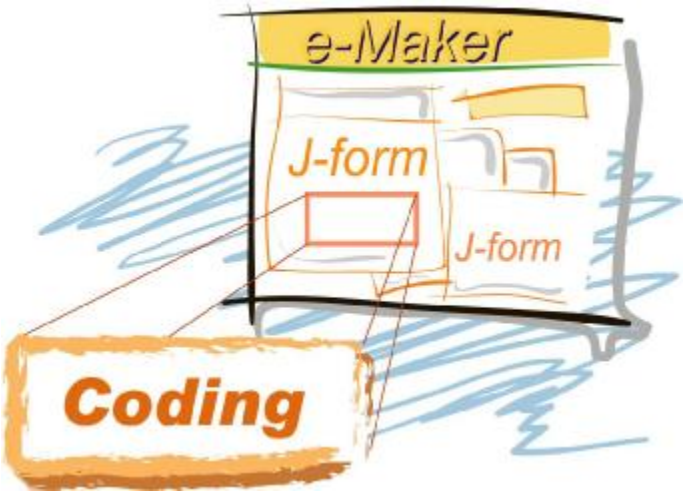
Emaker express 常用 API 與常用 Java 語法說明與範例



Emaker express 已經提供了一個相當簡易的系統開發環境，但是在商業系統的開發上，還是有些處理需要自己寫程式碼去完成，這一章的目的，就是介紹 Emaker express 中常用的 API 與常用的 Java 語法的運用，系統的開發就是這些 API 的組合運用了。

在閱讀本章之前，您需要對表單設計有基礎的了解，表單的設計，請參考使用手冊。

■ 常用 API 與說明

一般欄位使用的 API	
getValue();	取的物件的值, 傳回值爲字串
setValue();	設定物件的值
setEditable();	設定物件可不可以輸入資料
setVisible();	設定畫面上物件的可視狀態
setReference();	設定下拉選單中的值
requestFocus()	將滑鼠游標設定在某個物件上
取得系統物件, 系統狀態	
POSITION	表單目前的狀態 1, 新增狀態 3, 列印狀態 4, 查詢狀態 5, 流程狀態
getButton();	取得系統按鍵 按紐編號(1:新增 2:查詢 3:修改 4:刪除 5:列印 6:直接列印(不預覽) 7:詳細列表 8:流程記錄 9:重整畫面).
getSliderPanel();	取的查詢結果畫面上的筆數捲動物件
getFunctionName();	取得目前表單名稱
getInternalFrame();	在多視窗模式中,取的某表單的 JInternalFrame 物件
changeForm();	顯示另一個表單(須對此表單有權限)
showDialog();	以 Dialog 的方式顯示另一張表單
showForm();	跳出視窗顯示功能表單
表格物件常用的 API	
getTableData();	取的表格物件中的值, 傳回二維陣列
getValueAt();	去得表格物件中某個欄位的值
getSelectedRow();	取的 JTable 中目前游標所在的資料列
getSelectedColumn();	取的 JTable 中目前游標所在的資料行
setTableData();	設定表格物件中的值
setValueAt();	設定表格物件中某個欄位的值
getRowCount();	取的 JTable 中目前的資料筆數
getTableDataSorted();	取得排序過的表格物件中的值,傳回二維陣列
資料庫常用的 API	
getTalk();	取得資料庫連結
queryFromPool();	執行查詢的 SQL 傳回二維陣列
execFromPool();	執行異動資料庫(insert, delete, update)的 SQL
其他 Swing 物件常用的 API	
getLabel ();	取得 JLabel 物件
getButton()	取得 JButton 物件。

getTextField()	取得 JTextField 物件。
getTextArea()	取得 JTextArea 物件。
getComboBox()	取得 JComboBox 物件。
getCheckBox()	取得 JCheckBox 物件。
其他	
put()	將物件放到系統共用的記憶體空間中(可用來做全域變數用)
get()	取的系統共用記憶體空間中的物件
action()、showForm()應用	立刻執行按鈕狀態

- eMaker 常用 Java 語法

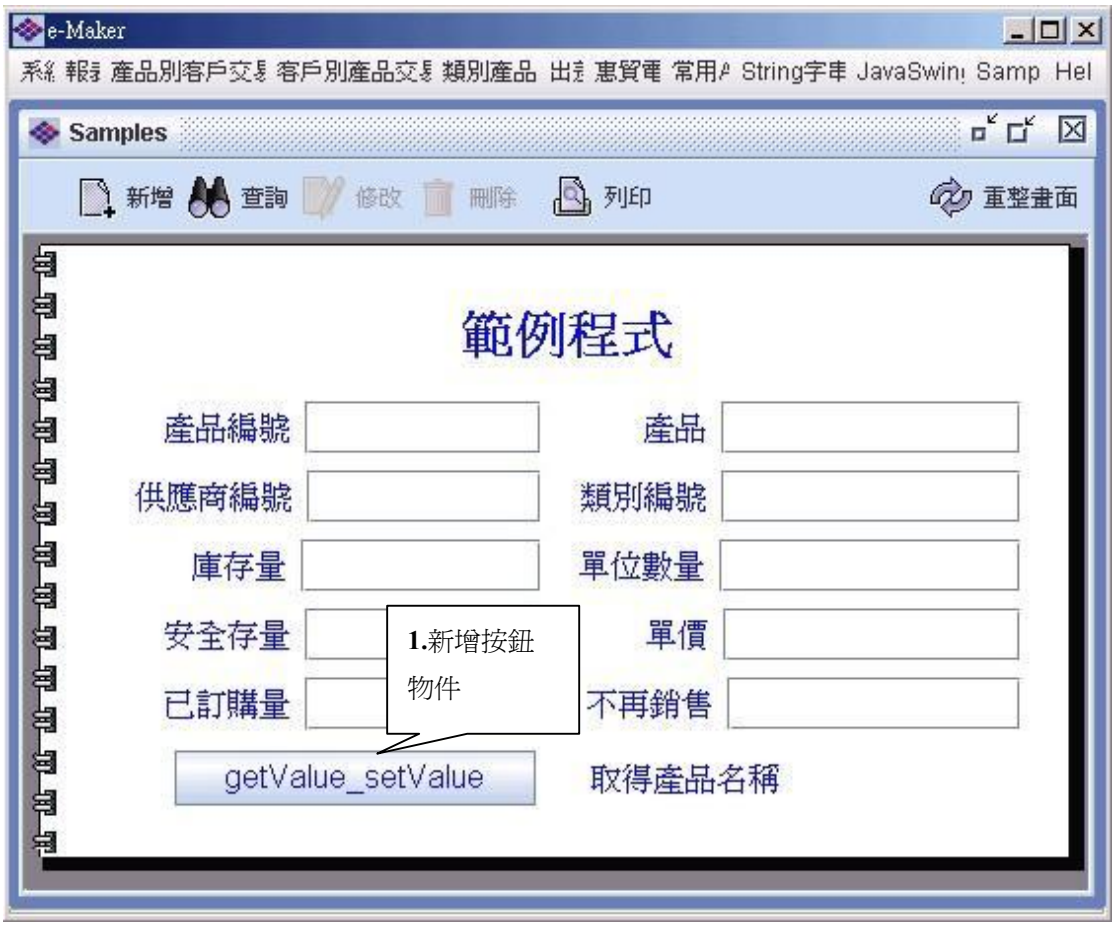
字串處理	
字串轉爲整數，浮點數..	
邏輯判斷	
For 迴圈	
二維陣列說明與運用	
Vecotor 物件說明與運用	
Hashtable 物件說明與運用	

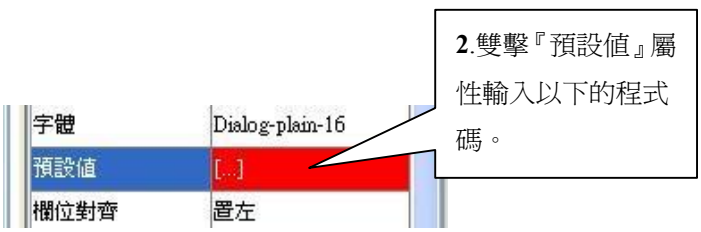
■ getValue()、setValue()

說明：

- ◆ getValue(“物件名稱”)：取得欄位資料。
- ◆ setValue(“物件名稱” , “值”)：設定欄位資料。

範例：利用 Button 來取得產品名稱，並將名稱設定到其他欄位 or 文字物件上。





```
//取值
String Value = getValue("產品").trim();
//設定值
setValue("text2" , Value );
return value;
```

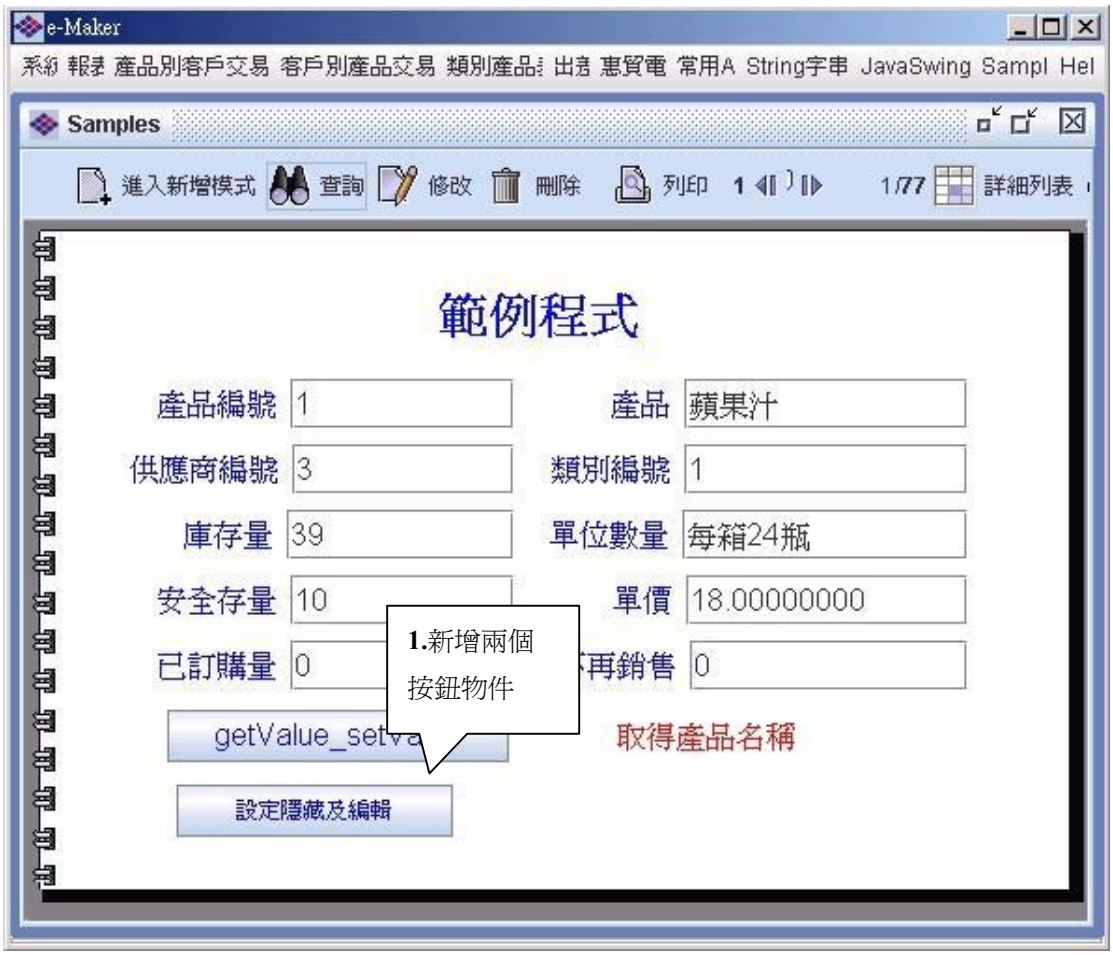
3.點選編譯按鈕完成後。

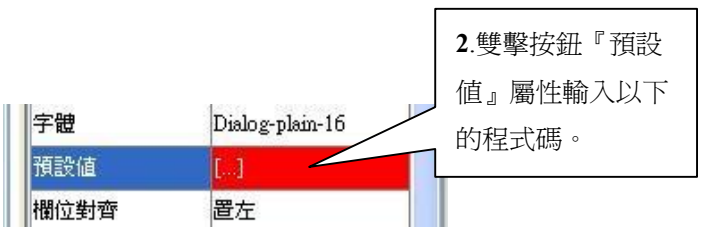
■ setEditable ()、setVisible()

說明：

- setEditable(“物件名稱” , boolean)：將欄位設定是否可編輯。
- setVisible(“物件名稱” , boolean)：將欄位設定是否隱藏。

