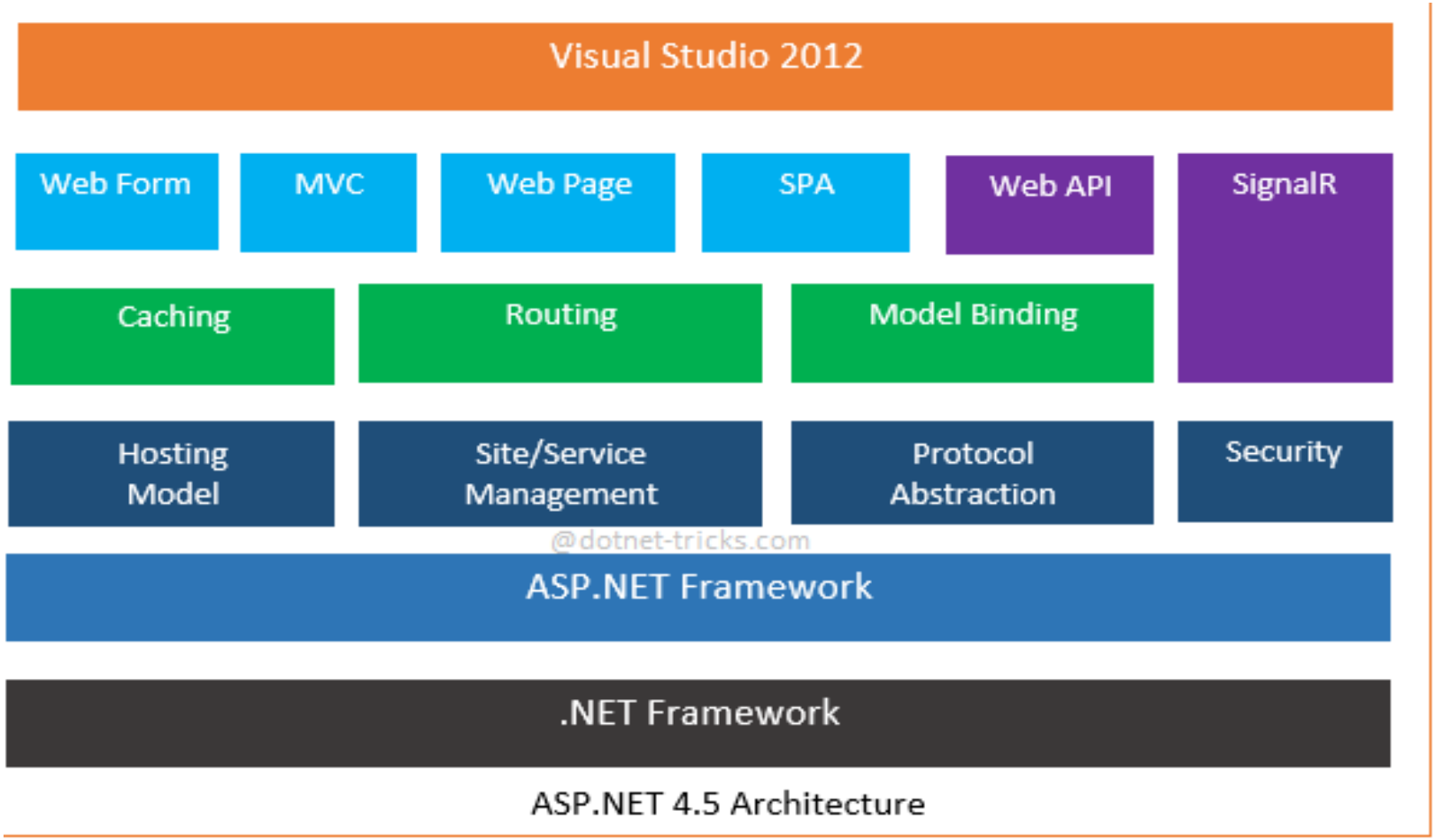
Interactive system development Frame work

- A *development framework* refers to a modular approach for interactive program development where the core computational and interface parts are developed in a modularized fashion and combined in a flexible manner.
- Such a development framework is often based on the UI toolkit, which provides the abstraction for the interface parts. For one, the framework allows the concept of plugging in different interfaces for the same model computation and easier maintenance of the overall program.
- In addition, such a practice also promotes the productivity as well as easier and less costly pos tmaintenance. MVC (model, view, and controller) is one such major framework.



ASP.NET 4.5 Architecture

Introduction

- **M**odel **V**iew **C**ontroller or MVC as it is popularly called, is a software design pattern for developing web applications.

- MVC is one of three ASP.NET programming models.

- **Model–view–controller** (**MVC**) is a software architecture pattern which separates the representation of information from the user's interaction with it .

History of MVC

- Presented by Trygve Reenskaug in 1979
- First used in the Smalltalk-80 framework
  - Used in making Apple interfaces (Lisa and Macintosh

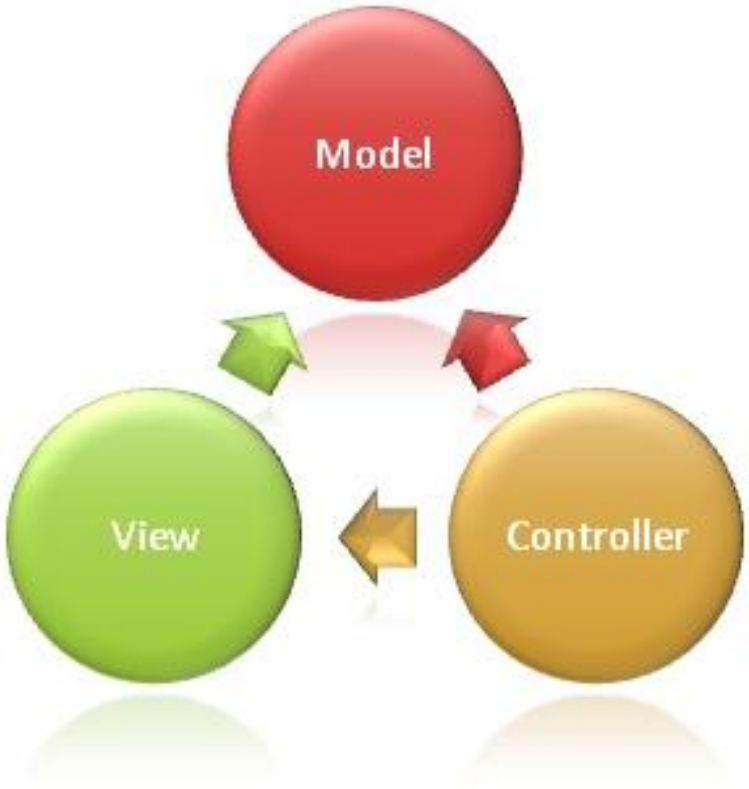
MVC uses

Smalltalk's MVC implementation inspired many other GUI frameworks such as:

- The NEXTSTEP and OPENSTEP development environments encourage the use of MVC.
- Cocoa and GNUstep, based on these technologies, also use MVC.
- Microsoft Foundation Classes (MFC) (also called Document/View architecture)
- Java Swing
- The Qt Toolkit (since Qt4 Release).

