UT-BATTELLE

Introducing Bricks

A Java Swing User Interface Builder

Thomas Pelaia II, Ph.D.

Physics / Application Programming

SNS SPALLATION NEUTRON SOURCE

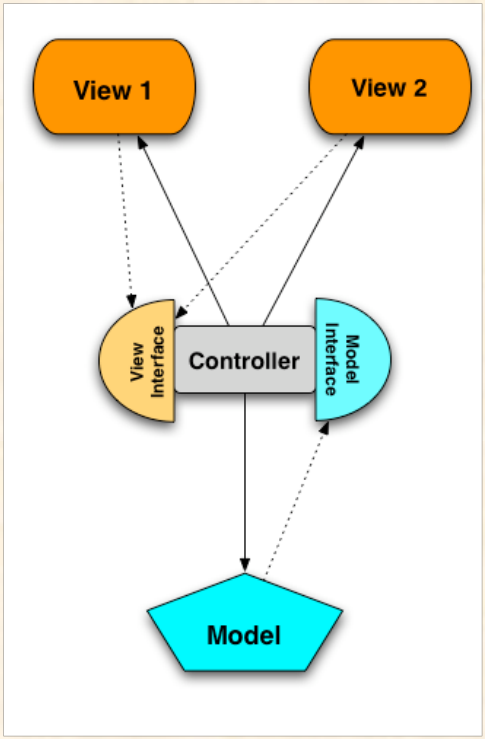
OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

Motivation

- Want to accelerate XAL application development
  - Allow developers to focus more on business logic and less on GUI source code
  - Encourage Model-View-Controller (MVC) architecture
- Problems with many popular Java GUI Builders
  - Not really MVC compliant despite claims
  - Generate lots of Java source code that you aren't supposed to edit
  - Restrict how you organize your code
  - Tie you to a particular IDE
  - Can't write your code in other languages (e.g. jython)
  - Builder doesn't always reflect what you get

Model-View-Controller (MVC)

- Model (business logic) and View code are independent of each other and are glued together by controllers
- Models and Views define listener interfaces and communicate by events
- Controllers directly access models and views and handle model and view events as necessary
- Well accepted as a good design pattern
- Allows for applications to scale in complexity
- Rarely implemented well



XAL Applications

- Currently over forty applications written
- Framework provides a common look and feel with a functioning, familiar menu bar and a document driven foundation for rapid development of applications
- Developers still had to write Java code for the user interface
- Use Jython for quick scripts which typically don't have a user interface

Jython side note

- Jython is a scripting language compatible with Java
- Conforms to Python syntax and standard libraries
- Ideal for scripting and prototyping
  - Un-typed language
  - Avoids strict exception handling enforcement
  - Interpreted language (no compilation)
- 100% pure Java
- Java bridging
  - Morph between Java and Jython classes/objects and allow direct calls to Java