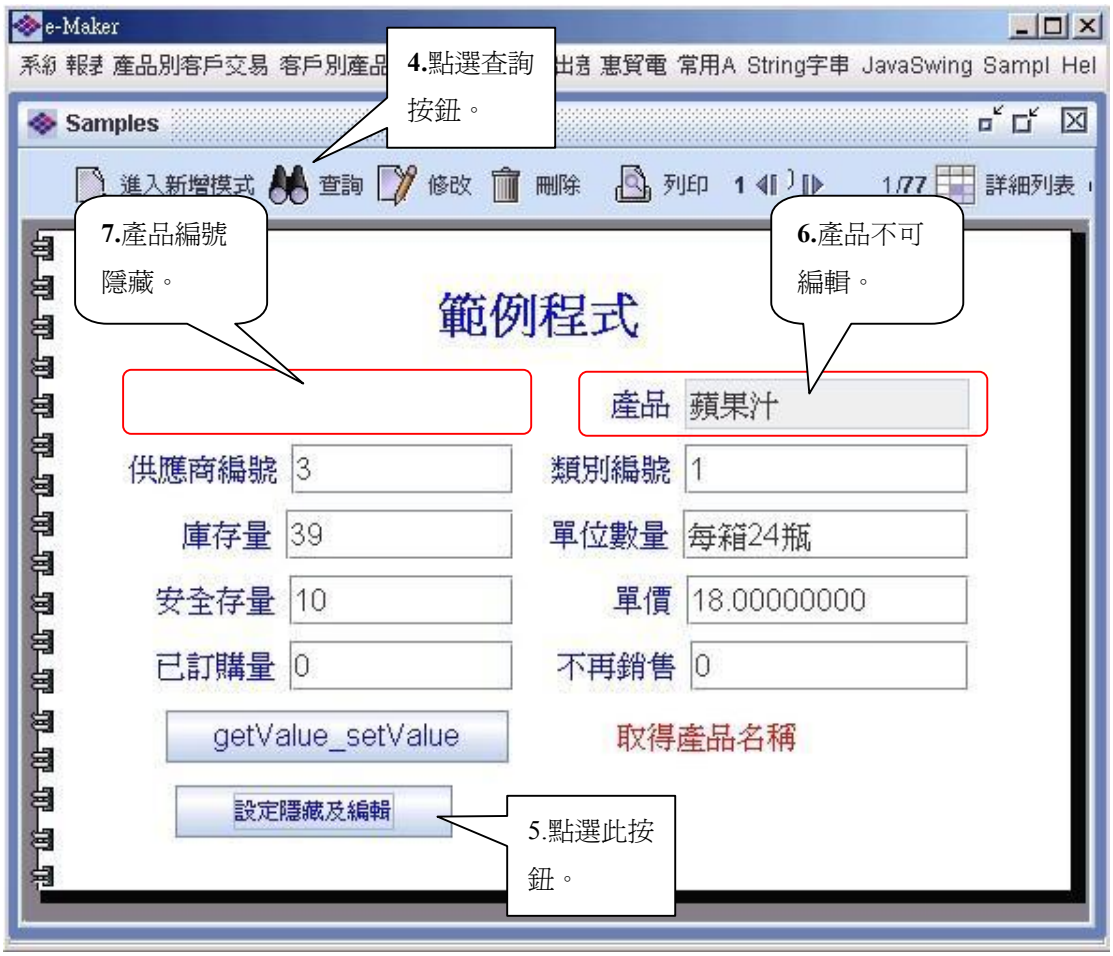
範例：新增兩個 Button 來將『產品編號』設定隱藏及『產品』設定不可編輯





```
//設定隱藏
setVisible("產品編號" , false);
//設定不可編輯
setEditable("產品" , false);
return value;
```

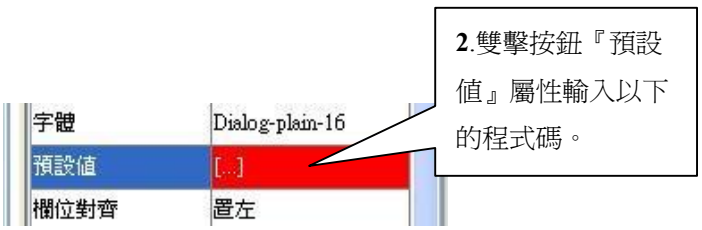
3. 點選編譯按鈕。



■ setReference()

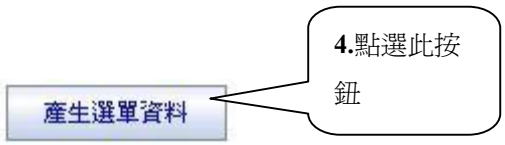
說明：setReference(“下拉選單物件名稱” , “顯示資料” , “實際資料”)：動態以程式控制下拉式選單中的值。

範例：點選『產生選單資料』按鈕後，將從資料庫中找到的值放到客戶名稱的下拉式選單中。



```
talk t=getTalk("nwcs");
Vector v1=new Vector();
Vector v2=new Vector();
String[][] ret=t.queryFromPool("select  客戶編號,公司名稱  from  客戶");
//在使用 qureyFromPool  可以加上 Sql 語法
for (int i=0; i< ret.length; i++){
            v1.addElement(ret[i][0]); //CustomerID
             v2.addElement(ret[i][1]);//CompanyName
}
setReference("客戶編號",v2,v1);

return value;
```

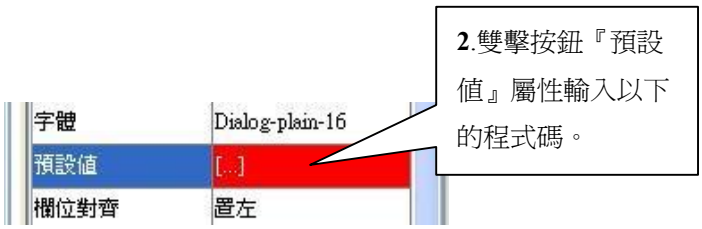
3.點選編譯按鈕完成後。



■ requestFocus()

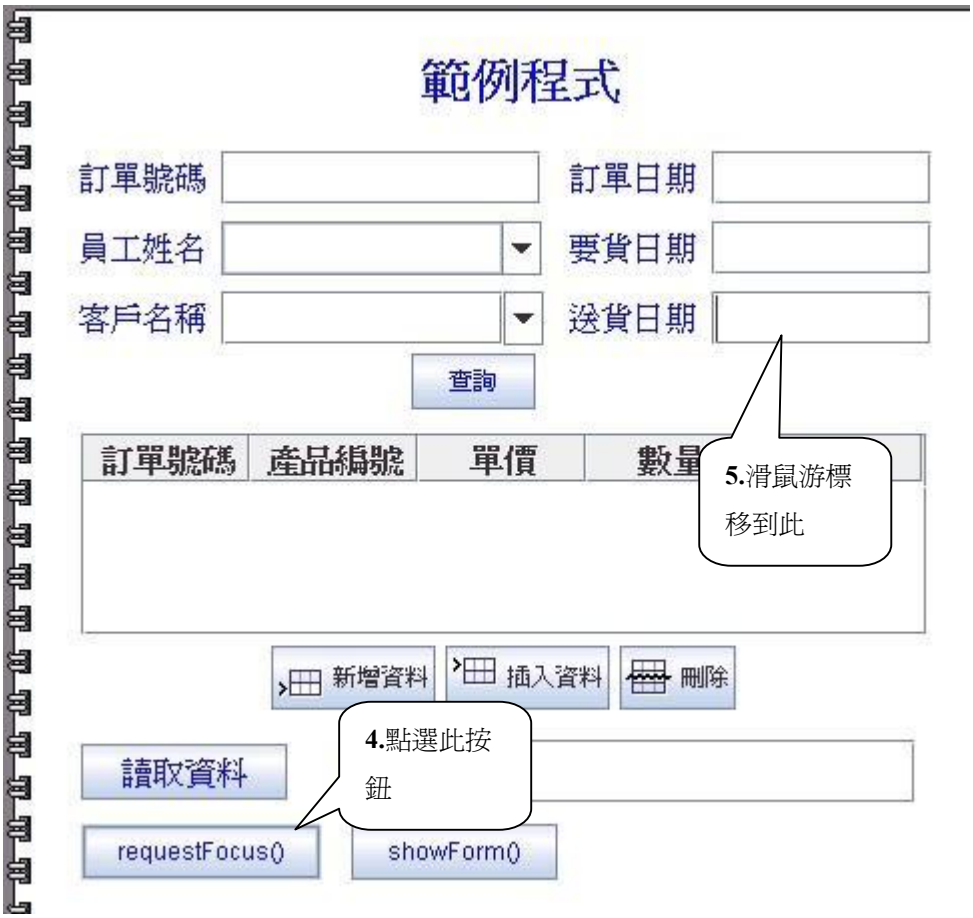
說明：requestFocus( )：將滑鼠游標設定在某個物件上

範例：點選『requestFocus()』按鈕，將移標移到『送貨日期』欄位上。



```
getcLabel("送貨日期").requestFocus();
return value;
```

3.點選編譯按鈕完成後。

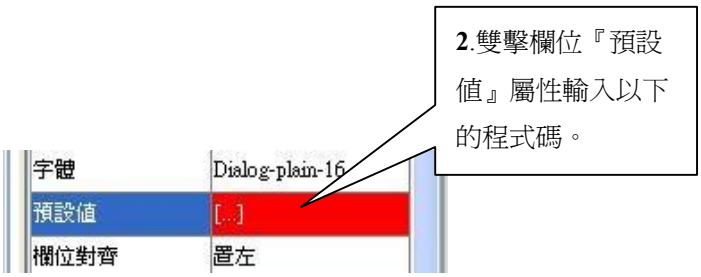


■ POSITION

說明：POSITION：目前的表單狀態

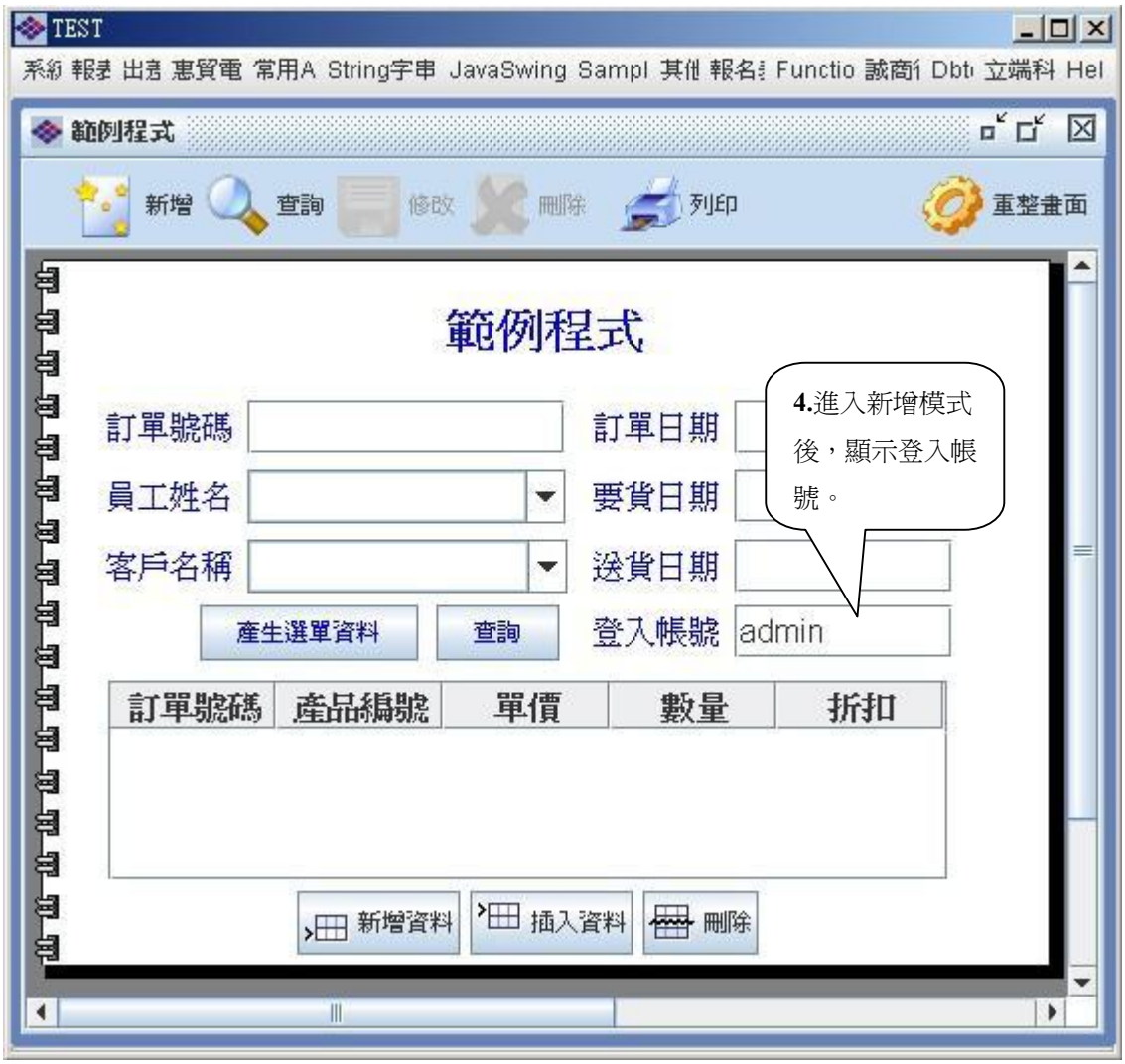
- ◆ 1、新增狀態 ：開啓表單就是處於新增狀態 or 點選進入新增模式。
- ◆ 3、列印狀態 ：點選列印按鈕後進入列印狀態。
- ◆ 4、查詢狀態 ：點選查詢按鈕後進入查詢狀態。

範例：新增登入帳號欄位物件，表單進入新增模式時，將帶出登入者帳號。



```
// 可自定欄位預設值
// 傳入值 value 原輸入值
// POSITION==1(新增模式)
if(POSITION==1)  {
        value=getUser();
}
return value;
```

3.點選編譯按鈕。

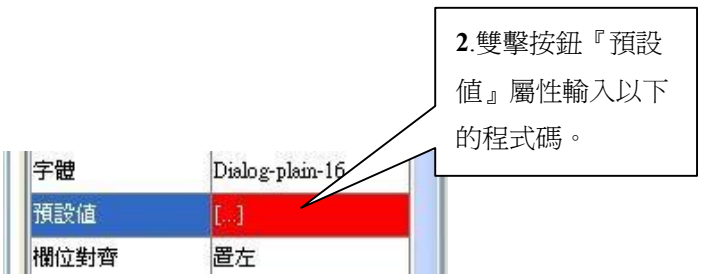


■ getButtton()

