Model View Controller (MVC) Architecture

Terminology and History

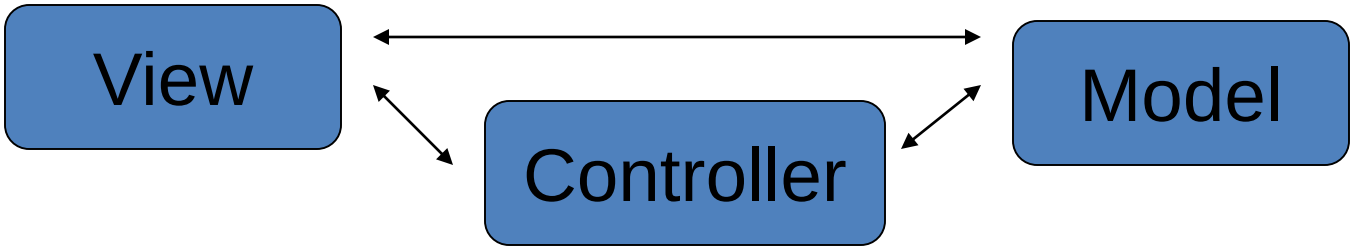
- MVC evolved from Smalltalk-80
- Has become a key pattern in web based applications
  - If you do not understand the MVC pattern you cannot succeed in web development
  - Popular frameworks that support MVC
    - Struts (apache.org)
    - .NET (Microsoft)

Terminology and History

- The MVC paradigm is a way of breaking an application, or even just a piece of an application's interface, into three parts: the model, the view, and the controller. MVC was originally developed to map the traditional input, processing, output roles into the GUI realm:
- Input --> Processing --> Output
  Controller --> Model --> View