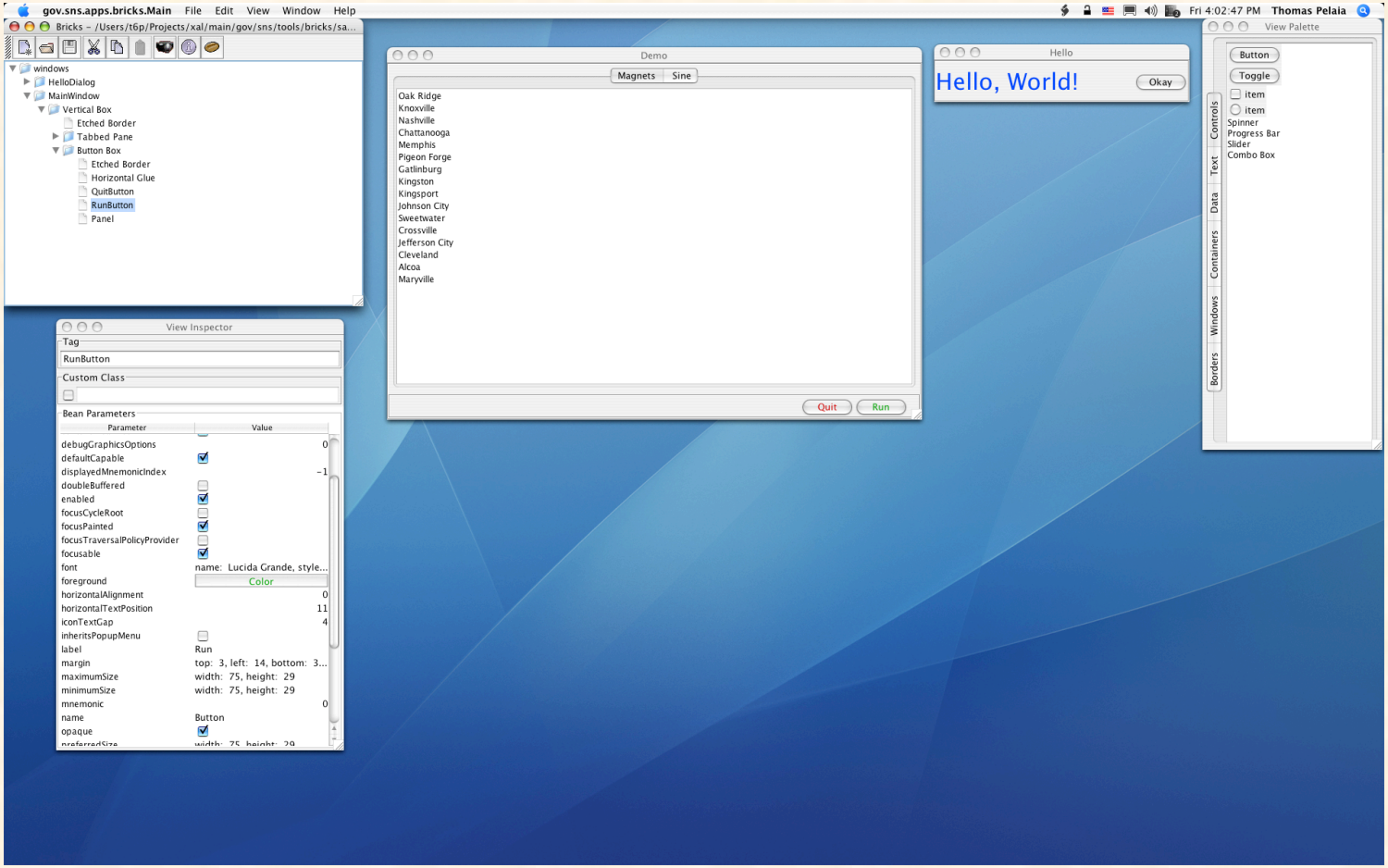
Goals

- Provide an application for rapidly creating user interfaces
- Conform to MVC
- Store the user interface definition in an XML file instead of generating source code
- Allow developers freedom to organize code as they wish
- IDE independent
- Allow applications to be written entirely in Java or Jython
- Allow extensibility
- Integrate with the application framework

Bricks Features

- 100% pure Java XAL application
  - Java 1.5 required
- IDE independent
- Conforms to MVC
- Bricks document is saved in XML format
  - Can be generated from a server
- Supports common Swing widgets
  - Allows for custom widget subclasses
- Supports cut, copy and paste for rapid development
- Java beans aware

