Software Design Pattern
- Model-View-Controller

CS580 Advanced Software Engineering
http://cs580.yusun.io
October 20, 2014

Yu Sun, Ph.D.
http://yusun.io
yusun@csupomona.edu

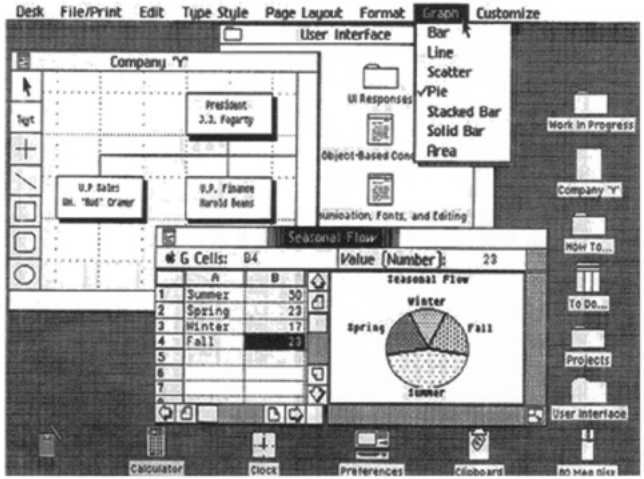
CAL POLY POMONA

Model–View–Controller (MVC)

- Invented by Trygve Reenskaug and introduced into the Smalltalk-80 programming environment developed at Xerox PARC
- MVC was central to the architecture of the multi-windowed Smalltalk environment used to create the first graphical user interfaces
- The approach taken was borrowed by the developers of the Apple Macintosh and many imitators in the years since
- Elements of MVC appear in many modern GUIs (MFC, Swing, …)

MVC Motivation

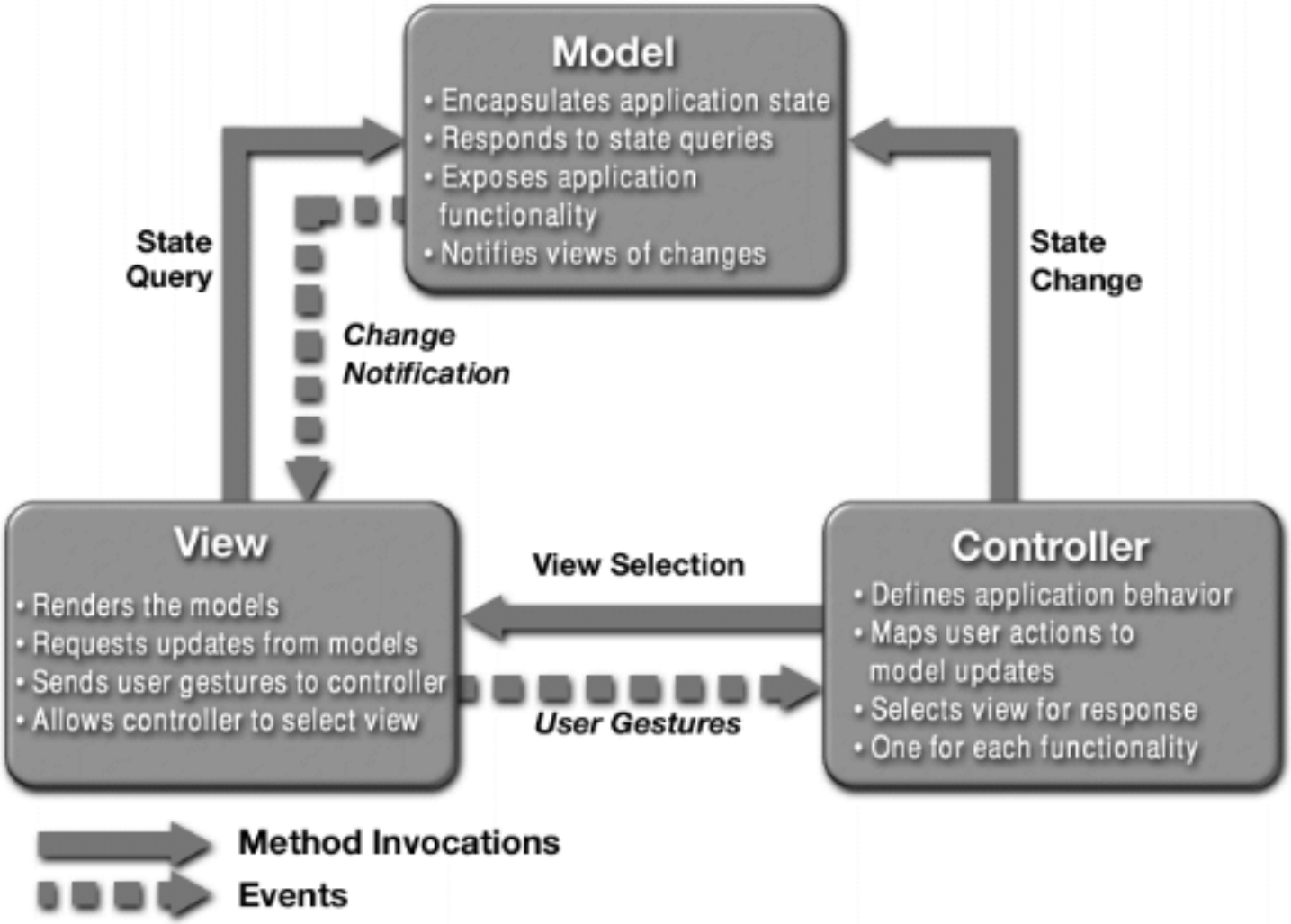
- The UI of an application is subject to many changes
  - Change of UI for different users
  - Same info can be shown in different windows
  - Changes to underlying data should be reflected quickly everywhere
  - Changes to UI should be easy, even at runtime
  - Different "look and feel" should not affect functional core
- So separate *processing*, *output*, and *input*



