# Open XAL GUI

**Thomas Pelaia II, Ph.D.**

**XAL Workshop 2012**

**December 13, 2012**

Managed by UT-Battelle
for the Department of Energy

OAK RIDGE National Laboratory

# Graphical User Interface Support

- Basically Same for XAL and Open XAL
- Package Names Change
- Some classes moved
- Obsolete code removed

# Two Foundations

- Application Framework
- Bricks

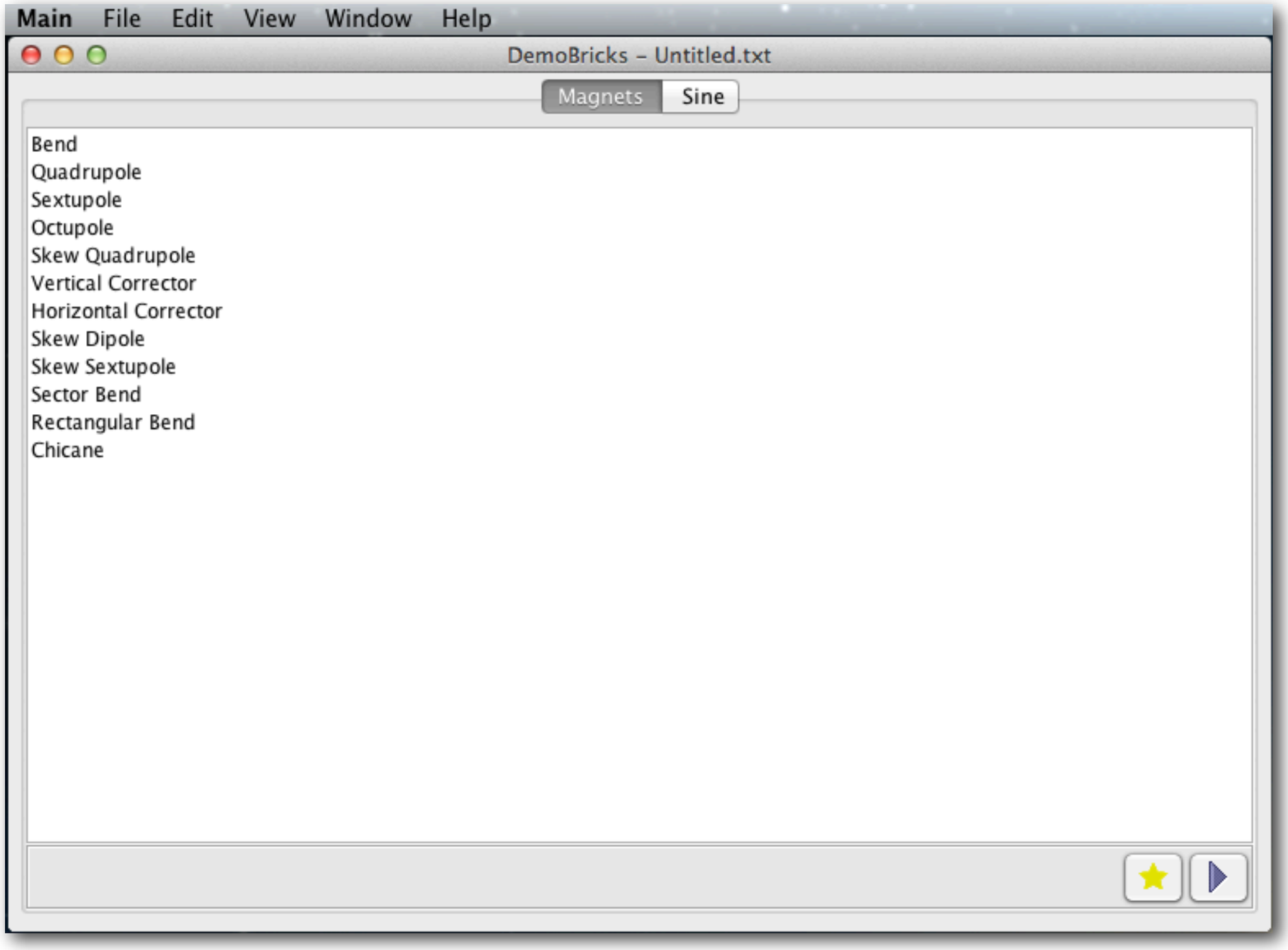
# Application Framework Goals

- Common Look and Feel
- Great user experience
- Rapid application development
- Developer focuses on model
- Evolutionary

# Application Framework

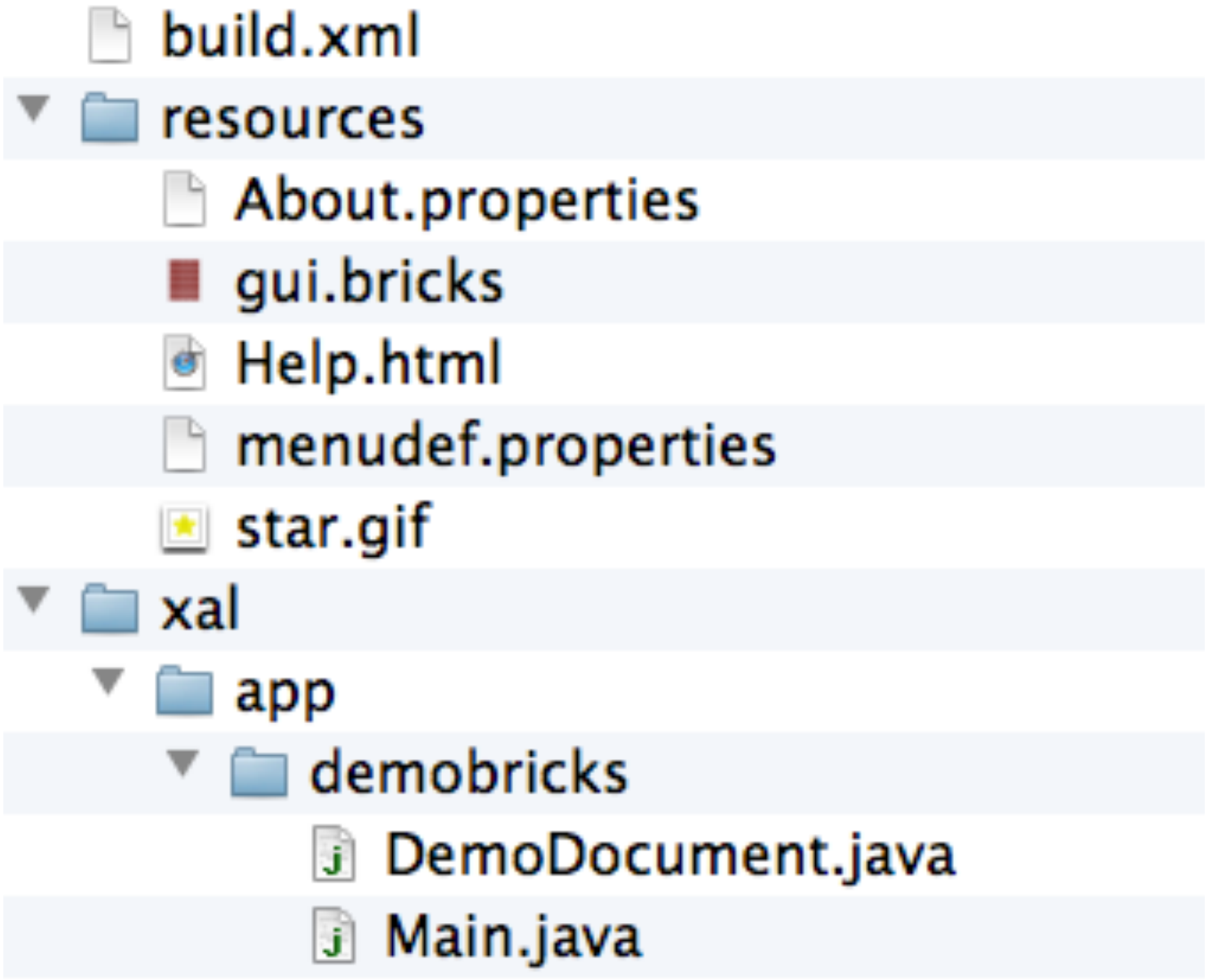
- Over five dozen applications
- Document based applications
  - One window per open document
  - Multiple documents open concurrently
  - Multiple document types supported per application
  - User preference for root document directory
  - Each application assigned subdirectory by name
- Accelerator based applications
  - Document associated with selected accelerator and accelerator sequence
  - Accelerator Menu

