Tutorial SWING-AWT: Il Layout Manager GridBagLayout

Di Michele de Nittis

Generalità

I *Gestori di Layout* sono particolari oggetti che aiutano a controllare la disposizione di componenti, quali gli elementi grafici o i controlli di una finestra, all'interno di un oggetto contenitore (*Container*)

Il più flessibile di questi gestori è l'oggetto di classe GridBagLayout, il cui impiego, più complesso ed articolato degli altri, viene di seguito illustrato.

Il principio di funzionamento

Il principio di funzionamento del gestore è semplice: i vari componenti vengono disposti su una *griglia immaginaria* di dimensioni non predeterminate. Le maglie, o celle, di questa griglia sono estese almeno quanto basta per rappresentare graficamente ogni componente aggiunto. In presenza di componenti di dimensioni grafiche diverse, quindi, le maglie della griglia sono di dimensioni diverse. Tutte le maglie di una stessa riga della griglia hanno altezza uguale a quella della maglia più alta, mentre tutte le maglie di una stessa colonna hanno larghezza uguale a quella della maglia più larga.

In generale, quindi, i componenti si trovano inseriti all'interno di maglie di dimensioni maggiori.

Per controllare il posizionamento di un componente all'interno della griglia, nonché il suo disegno all'interno di una maglia, si deve impiegare un oggetto supplementare di classe *GridBagConstraints* che, come suggerisce il nome, ha la funzione di raccogliere i vincoli di rappresentazione grafica di un componente.

I vincoli che si possono impostare mediante l'oggetto di classe GridBagConstraints riguardano:

- Il posizionamento del componente all'interno della griglia (assoluto, relativo);
- Il posizionamento del componente all'interno di una maglia;
- Il dimensionamento grafico del componente all'interno di una maglia;
- La disposizione di un componente su più maglie.

Impiego degli oggetti di classe GridBagLayout e GridBagConstraints

Un gestore di layout di classe GridBagLayout si istanzia, come tutti gli oggetti in Java, con l'operatore *new*:

```
GridBagLayout GBL = new GridBagLayout();
```

Per associare un gestore di layout ad un oggetto di classe *Container* si usa il metodo *setLayout()*:

```
CR.setLayout(GBL);
```

dove CR è una variabile che fa riferimento ad un oggetto di classe *Container*.

Una volta aggiunto un componente al contenitore mediante il metodo *add()* (nell'esempio aggiungiamo l'oggetto di classe JButton a cui fa riferimento la variabile R_Immetti)

```
CR.add(R_Immetti);
```

è possibile controllarne il layout (cioè il posizionamento ed il dimensionamento) impostando le opportune proprietà di un oggetto di classe GridBagConstraint, appositamente creato.

```
GridBagConstraints GBC = new GridBagConstraints();
… // Impostazione proprietà di GBC
```

Per associare l'oggetto GBC al componente ed al *layout manger* si può impiegare il metodo *setConstraints()* dell'oggetto di classe *GridBagLayout*.

```
GBL.setConstraints(R_Immetti, GBC);
```

Vediamo, quindi, come impostare opportunamente le proprietà dell'oggetto di classe GridBagConstraint per dimensionare e disporre un componente in un contenitore.

Posizionamento di un Componente

Per comprendere meglio il tema di questo paragrafo faremo riferimento ad un esempio concreto, definendo il metodo Show() di una data classe il cui compito è quello di disegnare graficamente una finestra.

Posizionamento Assoluto

Il posizionamento assoluto di un componente nella predetta griglia immaginaria si ottiene specificando le coordinate della maglia di destinazione mediante e proprietà *gridx* e *gridy* dell'oggetto di classe *GridBagConstraints*

Posizionamento Relativo

Una volta inserito un componente, è possibile inserire il successivo specificandone la posizione relativamente all'ultima maglia occupata.

Se impostiamo, infatti, la proprietà *gridx* dell'oggetto di classe *GridBagConstraints* al valore costante *RELATIVE* otteniamo un posizionamento del nuovo componente sulla stessa 'riga' del precedente, mentre se impostiamo la proprietà *gridy* al valore costante *RELATIVE,* posizioniamo il nuovo componente sulla stessa colonna del precedente.

