Java Swing Uygulamasında MVC Yapısını Kullanmak

Java swing uygulamalarında MVC yapısını kullanmak için genelde **Observer** tasarım deseni kullanılır. Swing kütüphanesi geliştirilirken MVC'nin farklı bir implementasyonu kullanılmıştır. **Controller** ile **View** tek bir sınıfta tutulurken, **delegate** olarak bilinir, **Model** ise ayrı bir sınıf tarafından temsil edilmiştir. Aşağıdaki listede bu mantıkla yaratılmış önemli sınıflar bulunmaktadır:

Component	Model Interface	Model Type
JButton	ButtonModel	GUI
JCheckBox	ButtonModel	GUI/data
JRadioButton	ButtonModel	GUI/data
JMenu	ButtonModel	GUI
JMenuItem	ButtonModel	GUI
JCheckBoxMenuItem	ButtonModel	GUI/data
JRadioButtonMenuItem	ButtonModel	GUI/data

Dikkat ederseniz bu tabloda **Model Type** sütünü bulunmaktadır. Bu sütundaki değerler **GUI (UI), data ve bu ikisinin karışımı** şeklinde ifade edilmiştir. MVC Swing'te bir model iki farklı türde olabilmektedir: **UI-Model** ve **Data-Model.**

GUI-Model: Bir Swing component'inin, örneğin JButton, durumunu temsil eder. Örneğin focus, enablement, selection vs. Bu türe örnek olarak tabloda görüldüğü gibi **ButtonModel** sınıfını söyleyebiliriz.
Data-Model: Swing elementlerinde gösterilecek veriyi temsil eder. Örneğin bir form uygulamasında kullanıcı kayıt işlemleri yapılıyor olsun. Textbox'lara girilen değerleri tutan User isminde bir sınıf yaratabiliriz. Bu sınıf data-model olur. Data-model, view component (örn JButton)'ine bağlı değildir. Bu sayede data-model sınıfını **değiştirmeden** bir başka view component ile kullanabiliriz.

Not: Swing uygulamaları geliştirirken MVC yapısını tam anlamıyla kullanabilmek zordur. Çünkü Swing kütüphanesi yukarıda belirtildiği gibi MVC yapısının modifiye edilmiş şeklini kullanmaktadır.

Not: MVC yapısında birden fazla **View** sınıfı bulunabilir. Buna karşın tek bir tane controller ve model bulunmalıdır.

Not: Bir uygulama birden fazla MVC yapısından oluşabilir.

Örnek Uygulama

JTable nesnesinin içeriğini scroll bar'ı aşağı kaydırdıkça güncelleyen bir uygulama tasarlayacağız.

Gerekli Sınıflar

Controller Sınıfı **(TableController)**
Model Sınıfı **(TableModel)**
View Sınıfı **(TableView)**
Initialize Sınıfı **(RunProgram)**
Dao Sınıfı **(Article)**
Main Sınıfı

TableModel Sınıfı

```
1   import javax.swing.*;
2   import java.util.ArrayList;
3   import java.util.List;
4   import java.util.Observable;
5   /**
6     * Tabmodel is an observable class which notifies its observer objects
7     * This class refers to MVC Model module.
8     */
9   public class TableModel extends Observable {
10      private List<Article> articleList;
11      public void createArticleList(int firstResult, int maxResult) {
12          List<Article> articleList=getArticleList(firstResult,maxResult);
13          setChanged();
```

```
            notifyObservers(articleList);
        }
        private List<Article> getArticleList(int firstResult,int maxResult){
            int lastResult=firstResult+maxResult;
            if(lastResult>1000){
                lastResult=1000;
            }
            return articleList.subList(firstResult,lastResult);
        }
        public void initArticleList(){
            articleList=new ArrayList<Article>();
            for(int i=0; i<1000; i++){
                Article article=new Article();
                article.setId(i);
                article.setTitle(i+ ". başlık");
                article.setTitleSearch(i+". arama başlığı");
                articleList.add(article);
            }
        }
}
```

TableView Sınıfı

```
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.AdjustmentListener;
import java.util.*;
import java.util.List;
/**
  * TableView MVC'nin View modulunu temsil eder
  */
public class TableView extends JFrame implements Observer {
        private JTable table;
        private JScrollPane scrollPane;
        public TableView() {
            createTable();
        }
        private void createTable() {
            Vector<String> columnNames = new Vector<String>();
            columnNames.add("ID");
            columnNames.add("TITLE");
            columnNames.add("TITLE SEARCH");
            tableModel.setColumnIdentifiers(columnNames);
            table = new JTable(tableModel);
            table.setAutoCreateRowSorter(true);
            table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
            int screenWidth = Toolkit.getDefaultToolkit().getScreenSize().width;
            table.getColumnModel().getColumn(0).setMaxWidth(50);
            scrollPane = new JScrollPane(table);
            add(scrollPane);
            setTitle("Article List");
            setSize(screenWidth / 2, 600);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        }
        private void addRowsToTable(java.util.List<Article> articleList) {
            int i = getTableRowCount();
            for (Article tempArticle : articleList) {
                Vector<String> vector = new Vector<String>();
                vector.add(String.valueOf(tempArticle.getId()));
                vector.add(tempArticle.getTitle());
                vector.add(tempArticle.getTitleSearch());
                tableModel.insertRow(i, vector);
                i++;
            }
        }
```

```
45      //Tabloda kac tane row varsa o sayiyi donderir
46      public int getTableRowCount() {
47          return tableModel.getRowCount();
48      }
49      public void addScrollPaneAdjustmentListener(AdjustmentListener adjustmentListener) {
50          scrollPane.getVerticalScrollBar().addAdjustmentListener(adjustmentListener);
51      }
52      public int getVisibleRowCount() {
53          if (table != null) {
54              Dimension paneSize = scrollPane.getSize();
55              return paneSize.height / table.getRowHeight();
56          } else {
57              return 0;
58          }
59      }
60      //instance table model
61      DefaultTableModel tableModel = new DefaultTableModel() {
62          @Override
63          public boolean isCellEditable(int row, int column) {
64              //all cells false
65              return false;
66          }
67      };
68      @Override
69      public void update(Observable o, Object arg) {
70          if (o instanceof TableModel) {
71              List<Article> articleList = (List<Article>) arg;
72              addRowsToTable(articleList);
73          }
74      }
75  }
```

TableController Sınıfı

```
1   import javax.swing.*;
2   import java.awt.event.AdjustmentEvent;
3   import java.awt.event.AdjustmentListener;
4   /**
5     * TableController MVC'nin Controller modulunu temsil eder
6     */
7   public class TableController {
8       private TableView view;
9       private TableModel model;
10      public TableController(TableView view, TableModel model) {
11          this.view= view;
12          this.model= model;
13          this.view.addScrollPaneAdjustmentListener(new ScrollPaneAdjustmentListener());
14          this.model.initArticleList();
15          //Ilk olarak 40 tane item gosterilmesi saglanir. 40 yerine dilediginiz ilk degeri
16  verebilirsiniz
17          this.model.createArticleList(0, 40);
18          view.setVisible(true);
19      }
20      private class ScrollPaneAdjustmentListener implements AdjustmentListener {
21          @Override
22          public void adjustmentValueChanged(AdjustmentEvent e) {
23              JScrollBar scrollBar = (JScrollBar) e.getAdjustable();
24              int extent = scrollBar.getModel().getExtent();
25              //Scrollbar en altta olup olmadigi kontrol edilir
26              if (scrollBar.getValue() + extent == scrollBar.getMaximum()) {
27                  int maxResult = view.getVisibleRowCount();
28                  int firstResult = view.getTableRowCount();
29                  model.createArticleList(firstResult, maxResult);
30              }
31          }
32      }
33  }
```

RunProgram Sınıfı

```
/**
  * RunProgram sinifi MVC'i initialize eder
  */
public class RunProgram {
     public void start() {
         TableView view=new TableView();
         TableModel model=new TableModel();
         model.addObserver(view);
         TableController controller=new TableController(view,model);
     }
}
```

Article Sınıfı

```
/**
  * Article sinifi
  */
public class Article {
     private int id;
     private String title;
     private String titleSearch;
     private String topic;
     public int getId() {
         return id;
     }
     public void setId(int id) {
         this.id= id;
     }
     public String getTitle() {
         return title;
     }
     public void setTitle(String title) {
         this.title= title;
     }
     public String getTitleSearch() {
         return titleSearch;
     }
     public void setTitleSearch(String titleSearch) {
         this.titleSearch= titleSearch;
     }
     public String getTopic() {
         return topic;
     }
     public void setTopic(String topic) {
         this.topic= topic;
     }
}
```

Main Sınıfı

```
import javax.swing.*;
/**
  * Main sinifi
  */
public class Main {
     public static void main(String[] args) {
         SwingUtilities.invokeLater(new Runnable() {
             @Override
             public void run() {
                 RunProgram program=new RunProgram();
                 program.start();
             }
         });
```

```
14          }
15      }
```

Not: Buradaki **Article** sınıfı **Model** sınıfı içerisinde kullanılmaktadır ve **ui-data'yı** temsil etmektedir.

Sonuç

MVC ile uygulamalar geliştirirken farklı implemantasyonları kullanılabilmektedir. Biz bu uygulamamızda Model ve View'i **Observer** tasarım deseni aracılığı ile birbirinden bağımsız yapan şekli kullandık. **Model** ve **View** örnekte görüldüğü gibi **Controller** sınıfı aracılığı ile etkileşimde bulunurlar. **Model** veriyi güncelledikten sonra **setChanged()** metodu ve **notifyObservers()** metodunu kullanarak **View** sınıfını günceller. **View** sınıfı güncellenen veriyi almak için **Observer** interface'i içerisinde tanımlanmış olan **update()** metodunu kullanır, direkt etkileşim halinde olmazlar. Bu sayede model sınıfını dilediğimiz view sınıfları ile kullanabiliriz.

Bu konu ile ilgili detaylı bilgi için: **http://www.codesenior.com/en/tutorial/Java-Swing-MVC-Usage**

Uygulamayı indirmek için **tıklayınız**