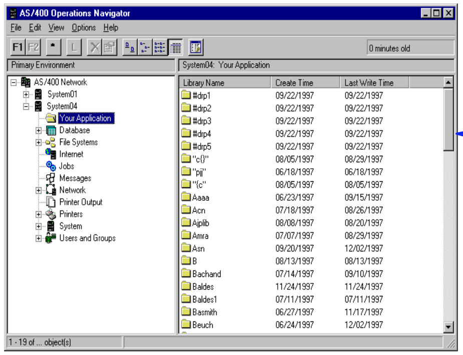
Data Beans and Button Handlers

- Overview
  - Panel Manager: engine which displays dialogs
  - Data Beans: classes which transfer data to and from the dialogs
  - Button Handlers: classes which respond to and process a pressed button
  - Formatters: check the data input by the user

Unity Console with Java Plug-ins

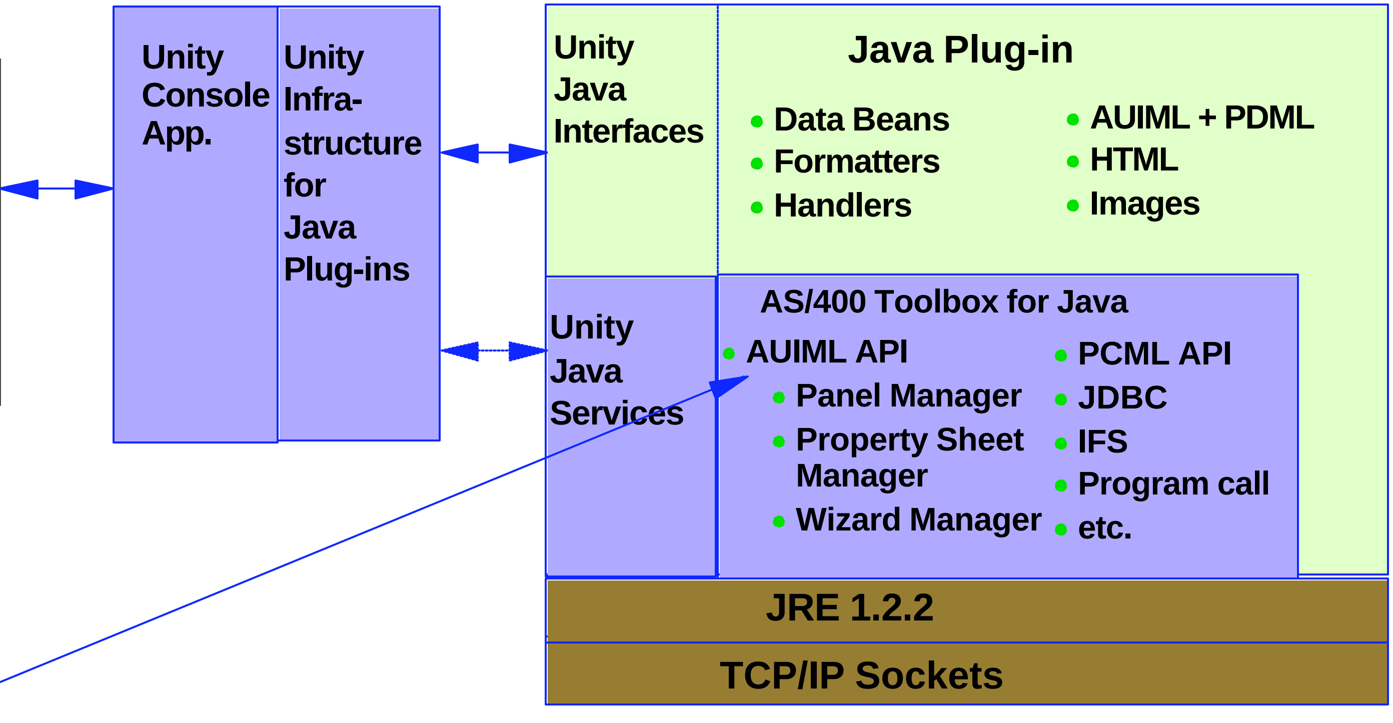
Unity Console



System04: Your Application

Library Name	Create Time	Last Write Time
#drp1	09/22/1997	09/22/1997
#drp2	09/22/1997	09/22/1997
#drp3	09/22/1997	09/22/1997
#drp4	09/22/1997	09/22/1997
#drp5	09/22/1997	09/22/1997
"c{}"	08/05/1997	08/29/1997
"pij"	06/18/1997	06/18/1997
"{c"	08/05/1997	08/05/1997
Aaaa	06/23/1997	09/15/1997
Acn	07/18/1997	08/26/1997
Ajplib	08/08/1997	08/20/1997
Amra	07/07/1997	08/29/1997
Asn	09/20/1997	12/02/1997
B	08/13/1997	08/13/1997
Bachand	07/14/1997	09/10/1997
Baldes	11/24/1997	11/24/1997
Baldes1	07/11/1997	07/11/1997
Basmith	06/27/1997	11/17/1997
Beuch	06/24/1997	12/02/1997

1 - 19 of ... object(s)


Unity Console App.

