C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Main.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import java.awt.*;
import javax.swing.*;

/**
 *
 * @author Brian Cullinan
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {

        // this is our main window frame
        new Application();
    }

}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Application.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
import java.awt.event.*;

/**
 *
 * @author Brian Cullinan
 */
public class Application extends JFrame {

    public Application()
    {
        super();
        setTitle("Media Server");
        setSize(640, 480);
        setExtendedState(JFrame.MAXIMIZED_BOTH);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        new Desktop(this);
    }
}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Desktop.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
import java.awt.event.*;

```

```
/**
 *
 * @author Brian Cullinan
 */
public class Desktop extends JDesktopPane implements ComponentListener, InternalFrameListener
{
    Application application;
    Taskbar taskbar;
    StartMenu startmenu;


    public Desktop(Application application)
    {
        super();
        //try {
        //    UIManager.addAuxiliaryLookAndFeel(new project1.ui.NewLookAndFeel());
        //} catch (Exception ex) {

        //}
        addComponentListener(this);
        this.application = application;
        application.setContentPane(this);
        application.setVisible(true);

        setTaskbar(new Taskbar(this));

        setStartMenu(new StartMenu(this));
    }

    public JInternalFrame add(JInternalFrame frame)
    {
        frame.addInternalFrameListener(this);
        frame.setIconifiable(false);
        super.add(frame);
        return frame;
    }

    public Module add(Module frame)
    {
        frame.addInternalFrameListener(this);
        taskbar.add(frame);
        frame.setIconifiable(false);
        super.add(frame);
        return frame;
    }

    public void setTaskbar(Taskbar taskbar)
    {
        this.taskbar = taskbar;
    }

    public void setStartMenu(StartMenu launcher)
    {
        this.startmenu = launcher;
    }

    public void componentHidden(ComponentEvent e)
    {

    }
    public void componentMoved(ComponentEvent e)
    {

    }
    public void componentResized(ComponentEvent e)
    {
        if(taskbar != null)
        taskbar.setBounds(0,
               getHeight() - taskbar.getHeight() - getInsets().top - getInsets().bottom,
               getWidth() - getInsets().left - getInsets().right,
               taskbar.getHeight());
    }
    public void componentShown(ComponentEvent e)
    {

    }

    public void internalFrameActivated(InternalFrameEvent e) {
        try {
            ((Module)e.getSource()).taskbutton.setSelected(true);
        } catch (Exception ex) {
```

```java

        }
    }
    public void internalFrameClosed(InternalFrameEvent e) {
        try {
            taskbar.taskbar.remove(((Module)e.getSource()).taskbutton);
        } catch (Exception ex) {

        }
    }
    public void internalFrameClosing(InternalFrameEvent e) {

    }
    public void internalFrameDeactivated(InternalFrameEvent e) {
        if(e.getSource() == startmenu)
        {
            startmenu.setVisible(false);
        }
        try {
            ((Module)e.getSource()).taskbutton.setSelected(false);
        } catch (Exception ex) {

        }
    }
    public void internalFrameDeiconified(InternalFrameEvent e) {

    }
    public void internalFrameIconified(InternalFrameEvent e) {

    }
    public void internalFrameOpened(InternalFrameEvent e) {

    }

}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Taskbar.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
import java.awt.event.*;
import project1.ui.*;

/**
 *
 * @author Brian Cullinan
 */
public class Taskbar extends JInternalFrame implements ActionListener, MouseListener {

    //private static final String uiClassID = "TaskbarUI";

    Desktop desktop;

    JPanel taskbar;
    // the start button on the taskbar
    JButton start;

    JPopupMenu popup;