# Example Application
## Demo Bricks

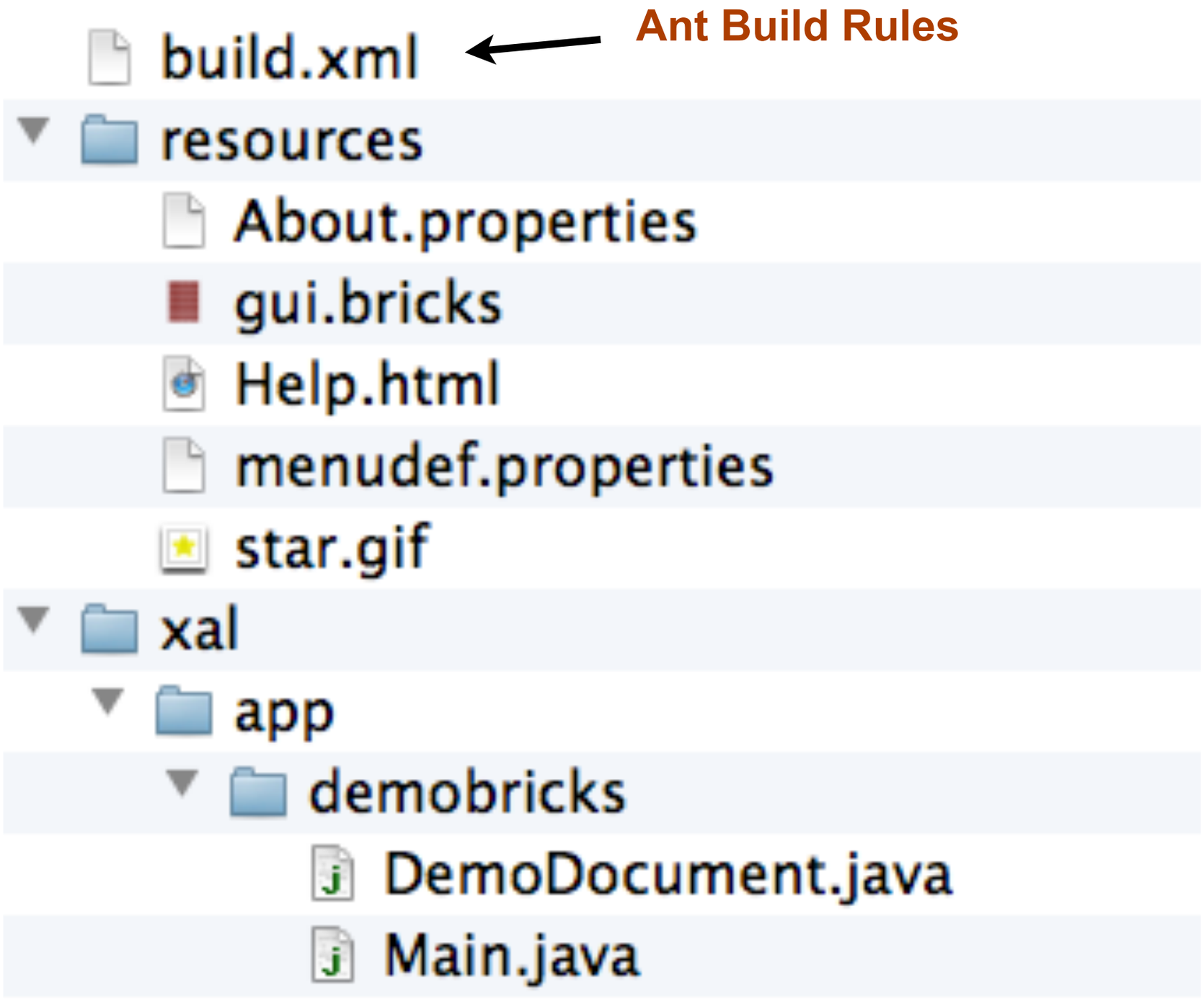


# Anatomy of an Application

## Demo Bricks

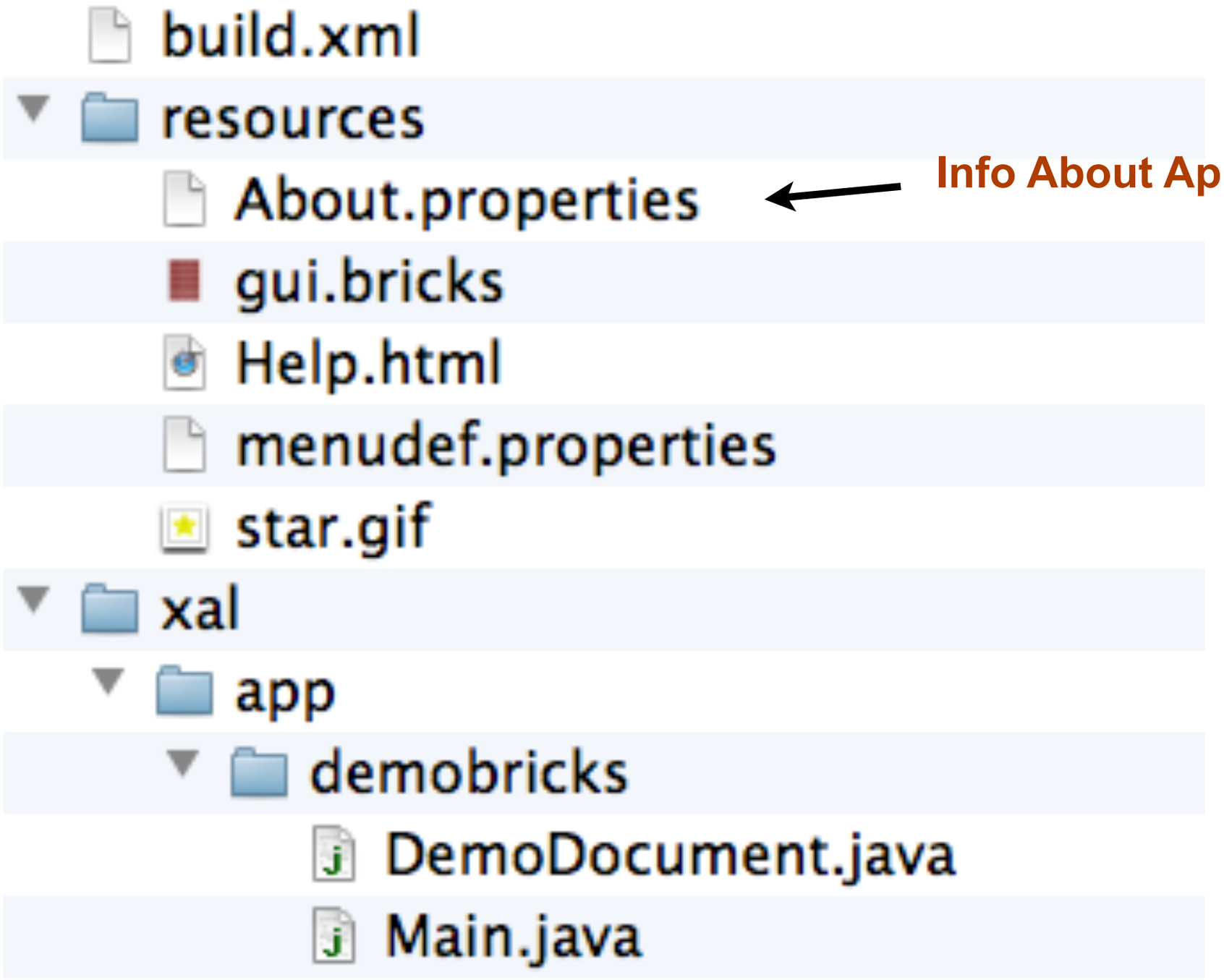


# Anatomy of an Application
## Demo Bricks



# Anatomy of an Application

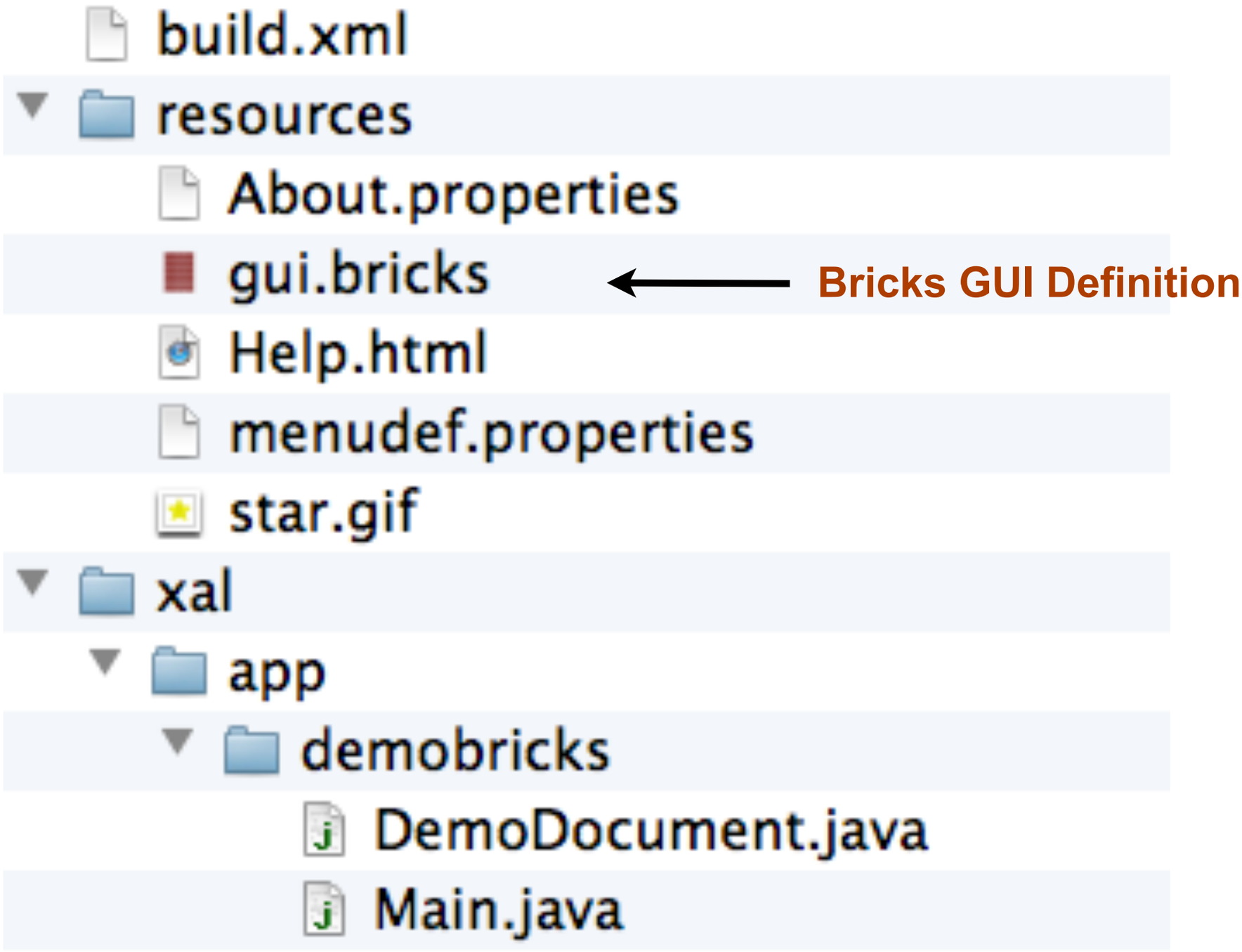
## Demo Bricks




Info About Application