Unity Infra-structure for Java Plug-ins

Unity Java Interfaces

Java Plug-in

- Data Beans
- Formatters
- Handlers
- AUIML + PDML
- HTML
- Images

Unity Java Services

AS/400 Toolbox for Java

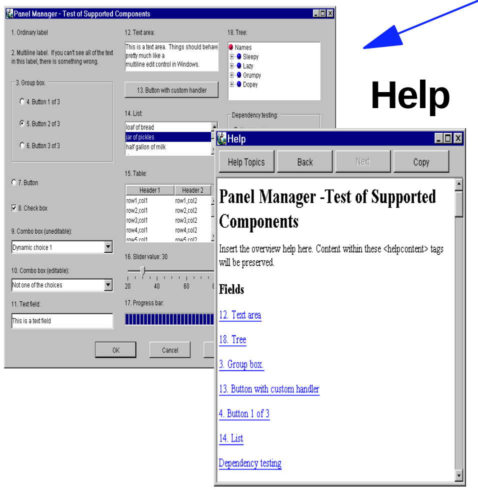
- AUIML API
  - Panel Manager
  - Property Sheet Manager
  - Wizard Manager
- PCML API
- JDBC
- IFS
- Program call
- etc.

JRE 1.2.2

TCP/IP Sockets

Panels

Help



1. Ordinary label

2. Multiline label. If you can't see all of the text in this label, there is something wrong.

3. Group box.

○ 4. Button 1 of 3

● 5. Button 2 of 3

○ 6. Button 3 of 3

○ 7. Button

☑ 8. Check box

9. Combo box (uneditable): Dynamic choice 1

10. Combo box (editable): Not one of the choices

11. Text field: This is a text field

12. Text area: This is a text area. Things should behave pretty much like a multiline edit control in Windows.

13. Button with custom handler

14. List: loaf of bread, jar of pickles, half gallon of milk

15. Table:

Header 1	Header 2
row1,col1	row1,col2
row2,col1	row2,col2
row3,col1	row3,col2
row4,col1	row4,col2
row5,col1	row5,col2

16. Slider value: 30

20 40 60

17. Progress bar:

18. Tree: Names, Sleepy, Lazy, Grumpy, Dopey

Dependency testing:

Panel Manager -Test of Supported Components

Insert the overview help here. Content within these <helpcontent> tags will be preserved.

Fields

12. Text area

18. Tree

3. Group box.

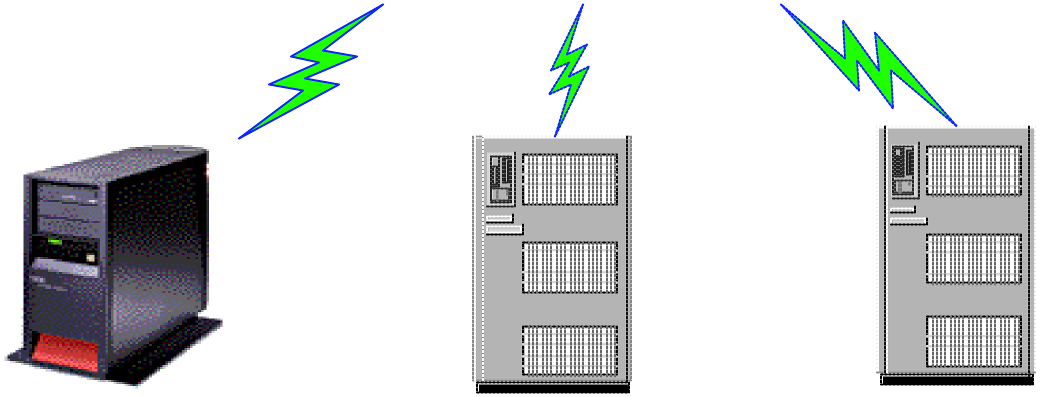
13. Button with custom handler

4. Button 1 of 3

14. List

Dependency testing

Created with the GUIBuilder



AS/400 S/390 RS6000

Java Connections (Server Jobs)

Panel Manager

- Engine which displays dialogs
- Used for common dialogs
- Base engine for other style dialog managers
  - DeckPaneManager
  - PaneManager
  - SplitPaneManager
  - TabbedPaneManager
  - PropertySheetManager
  - WizardManager

