**案例2：一个简单的Java MVC架构小程序**

如下图所示是一个简单的Java的GUI小程序，其功能是在对话框中显示一个能够反弹的小球，并同时能显示当前小球的状态（如坐标，所走步数）。通过这样一个实例我们能够更清楚的了解MVC的实现机制。



反弹球Applet界面

该程序可以很好的使用MVC模式来实现，我们可以定义模型Model，视图View，和控制器Controller三个类，具体类图如图3-18所示：

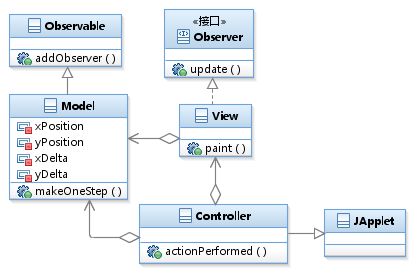


图3-18：反弹球程序类图

首先，在这个例子中，模型为小球的信息，为了知道球什么时候反弹，模型必须知道该窗口的大小。除此之外，该模型并不需要知道任何其他有关GUI信息。这使得视图和模型能够很好的分离开来。模型示例代码如下所示：

class Model extends Observable {

final int BALL\_SIZE = 20;

int xPosition = 0;

int yPosition = 0;

int xLimit, yLimit;

int xDelta = 6;

int yDelta = 4;

void makeOneStep() {

xPosition += xDelta;

if (xPosition < 0) {

xPosition = 0;

xDelta = -xDelta;

}

if (xPosition >= xLimit) {

xPosition = xLimit;

xDelta = -xDelta;

}

yPosition += yDelta;

if (yPosition < 0 || yPosition >= yLimit) {

yDelta = -yDelta;

yPosition += yDelta;

}

setChanged();

notifyObservers();

}

}

视图的作用是显示球的状态（x、y坐标位置）。对于静态绘图，视图不需要知道其他信息。代码如下所示：

class View extends Canvas implements Observer {

Controller controller;

Model model;

int stepNumber = 0;

View(Model model) {

this.model = model;

}

public void paint(Graphics g) {

g.setColor(Color.red);

g.fillOval(model.xPosition, model.yPosition, model.BALL\_SIZE,

model.BALL\_SIZE);

controller.showStatus("Step " + (stepNumber++) + ", x = "

+ model.xPosition + ", y = " + model.yPosition);

}

public void update(Observable obs, Object arg) {

repaint();

}

}

控制器告诉模型做什么。当需要刷新视图时，由控制器通知视图。控制器并不需要知道模型和视图的具体内部运作。

public class Controller extends JApplet {

JPanel buttonPanel = new JPanel();

JButton stepButton = new JButton("Step");

Model model = new Model();

View view = new View();

public void init()

setLayout(new BorderLayout());

buttonPanel.add(stepButton);

this.add(BorderLayout.SOUTH, buttonPanel);

this.add(BorderLayout.CENTER, view);

stepButton.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent event) {

model.makeOneStep();

}});

model.addObserver(view);

view.controller=this;

}

public void start() {

model.xLimit = view.getSize().width - model.BALL\_SIZE;

model.yLimit = view.getSize().height - model.BALL\_SIZE;

repaint();

}

}