說明：getButton( )：取得功能列上的按鈕物件。

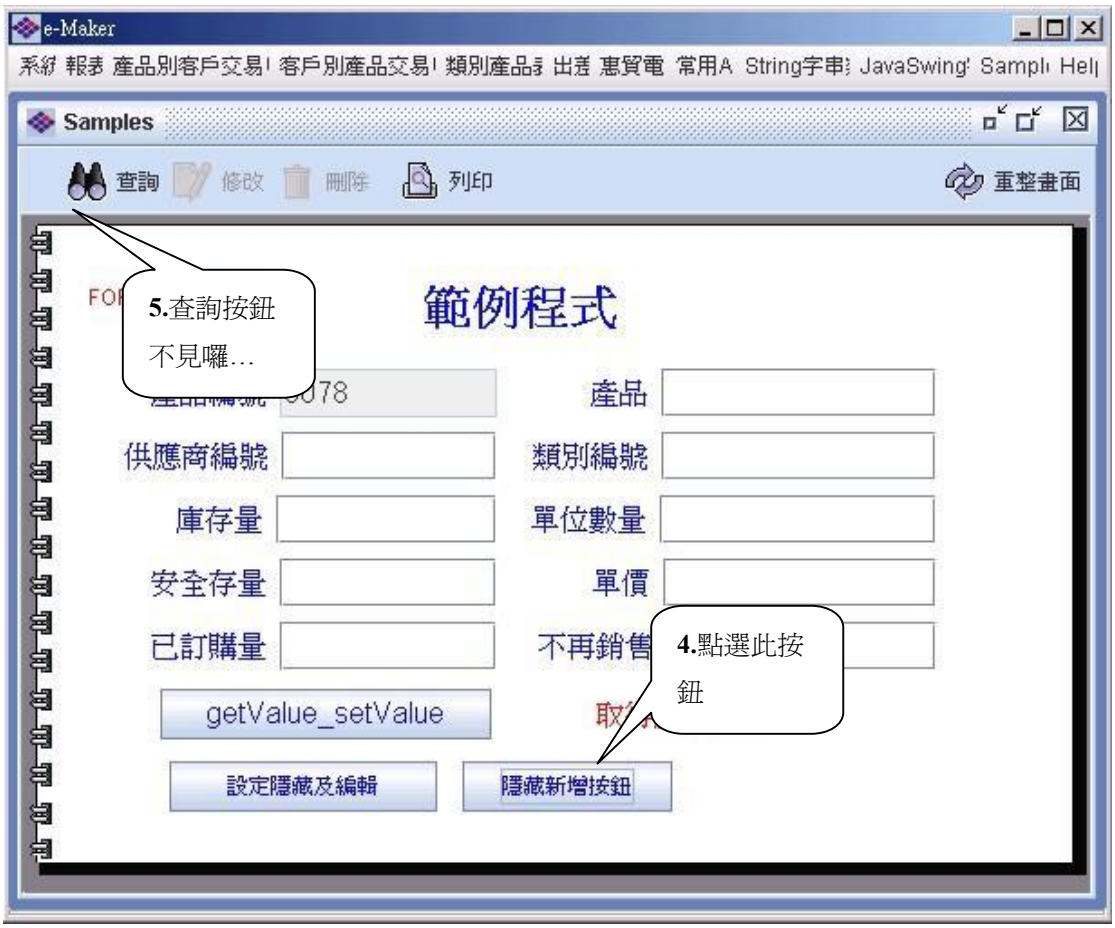
◆ 1:新增 2:查詢 3:修改 4:刪除 5:列印 6:直接列印(不預覽) 7:詳細列表 8:流程記錄 9:重整畫面

範例：將新增按鈕隱藏。



```
//取的新增按鈕
JButton jb = getButton(1);
jb.setVisible(false);
return value;
```

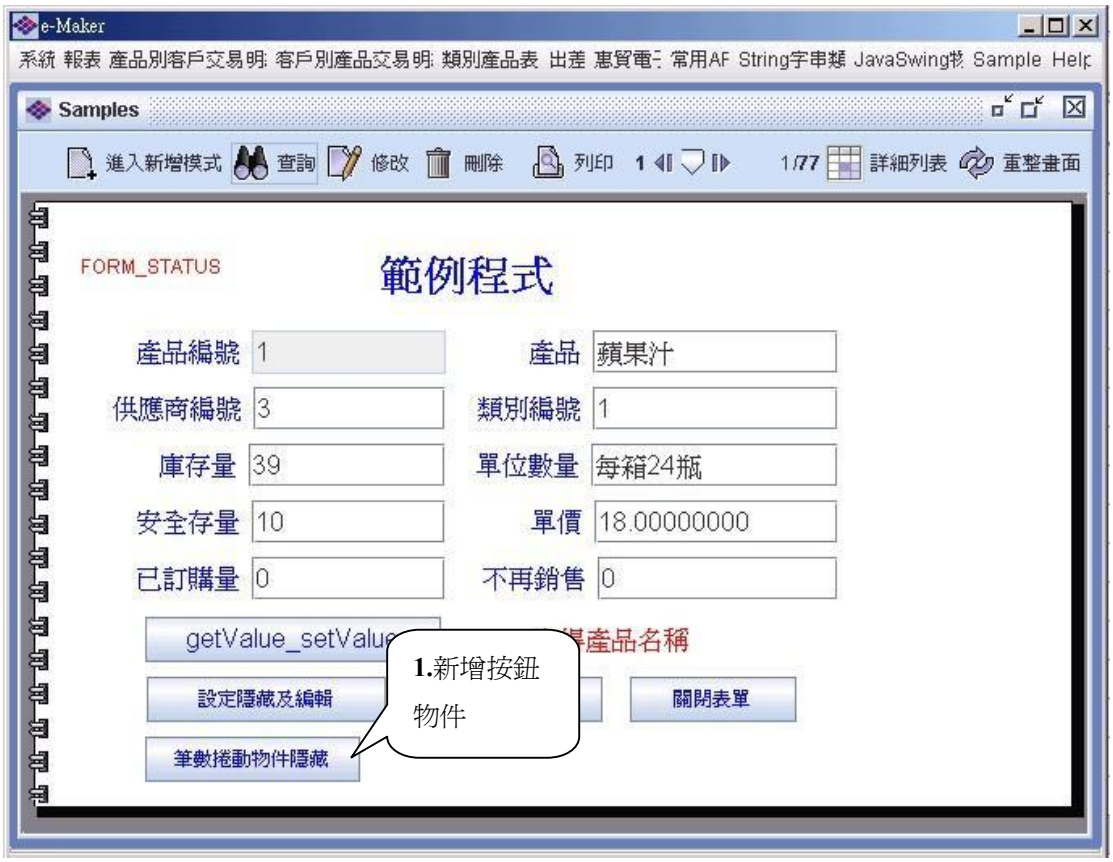
3.點選編譯按鈕。

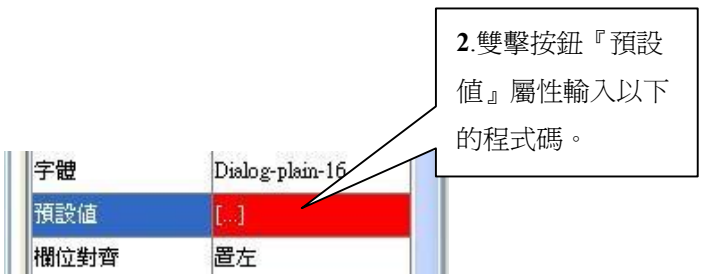


■ getSliderPanel()

說明：getSliderPanel()：畫面上的筆數捲動物件。

範例：將畫面上的捲動物件隱藏。





```
getSliderPanel().setVisible(true);
return value;
```

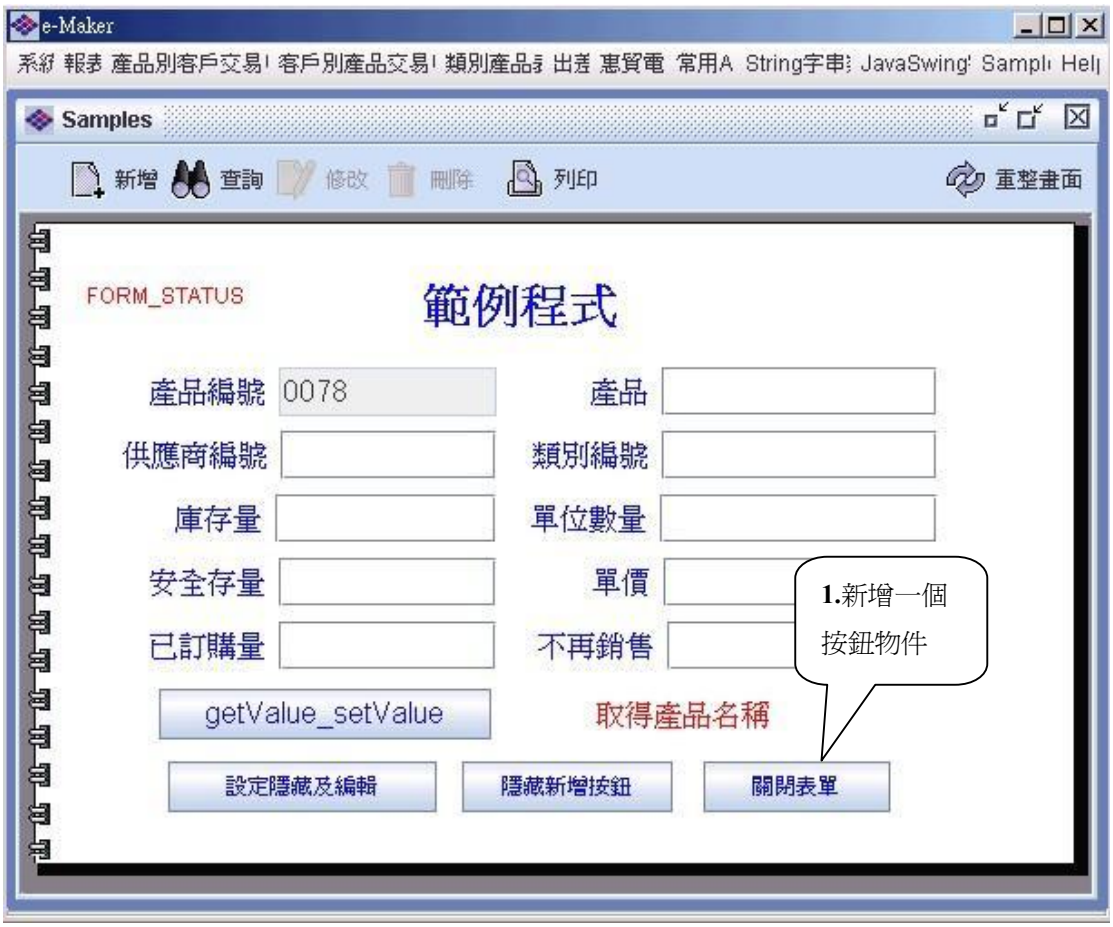
3.點選編譯按鈕。

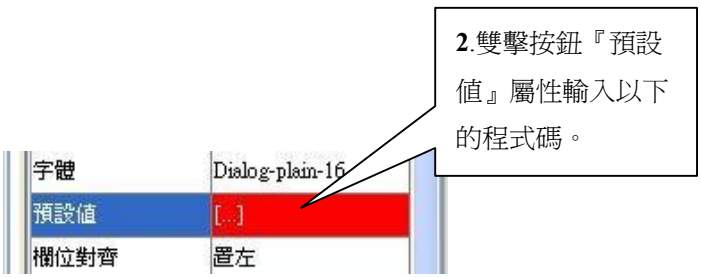
■ getFunctionName()、getInternalFrame()

說明：取得目前表單名稱

- ◆ getFunctionName()：取得目前表單名稱。
- ◆ getInternalFrame("表單名稱")：在多視窗模式中,取得某表單的 JInternalFrame 物件。

範例：使用 getFunctionName ()、getInternalFrame()來關閉表單。





```
JInternalFrame jif = getInternalFrame(getFunctionName());
//關閉表單
jif.setVisible(false);

return value;
```

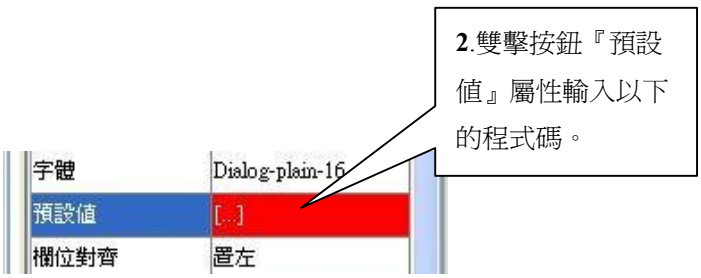
3.點選編譯按鈕。

4.點選『關閉表單』按鈕

■ changeForm()