Bricks Screenshot



Jython Example Code

/Users/t6p/Projects/xal/main/gov/sns/tools/bricks/samples/bricks_test.py

```
#!/usr/bin/env jython

# this script demonstrates how to load a windows from a bricks resource file, configure
        the components with data and display those windows

import sys

from jarray import *
from java.lang import *
from java.awt import *
from java.util import *
from java.lang.reflect import *
from javax.swing.event import *
from javax.swing import *
from java.awt.event import *
from java.net import *
from java.io import *

from gov.sns.tools.bricks import *
from gov.sns.tools.plot import *


# python Java additions
true = (1==1)
false = not true
null = None


# locate the sample folder and get the bricks file within it
sample_folder = File( sys.argv[0] ).getParentFile()
url = File( sample_folder, "test.bricks" ).toURL()
print url

# generate a window reference resource and pass the desired constructor arguments
window_ref = WindowReference( url, "MainWindow", ["Test Title"] )

# get the main window
window = window_ref.getWindow()

# get the magnet list and populate it with data
magnet_list = window_ref.getView( "MagnetList" );
magnets = Vector()
magnets.add( "Dipole" )
magnets.add( "Quadrupole" )
magnets.add( "Sextupole" )
magnets.add( "Octupole" )
magnets.add( "Skew Dipole" )
magnets.add( "Skew Quadrupole" )
magnets.add( "Skew Sextupole" )
magnet_list.setListData( magnets )

# get the plot component and add the data
plot = window_ref.getView( "SinePlot" )
graphData = BasicGraphData();
graphData.setGraphColor( Color.BLUE );
graphData.setGraphProperty( plot.getLegendKeyString(), "Sine" );
for x in range( 100 ):
```

Printed: 7/21/06 4:12 PM Page 1

/Users/t6p/Projects/xal/main/gov/sns/tools/bricks/samples/bricks_test.py

```
    graphData.addPoint( x, Math.sin( 0.1 * x ) );
series = Vector(1);
series.add( graphData );
plot.addGraphData( series );


# define the dialog Okay action handler
class DialogOkayAction( ActionListener ):
    def __init__( self, dialog ):
        self.dialog = dialog

    def actionPerformed( self, event ):
        self.dialog.setVisible( false )


# define the quit button action handler
class QuitAction( ActionListener ):
    def actionPerformed( self, event ):
        sys.exit( 0 )


# define the run action handler
class RunAction( ActionListener ):
    def actionPerformed( self, event ):
        # create the dialog box with the main window as the dialog box's owner
        dialog_ref = WindowReference( url, "HelloDialog", [window] )

        dialog = dialog_ref.getWindow()

        button = dialog_ref.getView( "OkayButton" )
        button.addActionListener( DialogOkayAction( dialog ) )

        dialog.setLocationRelativeTo( window )
        dialog.setVisible( true )


# get the run button and configure it with the run action
runButton = window_ref.getView( "RunButton" )
runButton.addActionListener( RunAction() )

# get the quit button and configure it with the quit action
quitButton = window_ref.getView( "QuitButton" )
quitButton.addActionListener( QuitAction() )

# display the main window
window.setVisible( true )
```

Printed: 7/21/06 4:12 PM Page 2

Loading the Window Resource

```
# locate the sample folder and get the bricks file within it
sample_folder = File( sys.argv[0] ).getParentFile()
url = File( sample_folder, "test.bricks" ).toURL()

# generate a window reference resource and pass the desired
constructor arguments
window_ref = WindowReference( url, "MainWindow", ["Test Title"] )

# get the main window
window = window_ref.getWindow()
```

Populating Data

```
# get the magnet list and populate it with data
magnet_list = window_ref.getView( "MagnetList" );
magnets = Vector()
magnets.add( "Dipole" )
magnets.add( "Quadrupole" )
magnets.add( "Sextupole" )
magnets.add( "Octupole" )
magnets.add( "Skew Dipole" )
magnets.add( "Skew Quadrupole" )
magnets.add( "Skew Sextupole" )
magnet_list.setListData( magnets )

# get the plot component and add the data
plot = window_ref.getView( "SinePlot" )
plot.addGraphData( series );
```