Il posizionamento relativo di un nuovo componente rispetto a quello precedente consente di inserire i vari oggetti di un contenitore in modo sequenziale per riga, da sinistra a destra, o per colonna, dall'alto in basso. E' possibile terminare una riga o una colonna e passare alla riga o alla colonna successiva della griglia impostando le proprietà *gridWidth* o *gridHeight* dell'oggetto GrdiBagConstraint al valore *REMAINDER* prima di inserire il componente successivo.

Il seguente è un esempio di posizionamento assoluto di sei componenti (due etichette, una casella di testo, un'area di testo e due bottoni) in un contenitore. Si omettono le definizioni non necessarie.

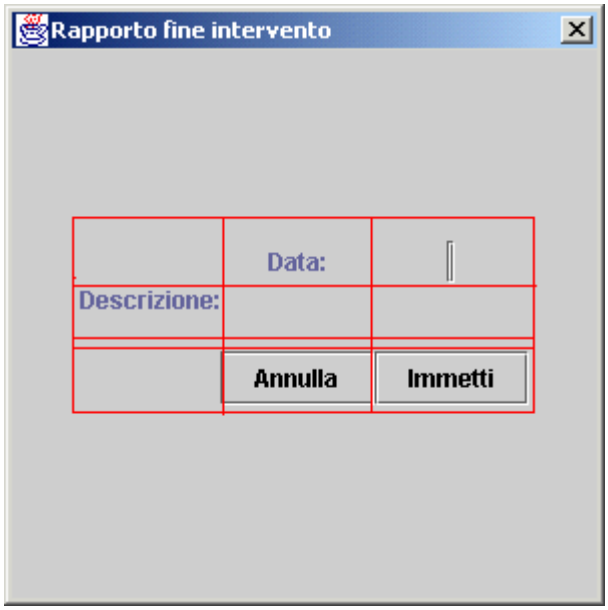
```
private JFrame MyFrame;
private JDialog MyDialog;
private Container CR;
private GridBagLayout GBL=null;
private GridBagConstraints GBC=null;
…….
public void Show(JFrame JF){
// Salvataggio dell'informazione sul frame della finestra chiamante
                MyFrame = JF;
// Istanziamento di un oggetto di classe JDialog (modale)
                MyDialog = new JDialog(MyFrame, "Rapporto
fine intervento", true);

// Prelievo del contenitore
                CR = MyDialog.getContentPane();
// Layout Manager
                GBL = new GridBagLayout();
                CR.setLayout(GBL);
// Creazione dei controlli
                if(R_data==null){
                        R_data = new JTextField();
                }
                if(L_data==null){
                        L_data = new JLabel("Data:");
                }
                if(R_descrizione==null){
                        R_descrizione = new JTextArea();
                }
                if(L_descrizione==null){
                        L_descrizione = new
JLabel("Descrizione:");
                }
                if(R_Annulla==null){
                        R_Annulla=new JButton("Annulla");
                }
                if(R_Immetti==null){
                        R_Immetti=new JButton("Immetti");
                }
// Posizionamento dei controlli
// L_data
                GBC = new GridBagConstraints();
                CR.add(L_data);
                GBC.gridx=1;
                GBC.gridy=0;
                GBL.setConstraints(L_data,GBC);
// R_data
                GBC = new GridBagConstraints();
                CR.add(R_data);
                GBC.gridx=2;
                GBC.gridy=0;
                GBL.setConstraints(R_data,GBC);
// L_descrizione
                GBC = new GridBagConstraints();
                CR.add(L_descrizione);
                GBC.gridx=0;
                GBC.gridy=1;
                GBL.setConstraints(L_descrizione,GBC);
// R_descrizione
                GBC = new GridBagConstraints();
                CR.add(R_descrizione);
                GBC.gridx=0;
                GBC.gridy=2;
                GBL.setConstraints(R_descrizione,GBC);
// R_Annulla
                GBC = new GridBagConstraints();
                CR.add(R_Annulla);
                GBC.gridx=1;
                GBC.gridy=4;
                GBL.setConstraints(R_Annulla,GBC);
// R_Immetti
                GBC = new GridBagConstraints();
                CR.add(R_Immetti);
                GBC.gridx=2;
                GBC.gridy=4;
                GBL.setConstraints(R_Immetti,GBC);

// mostra
                MyDialog.pack();
        MyDialog.setSize(300, 300);
                MyDialog.setVisible(true);

        }// fine Show()
```

Il risultato del precedente codice è mostrato nella seguente figura:

Il risultato finale non è proprio quello che ci si aspettava. In effetti, i componenti sono stati solo posizionati all'interno della griglia ed il *Layout Manager* ha utilizzato le poche informazioni disponibili per cercare di dimensionarli al meglio.