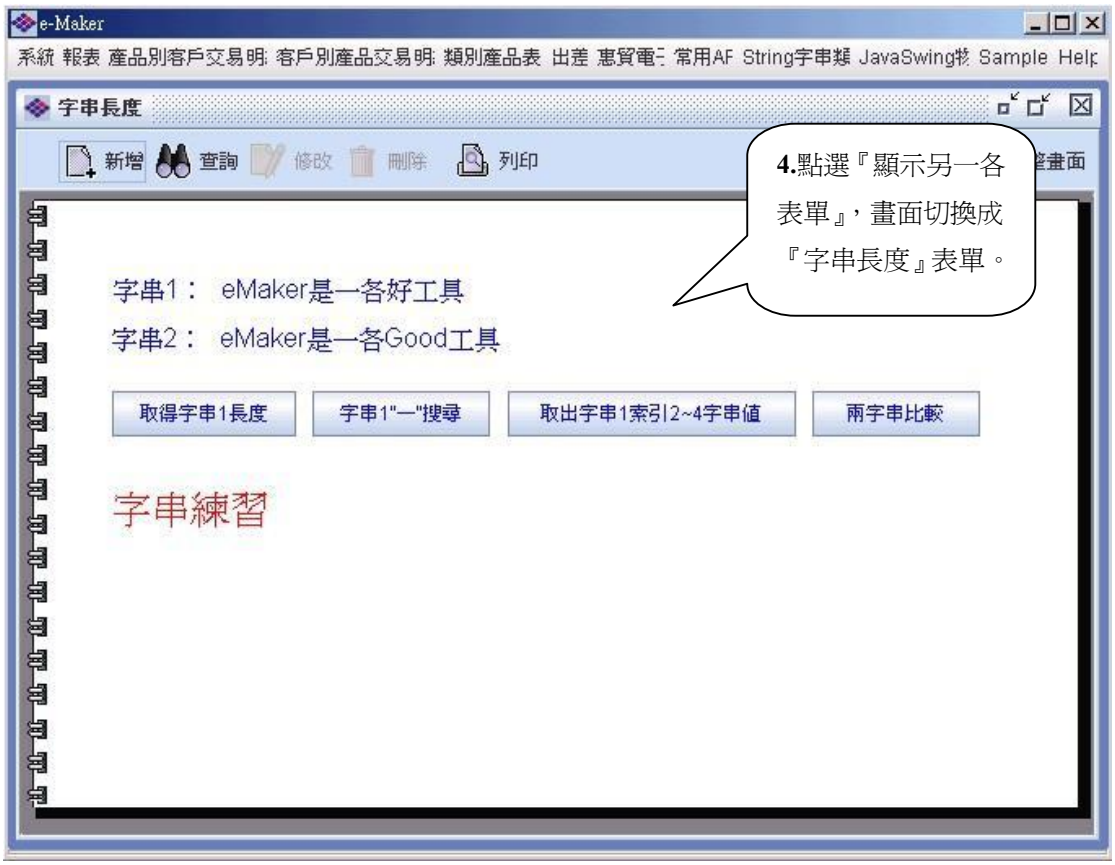
說明：changeForm(“表單名稱”)：切換表單(須對此表單有權限，User 在『帳號權限控制中心』，對此表單若無權限，則會無法切換)

範例：新增一個按鈕物件，當按下按鈕，會切換到另一張表單。



```
changeForm("字串長度");
return value;
```

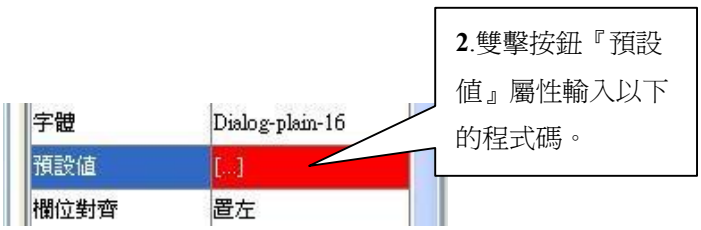
3.點選編譯按鈕。



■ showDialog()

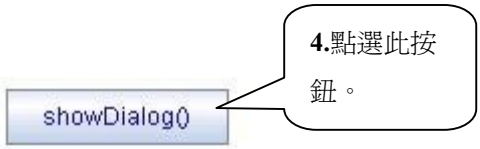
說明：showDialog(“表單名稱”)：以 Dialog 的方式彈跳出表單視窗(不檢查使用者是否有這項功能的權限)。

範例：點選按鈕以 Dialog 的方式彈跳出表單視窗。



```
//顯示員工表單
showDialog("員工");
return value;
```

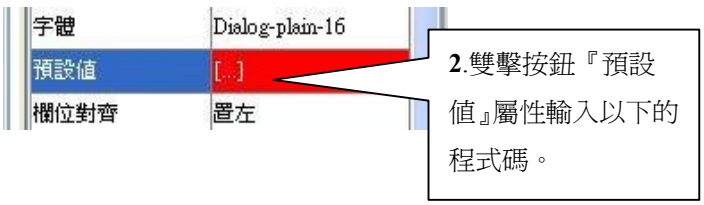
3.點選編譯按鈕完成後。



■ showForm()

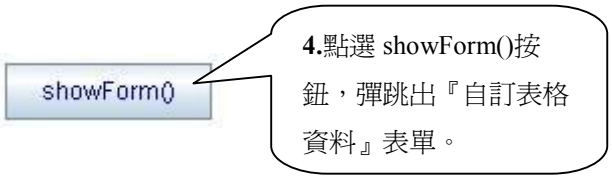
說明：showForm(“功能名稱”)：跳出視窗顯示功能表單(不檢查使用者是否有這項功能的權限)

範例：點選 showForm()按鈕，跳出『自訂表格資料』表單。



```
showForm("自訂表格資料");
return value;
```

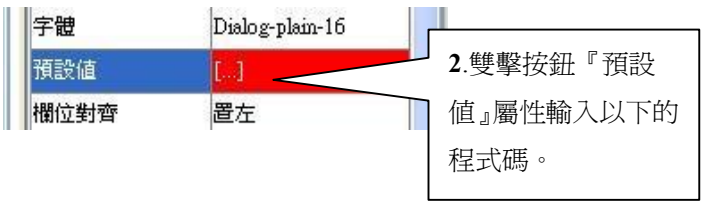
3.編譯完成後。



■ getTableData()

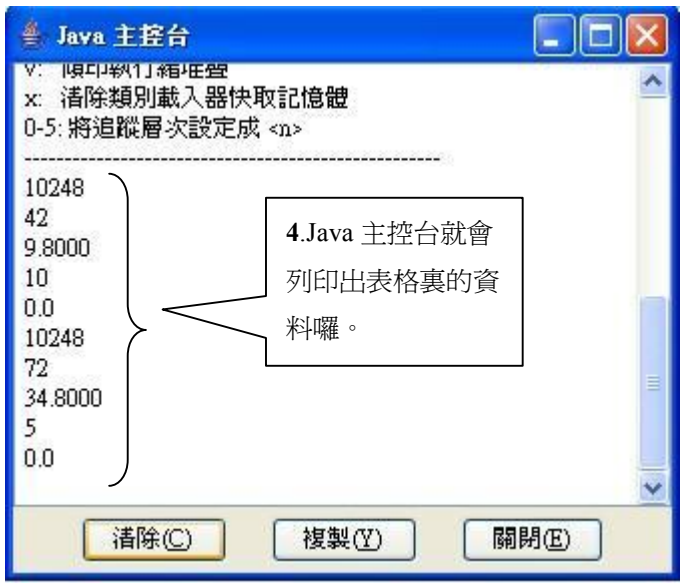
說明：getTableData(“表格名稱”)：取得表格資料。

範例：新增一個按鈕物件，於預設值屬性使用 getTableData()取得表格資料，並列印顯示出來。



```
if(POSITION != 4){              //4 查詢狀態
        message("請先查詢");
        return value;
}
//讀取表格資料,將結果輸出至 Java 主控台。
String ret [][] = getTableData("table1");
for(int i = 0 ; i < ret.length ; i++){
        for(int j = 0 ; j < ret[i].length ; j++){
                System.out.println(ret[i][j]);
        }
}
return value;
```

3.編譯完成後，於畫面上點選『讀取資料』按鈕。



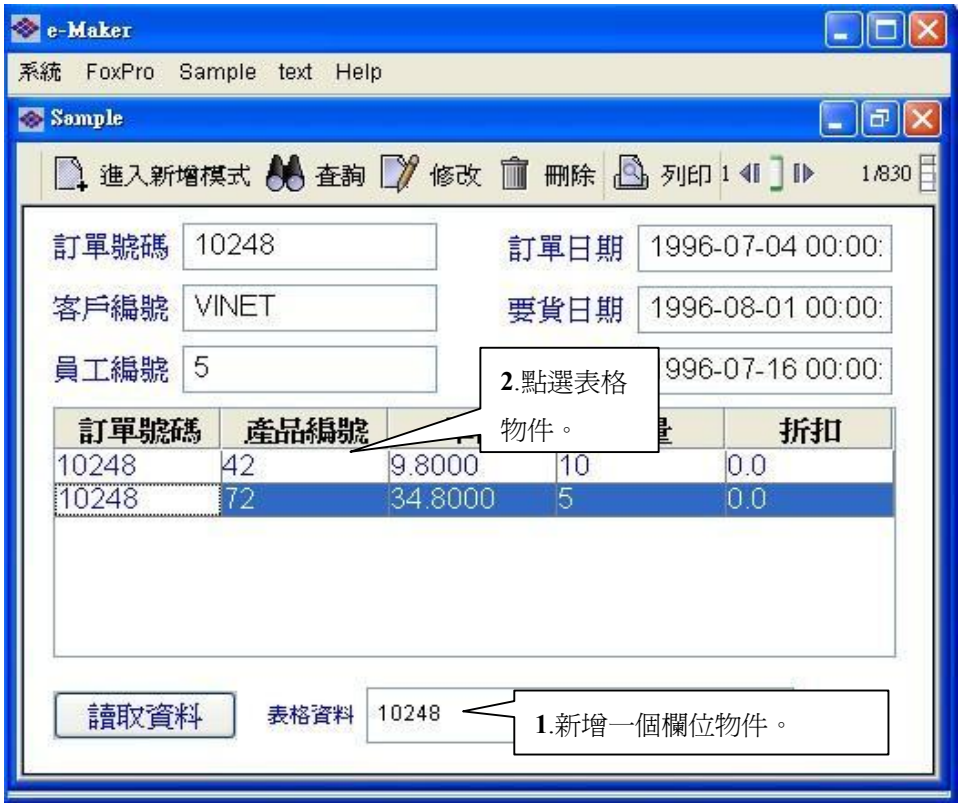
4.Java 主控台就會列印出表格裏的資料囉。

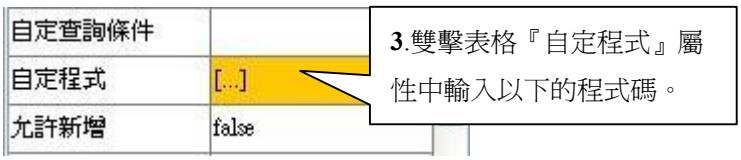
■ getValueAt()、getSelectedRow()、getSelectedColumn()

說明：

- getValueAt(“row” , “col”)：取得表格中欄位的資料。
- getSelectedRow()：取得表格選取的列數號。
- getSelectedColumn()：取得表格選取的欄位數號

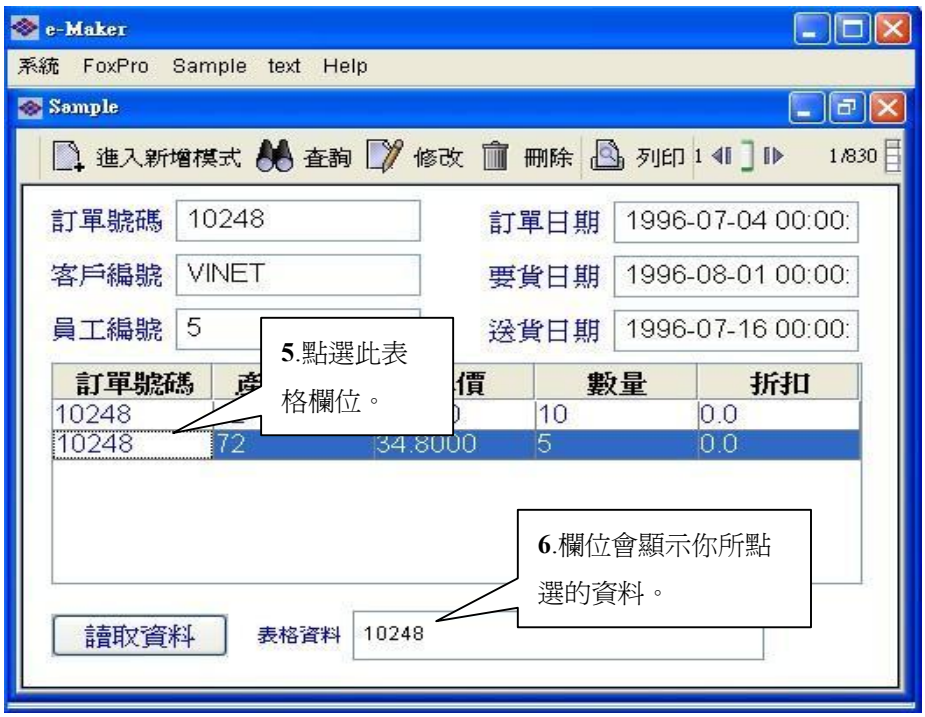
範例：點選表格資料，並將所點選的資料帶入另一個欄位資料中。





```
// 可自定當表格選取某一行後，要做什麼事 ,傳入值爲選取第幾行
//取得 Table 物件
JTable tb1 = getTable("table1");
//選取的列數
int row = tb1.getSelectedRow();
//選取的欄數
int col  = tb1.getSelectedColumn();
//取得表格中的資料
String s = (String)tb1.getValueAt(row, col);
setValue("field1" ,s);
return ;
```

