DataHub WebView Scripting

DataHub WebView 是一個 web-based 的資料視覺工具，具有後端的資料傳輸平台和一個能夠透過設計動畫來顯示 DataHub 的資料，並檢視這些資料的 Browser-based 編輯器，無論從任何地方，利用網際網路或企業網路，以及標準的 Web 瀏覽器就可以使用此編輯器。它可以與其他 Cogent DataHub 產品的網路功能連結，並進行即時的資料交換。

DataHub WebView1.4 的釋出包含了一個新的綜合性的 Script 功能。此功能是由 S#(Script.NET)，具有揭露和擴展應用程式的模式，並且能夠與內部的.NET 交換物件、類型和組件互動。

DataHub 的 WebView 內有幾個可以被巨集化的領域。概括地說明，它們為：

Script Bindings	能夠連結至 Script 結果的任何屬性的控制元件。
Dynamic Point Bindings	這些是由 Script 表示式計算的 DataHub 點名稱。
Events and Event Handlers	事件處理可以被加在三種類型的事件：控制元件、頁面和應用程式。

以附加的 WebView Script，你可以利用三個不同的 Script 領域來協同 Cogent DataHub 運作：

- DataHub 中執行的 Gamma
- 瀏覽器中執行的 Javascript
- 在 DataHub WebView 中執行的 S#

Scripting 能力

DataHub WebView 1.4 的發行，包括一個新的綜合性 Scripting 功能，能夠使網頁設計師和使用者在 DataHub WebView 中自行客製化動作以得到令人滿意的使用者體驗。

有了這個新功能，您可以：

- 在執行模式下以使用者選擇的結果來設定動態點連結（dynamic point bindings)。
- 以使用者名稱或是其他的範圍資料爲資料來源連結，來設計模板頁面(templated pages)。
- 構建多頁的解決方案，可與頁面資料和全域變數交互動作。
- 新增動畫效果以模擬流程及增強告警通知。

範圍和環境

變數的範圍和 Script 環境表示式是每一個編輯程式環境的重要課題。 S#也不例外。

Variable Scope	變數的範圍，有時也被稱為作為一個變數的可存取性，是指其中的變數可以讀取和/或寫入，和變數的壽命，或它能夠在記憶體內保留多久。對不同的程式語言/Script 語言和工具而言，變數宣告語法和範圍規則是不同的。
Script Context	Script 環境是 Script 執行時，存儲訊息的一部分，如：範圍、範圍內的變數、旗標、函數等。在 DataHub WebView 中的變數，環境取決於 Script 是如何被關聯的。環境通常是一個：頁面，控制元件或事件處理程序。

變數範圍

所有 S#變數都是無類型的。他們可以接受布林值、數字、字串和任何其它資料類型。變數存儲在 Script 環境內，這意味著它們和它們宣告的 Script 具有相同的範圍，。

區域範圍 – 區域變數

執行 Script 時，它執行在自己私有的變數範圍。私有範圍內宣告的區域變數在 Script 完成後才算有效。最好的做法是使用關鍵字 var 來宣告變數，但它不是絕對必要的。例如，變數 x 和 y 都是在 Script 片段中被宣告的環境內。測試功能具有它自己的環境，宣告的變數 z 被限制在函數的範圍。

```
x = 5;
var y = 20;
var result = x * y;
DEBUG("The result is " + result);

function test() {
  var z = 10;
}

DEBUG(z);  /* generates error: 'Identifier not found: Namespace 'z' is not found' */
```

由於函數有它們自己的環境，他們只能存取其內宣告的變數。例如，test2 的函數沒有存取變數 msg 的資訊。

```
var msg = "Hello World!";

function test2() {
  DEBUG(msg);  /* generates error: 'Identifier not found: Namespace 'msg' is not found' */
}
```

區域範圍 - 內部物件

除了區域變數，DataHub WebView 中也有一些特殊的變數，稱為內部物件，它提供方便地存取所使用的 Script 連結和事件處理程序的共用物件。例如，以下程式碼可用於控制元件事件處理程序：

```
var thisPageName = Page.PageName;
var thisElement = Element;
var thisEvent = EventName;
```

thisPageName、thisElement，還有 thisEvent 都是區域變數。Page、Element 和 EventName 是內部物件。內部物件僅在私有範圍內是有效的。內部物件的可用性取決於 Script 環境。

全域範圍

除了私有的範圍外，所有 Script 都能夠存取全域範圍內的資源。全域範圍是讓所有 Script 和所有 DataHub WebView 頁面在 WebView session 的持續時間內共享的。重設全域範圍的唯一方法是重新啟動 DataHub WebView。若使用在 Script 中的變數，沒有明確使用 var 這個關鍵字宣告，則該變數屬於全域範圍之內。在全域範圍內分配一個值給全域變數的簡單動作，就可讓該變數存在。

函數可以透過全域關鍵字來存取全域變數。

資料點

資料點被存儲在一個分隔開的命名空間，且不被視爲 Script 變數。也就是說，資料點不能做為內部引用。相反的，資料點可使用 VAL、GET 和 SET 函數來引用。

動態的全域變數

動態的全域變數被存儲在一個分隔開的命名空間，如同資料點。他們也使用 VAL、GET 和 SET 函數來引用。

元素屬性

控制元件的屬性也不被視爲 Script 變數。像資料點一樣，他們可以使用特殊的存取函數，GETP 和 SETP 來引用。

Script 環境

Script 執行時，它有對於一些特殊變數的存取，取決於它執行的環境。這些特殊的變數被稱為內部物件。此外，所有的 Script 可存取在全域範圍內、動態的全域變數和資料點（通過存取函數）。環境可以是：

- DataHub WebView 頁面（頁面環境）
- 頁面上的控制元件（控制元件環境）
- 頁面上控制元件的事件處理程序（控制元件事件環境）

頁面環境

網頁事件處理程序執行在頁面環境，可存取 Page 和 EventName 內部物件。

控制元件環境

當 script 是一個點名稱表示式、script 連結，或一個事件處理程序時，script 與控制元件有著連結。當 script 被執行時，控制元件本身是可使用 Element 存取的內部物件。此物件引用承載控制元件視覺化呈現的容器。

註：在本文中，術語“控制元件(control)”和“元素(element)”是同義詞，交替使用。

Element 物件公開一些屬性和方法可以用來操縱控制元件的屬性。

所有的控制中有一個特殊的訊息存儲，稱為 ElementData。這是一個特殊的 expando 物件，它的成員變數是在它們被存取時動態產生的。這使 script 編寫者能夠簡單、自然地新增新項目至 ElementData。例如，此 script 分配值至 ElementData 的 LastInput 屬性：

```
ElementData.LastInput = the_value;
```

之後，ElementData.LastInput 在同一 Script 環境內所有執行的 Script 中都為有效（Script 與同一控制元件關聯）。每個控制元件都有其自己的 ElementData，這是只適用於與該控制元件連結的 Script。任意數量的成員都可被加入 ElementData。

