



# JavaOne<sup>SM</sup>

Sun's 2002 Worldwide Java Developer Conference™

## A Framework for Assembling Apps From Modular Components

*Making Life Easier*

**Dave Wilson**

Architect

Portal Software, Inc.

# Overall Presentation Goal

Learn how to make **life easier for your developers** with modular development, and learn how to **make life easier for your customers** by combining multiple separate application modules into “Centers”

# Learning Objectives

- As a result of this presentation, you will be able to:
  - Make user interface decisions about combining components into a single application
  - Design a Java™ technology-based framework (“Java framework”) that allows components to be built separately, and then combined
  - Use properties files to define how components should be combined



# Speaker's Qualifications

- Dave Wilson
  - Architect at Portal Software (Cupertino, CA and Hamburg, Germany)
  - Have used many applications and problem—domain frameworks for over the last 14 years
  - Have written a number of my own frameworks using Object Pascal, C++, and the Java programming language
  - I used the BMF framework described today to help develop an integrated suite of developer tools



# Co-authors' Qualifications

- Raghu Venkateshwaran
  - Staff Engineer at Portal Software (Cupertino)
  - Developer of the BMF framework
- Guido Krüger
  - Principal Engineer at Portal Software (Hamburg)
  - Java framework and library developer
  - Author of best-selling books for Java™ technology in Germany

# Are Your Customers Confused, Frustrated, and Angry?



# Are Your Customers Confused, Frustrated, and Angry?

- Your job is not to just write software



# Are Your Customers Confused, Frustrated, and Angry?

- Your job is not to just write software
- Your job is to make life simple for your customer





# Are Your Customers Confused, Frustrated, and Angry?

- Your job is not to just write software
- Your job is to make life simple for your customer
- Large collections of unrelated tools make life more difficult for customers. This makes customers frustrated (and angry)



# Are Your Customers Confused, Frustrated, and Angry?

- Your job is not to just write software
- Your job is to make life simple for your customer
- Large collections of unrelated tools make life more difficult for customers. This makes customers frustrated (and angry)
- Integrating your tools into functional "centers" of activity can make your customers happier and more productive



# Are Your Customers Confused, Frustrated, and Angry?

- Your job is not to just write software
- Your job is to make life simple for your customer
- Large collections of unrelated tools make life more difficult for customers. This makes customers frustrated (and angry)
- Integrating your tools into functional "centers" of activity can make your customers happier and more productive



# Presentation Agenda

- Market requirements (motivation)
- Terminology (what is a component???!?)
- Phase I: **B**usiness **M**anagement **F**ramework
  - Live demo
  - GUI design
  - Framework architecture
  - Framework implementation
  - Lessons learned
- Phase II: **P**ortal “**J**ava **F**ramework”
  - Work in progress
  - Demo of prototype
  - Architecture



# Market Requirements

- Portal's Intranet Customer Management and Billing system requires tools for:
  - Installation and Configuration
  - Schema and Code customization
  - Defining price plans
  - Customer management
  - Bulk loading of batch usage data
  - Billing
  - Accounts Receivable
  - Etc.
- Complex system; long-learning curve



# Market Requirements

- Support multiple client platforms
  - Solaris, HP-UX, IBM AIX, Windows
  - Solution: Java technology
- Provide easy-to-use GUIs
  - Solution: Java Foundation Classes (JFC/SWING) API
- Support for world-wide customer base: I18N, L10N
  - Solution: Java technology resources and careful attention to detail



# Market Requirements

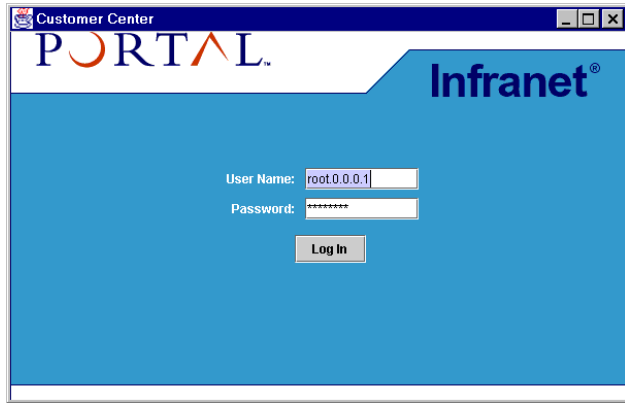
- Manage complexity for our developers and customers
  - Solution: Custom framework for combining related components into application “Centers”
  - Customer Center
  - Developer Center
  - Pricing Center
  - Etc.



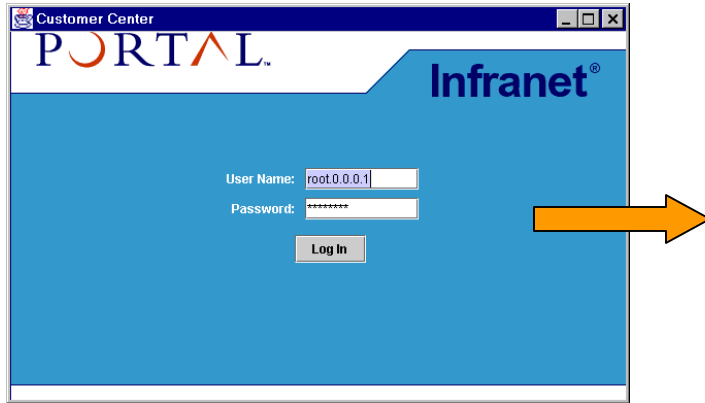
# Bad: Too Many Separate Applications w/Separate Logins



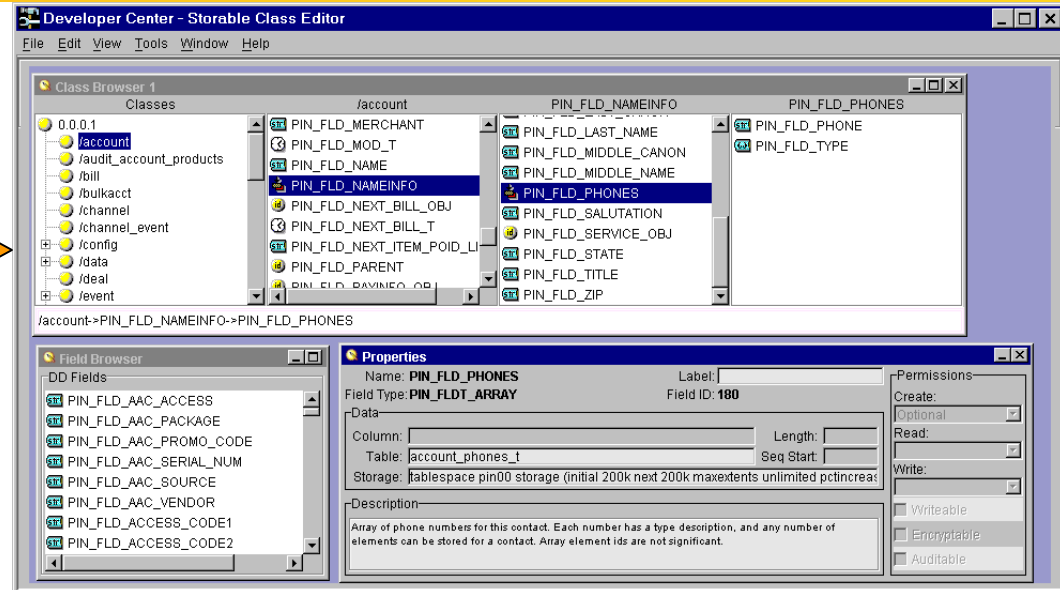
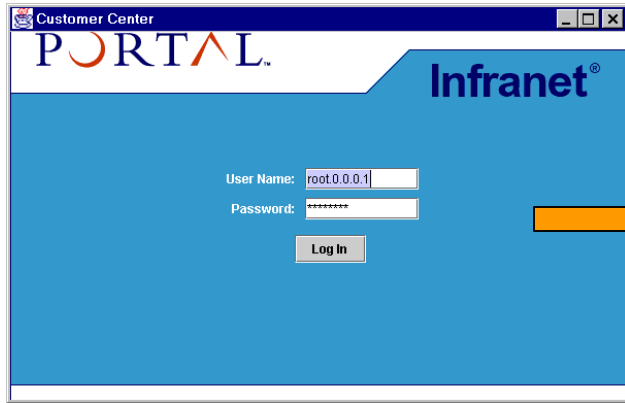
# Bad: Too Many Separate Applications w/Separate Logins



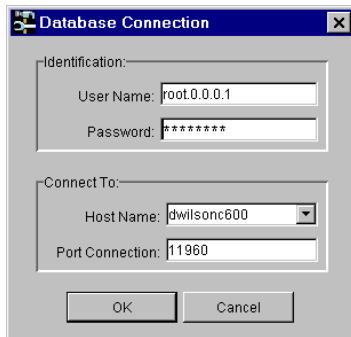
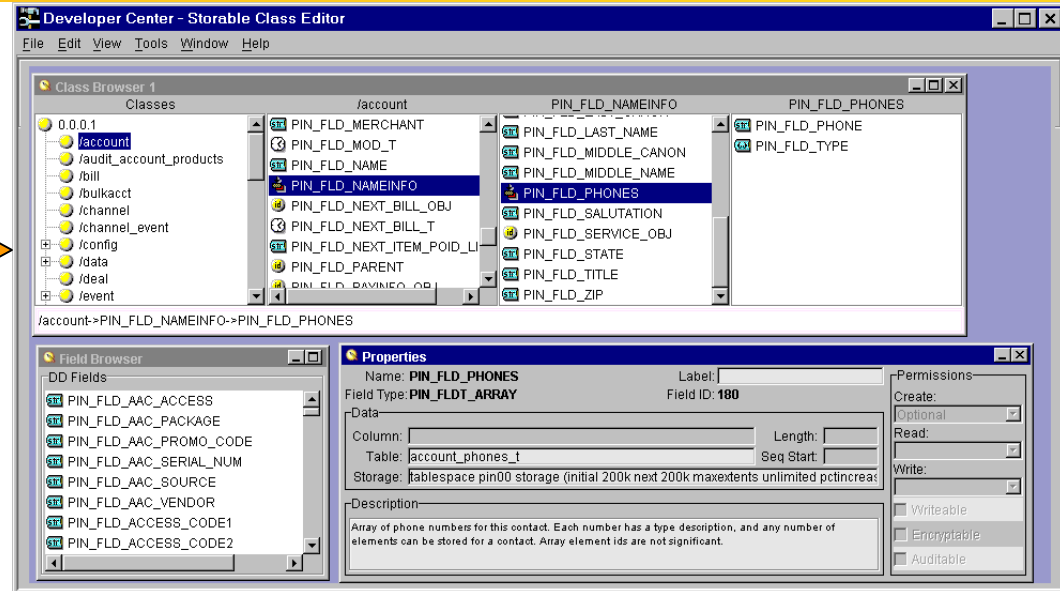
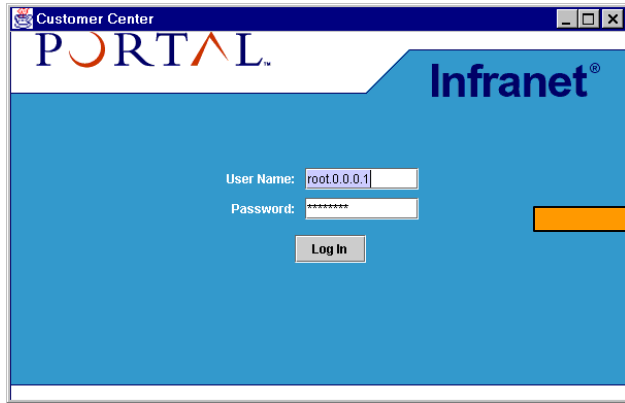
# Bad: Too Many Separate Applications w/Separate Logins