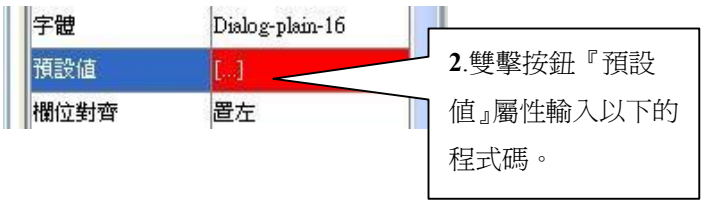
4. 編譯完成後，於畫面表格上點選表格欄位資料。



■ queryFromPool()

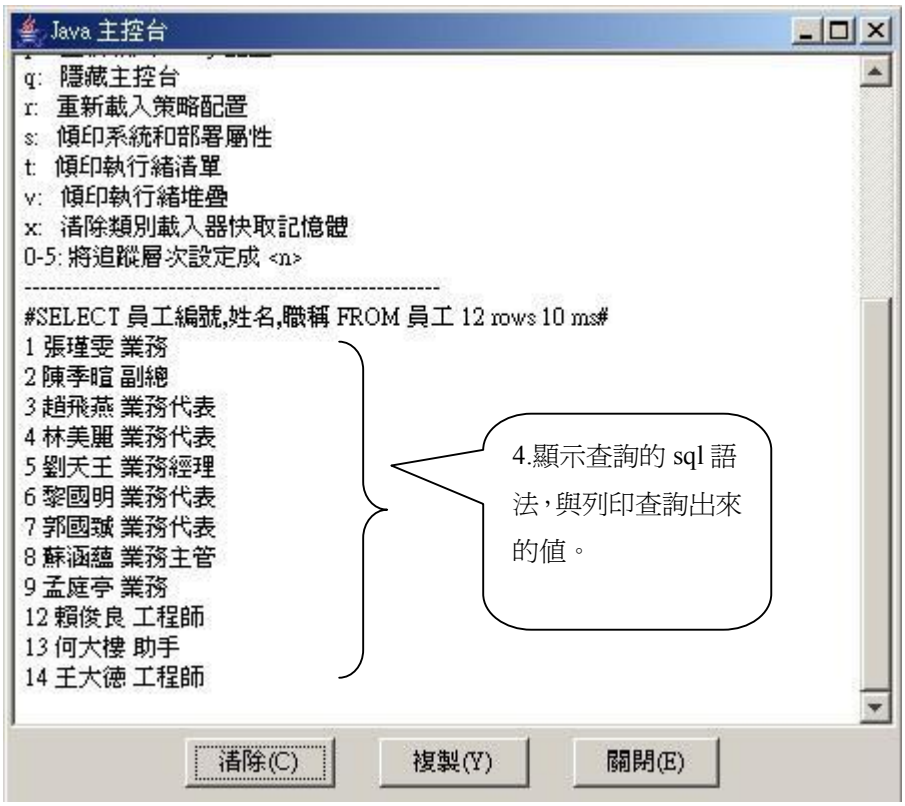
說明：queryFromPool(“sql 語法”)：執行查詢的 SQL 語法，並將值傳回二維陣列

範例：使用一個 Button 物件將執行查詢的 sql 語法，傳入二維陣列並將它列印在 Java 主控台。




```
talk t = getTalk("nwcs");
String sql = "SELECT  員工編號,姓名,職稱  FROM  員工";
String ret[][] = t.queryFromPool(sql);
if(ret.length > 0){
        for( int i = 0 ; i < ret.length ; i++){
                for(int j = 0 ; j < ret[i].length ; j ++){
                        System.out.print(ret[i][j]+" ");
                }
                System.out.println("");
        }
}
return value;
```

3. 編譯完成後，於畫面上點選『queryFromPool』按鈕。

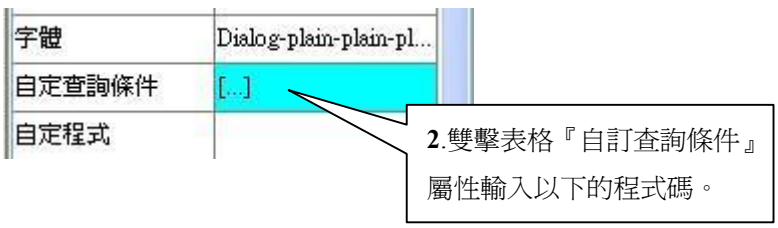


■ getTalk()、setTableData()、queryFromPool()

說明：

- getTalk(“資料庫名稱”)：取得連結資料庫物件。
- setTableData(“表格名稱” , “二維陣列值”)：設定表格資料。
- queryFromPool(“sql 語法”)：執行 SQL 指令。

範例：自訂表格資料。



```
// 回傳值爲自定查詢條件
// 回傳值必須是空白或以 and 開始,如 "and FIELD1='ABC'"
// 也可以回傳完整的 SQL 語法取代原設定的值 如 select distinct
display_field,data_field from table1 where type=100
//取得連結資料庫物件
talk t = getTalk("Nwind");
String sql = " SELECT 員工編號, 姓名, 名, 職稱 FROM 員工 ";
//執行 SQl 指令
String ret[][] = t.queryFromPool(sql);
//設定 table 資料
setTableData("table1", ret);
return "";
```

3. 編譯完成後，按鈕查詢資料。

員工編號	姓名	名	員工職稱
1	張瑾雯	Mary	業務
2	陳季暄	Bradley	業務經理
3	趙飛燕	Kim	業務
4	林美麗	Chris	業務
5	劉天王	Mike	
6	黎國明	Bill	
7	郭國強	Steven	
8	蘇涵蘊	Maggie	
9	孟庭亭	Linda	業務
12	賴傑良	Eddie	資深工程師
13	何大樓	David	助手
14	王大德	John	工程師

■ execFromPool()

說明：execFromPool(“sql 語法”)：執行異動資料庫(insert, delete, update)的 SQL

範例：對客戶資料表單，自行撰寫『修改』、『刪除』、『新增』按鈕。

3.撰寫修改按鈕預設值程式碼

```
String cusNo = getValue("客戶編號").trim();
String cusName = getValue("公司名稱").trim();
String contact = getValue("連絡人").trim();
String contactTitle = getValue("連絡人職稱").trim();
String phone = getValue("電話").trim();
String fax = getValue("傳真電話").trim();
String sql = "UPDATE  客戶 SET 公司名稱='"+cusName+"', 連絡人='"+contact+"',連絡人職稱
='"+contactTitle;
sql += "', 電話='"+phone+"', 傳真電話='"+fax+"' WHERE  客戶編號='"+cusNo+"'";
try{
      getTalk("nwcs").execFromPool(sql);
}catch(Exception e){
      message(e+"");
      return value;
}
message("異動成功");
return value;
```

4. 撰寫完成後，點選 Apply 編譯程式。

5. 撰寫刪除按鈕預設值程式碼。

```
String cusNo = getValue("客戶編號").trim();
Hashtable ht = new Hashtable();
ht.put("客戶編號" , "");
String sql = "DELETE FROM 客戶 WHERE 客戶編號='"+cusNo+"'";
try{
      getTalk("nwcs").execFromPool(sql);
}catch(Exception e){
      message(e+"");
      return value;
}
message("刪除成功");
action(2, ht);
return value;
```

6. 撰寫完成後，點選 Apply 編譯程式。

7. 撰寫新增按鈕預設值程式碼。

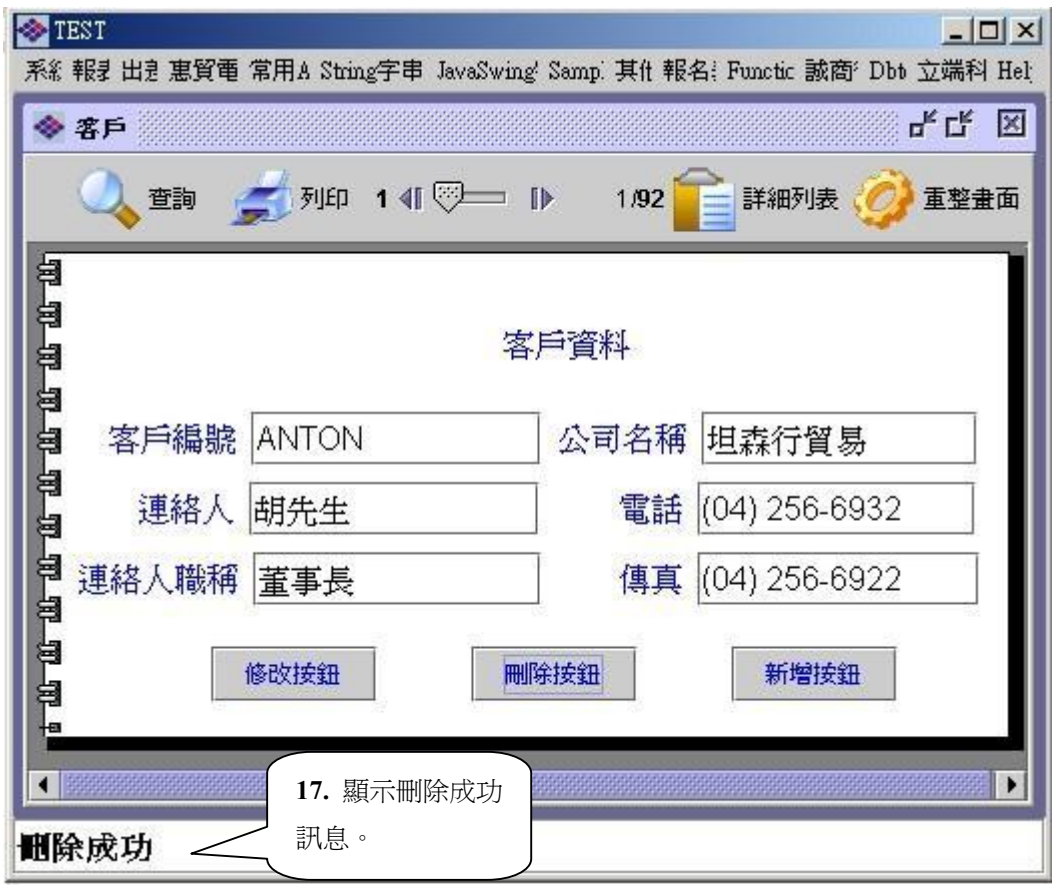
```
Hashtable ht = new Hashtable();
ht.put("客戶編號" , "");
String cusNo = getValue("客戶編號").trim();
String cusName = getValue("公司名稱").trim();
String contact = getValue("連絡人").trim();
String contactTitle = getValue("連絡人職稱").trim();
String phone = getValue("電話").trim();
String fax = getValue("傳真電話").trim();
String sql = "INSERT INTO 客戶 (客戶編號, 公司名稱, 連絡人, 連絡人職稱, 電話, 傳真電話)";
                    sql += " VALUES('"+cusNo+"' , '"+cusName+"' , '"+contact+"' , '"+contactTitle+"' , '"+phone+"' , '"+fax+"')";
try{
        getTalk("nwcs").execFromPool(sql);
}catch(Exception e){
        message(e+"");
        return value;
}
message("新增成功");
action(2 , ht);
return value;
```