Parts of MVC

- A Model View Controller pattern is made up of the following three parts:
- **Model**
- **View**
- **Controller**

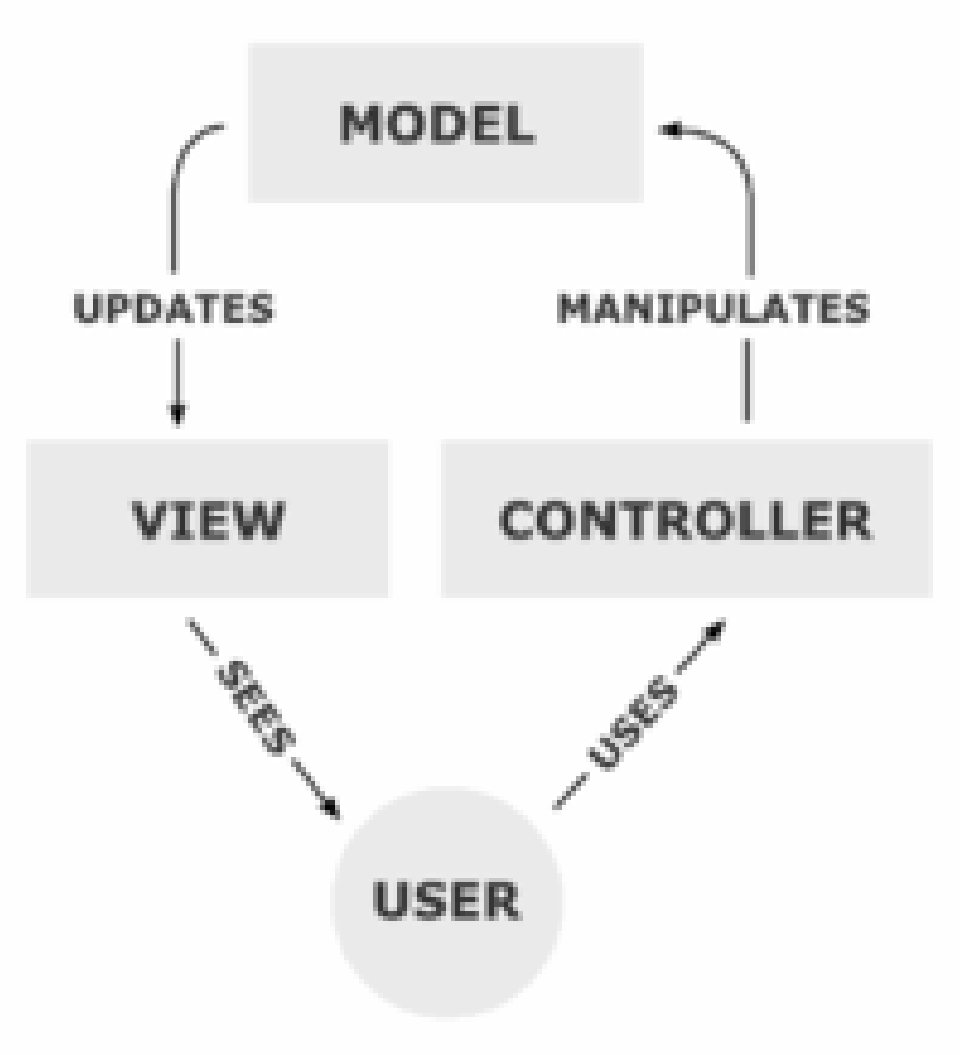


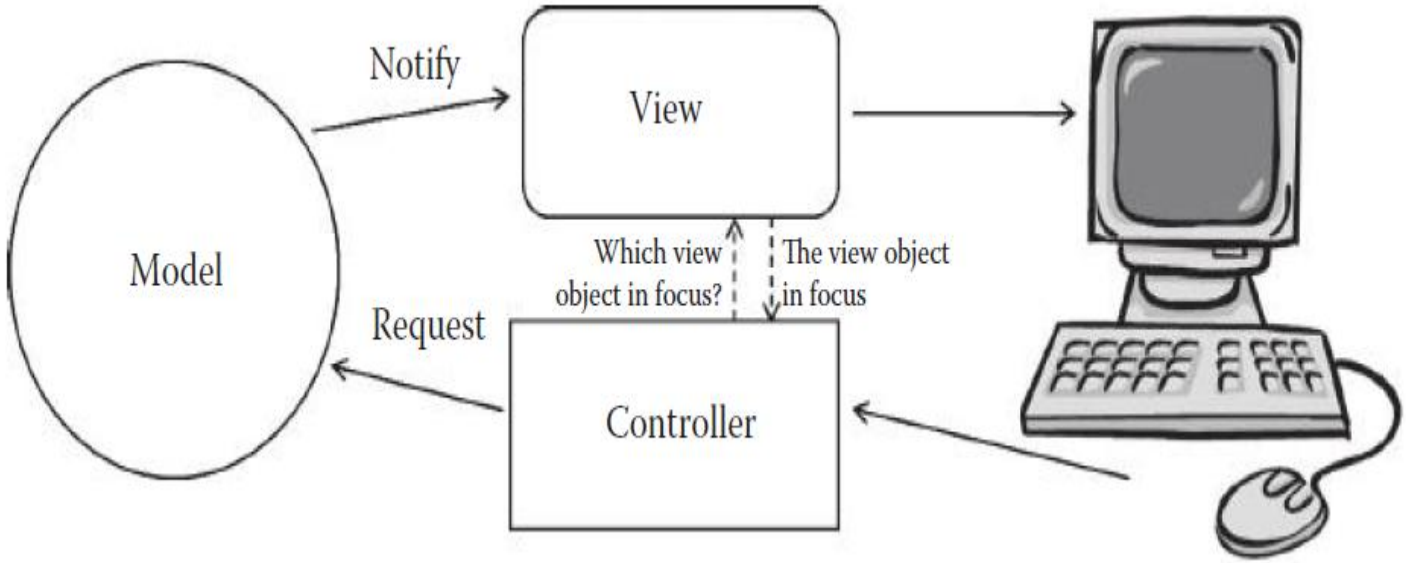
The MVC model defines web applications with 3 logic layers:

