

APP-Code sichtbar machen

Ein kleiner Einblick in die Möglichkeiten der BAPFRMINFO

Mit der aktualisierten **BAPFRMINFO** ist es möglich, Informationen aus dem Aufbau einer AvERP-App auszulesen und innerhalb der App anzuzeigen.

Die folgenden Tests zeigen, welche Möglichkeiten sich daraus ergeben – aber auch, wo die Grenzen der Darstellung am PC liegen.

1. Ein OnClick-Ereignis muss nicht angeklickt werden

Beim Test der APP **SCAN_LVPlan** zeigte sich eine interessante Möglichkeit:

```
text  
  
BtnAkt.Click();
```

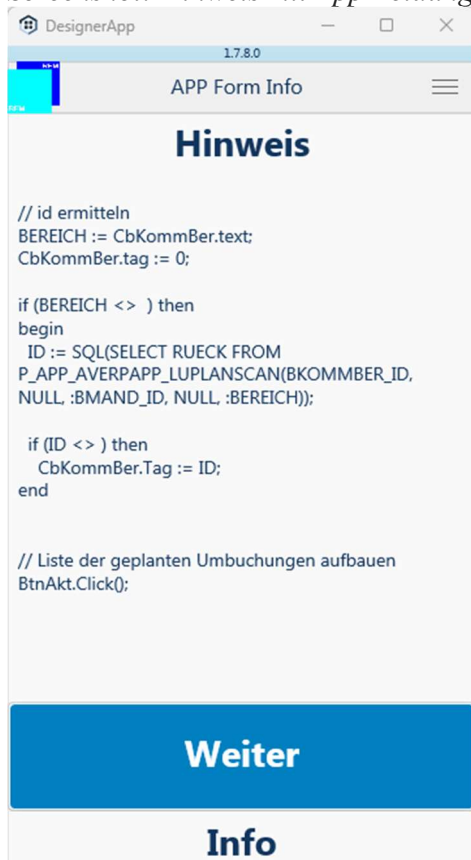
Mit diesem einfachen Befehl kann das **OnClick-Ereignis eines Buttons direkt aus dem Programmcode heraus aufgerufen werden**.

Ein tatsächlicher Mausklick auf den Button ist dafür nicht erforderlich.

Das eröffnet interessante Möglichkeiten, wenn bestimmte Aktionen innerhalb einer APP automatisch ausgelöst werden sollen.

Kurze Hinweise lassen sich dagegen sehr einfach mit dem Befehl `AppMeldung` anzeigen.

Screenshot: Hinweis mit AppMeldung



2. APP-Code direkt in der APP betrachten

Ein weiteres Experiment wurde mit der APP **Scan_LV** durchgeführt.

Beim Start der APP werden alle Objekte, für die Programmcode vorhanden ist, in einer Liste angezeigt.

In diesem Beispiel wurden insgesamt **fünf Objekte mit APP-Code** gefunden:

Nr.	Objekt	Inhalt
1	TModul	Modulaufbau aus dem AppDesigner
2	TButton	Code des ersten Buttons
3	TEdit	Code des Edit-Feldes
4	TButton	Code des zweiten Buttons
5	TListView	Code der ListView

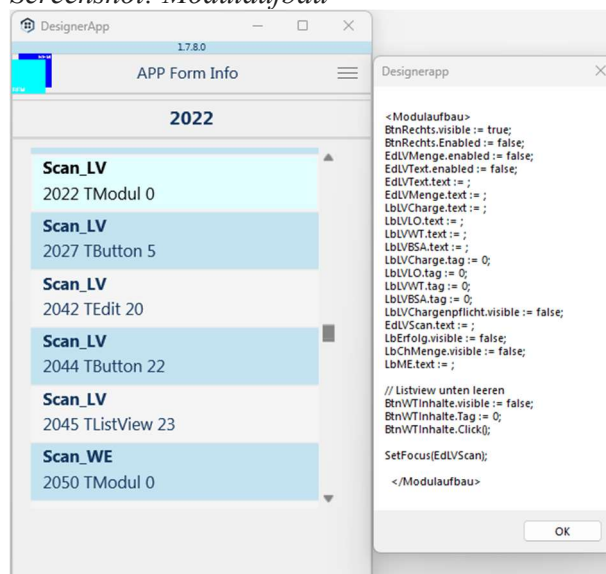
Durch Auswahl eines Listenelements kann der jeweils zugehörige APP-Code angezeigt werden.