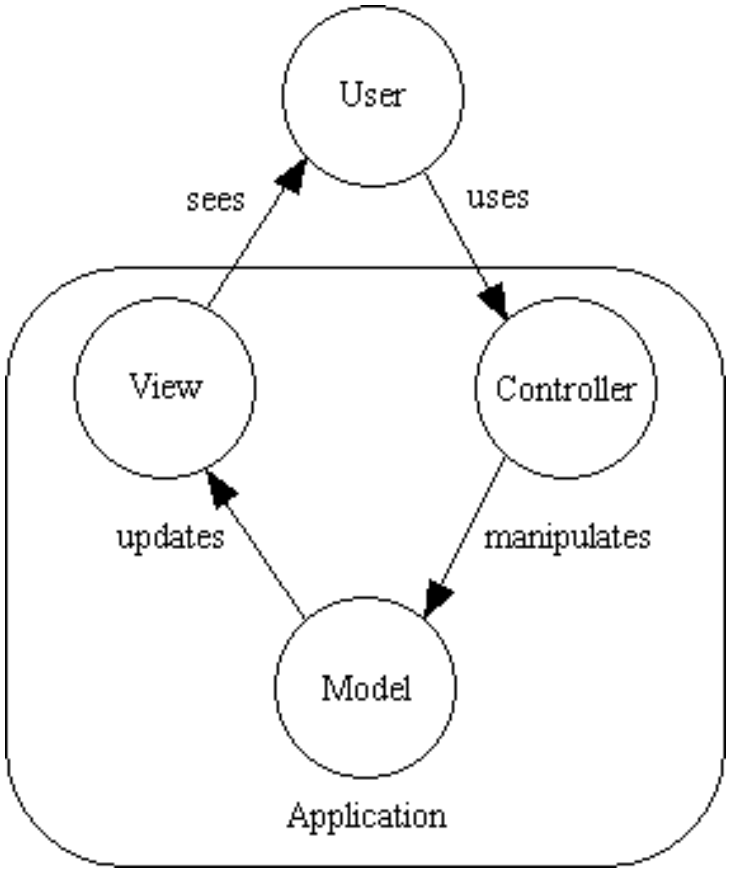
Model-View-Controller

- Many webapps are based on the well-known model-view-controller design pattern

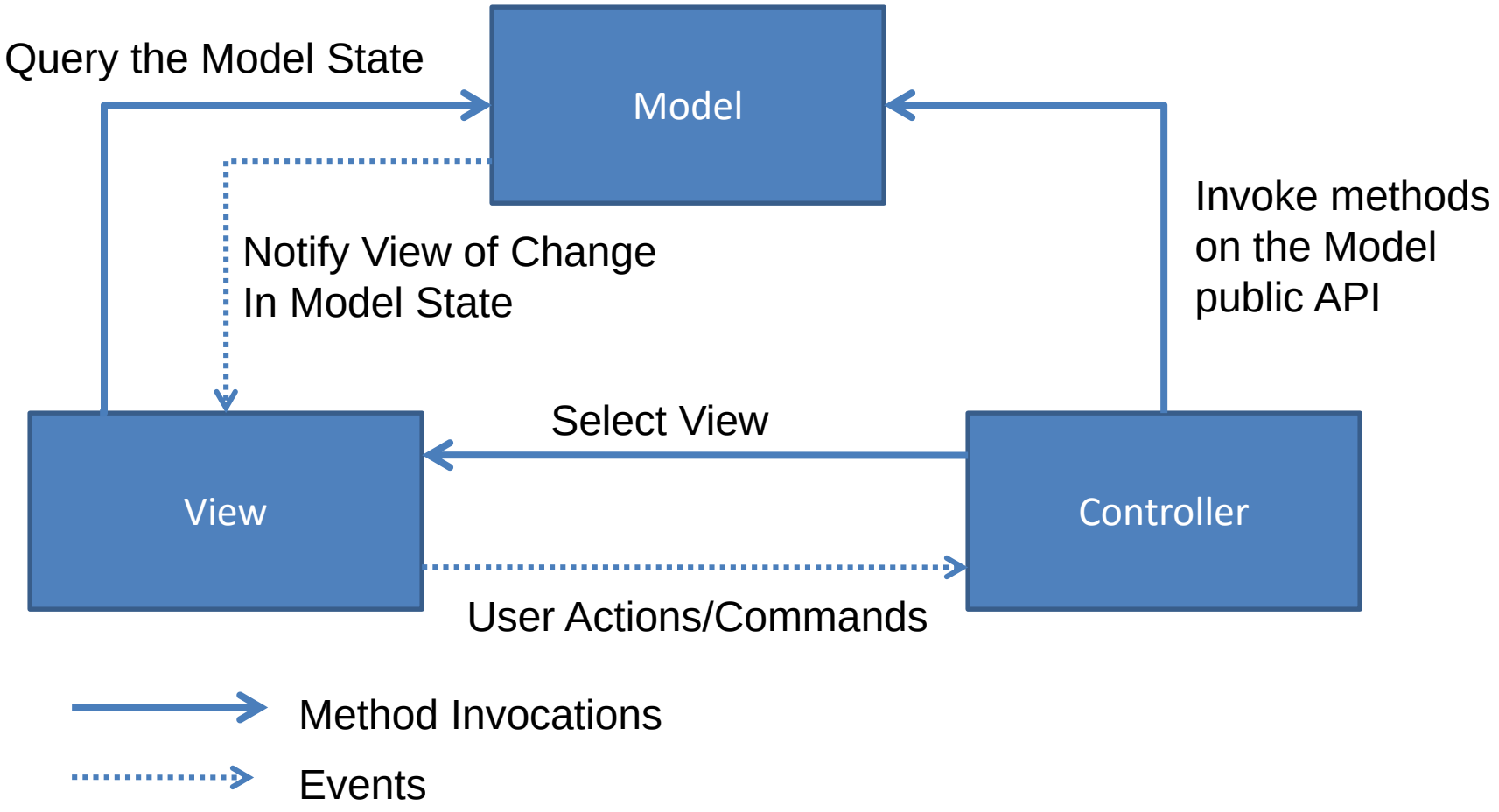


- The MVC pattern specifies the responsibilities of three major components, and the nature of their interactions

The MVC Paradigm – Basic Model



The MVC Paradigm – Basic Model



The Model

- In MVC, the model is the code that carries out some task.
- It is built with no necessary concern for how it will "look and feel" when presented to the user.
- A model is an object representing data or even activity, e.g. a database table or even some plant-floor production-machine process.
- The model
  - manages the behavior and data of the application domain,
  - responds to requests for information about its state and
  - responds to instructions to change state.

The Model

- The model often represents enterprise data and the business rules that govern access to and updates of this data. Often the model serves as a software approximation to a real-world process, so simple real-world modeling techniques apply when defining the model.
- The model is the piece that represents the state and low-level behavior of the component. It manages the state and conducts all transformations on that state.

