A Demo Project Using PowerBuilder 2022 R3

TABLE OF CONTENTS

1. INTRODUCTION
   1. INTRODUCTION TO POWERBUILDER AND PROJECT
   2. FUNCTIONAL AND NON FUNCTION REQUIREMENTS
2. BUILDING A CLIENT SERVER APPLICATION
   1. BUILDING A LOGIN WINDOW
   2. CONNECTING TO THE DATABASE
   3. CREATING ANCESTOR WINDOW
   4. SETTING UP THE MENUS
   5. BUILDING DATA WINDOW OBJECTS AND ATTACHING
   6. RUNNING THE DEBUGGER
   7. PREPARING THE APPLICATION FOR DEPLOYMENT
3. AUTOMATING BUILD AND DEPLOY
4. INTRODUCTION
   1. INTRODUCTION TO POWERBUILDER AND PROJECT

INTRODUCTION

PowerBuilder is an enterprise development tool that allows developers to create a variety of applications,   
 including client/server, multi-tier, and internet-based solutions. This document provides step-by-step guidance on building a   
 simple database application using PowerBuilder 2022 R3.

INTRODUCTION TO POWERBUILDER AND PROJECT

PowerBuilder enables developers to build applications with a rich graphical   
 interface, integrate business logic, and leverage event-driven programming. The core elements include:  
 - User Interface: Windows, menus, and controls.  
 - Event-Driven Logic: Scripts executed in response to user actions.  
 - PowerScript Language: Custom scripting for event handling.  
 - Object-Oriented Design: Supports encapsulation, inheritance, and polymorphism.  
 - Database Connectivity: Integration with databases via ODBC, JDBC, and native connections.  
  
 Project Scope  
 The project focuses on developing a database-driven application for managing customers and products.

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Functional Requirements  
 - User authentication via a login window.  
 - CRUD operations for customers and products.  
 - Menu-driven navigation.  
 - Database integration for storing and retrieving data.  
  
 Non-Functional Requirements  
 - Responsive and user-friendly interface.  
 - Secure data handling.  
 - Efficient query processing.  
 - Scalable architecture for future enhancements.

BUILDING A CLIENT SERVER APPLICATION

Steps  
 1. Create a new workspace in PowerBuilder.  
 2. Create a target to define the application framework.  
 3. Specify an icon for the application.  
 4. Adjust the main window size.

BUILDING A LOGIN WINDOW

1. Create a new window for user login.  
 2. Add controls: Username, password fields, and login button.  
 3. Define tab order for input fields.  
 4. Handle Help events and preview the window.  
 5. Write event scripts to validate credentials and open the main application.

CONNECTING TO THE DATABASE

1. Use the Demo Database provided by PowerBuilder.  
 2. Run the Connection Object Wizard to establish a connection.  
 3. Declare a global variable for database access.  
 4. Modify connection parameters as needed.  
 5. Implement login and logout scripts for database interaction.  
 6. Test the connection by running the application.

CREATING ANCESTOR WINDOW

1. Add a library to the search path.  
 2. Create an ancestor sheet window for reuse.  
 3. Define inheritance hierarchy for consistent UI elements.  
 4. Add DataWindow controls for master-detail data presentation.  
 5. Customize inherited scripts to handle user-specific requirements.

SETTING UP THE MENUS

1. Modify the main frame menu for navigation.  
 2. Create a new sheet menu with structured options.  
 3. Attach event scripts to handle menu actions.  
 4. Test the application to ensure proper navigation.

BUILDING DATA WINDOW OBJECTS AND ATTACHING

1. Create DataWindow objects for customer and product management.  
 2. Preview and adjust UI elements for clarity.  
 3. Save and attach DataWindow objects to respective controls.  
 4. Execute queries to test data retrieval.  
 5. Run the application and validate functionality.

RUNNING THE DEBUGGER

1. Set breakpoints in key application scripts.  
 2. Run the application in debug mode to analyze execution flow.  
 3. Use watch variables to track data state.  
 4. Apply conditional breakpoints for advanced debugging.

PREPARING THE APPLICATION FOR DEPLOYMENT

1. Create a Project object to manage deployment.  
 2. Generate an executable file for distribution.  
 3. Define a shortcut for easier access.  
 4. Test the application package before release.

AUTOMATING BUILD AND DEPLOY

1. Use PowerBuilder’s OrcaScript to automate builds.  
 2. Implement CI/CD pipelines for streamlined deployment.  
 3. Utilize PowerServer for cloud-based distribution.

PULLING CODE FROM GITHUB USING A BATCH SCRIPT

In this project, a batch script is used to automate pulling the latest code from a GitHub repository.   
This ensures that the latest updates are always available before building and deploying the application.  
  
 Steps to Automate Git Pull Using a Batch Script  
1. Create a new batch script file (`update\_code.bat`).  
2. Add the following script to pull the latest code:  
 ```batch  
 @echo off  
 cd /d "C:\Path\To\Your\Project"  
 git pull origin main  
 echo Code updated successfully!  
 pause  
 ```  
3. Save the file and run it before building the application to ensure the latest code is used.  
  
This method helps keep the local workspace synchronized with the GitHub repository, minimizing conflicts and ensuring all changes are up to date.

In this project, PowerBuilder's Source Control feature is used to connect to Git for version management.   
  
### \*\*Connecting PowerBuilder to Git for Source Control\*\*  
1. Open PowerBuilder and navigate to \*\*Tools > Source Control\*\*.  
2. Select \*\*Git\*\* as the source control provider.  
3. Configure the repository settings:  
 - Enter the \*\*Git Repository URL\*\*.  
 - Provide authentication details if required.  
 - Choose a local working directory.  
4. Click \*\*Connect\*\* to establish the link with Git.  
5. Use \*\*Check-in/Check-out\*\* options within PowerBuilder to manage code changes.  
  
By integrating Git with PowerBuilder, developers can track changes, collaborate efficiently, and maintain a structured version history while automating the build and deployment process.