    GridLayout layout;

    public Taskbar(Desktop desktop)
    {
        super();
        this.desktop = desktop;
        ((javax.swing.plaf.basic.BasicInternalFrameUI)getUI()).setNorthPane(null);
        setBounds(0,
                desktop.getHeight() - 50 - desktop.getInsets().top - desktop.getInsets().bottom,
                desktop.getWidth() - desktop.getInsets().left - desktop.getInsets().right,
                50);
        setLayout(new BorderLayout());
        setLayer(9999);

        start = new JButton("Start");
        start.addActionListener(this);
        add(start, BorderLayout.WEST);

        taskbar = new JPanel();
        layout = new GridLayout(1, 4);
        taskbar.setLayout(layout);
        add(taskbar, BorderLayout.CENTER);

        popup = new JPopupMenu();
        JMenuItem close = new JMenuItem("Close");
        close.addActionListener(this);
        popup.add(close);

        desktop.add(this);
        setVisible(true);
    }

    public TaskButton add(Module frame) {

        TaskButton taskbutton = new TaskButton(frame);
        if(taskbar.getComponents().length == layout.getColumns())
            layout.setColumns(layout.getColumns()+1);
        taskbar.add(taskbutton);
        frame.taskbutton = taskbutton;
        taskbutton.addActionListener(this);
        taskbutton.addMouseListener(this);

        return taskbutton;
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == start)
        {
            desktop.startmenu.setVisible(true);
            desktop.startmenu.setLocation(0, getY() - desktop.startmenu.getHeight());
            desktop.getDesktopManager().activateFrame(desktop.startmenu);
            desktop.startmenu.toFront();
            try {
                desktop.startmenu.setSelected(true);
            }
            catch (Exception ex) {
                System.out.println(ex);
            }
            desktop.startmenu.requestFocus();
        }
        if(e.getSource().getClass() == TaskButton.class)
        {
            TaskButton button = (TaskButton)e.getSource();
            Module frame = (Module)button.frame;
            if(frame.isIcon() || !frame.isSelected())
            {
                desktop.getDesktopManager().activateFrame(frame);
                frame.toFront();
                frame.requestFocus();
                try {
                    frame.setIcon(false);
                    frame.setSelected(true);
                } catch (Exception ex) {

                }
            }
            else
            {
                button.setSelected(false);
                desktop.getDesktopManager().activateFrame(desktop.taskbar);
                frame.toBack();
                try {
                    desktop.taskbar.setSelected(true);
                    frame.setIcon(true);
                    frame.setSelected(false);
                } catch (Exception ex) {

                }
            }
        }
        if(e.getSource().getClass() == JMenuItem.class)
        {
            JMenuItem item = (JMenuItem)e.getSource();
```

```java
                JPopupMenu menu = (JPopupMenu)item.getParent();
                TaskButton task = (TaskButton)menu.getInvoker();
                try {
                    task.frame.setClosed(true);
                } catch (Exception ex) {
                    System.out.println(ex);
                }
            }
        }

        public void mouseClicked(MouseEvent e) {

        }
        public void mouseEntered(MouseEvent e) {

        }
        public void mouseExited(MouseEvent e) {

        }
        public void mousePressed(MouseEvent e) {

        }
        public void mouseReleased(MouseEvent e) {
            if (e.isPopupTrigger()) {
                popup.show(e.getComponent(),
                           e.getX(), e.getY());
            }
        }


        //public String getUIClassID() {
        //    return uiClassID;
        //}
    }


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\TaskButton.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
/**
 *
 * @author Brian Cullinan
 */
public class TaskButton extends JToggleButton {

    Module frame;
    public TaskButton(Module frame)
    {
        super(frame.getTitle());
        this.frame = frame;
    }
}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\StartMenu.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import javax.swing.event.*;
import java.awt.event.*;
import org.xml.sax.*;
import java.util.*;
import org.xml.sax.helpers.*;
import javax.swing.table.*;
import java.awt.event.*;

/**
 *
 * @author Brian Cullinan
 */
public class StartMenu extends JInternalFrame implements ActionListener, ContentHandler {

    Desktop desktop;

    JButton home, downloads;
    JPanel quick_folders;

    public StartMenu(Desktop desktop)
    {
        super();
        this.desktop = desktop;