The Model

- The model has no specific knowledge of either
  - its controllers or
  - its views.
- However, a model must be able to "register" views and it must be able to "notify" all of its registered views when any of its functions cause its state to be changed.
- The system itself maintains links between model and views and notifies the views when the model changes state. The view is the piece that manages the visual display of the state represented by the model.
- A model can have more than one view.

The Model

- Note that the model may not necessarily have a persistent data store (database), such as when it deals with a control on a GUI screen.
- It has a purely functional interface, meaning that it has a set of public functions that can be used to achieve all of its functionality.
- Some of the functions are query methods that permit a "user" to get information about the current state of the model. Others are mutator methods that permit the state to be modified.
  - setTemperature(temp) and getTemperature() – Temperature Model
  - addItem(item, category), setScore(item, student, score)… - Gradebook

Model Example

```
public class TemperatureModel extends java.util.Observable {
   public double getF(){return temperatureF;}
   public double getC(){return (temperatureF - 32.0) * 5.0 / 9.0;}
   public void setF(double tempF) {
        temperatureF = tempF;
        setChanged();
        notifyObservers();
   }
   public void setC(double tempC) {
        temperatureF = tempC*9.0/5.0 + 32.0;
        setChanged();
        notifyObservers();
   }

   private double temperatureF = 32.0;
}
```

View

- A view is some form of visualization of the state of the model.
- The view manages the graphical and/or textual output to the portion of the bitmapped display that is allocated to its application.
- The view renders the contents of a model. It accesses enterprise data through the model and specifies how that data should be presented.
- The view is responsible for mapping graphics onto a device. A view typically has a one to one correspondence with a display surface and knows how to render to it. A view attaches to a model and renders its contents to the display surface.

View

- A model in MVC can have several views. For examples
  - the rows and columns view of a spreadsheet and
  - the pie chart view of some column in the same spreadsheet.
- A view provides graphical user interface (GUI) components for a model. It gets the values that it displays by querying the model of which it is a view.
- When a user manipulates a view of a model, the view informs a controller of the desired change.

View - Code Segment

```
TemperatureGUI(String label, TemperatureModel model, int h, int v)
   {
        this.label = label;
        this.model = model;
        temperatureFrame = new Frame(label);
        temperatureFrame.add("Center", display);
        Panel buttons = new Panel();
        buttons.add(upButton);
        buttons.add(downButton);
        temperatureFrame.add("South", buttons);
        temperatureFrame.addWindowListener(new CloseListener());
        model.addObserver(this); // Connect the View to the Model
        temperatureFrame.setSize(200,100);
        temperatureFrame.setLocation(h, v);
        temperatureFrame.setVisible(true);
   }
```

Controllers

- Views in MVC are associated with controllers that update the model as necessary when a user interacts with an associated view.
- The controller can call mutator methods of the model to get it to update its state. Of course, then the model will notify ALL registered views that a change has been made and so they will update what they display to the user as appropriate

Controllers

- A controller offers facilities to change the state of the model. The controller interprets the mouse and keyboard inputs from the user, commanding the model and/or the view to change as appropriate.
- A controller is the means by which the user interacts with the application. A controller accepts input from the user and instructs the model and view to perform actions based on that input. In effect, the controller is responsible for mapping end-user action to application response.