The business layer (Model logic)

The display layer (View logic)

The input control (Controller logic)


MODEL
UPDATES
MANIPULATES
VIEW
CONTROLLER
SEES
USES
USER


Notify
View
Model
Which view
object in focus?
The view object
in focus
Request
Controller

Model

- The model is responsible for managing the data of the application.
- It responds to the request from the view and it also responds to instructions from the controller to update itself
- It is the lowest level of the pattern which is responsible for maintaining data.
- The Model represents the application core (for instance a list of database records).
- It is also called the domain layer

View

- The View displays the data (the database records).
- A **view** requests information from the model, that it needs to generate an output representation.
- It presents data in a particular format like JSP, ASP, PHP.
- MVC is often seen in web applications, where the view is the HTML page.

Controller

- **The Controller** is the part of the application that handles user interaction.
- Typically controllers read data from a view, control user input, and send input data to the model.
- It handles the input, typically user actions and may invoke changes on the model and view.

View/Controller

- In many application architectures, the view and controller may be merged into one module or object because they are so tightly related to each other
- UI button object will be defined by attribute parameters such as its size, label, and color as well as the event handler that invokes the methods on the model for change or manipulation
- large-scale systems, to quickly explore and implement and modify various user interfaces (view/controller) for the same core functional model. This is based on a famous software engineering principle: the separation of concern

Workflow in MVC

Though MVC comes in different flavours, the control flow generally works as follows:

1. The user interacts with the user interface in some way (e.g., user presses a button)
2. A controller handles the input event from the user interface, often via a registered handler or callback.
3. The controller accesses the model, possibly updating it in a way appropriate to the user's action (e.g., controller updates user's shopping cart).

4. A view uses the model to generate an appropriate user interface (e.g., view produces a screen listing the shopping cart contents).

   The view gets its own data from the model. The model has no direct knowledge of the view.

5. The user interface waits for further user interactions, which begins the cycle anew.

Dependence hierarchy

- There is usually a kind of hierarchy in the MVC pattern.
- The Model knows only about itself.
- That is, the source code of the Model has no references to either the View or Controller.

- The View however, knows about the Model. It will poll the Model about the state, to know what to display.
- That way, the View can display something that is based on what the Model has done.
- But the View knows nothing about the Controller.

- The Controller knows about both the Model and the View.

- Take an example from a game: If you click on the "fire" button on the mouse, the Controller knows what fire function in the Model to call.

- If you press the button for switching between first and third person, the Controller knows what function in the View to call to request the display change.

Why dependence hierarchy is used?

- The reason to keep it this way is to minimize dependencies.
- No matter how the View class is modified, the Model will still work.
- Even if the system is moved from a desktop operating system to a smart phone, the Model can be moved with no changes.
- But the View probably needs to be updated, as will the Controller.

Use in web applications

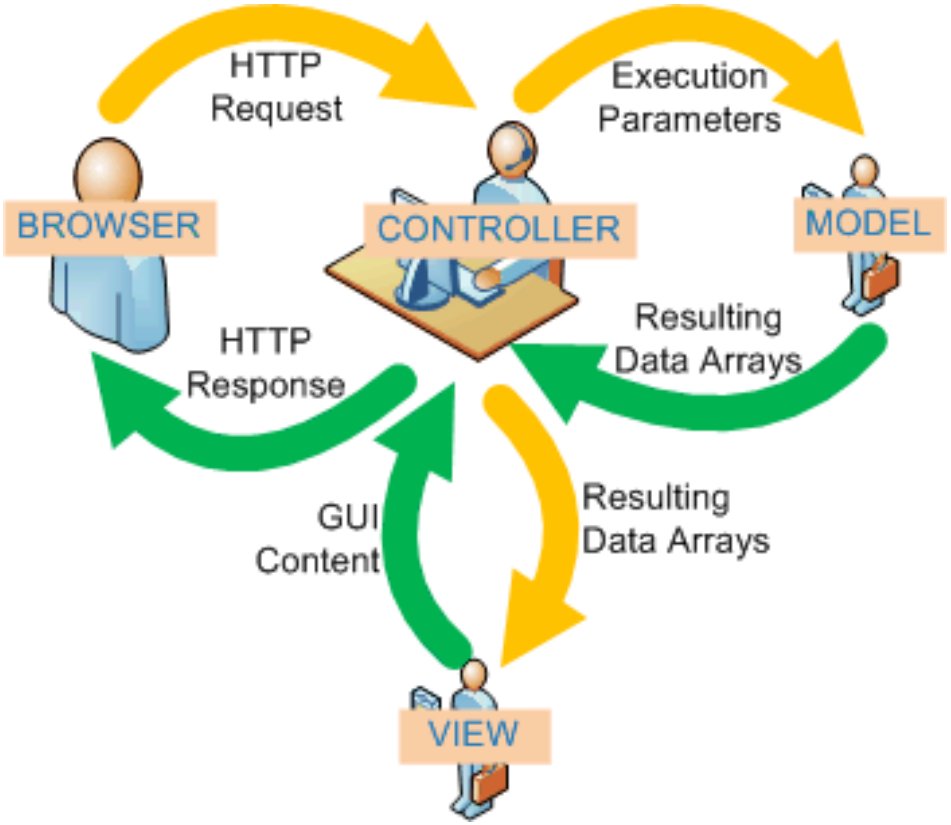
- Although originally developed for personal computing, Model View Controller has been widely adapted as an architecture for World Wide Web applications in all major programming languages.
- Several commercial and noncommercial application frameworks have been created that enforce the pattern.
- These frameworks vary in their interpretations, mainly in the way that the MVC responsibilities are divided between the client and server

- Early web MVC frameworks took a thin client approach that placed almost the entire model, view and controller logic on the server.