8. 撰寫完成後，點選 Apply 編譯程式。

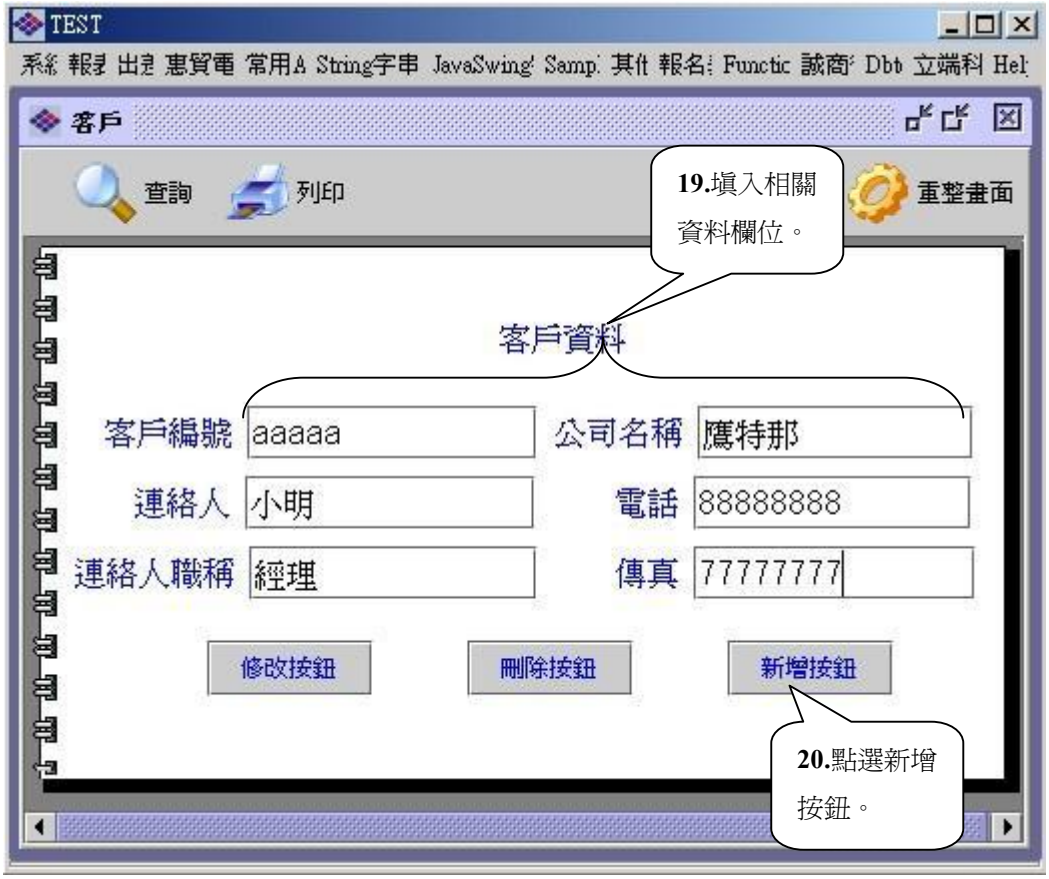
9.將第一筆資料查詢出來

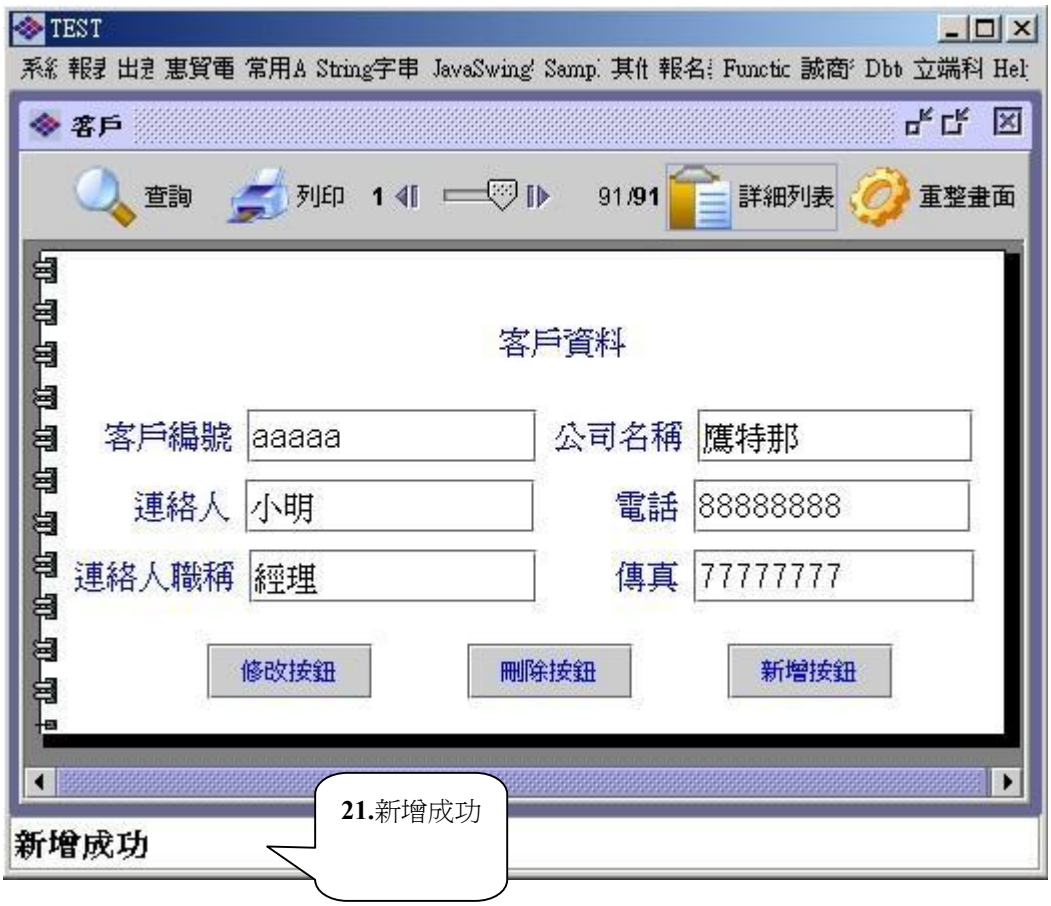
10.【修改按鈕範例】將『連絡人』欄位資料（陳小姐）改成（陳大明）

14.【刪除按鈕範例】將查詢出來的第一筆資料，客戶編號（ANATR）刪除。



18.【新增按鈕範例】新增一筆資料。





■ setValueAt()

說明：setValueAt(“表格名稱”, “值” , “列數”, “欄位名稱”)：設定表格欄位值。

範例：變更表格中的欄位資料

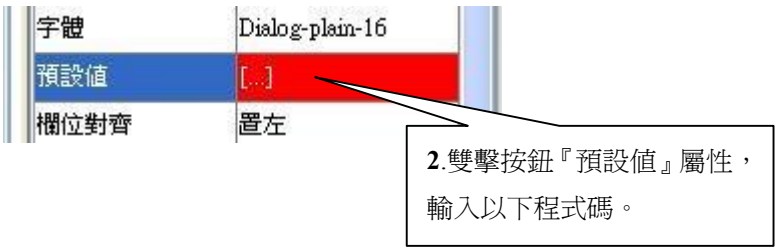
```
//變更表格欄位資料
//setValueAt(Tablename, value, row, ColumnName)
//Tablename -  表格的名稱.
//value -  資料.
//row -  列數.
//ColumnName -  欄位的名稱.
setValueAt("table1","總經理", 0, "職稱");
return value;
```

3.編譯完成後，將資料查詢出來後，按『變更資料』按鈕。

■ getRowCount()

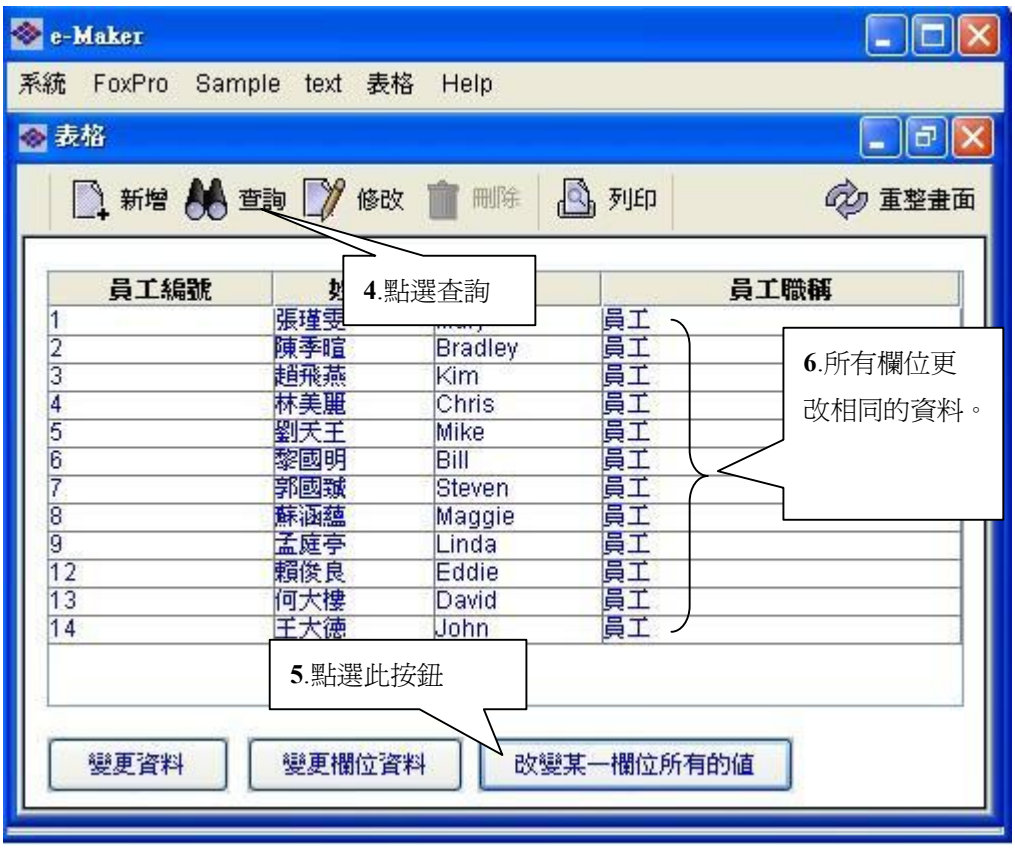
說明：getRowCount( )：取得表格資料中列數的總計數。

範例：將表格中某一欄位的資料全部更改所要的資料。



```
JTable tb1 = getTable("table1");
//取得 Table 的 Row 的總筆數
int row_num = tb1.getRowCount();
for(int i = 0 ; i < row_num ; i++){
        setValueAt( "table1" , "員工" , i , "職稱");
}
return value;
```

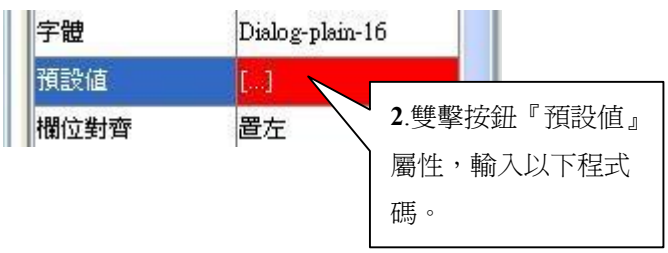
3.編譯完成後，將資料查詢出來後，按『改變某一欄位所有的值』按鈕。



■ getTableDataSorted()

說明：getTableDataSorted(“表格名稱”)：取得表格排序資料。

範例：將表格 1 排序過的資料存放到表格 2。



```
String ret[][] = getTableDataSorted("table1");
setTableData("table2",ret);
return value;
```

3.編譯完成後，將資料查詢出來後，按『排序資料』按鈕。

7.取得 table1 排序後資料。

■ getLabel ()、getButton()、getTextField()

getTextArea()、getComboBox()、getCheckBox()

說明：

- getLabel(“物件名稱”)：取得 JLabel 物件。
- getButton(“物件名稱”)：取得 JButton 物件。
- getTextField(“物件名稱”)：取得 JTextField 物件。
- getTextArea (“物件名稱”)：取得 JTextArea 物件。
- getComboBox (“物件名稱”)：取得 JComboBox 物件。
- getCheckBox (“物件名稱”)：取得 JCheckBox 物件。