MVC

- MVC divides application into:
  - Model of core functionality and data
  - Views displaying information to user
  - Controllers handling user input
- Views and Controllers comprise UI
- Change-propagation mechanism ensures consistency between Model and UI
  - Event-driven programming

MVC Architecture



http://java.sun.com/blueprints/patterns/MVC-detailed.html

Model

- Store and manage data elements, such as state information
- Encapsulates application-specific data and functionality
  - Methods to edit data, which Controller can call
  - Methods to access state, which View and Controller can request
- Maintains registry of dependent Views and Controllers to be notified about data changes
- Examples
  - Text editor: model is text string
  - Spreadsheet: collection of values related by functional constraints

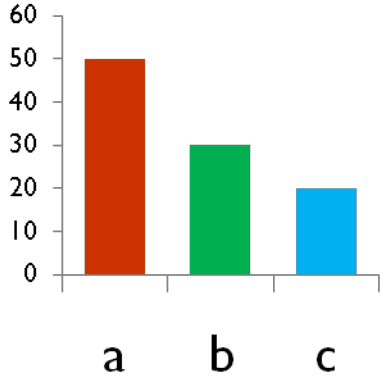
View

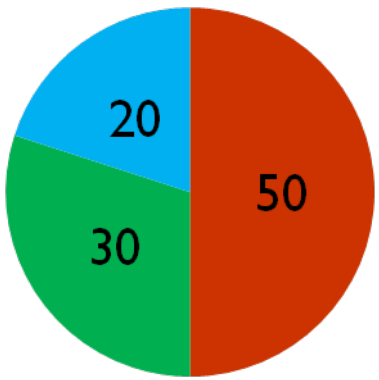
- Mechanism needed to map model data to rendition (view / display)
- When Model changes, View is informed
  - View requests relevant model information
  - View arranges to update screen
- Examples
  - Text editor: colored text (e.g., code editor)
  - Spreadsheet: tabular representation, bar chart, histogram

