Add / Delete rows in Table and Edit using Cell editors

Assume that you already know how to create a table and table viewer. In this tutorial we will learn about add a row in table, cell editor, cell modifier and function of modifier.

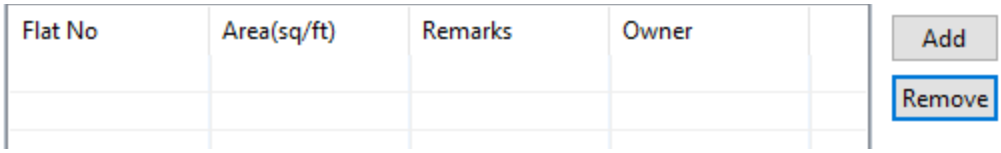
Add a row in table:

Step 1: add a button.

Add two button on your UI this button is responsible for add new row and delete a row from your table.

You can use the following code for add buttons

```
Button btnAdd = new Button(composite_1, SWT.NONE);
btnAdd.setLayoutData(new GridData(SWT.FILL, SWT.CENTER, false, false, 1, 1));
managedForm.getToolkit().adapt(btnAdd, true, true);
btnAdd.setText("Add");
```




```
Button btnRemove = new Button(composite_1, SWT.NONE);
btnRemove.setLayoutData(new GridData(SWT.FILL, SWT.TOP, false, false, 1, 1));
managedForm.getToolkit().adapt(btnRemove, true, true);
btnRemove.setText("Remove");
```

Step 2: add listener

Add row:-Now you have to add a listener to your button for listen your actions. Add AddSelectionListener() listener with "new SelectionAdapter()" parameter →add widgetSelected() method →set a default values in column → refresh your table viewer.

```
btnAdd.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e) {
        // TODO Auto-generated method stub
        Flat flat=ApartmentFactory.eINSTANCE.createFlat();
        flat.setNumber("00");
        flat.setSqrFeet("0");
        flat.setRemarks("remarks");
        Owner owner=ApartmentFactory.eINSTANCE.createOwner();
        owner.setName("Owner Name");
        flat.setOwner(owner);
        apartment2.getFlat().add(flat);

        tableViewer.refresh();
        super.widgetSelected(e);
    }
});
```

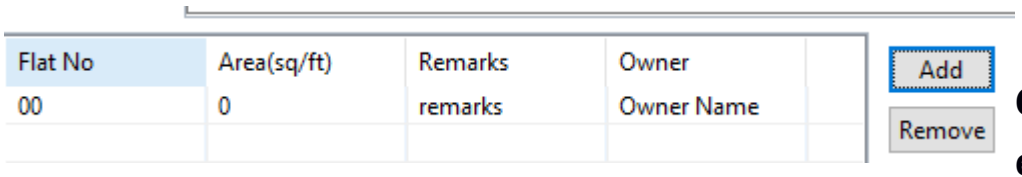
Default values for row

Author : Vikash Singh
Date : 28th October 2016

Delete row:-Now you have to add a listener to your button for listen your actions. Add AddSelectionListener() listener with "new SelectionAdapter()" parameter →add widgetSelected() method →take selection containing elements→remove the selected element→ refresh your table viewer.

```
btnRemove.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent e) {
        IStructuredSelection selection = (IStructuredSelection) tableViewer.getSelection();
        if(!selection.isEmpty()){
            Iterator iterator = selection.iterator();
            while (iterator.hasNext()) {
                Object object =  iterator.next();
                apartment2.getFlat().remove(object);
            }
            tableViewer.refresh();
        }
        // TODO Auto-generated method stub
        super.widgetSelected(e);
    }
});
```

Step 3: output.



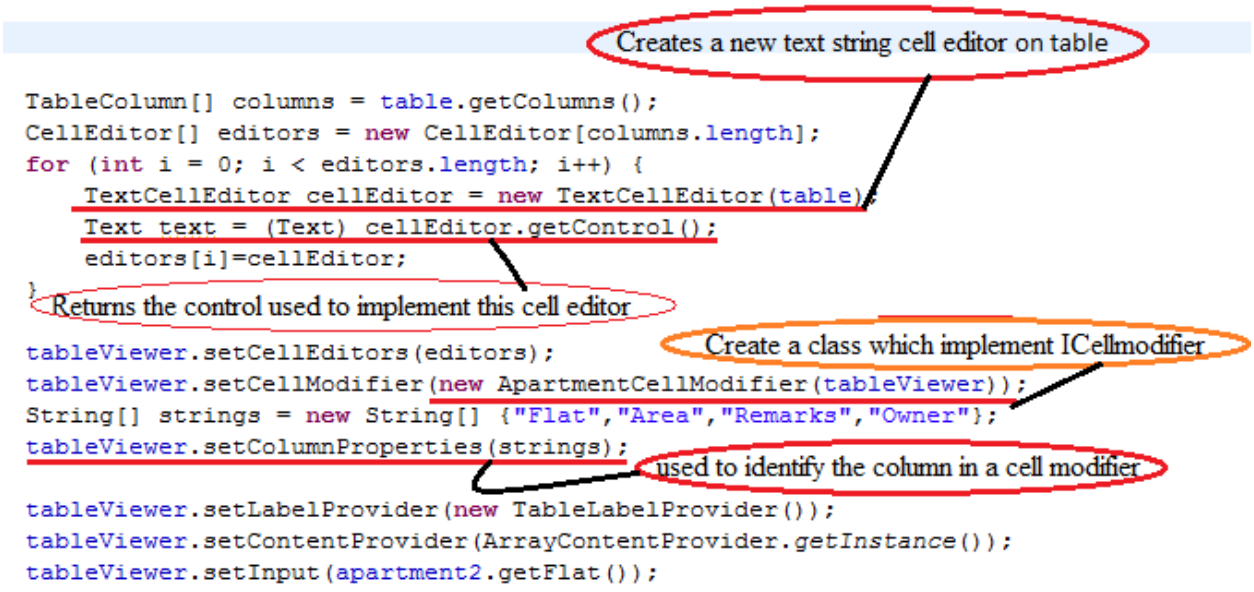
Cell editor and modifier:

Step 1: Add cell editor and cell modifier to table.

Find the table column length→get the cell control using a loop on columns → set cell editor in table viewer → set cell modifier in table viewer → set column properties in table viewer.

Note:- the cell modifier class implements ICellModifier we pass table viewer as a parameter of constructor

use setLabelProvider,SetContentProvider and SetInput methods after cell editor and cell modifier .




```
TableColumn[] columns = table.getColumns();
CellEditor[] editors = new CellEditor[columns.length];
for (int i = 0; i < editors.length; i++) {
    TextCellEditor cellEditor = new TextCellEditor(table);
    Text text = (Text) cellEditor.getControl();
    editors[i]=cellEditor;
}
tableViewer.setCellEditors(editors);
tableViewer.setCellModifier(new ApartmentCellModifier(tableViewer));
String[] strings = new String[] {"Flat","Area","Remarks","Owner"};
tableViewer.setColumnProperties(strings);

tableViewer.setLabelProvider(new TableLabelProvider());
tableViewer.setContentProvider(ArrayContentProvider.getInstance());
tableViewer.setInput(apartment2.getFlat());
```

Step 2: Edit getValue() and modify() method of cell modifier of class.

Open your cell modifier class there is three method **canModify()**,**getvalue()**,**modify()**.

Assign your parameter to field → Set return of **canModify()** method is true → check for instance in **getValue()** method if instance matched then check for property of column and return the values → check for instance in **modify()** method for table item if instance matched then check for property of column and set the value → refresh table viewer.

```
public ApartmentCellModifier(TableViewer tableViewer) {
    this.tableViewer = tableViewer;
    // TODO Auto-generated constructor stub
}

public boolean canModify(Object element, String property) {
    // TODO Auto-generated method stub
    return true;
}
```

```
public Object getValue(Object element, String property) {
    if (element instanceof Flat ) {
        Flat flat = (Flat) element;
        if(property.equals("Flat"))
        {
            return flat.getNumber();
        }
        else if(property.equals("Area"))
        {
            return flat.getSqrFeet();
        }
        else if(property.equals("Remarks"))
        {
            return flat.getRemarks();
        }
        else if(property.equals("Owner"))
        {
            return flat.getOwner().getName();
        }
    }
    // TODO Auto-generated method stub
    return null;
```

Same string used in setproperties method

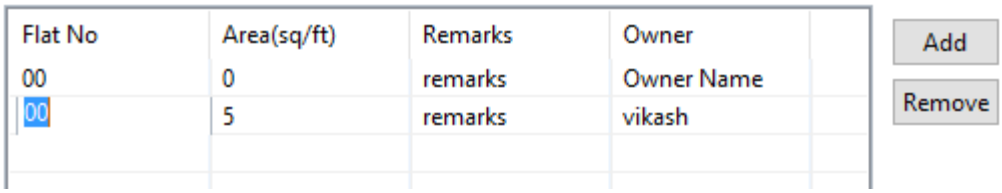
```
String[] strings = new String[] {"Flat","Area","Remarks","Owner"};
tableViewer.setColumnProperties(strings);
```

```
public void modify(Object element, String property, Object value) {
    // TODO Auto-generated method stub

    if (element instanceof TableItem) {
        Flat item = (Flat)((TableItem )element).getData();
        if (property.equals("Flat")) {
            item.setNumber(value.toString());
        }
        else if (property.equals("Area")) {
            item.setSqrFeet(value.toString());
        }
        else if (property.equals("Remarks")) {
            item.setRemarks(value.toString());
        }
        else if (property.equals("Owner")) {
            item.getOwner().setName(value.toString());
        }
        tableViewer.refresh();
    }
}
```

Author : Vikash Singh
Date : 28th October 2016

Step 3: output.



ABOUT ANCIT:

ANCIT Consulting is an Eclipse Consulting Firm located in the "Silicon Valley of Outsourcing", Bangalore. Offers professional Eclipse Support and Training for various Eclipse based Frameworks including RCP, EMF, GEF, GMF. Contact us on annamalai@ancitconsulting.com to learn more about our services.

Author : Vikash Singh
Date : 28th October 2016