［戻る］ Crazy Unit

複数のGtkTreeViewで一つのモデルを別々のソート順で表示する

モデルデータを共有し、ソート項目・順序はそれぞれ別のものを使用します。
ちょっと凝ったことをするにはいろいろ気をつけないといけないようです。

GtkTreeModelSortを使用したモデルデータの共有

モデルデータを共有するだけであれば複数の**GtkTreeView**で同じモデルを使用するだけで済むのですが、その場合はソート項目と順序も同じになってしまいます。
同じデータを使用するけれどソート順は別にしたい場合には**GtkTreeView**毎に同じモデルを持つ**GtkTreeModelSort**を作成して、それを**GtkTreeModel**として使用します。
この**GtkTreeModelSort**が間に入ることにより(プロクシとして動作し)データ共有と別々のソート順の設定を可能にします。

サンプルです。gtktreemodelsort.c
最初にボタンだけを持ったウィンドウを表示します。このボタンを押す毎に、追加・削除ボタンと**GtkTreeView**を持ったウィンドウを表示します。
追加ボタンを押すと**GtkTreeView**に連続した数と乱数の組を1行追加し、削除ボタンで選択中の行を削除します。
GtkTreeViewの項目ヘッダをクリックするとソート項目・順序を変更できます。
全てのウィンドウで同じモデルを使用しているので、1つのウィンドウで追加・削除をした結

果が他のウィンドウにも反映されますが、ソート項目・順序はウィンドウ毎に別の設定が可能になっています。
文字コードはUTF-8です。

[TOP]

GtkTreeModelSortの作成

GtkTreeModelSortの作成は下記関数を呼び出すだけです。

```
GtkTreeModel * gtk_tree_model_sort_new_with_model ( GtkTreeModel *child_model );
```

child_modelにあらかじめ作成しておいた**GtkListStore**や**GtkTreeStore**を渡します。
サンプルでは**create_tree_view_window()**の最初でこれを行っています。
GtkTreeViewに作成した**GtkTreeModelSort**を**GtkTreeModel**として渡した後、**GtkTreeView**の作成はこれといって特別なことをしていません。

[TOP]

モデルデータの取得

GtkTreeModelSortも**GtkTreeModel**インターフェイルを実装しているので、**GtkTreeView**にセットした後はいつもと同じように**gtk_tree_view_get_model()**や**gtk_tree_selection_get_selected_rows**でセットした**GtkTreeModelSort**を取得し、**gtk_tree_model_get()**で**GtkTreeModelSort**から指定した行と列のデータを取得できます。

サンプルでも**GtkTreeSelection**の**changed**シグナルハンドラ内で**GtkTreeModelSort**から数値データを取得しています。
ただし、**GtkTreeSelection**の**changed**シグナルハンドラを使用する場合は少し注意が必要です。

[TOP]

行の追加

モデルにデータを追加するときは、**GtkTreeModelSort**から内部のモデルを取得し、そこに対して追加処理を行います。

```
GtkTreeModel * gtk_tree_model_sort_get_model ( GtkTreeModelSort *tree_model );
```

モデルを取りだしたらそれを実際の型、例えば**GtkListStore**や**GtkTreeStore**に変換して、そのデータ追加関数を使用すれば、全ての**GtkTreeView**に変更が反映されます。
サンプルでは追加ボタンの**clicked**シグナルハンドラ**add_model_data()**

で**gtk_tree_model_sort_get_model()**を使用し、取得したモデルで**gtk_list_store_append()**と**gtk_list_store_set()**を呼んで新規行の追加を行っています。

[TOP]