MVC Concepts – *multiple views*

- Any number of Views can subscribe to the Model

a	b	c
50	30	20





a = 50%
b = 30%
c = 20%

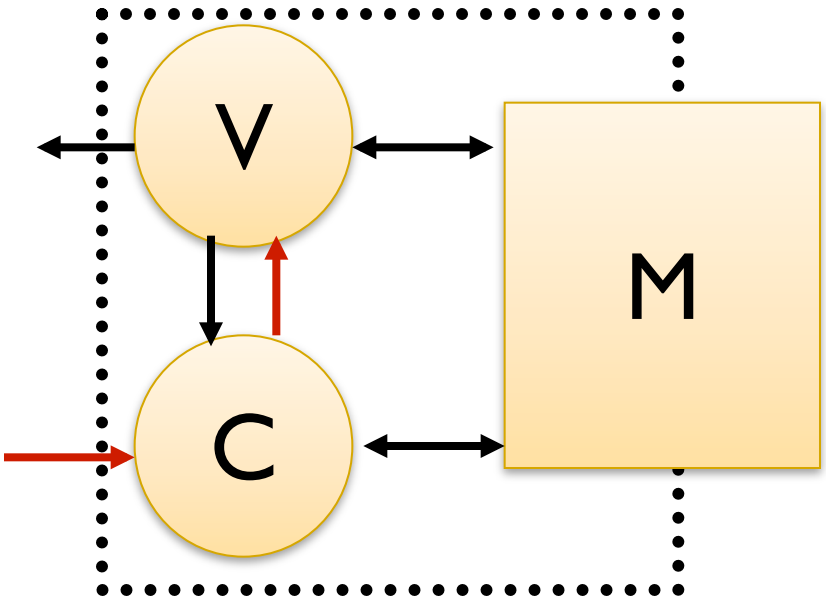
Controller Tasks

- Receive user inputs from mouse and keyboard
- Map these into commands that are sent to the Model and/or viewport to effect changes in the View
  - e.g., detect that a button has been pressed and inform the Model that the button state has changed
- Examples
  - Textual commands
  - Mouse (point and click) commands

MVC Dynamics

1. User input event routed by Window System to appropriate Controller

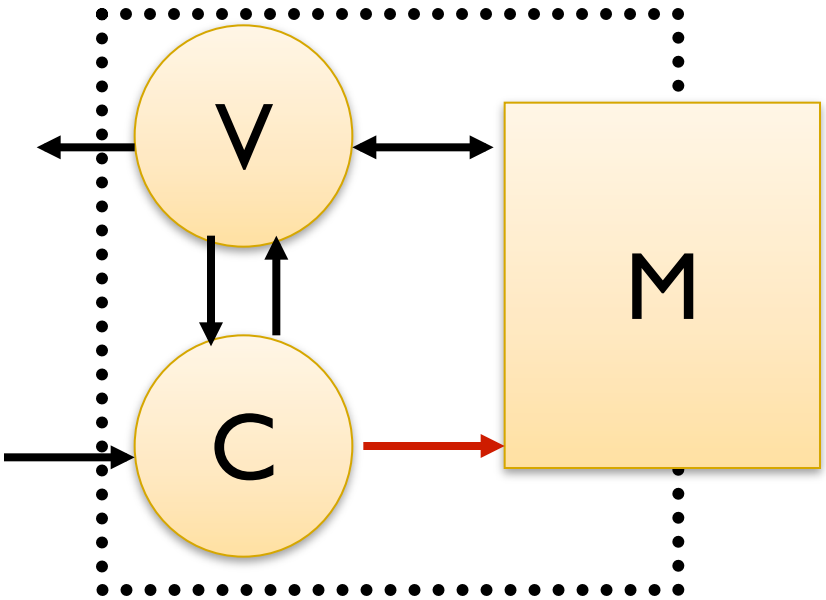
2. Controller may require View to "pick" object of focus for event