3. Der Modulaufbau

Das erste Listenelement enthält den **Modulaufbau der APP**.

Damit lässt sich bereits ein wichtiger Teil der im AppDesigner hinterlegten Informationen direkt auslesen und anzeigen.

Screenshot: Modulaufbau

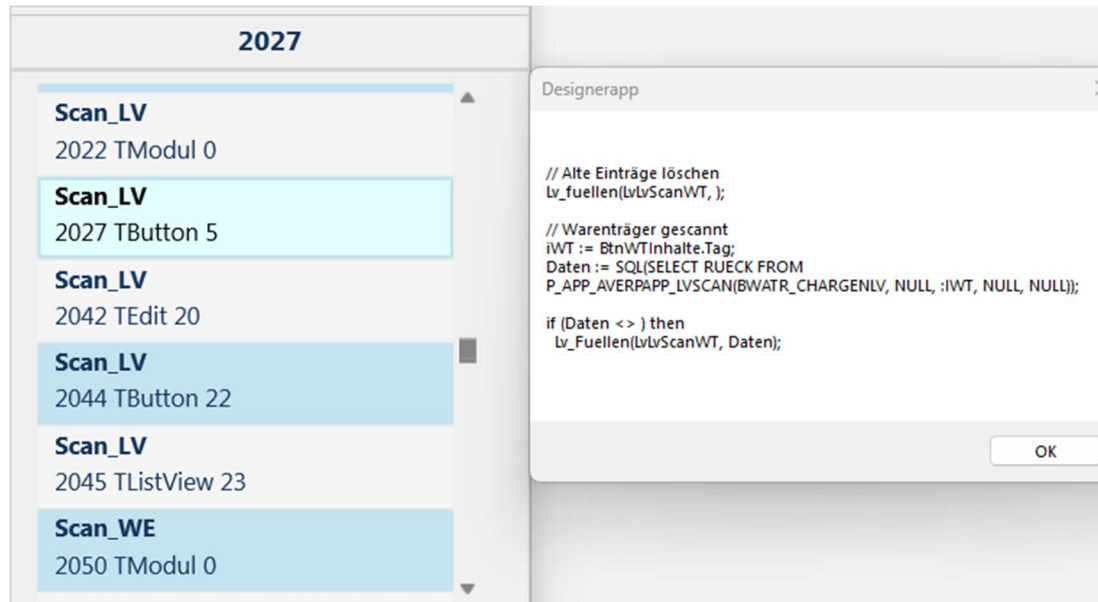


4. Der Code eines Buttons

Das zweite Listenelement zeigt den Programmcode des ersten Buttons.

Damit ist nicht nur erkennbar, **welche Objekte in einer APP vorhanden sind**, sondern auch, welcher Programmcode zu einem einzelnen Objekt gehört.

Screenshot: Code des ersten Buttons



5. Der Code eines Edit-Feldes – hier beginnt die Grenze

Das dritte Listenelement enthält den Programmcode des Edit-Feldes.

Bei diesem Beispiel wird allerdings eine Grenze der Darstellung am **PC** sichtbar:

Der angezeigte Bereich lässt sich nicht scrollen.

Bei längeren Programmtexten kann deshalb nur ein relativ kleiner Ausschnitt betrachtet werden.