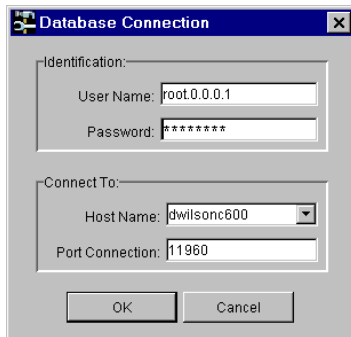
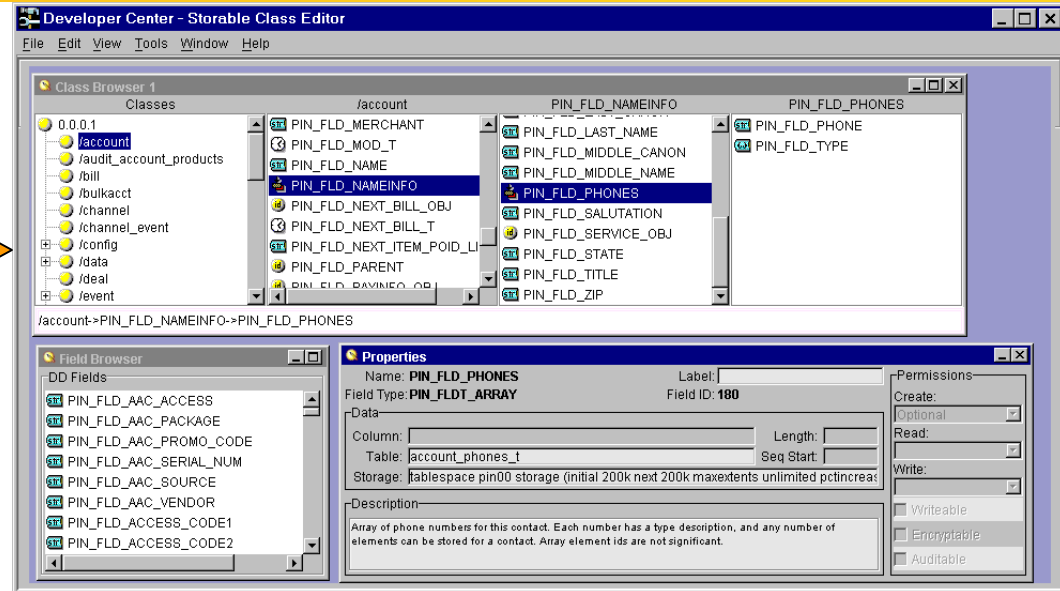
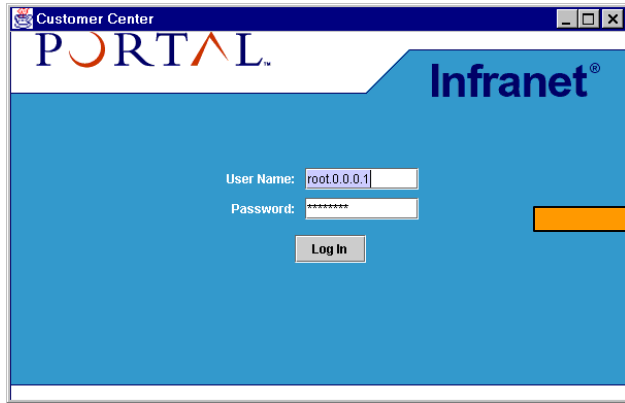
# Bad: Too Many Separate Applications w/Separate Logins



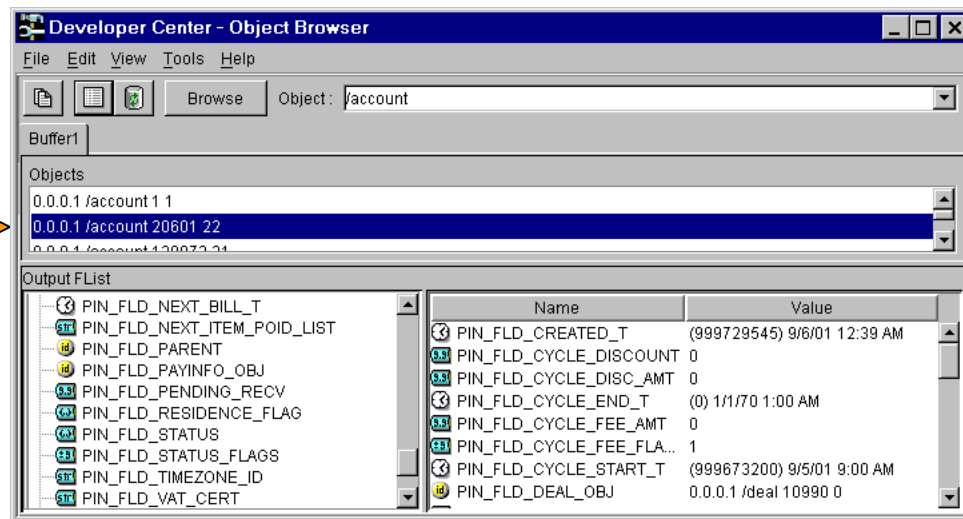
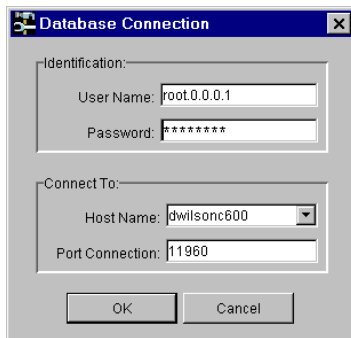
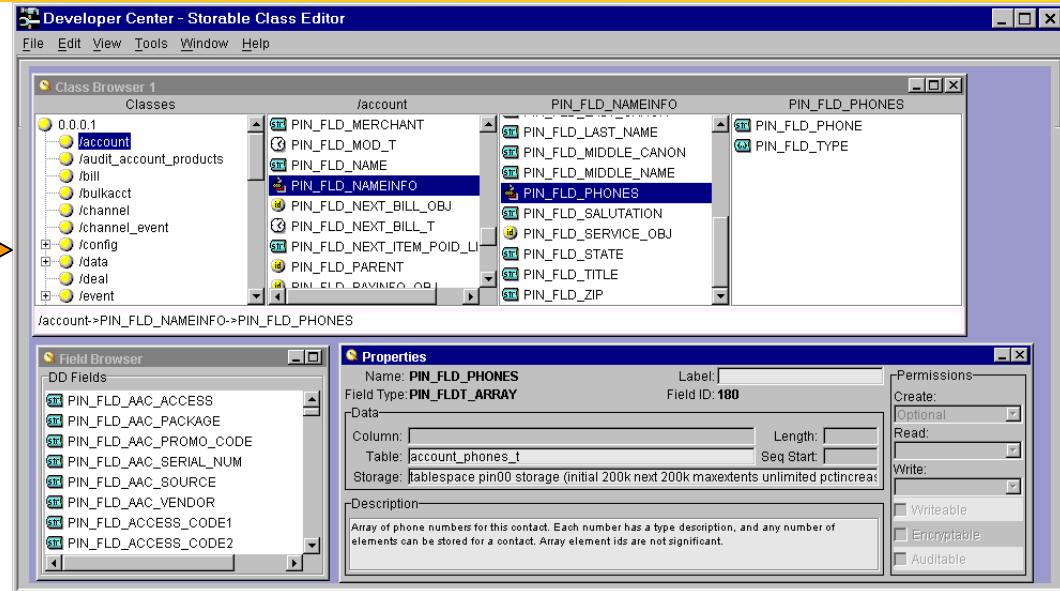
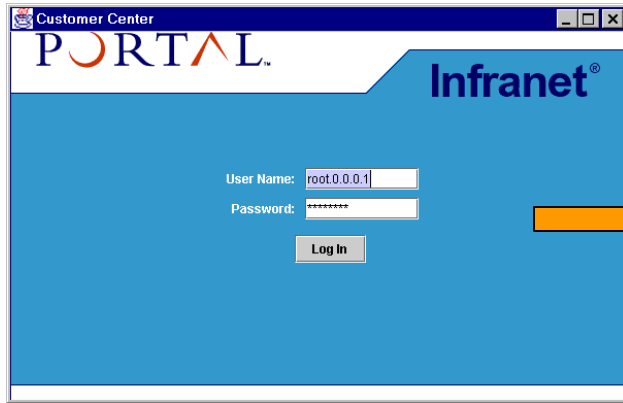
# Bad: Too Many Separate Applications w/Separate Logins



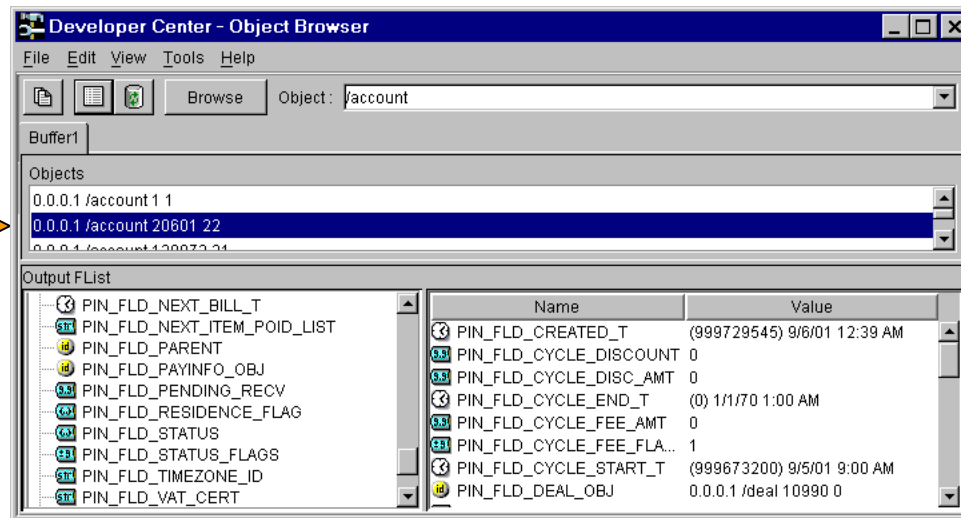
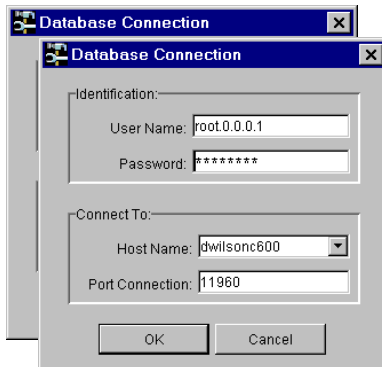
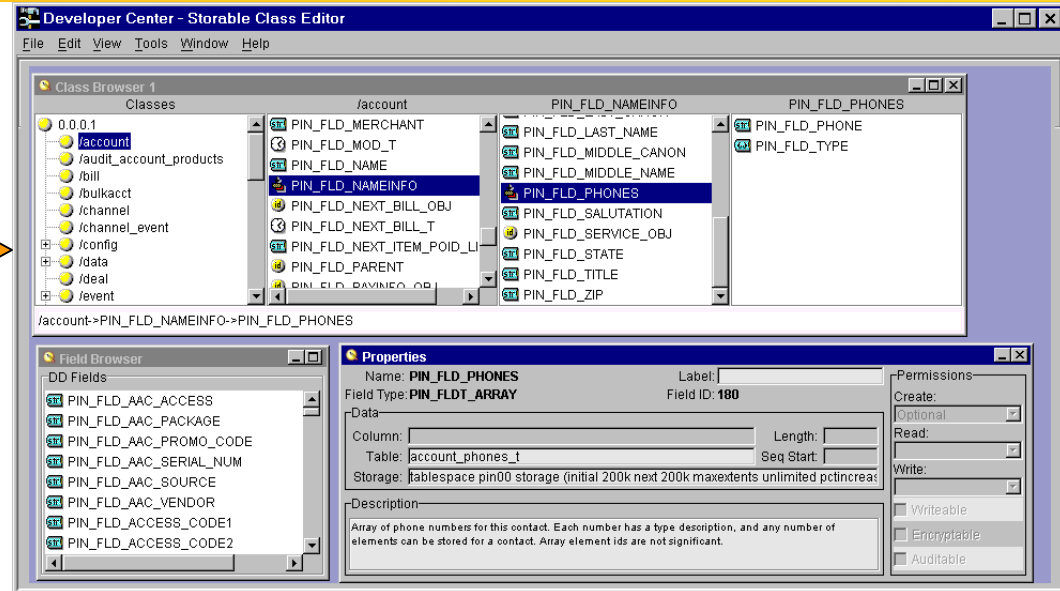
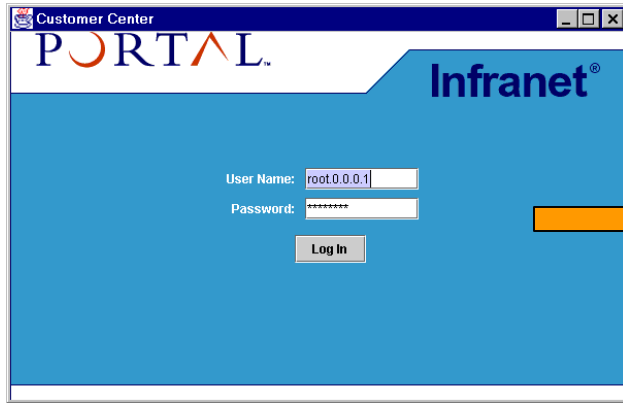
# Bad: Too Many Separate Applications w/Separate Logins