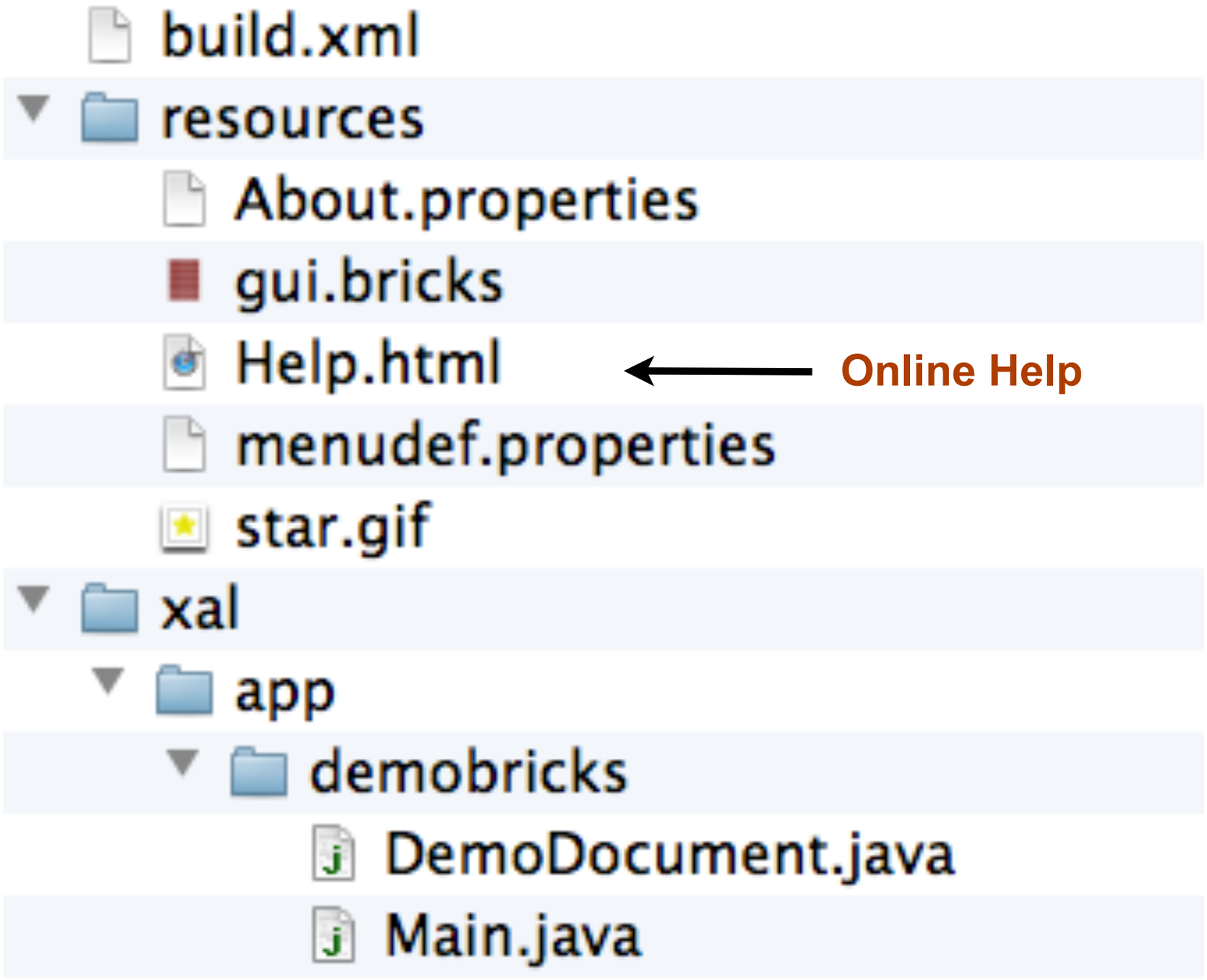
# Anatomy of an Application

## Demo Bricks



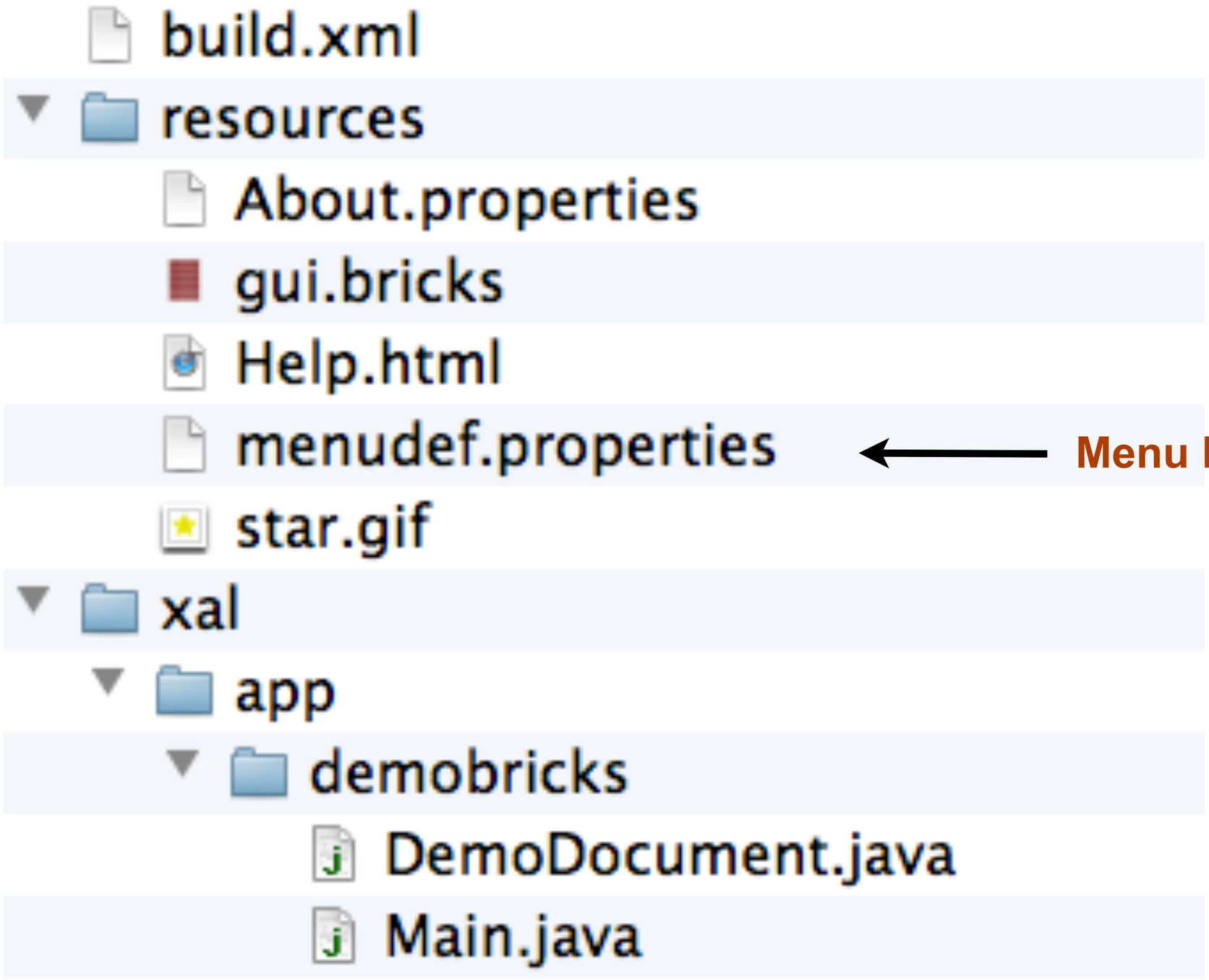
# Anatomy of an Application

## Demo Bricks



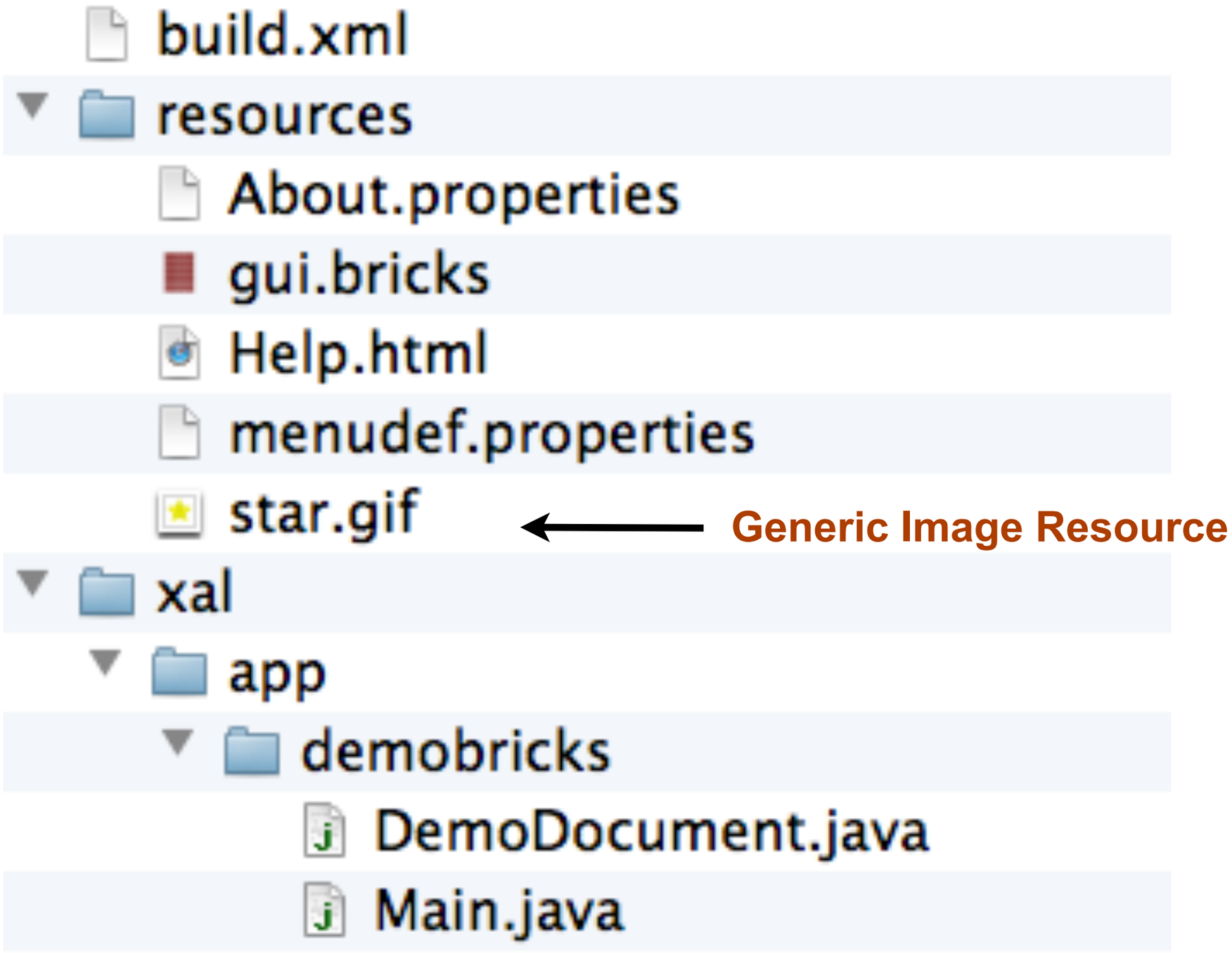
# Anatomy of an Application

## Demo Bricks



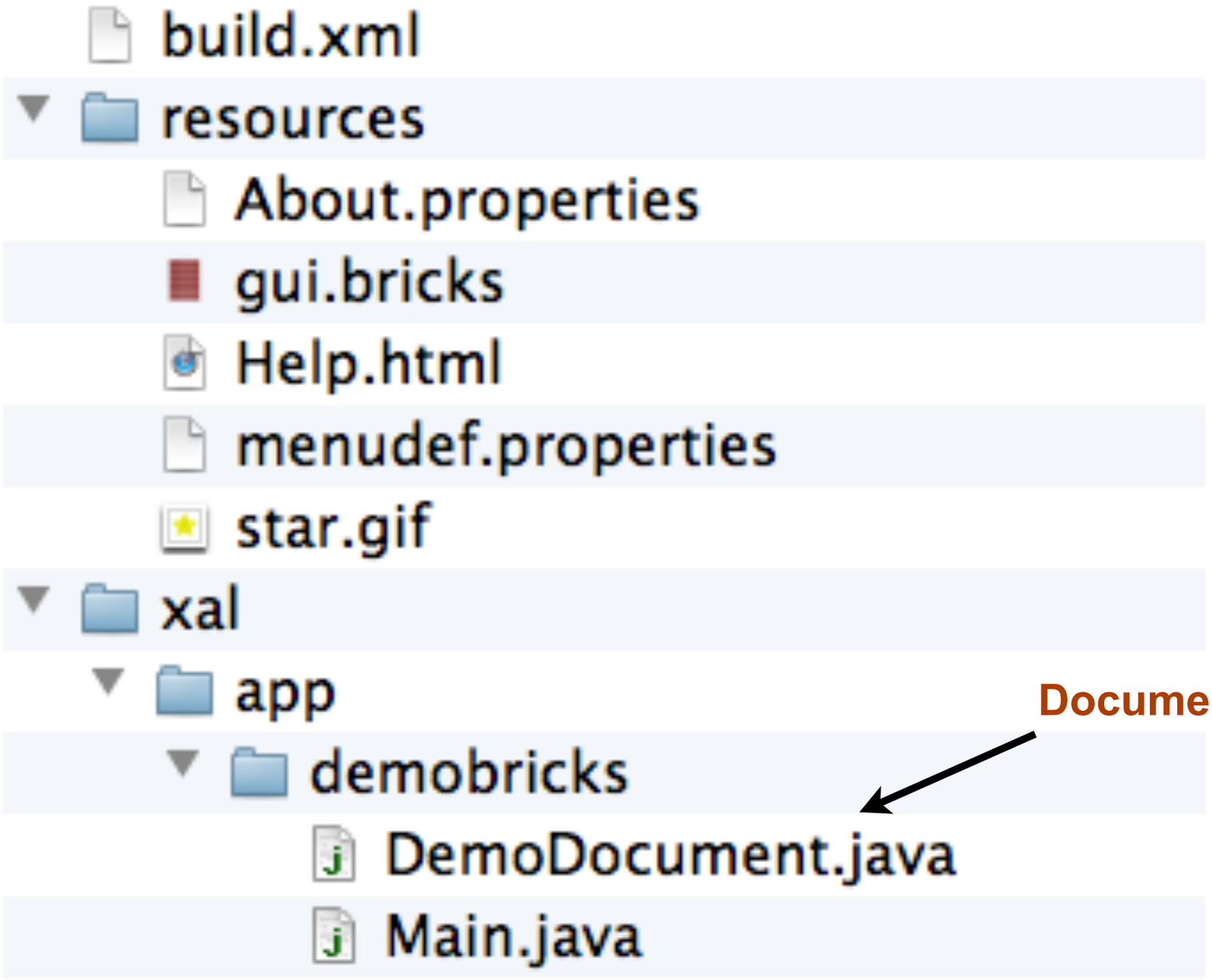