Screenshot: Code des Edit-Feldes – Anzeige am PC

```
// Lager zur Chargenposition ermitteln
iLO1 := ZeileZerlegen(s, #, 3);
// Lagernummer ermitteln
sLO1 := ZeileZerlegen(s, #, 4);
end;

// Menge zur Chargenposition ermitteln
sMenge := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(CH_MENGE, NULL, :SID, NULL, NULL));

// Lagerplatz mit Warenträgerlagerplatz vorbelegen
if (iWT <> 0) then
begin
  sID := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(BWATR_BLAGERPO, NULL, :IWT, NULL, NULL));

  if (sID <> ) then
  begin
    // Lagernummer ermitteln
    sLO1 := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(BLAGERPO_MASKENKEY, NULL, :SID, NULL, NULL));

    iLO1 := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(BLAGERPO_BARTLHPO, NULL, :SID, :IART, A));
  end;
end;
end;
end;

// Lagerort
if (sTab = BARTLHPO) then
begin
  iLO1 := sID;
  s := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(LAGER_NACHSCAN, NULL, :SID, NULL, NULL));

  // Lagernummer ermitteln
  sLO1 := ZeileZerlegen(s, #, 1);
  // Artikel ermitteln
  iArt := ZeileZerlegen(s, #, 2);
  // Artikelnummer ermitteln
  sArt := ZeileZerlegen(s, #, 3);
  // Artikel chargenpflichtig?
  sTeil := ZeileZerlegen(s, #, 4);
  if (sTeil = sJa) then
    Chargenpflicht := 1;
  // Menge vorbelegen
  sMenge := ZeileZerlegen(s, #, 5);
end;

// Lagerplatz (Stammdaten)
if (sTab = BLAGERPO) then
begin
  // Lager bei Menge ändern
  if (iArt = 0) then
    MessageBox(Bitte erst den Artikel scannen)
  else
  begin
    // Wenn schon der Warenträger gescannt ist, Lager vergleichen
    if (iWT <> 0) then
    begin
      s := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(BLAGERPO_WT_PRUEF, NULL, :SID, :IWT, NULL));

      if (s = ) then
      begin
        MessageBox(Der Warenträger passt nicht zum Lager);
        sID := ;
      end;
    end;
  end;

  if (sID <> ) then
  begin
    // Lagernummer ermitteln
    sLO1 := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(BLAGERPO_MASKENKEY, NULL, :SID, NULL, NULL));

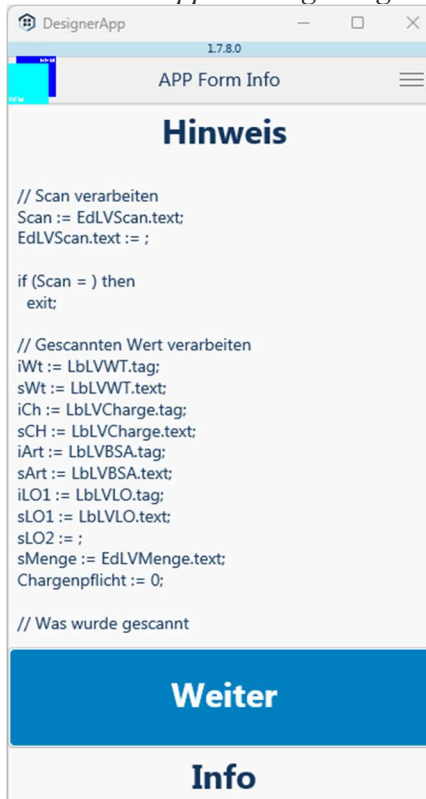
    iLO1 := SQL(SELECT RUECK FROM P_APP_AVERPAPP_LVSCAN(BLAGERPO_BARTLHPO, NULL, :SID, :IART, A));
  end;
end;
end;

// Warenträger
if (sTab = BWATR) then
begin
```

Auch die zunächst ausprobierte Variante, den Text über `AppMeldung` anzuzeigen, bringt hier keine zufriedenstellende Lösung.

Der verfügbare Ausschnitt ist dafür ebenfalls zu klein.

