

# SWT Text 欄位

在接下來的幾章中，我們會經由檢視過在 GUI 中經常會用到的四個 widget 來擴展你的 SWT 知識—text 欄位、button、list、與 combo。我們會先從檢視 SWT 的 Text 這個 class 開始，它是位在 org.eclipse.swt.widgets 這個 package 中。

text 欄位是任何 GUI 的標準元件之一—用來展示資訊給使用者，或讓使用者能夠編輯或輸入資訊。text 欄位可以當作單行的編輯欄位，這讓使用者可以輸入或觀察一組資訊（由欄位的字元數所指定），或多行的編輯欄位，讓使用者可以輸入或檢視可折行的開放文字。text 欄位也可以有多種外框，讓你可以開發出想要在使用者介面中展示的外觀與操作感受（例如 3D 或平面的方框）。

所有運用 text 欄位所需的 method 都被包裝在單一的 SWT class 中—Text class。這是與像是 AWT 等對 text 欄位有不同 class 來應付不同樣式的圖形化工具組有所不同的地方。

## 將 Text 欄位加入 Shell

如同所有的 SWT widget，text 欄位必須加到容器物件中—Shell、Composite、或 Group 的 instance。

## 我該怎麼做？

建構 text 欄位並加入到 window 中需要兩個程式步驟。首先要建構出 Text 這個 class 的 instance、將 Shell 的參考與所需的樣式屬性傳進 Text 的 constructor。

### 在這一章中：

- 將 Text 欄位加入 Shell
- 對 Text 欄位加邊框
- Text 欄位的定位
- 設定 Tab 的順序
- 設定 Text 大小的上限
- 與 Text 欄位的文字互動
- 從資料庫讀給 Text 欄位
- 插入文字到 Text 欄位
- 建構多行的 Text 欄位
- 折行的文字
- 加上 Scrollbar
- 對 Text 事件反應
- 執行欄位驗證

再來是指定 text 欄位於容器中的位置並設定它的寬與高。範例 5-1 展示出如何將 text 欄加入到 window 中。

#### 範例 5-1. 加入 text 欄位到 window 中

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.*;

public class TextFieldExample {

    Display d;
    Shell s;
    TextFieldExample( ) {
        d = new Display( );
        s = new Shell(d);
        s.setSize(250,250);
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));
        final Text text1 = new Text(s, SWT.SINGLE);
        text1.setBounds(10,10,100,20);
        s.open( );
        while(!s.isDisposed( )){
            if(!d.readAndDispatch( ))
                d.sleep( );
        }
        d.dispose( );
    }
}
```

建構出 TextFieldExample 的結果展示在圖 5-1。



圖 5-1. 執行 TextFieldExample 的結果

## 發生了什麼事？

將 text 欄位加入 shell 的程式碼是相當簡單的，僅由兩行程式組成：

```
Text text1 = new Text(s, SWT.SINGLE);
text1.setBounds(10,10,100,20);
```

在上面的程式碼中，s 代表要加入 text 欄位的 Shell。傳給 Text 的 constructor 的樣式為 SWT.SINGLE，表示要建構單行編輯樣式的 text 欄位。

雖然 text 欄位只需用第一行程式來建構並加入到 shell 中，但沒有第二行程式的話，它是不會顯示的。需要設定 text 欄位的邊界，也就是 x 與 y 座標，以及寬與高，widget 才會顯示出來。

## 對 Text 欄位加邊框

如你在圖 5-1 所看到的，沒有邊框的 text 欄位是不太明顯的。然而，SWT 讓 text 欄位可以很容易的經由使用額外的樣式屬性來改變外觀。對大部分的應用程式來說，你應該會藉由加入外框來加強 text 欄位的外觀。

## 我該怎麼做？

現在，你應該已經很熟悉如何指定 widget 的樣式—讓 shell、menu、與 toolbar 使用樣式。建構出有外框包圍的 text 欄位的樣式是 SWT.BORDER，它是以一般的方法傳遞給 Text 的 constructor：