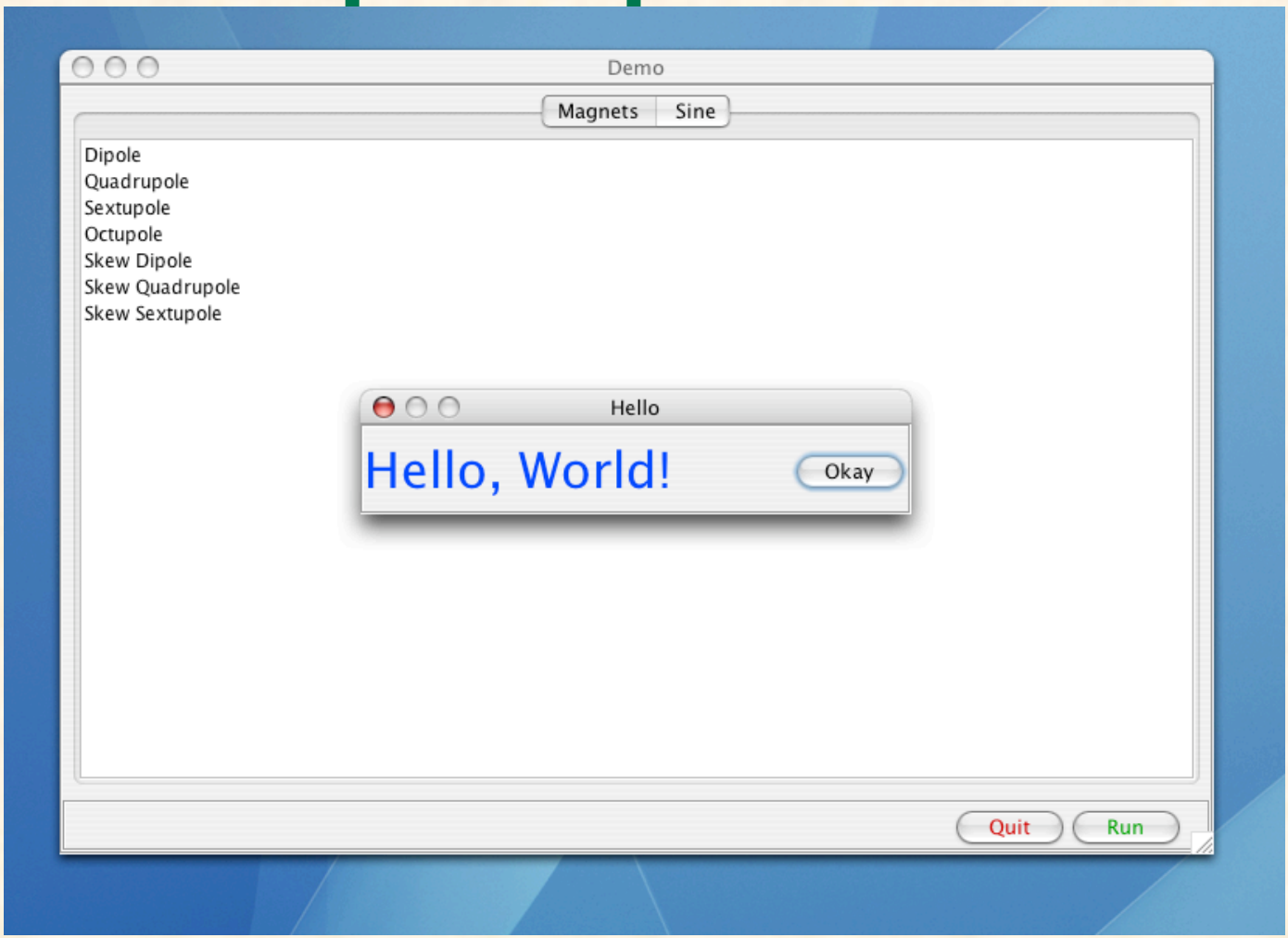
Defining Actions

```
# define the quit button action handler
class QuitAction( ActionListener ):
   def actionPerformed( self, event ):
      sys.exit( 0 )


# get the run button and configure it with the run action
runButton = window_ref.getView( "RunButton" )
runButton.addActionListener( RunAction() )

# get the quit button and configure it with the quit action
quitButton = window_ref.getView( "QuitButton" )
quitButton.addActionListener( QuitAction() )
```

Jython Sample Snapshot



XAL Application with Bricks

- Developer provides the following files
  - Application adaptor
    - Application wide properties, events
  - Document(s)
    - Read/write documents
    - Instantiate main window
    - May act as controller
  - Bricks resource file defining the user interface
  - Optional menu definition file(s)
  - Help HTML file and any supporting resources
  - About properties file