事件處理程序環境

有些事件處理程序可存取額外的內部物件。

範例

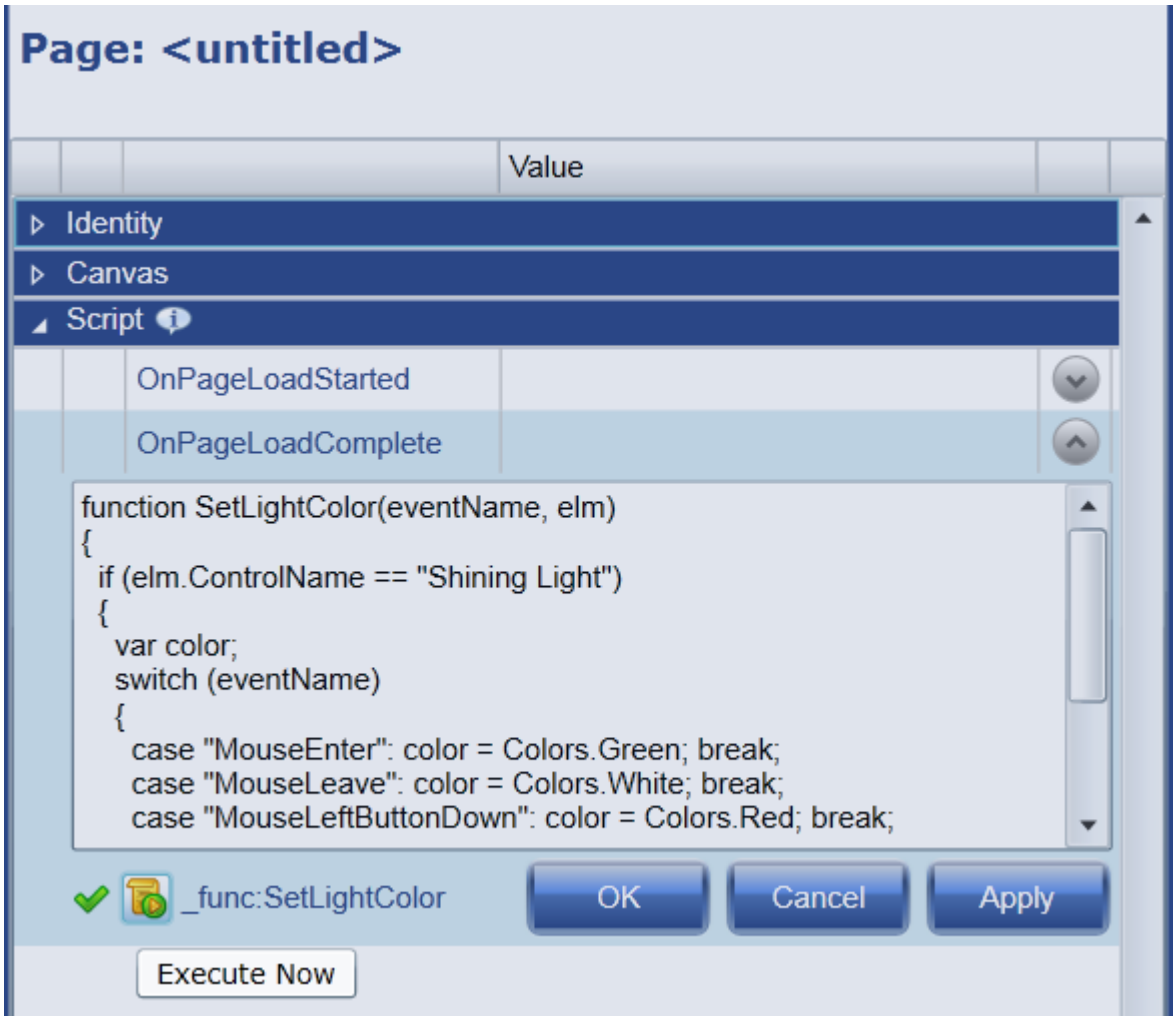
這個例子說明如何在頁面載入時宣告一個 Script 函數，然後從各種控制元件事件處理程序中呼叫該函數。每個事件處理程序呼叫函數，傳遞 EventName 和引用給 Element，就是擁有事件處理程序 Script 的控制元件。此函數依據觸發呼叫的事件名稱來設定控制元件的顏色屬性。

第 1 步：宣告函數

啟動一個新的 WebView 頁面，並新增下面 OnPageLoadComplete 事件處理程序碼。

```
function SetLightColor(eventName, elm)
 {
   if (elm.ControlName == "Shining Light")
   {
     var color;
     switch (eventName)
     {
       case "MouseEnter": color = Colors.Green; break;
       case "MouseLeave": color = Colors.White; break;
       case "MouseLeftButtonDown": color = Colors.Red; break;
       case "MouseLeftButtonUp": color = Colors.Green; break;
     }
     SETP(elm.PageElementName + "@PrimaryLightColor", color);
   }
 }
```

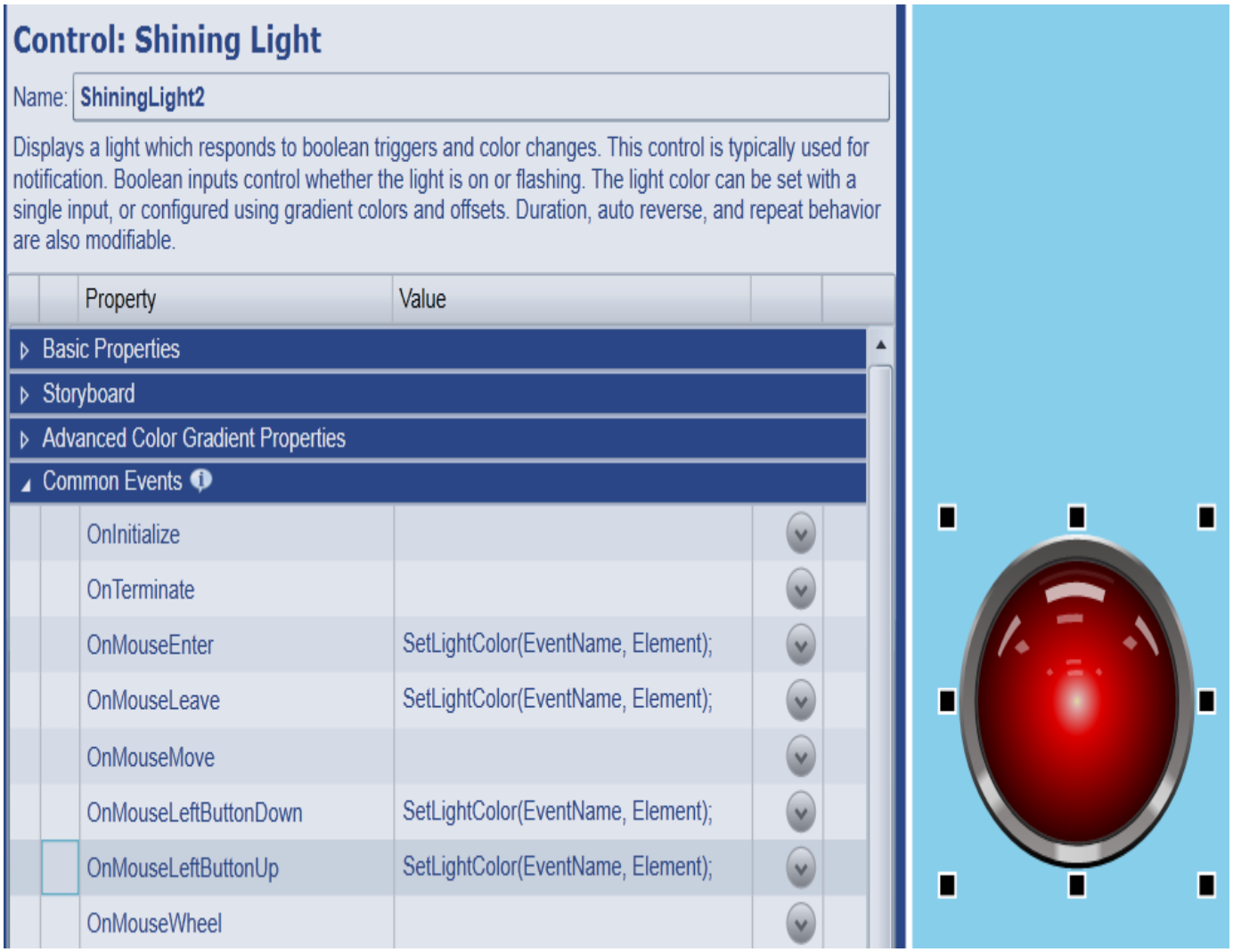
當頁面載入和 Script 被執行時，SetLightColor 函數被註冊。可以點擊“立即執行”按鈕來立即註冊函數（如下圖所示）。