```
nal Text text1 = new Text(s, SWT.SINGLE | SWT.BORDER);
```

如果你照著這樣對 TextFieldExample 作改變，這個 text 欄位會如同圖 5-2 所展示的一樣，絕對會比圖 5-2 沒有外框的 text 欄位好多了。

## Text 欄位的定位 (positioning)

在建構 GUI 的時候，你會需要精準的控制每個組成介面的 widget 的位置。在 SWT 中，有兩種方式可指定容器中 widget 的大小與位置。第一種方法是使用 setBounds() 或 setLocation() 這兩個 method，它們是所有 widget 的 superclass 的一部份。

顯示在 shell 上的 toolbar 會佔用交互區域左上角開始的空間，所以在計算 widget 位置時要把 toolbar 的高度也算進去。

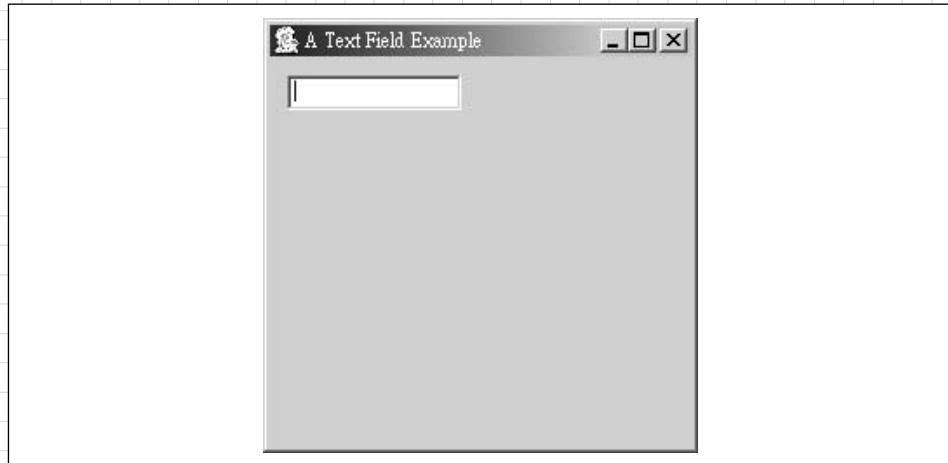


圖 5-2. 指定 SWT.BORDER 的 text 欄位

## 我該怎麼做？

在 `TextFieldExample` 中，`Text` 這個 widget 是以下列的方式來定位的：

```
text1.setBounds(10,10,100,20);
```

在 `window` 上特定的位置加進 `text` 欄位只需要調整前兩個參數，它們代表 `text` 欄位左上角相對於 `shell` 工作區域（在 `title bar` 與 `menu` 下方的區域）左上角的 `x` 與 `y` 座標。第一個 `setBounds()` 參數會因其值的大小決定 `text` 欄位左右的位置，而第二個參數值大小決定 `text` 欄位上緣離 `window` 底部的遠近。要將一群 widget 向左邊對齊只需要在傳給 `setBounds()` 參數時使用相同的第一個參數值就可以。範例 5-2 展示出如何將兩個 `Text` 的 widget 向左靠齊，結果顯示在圖 5-3。

### 範例 5-2. 將 `text` 欄位對齊

```
public class TextFieldExample {  
  
    Display d;  
    Shell s;  
    TextFieldExample( ) {  
        d = new Display( );  
        s = new Shell(d);  
        s.setSize(250,250);  
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));  
        s.setText("A Text Field Example");  
        nal Text text1 = new Text(s, SWT.SINGLE | SWT.BORDER);  
        text1.setBounds(100,50,100,20);  
        nal Text text2 = new Text(s, SWT.SINGLE | SWT.BORDER);
```

```

        text2.setBounds(100,75,100,20);
        s.open();
        while(!s.isDisposed()){
            if(!d.readAndDispatch()){
                d.sleep();
            }
        }
        d.dispose();
    }
}

```

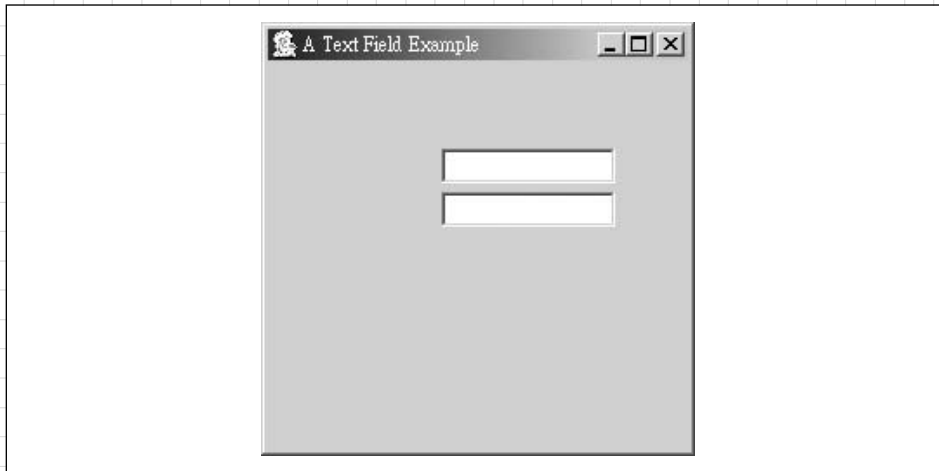


圖 5-3. 兩個 text 欄位的定位

你也可以使用 `setLocation()` 來指定此 widget 的 x 與 y 座標而無須指定大小。下面的程式碼會將範例 5-2 中的 `text1` 物件位置向下移動五個像素：

