## VI-B - Les composants visuels

Nous venons de passer en revue quelques un des conteneurs de Rialto. Nous avons vu dans le paragraphe précédent (conteneur Form) que Rialto dispose d'un ensemble de widgets simples d'affichage et de saisie (label, text, etc.) possédant de nombreuses options. Je vous propose maintenant de vous présenter trois autres widgets de Rialto.

### VI-B-1 - TreeView

Le composant TreeView permet de mettre en oeuvre des arbres. Le contenu de ces arbres peut être statique (défini dans le code) ou dynamique (défini au cours de l'exécution en utilisant AJAX par exemple).



Arbres permettant le drag and drop

Le code Javascript ci-dessous montre la création d'un TreeView avec Rialto.

```

1 var treeT = new rialto.widget.Tree({
    name: 'treeReload',
    top: '5',
    left: '0',
    width: '300',
    height: '200',
    parent: CADRE5});
2 treeT.createAndAddNode(
    treeDossier.id, {
    name: 'nodeRoot',
    text: 'RIALTO',
    
```

```
icon: 'agenda.gif',  
icon2: 'agenda_ouvert.gif'});
```

- 1 : instanciation de la classe Tree,
- 2 : ajout d'un noeud dans l'arbre.

## VI-B-2 - Grid

La classe `rialto.widget.Grid` permet de créer des tableaux. C'est une classe présentant énormément de possibilités (18 paramètres possibles et 31 méthodes définies dans cette classe). Dans le cadre de cette article, je ne peux qu'énumérer quelques une des caractéristiques de cette classe :

- mode paginé ou liste,
- impression du tableau,
- colonne triable ou non,
- sélection de cellule ou de ligne,
- simple ou multi sélection de ligne ou de cellule,
- saisie de cellule,
- insertion d'élément HTML,
- etc.

WRITABLE GRID just doubleclick on a cell...

Folder	Surname-First	Date of birth
1933-00001	DUPONT Pierre	05/11/1974
1933-00001	DUPONT Pierre	05/11/1974
✓ 1933-00001	DUPONT Pierre	05/11/1974
1933-00001	DUPONT Pierre	05/11/1974
1933-00001	DUPONT Pierre	05/11/1974

Grid with multiselect lines

Folder	Surname-First	Date of birth
1933-00001	DUPONT Pierre	05/11/1974
✓ 1933-00001	DUPONT Pierre	05/11/1974
1933-00002	DURAND Jule	05/11/1974
1933-00003	RENAUT Eric	05/11/1974
1933-00004	SMITH John	05/11/1974

*En haut Grid format page permettant la saisie, en bas Grid format liste*

Le code Javascript ci-dessous montre la création d'un Grid avec Rialto

```

1 oTableau = new rialto.widget.Grid({
    top:10,
    left:5,
    height:250,
2   TabEntete:["Folder","Surname-First name","Date of birth"],
    name:'oTableau',
    parent:CADRE1,
3   tabTypeCol:[["string",80],["string",100],["date",100]]);
4 oTableau.fillGrid([
    ["1933-00001","DUPONT Pierre","05/11/1974"],
    ["1933-00002","DURAND Jule","27/08/1942"],
    ["1933-00003","RENAUT Eric","01/11/1954"],
    ...]);

```

1 : instanciation de la classe Grid,

- 3 : définition des libellés des colonnes,
- 6 : définition des caractéristiques des colonnes : type et largeur,
- 7 : initialisation du tableau avec des données statiques.

## VI-B-3 - GridTreeView

Illustration ci-dessous montre un GridTreeView qui est une combinaison entre arbre et un tableau (Grid). Dans ce type de composant, nous disposons ainsi d'informations se rapportant à des noeuds d'un arbre sous forme de tableau.



	Folder	Surname-First	Date of birth
1			
2			
2.1	1933-00002	DURAND Jule	27/08/1942
2.2	1933-00002	DURAND Jule	27/08/1942
2.3	1933-00002	DURAND Jule	27/08/1942
2.4	1933-00002	DURAND Jule	27/08/1942

Exemple de GridTreeView

Le code Javascript ci-dessous montre la création d'un GridTreeView avec Rialto.