Panel Manager Usage

- Parameters
  - PDML file (*.pdml* or *pdml.ser* extension assumed)
    - treated as a resource name found using classpath and package
  - Dialog identifier
  - Databean array
  - Frame (optional - for modal dialogs)
- Common methods
  - setVisible - makes visible (modal or modeless)
  - setExitOnClose - causes dialog to go away on X (only used for applications)

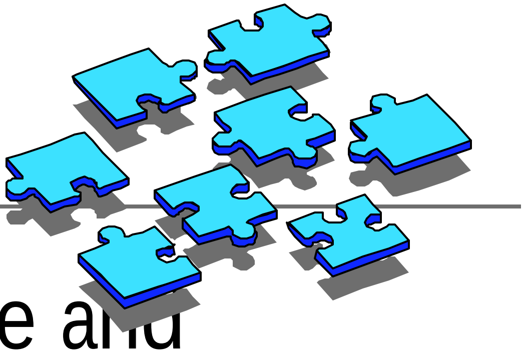
Panel Manager Example

```
public void Main(String[] args)
{
   MyDataBean dataBean = new MyDataBean( );
   dataBean.load( );

   DataBean[ ] dataBeans = { dataBean };

   PanelManager pm = null;
   try
   {
      pm = new PanelManager("MySample",   "MY_DIALOG",  dataBeans);
   }
   catch (DisplayManagerException e) { ... }

   pm.setExitOnClose(true);  // Only used in applications.
   pm.setVisible(true);
 }
```

Java System.exit Method

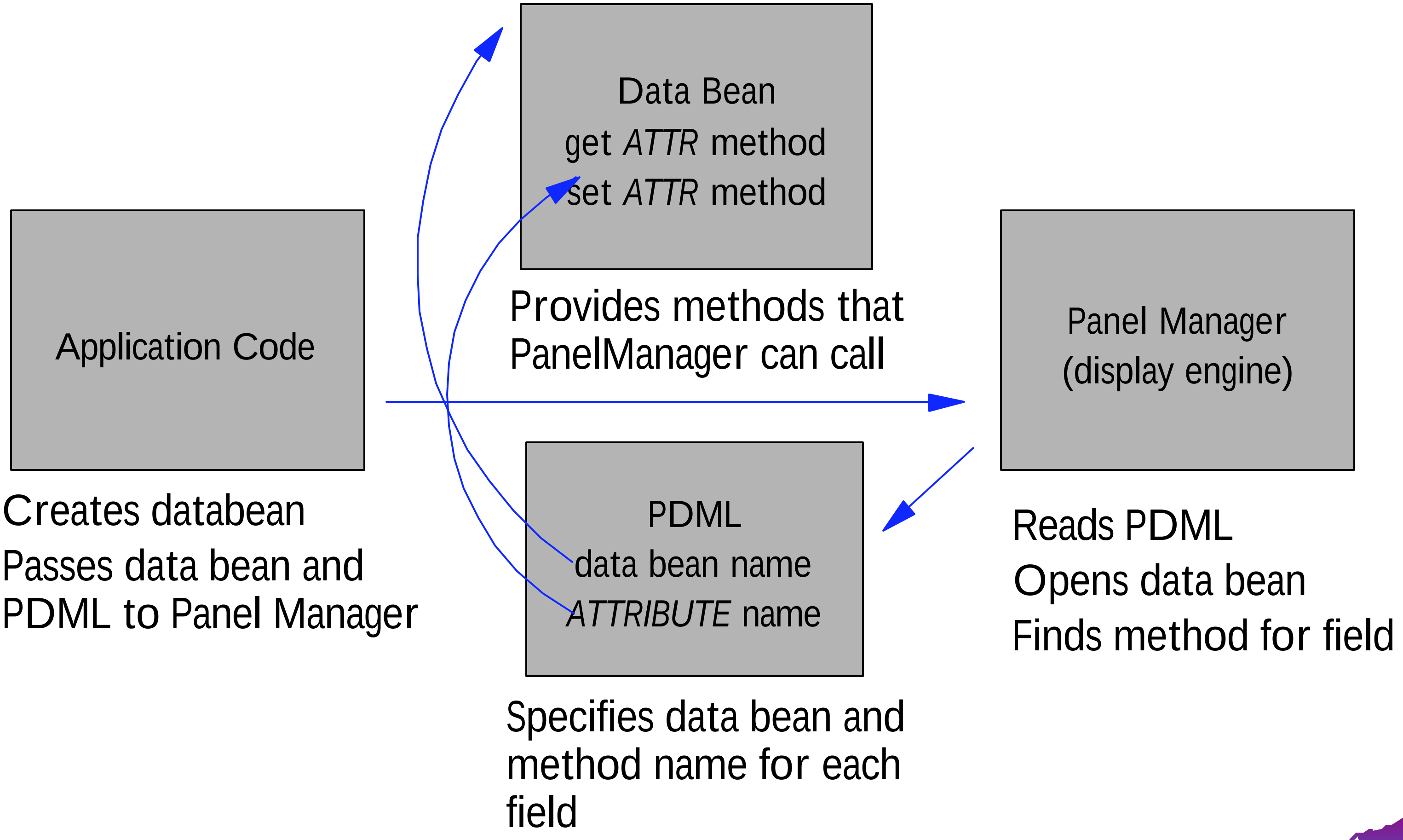
- System.exit terminates the Java Virtual Machine
- Useful when your application is called by operating system
  - Example used SystemExitOnClose
  - Example handled errors by calling SystemExit
- Deathly if used in the Unity Console
  - OpNav has single JVM
  - System.exit call terminates ***all*** java services
  - Exceptions are correct way to return errors from called programs