- In this approach, the client sends either hyperlink requests or form input to the controller and then receives a complete and updated web page from the view; the model exists entirely on the server.

- As client technologies have matured, frameworks such as JavaScript MVC and Backbone have been created that allow the MVC components to execute partly on the client

Working of MVC in web application

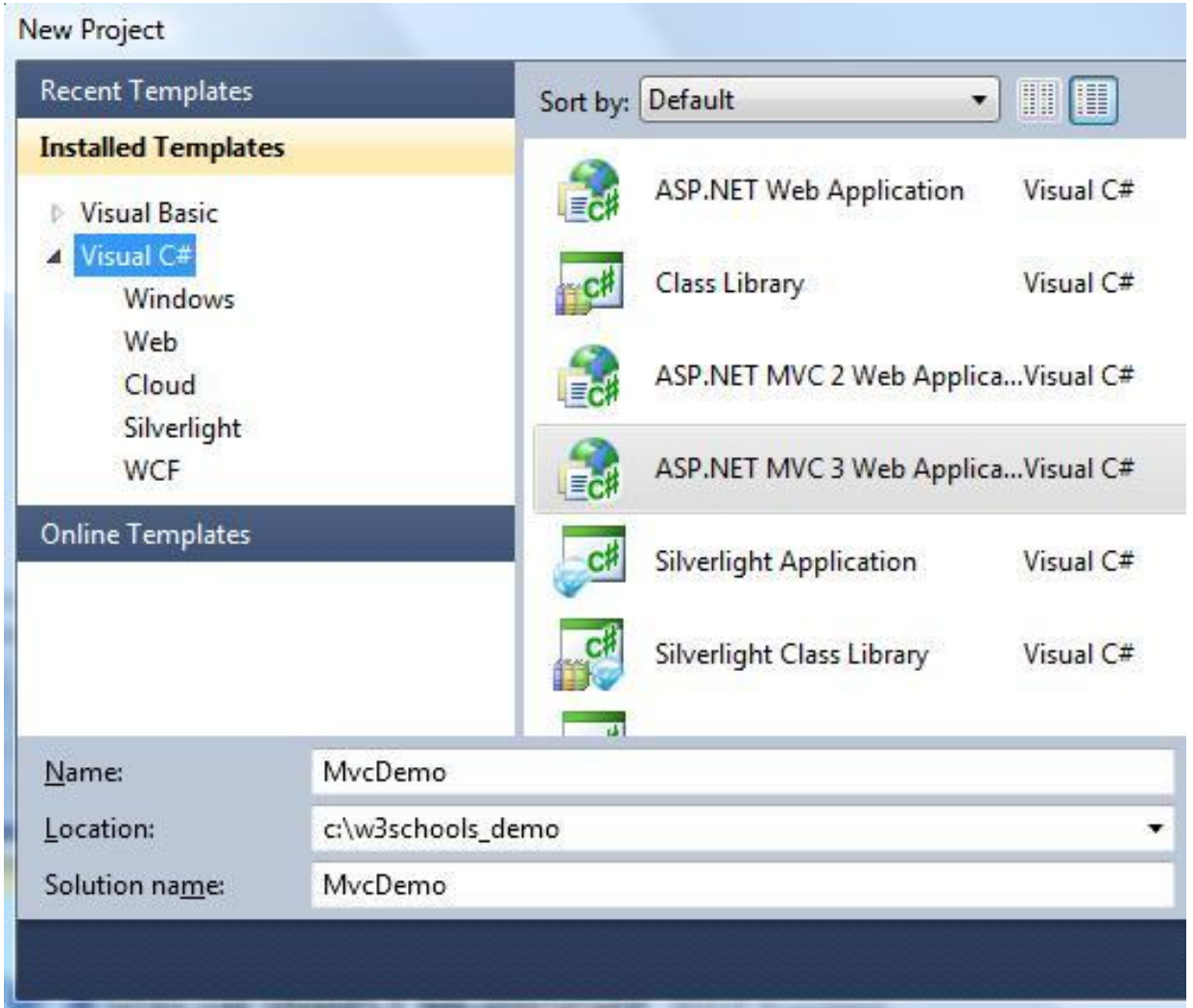


Web forms vs. MVC

- The MVC programming model is a lighter alternative to traditional ASP.NET (Web Forms).
- It is a lightweight, highly testable framework, integrated with all existing ASP.NET features, such as Master Pages, Security, and Authentication.

