Atypon Training
MVC Design Pattern

Instructor: Dr. Fahed Jubair

fjubair@atypon.com

ATYPON WILEY

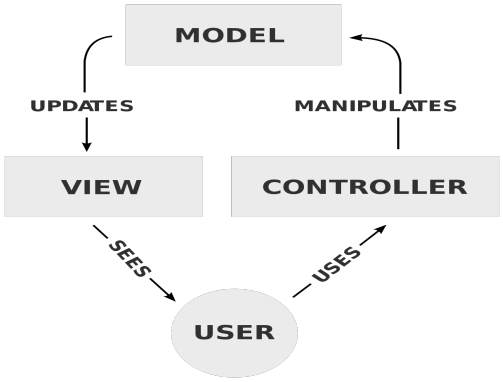
1

MVC Pattern

- Model-View-Control (MVC) is a commonly used pattern in GUI programming
- MVC divides the responsibility of a system into three parts
  1. Model: contains the underlying applications' data
  2. View: contains a convenient interface (typically GUI) to allow the user to view the model in some expected way
  3. Controller: acts as a mediator between the view and the model, i.e., the view sends requests to the model indirectly through the controller

2

MVC Pattern Architecture



Source: https://en.wikipedia.org/wiki/Model-view-controller

3

Remarks About the MVC Pattern

- Typically, the Observer pattern is used to ensure a model object notifies all view objects whenever a new update is available
  - A model can have multiple views
- The presence of the Controller objects ensures that the model and the view are loosely-coupled
  - The controller is responsible for interpreting requests and interaction with elements in the view, and changing the model
  - The view is therefore only responsible for the visual appearance of the system
  - The model focuses only on managing the information for the system
- MVC pattern uses the separation of concern design principle

4

MVC Pattern Example

- This example contains the following classes
  - TVStation (model): represents a tv station that contains a list of channels
  - TVStationViewer (view): represents a viewer of a tv station
  - TVStationController (control): represents the controller that mediates requests between a tv station with its reviewers
  - MVCPatternDemo: a test class
- The Observer pattern is used to allow a TVStation object to update attached TVStationViewer objects
- We do so using java.util.Observable and java.util.Observer (see next slides)

5

java.util.Observable Class

- Java's ready-to-use class for "subject" classes in the Observer design pattern
- This class represents an observable object, or "data" in the MVC paradigm
- See available methods of this class

  https://docs.oracle.com/javase/8/docs/api/index.html?java/util/Observable.html

6

java.util.Observer Interface

- Java's built-in interface to be used by "observer" objects in the Observer design pattern
- Requires implementing the following method

  void update(Observable o, Object arg)

  o - the observable object

  arg - an argument passed to the notifyObservers method

7

MVC Pattern Example Model

```
import java.util.ArrayList;
import java.util.List;
import java.util.Observable;
public class TVStation extends Observable {

    private final String name;
    private List<String> channelsList;

    public TVStation(String name){
        this.name = name;
        channelsList = new ArrayList<String>();
    }

    public String getName(){
        return name;
    }

    public void addChannel(String channelName){
        channelsList.add(channelName);
        this.setChanged();
        notifyObservers( arg: "A new channel is available: " + channelName);
    }

    public void removeChannel(String channelName){
        channelsList.remove(channelName);
        this.setChanged();
        notifyObservers( arg: channelName + " is no longer available");
    }

    public int getChannelID(String channelName){
        return channelsList.indexOf(channelName);
    }
    public String getChannelName(int channelID){
        return channelsList.get(channelID);
    }
}
```

8

MVC Pattern Example View

```
import java.util.Observable;
import java.util.Observer;

public class TVStationViewer implements Observer {

    private final TVStationController tvStationController;
    private final String name;

    public TVStationViewer(TVStationController tvStationController, String name){
        this.tvStationController = tvStationController;
        this.name = name;
        this.tvStationController.subscribe( tvStationViewer: this);
    }

    @Override
    public void update(Observable o, Object arg) {
        TVStation tvStation = (TVStation) o;
        String msg = (String) arg;
        System.out.println("[" + name + "]: notification from " + tvStation.getName() + " is received: " + msg);
    }
}
```


9

MVC Pattern Example Controller

```
public class TVStationController {

    private TVStation tvStation;

    public TVStationController(TVStation tvStation){
        this.tvStation = tvStation;
    }

    public void subscribe(TVStationViewer tvStationViewer){
        tvStation.addObserver(tvStationViewer);
    }

    public void unsubscribe(TVStationViewer tvStationViewer){
        tvStation.deleteObserver(tvStationViewer);
    }

    public int getChannelID( String channelName){
        return tvStation.getChannelID(channelName);
    }
}
```


10

MVC Pattern Example Demo

```
public class MVCDemo {

    public static void main(String[] args){

        TVStation tvStation1 = new TVStation( name: "nilesat");
        TVStationController tvStationController1 = new TVStationController(tvStation1);
        TVStationViewer viewer1 = new TVStationViewer(tvStationController1,  name: "ahmad");
        TVStationViewer viewer2 = new TVStationViewer(tvStationController1,  name: "rania");
        TVStationViewer viewer3 = new TVStationViewer(tvStationController1,  name: "suha");
        System.out.println(tvStation1.countObservers());

        tvStation1.addChannel( channelName: "mbc");
        tvStation1.addChannel( channelName: "roya");
        tvStation1.addChannel( channelName: "jazeera");
        tvStation1.addChannel( channelName: "arabia");
        tvStation1.removeChannel( channelName: "roya");
    }
}
```


11

Spring MVC

- A Spring MVC is a Java framework which is used to build web applications
- Spring MVC helps you build flexible and loosely coupled web applications
- Resources for learning Spring MVC:

  https://learning.oreilly.com/library/view/spring-mvc-beginners/9781785880636/

  https://learning.oreilly.com/videos/spring-mvc-for/9781789139341

  https://learning.oreilly.com/videos/learning-path-build/9781491961025

12