```
text1.setLocation(100, 55);
```

精確的對齊 widget 在設置使用者介面時是很重要的。

我會在第九章展示如何使用 `layout` 來作精確定位的另外一種技巧。

## 設定 Tab 的順序

建構讓使用者能夠接受的使用者介面關鍵因素之一是要讓 widget 的「tab 順序」正確。tab 順序是使用者在按下鍵盤上的 Tab 鍵時每個 widget 取得輸入焦點的順序。使用者會期待 tab 順序是合理的—如同它們與 window 互動時的順序。舉例來說，假設有一個 window 是要輸入姓名、地址、城市，省，與郵遞區號資訊。在這樣的介面中，使用者預期會依序輸入姓名、地址等。如果 window 開啟時焦點是在地址欄位上，然後在使用者按下 Tab 按鍵時會移動到姓名欄位，這樣的介面並不符合使用者所會預期輸入資訊的方式。

在沒有特定順序的情形下，以前面所討論的姓名與地址等為例，一般的做法是讓輸入焦點的移動是從上到下，從左到右。

## 我該怎麼做？

tab 順序是根據建構 widget 的順序來設定的：

```
nal Text text1 = new Text(s, SWT.SINGLE | SWT.BORDER);
text1.setBounds(100, 50, 100, 20);
nal Text text2 = new Text(s, SWT.SINGLE | SWT.BORDER);
text2.setBounds(100, 75, 100, 20);
```

在這個例子中，text1 是在 text2 之前建立的，所以它會是 tab 順序中的第一個 widget。

設定 tab 順序的另外一種方法是呼叫 Composite 這個 class 的 setTabList()。Composite 是 Shell 的 super class，對前面直接將 Text 的 widget 置入 Shell 的程式碼來說，加入下面的程式會讓 text2 成為 tab 順序中的第一個 widget：

```
Control[] c = {text2, text1};
s.setTabList(c);
```

## 設定 Text 大小的上限

有一個 text 欄位的大小屬性是必須提及的—text 欄位所能容納的字元數目。這個大小有兩種觀點。第一個是與有多少字元能夠在 text 方塊的範圍中顯示有關。使用者通常很不喜歡無法顯示所有字元的單行 text 方塊。雖然可見的字元數目與所選擇的字型大小有關（預設上是由作業系統來控制），你還是應該要嚐試限制合理預期下能夠放進 text 欄位的字元值數目。

其次，在程式設計的觀點上，限制 text 欄位的字元數目是為了要對資料作驗證。假設你設計個 GUI 介面讓使用者能夠輸入資料以儲存於資料庫中。如果 text 欄位要接收儲存的在某資料庫欄位的值，你能夠藉由限制可接受的數量來降低資料錯誤的可能性。

## 我該怎麼做？

下面的程式可以設定 Text 的字元數上限：

```
text1.setTextLimit(10);
```

如果你將此行加入 `TextFieldExample` 中，你會在嚐試輸入超過 10 個字元時看到它的效果：辦不到。

正確判別特定欄位所能允許的字元數目之後使用 `setBounds()` 來設定欄位中可見的字元數量，以及使用 `setTextLimit()` 來限制可輸入的字元輸入，兩者的設計應該要一致。這在可梁到不同的螢幕解析度與不同平台的字型大小時是個很複雜的程序。但這是邁向專業品質使用者介面時所必須採取的步驟。

## 與 Text 欄位的文字互動

如果你要建構出顯示從其他來源產生的文字給使用者觀察的介面－換句話說，你使用 `text` 欄位來在 `window` 開啟或讀出資料時顯示文字值－就必須要直接與 `text` 欄位的內容互動。

### 我該怎麼做？

與欄位中的文字有兩種可能的互動－讀取輸入的值或是將文字寫到欄位中。SWT 的 widget 都是使用與 `JavaBeans` 規格類似的 `getXXX()` 與 `setXXX()` 這些 `method`。

對 `Text` 這個 `class` 的 `setText()` 呼叫會讓文字值顯示出來：