Dimensionamento di un Componente

Per migliorare la qualità di rappresentazione grafica dei componenti all'interno di un contenitore si devono impartire, sempre tramite le proprietà dell'oggetto di classe *GridBagConstraints*, le specifiche di dimensionamento.



Osserviamo, innanzitutto, che il blocco dei componenti viene inserito al centro del contenitore e che la riga gridy=3 non ha spessore poiché non vi è alcun componente.

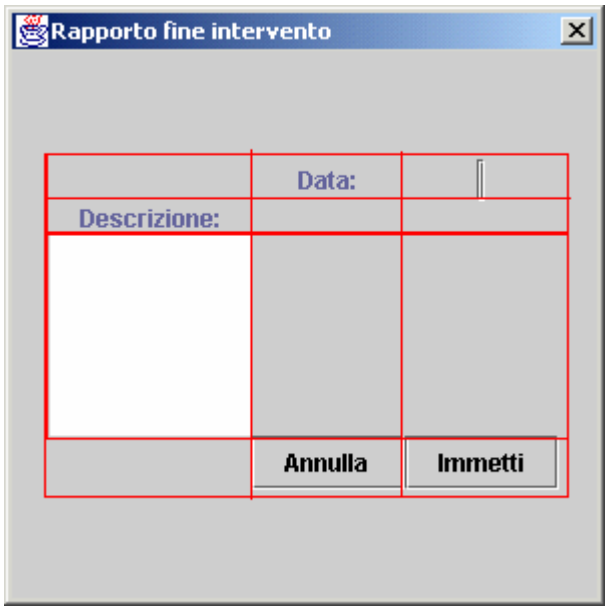
L'area della maglia gridx=0, gridy=2 è troppo piccola per poter visualizzare il componente di classe *JTextArea*, mentre l'area della maglia gridy=0, gridx=2 costringe il Layout Manager ad una rappresentazione estremamente ridotta del componente di classe *JTextField*.

Dimensionamento delle Maglie

E' possibile 'suggerire' al Layout Manager un dimensionamento diverso di una maglia impostando le dimensioni preferire del componente inserito mediante il metodo *setPreferredSize()*.

Applicando il suddetto dimensionamento all'esempio in questione, otteniamo il seguente risultato:

```
// R_descrizione
        GBC = new GridBagConstraints();
        Dimension D = new Dimension(100,100);
        R_descrizione.setPreferredSize(D);
        CR.add(R_descrizione);
        GBC.gridx=0;
        GBC.gridy=2;
        GBL.setConstraints(R_descrizione,GBC);
```



Osserviamo che è comparso il controllo di classe *JTextArea* e che l'intero blocco, grazie al metodo pack() dell'oggetto di classe JDialog, si è espanso sia orizzontalmente sia verticalmente per creare lo spazio sufficiente alle nuove dimensioni del componente, senza sprechi. La riga gridy=3 non è ancora presente.

Riempimento

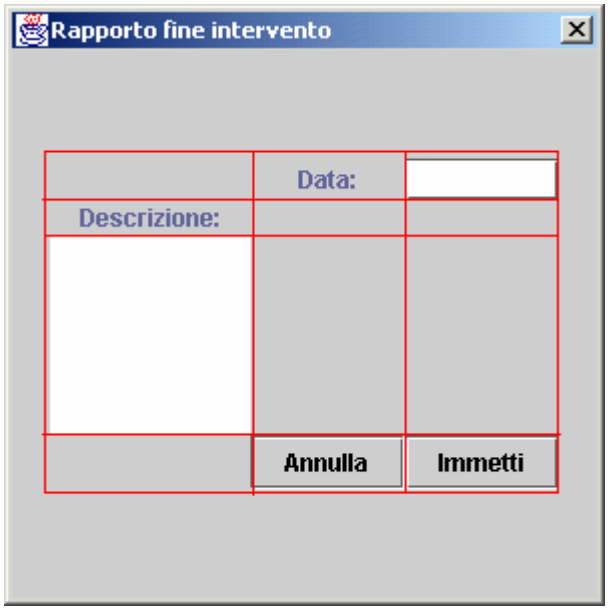
In generale un componente è disposto all'interno di una maglia senza occupare tutto lo spazio a disposizione.