第 2 步：新增事件處理程序

新增一個 Shining Light 控制元件實體到頁面上，並新增四個滑鼠事件到以下程式碼：OnMouseEnter、OnMouseLeave、OnMouseLeftButtonDown 和 OnMouseLeftButtonUp。

```
SetLightColor(EventName, Element);
```



步驟 3：執行模式下觸發事件

進入執行模式。當你移動滑鼠到燈號上，點擊滑鼠左鍵然後釋放，燈號就會改變顏色。只有燈號接收到事件會改變顏色，因為一個 Element 的引用被傳遞到 SetLightColor 函數。

連結

任何控制元件的屬性都可以被連結。連結類型有四種：

類型	說明
None	屬性沒有連接。
Point	屬性連結到一個資料點。
Simple	屬性連結到另一個頁面上其他控制元件的另一個屬性。
Script	屬性連結到一個 Script 的結果。

S#的 Script 表示式可以被用來建立：

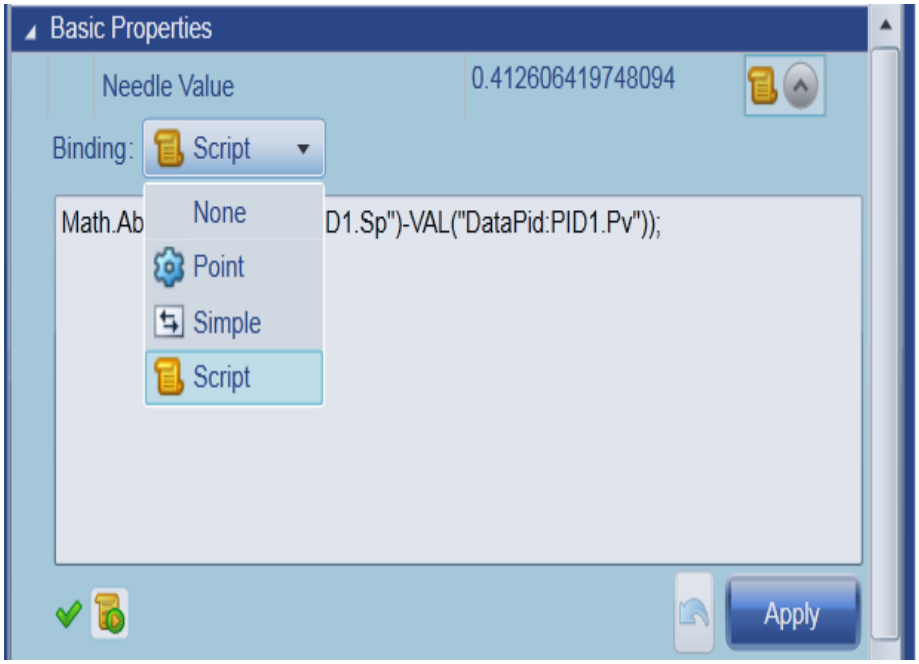
- Script Binding – 其 script 的結果為控制元件屬性所用。
- Dynamic Point Binding - Script 的結果為動態設定點名稱所用。

註：簡單的連結不支援 Script。

Script 連結

一個 Script 連結可以包含任意數量的表示式。屬性的值將會是 Script 最後一行執行的結果。

連結類型可以在設計模式中交互地選擇使用屬性瀏覽，如下圖所示。



連結類型也可以在 script 中透過 IPageElementParameter.SetBindingTypemethod 方法，以程式語言編輯的方式設定。在 script 中設定連結類型時，你必須使用 BindingType 列舉，如下面的程式碼。

```
var p = Page.GetParameter("myGauge@NeedleValue");
if (p != null)
{
  var expr =
"Math.Abs(VAL(\"DataPid:PID1.Sp\")-VAL(\"DataPid:PID1.Pv\"));";
  p.SetBindingExpression(BindingType.Script, expr);
  p.SetBindingType(BindingType.Script);
}
```