Creating the Web Application

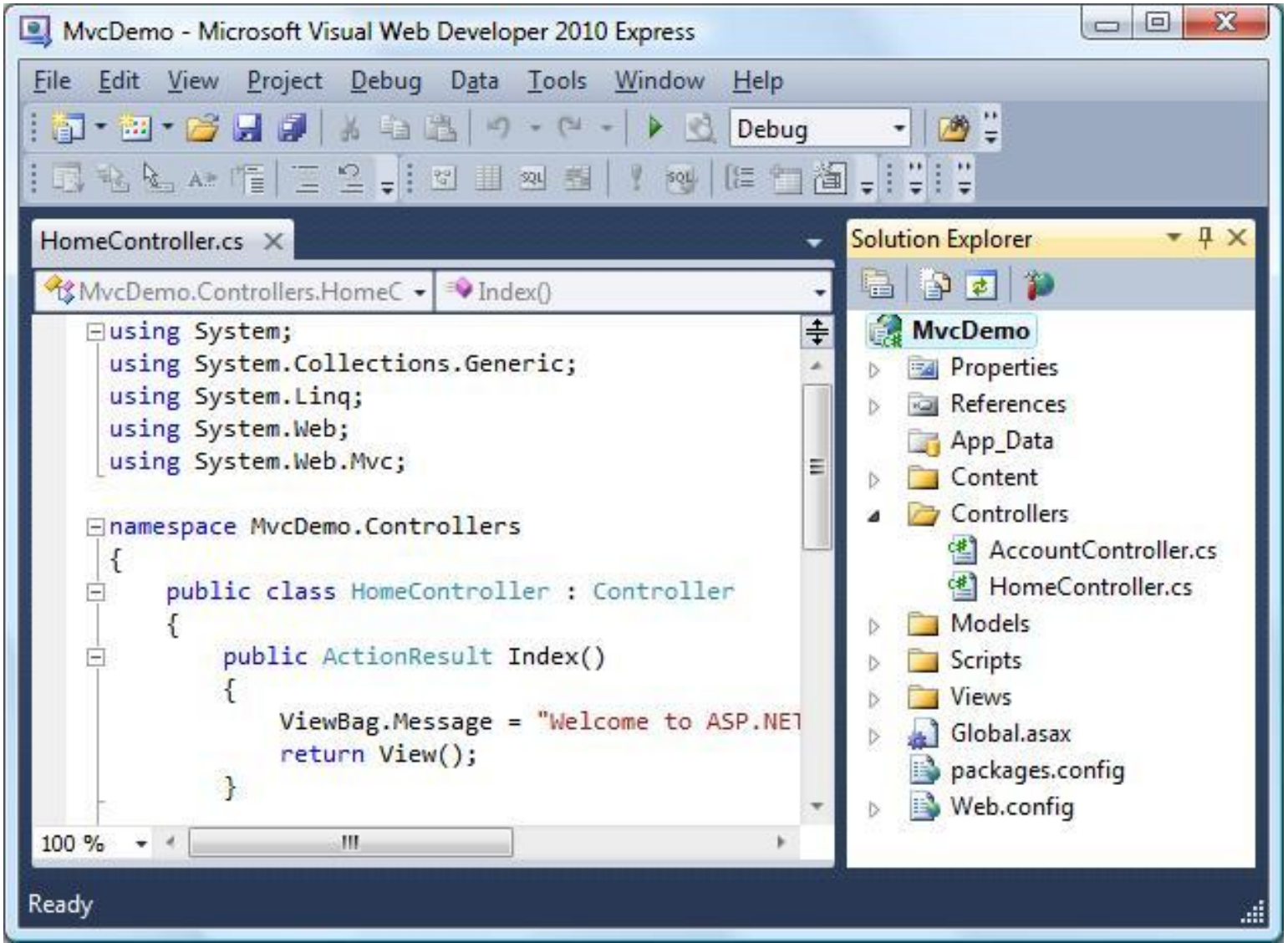
- If you have Visual Web Developer installed, start Visual Web Developer and select **New Project**.



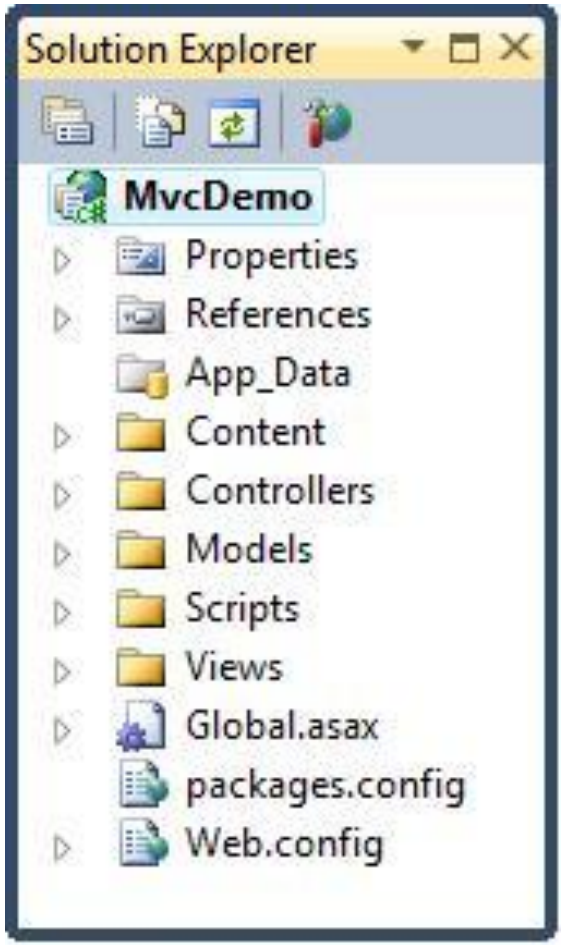
Steps

- In the New Project dialog box:
  Open the **Visual C#** templates
  Select the template **ASP.NET MVC 3 Web Application**
  Set the project name to **MvcDemo**
  Set the disk location to something like **c:\example_demo**
  Click **OK**
- When the New Project Dialog Box opens:
  Select the **Internet Application** template
  Select the **Razor Engine**
  Select **HTML5 Markup**
  Click **OK**

Visual Studio Express will create a project much like this:



MVC Folders



- **Application information**
  Properties
  References
- **Application folders**
  App_Data Folder
  Content Folder
  Controllers Folder
  Models Folder
  Scripts Folder
  Views Folder
  Configuration files
  Global.asax
  packages.config
  Web.config

Advantages

- Clear separation between presentation logic and business logic.
- Each object in mvc have distinct responsibilities.
- parallel development
- easy to maintain and future enhancements
- All objects and classes are independent of each other.

