Relazione SecureDataContainer

Giovanni Solimeno

20 novembre 2018

1 Scelte progettuali

In generale, si è deciso di creare una classe contenitrice `Element<E>` per incapsulare il tipo generico, in modo da rendere più facili le operazioni sui permessi, e una classe `User`, rappresentante il singolo utente.

1.1 `Element<E>`

La classe `Element<E>` contiene tre campi privati, con relativi metodi setter/getter:

- Il campo `owner` è una stringa contenente il proprietario del dato. Non è accessibile direttamente, ma è possibile controllare se un particolare utente è prioprietario del dato tramite il metodo `ownedBy(who)`.
- Il campo `allowed` è una lista di stringhe, contenente gli utenti autorizzati ad accedere al dato (escluso il proprietario). Non è possibile accedere direttamente alla lista, ma è possibile indicare che un utente deve essere autorizzato/disautorizzato tramite i rispettivi metodi `allowUser(other)` (che lancia `UserAlreadyAllowedException` se l'utente è già autorizzato) e `denyUser(other)` (che lancia `UserNotAllowedException` in caso l'utente non sia presente tra gli utenti autorizzati).
- Il campo `el` è una riferimento ad una istanza del tipo generico E. È possibile accedervi tramite il metodo `getEl()`, che restituisce un riferimento `el`, mentre non è possibile cambiarne il valore.

Inoltre, è possibile controllare se un utente può accedere a un dato tramite il metodo `canBeAccessedBy(other)`, che restituisce true se e solo se other è il proprietario oppure è presente nella lista degli utenti autorizzati.
La classe sovrascrive il metodo `Object.equals(other)`, in modo da ritornare true se e solo se `other.owner.equals(this.owner)` è true e `other.getEl().equals(this.getEl())` restituisce true.
Viene generata l'eccezione unchecked `NullPointerException` se `other` è nullo (la scelta di chiamare `equals` su `other.owner` e su `other.getEl()` è stata fatta in modo da lanciare in automatico l'eccezione se `other` è `null`).

1.2 `User`

La classe `User` contiene due campi privati:

- Il campo `userName` contiene il nome utente dell'utente, ed è possibile accedervi tramite il metodo `getUserName()`. Non è possibile in alcun modo modificarne il valore.
- Il campo `userPass` contiene la password dell'utente, ed è possibile soltanto modificarla, tramite il metodo `setUserPass( newPass )`, mentre non è possibile accedervi in alcun modo.

Inoltre, la classe implementa come meccanismo di login la sovrascrittura del metodo `Object.equals(other)`, che restituisce true se e solo se `other.getUserName().equals( this.getUserName )` e `other.userPass.equals (this.userPass)` (si è deciso di effettuare il confronto tramite i metodi/campi di other per lo stesso motivo di `Element<E>.equals()`), e implementa l'interfaccia `Comparable<T>`, in modo da ordinare gli utenti in base al nome (proprietà che viene usata nella classe `TreeMapSecureDataContainer`).

2 Scelte specifiche