E' possibile far sì che il controllo venga disegnato espandendosi su tutta l'area della maglia, ovvero solo in direzione verticale o orizzontale.

Per specificare il tipo di riempimento, è sufficiente impostare la proprietà *fill* dell'oggetto di classe *GridBagConstraint* alle costanti *BOTH*, *VERTICAL*, *HORIZONTAL*.

Nel nostro esempio si può fare riferimento al controllo di classe *JTextField*, a destra dell'etichetta 'Data:', il cui riempimento è stato limitato alla direzione orizzontale.

```
// R_data
        GBC = new GridBagConstraints();
        CR.add(R_data);
        GBC.gridx=2;
        GBC.gridy=0;
        GBC.fill=GridBagConstraints.HORIZONTAL;
        GBL.setConstraints(R_data,GBC);
```



Ancoraggio

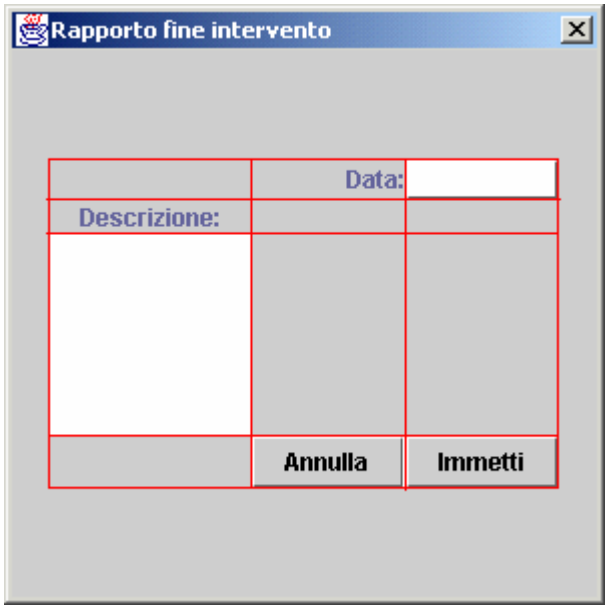
Un componente di dimensioni minori di quelle della maglia in cui si trova, quale l'etichetta (classe JLabel) 'Data:' dell'esempio in questione, viene disegnato in posizione centrale. E' possibile modificare l'ubicazione del componente nella maglia, attraverso la proprietà *anchor* dell'oggetto di classe *GridBagConstraints*, in modo che esso aderisca ad uno dei quattro lati o ad uno dei quattro angoli. La specifica di posizionamento viene fornita mediante nove costanti denominate come i punti cardinali:

- GridBagConstraints.NORTH, per il lato superiore;
- GridBagConstraints.SOUTH, per il lato inferiore;

- GridBagConstraints.EAST, per il lato destro;
- GridBagConstraints.WEST, per il lato sinistro;
- GridBagConstraints.NORTHEAST per l'angolo superiore destro;
- GridBagConstraints.NORTHWEST per l'angolo superiore sinistro;
- GridBagConstraints.SOUTHEAST per l'angolo inferiore destro;
- GridBagConstraints.NORTHWEST per l'angolo Inferiore sinistro;
- GridBagConstraints.CENTER, per il centro (default);

Impostando, quindi, l'ancoraggio dell'etichetta 'Data:' su EAST, si ottiene il seguente risultato:

```
GBC.anchor= GridBagConstraints.EAST;
```