        // this is the panel that holds the special locations and main folders
        ((javax.swing.plaf.basic.BasicInternalFrameUI)getUI()).setNorthPane(null);
        setSize(100, 480);
        setLayout(new BorderLayout());

        // this is the panel for the special locations (Downloads, etc.)
        JPanel locations = new JPanel();
        locations.setLayout(new GridLayout(0, 1));
        home = new JButton("Home");
        home.addActionListener(this);
        locations.add(home);
        add(locations, BorderLayout.NORTH);

        // this is the panel for the main directory "quick access"
        quick_folders = new JPanel();
        quick_folders.setLayout(new GridLayout(0, 1));
        quick_folders.setSize(100, 330);
        add(quick_folders, BorderLayout.CENTER);
        // get quick folders
        try {
            XMLReader xr = XMLReaderFactory.createXMLReader();
            xr.setContentHandler(this);
            xr.parse("http://dev.bjcullinan.com/plugins/select.php?template=extjs&dir=/Shared/&start=0&limit
        } catch (Exception ex) {
            System.out.println(ex);
        }

        desktop.add(this);
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == home)
        {
            new Portal(desktop);
        }
        else
        {
            if(e.getSource().getClass() == JButton.class)
                new Portal(desktop, "/Shared/" + ((JButton)e.getSource()).getText() + "/");
        }
    }
    public void setDocumentLocator (Locator locator)
    {   }

    public void startDocument ()
        throws SAXException
    {   }
    public void endDocument()
        throws SAXException
    {   }
    public void startPrefixMapping (String prefix, String uri)
        throws SAXException
    {   }
    public void endPrefixMapping (String prefix)
        throws SAXException
    {   }

    String current_el;
    boolean in_file = false;
    String filename = "";
    public void startElement (String uri, String localName,
                              String qName, Attributes atts)
        throws SAXException
    {
        current_el = qName;
        if(current_el.equals("file"))
        {
            in_file = true;
        }
    }
    public void endElement (String uri, String localName,
                            String qName)
        throws SAXException
    {
        if(qName.equals("file"))
        {
            in_file = false;
            JButton new_button = new JButton(filename);
            new_button.addActionListener(this);
            quick_folders.add(new_button);
        }

        current_el = "";
    }
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        if(in_file && current_el.equals("name"))
        {
            filename = "";
            for(int i = start; i < start+length; i++)
                filename += ch[i];
        }
    }
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {   }
    public void processingInstruction (String target, String data)
        throws SAXException
    {   }
    public void skippedEntity (String name)
        throws SAXException
    {   }

}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Module.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
/**
 *
 * @author Brian Cullinan
 */
public class Module extends JInternalFrame {
    JToggleButton taskbutton;
}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Portal.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.net.*;
import java.io.*;
```

```java
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.swing.tree.*;
import javax.swing.event.*;
/**
 *
 * @author Brian Cullinan
 */
public class Portal extends Module implements ActionListener, org.xml.sax.ContentHandler, TreeSelectionListe

    Desktop desktop;
    FolderView view;
    Address address;
    Stack<String> back_stack;
    Stack<String> forward_stack;
    JProgressBar progress;
    JPanel search_pane, folder_tasks;
    JScrollPane folders_pane;
    JToggleButton search, folders;
    JTextField contains;
    JTree tree;

    // toolbar buttons that need controlling from other classes
    JButton back, forward, up;

    public Portal(Desktop desktop)
    {
        this(desktop, "/");
    }

    public Portal(Desktop desktop, String location)
    {
        super();
        this.desktop = desktop;
        back_stack = new Stack<String>();
        forward_stack = new Stack<String>();

        setTitle("Portal");
        setResizable(true);
        setClosable(true);
        setMaximizable(true);
        setLayout(new BorderLayout());
        setSize(540, 480);

        JPanel tool_container = new JPanel();
        tool_container.setLayout(new GridLayout(0, 1));
        add(tool_container, BorderLayout.NORTH);