動態點連結

動態點連結提供了強大的能力來建立：

- 在執行模式中，讓使用者能夠選擇資料點連結的頁面。
- 設備或程序的呈現類似陣列的模板頁面。

動態點連結讓您能夠建立一個單一的 DataHub WebView 頁面，取決於控制元件狀態、其他的 Script、使用者的輸入或頁面啟起始參數來顯示不同的資料。

點連結是以“=”表示式開始，後面接着回傳資料點名稱的 Script 表示式，動態地被生成。

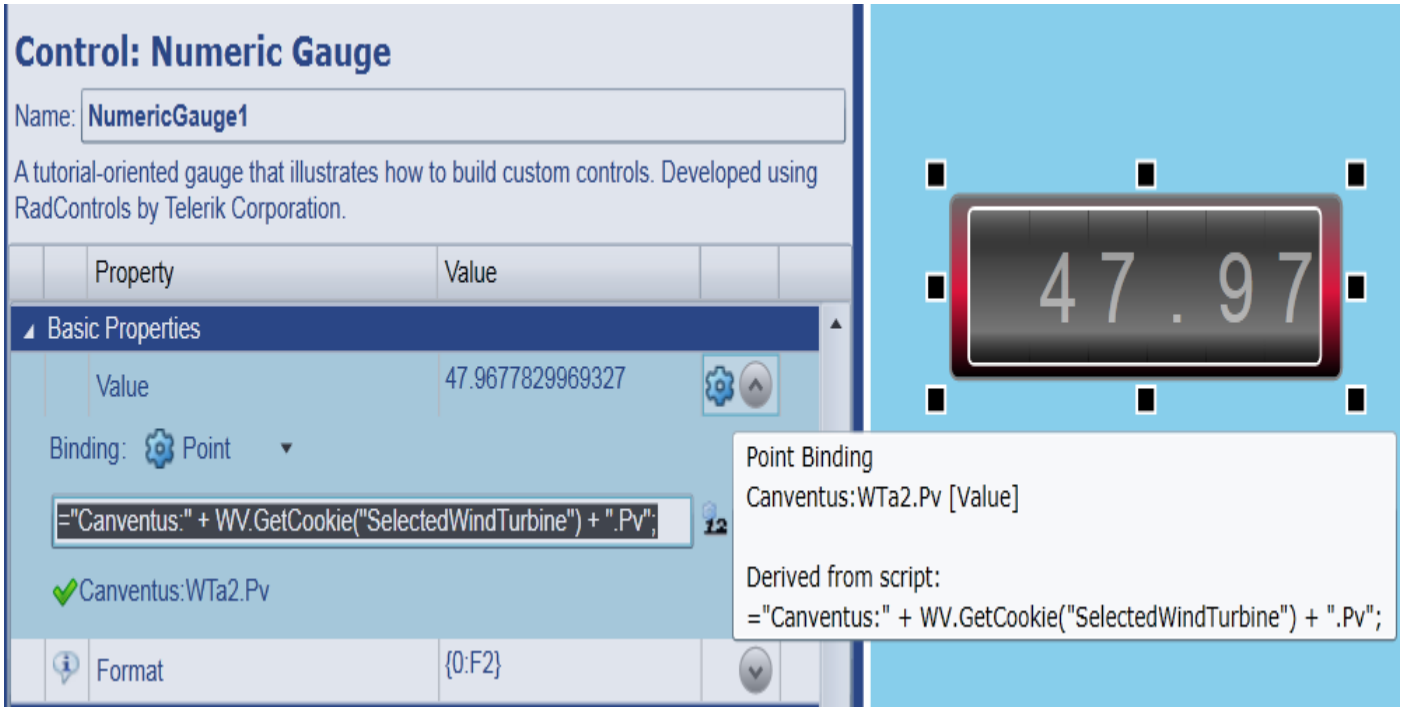
點名稱表示式的結果必須是一個字串，其中包含完全符合規定的點名稱，以資料 domain 和一個冒號爲起始。點名稱的表示式是自動執行，所以動態全域變數、資料點和控制元件屬性的引用，將在引用值變化是導致點名稱被重新計算。

範例

這個例子說明了如何建立一個使用 Script 表示式來確定資料點名稱的點連結，。

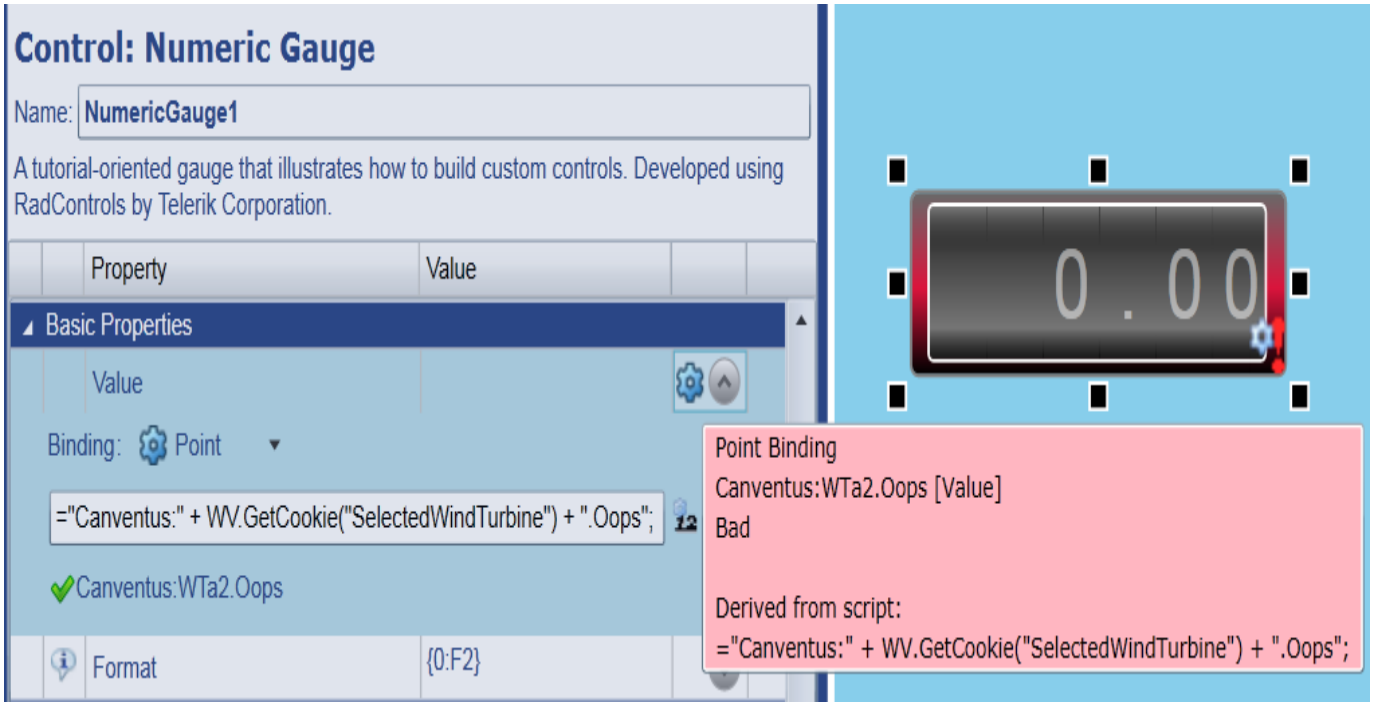
```
="Canventus:" + WV.GetCookie("SelectedWindTurbine") + ".Pv";
```

此 Scriptc 藉由連結一個表示資料 domain(“Canventus：”)的字串，與一個 Script cookie 的值，和點名稱(“.PV”)，來回傳一個點名稱。當前的 cookie 值是“WTa2”，所以數字壓力表被連結到 Canventus：WTa2.Pv.