# Bad: Too Many Separate Applications w/Separate Logins



# Bad: Too Many Separate Applications w/Separate Logins



# Bad: Too Many Separate Applications w/Separate Logins

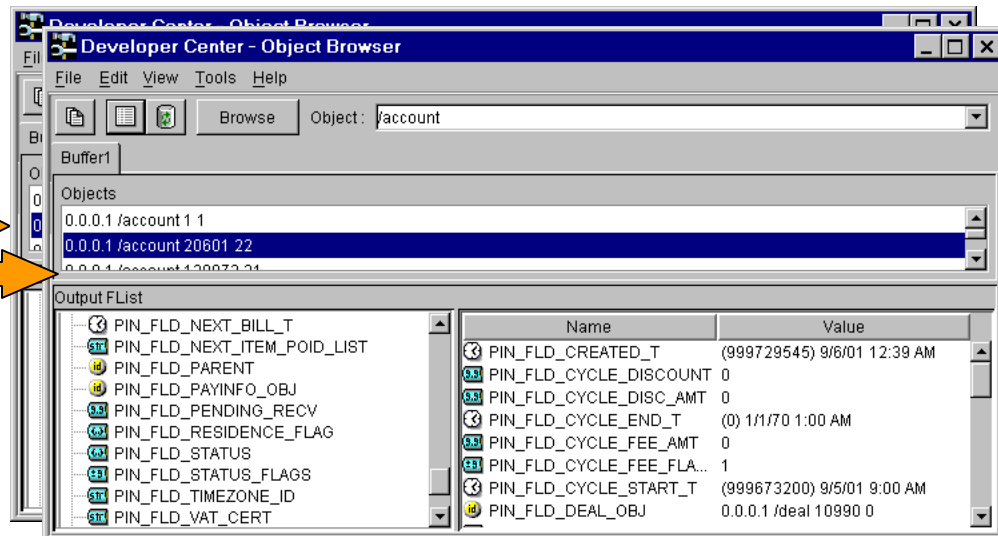
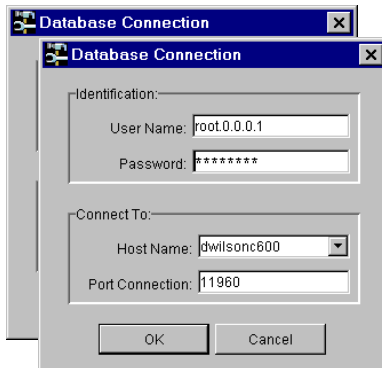
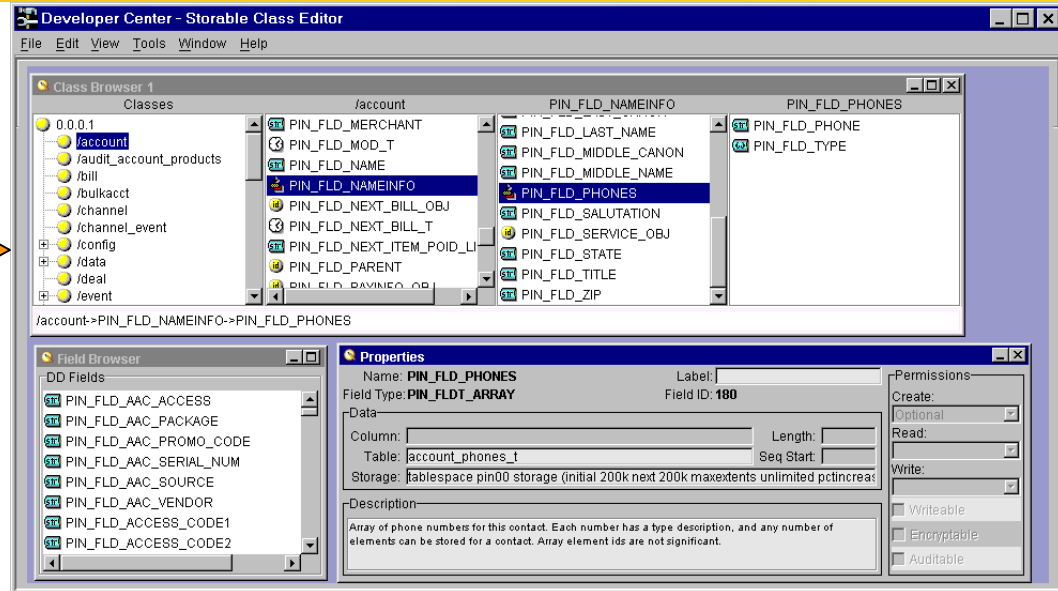
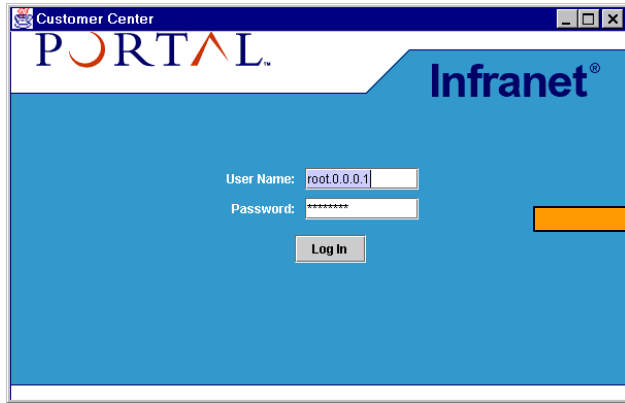
The diagram illustrates the complexity of having multiple separate applications for a single system. It shows three distinct windows:

- Customer Center Portal:** A web interface for user login. It features a "PORTAL" logo, the "Infranet" brand name, and a login form with fields for "User Name" (containing "root.0.0.0.1") and "Password" (masked with asterisks), and a "Log In" button.
- Developer Center - Storable Class Editor:** A tool for defining classes and fields. The "Class Browser" shows a tree structure with "Classes" and "PIN\_FLD\_NAMEINFO" selected. The "Field Browser" lists various fields like "PIN\_FLD\_AAC\_ACCESS". The "Properties" window for "PIN\_FLD\_PHONES" shows it is a "PIN\_FLD\_ARRAY" field with a "Field ID" of 180, associated with the "account\_phones\_t" table.
- Developer Center - Object Browser:** A tool for viewing database objects. The "Object" dropdown is set to "/account". The "Objects" list shows several instances of "/account". The "Output FList" window displays a list of fields and their values, such as "PIN\_FLD\_CREATED\_T" with a value of "9/6/01 12:39 AM".

Arrows indicate the flow of data and interaction: from the Customer Center Portal to the Developer Center - Storable Class Editor, and from the Database Connection dialog to the Developer Center - Object Browser.

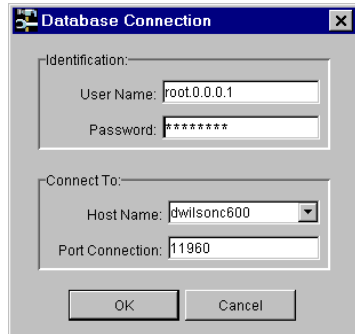


# Bad: Too Many Separate Applications w/Separate Logins



# Good: “Developer Center”: One Application = Four Modules

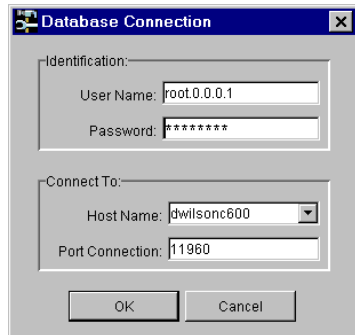
# Good: “Developer Center”: One Application = Four Modules



The image shows a 'Database Connection' dialog box with the following fields and controls:

- Identification:**
  - User Name:
  - Password:
- Connect To:**
  - Host Name:
  - Port Connection:
- Buttons:  and

# Good: “Developer Center”: One Application = Four Modules



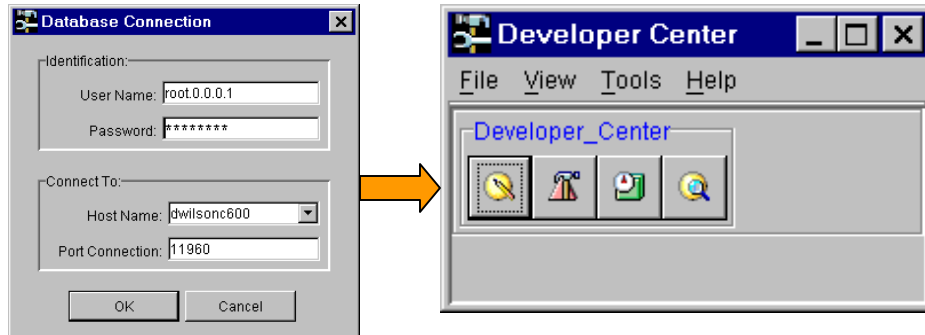
The image shows a 'Database Connection' dialog box with the following fields and values:

- Identification:
  - User Name: root.0.0.0.1
  - Password: \*\*\*\*\*
- Connect To:
  - Host Name: jwilsonc600
  - Port Connection: 11960

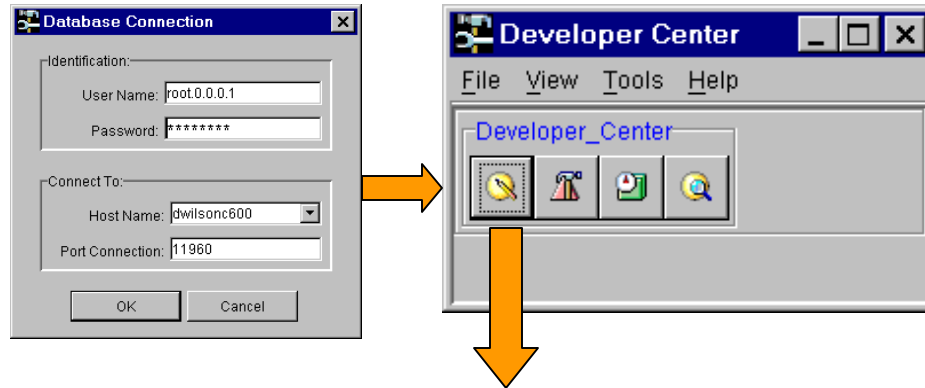
Buttons: OK, Cancel

An orange arrow points from the right side of the dialog box towards the right edge of the slide.

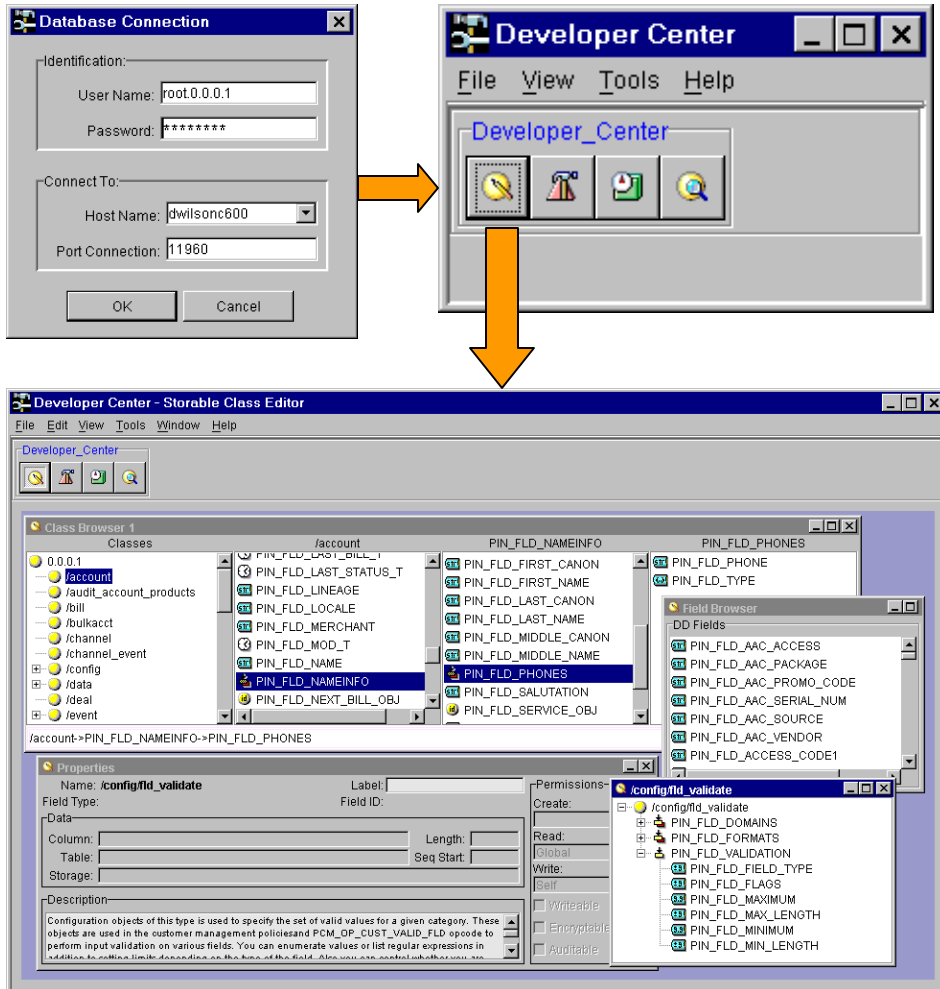
# Good: “Developer Center”: One Application = Four Modules