```
text1.setText("Some text goes here");
```

---

#### Warning

你確實可以寫入超過由 `setTextLimit()` 所設定字元上限的字數。這個動作不會有什麼問題，但若你稍後嚐試取出此 `text` 欄位的值並將它塞到僅能容納該上限的資料庫欄位時，那就會有錯誤出現。在以程式來產生資料給 `text` 欄位時必須要很小心的確保不會超過字元上限。

---

反向的操作是從 `text` 欄位中讀取出文字並將它儲存在變數中以供程式運用。對這種工作而言，要使用的是 `getText()` 這個 `method`：

```
String contents = text1.getText();
```

## 從資料庫讀給 Text 欄位

如果你開發的是企業資訊應用程式，無可避免的會需要從資料庫查詢中讀出資料餵給一組 text 欄位。要這麼做的話，就必須了解如何運用 JDBC 的 ResultSet。

### 我該怎麼做？

完成此任務的程式碼是很直接了當的。下面的程式碼會使用 JDBC/ODBC Bridge driver 連接到 MS Access 的 Northwind 範例資料庫：

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection conn = DriverManager.getConnection("jdbc:odbc:NorthWind");
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("SELECT * FROM Employees");
if(rs.next()) {
    text1.setText(rs.getString("FirstName"));
    text2.setText(rs.getString("LastName"));
}
```

text1 與 text2 這兩個 text 欄位值是由此 result set 的第一個資料列中的 FirstName 與 LastName 兩個欄位的 String 值所產生的。

### 有關於...

ResultSet 中的資料庫欄位值不是 String 時要怎麼辦？對於這些情況，使用其他的 method，像是 getInt() 來取出資料並將其轉換成 String 值以傳入 setText()，如下列所示：

```
text2.setText(Integer.toString(rs.getInteger("EmployeeID")));
```

## 插入文字到 Text 欄位

有時候，你會需要插入文字到已經有資料的文字欄位後面，或插到 text 欄位資料的中間。

### 我該怎麼做？

要在 text 欄位後面加入文字，你可以使用 append() 這個 method：

```
text1.append(" This will be added to the end of what's already there");
```

也可以插入到目前內容文字的前面：

```
text1.insert("This goes before ");
```

## 有關於...

當你想要插入文字到現有文字的中間要怎麼做？`insert()` 也可以與 `selection` 類型的 `method`（稍後立即說明）併用來完成這個目標。你必須要在呼叫 `insert()` 之前選出想要插入位置的文字範圍。被插入的文字會替換掉所選擇的區域。如果你不想要取代任何的文字，只要讓選擇兩字之間一個空白位置區域就好。

## 建構多行的 Text 欄位

你的介面設計經常會需要可有讓使用者輸入更多資料的 `text` 欄位。對這種類型的應用程式，像是文字編輯器或者可允許輸入開放格式文字（例如說備註欄位）的資料庫應用程式而言，你必須要使用多行編輯樣式的 `text` 欄位。

## 我該怎麼做？

建構多行 `text` 欄位是很容易的—只要在 `Text` 的 `constructor` 中指定 `SWT.MULTI` 樣式就可以：

```
Text text1 = new Text(s, SWT.MULTI | SWT.BORDER);
```

範例 5-3 會建構出顯示在圖 5-4 的 `window`，你會發現有一個多行的 `text` 欄位填滿整個 `window` 的工作區域。

### 範例 5-3. 多行 `text` 欄位

```
public class TextFieldExample {  
  
    Display d;  
    Shell s;  
    TextFieldExample() {  
        d = new Display();  
        s = new Shell(d);  
        s.setSize(250,250);  
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));  
        s.setText("A Text Field Example");  
        Text text1 = new Text(s, SWT.MULTI | SWT.BORDER);  
        text1.setBounds(0,0,250,250);  
        s.open();  
        while(!s.isDisposed()){
```

