

Tutoriel pour développeur WebObjects: Composants classiques

Vous avez appris dans ce billet à manipuler les composants de WebObjects. Vous devriez avoir compris comment les pages fonctionnent, mais il vous manque encore de connaître les composants que l'on utilise au quotidien.

Voici une liste de ces composants avec leur syntaxe à l'ancienne, la nouvelle syntaxe et quelques exemples d'utilisation.

WOString

Rôle

C'est probablement le composant le plus utilisé du framework. Son but est simple, afficher une chaîne de caractère. Vous pouvez l'inclure un peu où vous voulez pour y insérer une chaîne de caractère.

Bindings importants

value: chaîne de caractère que vous souhaitez afficher, tout simplement.

numberFormatter: instance de NSNumberFormatter à utiliser s'il s'agit d'une valeur numérique qu'il faudra parser et/ou formatter.

dateFormatter: chaîne de caractère pour indiquer les informations l'affichage d'une valeur de type date.

Inline tags

```
<wo:str />
```

```
<wo:string>
```

WOConditional

Rôle

Permet d'afficher un bloc selon une condition, par exemple « Afficher ce bloc si l'utilisateur connecté est administrateur ».

Bindings importants

condition: condition à laquelle le code contenu sera affiché. Quelque soit le type de paramètre passé, un cast vers boolean est effectué. Comprenez que false, null, chaîne vide et 0 (zéro) valent FAUX. N'importe quoi d'autre est vrai.

negate: si ce binding est mis à vrai, c'est la négation de la condition qui déclanchera l'affichage du code. Par défaut ce binding est à faux.

Inline tags

```
<wo:if> ... </wo:if>
```

```
<wo:conditionnal> ... </wo:conditionnal>
```

```
<wo:condition> ... </wo:condition>
```

Notez qu'en utilisant ces tags inline vous disposez d'un petit raccourci avec un tag ayant pour sa part negate par défaut à vrai comme son nom l'indique.

```
<wo:not> ... </wo:not>
```

WORepetition

Rôle

En complément des trois éléments précédents, vous avez avec celui-ci de quoi faire les trois quarts de vos applications. L'élément WORepetition permet de parcourir une liste ou de faire un certain nombre d'itérations. C'est le composant dont le fonctionnement demande le plus d'explication sur son fonctionnement.

Admettons que vous utilisiez le `WORepetition` avec une liste d'éléments de type `User`. Vous allez alors mettre en place un binding sur item de type `User` appelé « `currentUser` », accessible en lecture et en écriture. À chaque tour de boucle, le setter de ce binding sera appelé afin d'y placer la valeur courante. Ainsi lorsque vous allez vouloir afficher par exemple le nom de votre utilisateur courant, vous utiliserez simplement un binding `currentUser.name`, donc inutile de vous prendre la tête avec des indices de liste quand vous codez le contenu du composant.

Bindings importants

list: si vous souhaitez itérer sur les composants d'une liste, c'est ici que vous devrez fournir la liste.

item: élément courant pendant l'exécution de la boucle.

count: il s'agit d'une alternative à `list`, permettant d'itérer un nombre de fois précis sans se référer au parcours d'une liste.

index: binding lecture/écriture qui contiendra la valeur `i` pour la `i`ème itération.

Inline tags

```
<wo:foreach> ... </wo:foreach>  
  
<wo:repeat> ... </wo:repeat>  
  
<wo:repetition> ... </wo:repetition>  
  
<wo:loop> ... </wo:loop>
```

AjaxUpdateContainer

Rôle

Par la suite, je vais parler de composants Ajax, il m'est donc obligatoire de parler de l'`AjaxUpdateContainer`. Son principe est très simple, le composant sera par la suite transformé en une zone `<div>` munie d'un id que vous lui aurez fourni. Vous pouvez alors ajouter à votre page des composants Ajax qui vont déclencher une action sur le serveur,

et indiquer l'id de votre AjaxUpdateContainer de façon à ce que le résultat de votre action vienne remplacer son contenu actuel.

Si vous aviez besoin de mettre à jour plusieurs panneaux de votre écran sur une seule action du serveur, regardez AjaxUpdateTrigger

Bindings importants

id: chaîne de caractère qui permet d'identifier votre élément de manière unique dans une page.

