Re-show introduction | Pause

Commit description

Refactor the requests for the virtual file system browser that still used work requests to use the Task API

Code changes

Below you find the code changes to review. The old version of the code is on the left, the new version is on the right.

To add a review remark, click on the respective line number. To delete it, click on it again and delete the remark's text. If a defect spans multiple lines, just mark one of those lines. If similar defects appear multiple times, please mark every occurrence. If you suspect something could be a defect but are not 100% sure, it's better to add a review remark.

At several of the change parts, you can show the whole changed method by clicking on "(Show more context)".

org/experiment/editor/browser/AbstractBrowserTask.java, constructor

```
40  ~~    /**~~
41  ~~     * Creates a new browser I/O request.~~
42  ~~     * @param browser The VFS browser instance~~
43  ~~     * @param path1 The first path name to operate on~~
44  ~~     * @param path2 The second path name to operate on~~
45  ~~     * @param loadInfo A two-element array filled out by the request;~~
46  ~~     * element 1 is the canonical path, element 2 is the file list.~~
47  ~~     */~~
48  ~~    AbstractBrowserTask(VFSBrowser browser,~~
49  ~~            Object session, VFS vfs, String path1, String path2,~~
50  ~~            Object[] loadInfo)~~
51  ~~    {~~
52  ~~        this(browser, session, vfs, path1, path2, loadInfo, null);~~
53  ~~    }~~
```

org/experiment/editor/browser/AbstractBrowserTask.java

```
40
```

org/experiment/editor/browser/AbstractBrowserTask.java, constructor

org/experiment/editor/browser/AbstractBrowserTask.java, constructor

```
55      /**
56       * Creates a new browser I/O request.
57       * @param browser The VFS browser instance
58       * @param path1 The first path name to operate on
59       * @param path2 The second path name to operate on
60       * @param loadInfo A two-element array filled out by the request;
61       * element 1 is the canonical path, element 2 is the file list.
62       */
63      AbstractBrowserTask(VFSBrowser browser,
64          Object session, VFS vfs, String path1, String path2,
65          Object[] loadInfo, Runnable awtTask)
66      {
67          this.browser = browser;
68          this.session = session;
69          this.vfs = vfs;
70          this.path1 = path1;
71          this.path2 = path2;
72          this.loadInfo = loadInfo;
73          if (awtTask != null)
74          {
75              MyTaskListener listener = new MyTaskListener(awtTask);
76              TaskManager.instance.addTaskListener(listener);
77          }
78      }
```

org/experiment/editor/browser/AbstractBrowserTask.java

```
40      /**
41       * Creates a new browser I/O request.
42       * @param browser The VFS browser instance
43       * @param path The first path name to operate on
44       */
45      AbstractBrowserTask(VFSBrowser browser,
46          Object session, VFS vfs, String path, Runnable awtTask)
47      {
48          this.browser = browser;
49          this.session = session;
50          this.vfs = vfs;
51          this.path = path;
52          if (awtTask != null)
53          {
54              MyTaskListener listener = new MyTaskListener(awtTask);
55              TaskManager.instance.addTaskListener(listener);
56          }
57      }
```