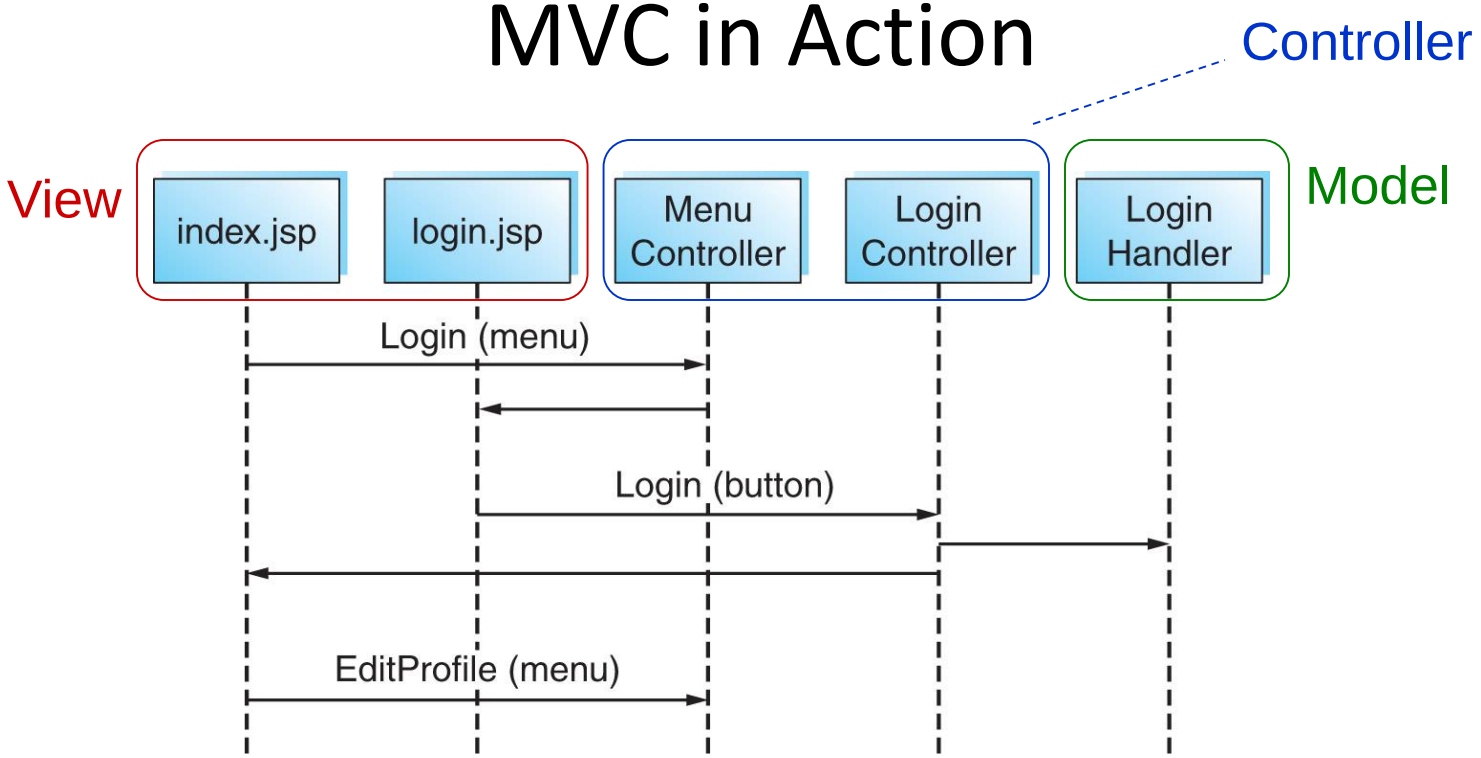
Controllers

- The controller translates interactions with the view into actions to be performed by the model. In a stand-alone GUI client, user interactions could be button clicks or menu selections, whereas in a Web application they appear as HTTP GET and POST requests.
- The actions performed by the model include activating business processes or changing the state of the model.