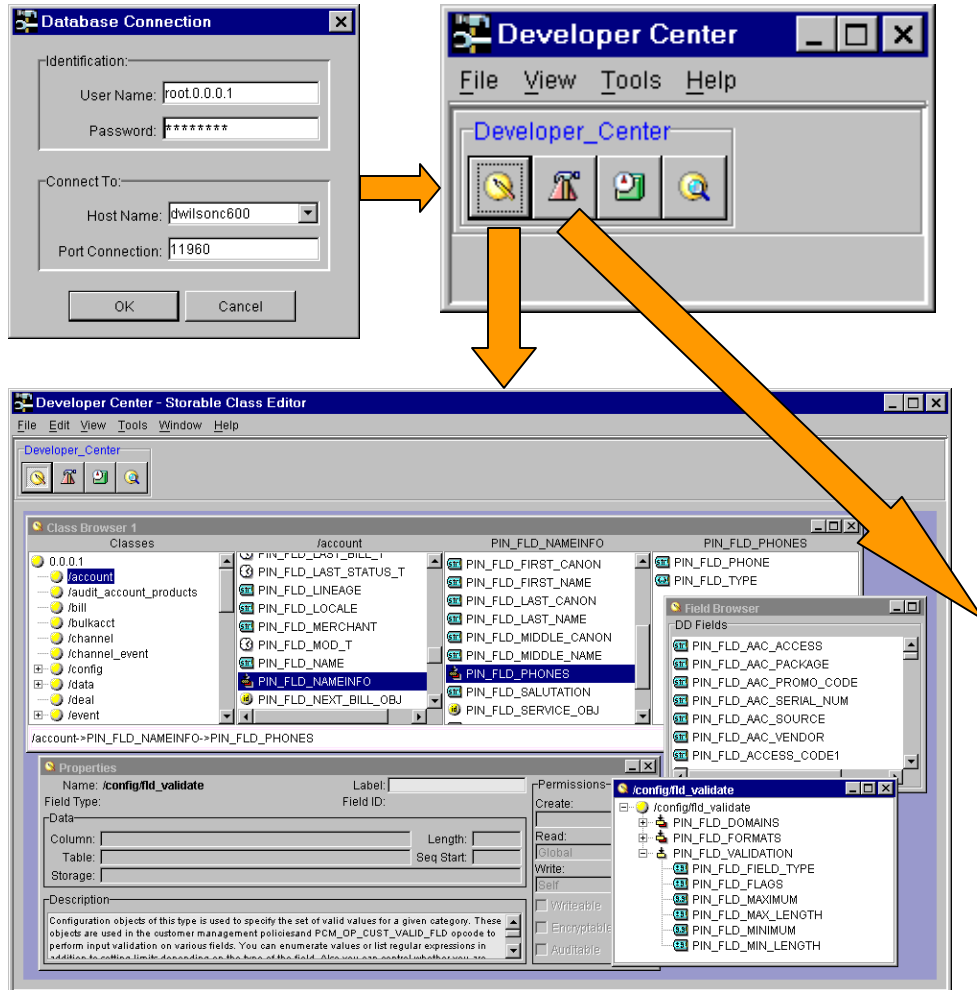
# Good: “Developer Center”: One Application = Four Modules