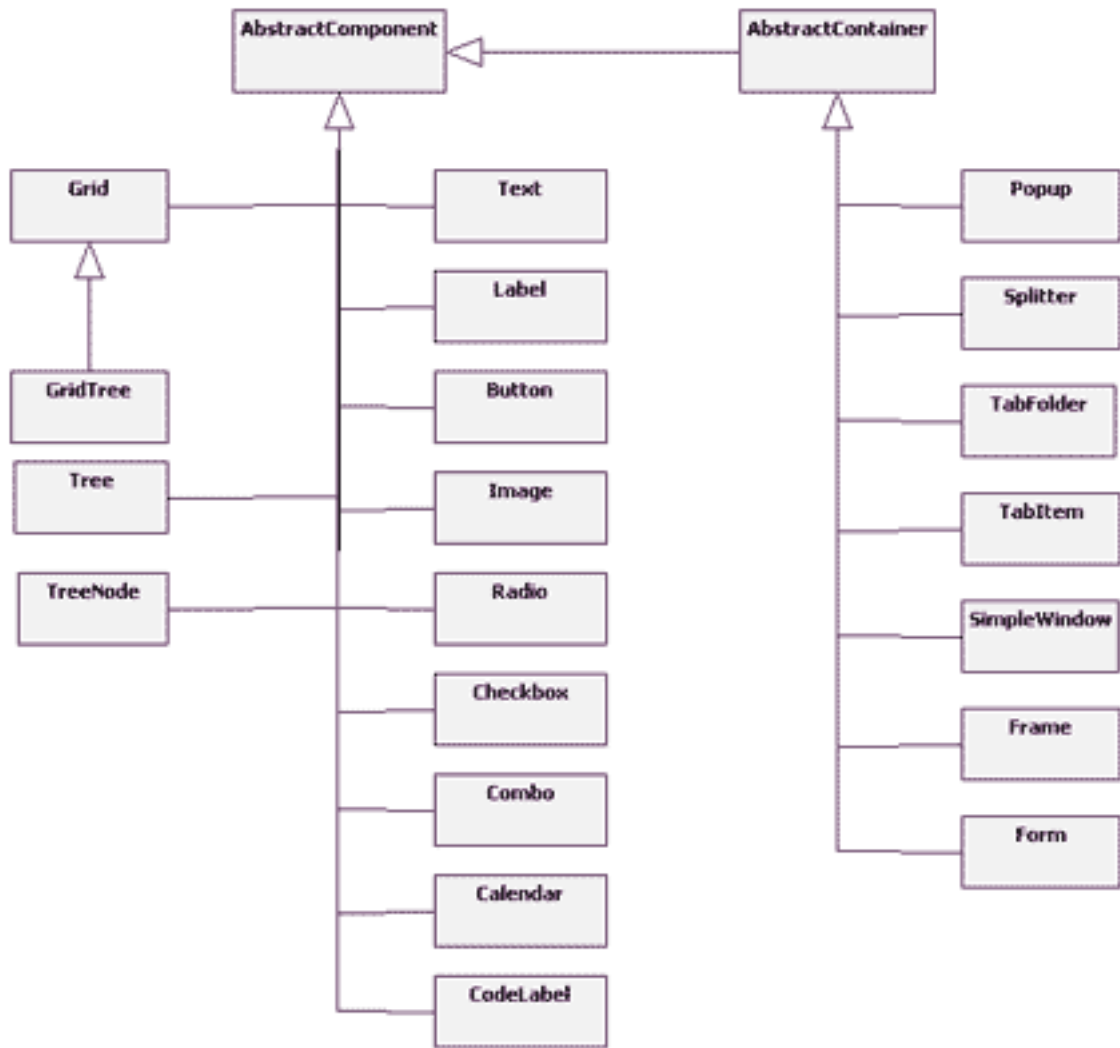
```

1  oTableau = new rialto.widget.GridTree( {
      widthFirstCol:220,
      top:10,
      left:5,
      height:400,
2  TabEntete:["Folder", "Surname-First name", "Date of birth"],
      name:'oTableau',
      parent:frmGridTreeview,
      rang:5,
3  tabTypeCol:[["string",80],["string",100],["date",100]];
4  var li2 = oTableau.addNodeLine( {name:'li2', text:'1'} );
5  oTableau.addNodeLine( {
      name:'li1',
      text:'1.1',
      parentLine:li2,
      tabData: ["1933-00002", "DURAND Jule", "27/08/1942"],
      open:false});
    
```

- 1 : instanciation de classe GridTree,
- 2 : définition des libellés de colonnes,
- 3 : description des colonnes : type et largeur,
- 4 : ajout d'un noeud dans oTableau,
- 5 : ajout d'un noeud dans oTableau mais étant fils du noeud défini en ligne 9 et définissant des données (cf ligne12).