        JToolBar tools = new JToolBar();
        tools.setFloatable(false);
        tool_container.add(tools, BorderLayout.NORTH);
        // these are some buttons for controlling the folder
        back = new JButton("Back");
        back.addActionListener(this);
        back.setEnabled(false);
        tools.add(back);
        forward = new JButton("Forward");
        forward.addActionListener(this);
        forward.setEnabled(false);
        tools.add(forward);
        up = new JButton("Up");
        up.addActionListener(this);
        up.setEnabled(false);
        tools.add(up);
        JButton refresh = new JButton("Refresh");
        refresh.addActionListener(this);
        tools.add(refresh);
        search = new JToggleButton("Search");
        search.addActionListener(this);
        tools.add(search);
        folders = new JToggleButton("Folders");
        folders.addActionListener(this);
        tools.add(folders);

        // the main address of the current folder
        address = new Address(this, location);
        tool_container.add(address, BorderLayout.NORTH);

        // create a view for the files
        view = new FolderView(this, address);
        add(view, BorderLayout.CENTER);
        progress = new JProgressBar();
        progress.setStringPainted(true);
        add(progress, BorderLayout.SOUTH);

        // this is the folder tasks panel
        JPanel tasks = new JPanel();
        tasks.setLayout(new OverlayLayout(tasks));
        add(tasks, BorderLayout.WEST);

        folder_tasks = new JPanel();
        folder_tasks.setLayout(new FlowLayout());
        tasks.add(folder_tasks);

        JButton send_to_downloads = new JButton("Download Selected");
        send_to_downloads.addActionListener(this);
        folder_tasks.add(send_to_downloads);

        search_pane = new JPanel();
        search_pane.setLayout(new FlowLayout());
        search_pane.setVisible(false);
        tasks.add(search_pane);

        contains = new JTextField();
        contains.setPreferredSize(new Dimension(100, 20));
        search_pane.add(contains);

        JButton search_btn = new JButton("Search");
        search_btn.addActionListener(this);
        search_pane.add(search_btn);

        folders_pane = new JScrollPane();
        folders_pane.setVisible(false);
        tasks.add(folders_pane);

        DefaultMutableTreeNode root = new DefaultMutableTreeNode("Shared");
        tree = new JTree(root);
        tree.addTreeSelectionListener(this);
        folders_pane.setViewportView(tree);

        desktop.add(this);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        if(e.getSource().getClass() == JButton.class)
        {
            if(((JButton)e.getSource()).getText().equals("Back"))
            {
                address.setAddress(back_stack.peek(), true);
                forward_stack.push(back_stack.pop());
                forward.setEnabled(true);
                if(back_stack.size() == 0)
                    back.setEnabled(false);
            }
            if(((JButton)e.getSource()).getText().equals("Forward"))
            {
                address.setAddress(forward_stack.peek(), true);
                back_stack.push(forward_stack.pop());
                back.setEnabled(true);
                if(forward_stack.size() == 0)
                    forward.setEnabled(false);
            }
            if(((JButton)e.getSource()).getText().equals("Up"))
            {
                // remove folder from address
                String folders[] = address.address.getText().split("/");
                String new_add = "/";
                if(folders.length >= 3)
                {
                    for(int i = 1; i < folders.length - 1; i++)
                    {
                        new_add += folders[i] + "/";
                    }
                }
                if(new_add.equals("/"))
                    up.setEnabled(false);
                else
                    up.setEnabled(true);
                address.setAddress(new_add);
            }

            if(((JButton)e.getSource()).getText().equals("Refresh"))
            {
                view.refresh(address.address.getText());
            }