# Anatomy of an Application

## Demo Bricks



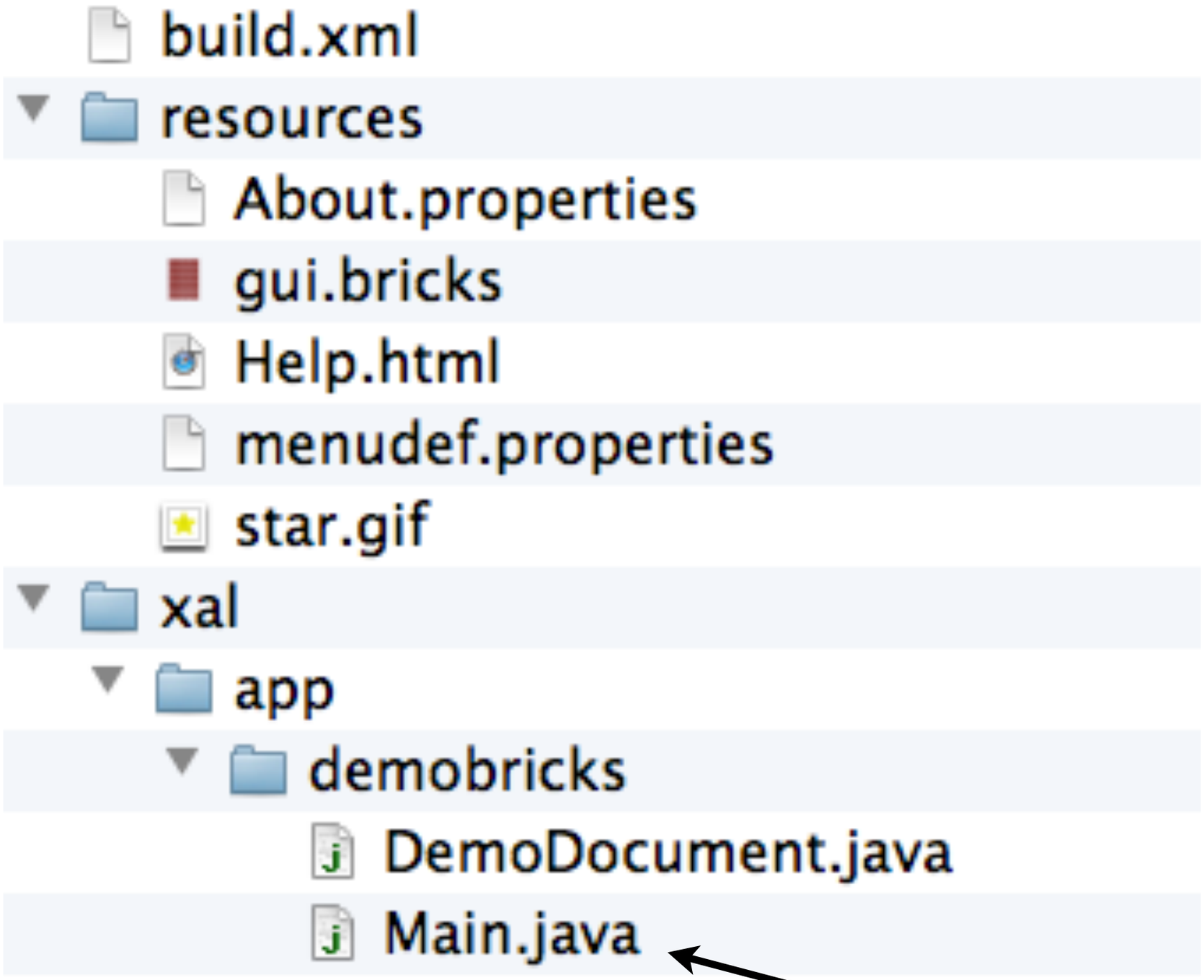
# Anatomy of an Application

## Demo Bricks



# Anatomy of an Application

## Demo Bricks



# File Roles

| File | Inherits From | Role |
|---|---|---|
| build.xml | ../common.xml | Build rule |
| About.properties | | About dialog, Launcher Info |
| gui.bricks | | View definition |
| Help.html | | Online Help |
| menudef.properties | application/ menudef.properties | Menu and Toolbar definition |
| DemoDocument.java | XalDocument | Document properties and state |
| Main.java | ApplicationAdaptor | Application properties and state |

# Ant Build File

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Build the application whose name matches that of the
project. -->
<project name="demobricks" basedir="." default="all">
    <property name="exclude.install" value="true" />
    <!--<property name="exclude.batch.build" value="true" />-->