使用 Script Cookies 視窗可以對 script cookies 做修改，可以透過選擇 Tools | Script Cookies 來顯示該視窗。

如果 Script 的表示式不回傳一個可識別的點名稱，或者如果點的值連線品質不好，控制系統將顯示一個警告圖標（取決於控制元件的 BadDataAdornmentType 屬性）。在設計模式中，您可以透過停留在連結類型的圖標，排除故障點連結問題，如下圖所示。



在執行模式選擇點

動態點連結表示式可以引用全域變數、Element 屬性和其他資料點。

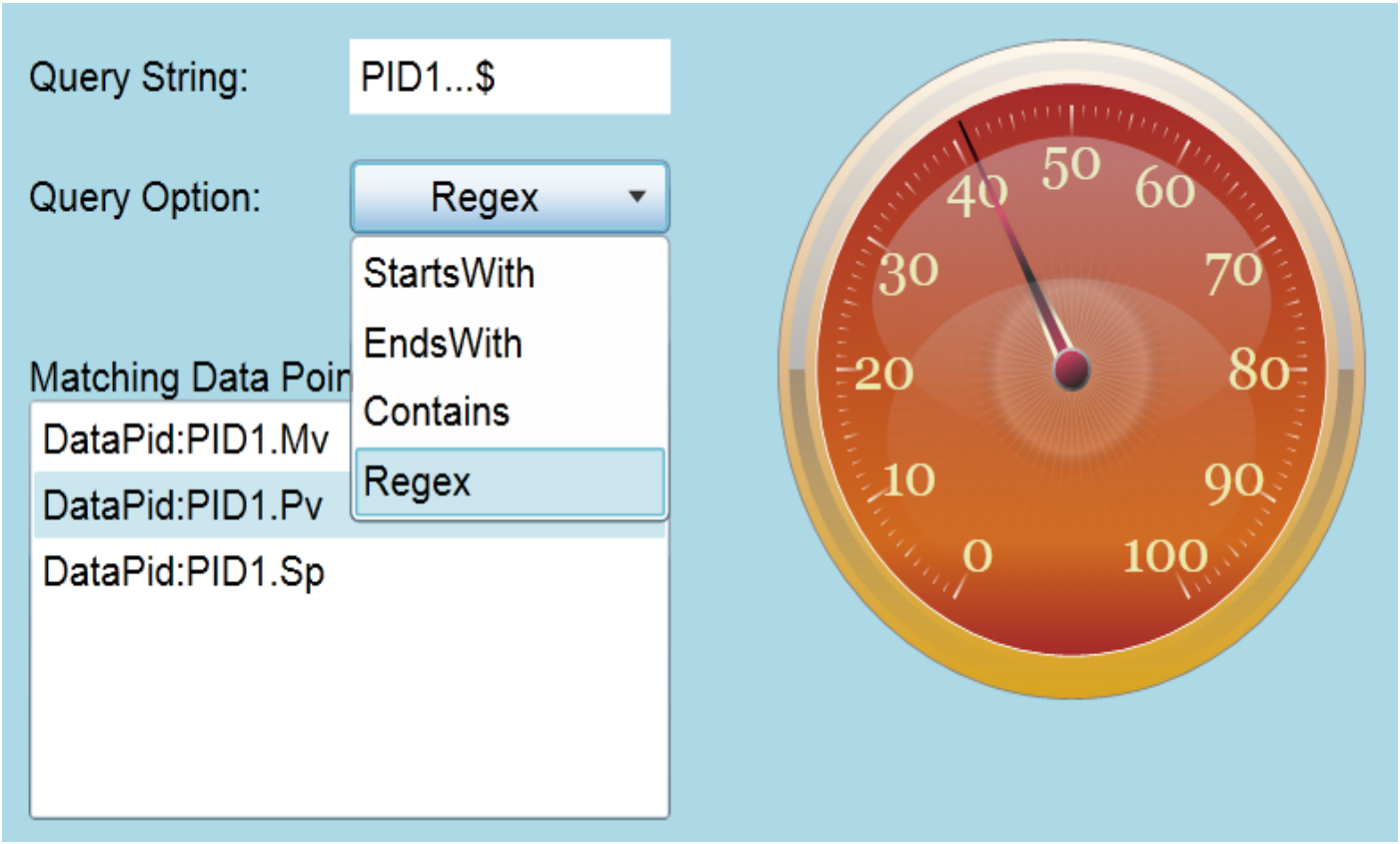
例如，你可以建立一個頁面，代表 PLC 的配置盤，然後使用點名稱的表示式填入配置盤作為數值。頁面上的其他地方，您可以提供一個選擇控制元件，來選擇您的製造程序中的別組 PLC。當你選取 PLC，面板將變更資料點和它的顯示，允許使用者快速地在 PLC 之間做切換而不用更換頁面。

範例

這個例子說明了如何建立一個資料點的選擇機制，讓使用者選擇執行模式中的資料點，並連結一個儀表。

第 1 步：頁面設計

此頁面包含一個 Text Entry element（txtQueryString），一個 Combo Box（cbQueryOption），一個 List Box（lstMatchingPointNames），和一個 Simple Radial Gauge（gauge）。



第 2 步：Script 連結

Combo Box 項目物件來源屬性是一個 Script 連結從 PointQueryOptions 列舉回傳的名稱列表。

```
WV.GetEnumNames(PointQueryOptions);
```

List Box 項目來源屬性是一個 Script 連結使用 GETP 來取得使用者輸入的查詢字串和選定的查詢選項，然後呼叫 GetPointNames 來回傳一個符合條件的資料點清單。

```
var query = GETP("txtQueryString@Value");
```

```
var option = GETP("cbQueryOption@SelectedItem");
WV.GetPointNames(query, Enum.Parse(PointQueryOptions, option, true));
```

第 3 步：動態點連結

儀表的指針數值的屬性使用了點連結，從符合條件的資料點的 List Box 中，動態地引用配對到的物件。

```
=GETP("lstMatchingPointNames@SelectedValue");
```

當使用者輸入一個查詢字串，並選擇查詢選項，符合條件的資料點將會出現在清單中。在清單中選擇一個資料點，將觸發儀表指針數的點數值連結表示式重計算，然後開始將所選定的資料點反映在儀表的數值上。

事件和事件處理程序

事件處理程式是當某個事件發生時所要執行的 Script。例如，當游標在一個控制元件上，使用者按下滑鼠按鈕使事件發生的時候。

有些事件處理程序需要特定事件的訊息。例如，控制元件的 OnMouseMove 事件可能需要知道滑鼠的位置。DataHub WebView 中使用的一般事件是以 Microsoft Silverlight 為基礎，但使用不同的事件處理程序機制。對於大多數的事件，都是 DataHub WebView 內展現的內部物件，而不需事件處理程序來要求的函數參數。

Type 類型	Event 參數
Element Events	透過內部物件來進行存取：Element、ElementData、Me、EventName 以及 EventArgs。
Page Events	透過內部物件來進行存取：Page 以及 EventName。
Application Events	作為函數參數來傳遞。應用程式的事件處理程序是被編碼為完整的 Script 函數（通常是.ss 檔）。Application_UserChanged 事件處理程序必須進行以使用者的名稱編碼，以作為函數參數。內部物件則是無效的。

Element 事件

所有控制元件都支援幾種常見的事件處理程序。一些控制元件還支援特定控制元件的事件處理程序（稱為“客製化事件”）。

例如，所有的控制元件支援按下，然後鬆開滑鼠左鍵(OnMouseLeftButtonDown)的事件。一個按鈕控制元件還支援一個 OnClick 事件，而一個 checkbox，支援 OnChecked 和 OnUnchecked 的事件。

Event handlers common to all controls include:

所有控制元件常用的事件處理程序包括：

事件名稱	何時發生
OnInitialize	此 Element 被新增到頁面上(在主頁面被載入時也可以新增)。您可以使用這個事件新增自行定義的初始化作業。
OnTerminate	這個元素在頁面上被刪除（在主頁面被卸載時也可以移除）。您可以使用這個事件新增自行定義的清理作業。
OnMouseEnter	滑鼠指標進入此元素的邊界。
OnMouseLeave	滑鼠指標離開該元素的邊界。
OnMouseMove	滑鼠指標在這個元素上移動。
OnMouseLeftButtonDown	當滑鼠游標在這個元素上，同時按下滑鼠左鍵。
OnMouseLeftButtonUp	當滑鼠游標在這個元素上，同時放開滑鼠左鍵。
OnMouseWheel	當滑鼠游標在這個元素上，同時轉動滑鼠滾輪。

Element 事件處理程序存取內部物件：Element、ElementData、Me、EventName 以及 EventArgs。

可以使用屬性瀏覽器來編輯 Element 的事件處理程序。

頁面事件

有兩頁的事件在載入頁面時被啟動。您可以為任何一個事件的處理程序新增 Script。

事件名稱	何時發生
OnPageLoadStarted	頁面開始被載入，且立即的，在頁面上的控制元件被載入之前（如，預載的 Script）。在此時，它對頁面上引用的元素是不安全的。
OnPageLoadComplete	所有其他頁面設定完成（如，後載 Script）。在此時，頁面元素可以安全地引用，且初始化或操縱。

使用這些 Script 事件處理程序來建立自行定義的 Script 函數，和初始化全域變數是很常見的。

頁面事件處理程序可存取內部物件：Page 以及 EventName。

頁面事件處理程序可以使用屬性瀏覽器來編輯。

應用程式事件

應用程式事件的觸發是獨立於頁面的載入和全域環境。

事件名稱	何時發生
Application_Startup	應用程式啟動。
Application_UserChanged	當應用程式為了特定的使用者而重新啟動。該使用者的名稱將會作為函數參數，並傳遞給事件處理程式。

使用 Application_UserChanged 事件處理程序來載入使用者別的起始頁面是很常見的。Application_UserChanged 事件處理程序必須進行以使用者的名聲編碼，以作為函數參數。內部物件則是無效的。

應用程式事件處理程序通常會被建立及管理在應用程式之外的應用程式，並自動由 DataHub WebView 載入.SS 檔案，並存在於 DataHub server 中。

應用程式事件處理程序不使用屬性瀏覽器做編輯。

函數和特殊物件

爲強化對 DataHub WebView Scripting 的瞭解，有必要去熟悉每個主題：

Working with Data Points	取得/設定資料點的函數。
Working with Element Properties	取得/設定元素屬性的函數。
WV Functions	存取應用程式的命令和行爲的函數。
Intrinsic Objects	允許事件處理程序去引用和操縱相關的 Page，Element 和 EventArgs 物件。

資料點運作

每個存儲在 Cogent DataHub 中的值在被稱為一個點。 點具有以下屬性：

- Name - 一個字串。目前唯一的上限是內部緩衝區的大小，預設情況下，約 1000 個字元。
- Value - 一個整數、浮點數，或字串。
- Timestamp - 最不顯著變化的點數值、連線品質或其他狀態訊息的日期和時間。
- Quality – 連接品質，由 Cogent DataHub 分派給此點，如 Good、Bad、Last known、Local override 等

DataHub WebView 支援區域和 server 端的資料點。區域資料點是由 WebView client 端進行管理，但不能持續保持。Server 端的資料點是由 Cogent DataHub 管理。

大多數情況下，您將使用點的值。不過，您也可以使用點物件來存取其他屬性。要使用點作為物件，你可以使用 IVqt 屬性。

您可以使用下面的 Script 函數來運用資料點：

VAL	取得指定點的值。
GET	取得指定的點以作為物件。支援 IVqt 和 IDataTable 介面。
SET	設定指定點的值。
WV.GetPointNames	使用指定的標準，來取得一個點名稱陣列。
WV.GetPoint	取得指定的點以作為物件。支援 IVqt 介面。

元素屬性運作

在設計模式中，您可以在您的頁面新增元素（或控制元件）。每個元素都有幾個屬性。大多數情況下，您可以使用屬性瀏覽器來配置元素。不過，你也可以在 Script 中使用以下函數來操縱屬性：

GETP	取得引用屬性的值。
SETP	設定引用屬性為指定的值。
IPageElementParameter.Value Property	透過 IPageElementParameter 界面成員，直接使用屬性物件來取得/設定屬性的值。