行の削除

モデルからデータを削除する場合は少し複雑になります。
削除データを指定するには**GtkTreeView**が表示している行の**GtkTreePath**や**GtkTreeIter**を使用することになると思いますが、今回これらは**GtkTreeModelSort**の行を指しています。
実際の削除はその内側にある**GtkTreeModel**に対して行う必要があり、**GtkTreeModelSort**の行を指す**GtkTreePath**や**GtkTreeIter**はそのまま使用できません。
そこで以下の変換関数を使用して、**GtkTreePath**あるいは**GtkTreeIter**を**GtkTreeModelSort**のものから、内側の**GtkTreeModel**のものへと変換します。

```
GtkTreePath * gtk_tree_model_sort_convert_path_to_child_path ( GtkTreeModelSort *tree_model_sort,
                                                               GtkTreePath      *sorted_path );

void          gtk_tree_model_sort_convert_iter_to_child_iter ( GtkTreeModelSort *tree_model_sort,
                                                               GtkTreeIter      *child_iter,
                                                               GtkTreeIter      *sorted_iter );
```

gtk_tree_model_sort_convert_path_to_child_path()の**sorted_path**に**GtkTreeModelSort**の**GtkTreePath**を渡すと、**GtkTreeModelSort**の内側のモデルの**GtkTreePath**が返されます。
gtk_tree_model_sort_convert_iter_to_child_iter()の場合は**sorted_iter**に**GtkTreeModelSort**の**GtkTreeIter**を渡すと、**child_iter**に変換された**GtkTreeIter**がセットされます。

サンプルでは削除ボタンの**clicked**シグナルハンドラ**remove_model_data()**で**gtk_tree_selection_get_selected_rows()**で取得した**GtkTreePath**を、実際のモデルを指すものへ変換しています。
また、データの削除を行うと取得済の**GtkTreePath**は無効になるので、削除対象の行を全て**GtkTreeRowReference**に変換して、削除する直前に**GtkTreePath**に戻すという事をしています。
実際のモデルの**GtkTreePath**が取得できたら後は**GtkTreeIter**に変換して、**gtk_list_store_remove()**を使用すれば同じモデルを使用している**GtkTreeModelSort**を表示している**GtkTreeView**に変更が反映されます。

サンプルでは削除の前後で**GtkTreeSelection**の**changed**シグナルハンドラの呼び出しを一時的にブロックしています。
これは注意点で説明します。

内側のGtkTreeModelにセットしたモデルの行を指すGtkTreePathやGtkTreeIterからGtkTreeModelSortのものへ変換するには以下の関数が使用できます。

```
GtkTreePath * gtk_tree_model_sort_convert_child_path_to_path ( GtkTreeModelSort *tree_model_sort,
                                                               GtkTreePath      *child_path );

void          gtk_tree_model_sort_convert_child_iter_to_iter ( GtkTreeModelSort *tree_model_sort,
                                                               GtkTreeIter      *sort_iter,
                                                               GtkTreeIter      *child_iter );
```

gtk_tree_model_sort_convert_child_path_to_path()はchild_pathに実際のモデルを指すGtkTreePathをセットするとGtkTreeModelSortのGtkTreePathが返されます。
gtk_tree_model_sort_convert_child_iter_to_iter()はchild_iterに実際のモデルのGtkTreeIterを渡して呼びだすとsort_iterに変換されたものがセットされます。
先程のものとちょっと名前が違うだけなので慣れないとどちらを使えば良いのか迷うかもしれません。

[TOP]

注意点

GtkTreeModelFilterのときにも似たような現象に遭遇したのですが、 GtkTreeSelectionのchangedシグナルハンドラを使用しているとき、 別のGtkTreeViewで行なった行削除により選択状態が変更になるとchangedシグナルハンドラが呼び出されますが、 このときgtk_tree_selection_get_selected_rows()を使用すると取得したGtkTreePathのリストに無効な値が含まれるようです。
内部状態が複雑になる為か一時的に整合性の破綻した状態でシグナルハンドラが呼び出されているのかもしれません。
普通であれば取得したGtkTreePathをgtk_tree_model_get_iter()でGtkTreeIterに変換し、gtk_tree_model_get()を呼び出せばその行のデータが取得できますが、無効なGtkTreePathを使用するとコアダンプして落ちます。