Controllers

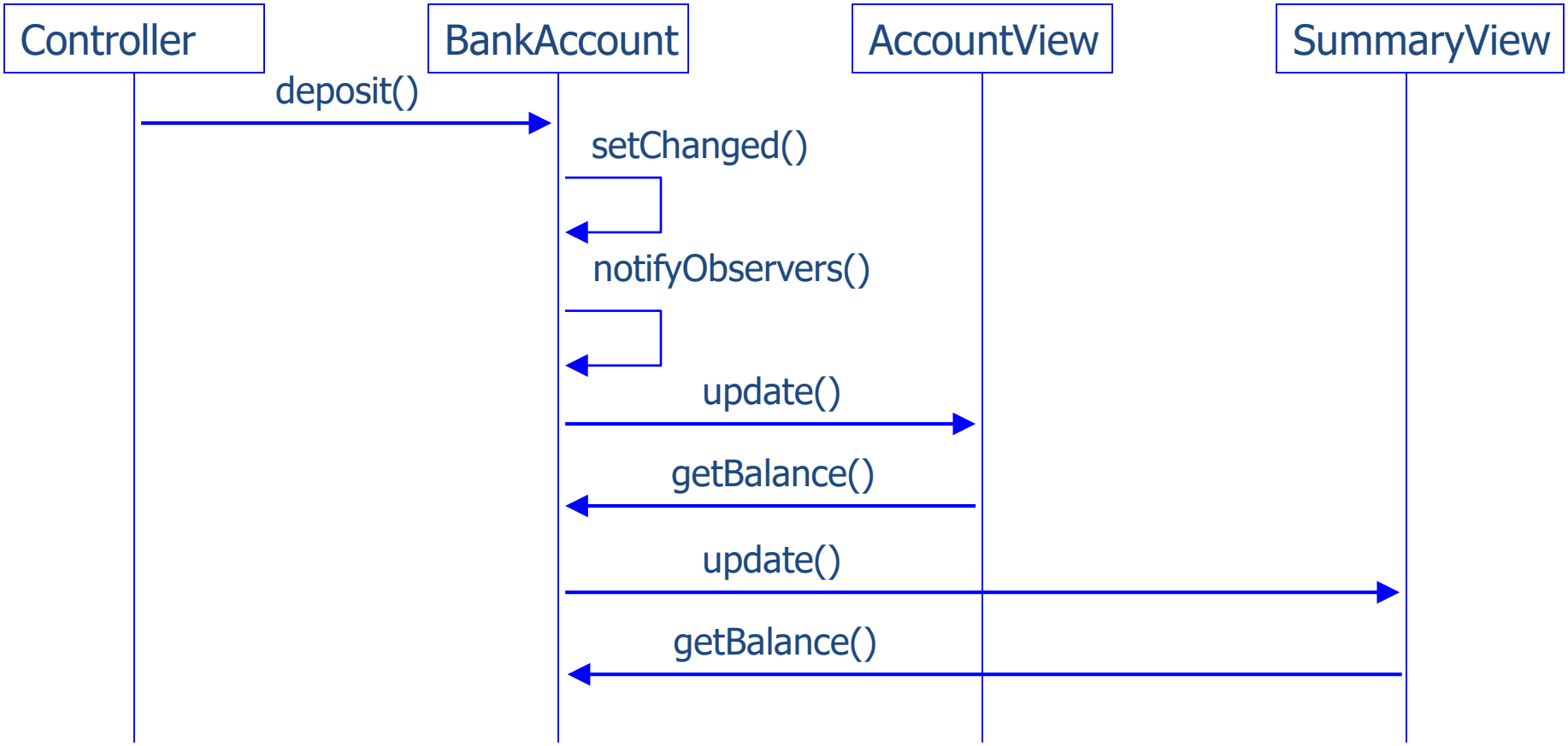
- Based on the user interactions and the outcome of the model actions, the controller responds by selecting an appropriate view.
- The controller is the piece that manages user interaction with the model. It provides the mechanism by which changes are made to the state of the model.

MVC in Action



1. User requests "login" option; Controller sets "login.jsp" as the next view.
2. User enters credentials; Controller invokes LoginHandler to handle transcation
3. etc.

Transactions Happen!



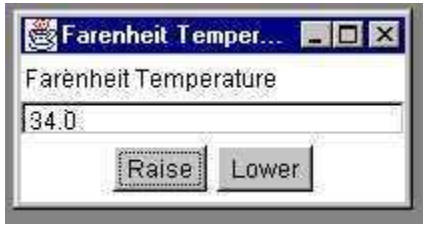
Scalable MVC Framework

- A more scalable framework can be created by defining each transaction in a configuration file
- New transactions can be created by adding View and Model components as needed, and modify the configuration

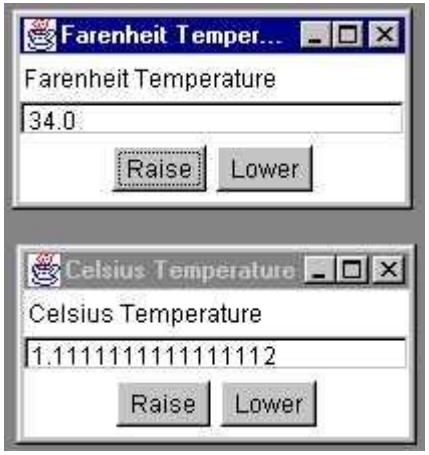
```
<control-mappings>
 <action-mapping>
  <option>login</option>
  <handler>bean.LoginHandler
   </handler>
  <parameter>userid
   </parameter>
  <parameter>password
   </parameter>
  </action-mapping>
 <view-mapping>
  <option>login</option>
  <outcome value="success">
    menu.jsp</outcome>
  <outcome value="failure">
    login.jsp</outcome>
 </view-mapping>
</control-mappings
```