### 範例 5-3. 多行 text 欄位 (續)

```
        if(!d.readAndDispatch())
            d.sleep();
    }
    d.dispose();
}
```

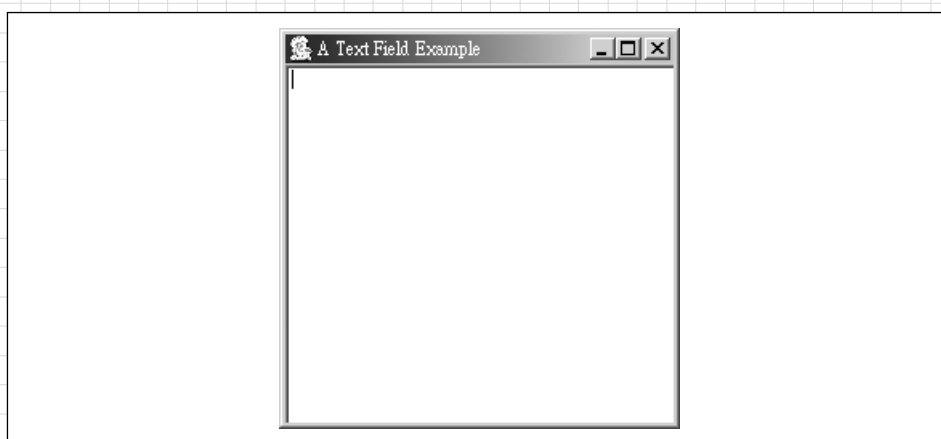


圖 5-4. 多行的範例

## 折行 (Wrapping) 文字

如果執行範例 5-3，你會發現在該欄位輸入文字並不會產出如預期般的結果。如果你輸入超過該欄所能顯示的文字，這些文字不會折到下一行；它會出現在右邊某些地方。這會一直往右下去直到你按下 **Enter** 才會換到下一行。

### 我該怎麼辦？

如你所預期的，建構會折行的欄位只需要指定另外一種樣式，`SWT.WRAP` 就好：

```
nal Text text1 = new Text(s, SWT.MULTI | SWT.WRAP | SWT.BORDER);
```

改變範例 5-3 會建構出如圖 5-5 所示會折行的 text 欄位。

使用者通常會預期  
滿一行的時候  
文字會折到下一  
行。



圖 5-5. 折行的文字

## 有關於...

需要將 text 欄位的樣式從折行改變成不折行，甚或將單行改變成多行時要怎麼辦？使用 SWT 是不可能的，因為這些樣式是在建構 widget 的時候指定的。解決辦法是要建構出兩個獨立的 widget，各有不同的樣式，然後在需要的時候顯示或隱藏起來：

```
nal Text text1 = new Text(s, SWT.MULTI | SWT.WRAP | SWT.BORDER);
text1.setBounds(100,50,100,100);
text1.setVisible(true);
nal Text text2 = new Text(s, SWT.MULTI | SWT.BORDER);
text2.setBounds(100,50,100,100);
text2.setVisible(false);
```

在需要不折行版本的時候，執行下面的程式碼：

```
text1.setVisible(false);
text2.setVisible(true);
```

好吧，這是怪招，但還是能夠在 SWT 的樣式系統上發揮作用。這是目前最好的招數。

很幸運的，我的經驗顯示需要 widget 改變樣式的情形不多見。

大部分的文字編輯器能讓使用者指定折行或不折行。

## 加上 Scrollbar

如果你把 text 欄位填滿，文字會從上方消失。其實它還在，但只是看不到了——還是可以用鍵盤上的方向鍵來移動到最上面。然而，使用者會期待不必經由方向鍵就可以上下瀏覽 text 欄位。要解決這個問題，你就需要對 text 欄位加入 scrollbar。

## 我該怎麼做？

跟斯斯一樣，scrollbar 也有兩種，水平跟垂直的。要使用哪一種要看你在哪個方向上有需要用到捲動。因為範例 5-3 使用 WRAP 樣式而不需水平 scrollbar，所以只要垂直的 scrollbar 就好。

指定 scrollbar 也是透過傳入正確的樣式屬性就好。有兩種可以選擇－SWT.H\_SCROLL 與 SWT.V\_SCROLL。要加入垂直的 scrollbar，只要在傳給 constructor 中的樣式屬性加入 SWT.V\_SCROLL 就好：

```
nal Text text1 = new Text(s, SWT.MULTI | SWT.WRAP | SWT.BORDER | SWT.V_SCROLL);
```

