1

Web dynamique avec Seaside

Main Author(s): N. Bouraqadi, Université Libre de Bruxelles, bouraqadi@ensm-douai.fr

Currently developed on: Squeak, Can be used for lectures using ?Squeak/VisualWorks?, latest version ?1.2? available on ?www.squeaksource.com/RegConf?, contact person: ??

1.1 Compléments sur Seaside

Quelques messages pour générer du html. Le destinataire de ces messages est l'objet passé en paramètre de la méthode renderOn: (instance de WAHtmlRender).

- text: 'chaine de caracteres' affiche simplement la chane de caractères.
- heading: 'texte du titre' level: niveau affiche un titre. Le deuxième paramètre est un entier qui correspond au niveau hiérarchique du titre (1 correspond au le plus grand)
- break introduit un retour à la ligne
- horizontalRule introduit une ligne horizontale
- form: ["definition de boutons, zones de saisies, "] définit un formulaire au sens Html. Nécessaire pour avoir des boutons et autres zones de saisies dans une page Html. Reoit en paramètre un bloc qui contient les messages de création des boutons, zones de saisie,
- textInputWithValue: valeurInitiale callback: [:valeur |"traitements"] crée une zone de saisie simple (sans barre de défilement). La valeur initiale est celle qui est affichée au démarrage (nil pour ne rien afficher). Le dernier argument est un bloc qui reçoit comme paramètre la valeur saisie (valeur) dans le champ. Cette valeur peut être utilisée dans le traitement défini par le bloc. Ce bloc est exécuté quand la touche "Entrée" est pressée ou quand on clic sur un bouton du formulaire dans lequel se trouve la zone de saisie.
- submitButtonWitAction: ["traitements"] text: 'titre du bouton' ajoute un bouton qui a pour titre la chane de caractères passée comme deuxième argument. Un clic sur le bouton provoque l'exécution des traitements définis dans le bloc passé comme premier paramètre.

1.2 Encore des compteurs !

Il s'agit de réaliser encore un compteur, mais cette fois, il devra être accessible via le web (utilisation de Seaside). De plus, il devra être personnalisable dans la mesure où l'utilisateur doit pouvoir modifier directement la valeur du compteur et modifier l'incrément. Concrètement, vous devez définir une classe CompteurPersonnalise sous-classe de WAComponent qui représente une application Seaside. CompteurPersonnalise sera munie de :

- deux champs (value et increment),
- une méthode d'initialisation (initialize),

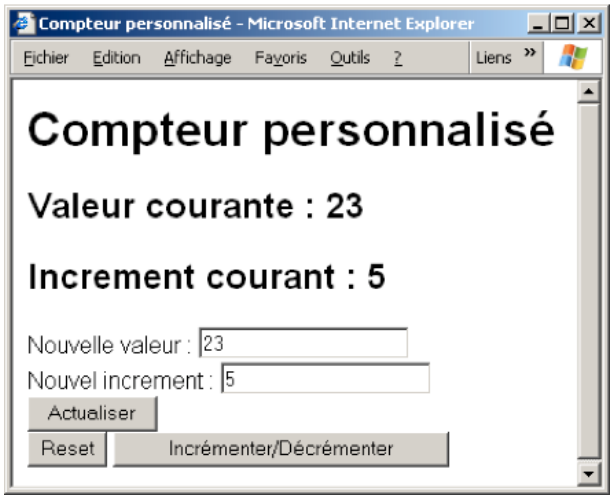

Figure 1.1: L'interface du compteur personnalisé

- ainsi que la méthode de génération du code html (renderOn:).

L'interface utilisateur doit être analogue à celle de la figure 1.1. Deux champs de saisie permettent de modifier la valeur du compteur et son incrément après clic sur le bouton "Actualiser". Le bouton "Reset" réinitialise le compteur (value mise à 0 et increment mis à 1). Enfin, le bouton "Incrémenter/Décrémenter" permet d'ajouter l'incrément au compteur et donc de l'incrémenter si l'incrément est positif ou de le décrémenter dans le cas contraire.

1.3 Séparer l'interface du code métier

La structure suggérée pour l'exercice précédent n'est pas très propre. En effet, un même objet prend en charge à la fois le traitement (code métier : incrémenter/décrémenter, modification de l'incrément,) et l'interface utilisateur. Ce choix de conception rend difficile les éventuelles évolutions ou réutilisation. En particulier, si l'on souhaite changer d'interface utilisateur, voire de modèle de communication distante.

Dans cet exercice, on se propose de faire la séparation entre code métier et code d'interface et en illustrer l'utilité à l'aide d'un exemple simple. Cet exemple tourne autour d'une calculatrice arithmétique. Vous définirez tout d'abord la classe Calculatrice qui dispose de deux champs qui représentent respectivement l'opérande gauche et l'opérande droite. Munissez la classe d'accesseurs en lecture écriture à ces deux champs, ainsi que de 4 méthodes pour réaliser les 4 opérations arithmétiques. Bien entendu, ces quatre méthodes :

- ne prennent pas de paramètres,
- effectuent le calcul en utilisant les champs représentant les deux opérandes,
- et retournent le résultat du calcul

Définissez ensuite la classe CalculatriceWeb sous-classe de WAComponent qui représente une application Seaside. CalculatriceWeb permet l'utilisation à travers le web des opérations fournies par Calculatrice. Son interface s'apparente à celle donnée par la figure 1.2.

Vous allez maintenant exploiter la séparation entre code métier et code d'interface utilisateur. En effet, vous allez réutiliser la classe Calculatrice pour faire un nouveau compteur accessible via le web. L'interface devra être identique à celle du compteur de l'exercice précédent.

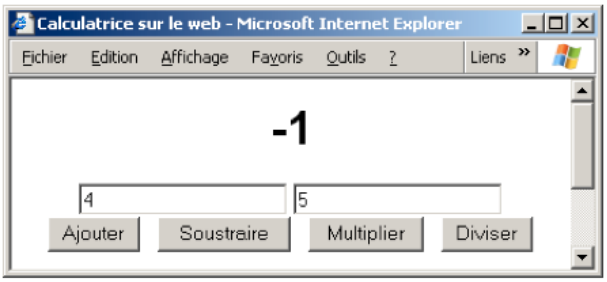

Figure 1.2: L'interface de la calculatrice.

1.4 Une application un peu plus sophistiquée

Il s'agit ici de définir un outil qui permet de gérer des tableaux blancs partagés via le web. Un tableau blanc est une zone de texte que plusieurs utilisateurs peuvent modifier. Chaque tableau est caractérisé par un nom et dispose d'une liste identifiants les utilisateurs qui ont le droit d'y accéder.

Chaque utilisateur dispose d'un identifiant et d'un mot de passe qu'il fournit pour se connecter. Une fois connecté il a le choix entre créer un nouveau tableau ou modifier tableau existant. Les utilisateurs qui ont accès à un tableau peuvent en modifier le contenu ainsi que la liste des utilisateurs qui ont accès au tableau.