Data Beans



- A class which implements an interface and a set of pre-defined methods.
- Data beans handle all transfer of data to and from the GUI created by GUI builder.
- Each field with values will have a databean method to handle data transfer.
- The data bean and the handler method are specified in the PDML file.

Data Bean Process



Application Code

Creates databean

Passes data bean and PDML to Panel Manager

Data Bean

get *ATTR* method

set *ATTR* method

Provides methods that PanelManager can call

PDML

data bean name

ATTRIBUTE name

Specifies data bean and method name for each field

Panel Manager

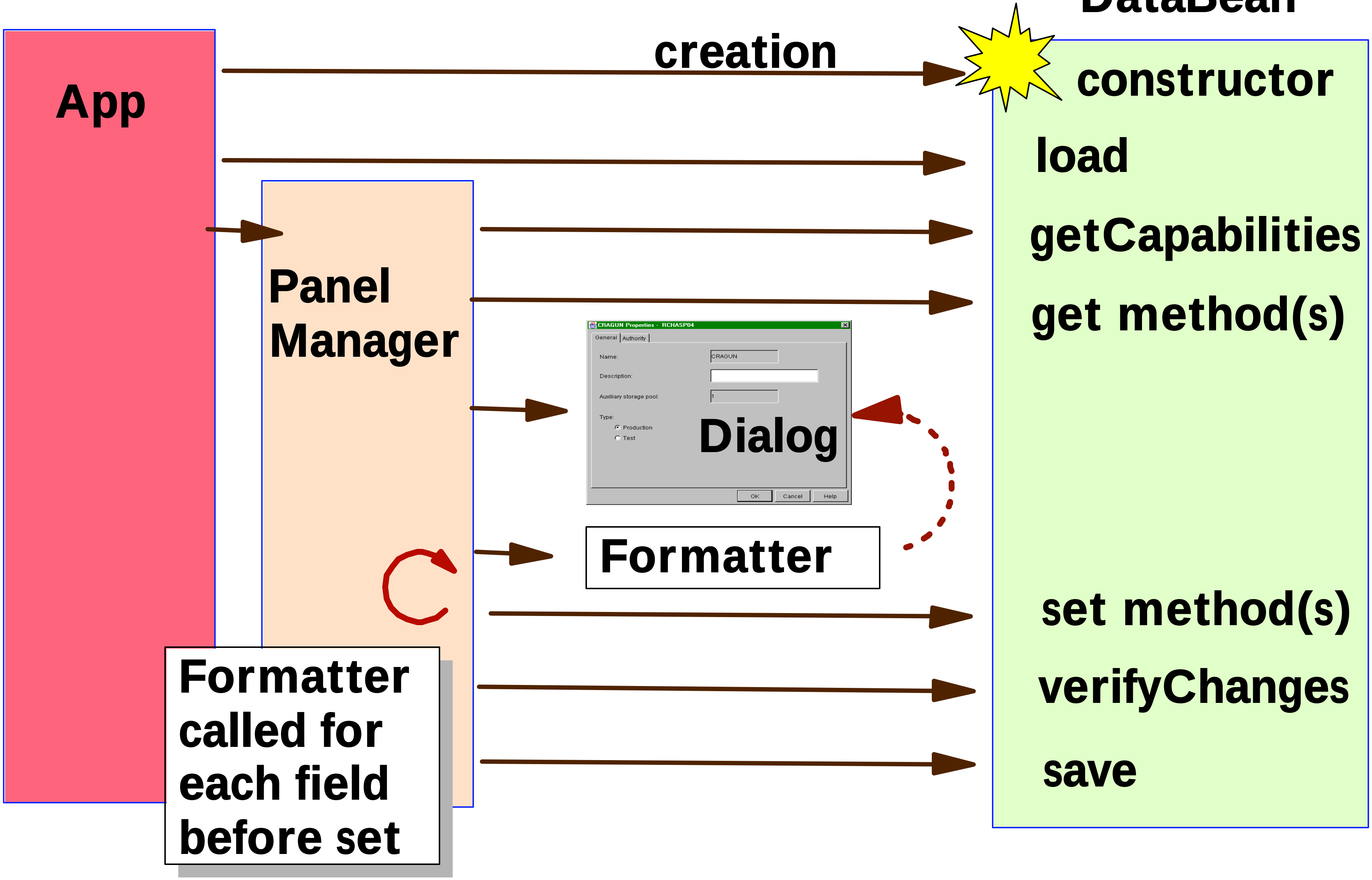
(display engine)