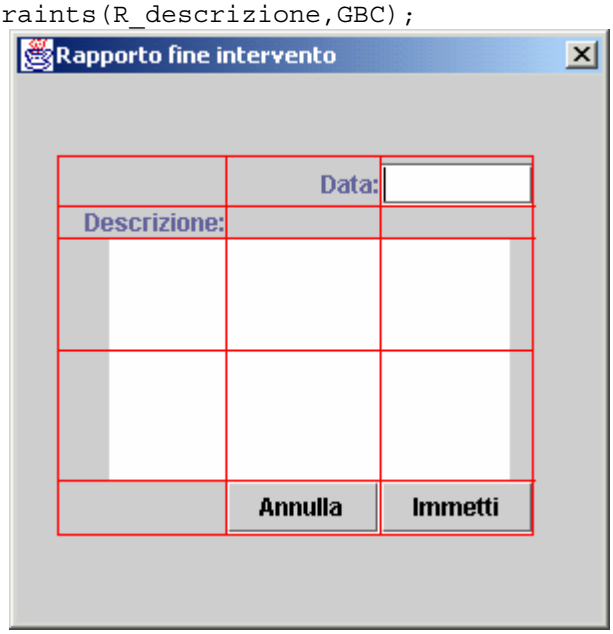
Espansione di un componente su più maglie

Per conseguire una distribuzione migliore, più regolare e gradevole dei componenti, è possibile far si che essi occupino più di una maglia. Ciò si può ottenete impostando le proprietà *gridwidth* e *gridheight* dell'oggetto d classe *GridBagConstraints*.

Nell'esempio in questione, ridimensioniamo il controllo di classe *JTextArea* e facciamo in modo che occupi le colonne gridx=1 e gridx=2 e le righe gridy=2, gridy=3.

```
// R_descrizione
      GBC = new GridBagConstraints();
      Dimension D = new Dimension(120,200);
      R_descrizione.setPreferredSize(D);
      CR.add(R_descrizione);
      GBC.gridx=0;
      GBC.gridy=2;
      GBC.gridwidth=3;
      GBC.gridheight=2;
```

```
GBL.setConstraints(R_descrizione,GBC);
```

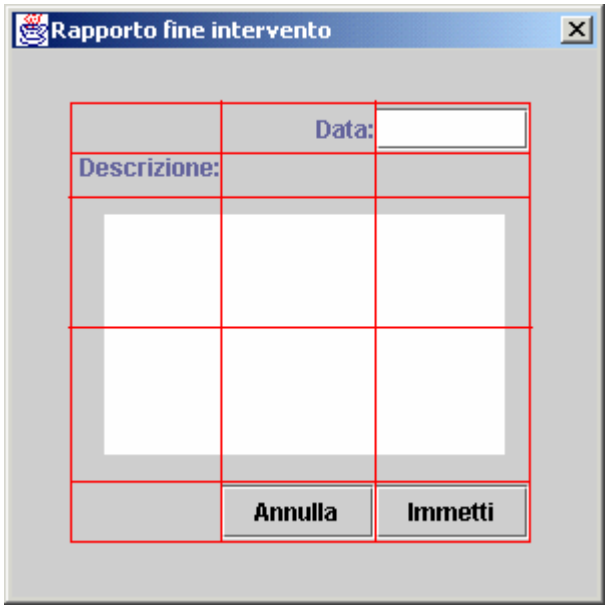


Margini

E' possibile, attraverso la proprietà *insets* dell'oggetto di classe *GridBagConstraints*, impostare la distanza di un componente dai margini della maglia in cui è contenuto. La proprietà *insets*, che è un riferimento ad un oggetto di classe *Insets*, viene impiegato dal Layout Manager per impostare la distanza in pixel del componente dai margini della maglia in cui è inserito.

La classe *Insets* è definita con sole quattro proprietà: *top, bottom, left e right* che rappresentano le distanze dai margini della maglia.

Impostiamo, dunque, l'area di testo (JTextArea) della finestra del nostro esempio in modo che abbia una distanza di 15 pixel dai margini superiore ed inferiore delle tre maglie occupate.