# Good: “Developer Center”: One Application = Four Modules

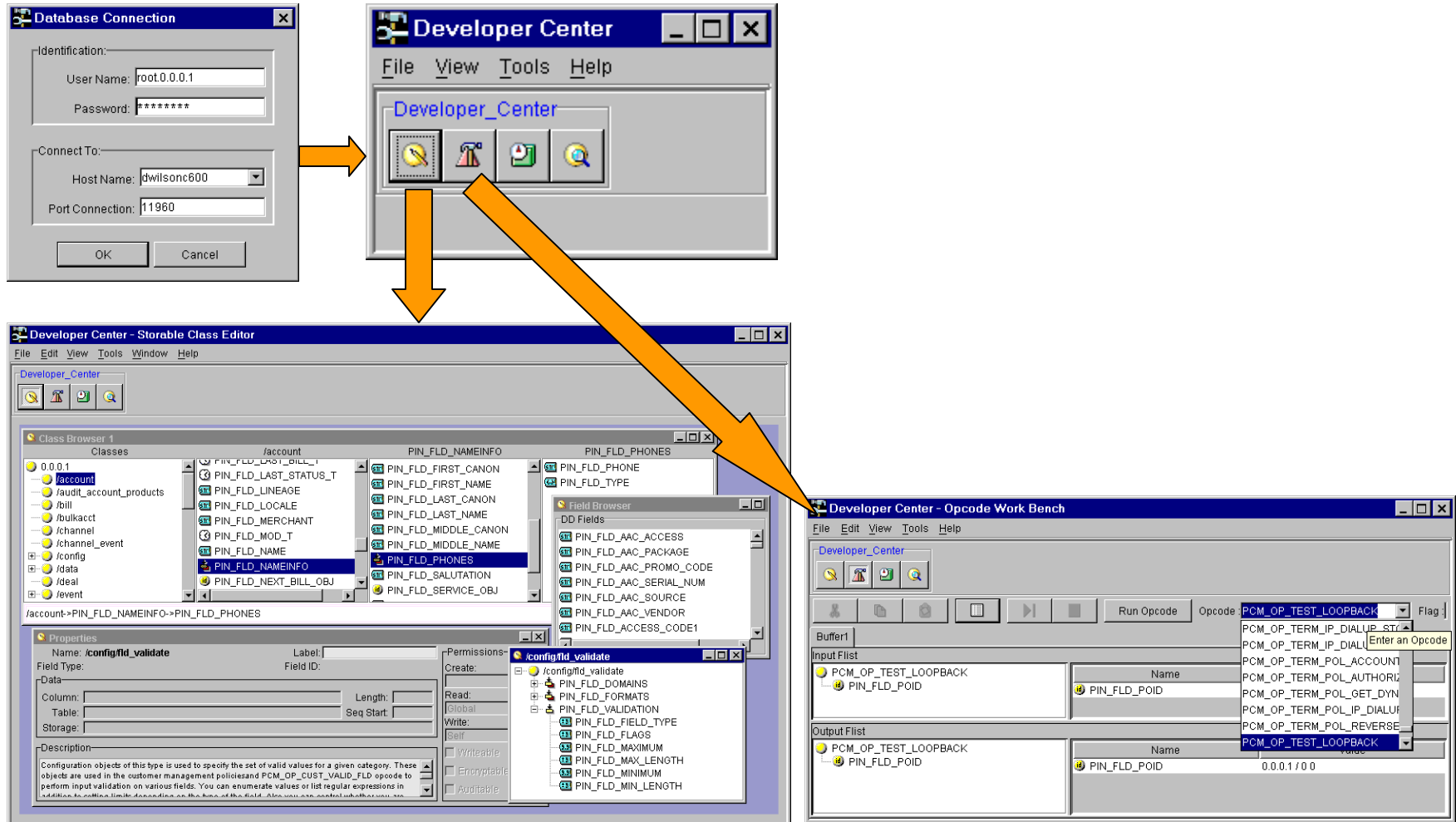


# Good: “Developer Center”: One Application = Four Modules

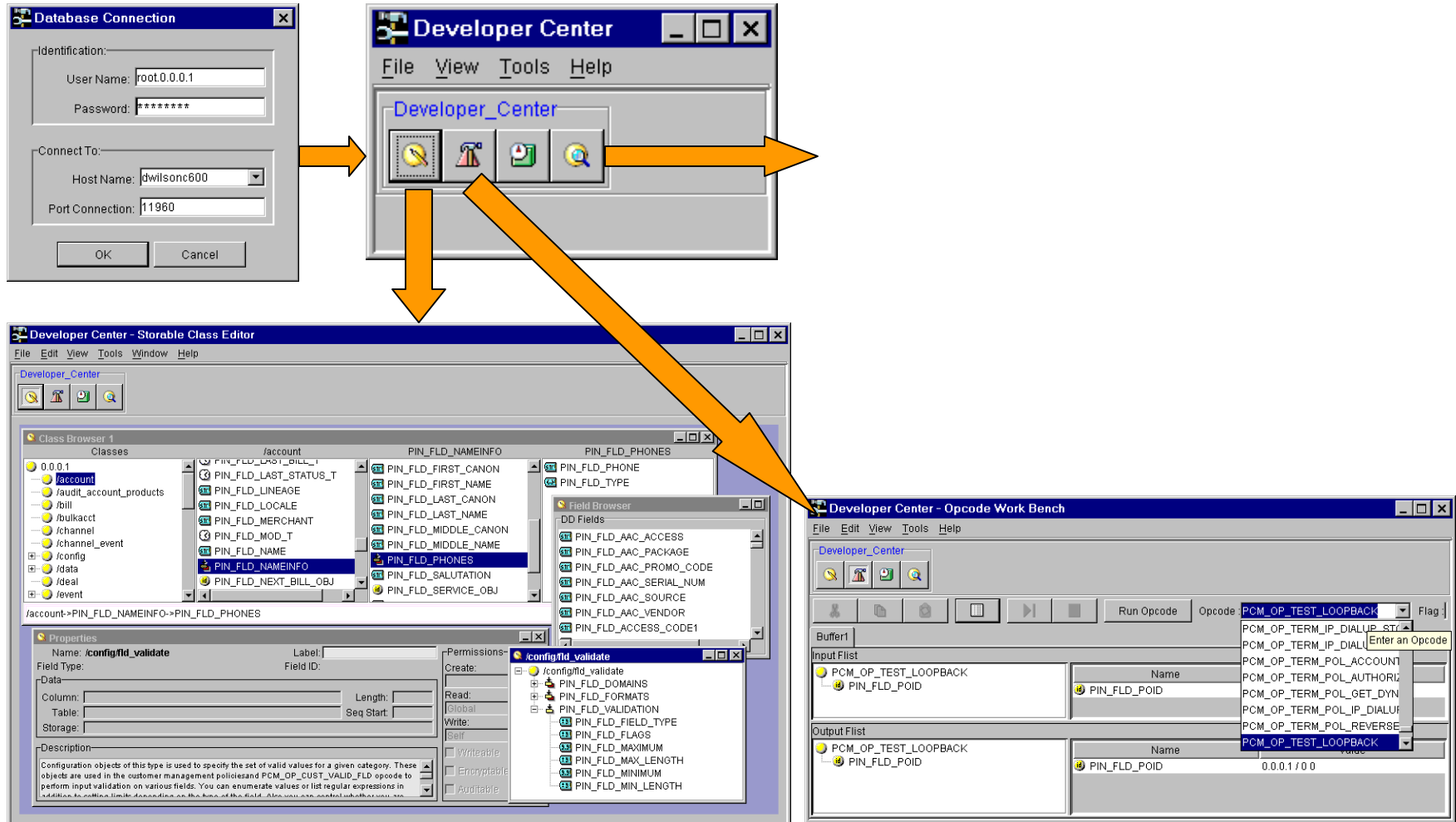




# Good: "Developer Center": One Application = Four Modules



# Good: "Developer Center": One Application = Four Modules

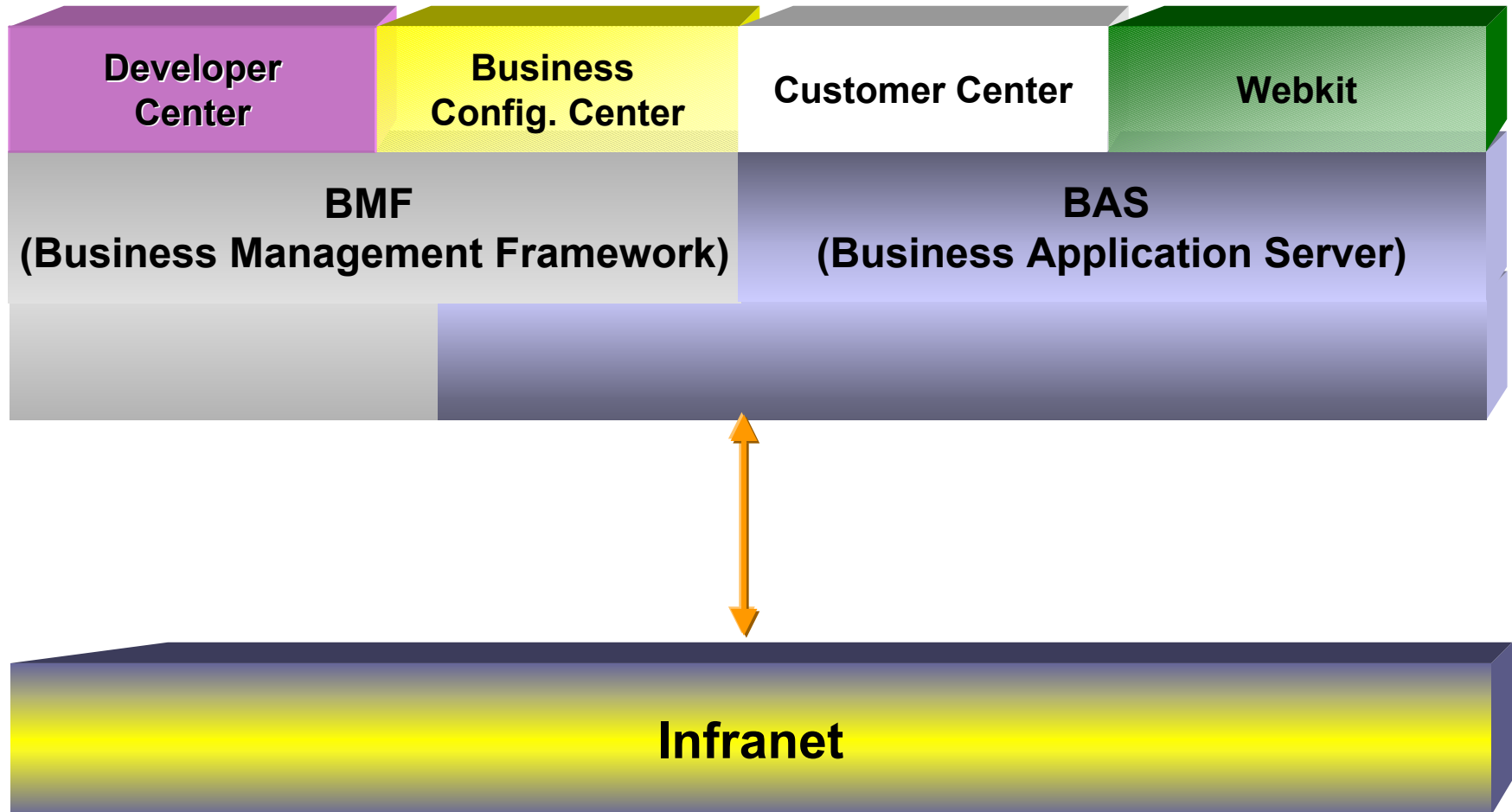


# Good: "Developer Center": One Application = Four Modules

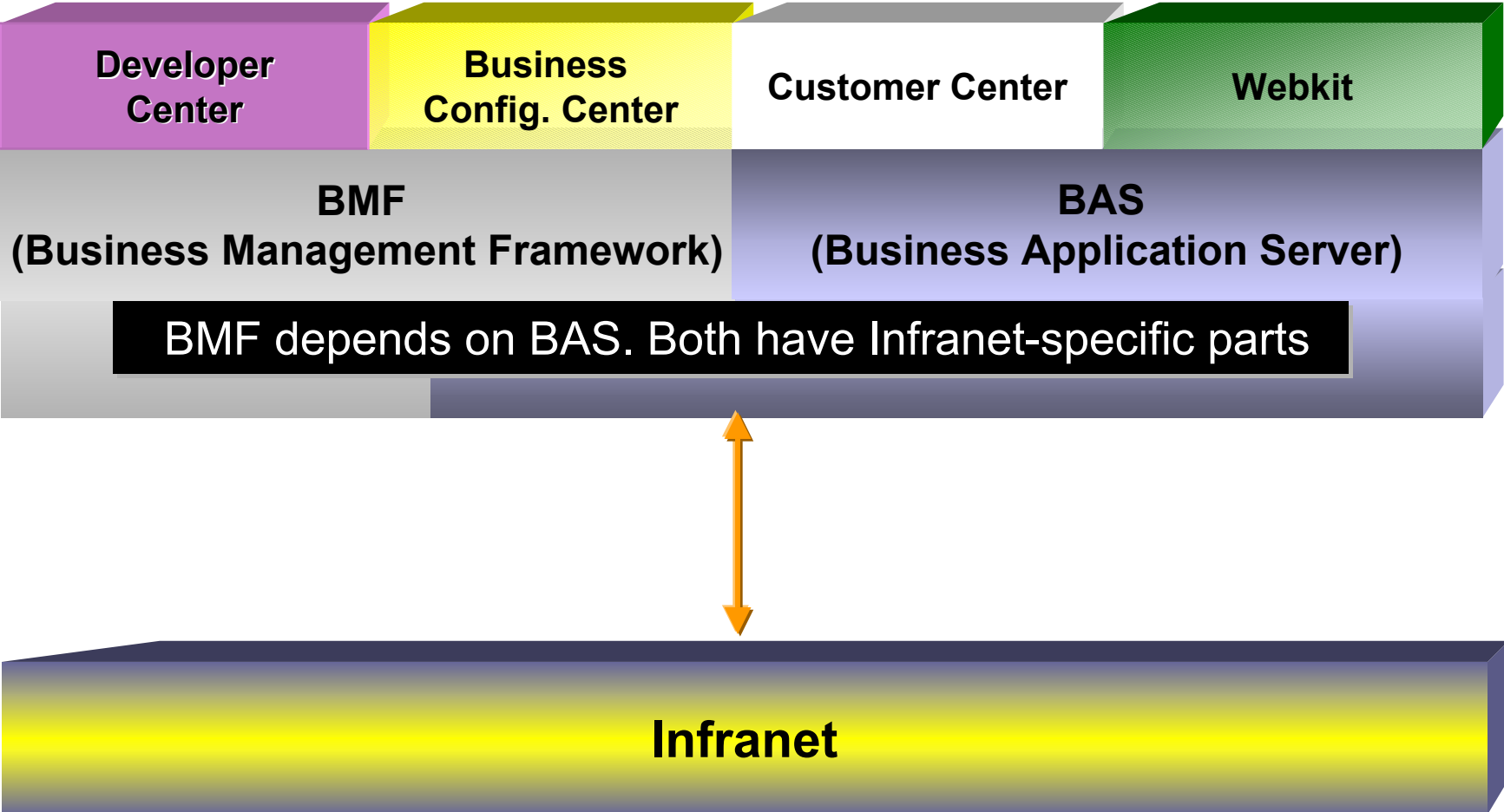
The diagram illustrates the workflow of the Developer Center application, showing how one application is composed of four modules:

- Database Connection:** A dialog box for configuring database access, including fields for User Name, Password, Host Name, and Port Connection.
- Developer Center:** The main application window, which serves as the central hub for the other modules.
- Developer Center - Object Browser:** A module for browsing objects, showing a tree view of objects for the 'account' class, including fields like 'PIN\_FLD\_NEXT\_ITEM\_POID\_LIST', 'PIN\_FLD\_PARENT', and 'PIN\_FLD\_PRODUCTS(1)'. It includes a table with columns for Name and Value.
- Developer Center - Storable Class Editor:** A module for editing storable classes, showing a tree view of classes for the 'account' class, including fields like 'PIN\_FLD\_NAMEINFO' and 'PIN\_FLD\_PHONES'. It includes a table with columns for Name and Value.
- Developer Center - Opcode Work Bench:** A module for working with opcodes, showing a list of opcodes for the 'PIN\_FLD\_POID' field, including 'PCM\_OP\_TEST\_LOOPBACK' and 'PCM\_OP\_TERM\_IP\_DIALUP\_ST'. It includes a table with columns for Name and Value.

# Infranet Client/Server Architecture BMF and BAS Client Frameworks



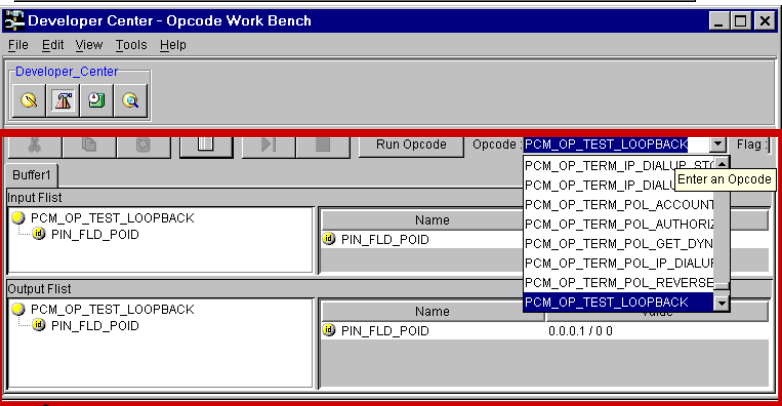
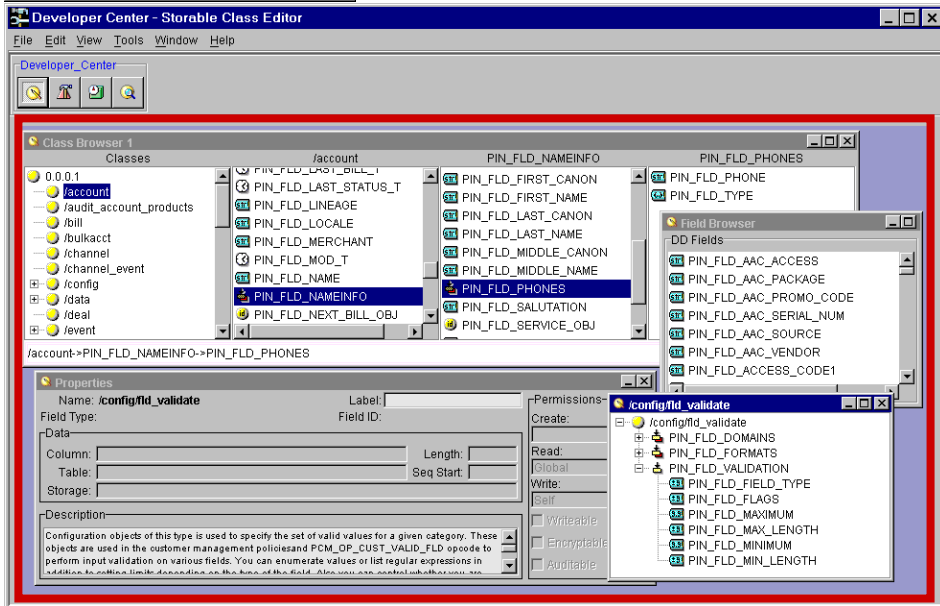
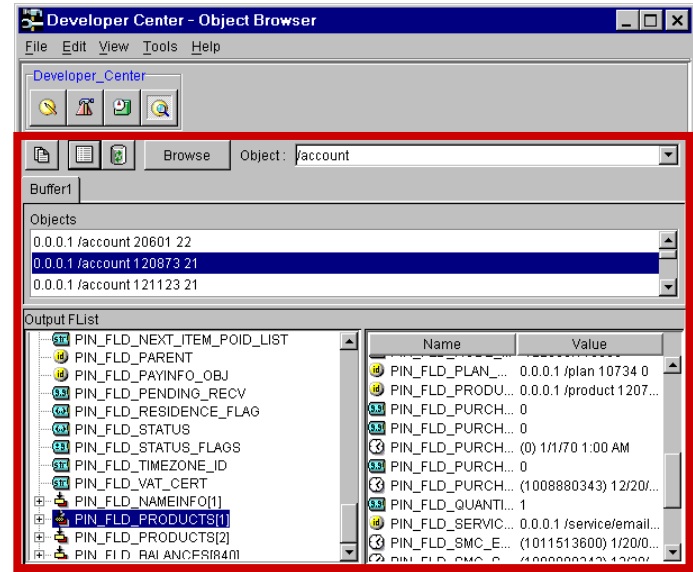
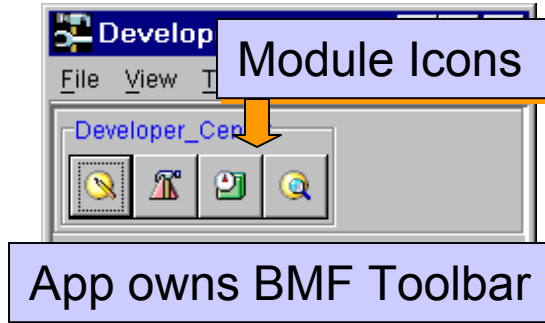
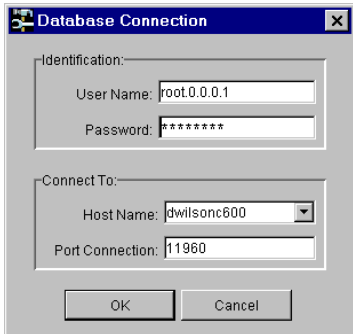
# Infranet Client/Server Architecture BMF and BAS Client Frameworks