## VI-C - Le modèle objet

Tous les composants de Rialto sont organisés et basés sur deux classes abstraites (cf Modèle objet de Rialto). AbstractComponent est la classe dérivée des classes concrètes des composants visuels et d'une autre classe abstraite AbstractContainer. Cette dernière est la classe dérivée des classes des conteneurs de Rialto. AbstractComponent définit un certains nombres de méthodes permettant de gérer le positionnement du composant sur l'écran, son aspect : taille, couleur, style etc. AbstractContainer permet aux conteneurs de leur ajouter ou de leurs supprimer des composants.



Modèle objet de Rialto

Ce modèle objet permet d'entreprendre l'extension des widgets existants ou d'en écrire d'autres en se basant sur ces deux classes abstraites. De plus, Rialto permet d'utiliser d'autres bibliothèques graphiques (l'éditeur WYSIWYG Tiny mce, etc.), d'autres composants graphiques par exemple du SVG ou des balises HTML.

## VII - Ajax


Comme je l'ai déjà mentionné le framework Rialto met à notre disposition la technique AJAX. Elle est soit intégrée dans un widget pour son fonctionnement ou bien nous pouvons l'utiliser directement. C'est cette approche que nous allons aborder dans ce chapitre. Gardant la même approche, Rialto nous simplifie la mise en oeuvre à travers une classe `rialto.io.AjaxRequest`. Le code Javascript ci-dessous montre un exemple d'utilisation de cette classe :

```
1 var remote=new rialto.io.AjaxRequest(  
  {  
2  url:"infoesp.php",  
3  method: 'get',  
4  withWaitWindow:false,  
5  callBackObjectOnSuccess:this,  
6  onSuccess : myMethod  
  }) ;  
7 remote.load( 'nom='+name+'&pays=France' );
```

Voici quelques explications concernant ce code :

- 1 : instanciation de la classe `rialto.io.AjaxRequest`,
- 2 : url à appeler. Ici, c'est un script Php mais cela peut être une servlet, un fichier XML, etc.,
- 3 : méthode utilisée : `get`, `post`,
- 4 : si `withWaitWindow = true`, un Popup apparaît à l'écran affichant un message d'attente tant que la réponse du serveur n'est pas reçue. Un bouton d'annulation permet à l'utilisateur d'interrompre l'action,
- 5 : Objet ayant la méthode `onSuccess`,
- 6 : Méthode `onSuccess`. Cette méthode permet de récupérer les informations transmises par le serveur,
- 7 : Appel de la méthode `load` déclenchant la requête auprès du serveur. Il est possible de passer des paramètres. Le code ci-dessous montre un exemple de réception par la méthode `myMethod` des données qui sont ensuite introduites dans une liste.

```
function myMethod( request )  
{  
  var data = eval("(" + request.responseText + ")");  
  mygrid.fillGrid( data);  
}
```

 Cette présentation n'a pas l'objet d'une présentation d'AJAX. Je vous renvoie vers la partie du site [Developpez.com](http://ajax.developpez.com) consacrée à cette technique <http://ajax.developpez.com/> pour plus de précision.