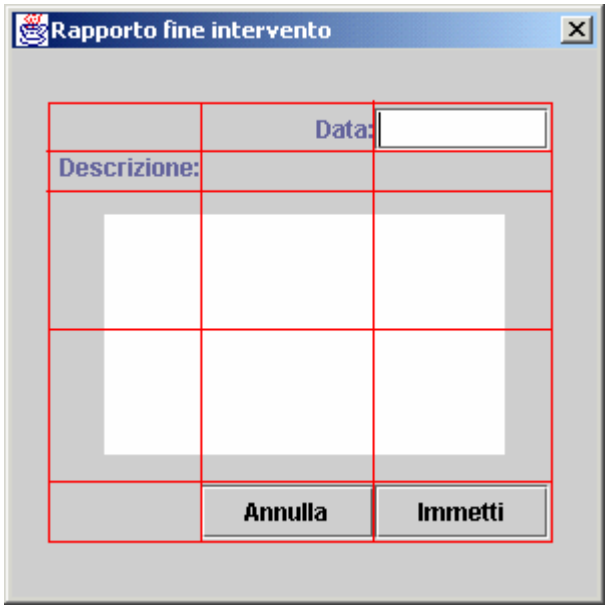
```
// R_descrizione
                    GBC = new GridBagConstraints();
                    Dimension D = new Dimension(200,120);
                    R_descrizione.setPreferredSize(D);
                    CR.add(R_descrizione);
                    GBC.gridx=0;
                    GBC.gridy=2;
                    GBC.gridwidth=3;
                    GBC.gridheight=2;
                    GBC.insets.top=15;
                    GBC.insets.bottom=15;
                    GBL.setConstraints(R_descrizione,GBC);
```

Aggiustare le dimensioni (padding)dei componenti

Impostando le proprietà *ipadx* e *ipady* dell'oggetto di classe *GridBagConstraints* si ottiene un effetto simile a quello del riempimento (vedi pagina 7) perché si fa in modo di aumentare le dimensioni reali del componente lungo le direzioni X ed Y di un certo numero di pixel. Dunque, ad esempio, impostando un valore *ipadx*=40 si specifica al Layout Manager di considerare il componente più largo di 40 pixel (20 per lato).

Tornando al nostro esempio, volendo ingrandire i due pulsanti di 10 pixel lungo l'asse X, è possibile impostare la proprietà *ipadx* dell'oggetto di classe *GridBagConstraints* al valore 10.

```
// R_Annulla
      GBC = new GridBagConstraints();
      CR.add(R_Annulla);
      GBC.gridx=1;
      GBC.gridy=4;
      GBC.ipadx=10;
      GBL.setConstraints(R_Annulla,GBC);
// R_Immetti
      GBC = new GridBagConstraints();
      CR.add(R_Immetti);
      GBC.gridx=2;
      GBC.gridy=4;
      GBC.ipadx=10;
      GBL.setConstraints(R_Immetti,GBC);
```



Distribuzione pesata dello spazio (weightx, weighty)

Un'altra caratteristica del layout manager *GridBagLayout* è quella di poter gestire la distribuzione dello spazio a disposizione tra le righe e le colonne della maglia.

Si osservi, dunque, la figura relativa all'ultima modifica apportata alla maschera di esempio. Come si vede il layout manager ha assegnato ai due bottoni 'Annulla' ed 'Immetti' due maglie di uguali dimensioni, avendo queste ultime ugual peso per impostazione di default.

Impostando le proprietà *weightx* e *weighty* del controllo *GridBagConstraints* si possono modificare i pesi con cui il layout manager distribuisce lo spazio disponibile orizzontalmente e verticalmente tra le maglie dei due componenti.

Modifichiamo, ad esempio, i pesi delle due maglie lungo la direzione orizzontale in modo che quella relativa al bottone 'Immetti' riceva più spazio di quella relativa al bottone 'Annulla'.

Se per ipotesi lo spazio disponibile fosse di P pixel ed il peso della maglia del tasto 'Immetti' fosse weightx=5 e quello della maglia del tasto 'Annulla' fosse weightx=2, allora il peso totale della riga sarebbe 7, le dimensioni della maglia del bottone 'Immetti' sarebbero X*5/7 pixel e quelle della maglia del bottone 'Annulla' sarebbero X*2/7.

Attenzione: Non è detto che distribuendo diversamente lo spazio tra le due maglie con pesi diversi, anche i relativi bottoni vengano disegnati con dimensioni diverse. Ricordiamo, infatti, che un componente non necessariamente viene disegnato impiegando tutto lo spazio a disposizione.