Re-show introduction | Pause

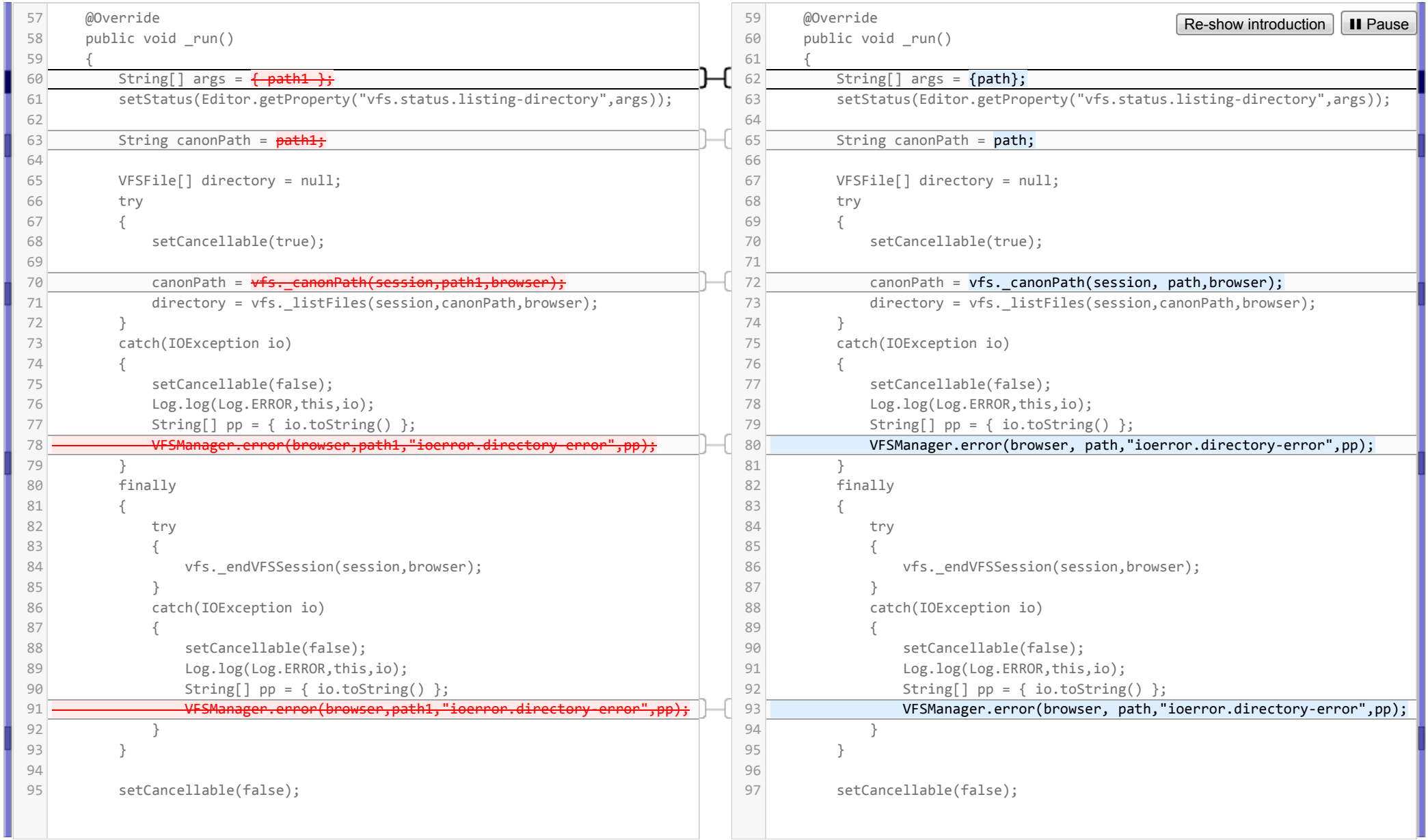
org/experiment/editor/browser/AbstractBrowserTask.java

```
81      protected VFSBrowser browser;
82      protected Object session;
83      protected VFS vfs;
84      protected String path1;
85      protected String path2;
86      protected Object[] loadInfo;
```

org/experiment/editor/browser/ListDirectoryBrowserTask.java, _run()

```
60      protected VFSBrowser browser;
61      protected Object session;
62      protected VFS vfs;
63      protected String path;
```

org/experiment/editor/browser/ListDirectoryBrowserTask.java, _run()
(Show more context)



org/experiment/editor/browser/ListDirectoryBrowserTask.java, constructor

org/experiment/editor/browser/ListDirectoryBrowserTask.java, constructor

```
42      /**
43       * Creates a new browser I/O request.
44       * @param browser The VFS browser instance
45       * @param path1 The first path name to operate on
46       * @param loadInfo A two-element array filled out by the request;
47       * element 1 is the canonical path, element 2 is the file list.
48       */
49      ListDirectoryBrowserTask(VFSBrowser browser,
50          Object session, VFS vfs, String path1,
51          Object[] loadInfo, Runnable awtRunnable)
52      {
53          super(browser, session, vfs, path1, null, loadInfo, awtRunnable);
54      }
```

```
43      /**
44       * Creates a new browser I/O request.
45       * @param browser The VFS browser instance
46       * @param path The first path name to operate on
47       * @param loadInfo A two-element array filled out by the request;
48       * element 1 is the canonical path, element 2 is the file list.
49       */
50      ListDirectoryBrowserTask(VFSBrowser browser,
51          Object session, VFS vfs, String path,
52          Object[] loadInfo, Runnable awtRunnable)
53      {
54          super(browser, session, vfs, path, awtRunnable);
55      }
```

Re-show introduction | Pause

org/experiment/editor/browser/ListDirectoryBrowserTask.java

```
42
```

org/experiment/editor/browser/ListDirectoryBrowserTask.java

```
42      private Object[] loadInfo;
```

org/experiment/editor/browser/ListDirectoryBrowserTask.java, toString()

```
102     public String toString()
103     {
104         return getClass().getName() + "[type=LIST_DIRECTORY"
105             + ",vfs=" + vfs + ",path1=" + path1 + ']';
106     }
```

org/experiment/editor/browser/ListDirectoryBrowserTask.java, toString()

```
104     public String toString()
105     {
106         return getClass().getName() + "[type=LIST_DIRECTORY"
107             + ",vfs=" + vfs + ",pat=" + path + ']';
108     }
```

org/experiment/editor/browser/RenameBrowserTask.java

```
1
```

org/experiment/editor/browser/RenameBrowserTask.java

```
1   package org.experiment.editor.browser;
2
3   import org.experiment.editor.OperatingSystem;
4   import org.experiment.editor.io.VFS;
5   import org.experiment.editor.io.VFSFile;
6   import org.experiment.editor.io.VFSManager;
7   import org.experiment.editor.Editor;
8   import org.experiment.util.Log;
9
10  import java.io.IOException;
11
12  /**
13   * @author John Doe
14   */
15  class RenameBrowserTask extends AbstractBrowserTask
16  {
17      /**
18       * Creates a new browser I/O request.
19       *
```

Re-show introduction Pause

```
     * @param browser  The VFS browser instance
     * @param path1    The first path name to operate on
     * @param path2    The second path name to operate on
     */
    RenameBrowserTask(VFSBrowser browser,
              Object session, VFS vfs, String path1, String path2,
             Runnable awtRunnable)
    {
        super(browser, session, vfs, path1, awtRunnable);
        this.path2 = path1;
    }

    @Override
    public void _run()
    {
        try
        {
            setCancellable(true);
            String[] args = {path, path2};
            setStatus(Editor.getProperty("vfs.status.renaming", args));

            path = vfs._canonPath(session, path, browser);
            path2 = vfs._canonPath(session, path2, browser);

            VFSFile file = vfs._getFile(session, path2, browser);
            if (file != null)
            {
                if ((OperatingSystem.isCaseInsensitiveFS())
                    && path.equalsIgnoreCase(path2))
                {
                    // allow user to change name
                    // case
                }
                else
                {
                    VFSManager.error(browser, path,
                        "ioerror.rename-exists",
                        new String[]{path2});
                    return;
                }
            }

            if (!vfs._rename(session, path, path2, browser))
                VFSManager.error(browser, path, "ioerror.rename-error",
                    new String[]{path2});
        }
        catch (IOException io)
        {
```

Re-show introduction Pause

```
            setCancellable(false);
            Log.log(Log.ERROR, this, io);
            String[] pp = {io.toString()};
            VFSManager.error(browser, path, "ioerror.directory-error", pp);
        }
        finally
        {
            try
            {
                vfs._endVFSSession(session, browser);
            }
            catch (IOException io)
            {
                setCancellable(false);
                Log.log(Log.ERROR, this, io);
                String[] pp = {io.toString()};
                VFSManager.error(browser, path, "ioerror.directory-error",
pp);
            }
        }
    }

    public String toString()
    {
        return getClass().getName() + "[type=RENAME"
            + ",vfs=" + vfs + ",path=" + path
            + ",path2=" + path2 + ']';
    }

    private String path2;

}
```

org/experiment/editor/browser/MkDirBrowserTask.java

```
1
```

org/experiment/editor/browser/MkDirBrowserTask.java

```
package org.experiment.editor.browser;

import org.experiment.editor.io.VFS;
import org.experiment.editor.io.VFSManager;
import org.experiment.editor.Editor;
import org.experiment.util.Log;

import java.io.IOException;

/**
 * @author John Doe
```

Re-show introduction ⏸ Pause

```
 */
class MkDirBrowserTask extends AbstractBrowserTask
{
    /**
     * Creates a new browser I/O request.
     * @param browser The VFS browser instance
     * @param path The first path name to operate on
     */
    MkDirBrowserTask(VFSBrowser browser,
             Object session, VFS vfs, String path,
             Runnable awtRunnable)
    {
        super(browser, session, vfs, path, awtRunnable);
    }

    @Override
    public void _run()
    {
        String[] args = {path};
        try
        {
            setCancellable(true);
            setStatus(Editor.getProperty("vfs.status.mkdir",args));

            path = vfs._canonPath(session, path,browser);

            if(!vfs._mkdir(session, path,browser))
                VFSManager.error(browser, path,"ioerror.mkdir-error",null);
        }
        catch(IOException io)
        {
            setCancellable(false);
            Log.log(Log.ERROR,this,io);
            args[0] = io.toString();
            VFSManager.error(browser, path,"ioerror",args);
        }
        finally
        {
            try
            {
                vfs._endVFSSession(session,browser);
            }
            catch(IOException io)
            {
                setCancellable(false);
                Log.log(Log.ERROR,this,io);
                args[0] = io.toString();
                VFSManager.error(browser, path,"ioerror",args);
```

```
            }
        }
    }

    public String toString()
    {
        return getClass().getName() + "[type=DELETE"
            + ",vfs=" + vfs + ",path=" + path + ']';
    }
}
```

Re-show introduction | Pause

org/experiment/editor/browser/BrowserIORequest.java

```
package org.experiment.editor.browser;

import java.io.*;
import org.experiment.editor.io.*;
import org.experiment.editor.*;
import org.experiment.util.*;

/**
 * A VFS browser I/O request.
 * @author Jane Doe
 */
class BrowserIORequest extends WorkRequest
{
    /**
     * Directory listing I/O request.
     */
    public static final int LIST_DIRECTORY = 0;

    /**
     * Delete file I/O request.
     */
    public static final int DELETE = 1;

    /**
     * Rename file I/O request.
     */
    public static final int RENAME = 2;

    /**
     * Make directory I/O request.
     */
    public static final int MKDIR = 3;


```

org/experiment/editor/browser/BrowserIORequest.java

```
    /**
     * Creates a new browser I/O request.
     * @param type The request type
     * @param browser The VFS browser instance
     * @param path1 The first path name to operate on
     * @param path2 The second path name to operate on
     * @param loadInfo A two-element array filled out by the request;
     * element 1 is the canonical path, element 2 is the file list.
     */
    BrowserIORequest(int type, VFSBrowser browser,
        Object session, VFS vfs, String path1, String path2,
        Object[] loadInfo)
    {
        this.type = type;
        this.browser = browser;
        this.session = session;
        this.vfs = vfs;
        this.path1 = path1;
        this.path2 = path2;
        this.loadInfo = loadInfo;
    }

    public void run()
    {
        switch(type)
        {
        case LIST_DIRECTORY:
            listDirectory();
            break;
        case DELETE:
            delete();
            break;
        case RENAME:
            rename();
            break;
        case MKDIR:
            mkdir();
            break;
        }
    }

    public String toString()
    {
        String typeString;
        switch(type)
        {
        case LIST_DIRECTORY:
            typeString = "LIST_DIRECTORY";
```

Re-show introduction | Pause

```
            break;
        case DELETE:
            typeString = "DELETE";
            break;
        case RENAME:
            typeString = "RENAME";
            break;
        case MKDIR:
            typeString = "MKDIR";
            break;
        default:
            typeString = "UNKNOWN!!!";
            break;
        }

        return getClass().getName() + "[type=" + typeString
            + ",vfs=" + vfs + ",path1=" + path1
            + ",path2=" + path2 + "]";
    }

    //Private members

    private int type;
    private VFSBrowser browser;
    private Object session;
    private VFS vfs;
    private String path1;
    private String path2;
    private Object[] loadInfo;

    private void listDirectory()
    {
        VFSFile[] directory = null;

        String[] args = { path1 };
        setStatus(Editor.getProperty("vfs.status.listing-directory",args));

        String canonPath = path1;

        try
        {
            setAbortable(true);

            canonPath = vfs._canonPath(session,path1,browser);
            directory = vfs._listFiles(session,canonPath,browser);
        }
        catch(IOException io)
        {
```

Re-show introduction | ⏸ Pause

```
131             setAbortable(false);
132             Log.log(Log.ERROR,this,io);
133             String[] pp = { io.toString() };
134             VFSManager.error(browser,path1,"ioerror.directory-error",pp);
135         }
136         catch(WorkThread.Abort a)
137         {
138         }
139         finally
140         {
141             try
142             {
143                 vfs._endVFSSession(session,browser);
144             }
145             catch(IOException io)
146             {
147                 setAbortable(false);
148                 Log.log(Log.ERROR,this,io);
149                 String[] pp = { io.toString() };
150                 VFSManager.error(browser,path1,"ioerror.directory-error",pp);
151             }
152         }
153
154         setAbortable(false);
155
156         loadInfo[0] = canonPath;
157         loadInfo[1] = directory;
158     }
159
160     private void delete()
161     {
162         try
163         {
164             setAbortable(true);
165             String[] args = { path1 };
166             setStatus(Editor.getProperty("vfs.status.deleting",args));
167
168             try
169             {
170                 path1 = vfs._canonPath(session,path1,browser);
171
172
173                 if(!vfs._delete(session,path1,browser))
174                     VFSManager.error(browser,path1,"ioerror.delete-
    error",null);
175             }
176             catch(IOException io)
177             {
```

Re-show introduction | ⏸ Pause

```
178                 setAbortable(false);
179                 Log.log(Log.ERROR,this,io);
180                 String[] pp = { io.toString() };
181                 VFSManager.error(browser,path1,"ioerror.directory-error",pp);
182             }
183         }
184         catch(WorkThread.Abort a)
185         {
186         }
187         finally
188         {
189             try
190             {
191                 vfs._endVFSSession(session,browser);
192             }
193             catch(IOException io)
194             {
195                 setAbortable(false);
196                 Log.log(Log.ERROR,this,io);
197                 String[] pp = { io.toString() };
198                 VFSManager.error(browser,path1,"ioerror.directory-error",pp);
199             }
200         }
201     }
202
203     private void rename()
204     {
205         try
206         {
207             setAbortable(true);
208             String[] args = { path1, path2 };
209             setStatus(Editor.getProperty("vfs.status.renaming",args));
210
211             try
212             {
213                 path1 = vfs._canonPath(session,path1,browser);
214                 path2 = vfs._canonPath(session,path2,browser);
215
216                 VFSFile file = vfs._getFile(session,path2,browser);
217                 if(file != null)
218                 {
219                     if((OperatingSystem.isCaseInsensitiveFS())
220                         && path1.equalsIgnoreCase(path2))
221                     {
222                         // allow user to change name
223                         // case
224                     }
225                     else
```

Re-show introduction | ⏸ Pause

```
226                     {
227                         VFSManager.error(browser,path1,
228                             "ioerror.rename-exists",
229                             new String[] { path2 });
230                         return;
231                     }
232                 }
233
234                 if(!vfs._rename(session,path1,path2,browser))
235                     VFSManager.error(browser,path1,"ioerror.rename-error",
236                         new String[] { path2 });
237             }
238             catch(IOException io)
239             {
240                 setAbortable(false);
241                 Log.log(Log.ERROR,this,io);
242                 String[] pp = { io.toString() };
243                 VFSManager.error(browser,path1,"ioerror.directory-error",pp);
244             }
245         }
246         catch(WorkThread.Abort a)
247         {
248         }
249         finally
250         {
251             try
252             {
253                 vfs._endVFSSession(session,browser);
254             }
255             catch(IOException io)
256             {
257                 setAbortable(false);
258                 Log.log(Log.ERROR,this,io);
259                 String[] pp = { io.toString() };
260                 VFSManager.error(browser,path1,"ioerror.directory-error",pp);
261             }
262         }
263     }
264
265     private void mkdir()
266     {
267         try
268         {
269             setAbortable(true);
270             String[] args = { path1 };
271             setStatus(Editor.getProperty("vfs.status.mkdir",args));
272
273             try
```

Re-show introduction | ⏸ Pause

```
274             {
275                 path1 = vfs._canonPath(session,path1,browser);
276
277                 if(!vfs._mkdir(session,path1,browser))
278                     VFSManager.error(browser,path1,"ioerror.mkdir-
error",null);
279             }
280             catch(IOException io)
281             {
282                 setAbortable(false);
283                 Log.log(Log.ERROR,this,io);
284                 args[0] = io.toString();
285                 VFSManager.error(browser,path1,"ioerror",args);
286             }
287         }
288         catch(WorkThread.Abort a)
289         {
290         }
291         finally
292         {
293             try
294             {
295                 vfs._endVFSSession(session,browser);
296             }
297             catch(IOException io)
298             {
299                 setAbortable(false);
300                 Log.log(Log.ERROR,this,io);
301                 String[] args = { io.toString() };
302                 VFSManager.error(browser,path1,"ioerror",args);
303             }
304         }
305     }
306
307 }
```

Re-show introduction | Pause

org/experiment/editor/browser/DeleteBrowserTask.java

```
1
```

org/experiment/editor/browser/DeleteBrowserTask.java

```
1 package org.experiment.editor.browser;
2
3 import org.experiment.editor.io.VFS;
4 import org.experiment.editor.io.VFSManager;
```

Re-show introduction Pause

```
import org.experiment.editor.Editor;
import org.experiment.util.Log;

import java.io.IOException;

/**
 * @author John Doe
 */
class DeleteBrowserTask extends AbstractBrowserTask
{
    /**
     * Creates a new browser I/O request.
     *
     * @param browser The VFS browser instance
     * @param path    The first path name to operate on
     */
    DeleteBrowserTask(VFSBrowser browser,
              Object session, VFS vfs, String path)
    {
        super(browser, session, vfs, path, null);
    }

    @Override
    public void _run()
    {
        try
        {
            setCancellable(false);
            String[] args = {path};
            setStatus(Editor.getProperty("vfs.status.deleting", args));

            path = vfs._canonPath(session, path, browser);


            if (!vfs._delete(session, path, browser))
                VFSManager.error(browser, path, "ioerror.delete-error",
null);
        }
        catch (IOException io)
        {
            setCancellable(false);
            Log.log(Log.ERROR, this, io);
            String[] pp = {io.toString()};
            VFSManager.error(browser, path, "ioerror.directory-error", pp);
        }
        finally
        {
            try
```

```
52              {
53                  vfs._endVFSSession(session, browser);
54              }
55              catch (IOException io)
56              {
57                  setCancellable(false);
58                  Log.log(Log.ERROR, this, io);
59                  String[] pp = {io.toString()};
60                  VFSManager.error(browser, path, "ioerror.directory-error",
pp);
61              }
62          }
63      }
64
65      public String toString()
66      {
67          return getClass().getName() + "[type=DELETE"
68              + ",vfs=" + vfs + ",path=" + path +
69              ']';
70      }
71  }
```

Re-show introduction | Pause

org/experiment/editor/browser/VFSBrowser.java, delete()

```
696
697         if(!startRequest())
698             return;
699
700         for(int i = 0; i < files.length; i++)
701         {
702             Object session =
703                 vfs.createVFSSession(files[i].getDeletePath(),this);
704             if(session == null)
705             {
706                 continue;
707             }
708
709             VFSManager.runInWorkThread(new BrowserIORequest(
710                 BrowserIORequest.DELETE,this,
711                 session,vfs,files[i].getDeletePath(),
712                 null,null));
713         }
714
715         // Do not change this until all VFS Browser tasks are
716         // done in ThreadUtilities
717         VFSManager.runInAWTThread(new Runnable()
718         {
719             public void run()
```

org/experiment/editor/browser/VFSBrowser.java, delete() (Show more context)

```
696
697         if(!startRequest())
698             return;
699
700         final CountDownLatch latch = new CountDownLatch(files.length);
701         for(int i = 0; i < files.length; i++)
702         {
703             Object session =
704                 vfs.createVFSSession(files[i].getDeletePath(),this);
705             if(session == null)
706             {
707                 continue;
708             }
709
710             final Task task = new DeleteBrowserTask(
711                     this, session, vfs, files[i].getDeletePath());
712             TaskManager.instance.addTaskListener(new TaskAdapter()
713             {
714                 @Override
715                 public void done(Task t)
716                 {
717                     if (task == t)
718                     {
719                         latch.countDown();
```

```
720             {
721                 endRequest();
722             }
723         });
724     }
```

```
720                         TaskManager.instance.removeTaskListener(this);
721                     }
722                 }
723             });
724             ThreadUtilities.runInBackground(task);
725         }
726
727         try
728         {
729             latch.await();
730         }
731         catch (InterruptedException e)
732         {
733             Log.log(Log.ERROR, this, e, e);
734         }
735
736         EventQueue.invokeLater(new Runnable()
737         {
738             @Override
739             public void run()
740             {
741                 endRequest();
742             }
743         });
744     }
```

Re-show introduction | Pause

org/experiment/editor/browser/VFSBrowser.java, rename()

```
740
741         if(!startRequest())
742             return;
743
744         VFSManager.runInWorkThread(new BrowserIORequest(
745             BrowserIORequest.RENAME,this,
746             session,vfs,from,to,null));
747
748         // Do not change this until all VFS Browser tasks are
749         // done in ThreadUtilities
750         VFSManager.runInAWTThread(new Runnable()
751         {
752             public void run()
753             {
754                 endRequest();
755             }
756         });
757     }
```

org/experiment/editor/browser/VFSBrowser.java, rename() (Show more context)

```
762
763         if(!startRequest())
764             return;
765
766         Runnable delatedAWT = new Runnable()
767         {
768             @Override
769             public void run()
770             {
771                 endRequest();
772             }
773         };
774         Task renameTask = new RenameBrowserTask(
775             this, session, vfs, to, from, delatedAWT);
776         ThreadUtilities.runInBackground(renameTask);
777     }
```

Re-show introduction | ⏸ Pause

org/experiment/editor/browser/VFSBrowser.java, rename()

```
775
776         if(!startRequest())
777             return;
778
779         VFSManager.runInWorkThread(new BrowserIORequest(
780             BrowserIORequest.RENAME,this,
781             session,vfs,from,to,null));
782
783         // Do not change this until all VFS Browser tasks are
784         // done in ThreadUtilities
785         VFSManager.runInAWTThread(new Runnable()
786         {
787             public void run()
788             {
789                 endRequest();
790             }
791         });
792     }
```

org/experiment/editor/browser/VFSBrowser.java, rename() (Show more context)

```
794
795         if(!startRequest())
796             return;
797
798         Runnable delayedAWT = new Runnable()
799         {
800             @Override
801             public void run()
802             {
803                 endRequest();
804             }
805         };
806         Task task = new RenameBrowserTask(
807             this, session, vfs, to, from, delayedAWT);
808         ThreadUtilities.runInBackground(task);
809     }
```

org/experiment/editor/browser/VFSBrowser.java, mkdir()

org/experiment/editor/browser/VFSBrowser.java, mkdir() (Show more context)

```
825
826         if(!startRequest())
827             return;
828
829         VFSManager.runInWorkThread(new BrowserIORequest(
830             BrowserIORequest.MKDIR,this,
831             session,vfs,newDirectory,null,null));
832
833         // Do not change this until all VFS Browser tasks are
834         // done in ThreadUtilities
835         VFSManager.runInAWTThread(new Runnable()
836         {
837             public void run()
838             {
839                 endRequest();
840                 if (selected.length != 0
841                     && selected[0].getType() != VFSFile.FILE)
842                 {
843                     VFSDirectoryEntryTable directoryEntryTable =
844                         browserView.getTable();
845                     int selectedRow =
846                         directoryEntryTable.getSelectedRow();
847                     VFSDirectoryEntryTableModel model =
848                         (VFSDirectoryEntryTableModel)
849                             directoryEntryTable.getModel();
850                     VFSDirectoryEntryTableModel.Entry entry =
851                         model.files[selectedRow];
852                     if (!entry.expanded)
853                     {
854                         browserView.clearExpansionState();
855                         browserView.loadDirectory(
856                             entry, entry.dirEntry.getPath(), false);
857                     }
858                 }
859             }
860         });
861     }
```

```
841
842         if(!startRequest())
843             return;
844
845         Runnable runnable = new Runnable()
846         {
847             @Override
848             public void run()
849             {
850                 endRequest();
851                 if (selected.length != 0
852                     && selected[0].getType() != VFSFile.FILE)
853                 {
854                     VFSDirectoryEntryTable directoryEntryTable =
855                         browserView.getTable();
856                     int selectedRow =
857                         directoryEntryTable.getSelectedRow();
858                     VFSDirectoryEntryTableModel model =
859                         (VFSDirectoryEntryTableModel)
860                             directoryEntryTable.getModel();
861                     VFSDirectoryEntryTableModel.Entry entry =
862                         model.files[selectedRow];
863                     if (!entry.expanded)
864                     {
865                         browserView.clearExpansionState();
866                         browserView.loadDirectory(
867                             entry, entry.dirEntry.getPath(), false);
868                     }
869                 }
870             }
871         };
872         Task mkdirTask = new MkDirBrowserTask(
873             this, session, vfs, newDirectory, runnable);
874         ThreadUtilities.runInBackground(mkdirTask);
875     }
```

Re-show introduction | ⏸ Pause

We would now like to ask some questions on the change you just reviewed:

Which of these statements applies?

- ○ Several AbstractBrowserTasks were combined into BrowserIORequest
- ○ The functionality of BrowserIORequest was moved to several subclasses of AbstractBrowserTask
- ○ I don't know

Re-show introduction | ⏸ Pause

The "loadInfo" array is used to ...

- (•) transfer results from MkDirBrowserTask back to the caller
- ( ) transfer results from ListDirectoryBrowserTask back to the caller
- ( ) provide settings for ListDirectoryBrowserTask
- ( ) provide settings for MkDirBrowserTask
- ( ) I don't know

The method for deletion (in VFSBrowser) ...

- ( ) deletes one file after another when multiple files are given
- ( ) does not allow deleting more than one file
- ( ) deletes files in parallel when multiple files are given
- ( ) I don't know

What is true about renaming?

- ( ) There are three rename methods in VFSBrowser: One for a known target filename, one for asking the user for the new filename and one for changing the case of a file on case insensitive file systems. All three call the same code in RenameBrowserTask.
- ( ) There are two rename methods in VFSBrowser: One for a known target filename and one for asking the user for the new filename. They delegate to different subclasses of AbstractBrowserTask.
- ( ) There is one method in VFSBrowser to handle renaming.
- ( ) There are two rename methods in VFSBrowser: One for a known target filename and one for asking the user for the new filename. They delegate to different code in RenameBrowserTask.
- ( ) There are two rename methods in VFSBrowser: One for a known target filename and one for asking the user for the new filename. Both call the same code in RenameBrowserTask.
- ( ) I don't know

Continue ►