## VIII - L'apparence

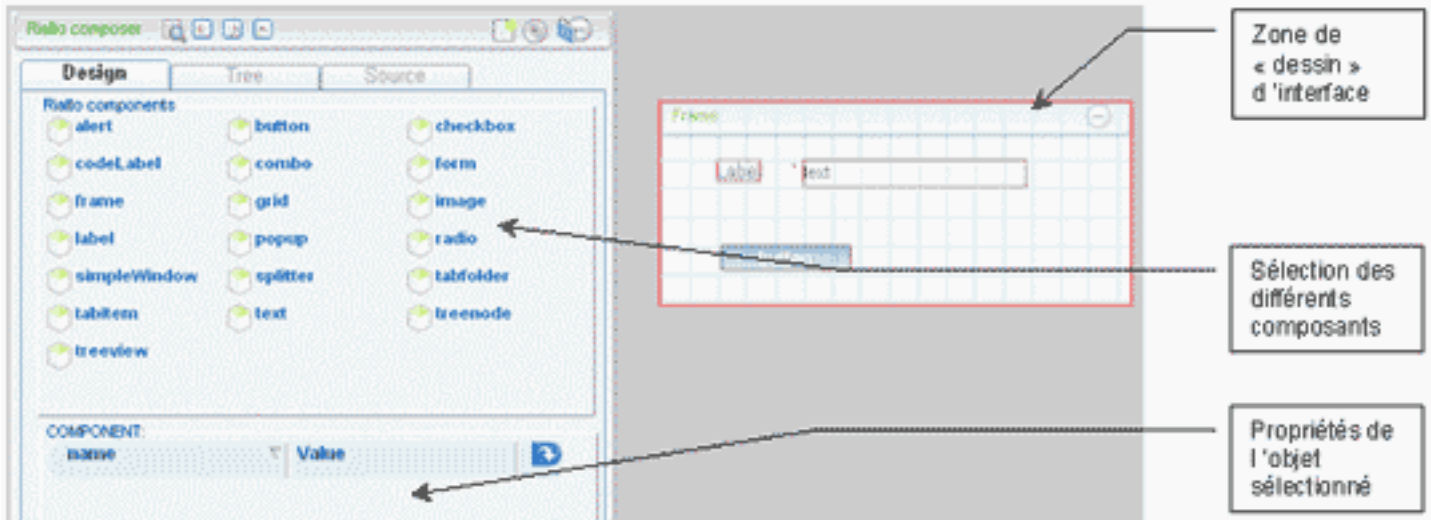
Rialto possède deux skins. Une est très élémentaire et l'autre est à dominante de gris et de bleu (voir les captures d'écran ci-dessus). L'ensemble des éléments intervenant dans l'apparence des composants est localisé dans des feuilles de styles. Elles sont bien commentées rendant la personnalisation plus aisée.

## IX - Gestion des langues

Rialto possède un module permettant de gérer les langues. Ainsi, vous pouvez facilement y ajouter vos propres textes dans les langues que vous voulez utiliser. Les textes utilisés en interne par le framework sont en trois langues : français, anglais et allemand.