對範例 5-3 做這樣修改後的結果顯示在圖 5-6。



圖 5-6. 有垂直 scrollbar 的多行欄位

## 對 Text 事件反應

有時應用程式底層的商業邏輯需要你對使用者與 text 欄位互動時所發生的事件作出回應。一個例子是在讓使用者移動到 window 上的其他 widget 之前先對現在欄位所輸入的內容作資料驗證動作。另外一例是很常見的介面功能是在欄位收到輸入焦點的同時將欄位中的文字全部選取。

如同 MenuItem 與 ToolItem 這兩個 class，要讓 text 欄位對事件作回應就必須加入一個 Listener 給 Text 物件。Text 物件能夠回應下列三種 text 專有的 listener 型態：

- `ModifyListener`，這在欄位的文字變更時會被啟動。
- `SelectionListener`，這在欄位的文字被選取時作用。
- `VerifyListener`，在 `text` 欄位中的文字要被修改但使用者還沒看到修改內容之前啟動。

對這些事件的 `add()` 是定義在 `Control` class 中，它幾乎是所有 widget 的祖先。

`text` 欄位也可以回應其他幾個對所有 widget 都有定義的通用事件。

其中一個對 `text` 欄位來說很常用的事件是 `focus` 事件。`focus` 事件發生在欄位取得或失去輸入焦點的時候。這通常都是由使用者在欄位間切換或者點選所引發的。`focus` 事件是執行欄位層級驗證或選取文字的好地方。

## 我該怎麼做？

對項目選取是 `text` 欄位的一種功能，能夠在收到輸入焦點的時候全選欄位的內容。範例 5-4 展示出如何使用 `focus` 事件來建構出這樣的效果。

### 範例 5-4. 執行項目選取

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.FocusEvent;
import org.eclipse.swt.events.FocusListener;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.widgets.*;

public class TextFieldExample {

    Display d;
    Shell s;
    TextFieldExample() {
        d = new Display(-);
        s = new Shell(d);
        s.setSize(250,250);
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));
        s.setText("A Text Field Example");
        Text text1 = new Text(s, SWT.WRAP | SWT.BORDER);
        text1.setBounds(100,50,100,20);
        text1.setTextLimit(5);
        text1.setText("12345");
        Text text2 = new Text(s, SWT.SINGLE | SWT.BORDER);
        text2.setBounds(100,75,100,20);
        text2.setTextLimit(30);

        // 加入 focus listener
        FocusListener focusListener = new FocusListener() {
            public void focusGained(FocusEvent e) {
                Text t = (Text)e.widget;
            }
        };
    }
}
```

#### 範例 5-4. 執行項目選取 (續)