やっかいな事にこの無効な値を簡単には判別できませんでした。
GtkTreeIterに変換後、gtk_tree_model_sort_iter_is_valid()で検査してもTRUEしか返らず正常な値であると判定されます。
試行錯誤したところ不正な値の場合、gtk_tree_model_sort_convert_path_to_child_path()で内側のモデルのGtkTreePathに変換し、 さらにgtk_tree_row_reference_new()でGtkTreeRowReferenceに変換しようとするとNULLが返されるようです。
しかし、このようなちょっと実装が変れば使えなくなるような方法は使うべきではないです

し、もっと簡単な方法がありました。
削除処理中には**changed**シグナルハンドラの呼び出しをブロックするだけです。

ブロックするのは同じモデルから作成した**GtkTreeModelSort**を使用している**GtkTreeView**に関連する**GtkTreeSelection**の**changed**シグナルハンドラ全てです。
サンプルでは**create_window()**の最後、**changed**シグナルハンドラを登録している所で**g_object_set_data()**を使用して **GtkTreeSelection**にシグナルハンドラのIDを、モデルに**GtkTreeSelection**のリストを登録して、後で参照できるようにしています。
ここで登録した値を削除ボタンの**clicked**シグナルハンドラで使用します。

```
void     g_object_set_data ( GObject     *object,
                             const gchar *key,
                             gpointer    data );

gpointer g_object_get_data ( GObject     *object,
                             const gchar *key );
```

g_object_set_data()で**key**に関連するデータを登録し、 **g_object_get_data()**で登録したデータを取り出します。
keyは任意の文字列を指定できます。

サンプルの削除ボタンの**clicked**シグナルハンドラでは削除対象の行の**GtkTreeRowReference**を取得した後、実際に削除する前に **g_signal_handler_block()**で全ての**changed**シグナルハンドラをブロックし、 全て削除した後に**g_signal_handler_unblock()**を呼び出してブロック解除しています。
（実際のブロックとブロック解除は**block_changed_signal()**と**unblock_changed_signal()**で行っています）

```
void g_signal_handler_block ( gpointer instance,
                              gulong   handler_id );

void g_signal_handler_unblock ( gpointer instance,
                                gulong   handler_id );
```

instanceにブロックまたはブロック解除するオジェクトを、**handler_id**に**g_signal_connect()**でハンドラ登録した時の戻り値をセットして呼び出します。
同じ**instance**と**handler_id**に対して**g_signal_handler_block()**を複数回呼び出した時には、同じ回数だけ**g_signal_handler_unblock()**を呼び出さないとブロック解除されません。

最後に、ブロック解除した後、ブロックしている間に変更された選択状態を反映させる為、**GtkTreeSelection**の**changed**シグナルハンドラを直接呼び出しています。

この症状は環境依存かもしれません。
FreeBSD 7.1-RELEASE-p5でportsコレクションからインストールしたGTK+ 2.16.1で発生しました。 GTK+が同じバージョンでも他の環境では発生しないかもしれません。

[TOP]

GtkIconViewでGtkTreeModelSortを使用すると…

GtkIconViewで**GtkTreeModelSort**を使用したところ、削除の反映が正しく行なわれませんでした。
見かけ上は最後のアイテムがいくつか複製された状態になりますが、実際には存在しないアイテムなので値の取得や削除はできません。
GtkTreeViewに比べて**GtkIconView**はまだまだといった感はありますが、ここでもそういった結果になりました。
解決方法は調べていません。
サンプルです。gtktreemodelsort_iconview.c

この症状は環境依存かもしれません。
FreeBSD 7.1-RELEASE-p5でportsコレクションからインストールしたGTK+ 2.16.1で発生しました。 GTK+が同じバージョンでも他の環境では発生しないかもしれません。

[TOP]

[戻る]