XAL Demo Document Code

/Users/t6p/Projects/xal/main/gov/sns/application/demobricks/DemoDocument.java

```java
/*
 * MyDocument.java
 *
 * Created on March 19, 2003, 1:32 PM
 */

package gov.sns.application.demobricks;

import java.net.*;
import java.io.*;
import java.awt.Color;
import javax.swing.*;
import javax.swing.JToggleButton.ToggleButtonModel;
import javax.swing.text.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.util.logging.*;
import java.util.Vector;

import gov.sns.application.*;
import gov.sns.tools.bricks.WindowReference;
import gov.sns.tools.plot.*;


/**
 * DemoDocument is a custom XalDocument for this application.
 * @author  t6p
 */
public class DemoDocument extends XalDocument {
    /** Create a new empty document */
    public DemoDocument() {
        this( null );
    }


    /**
     * Create a new document loaded from the URL file
     * @param url The URL of the file to load into the new document.
     */
    public DemoDocument( final URL url ) {
        setSource( url );
    }


    /**
     * Make a main window by instantiating the my custom window.  Set the text
     * pane to use the textDocument variable as its document.
     */
    public void makeMainWindow() {
        final WindowReference windowReference = getDefaultWindowReference( "MainWindow",
                this );
        mainWindow = (XalWindow)windowReference.getWindow();

        final JList magnetList = (JList)windowReference.getView( "MagnetList" );
        final Object[] magnets = { "Bend", "Quadrupole", "Sextupole", "Octupole", "Skew
              Quadrupole", "Vertical Corrector", "Horizontal Corrector", "Skew Dipole",
              "Skew Sextupole", "Sector Bend", "Rectangular Bend", "Chicane" };
```

Printed: 7/21/06 4:18 PM Page 1

/Users/t6p/Projects/xal/main/gov/sns/application/demobricks/DemoDocument.java

```java
        magnetList.setListData( magnets );

        final JButton runButton = (JButton)windowReference.getView( "RunButton" );
        runButton.addActionListener( new ActionListener() {
            public void actionPerformed( final ActionEvent event ) {
                showHelloDialog();
            }
        });

        final FunctionGraphsJPanel plot = (FunctionGraphsJPanel)windowReference.getView
                ( "SinePlot" );
        makeSinePlot( plot );
    }


    /** make the plot */
    protected void makeSinePlot( final FunctionGraphsJPanel plot ) {
        final BasicGraphData graphData = new BasicGraphData();
        graphData.setGraphColor( Color.BLUE );
        graphData.setGraphProperty( plot.getLegendKeyString(), "Sine" );
        for ( double x = 0 ; x < 10.0 ; x+=0.2 ) {
            graphData.addPoint( x, Math.sin( x ) );
        }

        final Vector series = new Vector(1);
        series.add( graphData );
        plot.addGraphData( series );
    }


    /** show the hello dialog */
    protected void showHelloDialog() {
        final WindowReference helloDialogRef = getDefaultWindowReference( "HelloDialog",
                mainWindow );
        final JDialog dialog = (JDialog)helloDialogRef.getWindow();

        final JButton okayButton = (JButton)helloDialogRef.getView( "OkayButton" );
        okayButton.addActionListener( new ActionListener() {
            public void actionPerformed( final ActionEvent event ) {
                dialog.setVisible( false );
            }
        });

        dialog.setLocationRelativeTo( mainWindow );
        dialog.setVisible( true );
    }


    /**
     * Save the document to the specified URL.
     * @param url The URL to which the document should be saved.
     */
    public void saveDocumentAs( final URL url ) {
    }
}
```