            if(((JButton)e.getSource()).getText().equals("Download Selected"))
            {
                if(view.files.getSelectedRowCount() > 0)
                {
                    JFileChooser chooser = new JFileChooser();
                    int returnval = chooser.showSaveDialog(this);
                    if(returnval == JFileChooser.APPROVE_OPTION)
                    {
                        String files = "";
                        for(int row : view.files.getSelectedRows())
                        {
                            if(!view.files.getValueAt(row, 6).equals("FOLDER"))
                            {
                                files += view.files.getValueAt(row, 0) + ",";
                            }
                        }
                        files = files.substring(0, files.length()-1);
                        try {
                            URL url = new URL("http://dev.bjcullinan.com/plugins/zip/" + files + "/file.zip'
                            progress.setValue(0);
                            Download down = new Download(url, chooser.getSelectedFile(), progress);
                            down.start();
                        } catch (Exception ex) {
                            System.out.println(ex);
                        }
                    }
                }
            }
            if(((JButton)e.getSource()).getText().equals("Search"))
            {
                view.refresh(new String[]{"template","dir","start","limit","includes"}, new String[]{"extjs'
            }
        }
        if(e.getSource().getClass() == JToggleButton.class)
        {
            if(((JToggleButton)e.getSource()).getText().equals("Search"))
            {
                JToggleButton button = (JToggleButton)e.getSource();
                if(button.isSelected())
                {
                    search_pane.setVisible(true);
                    folder_tasks.setVisible(false);
                    folders_pane.setVisible(false);
                    folders.setSelected(false);
                }
                else
                {
                    if(!folders.isSelected())
                    {
                        search_pane.setVisible(false);
                        folder_tasks.setVisible(true);
                        folders_pane.setVisible(false);
                    }
                }
            }
            if(((JToggleButton)e.getSource()).getText().equals("Folders"))
            {
                JToggleButton button = (JToggleButton)e.getSource();
                if(button.isSelected())
                {
                    search_pane.setVisible(false);
                    folder_tasks.setVisible(false);
                    folders_pane.setVisible(true);
                    search.setSelected(false);
                }
                else
                {
                    if(!folders.isSelected())
                    {
                        search_pane.setVisible(false);
                        folder_tasks.setVisible(true);
                        folders_pane.setVisible(false);
                    }
                }
            }
        }
```

```java
    }
    class Download extends Thread implements Runnable {
        URL url;
        File output;
        JProgressBar progress;
        public Download(URL url, File output, JProgressBar progress)
        {
            this.url = url;
            this.output = output;
            this.progress = progress;
        }
        @Override
        public void run()
        {
            try {
                URLConnection connection = url.openConnection();
                int length = Integer.parseInt(connection.getHeaderField("Content-Length"));
                progress.setMaximum(length);
                InputStream in = connection.getInputStream();
                FileOutputStream out = new FileOutputStream(output);
                byte[] buf = new byte[4 * 1024]; // 4K buffer
                int bytesRead;
                while ((bytesRead = in.read(buf)) != -1) {
                    out.write(buf, 0, bytesRead);
                    progress.setValue(progress.getValue() + buf.length);
                    progress.setString("Downloading " + Math.round(((double)progress.getValue() / (double)pr
                }
                in.close();
                out.close();
            } catch (Exception ex) {
                System.out.println(ex);
            }
        }
    }

    public void setDocumentLocator (Locator locator)
    {   }

    public void startDocument ()
        throws SAXException
    {   }
    public void endDocument()
        throws SAXException
    {   }
    public void startPrefixMapping (String prefix, String uri)
        throws SAXException
    {   }
    public void endPrefixMapping (String prefix)
        throws SAXException
    {   }

    String current_el;
    boolean in_file = false;
    String filename = "";
    public void startElement (String uri, String localName,
                              String qName, Attributes atts)
        throws SAXException
    {
        current_el = qName;
        if(current_el.equals("file"))
        {
            in_file = true;
        }
    }
    public void endElement (String uri, String localName,
                            String qName)
        throws SAXException
    {
        if(qName.equals("file"))
        {
            in_file = false;
            DefaultMutableTreeNode node = new DefaultMutableTreeNode(filename);
            ((DefaultMutableTreeNode)tree.getLastSelectedPathComponent()).add(node);
        }

        current_el = "";
    }
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        if(in_file && current_el.equals("name"))
        {
            filename = "";
            for(int i = start; i < start+length; i++)
                filename += ch[i];
        }
    }
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {   }
    public void processingInstruction (String target, String data)
        throws SAXException
    {   }
    public void skippedEntity (String name)
        throws SAXException
    {   }

    public void valueChanged(TreeSelectionEvent e) {
        DefaultMutableTreeNode node = (DefaultMutableTreeNode)
                           tree.getLastSelectedPathComponent();

        String path = "/";
        for(int i = 0; i < node.getPath().length; i++)
        {
            path += node.getPath()[i].toString() + "/";
        }

        if (node == null) return;

        try {
            XMLReader xr = XMLReaderFactory.createXMLReader();
            xr.setContentHandler(this);
            xr.parse("http://dev.bjcullinan.com/plugins/select.php?template=extjs&dir=" + path + "&start=0&l
        } catch (Exception ex) {
            System.out.println(ex);
        }

        address.setAddress(path);
        tree.expandPath(tree.getSelectionPath());
    }

}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\Address.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
/**
 *
 * @author Brian Cullinan
 */
public class Address extends JToolBar {

    JTextField address;
    FolderView view;
    Portal portal;

    public Address(Portal portal, String value)
    {
        this.portal = portal;
        address = new JTextField(value);

        setFloatable(false);
        add(new JLabel("Address"));
        add(address);
    }

    public void setAddress(String value)
    {
        setAddress(value, false);
    }
    public void setAddress(String value, boolean dont_change_stack)
    {
        if(!dont_change_stack)
        {
            portal.back.setEnabled(true);
            portal.back_stack.push(address.getText());
            if(portal.forward_stack.size() > 0 && !value.equals(portal.forward_stack.peek()))
            {
                portal.forward.setEnabled(false);
                portal.forward_stack = new Stack<String>();
            }
        }
        if(!value.equals("/"))
            portal.up.setEnabled(true);
        address.setText(value);
        // get last folder to set as title
        String title = "Portal - ";
        String folders[] = value.split("/");
        if(folders.length > 1)
            title = "Portal - " + folders[folders.length-1];
        else
            title = "Portal - Home";
        portal.setTitle(title);
        if(portal.taskbutton != null)
            portal.taskbutton.setText(portal.getTitle());
        view.refresh(value);
    }
}


```

C:\Documents and Settings\Brian Cullinan\My Documents\My Homework\CS 477\Project 1\src\project1\FolderView.java

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package project1;

import javax.swing.*;
import java.awt.*;
import javax.xml.transform.stream.*;
import javax.xml.transform.sax.*;
import javax.xml.transform.*;
import javax.xml.*;
import org.xml.sax.*;
import java.io.*;
import java.util.*;
import org.xml.sax.helpers.*;
import javax.swing.table.*;
import java.awt.event.*;
/**
 *
 * @author Brian Cullinan
 */
public class FolderView extends JScrollPane implements ContentHandler
{

    JTable files;
    Address address;
    Portal portal;
    DefaultTableModel model;
    boolean column_empty[];

    public FolderView(Portal portal, Address value)
    {
        this.portal = portal;
        this.address = value;
        address.view = this;
        // this is the main folder view

        files = new JTable()
        {
            public boolean isCellEditable(int row, int col)
            {
                return false;
            }
        };
        model = new DefaultTableModel();
        files.addMouseListener(new MouseAdapter(){
            public void mouseClicked(MouseEvent e){
                if (e.getClickCount() == 2){
                    int row = files.rowAtPoint(e.getPoint());
                    address.setAddress((String)files.getModel().getValueAt(row, 6));
                }
            }
        });

        try {
            // get the columns from the server
            XMLReader xr = XMLReaderFactory.createXMLReader();
            xr.setContentHandler(this);
            xr.parse("http://dev.bjcullinan.com/plugins/display.php?template=extjs");
        } catch (Exception ex) {
            System.out.println(ex);
        }

        // get the files and folders
        address.setAddress(address.address.getText(), true);

        setViewportView(files);
    }

    public void refresh(String dir)
    {
        refresh(new String[]{"template","dir","start","limit"}, new String[]{"extjs",dir,"0", "300"});
    }

    public void refresh(String[] keys, String[] values)
    {
        // reset empty values
        for(int i = 0; i < column_empty.length; i++)
            column_empty[i] = true;
        // remove existing rows
        rows = new Vector<Vector>();
        // get extra fields
        String field_str = "";
        for(int i = 0; i < keys.length; i++)
        {
            field_str += "&" + keys[i] + "=" + values[i];
        }
        // get the rows from the server
        try {
            XMLReader xr = XMLReaderFactory.createXMLReader();
            xr.setContentHandler(this);
            xr.parse("http://dev.bjcullinan.com/plugins/select.php?" + field_str);
        } catch (Exception ex) {
            System.out.println(ex);
        }
        model.setDataVector(rows, column_arr);
        hideBlankColumns();
    }

    public void hideBlankColumns()
    {
        Vector<String> new_columns = new Vector<String>();
        // hide blank columns
        for(int i = column_empty.length-1; i >= 0; i--)
        {
            if(column_empty[i] == false || i < 6)
            {
                new_columns.add(0, column_arr.get(i));
            }
            else
            {
                for(int j = 0; j < rows.size(); j++)
                {
                    rows.get(j).remove(i);
                }
            }
        }
        model.setColumnIdentifiers(new_columns);
        files.setModel(model);
    }

    public void setDocumentLocator (Locator locator)
    {   }

    public void startDocument ()
        throws SAXException
    {   }
    public void endDocument()
        throws SAXException
    {   }
    public void startPrefixMapping (String prefix, String uri)
        throws SAXException
    {   }
    public void endPrefixMapping (String prefix)
        throws SAXException
    {   }

    String current_el;
    String columns = "";
    Vector<String> column_arr;
    Vector<String> row;
    Vector<Vector> rows;
    boolean in_file = false;

    public void startElement (String uri, String localName,
                              String qName, Attributes atts)
        throws SAXException
    {
        current_el = qName;
        if(current_el.equals("file"))
        {
            in_file = true;
            row = new Vector<String>(column_arr.size(), column_arr.size());
            for(int i = 0; i < column_arr.size(); i++)
                row.add("");
        }
    }
    public void endElement (String uri, String localName,
                            String qName)
        throws SAXException
    {

        if(qName.equals("columns"))
        {
            column_arr = new Vector<String>();
            String column_tmp[] = columns.split(",");
            for(String column_str : column_tmp)
            {
                if(!column_str.equals(""))
                {
                    column_arr.add(column_str);
                    model.addColumn(column_str);
                }
            }
            column_empty = new boolean[column_arr.size()];
        }
        if(qName.equals("file"))
        {
            in_file = false;
            rows.add(row);
        }

        current_el = "";
    }
    public void characters (char ch[], int start, int length)
        throws SAXException
    {
        if(current_el.equals("columns"))
        {
            for(int i = start; i < start+length; i++)
            {
                columns += ch[i];
            }
        }
        if(in_file && current_el.length() > 5 && current_el.substring(0, 5).equals("info-"))
        {
            int index = column_arr.indexOf(current_el.substring(5, current_el.length()));
            if(index != -1)
            {
                String value = "";
                for(int i = start; i < start+length; i++)
                {
                    value += ch[i];
                }
                value = value.trim();
                if(!value.equals(""))
                    column_empty[index] = false;
                row.set(index, row.get(index) + value);
            }
        }
    }
    public void ignorableWhitespace (char ch[], int start, int length)
        throws SAXException
    {   }
    public void processingInstruction (String target, String data)
        throws SAXException
    {   }
    public void skippedEntity (String name)
        throws SAXException
    {   }

}


```