Inline tags

```
<wo:AjaxUpdateContainer> ... </wo:AjaxUpdateContainer>
```

WOHyperlink / AjaxHyperlink

Rôle

Inutile de vous faire un dessin. Il s'agit de liens classiques et ajax.

Bindings importants

action: il est vivement recommandé d'utiliser ce bindings plutôt que href ou pageName car il est plus facile de détecter une erreur dans un fichier Java que dans un fichier HTML.

directActionName: pour appeler une directAction plutôt qu'une action du composant sous-jacent.

string: chaîne de caractère à afficher pour représenter le lien (vous pouvez aussi l'insérer dans les balises)

updateContainerID: pour AjaxHyperlink, partie de la page à rafraichir avec le composant renvoyé par l'action appelée.

Inline tags

Pour le WOHyperlink

```
<wo:link> ... </wo:link>  
<wo:hyperlink> ... </wo:hyperlink>
```

Le AjaxHyperlink n'a par défaut pas de raccourci.

```
<wo:AjaxHyperlink> ... </AjaxHyperlink>
```

WOSubmitButton / AjaxSubmitButton

Rôle

Vous aurez compris que vous avez affaire aux éléments qui vont générer un input de type submit. Ces deux composants doivent se trouver dans un formulaire (créé de préférence à l'aide d'un WOForm, aucun binding important).

Bindings importants

action: le fonctionnement est le même que pour le WOHyperlink à ceci près que les données contenues dans le formulaire contenant le bouton vont être envoyées au serveur, ce qui n'est pas le cas avec le WOHyperlink (ni avec le AjaxUpdateLink).

value: chaîne de caractère à afficher sur le bouton.

updateContainerID: pour AjaxSubmitButton, partie de la page à rafraichir avec le composant renvoyé par l'action appelée.

Inline tags

Pour le WOSubmitButton

```
<wo:submit> ... </wo:submit>  
<wo:submitButton> ... </wo:submitButton>
```

Pour le AjaxSubmitButton

```
<wo:AjaxSubmitButton> ... </wo:AjaxSubmitButton>
```

ERXElse

Rôle

Pas un essentiel du tout mais un apport particulièrement appréciable de Wonder. Souvent lors de l'usage d'un WOConditionnal, on a également un comportement à définir pour le cas où la condition n'est pas satisfaite. C'est une partie de l'origine du binding negate, puisqu'au lieu d'écrire deux conditions, il suffisait d'utiliser deux fois la même et d'utiliser ce binding. Wonder offre une possibilité encore plus simple, le else ! Situé après un WOConditionnal, son contenu ne sera affiché qu'à condition que celle du WOConditionnal ne le soit pas. Ce composant n'a aucun binding.

Inline tags

```
<wo:else> ... </wo:else>
```

Aller plus loin

Ce tutoriel s'arrête ici. Le but n'est pas de faire toute la chevauchée avec vous mais simplement de vous mettre le pied à l'étrier. J'espère que vous suivez ces tutoriaux en faisant vos propres tentatives à côté et non pas simplement en lisant ces pages l'une après l'autre. La mise en application est très importante pour vérifier que vous avez mémorisé ce qui a été abordé et compris les principes.

Pour aller plus loin dans la création de votre propre application, voici une liste de noms de composants qui vous seront utiles. Vous avez un nom, cherchez simplement comment l'utiliser sur internet et dans le pire des cas, servez vous de la fonction d'auto-complétion d'eclipse pour comprendre le fonctionnement du composant. eclipse est un excellent outil pour apprendre notamment les bindings obligatoires.

Faites attention aux bindings GET et aux bindings GET/SET. Attention également aux chaînes de caractères, qui sont à mettre des fois sur le binding string et des fois sur value. Vous aurez remarqué d'ailleurs qu'il ne faut pas utiliser le même pour le WOHyperlink et le WOSubmitButton. Il n'y a pas vraiment de règle pour savoir lequel des deux il faudra utiliser, la seule chose certaine est que si le binding string existe, c'est celui à utiliser.

WOBody, WOForm, WORadioButton, WORadioButtonList, WOCheckBox, WOCheckBoxList, WOTextField, WOPasswordField, WOText, WOPopUpButton, WOResetButton, WOImage, AjaxModalDialog, ...