Reads PDML

Opens data bean

Finds method for field

Life of a DataBean



Formatter called for each field before set

CRAGUN Properties - RCHASP04

Name: CRAGUN

Description:

Auxiliary storage pool: 1

Type:

● Production

○ Test

Data Bean Data

- Where does the data bean get the data?
- What does the data bean do with the data?
- Can hard code the value, get it locally, or obtain the value from an external source i.e. Database server
- Can store the data for access by other handlers or beans

Data Bean Methods

- **set** and **get** method for each attribute
- Appropriate parameter types for each GUI object

single field - text, numeric	String
flag , button ,	Boolean
List	get*ATTR*List get*ATTR*Selection array ItemDescriptor[]
Selectable List	get*ATTR*Choices array ChoiceDescriptor[]
Radio button group	String - the selected button in group

Other Data Bean Methods

- **load** -- initializes the bean
  - called before handing off to Panel manager
  - Leaves object ready to return data on get/set methods
- **save** -- called after the sets
  - When OK (COMMIT) selected
  - Should save all changes to object from get/set
- **verifyChanges** - check before commit
  - Validates any changes from get/set
  - Called just prior to save to allow checking

One More Data Bean Method

- **getCapabilities**
  - Returns a Capabilities object identifying unsupported attributes and a handler for each
  - Reasons for unsupported attributes:
    - Not supported by server
    - Not supported by operating system version
    - Current user does not have authorization
  - Capabilities object
    - lists of HandlerTasks to have framework perform
    - ex. unsupported field is hidden or read only

Data Bean Example

- MsgQueuesSample1 uses MqNewMessageBean
- addMessage method in MqActionsManager creates a new MqNewMessageBean
- The data bean is loaded into the DataBean array
- The DataBean array is passed along with the PDML file as Panel Manager is invoked.

Data Bean Example (code)

```
MqNewMessageBean msgBean = new MqNewMessageBean(server, m_owner);
msgBean.load();
DataBean[ ] dataBeans = { msgBean };   // Set up to pass to PanelManager

PanelManager pm = null;    // Create the panel
try  { pm = new PanelManager("com.ibm.as400.opnav.MsgQueueSample1.MessageQueueGUI",
                         "IDD_MSGQ_ADD",
                         dataBeans,
                         m_owner);       }
catch (DisplayManagerException e)       { ... }

pm.setVisible(true);  // Display the panel (we give up control here)

if (!msgBean.m_actionPerformed)  // If no new message was created, simply return
    return;

// Refresh the list view to show the new added message
try { new UIServices().refreshList(m_owner, m_loader.getString("message.text.newmsg")); }
catch (UIServicesException e)       {  ...   }
```

Data Bean Example (con't)

- MessageQueueGUI.pdml specifies the data bean and the attribute in the data bean for each field.
- NewMessagesBean provides the methods specified in MessageQueueGUI.pdml
- The java engine displays the dialog and calls the specified method as needed.

Data Bean Example (pdml)

```
<PANEL name="IDD_MSGQ_ADD">
    <TITLE>IDD_MSGQ_ADD</TITLE>
    <SIZE>337,328</SIZE>
    <TEXTAREA name="IDC_MSGQ_MESSAGE" editable="yes" disabled="no">
        <TITLE>IDC_MSGQ_MESSAGE</TITLE>
        <LOCATION>17,47</LOCATION>
        <SIZE>303,209</SIZE>
        <DATACLASS>com.ibm.as400.opnav.MsgQueueSample1.MqNewMessageBean</DATACLASS>
        <ATTRIBUTE>Message</ATTRIBUTE>
        <HELPALIAS>IDC_MSGQ_ADD_TEXT</HELPALIAS>
    </TEXTAREA>

</PANEL>
```