Screenshot: AppMeldung – begrenzte Anzeige



The screenshot shows a mobile application window titled 'DesignerApp' with version '1.7.8.0'. The interface has a top bar with 'APP Form Info' and a hamburger menu icon. Below this is a section titled 'Hinweis' (Note). The main content area displays a snippet of code, which is partially obscured by a large blue button labeled 'Weiter' (Next) at the bottom. Below the button is a section labeled 'Info'. The code visible includes comments and variable assignments in a pseudo-code language.

```
// Scan verarbeiten
Scan := EdLVScan.text;
EdLVScan.text := ;

if (Scan = ) then
    exit;

// Gescannten Wert verarbeiten
iWt := LbLVWT.tag;
sWt := LbLVWT.text;
iCh := LbLVCharge.tag;
sCH := LbLVCharge.text;
iArt := LbLVBSA.tag;
sArt := LbLVBSA.text;
iLO1 := LbLVLO.tag;
sLO1 := LbLVLO.text;
sLO2 := ;
sMenge := EdLVMenge.text;
Chargenpflicht := 0;

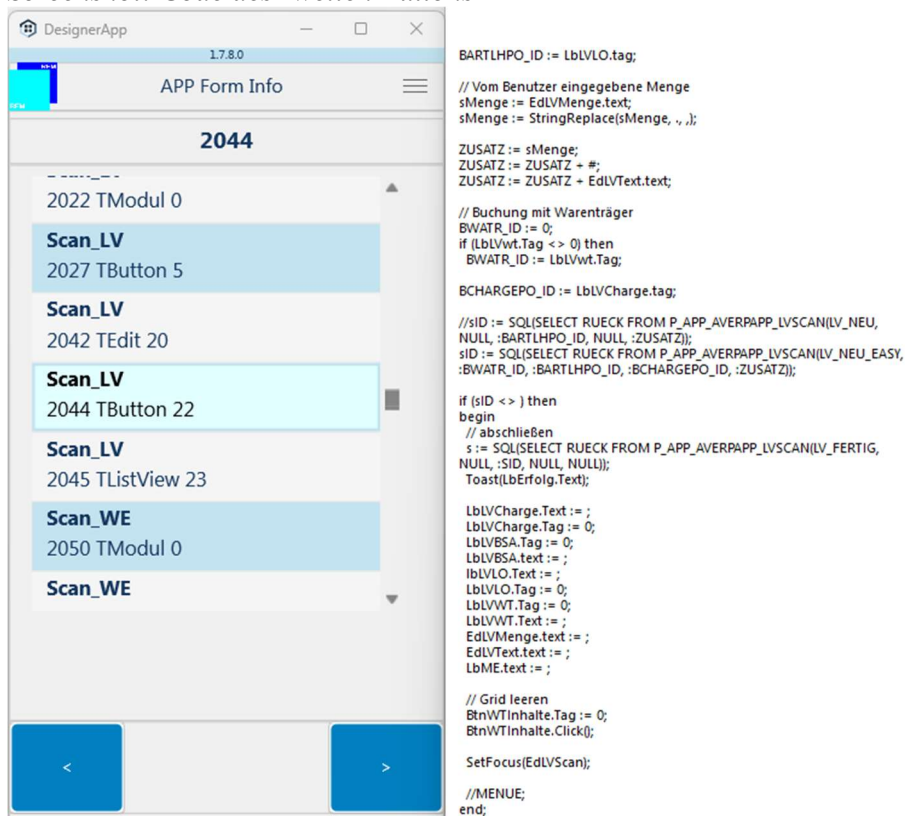
// Was wurde gescannt
```

Damit ist diese Form der Darstellung für umfangreicheren APP-Code am PC nur eingeschränkt geeignet.

6. Der zweite Button

Das vierte Listenelement zeigt den Programmcode des zweiten Buttons.