  <import file="../common.xml"/>
</project>
```

# About Properties

```
name = Bricks Application Demo
version = 1.0.0
date = July 18, 2006
authors = Tom Pelaia
organization = Spallation Neutron Source
description = This application demonstrates how to create a custom
application using the XAL Application Framework and a gui defined
using bricks.
```

# GUI Bricks
# Generated by Bricks GUI Builder

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<BricksDocument date="Mon Jun 22 09:24:13 EDT 2009" version="1.0.0">
    <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="40" 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>
            </ViewNode>
        </ViewNode>
        <ViewNode customBeanClass="xal.application.DefaultXalWindow" height="547" tag="MainWindow" width="772">
            <ViewProxy type="javax.swing.JFrame"/>
            <BeanProperty name="title" value="Demo"/>
...
```

# HTML Help

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
<HEAD>
<style type="text/css">
P.warning {font-size: large; font-weight: bold;}
</style>
<TITLE>Demo Bricks Help</TITLE>
</HEAD>
<BODY>

<center><u>Demo BricksHelp</u></center>

<hr><br>
<a name="intro_ref" <b>Introduction:</b></a>
<p>
The Demo application was developed for the purpose of demonstrating how to an XAL
application using the XAL application framework and the bricks application to define the
GUI.</p>

</BODY>
</HTML>
```

# Menu Definition

Empty For this Example

# Demo Document

```
    public DemoDocument( final URL url ) {
        setSource( url );
    }
```

```
    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" };
         magnetList.setListData( magnets );

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

```
    public void saveDocumentAs( final URL url ) {
    }
```

# Main Application Adaptor

```
public String[] readableDocumentTypes() {
    return new String[] {"txt", "text"};
}

public String[] writableDocumentTypes() {
    return new String[] {"txt", "text"};
}

public XalDocument newEmptyDocument() {
    return new DemoDocument();
}

public XalDocument newDocument( final java.net.URL url ) {
    return new DemoDocument( url );
}

public String applicationName() {
    return "DemoBricks";
}

protected void customizeCommands(Commander commander) {
}

public void applicationFinishedLaunching() {
    System.out.println("Application has finished launching!");
   Logger.getLogger("global").log( Level.INFO, "Application has finished launching." );
}

public Main() {}
```

# Existing Java GUI Builders

- Tied to specific IDE
- Not Model-View-Controller (MVC)
  - Despite claims to the contrary
- Don’t work well with scripts
- Generated source code
  - Includes view decorations
  - Violates descriptive symbol naming rule
  - “Do not touch” sections
  - One way code generation

# Bricks Goals

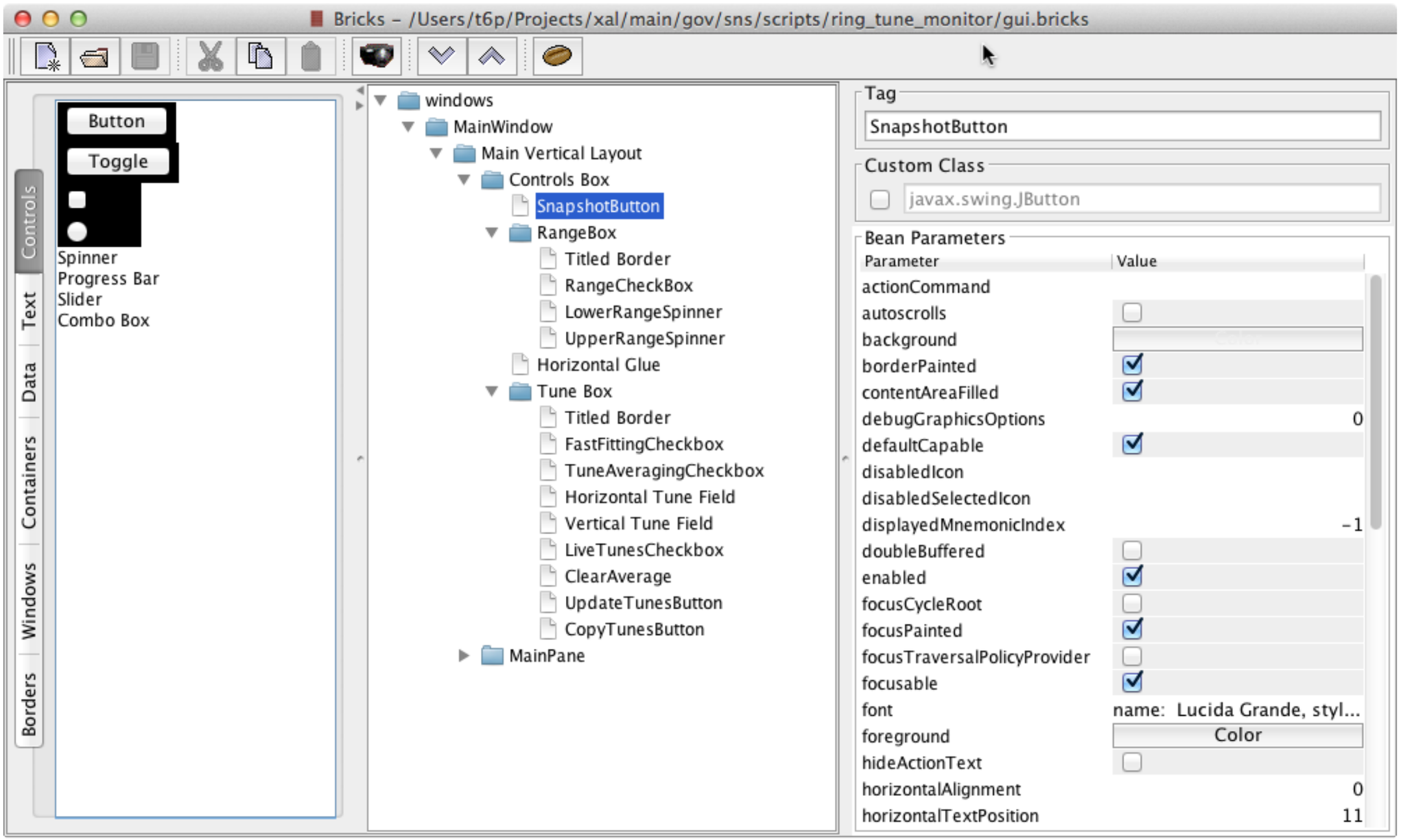
- Encourage true Model-View-Controller Pattern
  - View is defined entirely in XML
  - View decorations specified in bricks
  - Focus on behavior in controller source code
  - No generated source code
- Work with Applications and Scripts
  - No compiler necessary
- Two Components
  - GUI Builder
  - Runtime