Tornando all'esempio in argomento, per mostrare le conseguenze dell'impostazione dei pesi, facciamo in modo che i due bottoni vengano disegnati su tutto lo spazio a disposizione (proprietà *fill*).

```
// R_Annulla
      GBC = new GridBagConstraints();
      CR.add(R_Annulla);
      GBC.gridx=1;
      GBC.gridy=4;
      GBC.ipadx=10;
      GBC.weightx=2; // Test
      GBC.fill=GridBagConstraints.HORIZONTAL; //test
      GBL.setConstraints(R_Annulla,GBC);
// R_Immetti
      GBC = new GridBagConstraints();
      CR.add(R_Immetti);
      GBC.gridx=2;
      GBC.gridy=4;
      GBC.ipadx=10;
      GBC.fill=GridBagConstraints.HORIZONTAL; //test
      GBC.weightx=5; // Test
      GBL.setConstraints(R_Immetti,GBC);
```

Conclusione

Abbiamo visto nei precedenti paragrafi come sia possibile impostare il Layout di un insieme di componenti all'interno di un contenitore (*Container*) impiegando il gestore di layout GridBagLayout.

Di seguito viene riportato un frammento di codice che propone un layout elaborato dei componenti visti nell'esempio precedente.

```
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;

public class XXXX {
```

```
….
private JFrame MyFrame;
private JDialog MyDialog;
private Container CR;

// controlli
private JTextField R_data=null;
private JTextArea R_descrizione=null;
private JButton R_Immetti=null;
private JButton R_Annulla=null;
private JLabel L_data=null;
private JLabel L_descrizione=null;

private GridBagLayout GBL=null;
private GridBagConstraints GBC=null;

….
public void Show(JFrame JF){

// Salvataggio dell'informazione sul frame della finestra chiamante
      MyFrame = JF;

// Istanziamento di un oggetto di classe JDialog (modale)
      MyDialog = new JDialog(MyFrame, "Rapporto fine intervento", true);

// Prelievo del contenitore
      CR = MyDialog.getContentPane();

// Layout Manager
      GBL = new GridBagLayout();
      CR.setLayout(GBL);

// Creazione dei controlli
      if(R_data==null){
            R_data = new JTextField();
      }
      if(L_data==null){
            L_data = new JLabel("Data:");
      }
      if(R_descrizione==null){
            R_descrizione = new JTextArea();
      }
      if(L_descrizione==null){
            L_descrizione = new JLabel("Descrizione:");
      }
      if(R_Annulla==null){
            R_Annulla=new JButton("Annulla");
      }
      if(R_Immetti==null){
            R_Immetti=new JButton("Immetti");
      }

      // Posizionamento dei controlli
      // L_data
      GBC = new GridBagConstraints();
      CR.add(L_data);
      GBC.gridx=1;
      GBC.gridy=0;
      GBC.anchor= GridBagConstraints.EAST;
      GBL.setConstraints(L_data,GBC);
      // R_data
      GBC = new GridBagConstraints();
      CR.add(R_data);
```

```
        GBC.gridx=2;
        GBC.gridy=0;
        GBC.fill=GridBagConstraints.HORIZONTAL;
        GBL.setConstraints(R_data,GBC);
        // L_descrizione
        GBC = new GridBagConstraints();
        CR.add(L_descrizione);
        GBC.gridx=0;
        GBC.gridy=1;
        GBL.setConstraints(L_descrizione,GBC);
        // R_descrizione
        GBC = new GridBagConstraints();
        Dimension D = new Dimension(200,120);
        R_descrizione.setPreferredSize(D);
        CR.add(R_descrizione);
        GBC.gridx=0;
        GBC.gridy=2;
        GBC.gridwidth=3;
        GBC.gridheight=2;
        GBC.insets.top=2;
        GBC.insets.bottom=15;
        GBC.insets.left=2;
        GBC.insets.right=2;
        GBC.fill=GridBagConstraints.HORIZONTAL;
        GBL.setConstraints(R_descrizione,GBC);
        // R_Annulla
        GBC = new GridBagConstraints();
        CR.add(R_Annulla);
        GBC.gridx=1;
        GBC.gridy=4;
        GBC.ipadx=10;
        GBL.setConstraints(R_Annulla,GBC);
        // R_Immetti
        GBC = new GridBagConstraints();
        CR.add(R_Immetti);
        GBC.gridx=2;
        GBC.gridy=4;
        GBC.ipadx=10;
        GBL.setConstraints(R_Immetti,GBC);

        // mostra
        MyDialog.pack();
          MyDialog.setSize(300, 300);
        MyDialog.setVisible(true);

}// fine Show()

…
} // fine classe XXXX
```

Il risultato del predetto frammento di codice è il seguente: