

TableFrag.java

```
1 package com.NAI.corelog.logViews;
2
3 import com.NAI.utils.UCursor.Pointer;
4
5 import android.app.Fragment;
6 import android.os.Bundle;
7 import android.util.Log;
8 import android.view.LayoutInflater;
9 import android.view.View;
10 import android.view.ViewGroup;
11
12 /**Base Class for individual table fragments
13 * @author Keith.Schneider
14 */
15 public abstract class TableFrag extends Fragment{
16
17     /**The Parent ViewGroup*/
18     protected ViewGroup rootView;
19
20     /** Entry Point For Code Execution*/
21     @Override
22     public void onActivityCreated(Bundle savedInstanceState) {
23         super.onActivityCreated(savedInstanceState);
24         getRootTable();
25     }
26
27     /**Get a reference to a database pointer
28      * @return {@link Pointer} to the appropriate table */
29     public abstract Pointer getPointer();
30
31     /** For Future Use */
32     private void getRootTable() {
33         //Do Nothing
34     }
35
36     /**Instantiate and display the view*/
37     @Override
38     public View onCreateView(LayoutInflater inflater, ViewGroup container,
39             Bundle savedInstanceState) {
40         setRetainInstance(false);
41         rootView = (ViewGroup)inflater.inflate(getTableView(), container,false);
42         //Call to abstract method to set view wrappers for components.
43         this.setWrappers();
44         return rootView;
45     }
46
47     /**Called when the view is about to be destroyed*/
48     @Override
49     public void onDestroyView() {
50         Log.d(this.getClass().toString(), "Destroy View Called");
51         super.onDestroyView();
52     }
53
54     /**Called when the fragment is about to be destroyed*/
55     @Override
56     public void onDestroy() {
```

TableFrag.java

```
57     Log.d(this.getClass().toString(), "Destroy Called");
58     super.onDestroy();
59 }
60
61
62
63     /**Getter for the rootView Reference*/
64     public ViewGroup getRootView() {
65         return rootView;
66     }
67
68     /**Save any state information prior too destruction*/
69     @Override
70     public void onSaveInstanceState(Bundle outState) {
71         super.onSaveInstanceState(outState);
72     }
73
74     /** save all pending changes to database */
75     public void saveAll(){
76         getPointer().saveAll();
77     }
78
79     /**Set any wrapper classes or features for UI objects*/
80     abstract protected void setWrappers();
81
82     /** @return A reference integer that maps to the view resource.*/
83     protected abstract int getTableView();
84
85 }
86
```

Oxide.java

```
1 package com.NAI.corelog.logViews;
2
3 import android.support.v4.widget.SimpleCursorAdapter;
4
5 /**
6  * This is a fragment class that specifies how to connect a table to a view
7  * and how it should behave.
8  * @author Keith.Schneider
9  */
10 public class Oxide extends TableFrag{
11
12     Pointer pointer;
13
14     /**Return a reference to the appropriate View*/
15     @Override
16     protected int getTableView() {
17         return R.layout.oxide;
18     }
19
20     protected void setWrappers()
21     {
22         //Get a reference to the appropriate table (Singleton object)
23         pointer = ((LogView)this.getActivity()).coredb.getTable("Oxide").getPointer();
24
25         //Drop any previously registered observers
26         //Often occurs when the view is destroyed and recreated.
27         //Which happens every time the view orientation changes.
28         pointer.dropObservers();
29
30         //Reference to the side bar
31         ListView lv = (ListView)rootView.findViewById(R.id.oxide_ft_sidebar);
32
33         //Connects editable text fields to the database.
34         new
35             EditTextConnect((EditText)rootView.findViewById(R.id.oxide_ft_from_et),pointer,"From_m");
36         );
37         new
38             EditTextConnect((EditText)rootView.findViewById(R.id.oxide_ft_to_et),pointer,"To_m");
39
40             //Adds QC functionality for From/To interval fields
41             new FromETWidget((EditText)rootView.findViewById(R.id.oxide_ft_from_et),
42                 getActivity());
43             new ToETWidget((EditText)rootView.findViewById(R.id.oxide_ft_to_et),
44                 getActivity());
45
46             new
47                 EditTextConnect((EditText)rootView.findViewById(R.id.oxide_comment_et),pointer,"Oxidation_Comments");
48
49             //Enables quick look up and paste functionality based on previous entries
50             new LookupWidget(
51                 (EditText)rootView.findViewById(R.id.oxide_comment_et),
52                 getActivity(),
53                 "Select Oxidation_Comments, _id From Oxide WHERE Oxidation_Comments IS
54                 NOT NULL ORDER BY _id DESC;",
55                 "Oxidation_Comments");
56
57
58
59
60
61
62
63
64
65
66
67
68
```

Oxide.java

```
69
70     //Navigate through records in the data set
71     new NextNavButton((Button)rootView.findViewById(R.id.oxide_next_btn),pointer);
72     new PrevNavButton((Button)rootView.findViewById(R.id.oxide_prev_btn),pointer);
73     new
74         CreateNewFTRecordBtn((Button)rootView.findViewById(R.id.oxide_new_btn),pointer);
75
76     //Widget that takes photos and associates them with the current record
77     MultiPhotoWidget photoWidget = new MultiPhotoWidget(
78         (ImageButton)rootView.findViewById(R.id.oxide_photo_button),
79         getActivity(),
80         "Oxide",
81         pointer,
82         new String[] {"From_m", "To_m"},
83         PhotoWidget.AUTO_INSERT_M,
84         PhotoWidget.USE_HOLE_LOGGER_ID);
85     //Refresh the photo widget
86     photoWidget.onRecordChanged();
87
88     //Connect spinner widgets with the database and populate their lookup tables.
89     new SpinnerConnect(
90         (Spinner)rootView.findViewById(R.id.oxide_oxidation_state_sp),
91         ((LogView)this.getActivity()).coredb,
92         pointer,
93         "Oxidation_Intensity",
94         "Oxidation_State_Of_Sulfide");
95     new SpinnerConnect(
96         (Spinner)rootView.findViewById(R.id.oxide_limonite_style_sp),
97         ((LogView)this.getActivity()).coredb,
98         pointer,
99         "Alt_Style",
100        "Limonite_style");
101    new SpinnerConnect(
102        (Spinner)rootView.findViewById(R.id.oxide_limonite_int_sp),
103        ((LogView)this.getActivity()).coredb,
104        pointer,
105        "Intensity",
106        "Limonite_int");
107    new SpinnerConnect(
108        (Spinner)rootView.findViewById(R.id.oxide_hematite_style_sp),
109        ((LogView)this.getActivity()).coredb,
110        pointer,
111        "Alt_Style",
112        "Hematite_style");
113    new SpinnerConnect(
114        (Spinner)rootView.findViewById(R.id.oxide_hematite_int_sp),
115        ((LogView)this.getActivity()).coredb,
116        pointer,
117        "Intensity",
118        "Hematite_int");
119    new SpinnerConnect(
120        (Spinner)rootView.findViewById(R.id.oxide_mnox_style_sp),
121        ((LogView)this.getActivity()).coredb,
122        pointer,
123        "Alt_Style",
124        "MnOx_style");
```

Oxide.java

```
124     new SpinnerConnect(
125         (Spinner)rootView.findViewById(R.id.oxide_mnox_int_sp),
126         ((LogView)this.getActivity()).coredb,
127         pointer,
128         "Intensity",
129         "MnOx_int");
130
131     //This sets the side panel view and allows for touch navigation
132     new FTListView(
133         getActivity(),
134         R.layout.oxide_list_item,
135         new String[]{"From_m", "To_m",
136             "Oxidation_State_Of_Sulfide", "Limonite_style", "Hematite_style", "MnOx_style"},
137         new int[] {R.id.oxide_list_from_tv,
138             R.id.oxide_list_to_tv,
139             R.id.oxide_list_ox_state_tv,
140             R.id.oxide_list_lim_style_tv,
141             R.id.oxide_list_hem_style_tv,
142             R.id.oxide_list_mnox_style_tv},
143         SimpleCursorAdapter.FLAG_REGISTER_CONTENT_OBSERVER,
144         lv,
145         pointer,
146         ((LogView)this.getActivity()).coredb.getTable("Oxide")));
147
148
149 /**
150 * @return {@link Pointer} return a reference to the database pointer
151 */
152 @Override
153 public Pointer getPointer() {
154     return pointer;
155 }
156 }
```