Application Program Development

Segment : Separating Models, Views and Controller Components

Mahboob Ali

Agenda for Week 4

- Lecture
  - Re-visiting MVC
  - Talking about designing of classes
  - Preparing proper Model classes
  - Preparing proper Views
  - Preparing proper controllers
  - Putting everything together
- Lab
  - Part 1 : In-Lab – Design and create a GUI based Application
  - Part 2 : DIY – using the GUI designed in-lab to write events

Outcomes

- Re-visiting MVC

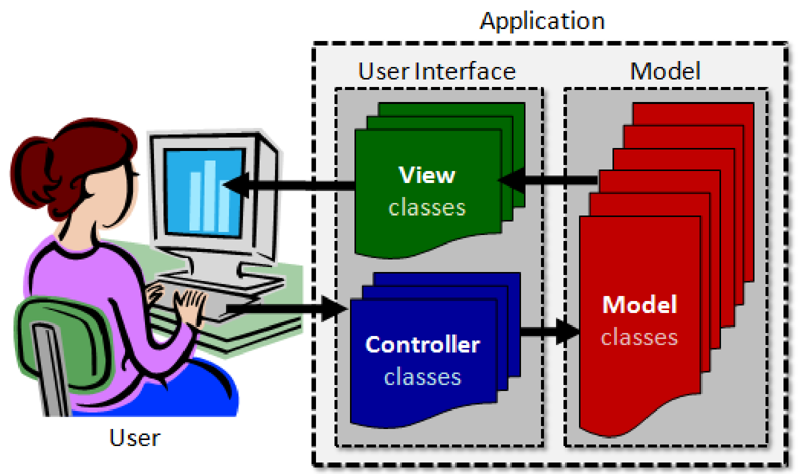
- As we discussed the difference between **model** classes and those classes that are part of the **user interface**.
- The **model** classes deal with the *business logic* aspects of the application and the user interface is the "front end" which allows the user to interact with the model classes.
- We can further split the user interface classes into two portions called the ***view*** and the ***controller***:

- *The **view** displays the necessary information from the model into a form suitable for interaction, typically a user interface element.*

- *The **controller** accepts input from the user and modifies the model accordingly.*

Motivation

- The user sees the **view** of the application and then interacts with the **controller**.
- Such interaction usually results in the **model** being modified in some way.
- Then these **model** changes are reflected back to the **view** of the user interface and the user often gets visual feedback that the **model** has changed



The need

- So far, in our examples, we did not have a really useful **model** and the notion of a **view** and a **controller** was not identifiable as all our GUI code was lumped together into one **Application** class, with perhaps an extra **Pane** subclass.

- Now we will discuss the "proper" way of splitting up the **model**, the **view** and the **controller** into a nice & clean modular style that allows us to modify and replace any of the three components cleanly.

- This arrangement represents what is called the **MVC** software architecture and sometimes referred to as a software **design pattern**.

The need

- There are 3 main advantages of using the MVC architecture:

- it decouples the models and views,
- it reduces the complexity of the overall architectural design and
- it increases flexibility and maintainability of code.

- There are many ways to implement the MVC architecture in your programs.
- We will consider just one specific way of writing the code.
- In industry, however, you will see various other ways of implementing the same architecture.

- In order to create a well-structured, stable, reliable and maintainable application ... it is necessary to have a properly working model that is designed nicely so that the user interface can connect to it in a simple and safe way.

- In the next segment, we will discuss what is necessary to create this "proper" kind of model.