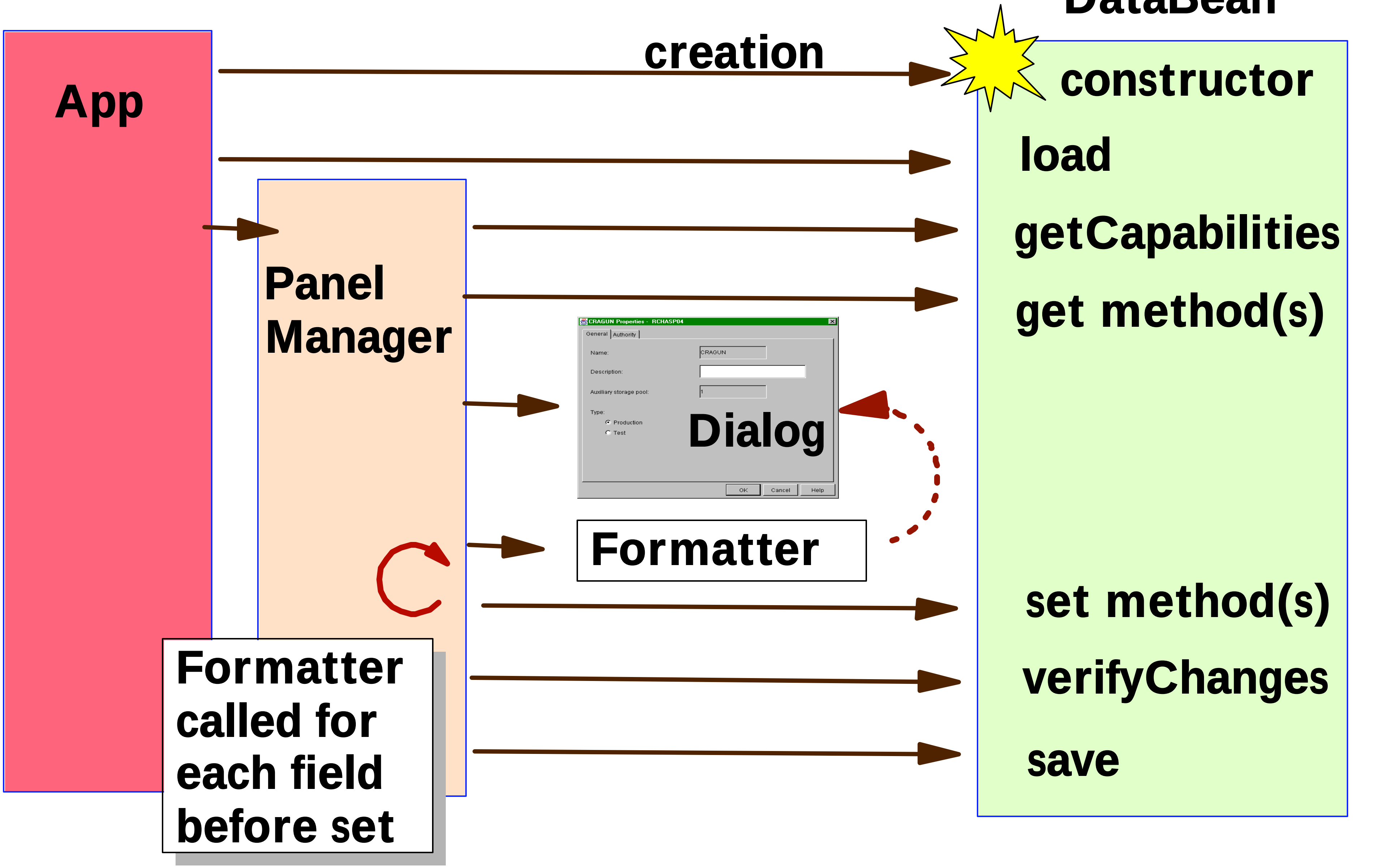
Data Bean Ex. (MqNewMessageBean)

```
public MqNewMessageBean(AS400 as400, Frame owner)
 {
    // Create the Toolbox message queue object
    m_queue = new MessageQueue(as400, MessageQueue.CURRENT);

    // Store the owning frame
    m_owner = owner;
 }

 // Initialize
 public void load()
 {
    m_sMessage = "";
 }
```

Data Bean Ex. (MqNewMessageBean - 2)

```
// Returns the Message attribute.
  public String getMessage()
  {
      return m_sMessage;
  }

   // Sets the Message attribute.
  public void setMessage(String s)
  {
      m_sMessage = s;
  }
```

Life of a DataBean



CRAGUN Properties - RCHASP04

General | Authority

Name: CRAGUN

Description:

Auxiliary storage pool: 1

Type:

● Production

○ Test

Formatter called for each field before set

Data Bean Ex. (MqNewMessageBean - 3)

```
public void save()
{
    // Send the new message to the message queue
    try { m_queue.sendInformational(m_sMessage); }
    catch (Exception e)
    {
        Monitor.logThrowable(e);     // Log the error
        String msg = m_loader.getString("error.text.send.msg");  // Load the error message text
        Object[] args = { m_queue.getSystem().getSystemName() };    // Subst.  system name
        String usrMsg = MessageFormat.format(msg, args);            // Format the error message
        MessageBoxDialog.showMessageDialog(m_owner,                 // Display it to the user
                                           usrMsg,
                                           m_loader.getString("error.title.msgbox"),
                                           JOptionPane.WARNING_MESSAGE);
        return;
    }

    // Indicate create performed
    m_actionPerformed = true;
}
```

Example 2 - List

- List box has list contents and list selection
- Corresponding getList and getSelection methods

Example 2 - pdml

```
<TABLE name="APP_LIBRARY_TABLE" selection="multiinterval" disabled="no">
    <LOCATION>16,170</LOCATION>
    <SIZE>296,211</SIZE>
    <COLUMN primary="yes" editable="no">
        <TITLE>APP_LIBRARY_TABLE.COLUMN_1</TITLE>
        <DATACLASS>SampleApplicationDataBean</DATACLASS>
        <ATTRIBUTE>LibraryName</ATTRIBUTE>
    </COLUMN>
</TABLE>
```

Example 2 - getLibraryNameList

```
public ItemDescriptor[ ] getLibraryNameList()   // Fills the library name column
{  Vector data = new Vector();
     :
   if (m_libraryList != null)
   {  for (Enumeration e = m_libraryList.elements(); e.hasMoreElements() ; )
      {  Library lib = (Library) e.nextElement();
         String name = lib.getName();
         data.addElement(new ItemDescriptor(name, name));
      }
      // Set the first library as the selected one
      Library first = (Library) m_libraryList.firstElement();
      if (first != null)
      {  String name = first.getName();
         m_selectedLibNames = new String [ ] {name};
       }
   }
   ItemDescriptor[ ] items = new ItemDescriptor[data.size()];
   data.copyInto(items);
   return items;
}
```

Example 2 - getLibraryNameSelection

```
// Selects a Library name column initially in the list
public String[ ] getLibraryNameSelection()
{
   return m_selectedLibNames;
 }


// A selected Library name column will be returned as the users choice.
public void setLibraryNameSelection(String[ ] selection)
{
   if (selection.length > 0)
   {
      // Just take the first item selected since this a single select list
      m_selectedLibNames = selection;
   }
}
```

Example 3 - getCapabilities

```
public Capabilities getCapabilities()
{
        Capabilities cp = new Capabilities();

        HandlerTask ht = new HandlerTask(HandlerTask.DISABLE);
        String[] names = { "IDC_MYDESCRIPTION","IDC_MYFIELD" };
        ht.setComponents(names);

        HandlerTask[ ] htList = { null };
        htList[0] = ht;
        cp.setNotCapable("MyAttribute", htList);

        return cp;
}
```

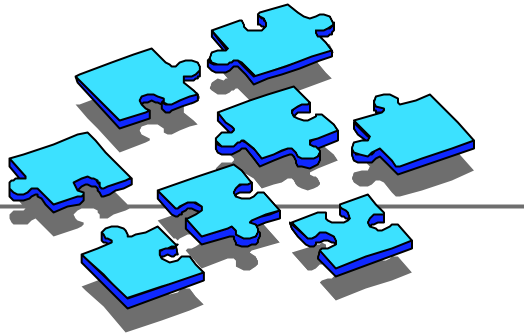
Handle by disabling

These are the fields

Capabilities is told MyAttribute is "not capable"

Note: You get column disabling for free without HandlerTasks

Button Handler



- A specialized class called when a button is clicked.
- Has access to the instance of PanelManager
- Can access data beans
- Can start up new dialogs with new instance of PanelManager

Default Button Handling

- OK - Normally set COMMIT
- Cancel - Normally set CANCEL
- Help - Normally set HELP

Ex. 4 propertiesLibraryButtonHandler

```
public PropertiesLibraryButtonHandler(PanelManager pm)
{
   super(pm);
   // Set up access to the DataBean class for the wizard this button is associated with
   DataBean[] dataBeans = pm.getDataBeans();
   // In this application, only one databean was created for the application
   if (dataBeans != null)
   {
     for (int i = 0; i < dataBeans.length; i++)
      {
         DataBean tmp = dataBeans[i];
         if (tmp instanceof SampleApplicationDataBean)
         {
            m_appBean = (SampleApplicationDataBean) tmp;
            break;
         }
      }
   }
}
```