WV 函數

WV 函數是一組為所有 Script 提供資源查詢的方法。

例如，可以使用 WV.GetPoint 從 DataHub 內取得一個點，而 WV.MsgBox 可用來在一個對話視窗中顯示點的屬性。

所有 WV 方法是靜態的，這意味著並不需要建立 WV 物件的實體。相反的，你只需簡單地通過直接使用 WV 類別來引用方法，。

WV 函數在整個本文內被廣泛使用在實例中。

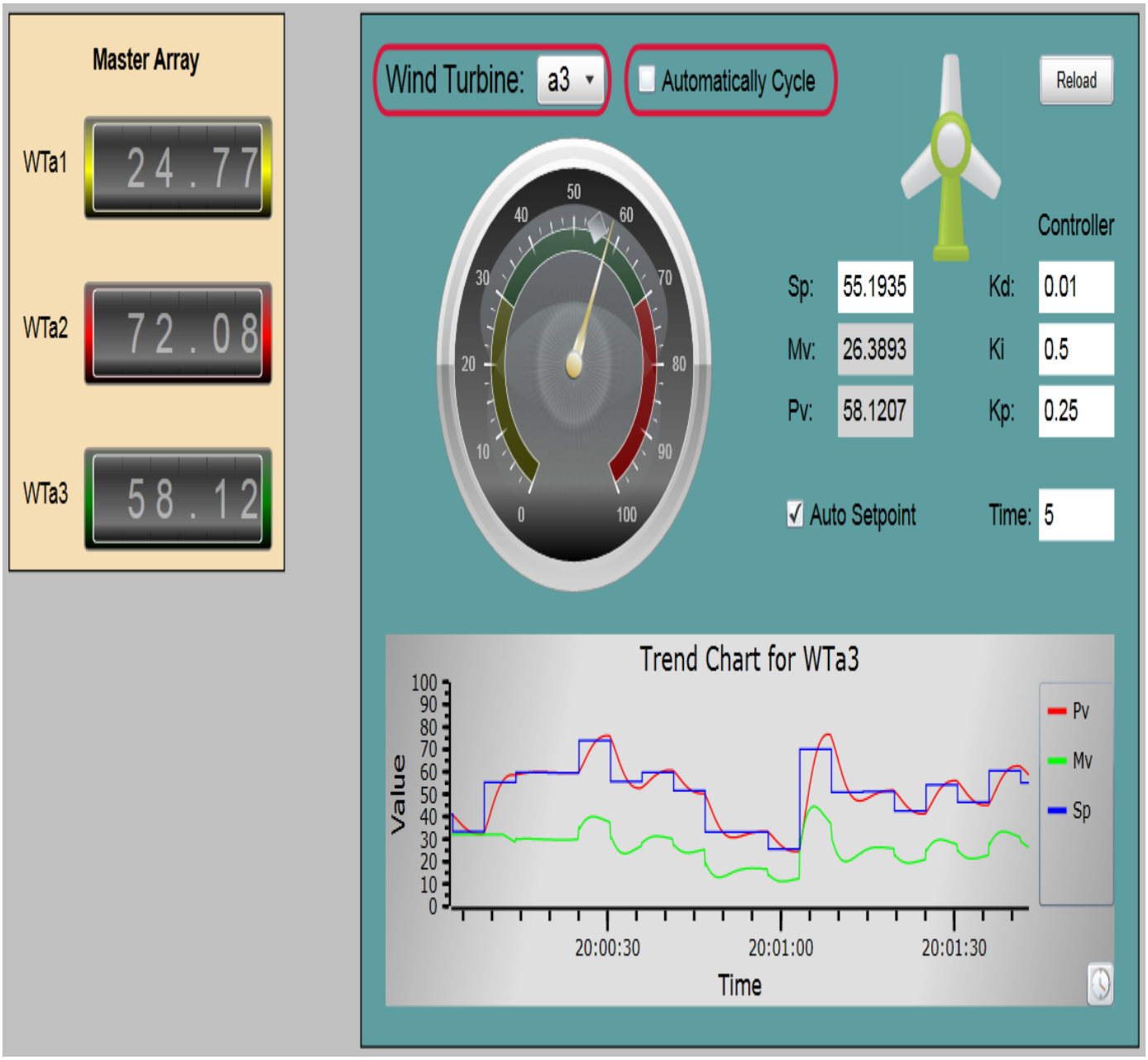
內部物件

內部物件使用各種 Script 連結和事件處理程序，提供存取通用物件的便利性。

例如，引用 Page 物件能夠提供存取像是 PageName和 andPageDescription 等屬性，以及像是 GetNamedElement 和 GetParameter 等方法。

範例

本實例說明一個從陣列中顯示三個風力渦輪機的訊息。儀表、趨勢圖和其他顯示元件反映目前選定的渦輪機。使用者可以從”Wind Turbine”的 combo box 中選擇一個渦輪機，或在清單中可以選擇”Automatically Cycle”。



爲讓使用者的選擇能夠持續，已新增下面的 Script 事件處理程序。

Wind Turbine combobox ('wtSelector')**OnDropDownClosed**：設定一個 cookie 來代表選定的風力渦輪機。

```
WV.SetCookie("SelectedWindTurbine", GETP("@SelectedIndex"));
```

Automatically Cycle checkbox ('chkAutoCycle')OnChecked：設定一個 cookie 來表示 auto-cycle 已被選擇。

```
WV.SetCookie("AutoCycleSelectedWindTurbine", 1);
```

Automatically Cycle checkbox ('chkAutoCycle')OnUnchecked：設定一個 cookie 代表自動循環不被選擇。

```
WV.SetCookie("AutoCycleSelectedWindTurbine", 0);
```

OnPageLoadStarted：宣告一個 Script 函數，以及為控制元件儲存 Cookie 的值。

```
function ApplyCookies()
{
  var wt = WV.GetCookie("SelectedWindTurbine");
  var auto = WV.GetCookie("AutoCycleSelectedWindTurbine");

  if (wt != null)
    SETP("wtSelector@SelectedIndex", wt);

  if (auto != null)
    SETP("chkAutoCycle@Value", auto);
}
```

OnPageLoadComplete：呼叫註冊的 Script 函數。

```
ApplyCookies();
```

Reload button ('btnReloadPage')OnClick：重整當前頁面。

```
WV.ExecuteCommand("OpenPage", Page.PageFullPath);
```

在執行模式下，每當使用者點擊 Automatically Cycle 的 checkbox，或從 Wind Turbine 的 combobox 進行選擇， Script Cookie 會被設定和儲存。

當使用者點擊刷新按鈕時，會重新開啟頁面，並保持 cookie 值的取得和套用。即使使用者關閉並重新啟動 DataHub WebView，Script Cookie 依然會持續保持。

最佳實作方法

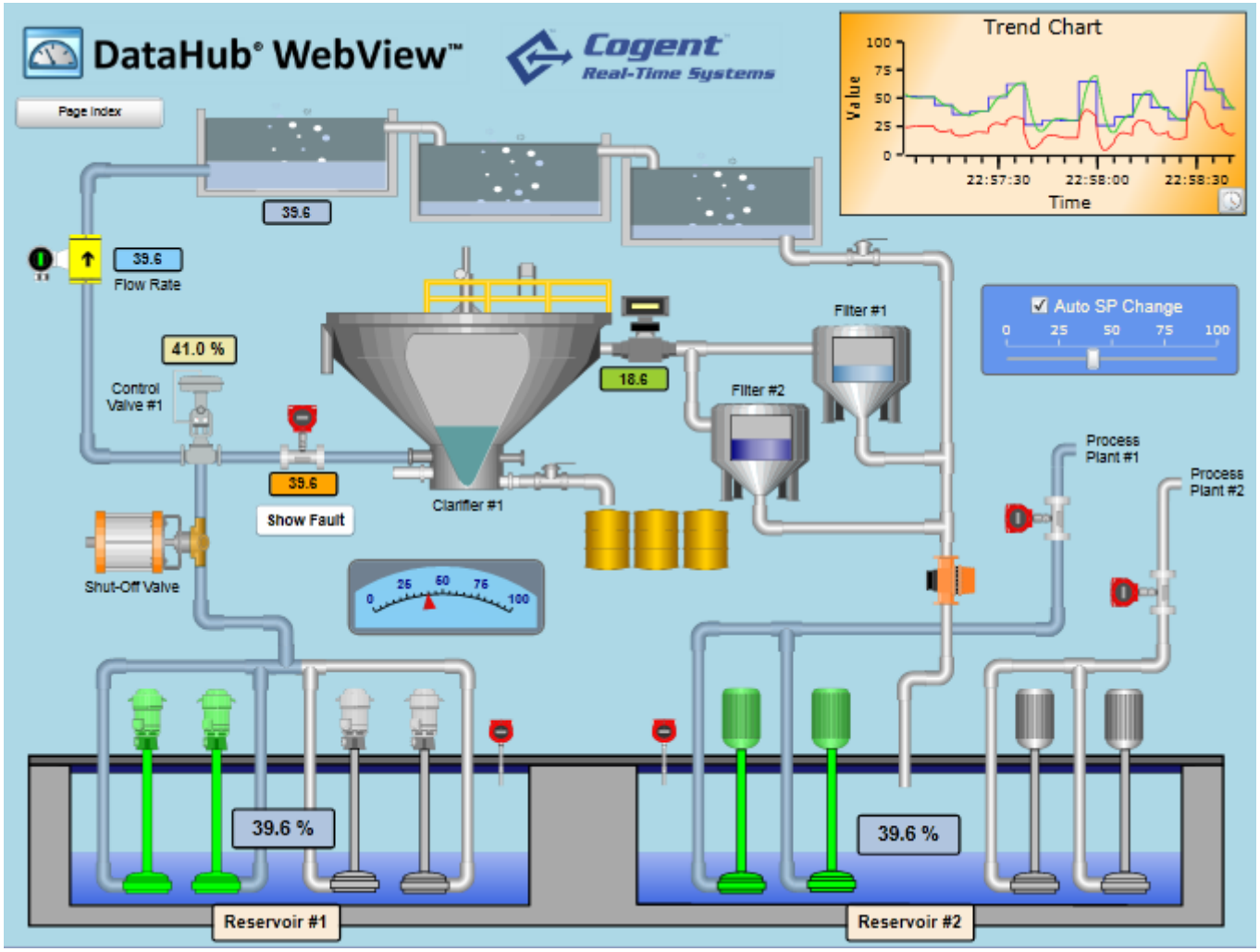
採用最佳實作方法，降低了維護需要和簡化故障排除。當您建立 DataHub WebView 解決方案，我們建議您考慮採取以下做法。