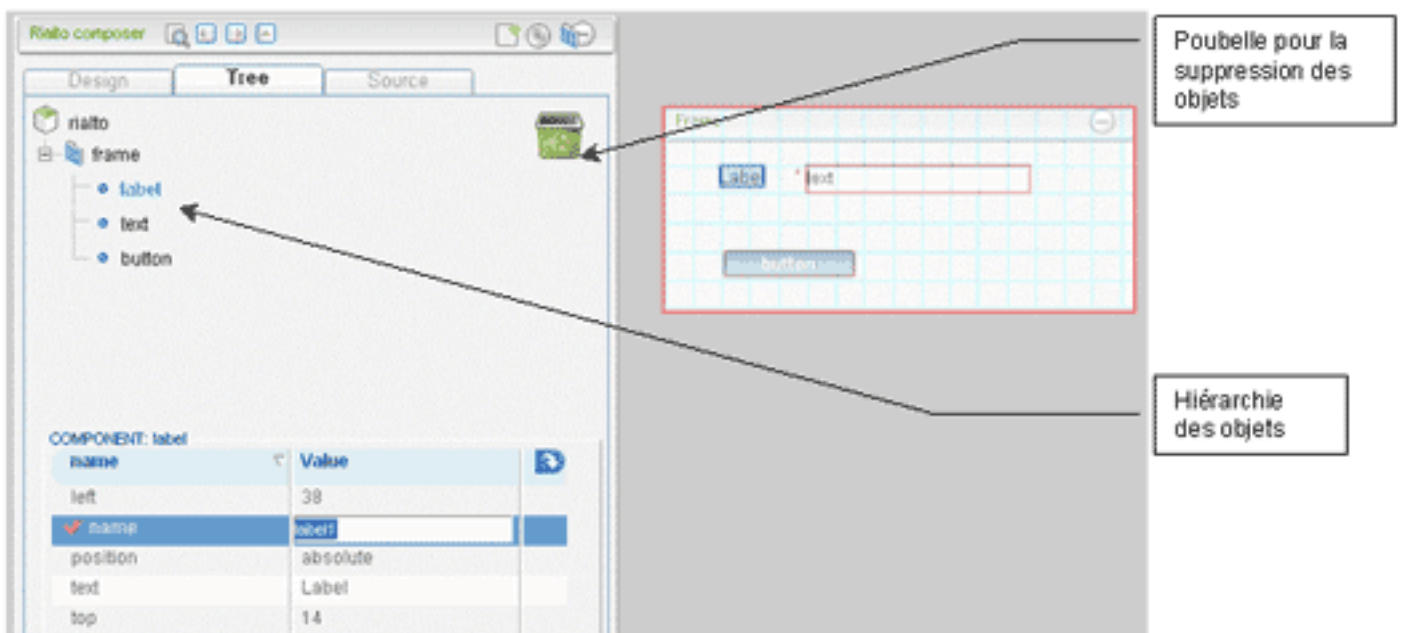
## X - Rialto studio

La mise en oeuvre de Rialto se fait naturellement écrivant des lignes en Javascript. Cependant, les créateurs de Rialto mettent à notre disposition un outil nous permettant de nous affranchir de cette étape souvent laborieuse. Cet outil, c'est un éditeur WYSIWYG appelé Rialto Studio réalisé lui même avec le framework Rialto.



Interface de Studio Rialto

Grâce une interface (cf ci-dessous) , il permet de « dessiner notre interface », de positionner les différents composants, de modifier leurs différents paramètres, etc.



Interface de Studio Rialto (paramétrage de composants)

Rialto Studio sauvegarde bien attendu en Javascript les écrans composés mais aussi sous format XML. A partir de ce format, il est possible d'obtenir l'interface défini d'autres formats qui vous sont propre PHP, JSF, JSP, etc. ou soutenus par le projet Rialto (RialtoTaglib, RialtoPHP, etc.).

## XI - Les sous projets Rialto

Le développement du framework Rialto en Javascript est l'élément central du projet. Cependant, le projet Rialto regroupe plusieurs autres sous projets. Leur but est de rendre accessible le framework Rialto sous différents langages et technologies. Il y a 6 sous projets :

Nom du sous projet	Description
RialtoGWT	Encapsulation du Framework Rialto afin d'être utilisé avec Google Web Toolkit (GWT)
RialtoTaglib	Utilisation des différents composants Rialto via des balises JSP (Java Server Page)
RialtoJSF	Utilisation des différents composants Rialto via JSF (Java Server Faces)
RialtoPHP	Encapsulation du Framework Rialto afin d'être utilisé avec PHP
RialtoPython	Encapsulation du Framework Rialto afin d'être utilisé avec Python
Rialto.Net	Encapsulation du Framework Rialto afin d'être utilisé avec .Net

Ainsi, il est possible à une personne experte dans une langage ou d'une technologie, ne connaissant pas le Javascript ou bien ne voulant pas utiliser ce langage dans son application, d'utiliser Rialto sans avoir trop à investir et d'obtenir un résultat satisfaisant rapidement.

## XII - Résumé

Nous venons de voir une présentation succincte du framework Rialto focalisée sur quelques composants et fonctionnalités me paraissant intéressants. Bien entendu, Rialto en possède nombreux autres. Tous les composants offrent de nombreuses options, sont ouverts aux évolutions et à la personnalisation les rendant très adaptables.

Pour résumer la présentation, je dirais que Rialto est un framework orienté pour les applications de gestion dont la principale caractéristique est **sa simplicité de mise en oeuvre et d'emploi**.

Rialto dispose d'un forum animé par les développeurs de Rialto où il est possible de trouver de l'aide et de faire partager son expérience.

Actuellement, Rialto est en version 0.9. La sortie de la version 1.0 est prévue durant le 1er trimestre 2008. Cette version sera une version plus optimisée et avec quelques nouveautés : fonctions autoresize behavior, un composant de formatage de données, etc.

Dans les versions suivantes v1.x, de nouveaux composants seront ajoutés comme menuBar.

## XIII - Références

Site de Rialto <http://rialto.application-servers.com/>  
**Javascript pour le Web 2.0** parus aux éditions Eyrolles