```
        t.selectAll();
    }
    public void focusLost(FocusEvent e) {
        Text t = (Text)e.widget;
        if(t.getSelectionCount() > 0){
            t.clearSelection();
        }
    }
};
text1.addFocusListener(focusListener);
text2.addFocusListener(focusListener);
s.open();
while(!s.isDisposed()){
    if(!d.readAndDispatch())
        d.sleep();
}
d.dispose();
}
```

範例 5-4 會建構出顯示在圖 5-7 的 window。注意到當 window 首次出先且第一個 text 欄位取得焦點時，文字會被預先選取。當欄位因為點選另一個 text 欄位而失去焦點時，其文字就會被 deselected 過。你可以在兩個欄位之間來回的切換、輸入文字以觀察它的效果。

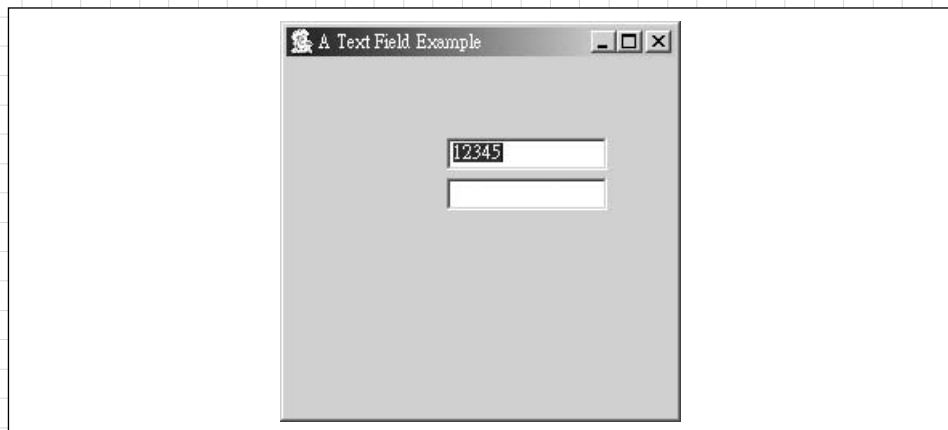


圖 5-7. 使用 focus listener

## 執行欄位驗證

focus 事件也可以用來執行欄位層級的驗證。你必須要在進行這種驗證的時候執行兩項工作。首先必須要在事件發生的同時將欄位的文字與期望值作比對，例如說比較輸入文字的長度是否正確。再來是於輸入文字沒有通過驗證的時候防止焦點離開 text 欄位。

### 我該怎麼做？

要執行欄位層級的驗證與預防使用者在沒有正確通過驗證測試前切換到欄位之外，你必須再度使用 FocusListener。範例 5-5 展示出如何對 text2 執行驗證來比對使用者是否輸入了正確的值。

範例 5-5. 在 FocusListener 中執行欄位層級的驗證

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.FocusEvent;
import org.eclipse.swt.events.FocusListener;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.widgets.*;

public class TextFieldExample {
    Display d;
    Shell s;
    TextFieldExample() {
        d = new Display( );
        s = new Shell(d);
        s.setSize(250,250);
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));
        s.setText("A Text Field Example");
        nal Text text1 = new Text(s, SWT.WRAP | SWT.BORDER);
        text1.setBounds(100,50,100,20);
        text1.setTextLimit(5);
        text1.setEchoChar('*');
        nal Text text2 = new Text(s, SWT.SINGLE | SWT.BORDER);
        text2.setBounds(100,75,100,20);
        text2.setTextLimit(30);

        // 加入 focus listener
        FocusListener focusListener = new FocusListener() {
            public void focusGained(FocusEvent e) {
            }
            public void focusLost(FocusEvent e) {
                Text t = (Text)e.widget;
                if(t==text2)
                {
                    if(t.getText().length( ) < 3)
                }
            }
        };
    }
}
```

### 範例 5-5. 在 FocusListener 中執行欄位層級的驗證 (續)

```
        {
            t.setFocus( );
        }
    }
};
text1.addFocusListener(focusListener);
text2.addFocusListener(focusListener);
s.open( );
while(!s.isDisposed( )){
    if(!d.readAndDispatch( ))
        d.sleep( );
}
d.dispose( );
}
```

## 發生了什麼事？

對 `println()` 的  
呼叫應該會在真  
正應用的程式中  
被換成  
`MessageBox`。  
見第七章。

關鍵程式碼是在 `FocusListener` 中的 `focusLost()` 這個 `method`：

```
public void focusLost(FocusEvent e) {
    Text t = (Text)e.widget;
    if(t==text2)
    {
        if(t.getText().length() < 3)
        {
            System.out.println("Data is less than 3 characters");
            t.setFocus( );
        }
    }
}
```

因為驗證規則的使用要看是哪一個 `text` 欄位擁有輸入焦點，所以第一個動作是要判斷事件由哪個 `widget` 所引發的。SWT 讓這個工作變得很簡單。每個 `Listener` 類型都帶有相關聯的事件物件。包裝在事件物件中的資訊是與事件發生時應用程式的狀態有關。在使用 `FocusListener` 時，`FocusEvent` 物件會被傳給 `focusLost()` 這個 `method`。物件內部是個帶有引發事件 `widget` 參考的欄位。你可以使用 `==` 運算元來對 `text` 欄位與這個參考作比較：

**【譯註】** 使用者  
有可能不知道應  
該輸入什麼值才  
能過關，小心別  
寫出一個進來之  
後就永遠無法離  
開的欄位。

```
Text t = (Text)e.widget;
if(t==text2)
```

如果焦點將離開 `text2`，`if` 述句中的等式檢查陳述就會為 `true`。然後在 `if` 內的程式碼對 `text2` 呼叫 `getText()` 所傳回的 `String` 長度。如果長度小於三個字元，對 `setFocus()` 的呼叫會強迫焦點留在 `text2` 上。

## 建構只做顯示的 Text 欄位

有時候你會需要將文字顯示給使用者看而不允許使用者修改或以其他方式來與其互動－建構只做顯示的 `text` 欄位能夠達到這個效果。

### 我該怎麼做？

要建構唯讀的 `text` 欄位，可以設定 `text` 欄位的 `enabled` 屬性為 `false`。這會建構出如圖 5-8 所示的灰暗 `text` 欄位。