# I. BMF Features and Benefits

- Provides a pluggable module architecture
- Provides an Intranet login module and obtains the Intranet connection
- Creates and manages Menu bar, toolbar and status bar
- Takes care of application Help, About, etc.
- Supports some product-specific features

# Terminology



BMF Module

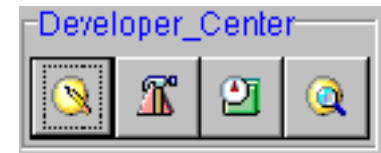
# Demo

## I.Developer Center

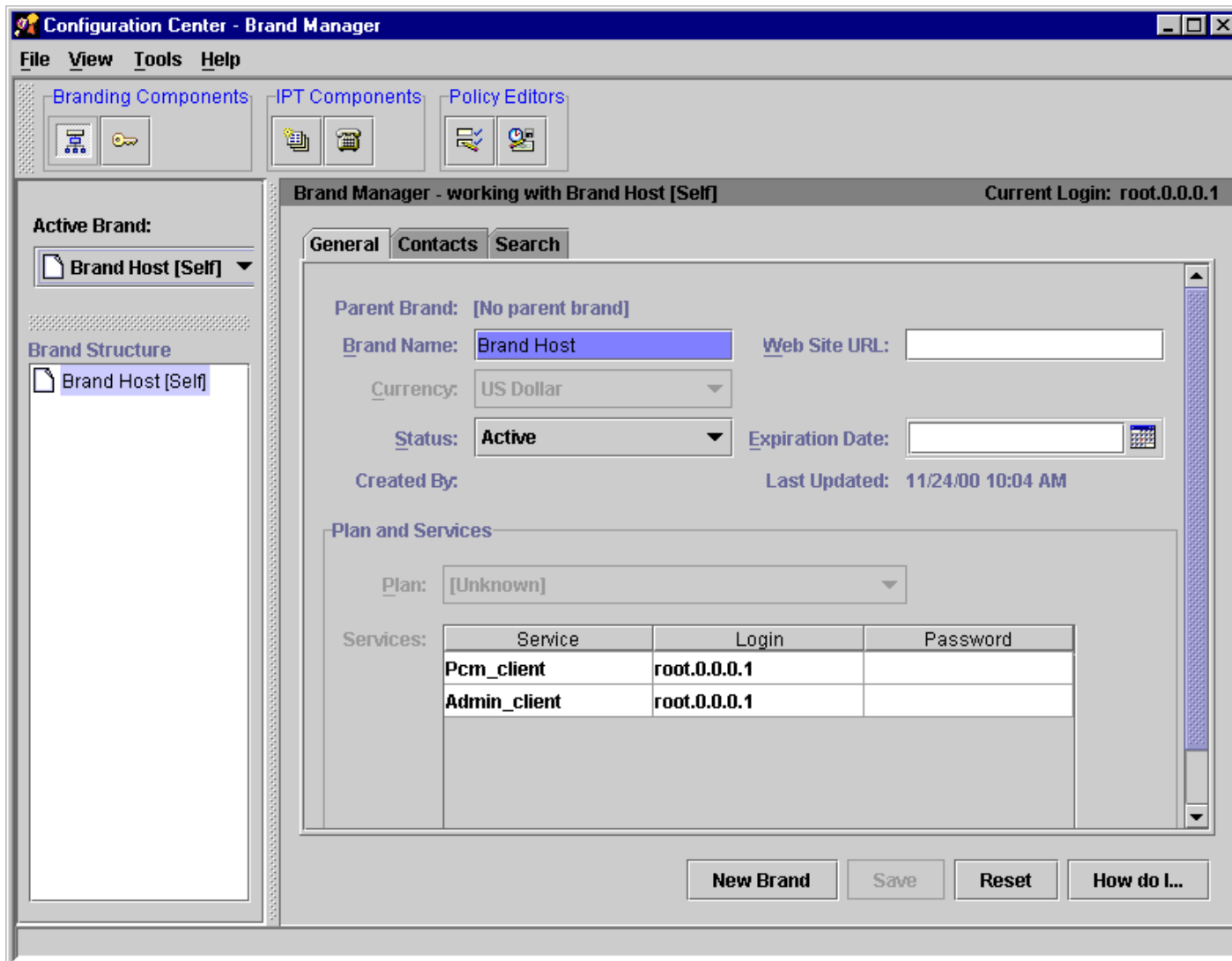


# I. Visual Design Choices

- Toolbar contains an icon for each module
- Related icons can be grouped
- Only one module body is visible at a time
  - Select visible module by clicking on toolbar icon, or selecting menu item
- Menu bar switches to show menus for visible module
- Status area on bottom
- Module contains a JComponent or JPanel
  - Can do anything you want inside the component
  - Other panels, tabbed panes, internal frames, tables, etc.



# I. Example BMF Application



# Designing an Applications Framework

- How does someone run the application?
  - Is there a “manager” object?
- What classes should the developer instantiate?
  - Public methods to call?
- What is the default behavior? Where is the flow of control?
- How is the framework customized?
  - Data-driven (properties files, XML files, etc.)
  - Subclassing and overriding
- What should the developer subclass
  - Protected methods to override?
- What developer methods are called when the user does things?



# I. Architectural Decision: Application

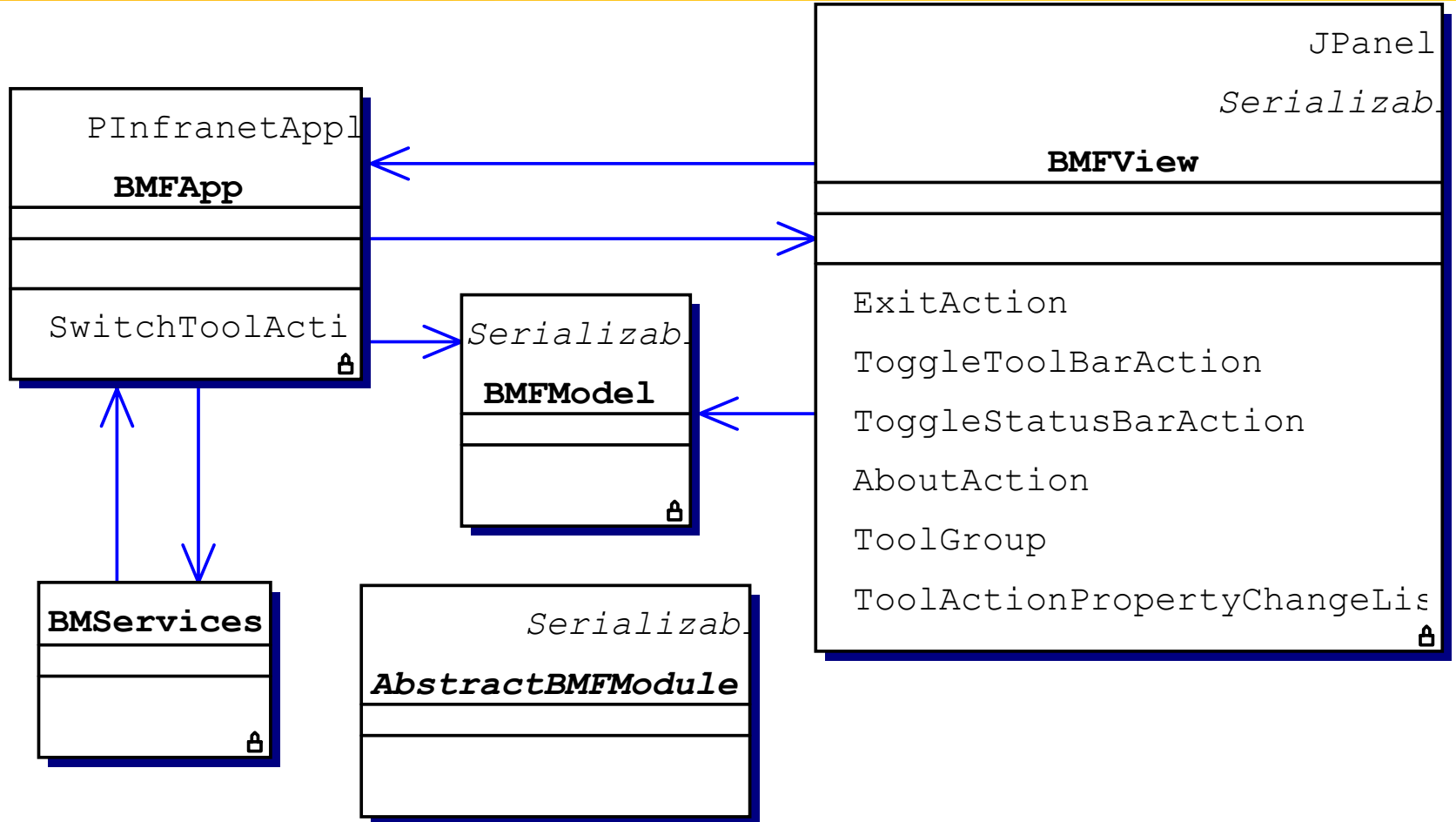
- The framework provides one concrete application class, **BMFApp**
  - It is the heart of the framework
  - It manages all the modules (with help from a few other classes such as **BMFModel**)
  - It is not subclassed
  - It calls methods of installed modules as necessary

# I. Architectural Decision: Modules

- Each module developer must provide a concrete subclass of **AbstractBMFModule**
  - The module must return a “root view” which is a **JComponent**. Do anything in the root view
  - There are **protected** methods called by the framework when a module starts, stops, etc. Override if necessary
  - The module must return a **Vector** of actions, used by menu items
- Modules can call upon an instance of **BMServices** for utility functions



# I. Framework Architecture Based on Model-View-Controller



“Module” = “Tool”



# Concrete Application Class

## Public Methods

PInfranetApplet

**BMFApp**

```
+BMFApp
+getAppProperties:Properties
+getAppResources:ResourceBundle
+getBusinessProperties:Properties
+getBusinessResources:ResourceBundle
+getBMServices:BMServices
+getAppFrame:Container
+getFrameTitle:String
+getBMFView:BMFView
+isApplet:boolean
+init:void
+main:void
```

SwitchToolAction



- The user runs an instance of **BMFApp**
- It controls all the modules
- It has a few public methods that can be called by module developers
- It has protected methods to run the framework



# Abstract Module Class

## Public Methods

*Serializable*

### ***AbstractBMFModule***

```
+AbstractBMFModule
+setDisabled:void
+getBMServices:BMServices
+getBusinessContext:AbstractBusinessCont
+getToolTitle:String
+getName:String
+getIcon:Icon
+getShortDescription:String
+getLongDescription:String
+addModuleStateChangeListener:void
+removeModuleStateChangeListener:void
+getToolMenuBar:PMenuBar
+registerHelpButton:void
```

- Each module developer must implement this module
- There is one protected abstract method to implement
- There are many optional empty protected methods to override for customization (see next slide)





# I. Framework Architecture

## Some Module Protected Overrides

```
// initialize the module
Vector getMenuActions()
abstract JComponent getToolRootView()
boolean init()
void start()

// called during normal operation
void activate()
void deactivate()
void fireModuleStateChangeEvent(Object eventData)

// called as user wants to quit
boolean canExit()
boolean isDirty()
boolean commit()
void stop()
void exit()

// provide module status
boolean isActivated()
boolean isHidden()
boolean isDisabled()
```

```
// return menu actions
// return main view
// called after login
// start threads here
```

```
// called when activated
// called when deactivated
```

```
// OK to exit app?
// Save on switch?
// Update user data
// Stop threads, etc. here
// called as app quits
```



# I. Framework Architecture

## Some Module Protected Overrides

```
// initialize the module
Vector getMenuActions()
abstract JComponent getToolRootView()
boolean init()
void start()

// called during normal operation
void activate()
void deactivate()
void fireModuleStateChangeEvent(Object eventData)

// called as user wants to quit
boolean canExit()
boolean isDirty()
boolean commit()
void stop()
void exit()

// provide module status
boolean isActivated()
boolean isHidden()
boolean isDisabled()
```

// return menu actions  
// return main view  
// called after login  
// start threads here

// called when activated  
// called when deactivated

// OK to exit app?  
// Save on switch?  
// Update user data  
// Stop threads, etc. here  
// called as app quits

- Only one abstract method
- Developer overrides others as needed



# Demo

Framework in Action

# I. Architectural Decisions: Resources

- The application must provide two resource files that contain all user-visible strings, and various application parameters
  - Strings for About dialog
  - Menu names
  - Menu item names, and mapping to Actions
  - Runtime decisions about layout and visibility
- Each developer create part of a pair of these resource files—a design weakness

