JavaBeans

Programmazione in rete e laboratorio
2001/02

I **JavaBeans** sono **componenti** software riutilizzabili, che possono essere usati nella *programmazione visuale* per realizzare applicazioni grafiche.

Gli ambienti di programmazione (costruzione) che supportano i bean permettono di realizzare rapidamente delle applicazioni assemblando componenti software predefiniti (i bean).

Il programmatore diventa un *component assembler*, che utilizza componenti già definiti per creare nuove funzionalità. Il programmatore deve solo conoscere i servizi forniti dai componenti, ma non i dettagli dell'implementazione.

Per usare i beans occorre un *builder* "Java-enabled".

I bean vengono mantenuti in un *toolbox*, da cui possono essere presi per costruire applicazioni.Esistono numerosi siti in cui ci si possono procurare beans.

Vantaggio rispetto ad altri tool di programmazione visuale: indipendenza dalla piattaforma.

La Sun fornisce **BeanBox**, un semplice *application builder* scaricabile liberamente.(Non è più mantenuto)

I bean sono degli oggetti Java.

Principali caratteristiche:

Proprietà: caratterizzano l'aspetto ed il comportamento. In generale sono modificabili.

Eventi: i bean comunicano fra loro con il meccanismo degli eventi di Java.

Metodi: tutti i metodi pubblici sono accessibili.

Persistenza: è possibile salvare lo stato di un bean con il meccanismo della serializzazione.

Un bean è una normale classe Java che segue alcune convenzioni di nomi.

- Per ogni proprietà **xxx** ci sono due metodi:

Tipo getXxx() e **void setXxx(Tipo p)** ,

dove **Tipo** è il tipo della proprietà

- Se la proprietà ha tipo **boolean**, invece di **getXxx** si può usare **isXxx**.

Se un bean può essere la sorgente di un evento, deve implementare i due metodi:

public void addXxx(Xxx l)

public void removeXxx(Xxx l)

dove Xxx è il tipo dell'event listener.

In questo modo il bean sa a chi inviare gli eventi che genera.

Notare che i componenti **swing** rispettano le convenzioni dei bean.

Quando in un tool per costruire applicazioni si collegano fra loro due bean, il tool genera automaticamente il codice Java necessario per gestire le interazioni fra i due.

Ad esempio, se in BeanBox si collega l'evento "button push" del bottone con il metodo "startJuggling" del juggler, viene automaticamente creato un ActionListener il cui metodo "actionPerformed' chiama "startJuggling".

Questo listener verrà poi registrato presso il bottone con la addActionListener del bottone.

Per poter essere aggiunto al toolbox di un builder, un bean deve essere *impacchettato* in un file **JAR** (Java Archive).

Supponiamo che il bean si chiami **SimpleBean**. I passi da eseguire sono:

* Compilare **SimpleBean**.

* Creare un file, chiamato ad esempio **manifest.tmp**, contenente il testo:

```
Name: SimpleBean.class
Java-Bean: True
```

* Creare il file Jar con il comando:

jar cfm SimpleBean.jar manifest.tmp SimpleBean.class

Nel comando jar, invece del parametro SimpleBean.class si possono indicare più classi: ad esempio *.class, se SimpleBean usa delle classi ausiliarie.

Il file manifesto specifica quali, fra le classi impacchettate, sono dei bean.

Una volta che il file jar è stato generato, lo si prende e si inserisce nel *toolbox*. Ad esempio nel caso del **BeanBox**, dovrà essere inserito nella directory **jars**.

A questo punto il bean è disponibile per essere usato.

Tipi di proprietà dei bean

Proprietà semplici. Hanno un solo valore, ad esempio una stringa o un numero.

Proprietà indicizzate. Array di dati.

Proprietà collegate. Generano un evento ogni volta che cambia il loro valore.L'evento è di tipo **PropertyChange** e viene inviato a tutti i *listener* registrati.

Esiste una classe di servizio **PropertyChangeSupport** che si occupa della gestione dei *listener*.

Come fa un tool di programmazione visuale a sapere quali sono le caratteristiche di un nuovo bean che viene aggiunto al suo toolbox?

Si può usare il meccanismo della **reflection**, che consente di scoprire a runtime metodi e campi di una classe.

In particolare il package **java.beans** fornisce una classe **Introspector** con un metodo statico **getBeanInfo(...)**.

Passando un parametro di tipo **Class** a questo metodo, si ottiene un oggetto di tipo **BeanInfo** che fornisce tutte le proprietà, metodi ed eventi della classe passata come parametro.

L'interfaccia BeanInfo

Un *builder tool* estrae tutte le *features* di un bean (proprietà, eventi e metodi) usando la classe **Introspector**.

Alternativamente è possibile elencare esplicitamente tutte le *features* di un bean che si vogliono rendere accessibili, definendo una classe che implementa l'interfaccia **BeanInfo**.

Se il bean si chiama ad esempio **Bottone**, la classe si chiamerà **BottoneBeanInfo**

Conviene definire questa classe come estensione della classe **SimpleBeanInfo**, ridefinendone i metodi, come ad esempio **getPropertyDescriptors()**.

La classe **BottoneBeanInfo** deve essere messa nello stesso file JAR in cui si trova il bean **Bottone**.

Quando il bean viene caricato, il *builder tool* determina se il file JAR contiene una classe **BeanInfo** per il bean. Se questa esiste, la si usa per determinare le caratteristiche del bean. Altrimenti si usa l'introspezione standard.