# Bricks GUI Builder

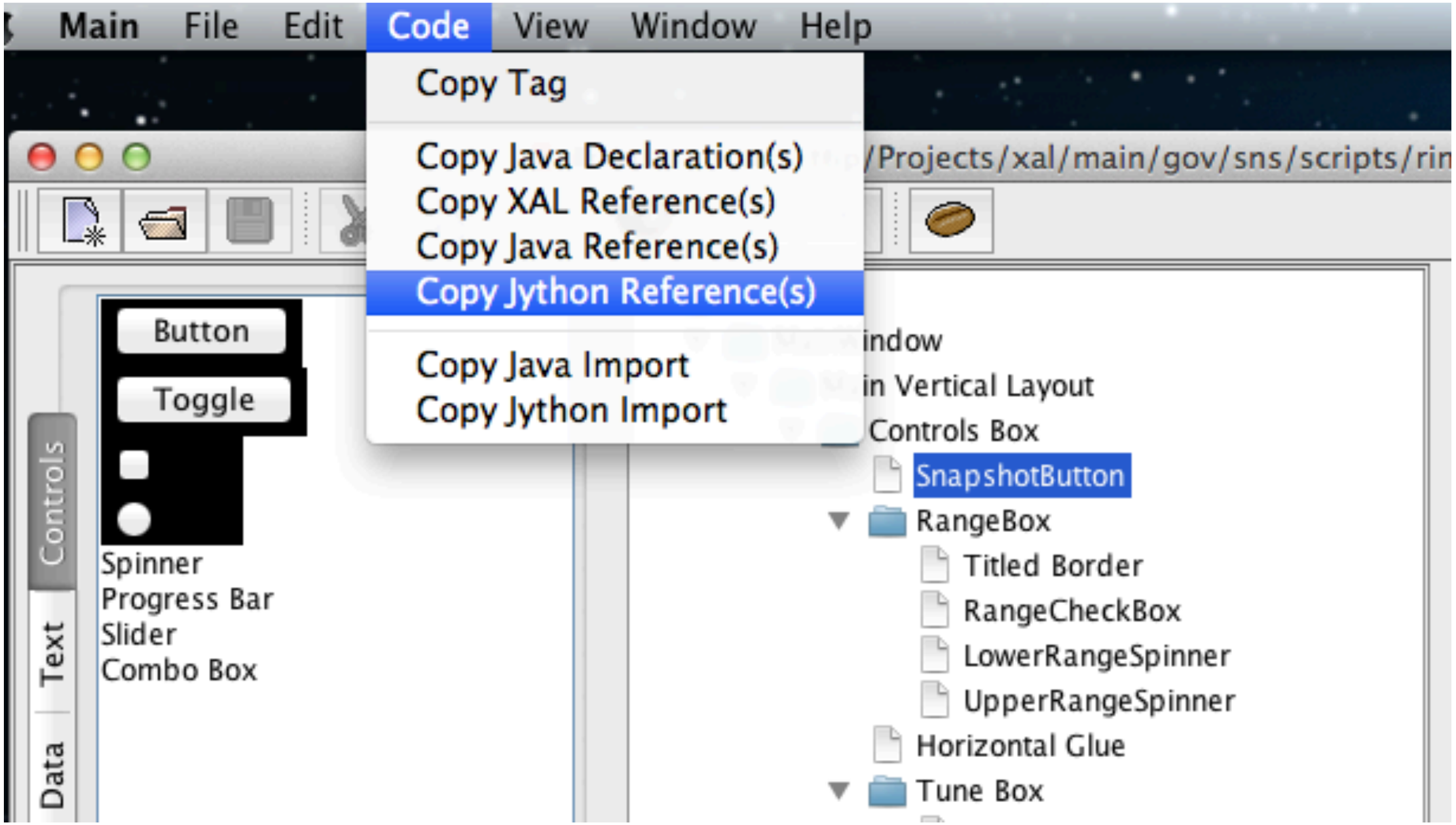
- WYSIWYG Swing Editor
- Box Layout
- Custom Views
- Java Beans Property Editor
- Records windows and dialogs in one XML file
- Code Snippets for view references
- Works with any IDE

# Bricks GUI Builder

## Main Window



# Bricks GUI Builder
## Code Snippets



# Bricks View Reference
# Example: JRuby

## Loading the User Interface

```
def load_user_interface()
   # locate the enclosing folder and get the bricks file within it
   folder = File.dirname( $0 )
   gui_path = File.join( folder, "gui.bricks" )
   url = Java::File.new( gui_path ).toURI.toURL

   # generate a window reference resource and pass the desired constructor arguments
   window_reference = WindowReference::getDefaultInstance( url, "MainWindow" )

   main_controller = ControlApp.new window_reference
   main_controller.displayWindow
end
```

## Implementing Button Handler

```
class SnapshotHandler
   include java.awt.event.ActionListener

   def initialize( window_reference )
        @window = window_reference.getWindow()
        snapshot_button = window_reference.getView( "SnapshotButton" )
        snapshot_button.addActionListener( self )
   end

   def actionPerformed( event )
        ImageCaptureManager.defaultManager().saveSnapshot( @window.getContentPane );
   end
end
```

# Bricks View Reference
# Example: Java Controller

## Loading the User Interface in XAL Document

```
WINDOW_REFERENCE = getDefaultWindowReference( "MainWindow", this );
```

## Implementing  Controller

```
/** Constructor */
public RunController( final LaunchModel model, final WindowReference windowReference ) {
   MODEL = model;
   FILE_WATCHER = model.getFileWatcher();

   final JButton appFilterClearButton = (JButton)windowReference.getView( "AppFilterClearButton" );
   appFilterClearButton.addActionListener( clearFilterAction() );

   final JButton refreshAppsButton = (JButton)windowReference.getView( "AppsRefreshButton" );
   refreshAppsButton.addActionListener( refreshAppsAction() );

   final JButton appLaunchButton = (JButton)windowReference.getView( "AppLaunchButton" );
   appLaunchButton.addActionListener( runAction() );

   FILTER_FIELD = (JTextField)windowReference.getView( "AppFilterField" );
   FILTER_FIELD.putClientProperty( "JTextField.variant", "search" );
   FILTER_FIELD.putClientProperty( "JTextField.Search.Prompt", "Application Filter" );
...
```

# Bricks vs. Others Java GUI Builders

| | Bricks | Them |
|---|---|---|
| Mature | X | ✔ |
| Advanced Layout | X | ✔ |
| IDE Independent | ✔ | X |
| True Model-View-Controller | ✔ | X |
| Works with Scripts | ✔ | X |
| Compiler Free | ✔ | X |

# Future Directions

- Application Framework
  - Extend to Scripts
- Bricks
  - Add support for Generics in Code Snippet
  - Support more Layout Managers
  - Simultaneous property edits to multiple views

# XAL GUI Status

- Application Framework
  - Demonstrated extensibility
  - Stable
  - Flexibile
- Bricks
  - Stable
  - Flexible