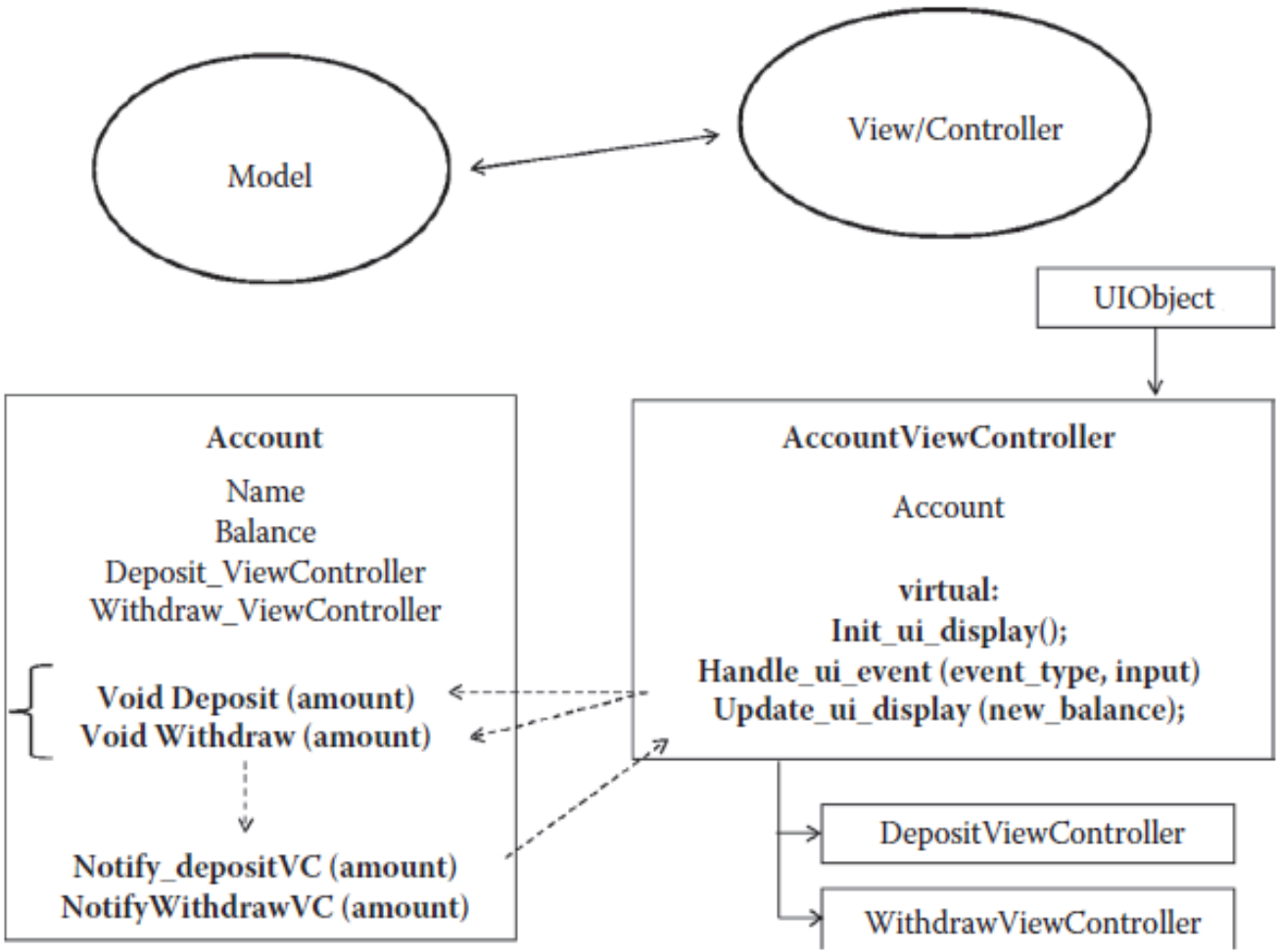
Disadvantages

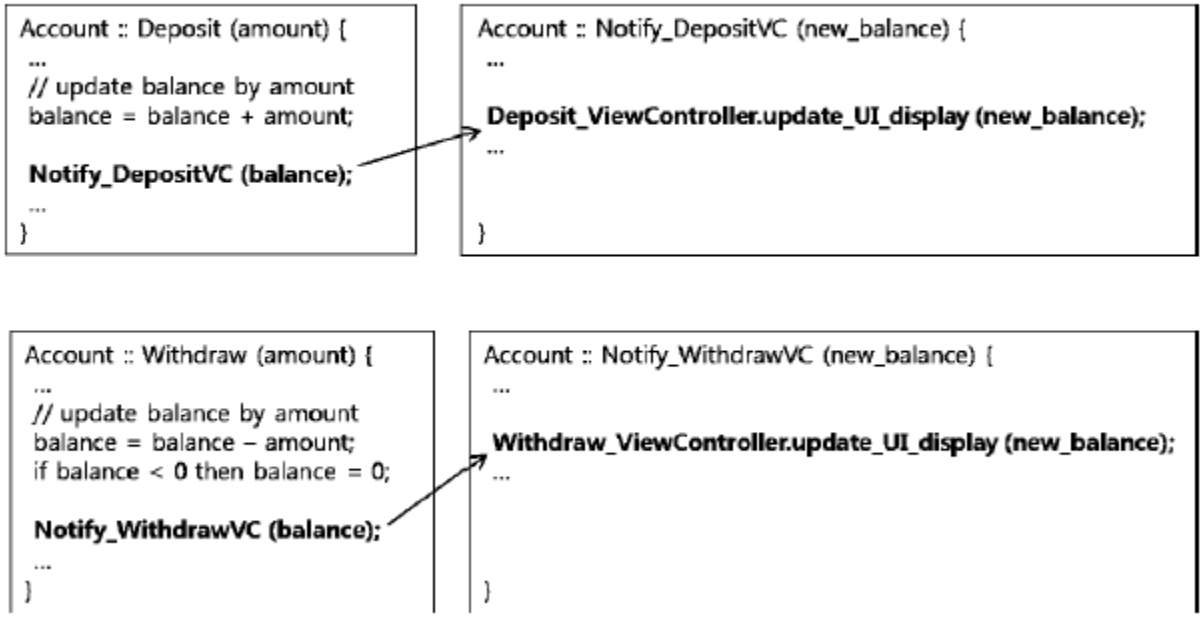
- Increased complexity
- Inefficiency of data access in view
- Difficulty of using MVC with modern user interface too.
- For parallel development there is a needed multiple programmers.
- Knowledge on multiple technologies is required.

Example

- The Observer pattern allows the BankAccount class to notify multiple views without minimal information.
- Observers can register themselves with their Subjects. No strings attached!