Visual Examples

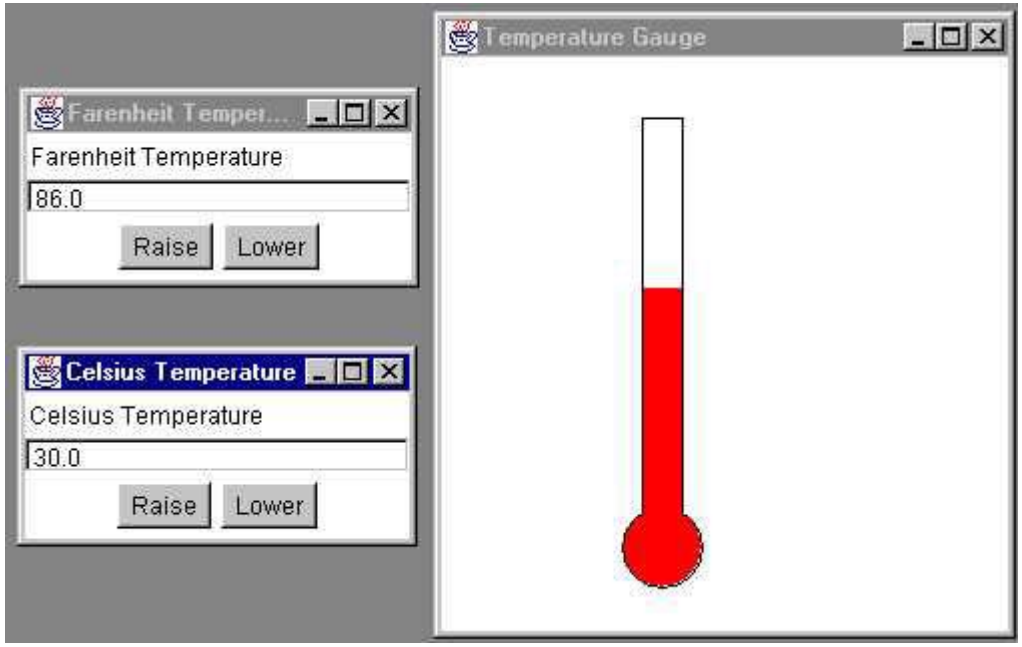
View 1



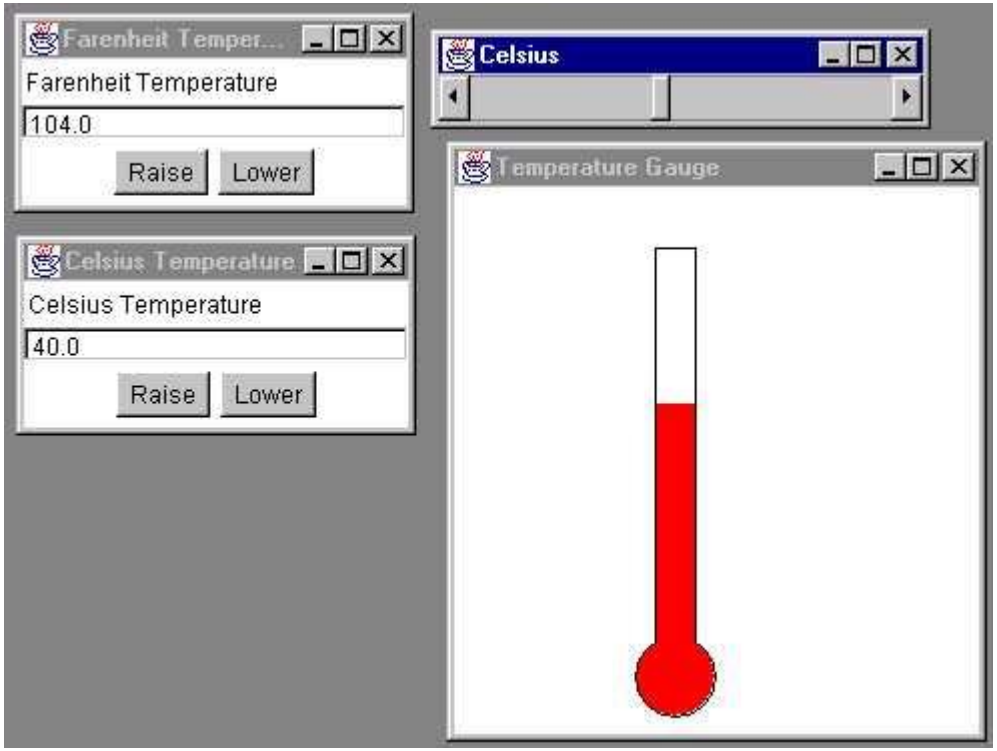
View 2


Farenheit Temper...
Farenheit Temperature
34.0
Raise
Lower
Celsius Temperature
Celsius Temperature
1.1111111111111112
Raise
Lower