Printed: 7/21/06 4:18 PM Page 2

Making an XAL Window

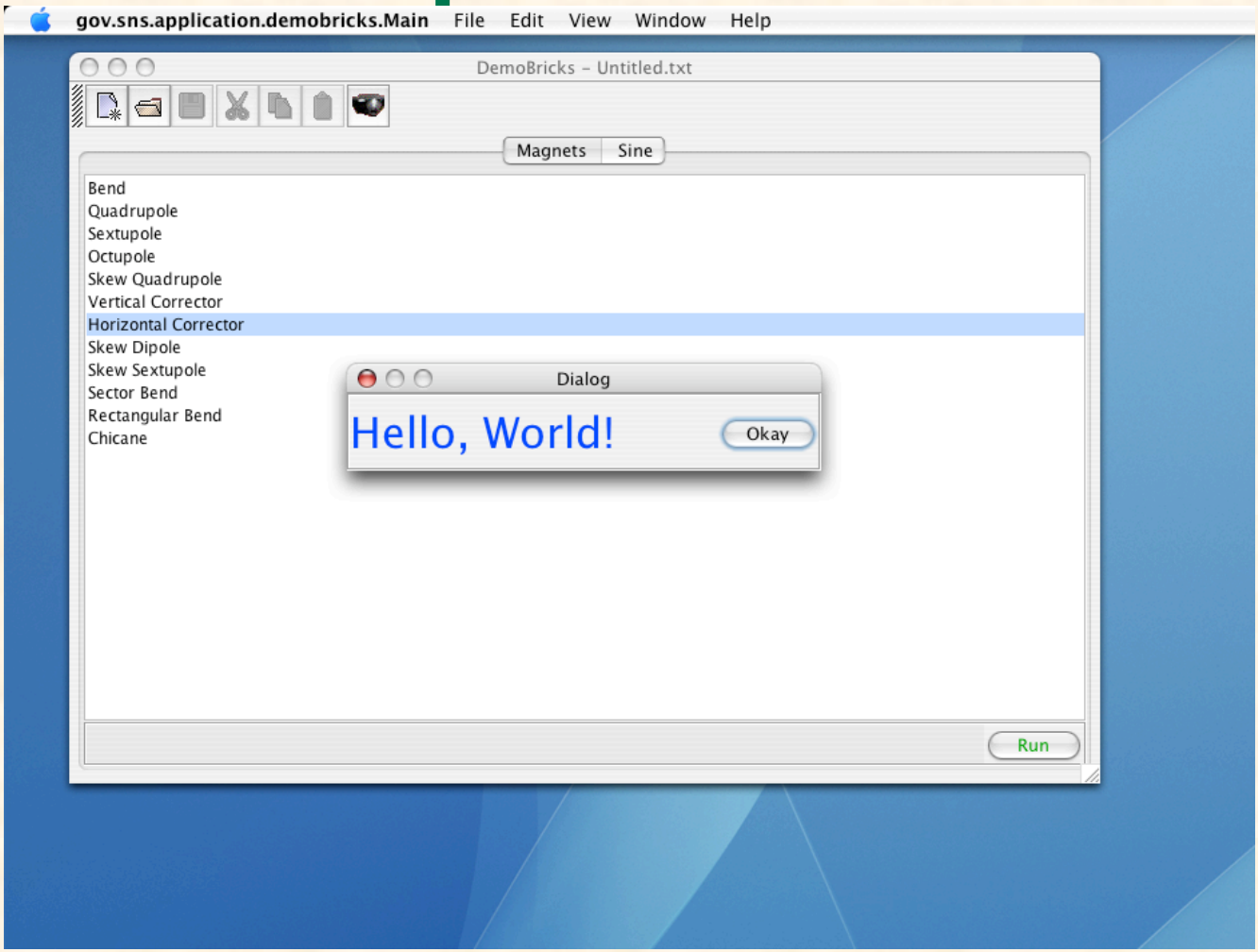
```
public void makeMainWindow() {
      final WindowReference windowReference =
         getDefaultWindowReference( "MainWindow", this );
      mainWindow = (XalWindow)windowReference.getWindow();

      final JList magnetList = (JList)windowReference.getView( "MagnetList" );
      poplulateMagnetList( magnetList );

      final JButton runButton = (JButton)windowReference.getView( "RunButton" );
      runButton.addActionListener( new ActionListener() {
         public void actionPerformed( final ActionEvent event ) {
                  showHelloDialog();
         }
      });

      final FunctionGraphsJPanel plot =
         (FunctionGraphsJPanel)windowReference.getView( "SinePlot" );
      makeSinePlot( plot );
}
```