Example of MVC Implementation 1: Simple Bank Application

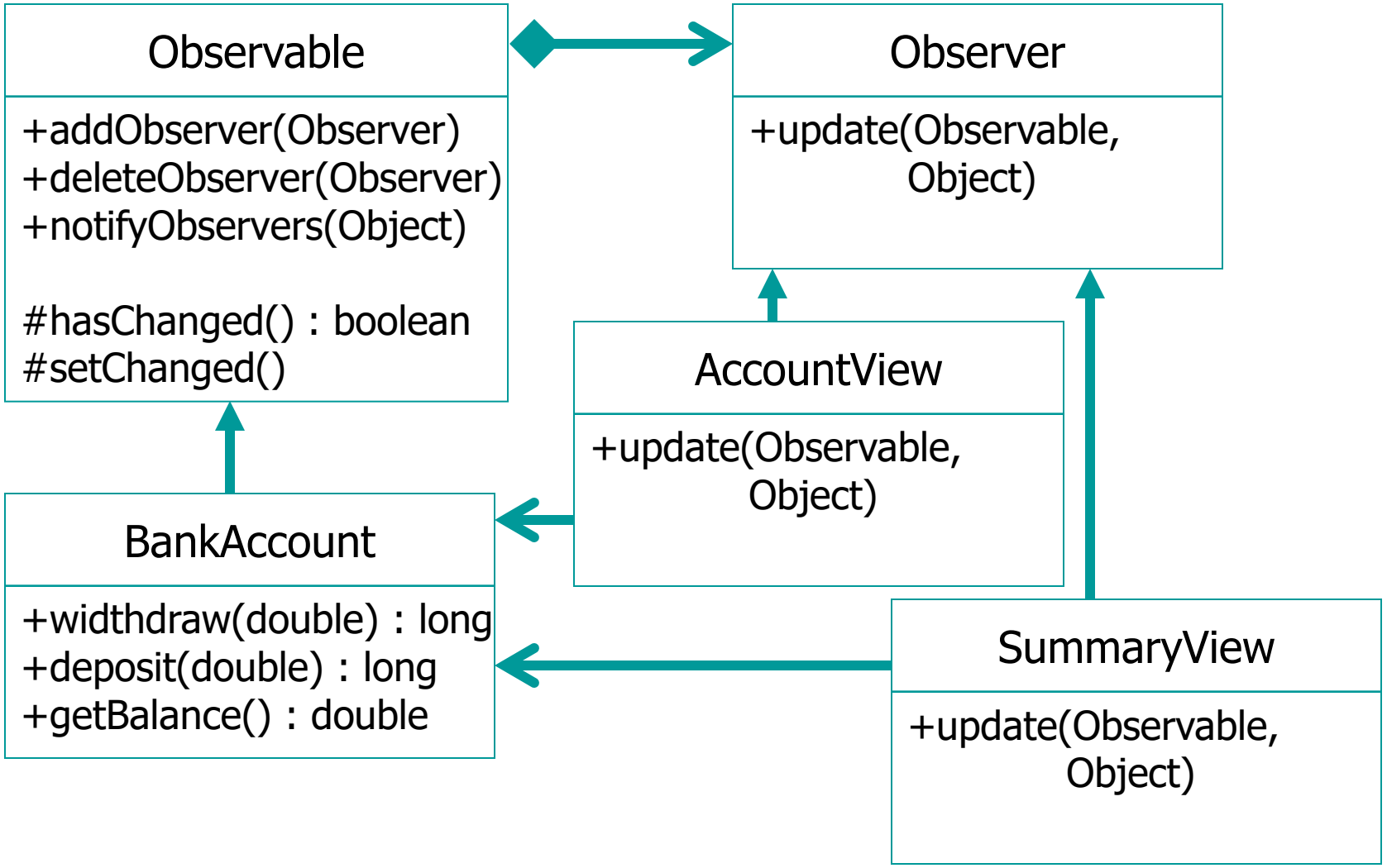



Account :: Deposit (amount) {
...
// update balance by amount
balance = balance + amount;
Notify_DepositVC (balance);
...
}
Account :: Notify_DepositVC (new_balance) {
...
Deposit_ViewController.update_UI_display (new_balance);
...
}
Account :: Withdraw (amount) {
...
// update balance by amount
balance = balance – amount;
if balance < 0 then balance = 0;
Notify_WithdrawVC (balance);
...
}
Account :: Notify_WithdrawVC (new_balance) {
...
Withdraw_ViewController.update_UI_display (new_balance);
...
}

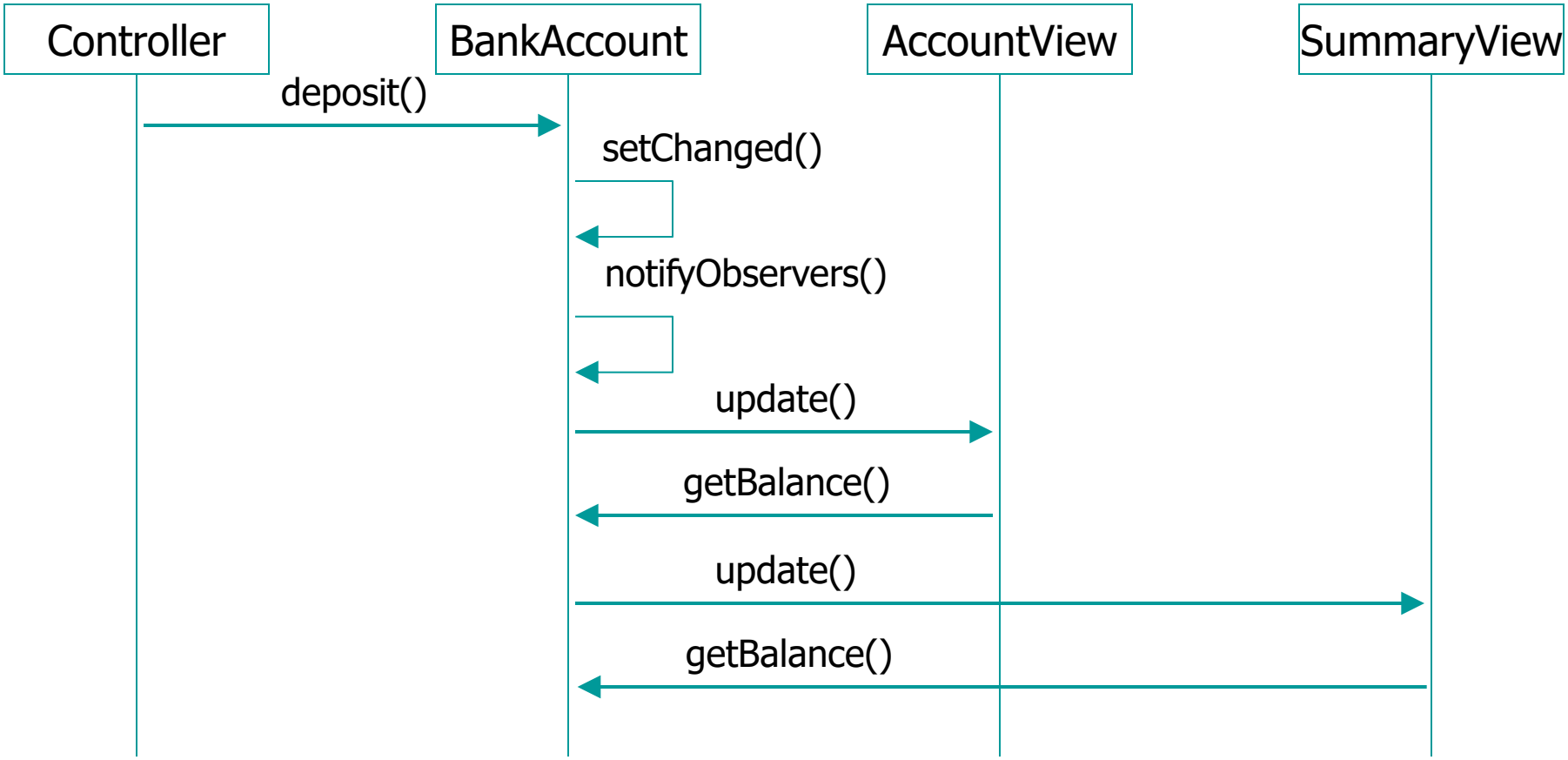
```
DepositViewController :: AccountViewController {

 ...

 // Implement the subclass specific virtual methods

 // create and initialize UI display (the view) for deposit
 void Init_ui_display ();

 // redraw the UI display with new_balance
 void Update_ui_display (new_balance);

 // handle events from user (the controller)
 void Handle_ui_event (event_type, input);

}
```

```
DepositViewController :: Handle_ui_event (event_type, input) {

 ...

 // get the model
 my_model = Get_model ();

 // if the user input is text input
 if (event_type == TEXT_INPUT) {
                // convert the text input to integer
                deposit_amount = convert_to_int (input);
                // make a request to model to deposit
                my_model->Deposit (deposit_amount);
```