# I. Sample Module "Hello World"

```
import java.awt.Font;  
import javax.swing.*;  
import com.portal.bmf.*;
```

```
public class HelloWorld extends AbstractBMFModule {
```

```
    private JLabel mRootView;
```

```
    public HelloWorld () {  
        mRootView = new JLabel("Hello World!!!", JLabel.CENTER);  
        mRootView.setFont( new Font("SanSerif", Font.BOLD, 24));  
    }
```

```
    public JComponent getToolRootView() {  
        return mRootView;  
    }
```

```
}
```



# I. Sample Partial Resource Properties File

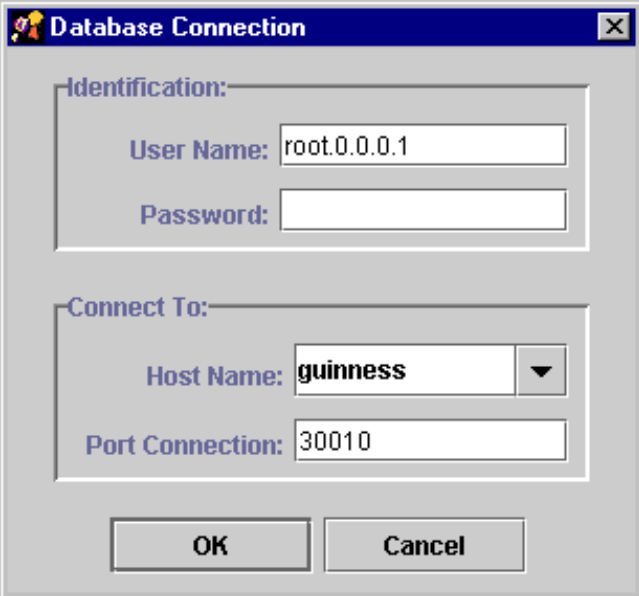
```
bmf.frame.title= Hello World
bmf.dialog.defaultTitle= Hello World
bmf.app.name= Hello World
bmf.about.image= /com/portal/bmf/images/JavaAbout.gif
bmf.about.dialogTitle=About Hello World
bmf.about.text1=Hello World, Release 6.0.2
bmf.splash.image=/com/portal/bmf/images/JavaSplash.gif
#-----BMF Modules -----
main.hello.name= Hello
main.hello.desc= Hello World
main.hello.menu= tool
main.hello.icon= /com/portal/bmf/images/Branding0.gif
main.hello.icond= /com/portal/bmf/images/Branding1.gif
#----- BMF Menus -----
bmf.menu.file = File
bmf.menu.edit = Edit
bmf.menu.view = View
bmf.menu.tool = Tools
bmf.menu.help = Help
main.open.name=  Open
main.open.desc=  Open file
main.open.menu=  file
main.open.icon=
main.open.accel=C-O
main.open.mnemonic=O
```

# I. Sample Business Properties File

```
bmf.modules= hello
bmf.startup.module= hello
resources= hello.TSTResources
bmf.toolGroups= helloGroup
bmf.toolGroup.helloGroup= hello
module.hello.class= hello.HelloWorld
bmf.startup.looknfeel= metal
bmf.hasLeftSidePanel= false
bmf.hasSplitPane= false
bmf.hasMenuBar= true
bmf.hasToolBar= false
bmf.toolBarsIsFloatable= true
bmf.hasObjectNavigationComponent= false
bmf.objectNavigationComponentIsFloatable= false
bmf.hasStatusBar= false
bmf.WIDTH=800
bmf.HEIGHT=600
```

# I. BMF Login Component

- BMF Login Component provides a connection context for the modules
- All modules can share the connection or request for separate additional connections
- You can create your own login component that logs into your server instead of Infranet



The screenshot shows a 'Database Connection' dialog box with the following fields:

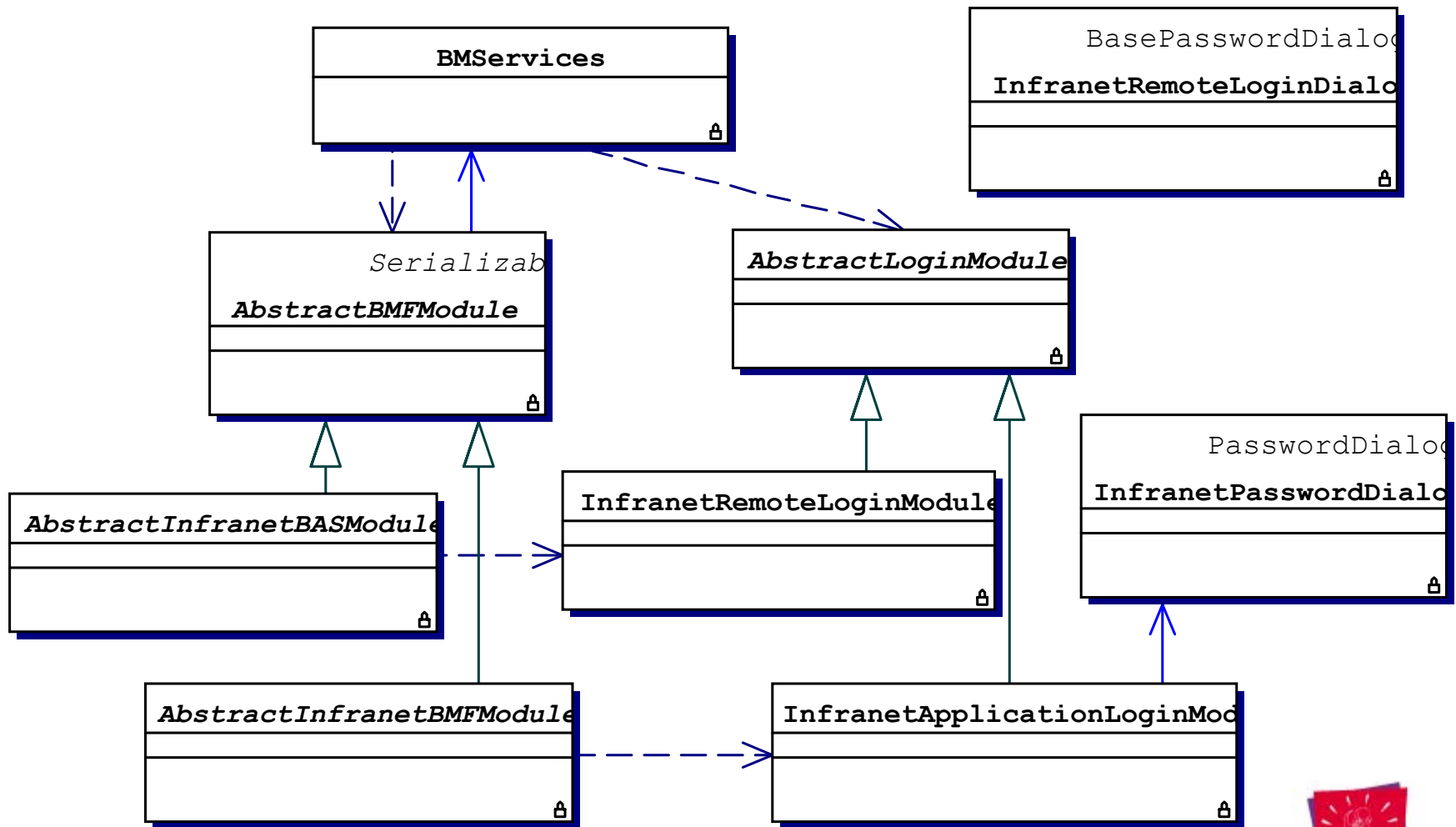
- Identification:**
  - User Name: root.0.0.0.1
  - Password: (empty)
- Connect To:**
  - Host Name: guinness (dropdown menu)
  - Port Connection: 30010

Buttons: OK, Cancel



# I. Framework Architecture

## Login Classes



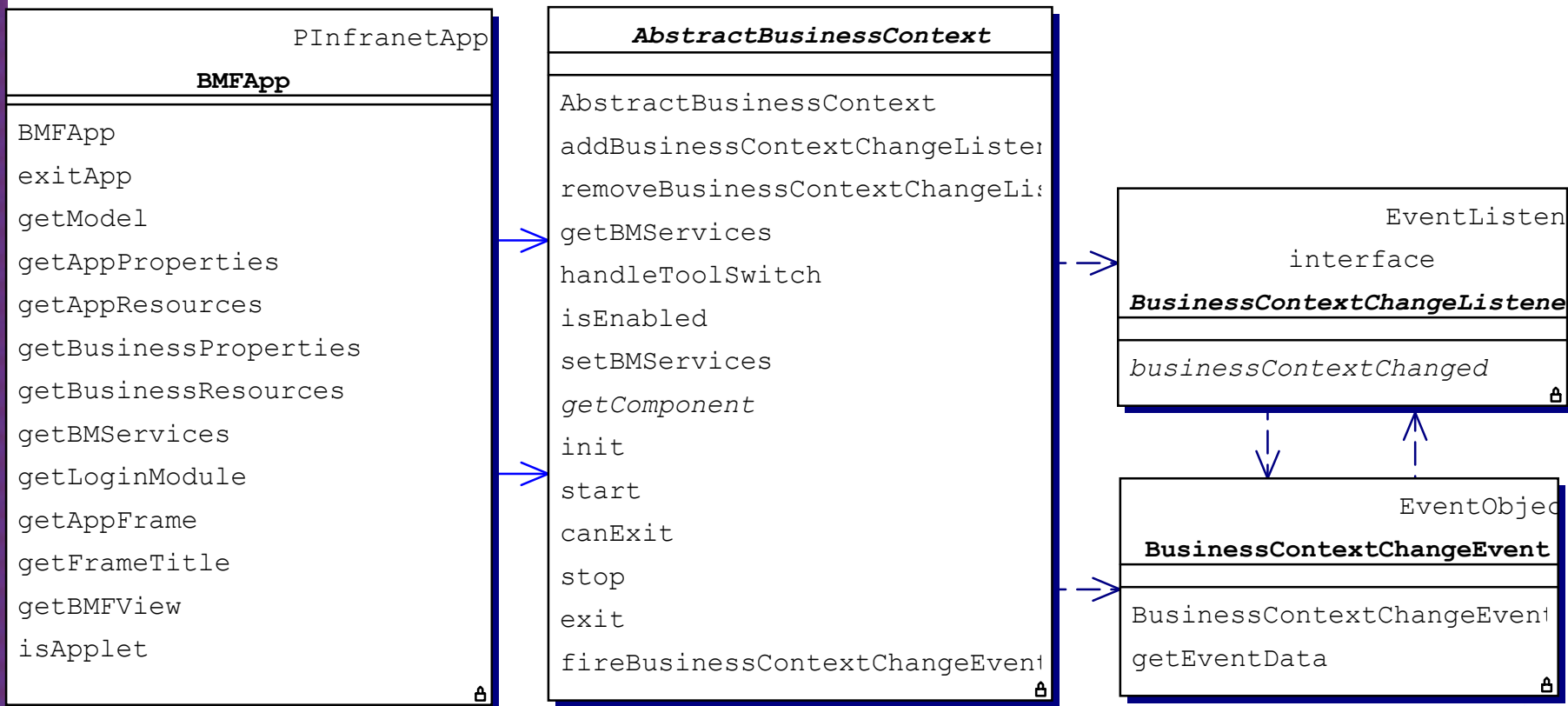
# I. Example Implementation

## Code BMFApp

```
void setActiveTool(AbstractBMFModule newActiveTool) {  
    if( newActiveTool == activeTool) {  
        return;  
    }  
    if( activeTool != null) {  
        activeTool.setActivated( false);  
    }  
    activeTool = newActiveTool;  
    if( activeTool != null) {  
        activeTool.setActivated( true);  
    }  
}
```

# I. Module Communication

- Context object can be shared between modules
- Can register for notification when it is changed