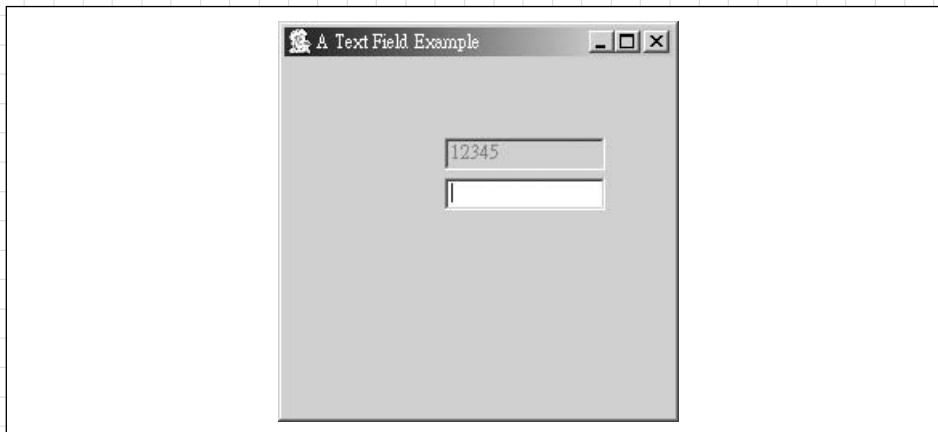


圖 5-8. 被抑止的 `text` 欄位

使用者可以看到文字，但無法與它互動。通常這種欄位稱為被 `disabled` 的：

```
text1.setEnabled(false);
```

### 有關於...

想要防止編輯 `text` 欄位，但又還想讓使用者能夠選取文字作複製並貼到其他欄位甚或其他應用程式時該怎麼辦？如果想讓使用者能夠與文字互動時，抑止 `text` 欄位不是適當的做法。對於這種情況，應該要使用 `setEditable()` 這個 `method`：

```
text1.setEditable(false);
```

這會讓 `text` 欄位在各方面都很正常，除了使用者無法更改文字之外。

【譯註】何不使用 `Label` 這個 widget 呢？

## 建構密碼欄位

有時候你會需要建構能輸入文字的欄位，但又想防止他人窺視內容。輸入密碼的欄位就是個典型的例子。因為讓輸入的文字顯示出來是不安全的，可以使用指定的字元來代表所輸入的文字。

### 我該怎麼做？

要建構密碼型態的欄位，可以使用 `setEchoChar()` 這個 method：

```
text1.setEchoChar('*');
```

這個結果顯示在圖 5-9。

注意到傳入這個 method 的是個加上單引號的字元，而不是雙引號的字串。

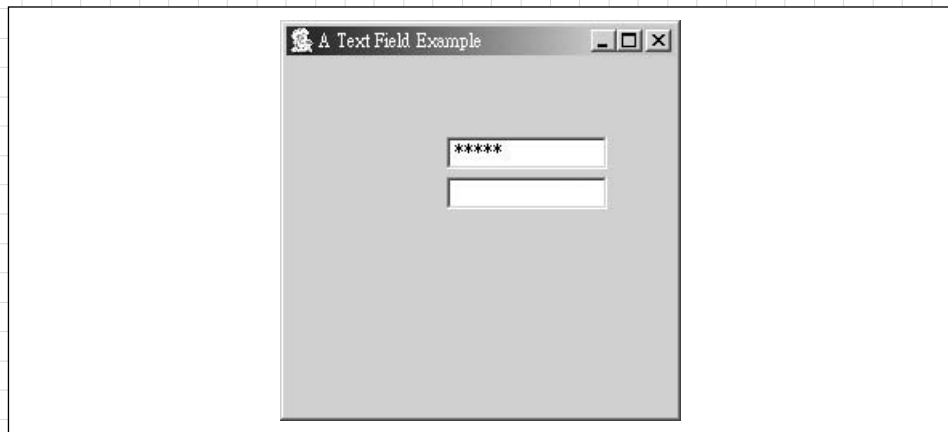


圖 5-9. 使用 `setEchoChar()` 的密碼欄位

如你所看到的，SWT 提供了非常有彈性的 widget 來顯示與輸入文字，這是每個 GUI 幾乎都有的基本元素。因為你使用的是 SWT，大部分的實作細節都隱藏在底層的 Java class 與原生函式庫中。