範例：JLabel 應用變更標題文字。

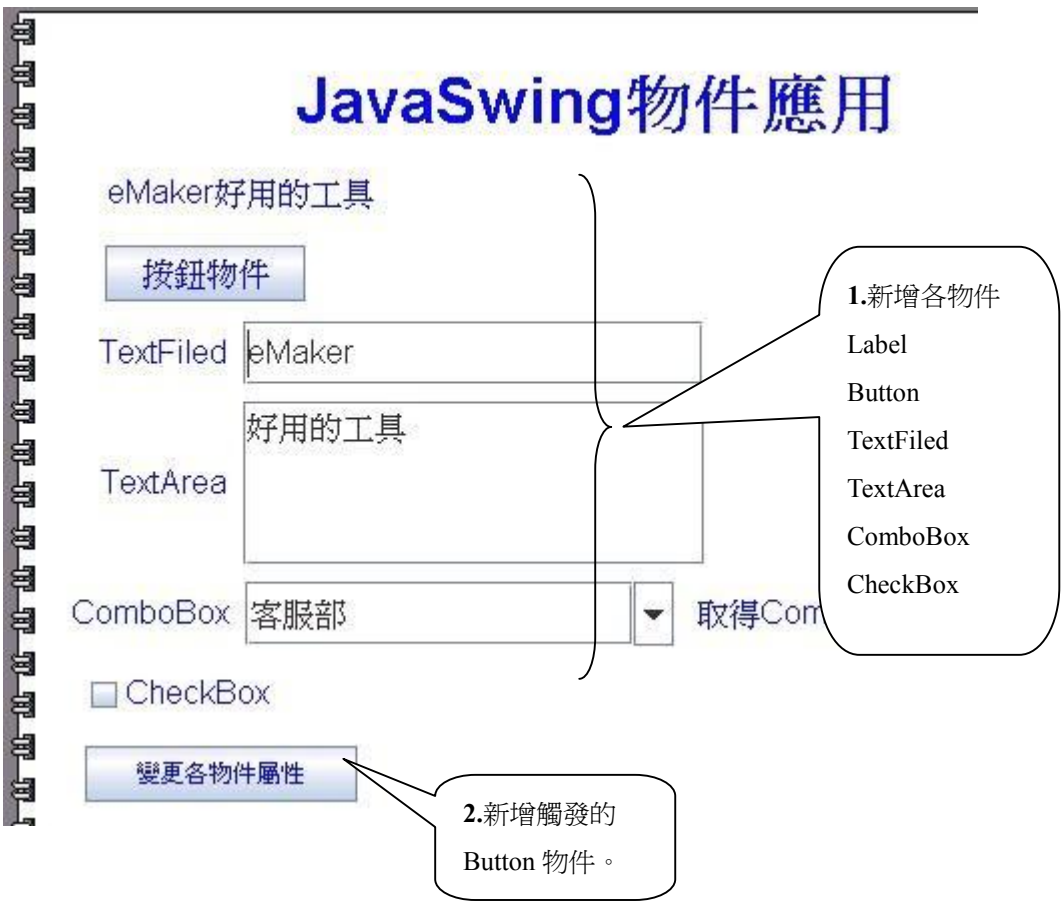
JButton 應用設定 Disabled

JTextField 應用設定 Disabled

JTextArea 應用設定 Disabled

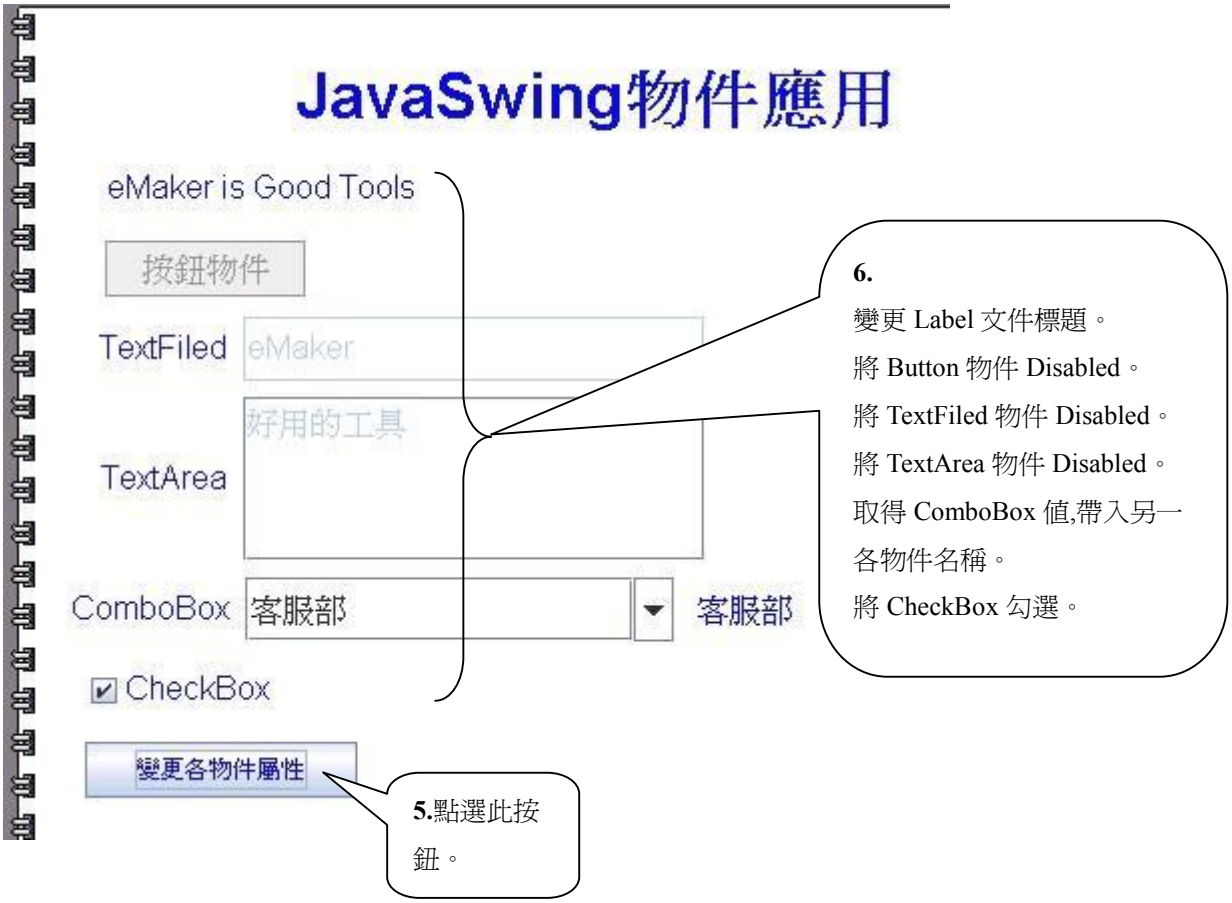
JComboBox 應用取得所選的資料

JCheckBox 應用設定爲打勾選項




```
//JLabel 應用設定標題文字
JLabel jl = getLabel("text1");
jl.setText("eMaker is Good Tools");

//JButton 應用設定 Disabled
JButton jb = getButton("button2");
jb.setEnabled(false);

//JTextField 應用設定 Disabled
JTextField jt = getTextField("field1");
jt.setEnabled(false);

//JTextArea 應用設定 Disabled
JTextArea jt1 = getTextArea("field2");
jt1.setEnabled(false);

//JComboBox 應用取得所選的資料
JComboBox jcb = getComboBox("ComboBox");
String textValue = (String)jcb.getSelectedItem();
setValue("text3" , textValue);

//JCheckBox 應用設定爲打勾選項
JCheckBox jchb = getCheckBox("field3");
jchb.setSelected(true);
return value;
```

4.點選編譯按鈕。

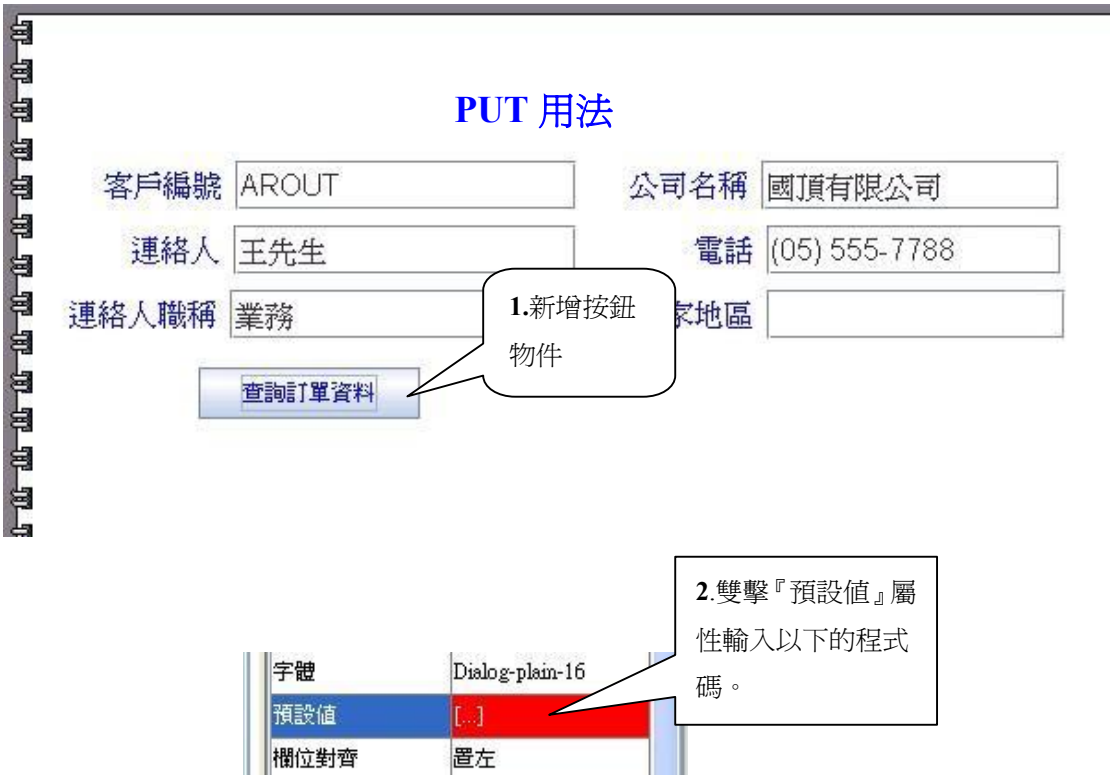

JavaSwing物件應用
eMaker is Good Tools
按鈕物件
TextFiled
eMaker
TextArea
好用的工具
ComboBox
客服部
客服部
CheckBox
變更各物件屬性
6.
變更 Label 文件標題。
將 Button 物件 Disabled。
將 TextFiled 物件 Disabled。
將 TextArea 物件 Disabled。
取得 ComboBox 值,帶入另一各物件名稱。
將 CheckBox 勾選。
5.點選此按鈕。

■ put()、get()

說明：

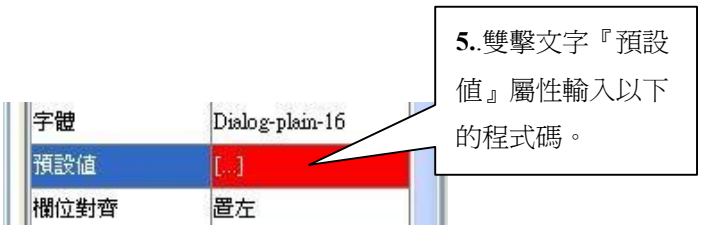
- put(“索引值” , “值”)：將物件放到系統共用的記憶體空間中(可用來做全域變數用)。
- get(“索引值”)：取得系統共用記憶體空間中的物件。

範例：利用客戶表單來查詢訂單主檔。



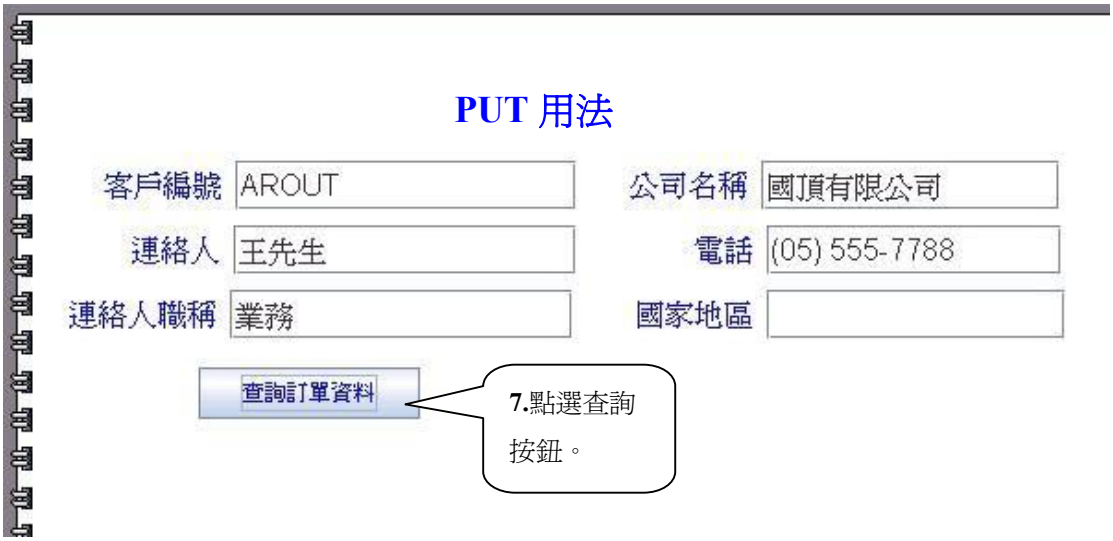

```
if(getValue("客戶編號").trim().equals("")){
        message("請先查詢公司資料");
        return value;
}
String comName = getValue("客戶編號");
//將值放入記憶體
put("comName" , comName);

String formName = "get".trim();
showDialog(formName);
//action(9);
return value;
```

3.點選編譯按鈕。




```
//取出記憶體內的值
String name = (String)get("comName");
talk t = getTalk("nwcs");
String sql = "select 訂單號碼,客戶編號,員工編號,訂單日期,要貨日期,送貨日期 from 訂貨主檔 where 客戶編號='"+name+"'";
String ret[][] = t.queryFromPool(sql);
setTableData("table2" , ret);
return value;
```

6.點選編譯按鈕。


PUT 用法
客戶編號
AROUT
公司名稱
國頂有限公司
連絡人
王先生
電話
(05) 555-7788
連絡人職稱
業務
國家地區
查詢訂單資料
7.點選查詢按鈕。


get
客戶訂單資料
訂單號碼 客戶編號 員工編號 訂單日期 發貨日期
10383 AROUT 8 1996-12-16 0... 1997-12-18 0...
10453 AROUT 1 1997-02-21 0... 1997-02-26 0...
10558 AROUT 1 1997-06-04 0... 1997-06-10 0...
10643 AROUT 6 1997-08-25 0... 1997-09-02 0...
10707 AROUT 4 1997-10-16 0... 1997-10-30 0... 1997-10-23 0...
10741 AROUT 4 1997-11-14 0... 1997-11-28 0... 1997-11-18 0...
10743 AROUT 1 1997-11-17 0... 1997-12-15 0... 1997-11-21 0...
10768 AROUT 3 1997-12-08 0... 1998-01-05 0... 1997-12-15 0...
10793 AROUT 3 1997-12-24 0... 1998-01-21 0... 1998-01-08 0...
10864 AROUT 4 1998-02-02 0... 1998-03-02 0... 1998-02-09 0...
10920 AROUT 4 1998-03-03 0... 1998-03-31 0... 1998-03-09 0...
10953 AROUT 9 1998-03-16 0... 1998-03-30 0... 1998-03-25 0...
11016 AROUT 9 1998-04-10 0... 1998-05-08 0... 1998-04-13 0...
8.顯示客戶的訂單資料