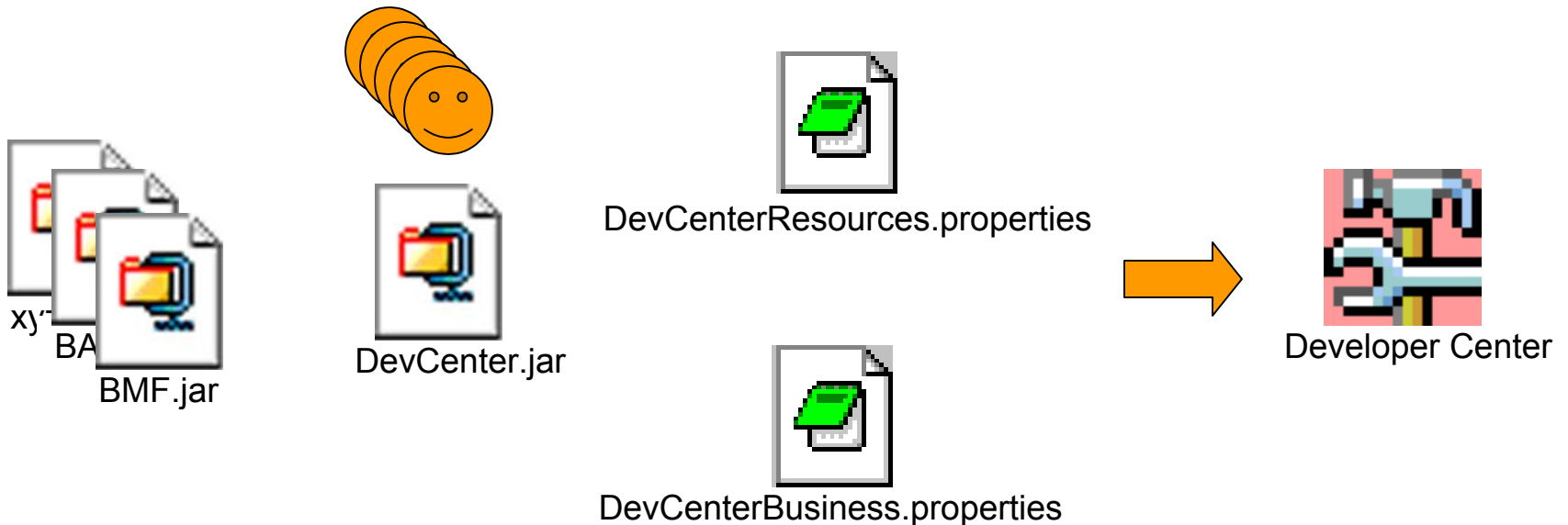
# Bad: Monolithic Application Development

Library

Application

Data

Application



# Good: Modular Development and Applications Assembly

Framework

Modules

Data

Application



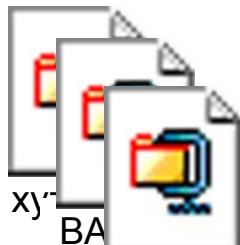
mod1.jar



DevCenterResources.properties



Developer Center



xy  
BA

BMF.jar



module5.jar



DevCenterBusiness.properties



OpBench.jar

# I. User Lessons Learned

- Icon toolbar takes up too much real estate
  - Need better ways for users to display and select from available modules
- Users may need to see more than one module at a time
- Different modules may need to login to different servers

# I. Developer Lessons Learned

- Need easy way to keep separate modules synchronized
- Need better way to combine resources from multiple modules
- Be nice to select the modules that users can see based on their names and/or roles
  - Dynamic applications assembly
- Web access important but applets not necessary
  - Use Java WebStart™ instead



# II. Portal Framework, for the Java™ Platform: Work In-Progress

- We shipped successful products with the BMF system
- Why design the new Portal Framework system?
  - Build a better foundation with a better architecture
  - Merge existing frameworks and libraries
  - Refactor existing code
  - Fix some limitations
  - Add some new features





# II. Portal Framework, for the Java Platform: Merging Existing Libraries

**U.S.**

BMF

BAS

Java Common

App Common

Controls

7. Applications

6. App Modules

5. App Frame

4. GUI Classes

3. Infranet Classes

2. Foundation Classes

1. Third Party Products

**Germany**

JSA

JavaBase 2

JavaBase 1

JavaBase 0



# II. Portal Framework, for the Java Platform: Why Seven Layers?

7. Applications	Apps (Centers)
6. App Modules	Modules
5. App Frame	Container for modules
4. GUI Classes	Widgets (big or small)
3. Infranet Classes	Domain-specific, Non-GUI
2. Foundation Classes	Non-GUI; Non-domain
1. Third-Party Products	Version mgt of JDK, etc.



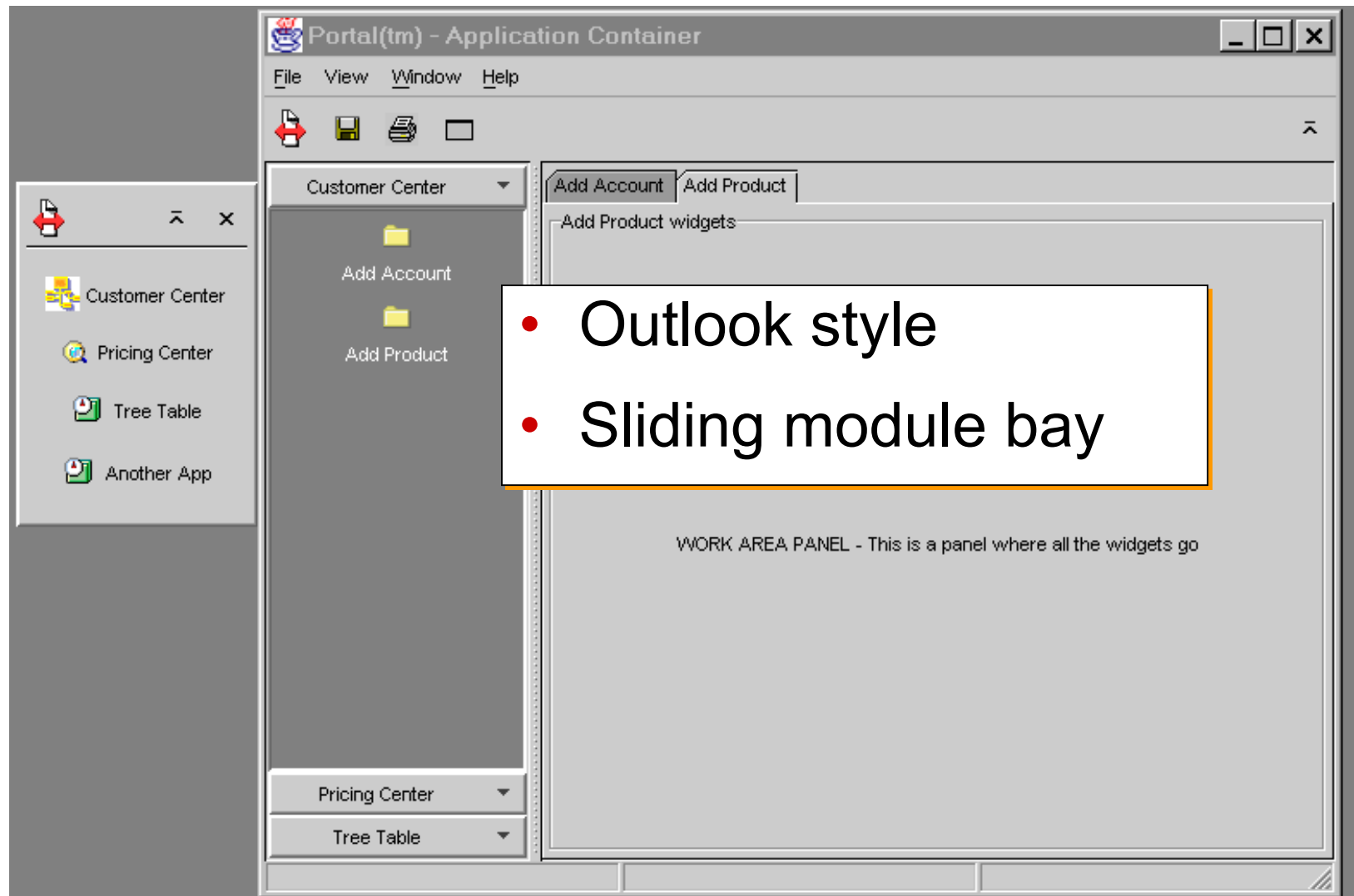
# ii. Portal Framework, for the Java Platform: One-way Dependencies



- Two-way, cross-dependencies are a maintenance nightmare



# II. Portal Framework Prototype



# Demo

Portal Java Framework  
Early Prototype

# Documentation

The screenshot shows a web browser window titled "solution42 JavaBase - Microsoft Internet Explorer". The page content is organized into several sections:

- Header:** "solution42 JavaBase" with a logo of a red cup and saucer. Below the title is the text "If you've any suggestions, please contact me."
- Help:** A section with a green background containing links for "About this site" and "Java Software Organization Scheme".
- Links:** A section with a green background listing various resources like "Russel", "Pathfinder", "portal.com", "solution42.de", "Zarquon-CVS", "DocBase", "java.sun.com", "Java Developer Connection", "Java World", and "Java Report Online".
- Search:** A section with a green background for "DejaNews search (now acquired by google)". It includes a search box and a "Search" button. Below it, it says "Search all entered keywords in \*comp.lang.java\*:".
- Search Flexibility:** A section with a green background suggesting "DejaNews PowerSearch" or "groups.google.com".
- Java Bug Database Search:** A section with a green background for searching by keyword(s) or bug id, including a search box and a "Search" button.
- Literature:** A section with a green background listing "Java-Tutorial", "Go To Java 2", "Thinking in Java", "Java Language Specification: 2nd edition", "1st edition", "ACM Articles", "Design Pattern Training", "HTML Express", and "Java for C++ Professionals Training".
- Downloads:** A section with a green background listing "Installer", "Style Report", "Optimizelt 3.0.2 und 3.1", "JDK 1.3.1", "JDK 1.3: Windows, Solaris, Linux", "JDK 1.2.2: Windows, Linux, JRE 1.2.2", "JDK 1.1.8: Windows, IBM JDK/JRE", and "Emacs 20.4.1 · JBuilder 3.5".
- Layer 4 (Applications & Customer Products):** A blue section listing "integRate (CVS)", "aggreGate (CVS)", "simuLate n-port (CVS)", and "b-c-mediate".
- Layer 3 (JSA):** A blue section listing "JSA-Homepage (CVS)".
- Layer 2 (Tools, APIs, Frameworks):** A blue section listing "RegEdit (CVS)", "LogView (CVS)", "GCP (CVS)", and "TaskManager (ClientCVS ServerCVS)".
- Layer 1 (JB/One):** A blue section listing "JB1-Homepage (CVS)".
- Layer 0 (Third-party software):** A blue section listing "JB0-Homepage (CVS)".
- Recent changes:** An orange section listing updates from May 21, 2001, to April 5, 2000, including "JDK 1.3.1 added", "JDK 1.2 removed", "Fixed all broken links, added GCP", "HTML Express Training Documents", "SUN JDK 1.3 for Linux", "JDK 1.3 for Solaris", "IBM JDK 1.3 for Linux", "JDK 1.3", "JDK 1.3 RC3", "JBuilder 3.5 Foundation", "Style Report to Layer 0", and "Style Report".
- Footer:** A note: "For less recent modifications, visit the [archive](#)".



# Docs

**JavaBase Layer 1 - Top Level Documentation**

**Documentation**

- [API documentation](#)
- [Guide to Features](#)
- [Feature Requests and Known Bugs](#)
- [How To Build](#)

**Shipments**

B	Release	Shipment Date	Build Info	Binaries	Release Notes	Summary
	<b>1.21</b>	Not yet scheduled	Not yet available	Not yet available	<a href="#">relnotes.html</a>	..
	<b>1.20</b>	Nov 28, 2001	<a href="#">buildinfo.txt</a>	Release: <a href="#">jbl1.jar</a> Debug: <a href="#">jbl1debug.jar</a> CORBA-Services: <a href="#">jbl1_javaid12.jar</a> <a href="#">jbl1_javaid13.jar</a> <a href="#">jbl1_javaid14.jar</a> <a href="#">jbl1_orbacus332.jar</a> <a href="#">jbl1_orbacus4.jar</a> <a href="#">jbl1_orbiz2000.jar</a> <a href="#">jbl1_jacorb.jar</a> Source: <a href="#">src.jar</a> API-Docs: <a href="#">docs.jar</a>	<a href="#">relnotes.html</a>	Performance improvements, changes in test package and JUnit compatibility, ORBACUS 4.0.x support.
	<b>1.19</b>	Oct 31, 2001	<a href="#">buildinfo.txt</a>	Release: <a href="#">jbl1.jar</a> Debug: <a href="#">jbl1debug.jar</a> CORBA-Services: <a href="#">jbl1_javaid12.jar</a> <a href="#">jbl1_javaid13.jar</a>	<a href="#">relnotes.html</a>	Improved <code>PrinzeKit</code> , enhanced testing.

# Summary

- Develop pieces of software as modules
- Use a framework to support combining related modules into application “Centers”
- Make behavior data-driven to minimize the need for subclassing and overriding to change simple behavior
- Minimize abstract methods
  - Provide reasonable default behavior instead
- Provide simple UI mechanisms for users to activate a module, and to switch between modules





# If You Only Remember One Thing...

Do **not** build large, monolithic applications

Do **not** build tons of separate small applications

Build functional modules and combine related ones into “centers” for specific activities



# Q&A



**JavaOne**<sup>SM</sup>

Sun's 2002 Worldwide Java Developer Conference™

**BEYOND**  
**BOUNDARIES**