Ex. 4 ButtonHandler: actionPerformed

```
// Note: some logic, conditionals, and error handling not shown
 public void actionPerformed(ActionEvent e)
 {
   // Determine whether this is the custom button or initial panel activation
   // See which panel this browse button call originated from
   // Get the indexes of all selected libs
   JTable table = (JTable) m_pm.getComponent(LIBRARY_TABLE);
   int [] selectedLibs = table.getSelectedRows();

   LibraryListVector libList =  m_appBean.getLibraryList();
   for (int i=0; i < selectedLibs.length; i++)  // For each selected library
   {
     int index = selectedLibs[i];
     Library lib = (Library) libList.elementAt(index);  // Get the selected library object
     lib.getAttributesFromAS400();    // Get attributes of this library from AS/400
     showLibraryPropertyPage(lib);   // Show the properties of library
   }  // endfor

   // Update and redisplay the library list if needed, or return.
 }
```

Ex. 4 ButtonHandler: showPage

```
public boolean showLibraryPropertyPage(Library lib)
{
  // Instantiate the libray property panel data bean
  m_dataBean = new PropertiesLibraryDataBean(lib);
  m_dataBean.load();
  DataBean[ ] dataBeans = { m_dataBean };

  // Get the application frame
  Frame win = new Frame();

  // Create the property sheet
  PropertySheetManager psm = null;
  try {  psm = new PropertySheetManager("LibrariesSample",
    "IDD_LIBRARY_PROPERTIES",
    dataBeans,
    win);    }   // Providing the frame makes this modal
  catch (DisplayManagerException e)
  { // display error }
```

Ex. 4 ButtonHandler: showPage

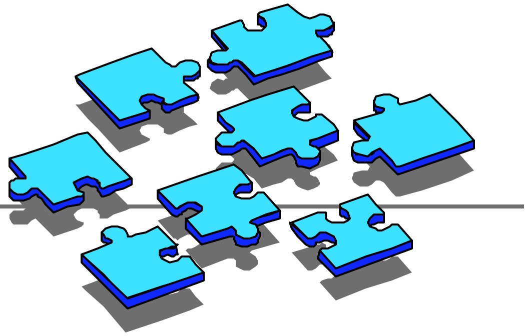
```
// continued
// Set the title of the property sheet
  String s = m_loader.getString("properties_title");
  Object[] args = { lib.getName(), lib.getSystemName() };
   String title = MessageFormat.format(s, args);
  psm.setTitle(title);

  // All text fields that need to have automatic checking for
  // valid AS/400 object names need to have a formatter object
  DataFormatter myFormatter= new AS400NameFormatter(lib.getSystemObject());
  boolean requiredField = true;

  psm.setFormatter("IDD_LIB_PROP_AUTHORITY. IDC_LIB_PROP_AUT_LIST_NAME",
  myFormatter,
  requiredField);
  // Display the panel, wait for result
  psm.setVisible(true);
  // See if the user pressed the OK button
  return m_dataBean.getButtonStateOK();
}
```

Load the resource, load the arguments, format the message

Subpanel notation for property sheets

Equivalent to DoModal because of Frame

Formatters



- What is a formatter?
- Checks input to be in a specific format
- Getting a formatter
  - Several built-in formats
  - Several packaged formats
  - Write your own formatter

Using a Built-in Formatter

- Integer
- Date
- Time
- Internet Address
- Percent

```
<TEXTFIELD name="IDC_TEXTFIELD">
    <LOCATION>7,254</LOCATION>
    <SIZE>124,14</SIZE>
    <DATACLASS>TestDataBean1</DATACLASS>
    <ATTRIBUTE>TextField</ATTRIBUTE>
    <FORMAT>PERCENT</FORMAT>
```

Using a Packaged or Custom Formatter

- Formatter without parameters

```
<TEXTFIELD name="IDC_TEXTFIELD">
<LOCATION>7,254</LOCATION>
<SIZE>124,14</SIZE>
<DATACLASS>TestDataBean1</DATACLASS>
<ATTRIBUTE>TextField</ATTRIBUTE>
<FORMAT>com.ibm.as400.MyPackage.MyFormatter</FORMAT>
```

Using a Packaged or Custom Formatter

- Formatters with parameters

```
      try  {   pm = new PanelManager("EduSample",
         "SYS_VAL_USER",  dataBeans);
      }
      catch (DisplayManagerException err) { ...}
:
 AS400 as400 = new AS400 (m_appBean.getAS400Name());
 DataFormatter myFormatter= new AS400NameFormatter(as400);
 boolean requiredField = true;
 pm.setFormatter("SVUSR_USR_NAME",
                         myFormatter,   requiredField);
```