Screenshot: Code des zweiten Buttons

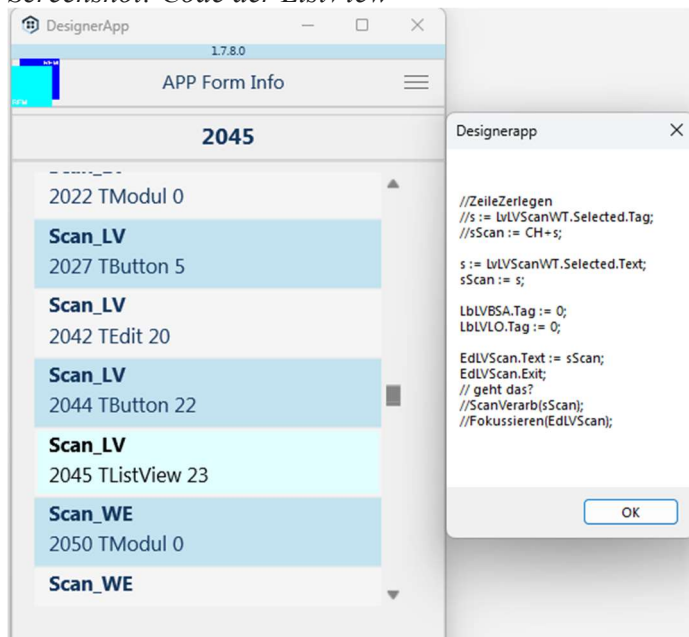


7. Der Code der ListView

Das fünfte Listenelement enthält schließlich den Programmcode der ListView selbst.

Damit kann die APP ihren eigenen Aufbau gewissermaßen **sichtbar machen**.

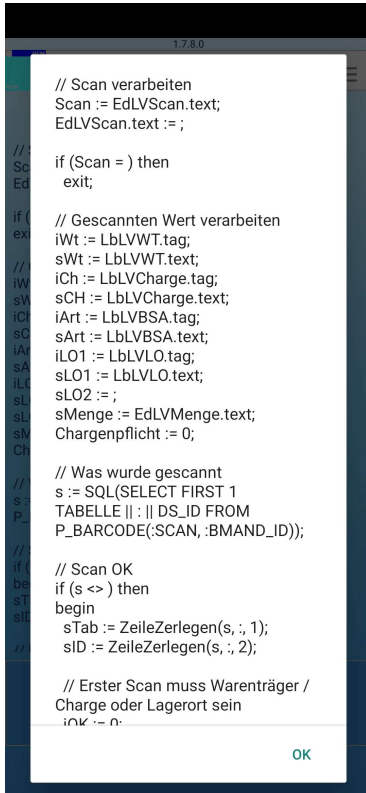
Screenshot: Code der ListView



8. PC und Mobiltelefon im Vergleich

Besonders interessant wurde der Vergleich mit der mobilen Variante der APP.

Wird **Scan_LV** auf einem **Mobiltelefon** aufgerufen, kann auch der längere Programmcode des dritten Listenelements problemlos betrachtet werden.



Der Text lässt sich dort durch **Scrollen** vollständig durchsehen und untersuchen.

Damit ergibt sich ein interessanter Unterschied:

Was am PC aufgrund der begrenzten Anzeige nicht sinnvoll dargestellt werden kann, lässt sich auf dem Mobiltelefon problemlos betrachten.

9. Fazit

Die Tests zeigen, dass sich mit der aktualisierten **BAPFRMINFO** bereits sehr weitreichende Informationen über den Aufbau einer APP auslesen lassen.

Nicht nur der Modulaufbau, sondern auch der zu einzelnen Objekten gehörende Programmcode kann ermittelt und innerhalb der APP dargestellt werden.

Die eigentliche Herausforderung liegt damit inzwischen weniger beim **Auslesen der Informationen**, sondern bei deren **geeigneter Darstellung**.

Gerade der Vergleich zwischen PC und Mobiltelefon zeigt, dass die vorhandenen Möglichkeiten stark von der jeweiligen Oberfläche und den verfügbaren Anzeigeelementen abhängen.

Damit ist ein weiterer Schritt auf dem Weg zu einer APP gelungen, die ihren eigenen Aufbau und den darin enthaltenen Programmcode sichtbar machen kann.