Pagination de données en Flex

Romain Vimont

Logica

mercredi 29 avril 2009

Plan

Objectifs

Données locales

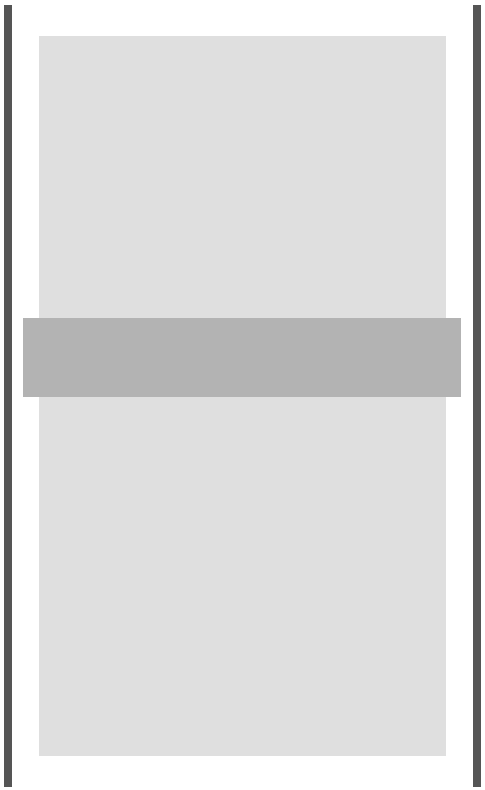
Un `DataGird` possède un `dataProvider`, contenant généralement des données disponibles localement :

- ☺ facile et rapide si peu de données ;
- ☹ problèmes de performances et de mémoire si beaucoup de données ;
- ☹ inenvisageable si énorme volumétrie.

Données distantes

- présence d'une seule "page" de données localement ;
- chargement dynamique de nouvelles données si nécessaire ;
- utilisation quasiment transparente pour le développeur.

Principe



LazyDataProvider

- proxy d'accès aux données ;
- se comporte comme un `ArrayCollection` ;
- charge dynamiquement de manière asynchrone les données nécessaires non présentes.

Chargement côté serveur

- communication par `BlazeDS`[1] ;
- `RemoteObject` pointant vers un objet Java.

Interface à implémenter

- `D[] load(int startIndex, int endIndex, ...);`
- `int getSize(...);`

[1] Autres implémentations possibles...

Prise en compte du filtrage et du tri

Comportement par défaut

Clic sur en-tête de colonne de `DataGrid` :

$\Longrightarrow$ tri local

$\Longrightarrow$ rapatriement de toutes les données localement pour trier

Comportement souhaité

1. envoi des informations de tri au serveur ;
2. tri des données par le serveur ;
3. retour de la page de données demandée.

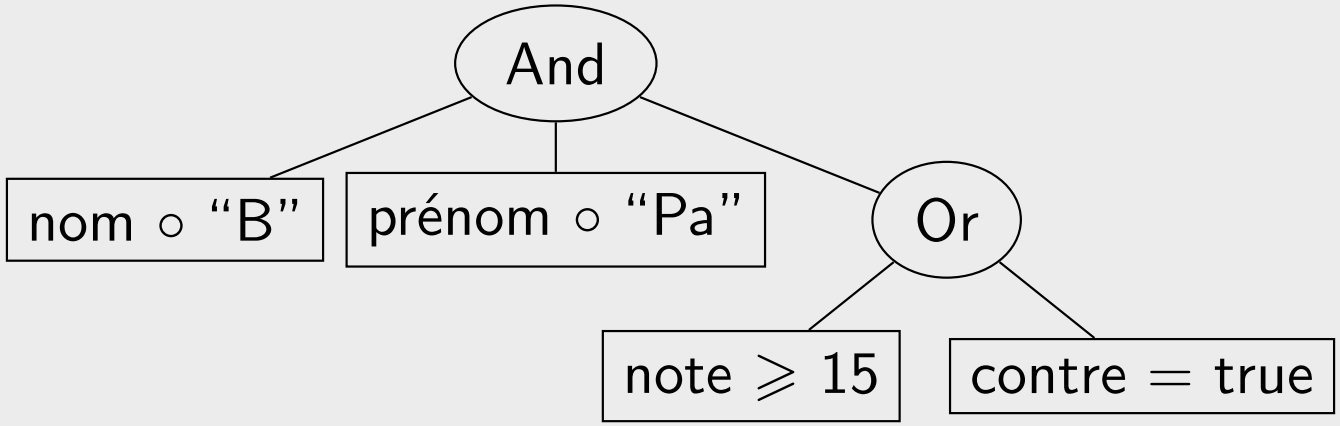
de même pour le filtrage. . .

Informations de filtrage et de tri

Informations de tri

(note, DESC)	(nom, ASC)	(prénom, ASC)

Informations de filtrage



$$nom \circ \text{“}B\text{”} \wedge prénom \circ \text{“}Pa\text{”} \wedge (note \geqslant 15 \vee contre = true)$$

Mode local

Lorsque peu d'élements sont présents (seuil), basculer en local.

Avantages

- ☺ meilleures performances ;
- ☺ évite de charger inutilement le serveur.

Côté client

Interface à implémenter

- `function localLoad(sortingInfos, filteringInfos):Array;`
- `function localGetSize(filteringInfos):uint;`
- `function filterCanBeAppliedLocally(filteringInfos):Boolean;`

Automatisation du filtrage et du tri

Données	Filtre	Tri	Résultat
nom | note Ève | 16 Alice | 11 Bob | 14	`note > 13`	`[(nom, ASC)]`	nom | note Bob | 14 Ève | 16

Filtrage et tri

- gestion du tri facilement automatisable
- gestion du filtrage automatisable

$\Longrightarrow$ `localLoad` et `localGetSize` automatisables.

Automatisation de la décision local/remote

- Ancien filtre : `note` $>$ `15` ;
- Données résultant du filtre présentes localement.

Nouveau filtre	Commentaire
`note` $>$ `17`	Filtrage local possible : il suffit de supprimer ceux qui ont entre 15 et 17.
`note` $>$ `12`	Filtrage local impossible : il manque potentiellement des données.

Décision de la possibilité du filtrage local automatisable ?

Automatisation de la décision local/remote (suite)

Nouveau filtre (F2) applicable localement à partir du résultat de l'ancien filtre (F1)

$\Longrightarrow$ résultat de F2 inclus dans celui de F1

$\Longrightarrow$ F2 "plus restrictif" F1

$\Longrightarrow$ $F2 \implies F1$

$\Longrightarrow$ $F2 \wedge \neg F1 \implies \bot$

Résolvable

Évaluer si $F2 \wedge \neg F1$ est insatisfiable (problème *UNSAT*) : résolvable par des algorithmes de *calcul propositionnel*.

Automatisation : conclusion

Création d'un `AutomaticLocalDataLoader` qui implémente toutes les méthodes de l'interface.

Résultat

À partir d'une pagination qui a toujours lieu côté serveur, ajouter la fonctionnalité "locale" lorsque peu de données sont présentes a un coût quasi-nul pour le développeur !

Questions

?