Revitalizing PowerBuilder Applications by Enhancing the User Experience

Matt Balent - Senior Software Engineer - McKesson Provider Technologies

PowerBuilder
Developers Conference

Agenda

Introduction

Enhancement Object

- Basic structure
- Type Ahead Search
- Treeview like Tooltips
- Pointer Effects
- Universal Search Object

Questions

Curriculum vitae

Matthew Balent

- Began with PB 5 at Computer Sciences Corporation
- Senior Software Engineer with McKesson Provider Technologies
- Healthcare, Insurance, Manufacturing, Logistics, and Public Utility software
- SQL Server, Sybase, Oracle database development
- Founding member of North Carolina PowerBuilder Users Group (NCPBUG.ORG) - ISUG user group
- PowerBuilder MVP
- Blog: www.anvil-of-time.com Email: matt.balent@anvil-of-time.com

Introduction

Many PowerBuilder applications which continue to fulfill important business needs were developed 10 – 20 years ago

Projects focused on changes to “internal use” applications tend to have limited management support

How can we improve functionality and increase user satisfaction? Why should we?

Society is undergoing fundamental change due to the consumerization of information technology

“Not so long ago, people's most sophisticated computing experience took place at work while computing at home was limited. A nearly complete role reversal has taken place, Gartner says. The consumerization of IT is a result of the availability of excellent services, interfaces and applications with minimal learning curves. As a result, people feel empowered and expect access to similar functionality both at work and at play and make fewer distinctions between work and non-work activities.”

Source: http://smallbusiness.foxbusiness.com/technology-web/2012/08/01/are-ready-for-tech-apocalypse/#ixzz22Ntfbjpk

What can be done?

Focus on improving basic application functionality expected by users to enhance productivity

- Access information quickly
- Application aids in process flow not dictating it
- Anticipate needs of the user

Leverage object oriented techniques to upgrade existing systems without large scale disruption

- Common behaviors, inputs, and outcomes
- Easy implementation
- Capable of expanding the functionality quickly

Toolkit

‘Enhancement Object (EO)’

Single User Object approach - inherited from datawindow

Parent window handles creation and registration of datawindow controls with the EO

Wire events on datawindow controls to methods on the EO

- Centralizes code in single place
- Less effort to expand functionality
- Easy Implementation
  - Single instance to add to existing code
  - Training is simplified
  - Can incorporate into ancestry

Object Functionality

Type ahead searches for DDDW columns

‘Treeview’ like tooltip text for partially hidden columns

Pointer indicator techniques

‘Universal’ searcher visual object

Demo

Type Ahead for drop down datawindow columns (aka ‘Quicken style’)

- On window declare instance variable of type: matt_uo_search

- In ‘Post Open’ type event register the datawindow(s) and parent window

- Register the specific drop down datawindow columns in each dw

Note: Registering the columns switches them to ‘Allow Edit’ which means you should validate the user entry in itemchanged/ itemerror events. Any getchild references to the dddw columns will have to be re-done if performed prior to this.

- Map the ‘getfocus’, ‘itemfocuschanged’ and ‘editchanged’ events to the corresponding events on the user object.

- If the first column in the datawindow object getting focus is the dddw column, need to trigger itemfocuschanged event when control gets focus

Demo

Treeview Tooltips for datawindow columns

- In the datawindow objects using this, add a text column to use for tracking pointer movement

- On window declare instance variable of type: matt_uo_search

- In ‘Post Open’ type event register the datawindow(s) and parent window

- Register the tracking column with the UO

- Wire a user event on the datawindow to the ‘pbm_dwnmousemove’ event and map it to the ue_nv_mousemove event on the UO

- Map the ‘losefocus’ and ‘resize’ events to the corresponding events on the user object.

Demo

Pointer Indicator (‘mouseover’) Effects

- In the datawindow objects using this, add a text column to use for tracking pointer movement - On window declare instance variable of type: matt_uo_search

-In ‘Post Open’ type event create the UO

-In each datawindow ‘getfocus’ register the datawindow(s) and parent window and the tracking column with the UO - Wire a user event on the datawindow to the ‘pbm_dwnmousemove’ event and map it to the ue_nv_mousemove event on the UO

-Map the ‘losefocus’ and ‘resize’ events to the corresponding events on the user object. Note: modifies the background color of the row. If the background of the fields are not transparent the effect will not show through.

Demo

‘Universal’ Search Object

- - On window declare instance variable of type: matt_uo_search

❑- For Datawindow Contols

1. Map the clicked event to the ue_obj_clicked event
2. In the rbuttondown event open the UO, register the control, and set parameters (columns to search in and event to post after search)

❑- For Treeview, MultiLineEdit, RichTextEdit Controls

- - In the rbuttondown event open the UO, register the control, and set parameters (event to post after search)

- Notes: Search is case insensitive and is top downward. In DW Search, if nothing is found, the search string has the last character removed for a more generic search. Normally <Ctl> + Rt Click on the searchable control activates the UO and a <Ctl> + Rt Click on the UO closes it.

Questions

Matt.balent@anvil-of-time.com

For further consideration on Software Design :

Information Software and the Graphical Interface by Bret Victor

http://worrydream.com/MagicInk/

Questions

Matt.balent@anvil-of-time.com

For further consideration on Software Design :

Information Software and the Graphical Interface by Bret Victor

http://worrydream.com/MagicInk/