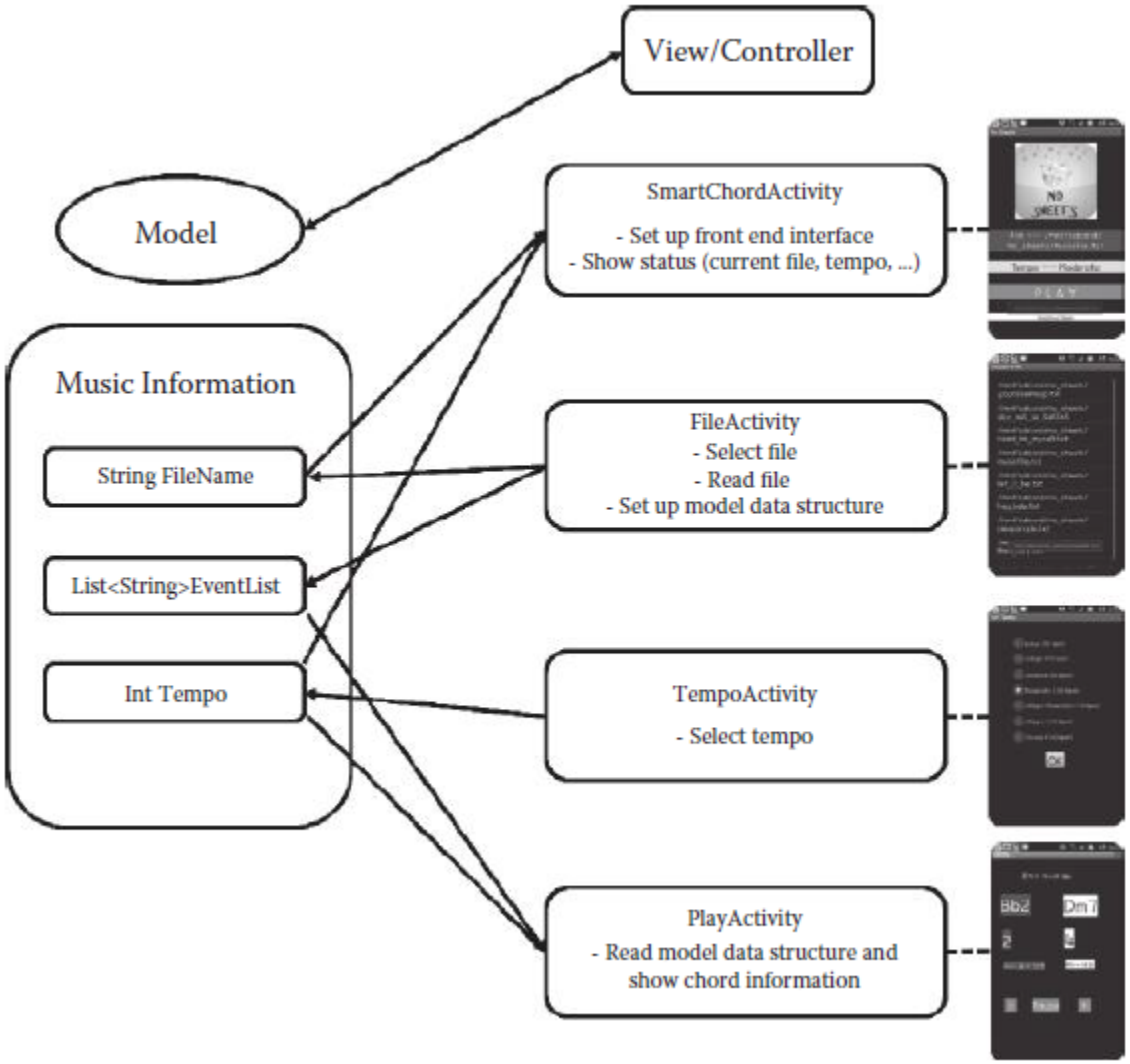
Observer Class Diagram



Transactions Happen!



No sheet


View/Controller
Model
Music Information
String FileName
List<String>EventList
Int Tempo
SmartChordActivity
- Set up front end interface
- Show status (current file, tempo, ...)
FileActivity
- Select file
- Read file
- Set up model data structure
TempoActivity
- Select tempo
PlayActivity
- Read model data structure and show chord information