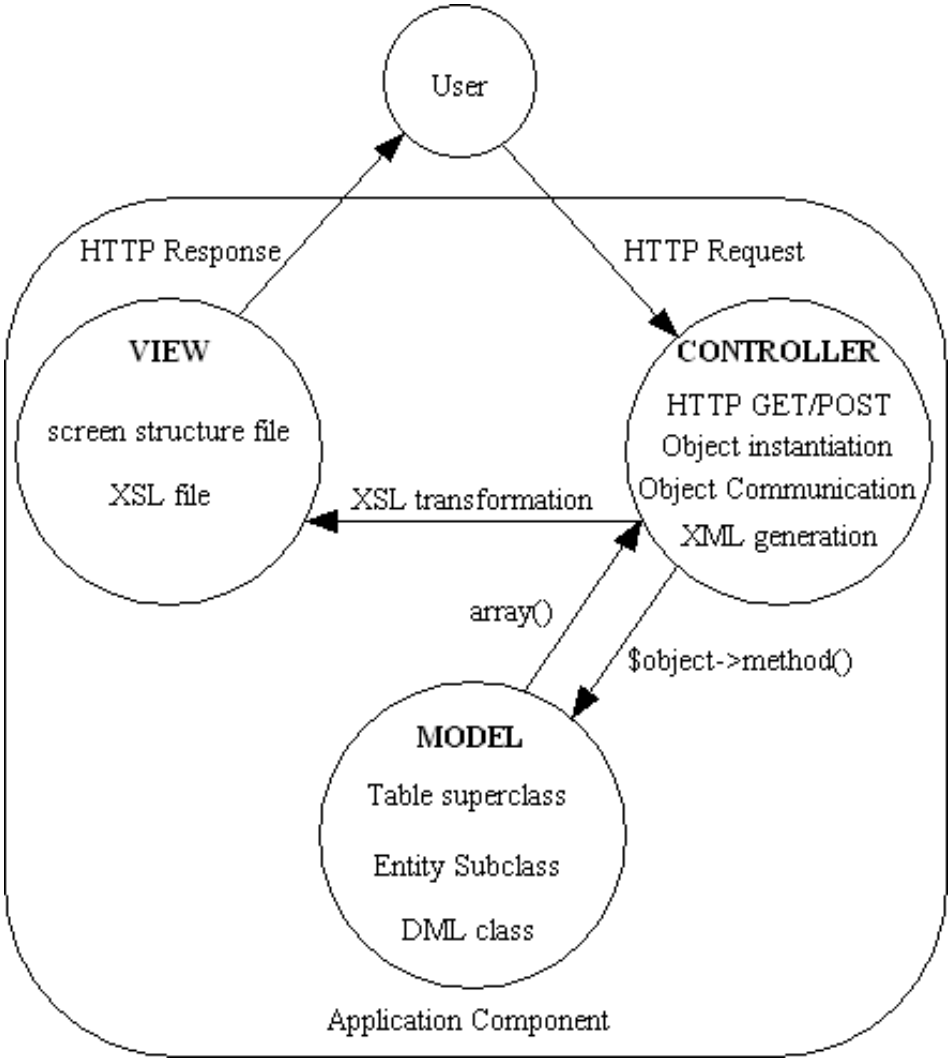
View 3


Temperature Gauge
Farenheit Temper...
Farenheit Temperature
86.0
Raise
Lower
Celsius Temperature
Celsius Temperature
30.0
Raise
Lower

View 4


Farenheit Temper...
Farenheit Temperature
104.0
Raise
Lower
Celsius Temperature
Celsius Temperature
40.0
Raise
Lower
Celsius
Temperature Gauge

The MVC Paradigm - Improved



MVC - Web Based Model