XAL Demo Snapshot



Demo Bricks XML File


```
<?xml version = '1.0' encoding = 'UTF-8'?>
<BricksDocument date="Thu Jul 20 10:51:01 EDT 2006" version="1.8.0">
    <RootBrick>
        <ViewNode height="80" tag="HelloDialog" width="355">
            <ViewProxy type="javax.swing.JDialog"/>
            <BeanProperty name="focusableWindowState" value="true"/>
            <BeanProperty name="resizable" value="false"/>
            <BeanProperty name="modal" value="true"/>
            <BeanProperty fontName="Dialog" name="font" size="48" style="0"/>
            <ViewNode tag="Horizontal Box">
                <ViewProxy type="javax.swing.Box_Horizontal"/>
                <BorderNode tag="Etched Border">
                    <BorderProxy type="javax.swing.border.EtchedBorder"/>
                </BorderNode>
                <ViewNode tag="Label">
                    <ViewProxy type="javax.swing.JLabel"/>
                    <BeanProperty name="text" value="Hello, World!"/>
                    <BeanProperty alpha="255" blue="255" green="51" name="foreground" red="0"/>
                    <BeanProperty fontName="Dialog" name="font" size="32" style="0"/>
                </ViewNode>
                <ViewNode tag="Horizontal Glue">
                    <ViewProxy type="javax.swing.Box_HorizontalGlue"/>
                </ViewNode>
                <ViewNode tag="OkayButton">
                    <ViewProxy type="javax.swing.JButton"/>
                    <BeanProperty name="text" value="Okay"/>
                </ViewNode>
                <BorderNode tag="Etched Border">
                    <BorderProxy type="javax.swing.border.EtchedBorder"/>
                </BorderNode>
            </ViewNode>
        </ViewNode>
        <ViewNode customBeanClass="gov.sns.application.DefaultXalWindow" height="547"
tag="MainWindow" width="712">
            <ViewProxy type="javax.swing.JFrame"/>
            <BeanProperty name="title" value="Demo"/>
            <ViewNode tag="Tabbed Pane">
                <ViewProxy type="javax.swing.JTabbedPane"/>
                <ViewNode tag="Magnets Box">
                    <ViewProxy type="javax.swing.Box_Vertical"/>
                    <BeanProperty name="name" value="Magnets"/>
                    <ViewNode tag="Scroll Pane">
                        <ViewProxy type="javax.swing.JScrollPane"/>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                        <ViewNode tag="MagnetList">
                            <ViewProxy type="javax.swing.JList"/>
                        </ViewNode>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                    </ViewNode>
                    <ViewNode tag="Horizontal Box">
                        <ViewProxy type="javax.swing.Box_Horizontal"/>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
```



```
                        </BorderNode>
                        <ViewNode tag="Horizontal Glue">
                            <ViewProxy type="javax.swing.Box_HorizontalGlue"/>
                        </ViewNode>
                        <ViewNode tag="RunButton">
                            <ViewProxy type="javax.swing.JButton"/>
                            <BeanProperty name="text" value="Run"/>
                            <BeanProperty alpha="255" blue="0" green="153" name="foreground"
red="0"/>
                        </ViewNode>
                        <BorderNode tag="Etched Border">
                            <BorderProxy type="javax.swing.border.EtchedBorder"/>
                        </BorderNode>
                    </ViewNode>
                </ViewNode>
                <ViewNode tag="SinePlot">
                    <ViewProxy type="gov.sns.tools.plot.FunctionGraphsJPanel"/>
                    <BeanProperty name="name" value="Sine"/>
                </ViewNode>
            </ViewNode>
        </ViewNode>
    </RootBrick>
</BricksDocument>
```

Conclusion

- The Bricks application allows rapid development of user interfaces without getting in your way
- Bricks supports development of Java applications, XAL applications and jython scripts
- Bricks is built upon an adaptor based model which allows for extensibility
- Opens the possibility of server generated user interfaces