定義有意義的控制元件名稱

當控制元件在頁面上被加入，它會被指定一個以其類型和後綴整數組成的預設名稱。例如，第一次在頁面上新增”SimpleButton”，它會命名為”SimpleButton1” - 除非頁面上有個控制元件已經有這個名字了，在這種情況下，計數器會遞增，直到找到一個可用的名稱。

很顯然，這樣的頁面，如下圖所示，這樣的 Script 變得很難理解，很難檢查與排除問題。

```
if (GETP("Symbol23@Condition") > GETP("Symbol72@Condition"))
  SETP("TextLabel2@Input", "Overflow");
```



控制元件名稱使用在 Script 表示式和簡單的連結表示式中。給予頁面上剛新增的控制元件適當的、有意義的名字是個很好的主意。

使用正確的變數環境

使用 var 宣告並初始化變數。這也適用於 Script 連結、事件處理程序，以及 Script 函數。沒有 var 宣告，變數會被新增至全域 Script 環境，這意味著不管是在同一頁上，或是其他頁面上的其他 Script 而言，該變數都是有效的。除非這是您希望如此做，否則，最好使用區域變數以避免造成凌亂全域命名的空間。

使用有意義的名稱給全域變數和 Script Cookie

大多數 Script 很短，所以非常容易理解在函數、Script 連結或事件處理程序內如何使用區域變數。因為區域變數只提供相關的 Script 使用，所以混亂或名稱衝突的風險不大。

然而，當使用全域 Script 變數的動態全域變數（通過 GET 和 SET）和 Script Cookie 時，使用很容易理解的名稱、並提供目的指示，會是一個好的習慣。例如：

```
/* These variables and cookies are not well-named */
SET("c", 15);
WV.SetCookie("st", "Pump is not running.");

/* These are better */
SET("ActivePumpCount", 15);
WV.SetCookie("ActivePumpState", "Pump is not running.");
```

DataHub WebView 的動態全域變數和 Script Cookie 都是列示在Debug Window 中。

Messages | Data Points | Dynamic Globals

Name	Value	Time	Quality
WV.Zoom	Fit	16:28:50.669	Good
SelectedWindTurbine	2	16:28:50.598	Good
AutoCycleSelectedWindTurbine	0	16:28:50.605	Good
ActivePumpState	Pump is not running.	16:28:50.613	Good
ActivePumpCount	15	16:29:31.168	Good

呈現結果在 DataHub WebView；產生邏輯在 DataHub

如同其他的 client – server 解決方案或是服務取向架構，資料和商業邏輯應該保留在服務器上。DataHub WebView 應該被用來管理使用者介面（即展示層）來產生令人滿意的使用者體驗，同時依賴 Cogent DataHub 來處理資料和商業邏輯的複雜系統。

組織 Scripts 成可重複使用的函數

隨著專案的發展，你可能會了解到相同的 Script 表示式或是編碼區塊會用在一個以上的地方。為了避免出現不一致的情況，以及維修問題，可以考慮建立 script 函數，甚至你自己的 script 函數庫。script 函數庫檔案有.ss 延伸檔名，存放在 Web server 上 installDirectory/Silverlight/Script/文件夾中。Script 函數庫可以由 DataHub 管理程式新增至 server 中。函數會在應用程式啓動時自動載入和註冊。

使用 PageData 與 initParams

Cogent DataHub 屬性視窗為 DataHub Web Server 和 DataHub WebView 中提供配置選項。這些配置設定(例如，”啓動在執行模式”和”在啓動時載入的頁面”)影響 DataHub WebView 預設的啟動 URL - installDirectory/Silverlight/DataHubWebView.asp。然而，你也可以建立自己的 HTML 或 ASP 頁面，並提供這些網址讓不同群體來啟動不同的 DataHub WebView。例如，您可能每個部門都有一個網址，每個不同的啟動頁面。或者，你可能需要一個專門網頁來啟動禁止進入設計模式的應用程式。

除此之外，要在 Silverlight <object>tag 上指定 initParams，你也可以在 url 上傳遞參數。例如，這個 URL 傳遞使用者認證、設定一個起始頁面，然後傳送頁面資料(可用於起始頁面上的 Script，以連結控制元件到要求處理的資料點)。

```
http://localhost/Silverlight/AnnualShowKioskOnly.asp&username=demo&password
=demo&page=AnnualShow/Home&pagedata=process=mfg1
```

有兩點需要注意...

1. 指定使用者認證在 url 或 ASP 頁面是一個潛在的安全風險。如果你選擇使用此功能，您應該確保這是僅被授權使用在執行模式，以及只是使用於公開展示的目的。此功能適合使用於像是會展中的自助服務電腦環境中，DataHub WebView 需要跳過登錄畫面，因爲會議的參加者不會有使用者認證。
2. PageData 是一組 name-value 配對。每個名稱都成為 S#的全域變數。在這種情況下，全域變數程序將會被分配字串值“mfg1”，該值就是可以用在點連結表示式，像 this:`="demodata:" + process + "_LineSpeed";`來產生點連結 ofdemodata：mfg1_LineSpeed。

以動態點連結建立模板頁面

考慮建立模板頁面，而不是建立多個相同的頁面給不同的點連結。例如使用 Script 表示式來設定點連結的頁面。

以 Script Cookie 記憶使用者的選擇

以 comboboxes 和 listboxes 提供給使用者選擇，是相當常見的作法。在某些情況下，使用者所做的選擇應該要保留，那麼頁面在下次開啟時，就不需要再次做選擇。大多數情況下，這些選擇是使用者別，且不應該與其他使用者共享。Script Cookie 最適合使用於此目的。

以設計模式中的解釋來提升頁面維護效率

理解和維護複雜的頁面設計可能是一個挑戰，特別是試圖支援頁面的人不是當初的設計者。詳細註解可以使用標準的文字標籤控制元件來新增。為了確保使用者在執行模式下看不到註解，只需將“Visible in Run Mode”屬性設定為 false(在”Common Properties: Content Visibility and Appearance”之下的清單)。

以控制元件工具提示提供在執行模式下的使用者幫助，

創造一個簡單而直觀的頁面設計，有時候並不容易。某些製造過程先天上就有點複雜。為了幫助操作者了解頁面的目的，以及如何與控制元件互動，可以考慮增加工具提示控制(在” Common Properties: ToolTip and Tags”之下的清單)。您甚至可以使用 Script 來設定動態工具提示，或許是為了提供內容訊息給報警和其他情況。