■ action()、showForm()應用

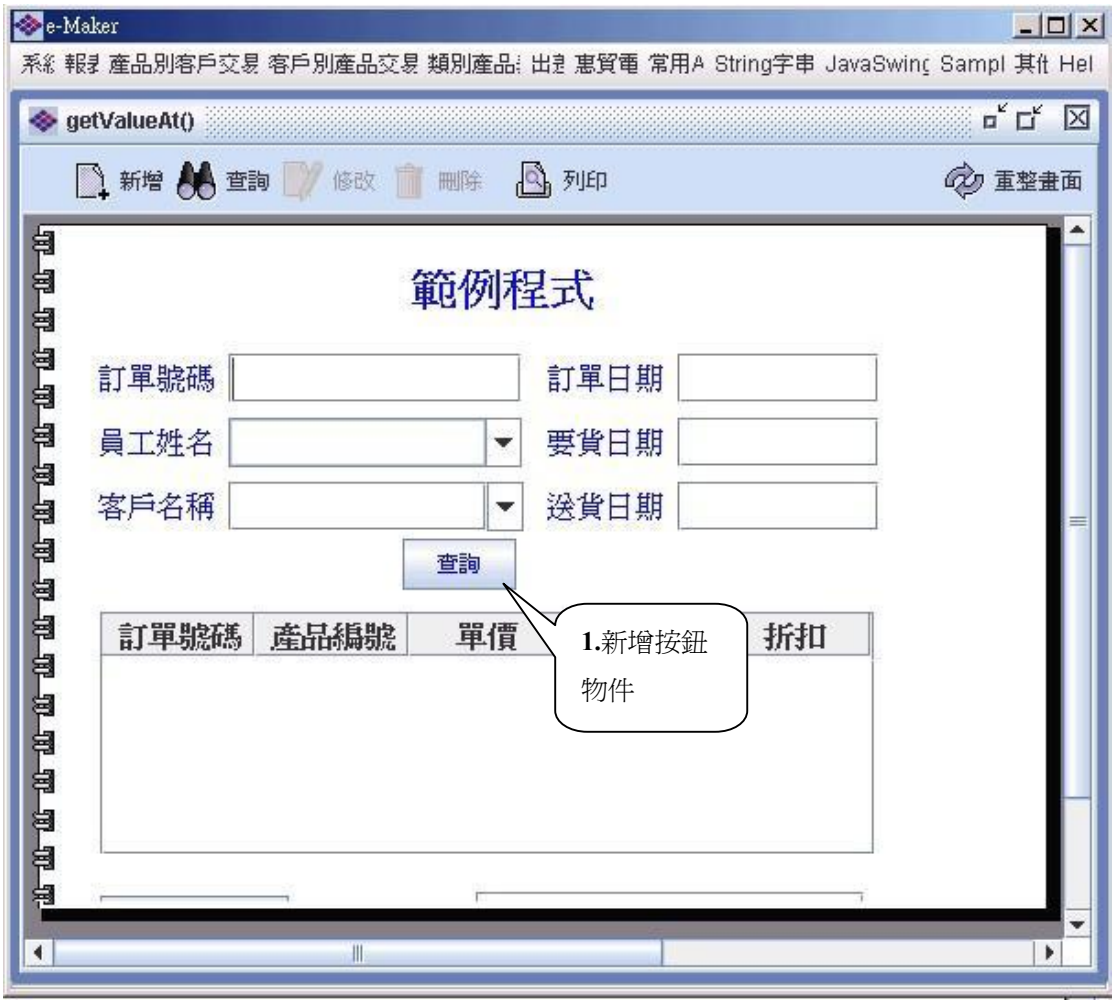
說明：

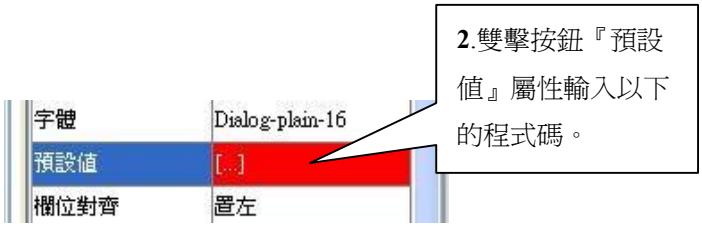
- action(“按鈕編號”)：立即執行按鈕。

  1:新增 2:查詢 3:修改 4:刪除 5:列印(先預覽) 6:直接列印(不預覽) 7:詳細列表 8:流程記錄 9:重整畫面

- action(“2” , “Hashtable ”)：帶入的 Hashtable 參數(僅對查詢按鈕有效)。
- showForm(“表單名稱”)：彈跳出表單視窗(User 在『帳號權限控制中心』對此表單無須有權限)。

範例：點選查詢按鈕顯示客戶名稱表單。





```
//點選要顯示出來的 Frame
String formName = "客戶資料".trim();
JFrame f = showForm(formName);
if(f != null){
        put(formName,f); action(9);
        //showForm 一顯示馬上執行重新整理 action(9)
}
//顯示出來的 Fram 執行查詢資料並將資料套入表格中
talk t = getTalk("nwcs");
String sql = " select  客戶編號,公司名稱,連絡人  from  客戶  ";
try{
        String ret[][] = t.queryFromPool(sql);
        setTableData("table1" , ret);
}catch(Exception e){
        message (e+"");
}
return value;
```

3.點選編譯按鈕完成後－＞點選『查詢』按鈕。

■ 字串處理

說明：

- charAt(int)：取得參數 int 的索引位置的字元。
- substring(int)：取出 int 開始取出剩下的字元字串。
- substring(int , int)：取出第一個 int 到第二個 int 間的字串。
- replace(char , char)：將第一個參數 char 取代第二個參數 char。
- concat(String)：將字串新增在字串物件之後。
- trim()：刪除字串前後的空白字元。
- equals(String)：比較字串是否相等

範例如下：

```
public class Samples
{
    // 主程式
    public static void main(String[] args)
    {
        // 字串物件宣告
        String str1=" Java 2 ";
        String str2= new String("程式設計範例教本");
        System.out.println("測試的英文字串: \"" + str1 + "\"");
        System.out.println("測試的中文字串: \"" + str2 + "\"");
        // 子字串和字元的處理
        System.out.println("英文字元 charAt(3): " + str1.charAt(3));
        System.out.println("中文字元 charAt(3): " + str2.charAt(3));
        System.out.println("英文字串 substring(2): " + str1.substring(2));
        System.out.println("中文字串 substring(2, 6): "+str2.substring(2, 6));
        System.out.println("取代英文字元 replace('a', 'b'): " + str1.replace('a','b'));
        System.out.println("空白字元 trim(): " + str1.trim());
        // 結合兩字串
        String str3 = str1.concat(str2);
        System.out.println("結合字串 str1.concat(str2): " + str3);
        If(str1.equals(“Java2”))
          System.out.println(“字串相同”);
        else
          System.out.println(“字串不相同”);
    }
}
```

執行結果：

```
測試的英文字串: " Java 2 "
測試的中文字串: "程式設計範例教本"
英文字元 charAt(3): v
中文字元 charAt(3):  計
英文字串 substring(2): ava 2
中文字串 substring(2, 6):  設計範例
取代英文字元 replace('a', 'b'):    Jbvb 2
空白字元 trim(): Java 2
結合字串 str1.concat(str2):    Java 2  程式設計範例教本
字串相同
```

■ 將字串轉成數字

說明：

- parseInt(“值”)：轉成 Integer 格式
- parseLong (“值”)：轉成 Long 格式
- parseDouble (“值”)：轉成 Double 格式
- parseFloat (“值”)：轉成 Float 格式

範例如下：

```
public class Samples7
{
    // 主程式
    public static void main(String[] args)
    {
        // 使用 valueOf()方法將字串轉換成數值
        byte   n1 = Byte.valueOf("10").byteValue();
        int    n2 = Integer.valueOf("10").intValue();
        double n3 = Double.valueOf("10.5").doubleValue();
        float  n4 = Float.valueOf("10.5").floatValue();
        long   n5 = Long.valueOf("135").longValue();
        short  n6 = Short.valueOf("135").shortValue();
        // 使用 parse???()方法將字串轉換成數值
        int    n7 = Integer.parseInt("10");
        long   n8 = Long.parseLong("135");
        double n9 = Double.parseDouble("245.678");
        float  n10 = Float.parseFloat("245.675");
        // 顯示數值
        System.out.println("byte 整數值: " + n1);
        System.out.println("short 整數值: " + n6);
        System.out.println("int 整數值: " + (n2+n7));
        System.out.println("long 整數值: " + (n5+n8));
        System.out.println("double 浮點數值: " + (n3+n9));
        System.out.println("float 浮點數值: " + (n4+n10));
    }
}
```

執行結果：

```
byte 整數值: 10
short 整數值: 135
int 整數值: 20
long 整數值: 270
double 浮點數值: 256.178
float 浮點數值: 256.175
```

■ 邏輯判斷

說明：

運算子	說明	運算子	說明
==	等於	!	NOT 運算
!=	不等於	&&	AND 運算
<	小於	||	OR 運算
>	大於	&	true & true = true
<=	小於等於	|	true | false = true
>=	大於等於	^	true ^ false = true

範例如下：

```
public class Samples4{
    public static void main(String[] args)  {
        int a = 6;
        int b = 3;
        boolean blnA = a > b;
        boolean blnB = a == b;
        // 測試關係運算子
        System.out.println("小於: 6<3  結果爲 " + (a < b));
        System.out.println("大於: 6>3  結果爲 " + (a > b));
        System.out.println("小於等於: 6<=3  結果爲 " + (a <= b));
        System.out.println("大於等於: 6>=3  結果爲 " + (a >= b));
        System.out.println("等於: 6==3  結果爲 " + (a == b));
        System.out.println("不等於: 6!=3  結果爲 " + (a != b));
        // 測試條件運算子
        System.out.println("A 條件運算式: " + blnA);
        System.out.println("B 條件運算式: " + blnB);
        System.out.println("NOT 條件運算: !A 結果爲 " + (!blnA));
        System.out.println("AND 條件運算: A && B 結果爲 " + (blnA && blnB));
        System.out.println("OR 條件運算: A || B 結果爲 " + (blnA || blnB));
        System.out.println("XOR 條件運算: A ^ B 結果爲 " + (blnA ^ blnB));
    }
}
```

執行結果：

```
小於: 6<3   結果爲  false
大於: 6>3   結果爲  true
小於等於: 6<=3  結果爲  false
大於等於: 6>=3   結果爲  true
等於: 6==3   結果爲  false
不等於: 6!=3   結果爲  true
A 條件運算式: true
B 條件運算式: false
NOT 條件運算: !A  結果爲  false
AND 條件運算: A && B  結果爲  false
OR 條件運算: A || B  結果爲  true
XOR 條件運算: A ^ B  結果爲  true
```

■ For 迴路敘述

說明：

for(初始值 ; 結束條件 ; 變數更新){

    程式敘述;

}

範例如下：

```
public class Samples5 {
    public static void main(String[] args) {
        int i;
        int total = 0;
        // 迴路敘述
        for ( i = 1; i <= 5; i++ ) {
            System.out.println("數字: " + i);
            total += i;
        }
        System.out.println("從小到大的總合: " + total);
        System.out.println(" ---------- ");
        total = 0;   // 重設總合
        for ( i = 5; i >= 1; i-- ) {
            System.out.println("數字: " + i);
            total += i;
        }
        System.out.println("從大到小的總合: " + total);
    }
}
```

執行結果：

```
數字: 1
數字: 2
數字: 3
數字: 4
數字: 5
從小到大的總合: 15
 ----------
數字: 5
數字: 4
數字: 3
數字: 2
數字: 1
從大到小的總合: 15
```

■ 二維陣列說明與運用

說明：String [][] aaa = new String[2][3]

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)

範例：

```
public class Samples6{
    public static void main(String[] args) {
        int i, j, total, sum;
        // 建立二維陣列
        int[][] scores = { { 54, 68 }, { 67, 78 }, { 89, 93 } };
        String[][] users = new String[3][];
        for ( i=0; i < users.length; i++)
            users[i] = new String[2];
        users[0][0] = "Joe";
        users[0][1] = "1234";
        users[1][0] = "Jane";
        users[1][1] = "5678";
        users[2][0] = "Tony";
        users[2][1] = "9012";
        // 使用巢狀迴路顯示陣列值
        for ( j=0; j < users.length; j++) {
            for( i=0; i < users[i].length; i++)
                System.out.print(users[j][i] + " ");
            System.out.println();
        }
        // 使用巢狀迴路計算總合
        total = 0;
        for ( j=0; j < scores.length; j++) {
            sum = 0;
            for( i=0; i < scores[j].length; i++) {
                System.out.print(scores[j][i] + " ");
                sum += scores[j][i];
                total += scores[j][i];
            }
            System.out.println();
            System.out.println("成績小計: " + sum);
        }
        System.out.println("成績總合: " + total);
    }
}
```

執行結果：

```
Joe 1234
Jane 5678
Tony 9012
54 68
成績小計: 122
67 78
成績小計: 145
89 93
成績小計: 182
成績總合: 449
```

■ Vector 應用

說明：Vector 運作很像 Array，但是它多了一個特性，當你需要將物件儲存至 Vector 中時，不需要宣告它的大小，Vector 會視內容成長。

範例：將兩陣列物件儲存至 Vector 中。

```
import java.util.*;
class Samples9{
    public static void main(String[] args){
        String[] str  = {"Adonis" , "Betty" , "David"};
        String[] str1 = {"95" , "98" , "100"};
        //宣告一各 Vector 物件 v1
        Vector v1 = new Vector();
        //將 str[]值儲存至 Vector 中
        for( int i = 0 ; i < str.length ; i++){
            v1.add(str[i]);
        }
        //直接將 str1[]物件儲存至 Vector 中
        v1.add(str1);
        for(int i = 0 ; i < v1.size() ; i++){
            System.out.println("Vector 第"+(i+1)+"筆資料  "+v1.get(i));
        }
        //取得 Vector 第四筆的資料,並將它印出
        String[] vArray = (String[])v1.get(3);
        for(int i = 0 ; i < vArray.length ; i++){
            System.out.println("vArray 陣列第"+(i+1)+"筆資料  "+vArray[i]);
        }
    }
}
```

執行結果：

```
Vector 第 1 筆資料  Adonis
Vector 第 2 筆資料  Betty
Vector 第 3 筆資料  David
Vector 第 4 筆資料  [Ljava.lang.String;@35ce36
vArray 陣列第 1 筆資料  95
vArray 陣列第 2 筆資料  98
vArray 陣列第 3 筆資料  100
```

■ HashTable

說明：將資料丟入 Hashtable 內並給每筆資料索引值，可用來判斷使用及找尋資料。

範例：

```
import java.util.*;
class Samples10{
        public static void main(String[] args){
                 Hashtable ht1 = new Hashtable();
                ht1.put("item3", "value3");
                ht1.put("item1", "value1");
                ht1.put("item4", "value4");
                ht1.put("item2", "value2");
                ht1.put("item0", "value0");

                 System.out.println(ht1.size());
                 for(int i=0; i<ht1.size(); i++) {
                   String item = "item" + i;
                   System.out.println(item + "/" + ht1.get(item));
                }
        }
}
```

執行結果：

```
Hashtable 大小: 5
item0/value0
item1/value1
item2/value2
item3/value3
item4/value4
```