MVC Dynamics

3. Controller requests method of Model to change its state

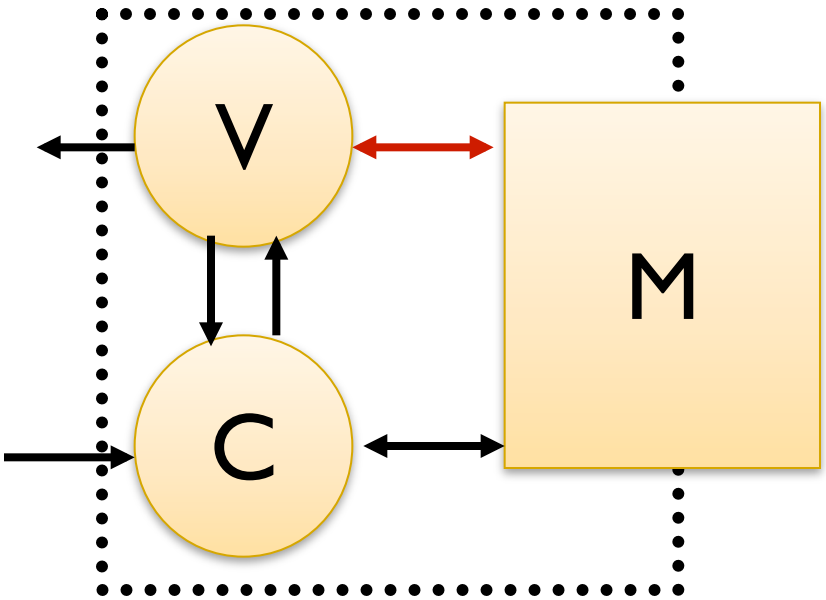
4. Model changes its internal state



MVC Dynamics

5. Model notifies all dependent Views that data has changed

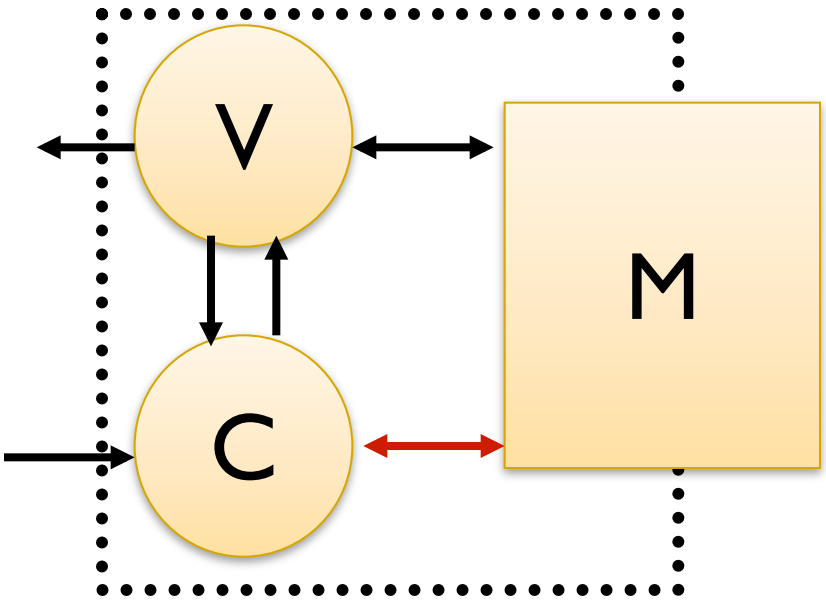
6. View requests from Model current data values



MVC Dynamics

7. Model notifies all dependent Controllers that data has changed

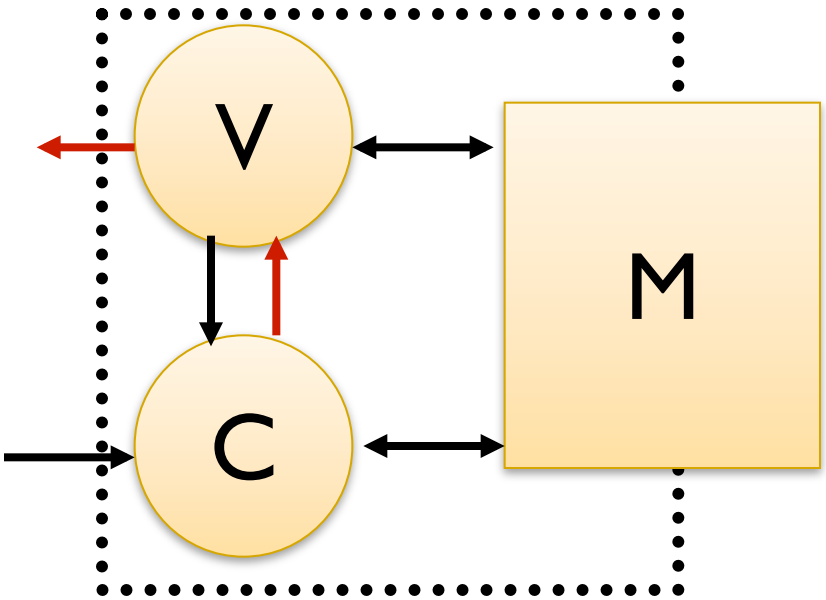
8. Controller requests from Model current data values



MVC Dynamics

9. Controller informs View if elements are disabled

10. View requests redraw



View + Controller Linking

- Controller almost always has to "talk to" view
  - Need geometry of output to interpret input (e.g., picking)
  - Need to do feedback
- As a result, the View and Controller tend to be very tightly coupled, and considered as one
  - M(VC)
- Multiple View/Controller pairs can be attached to a single Model

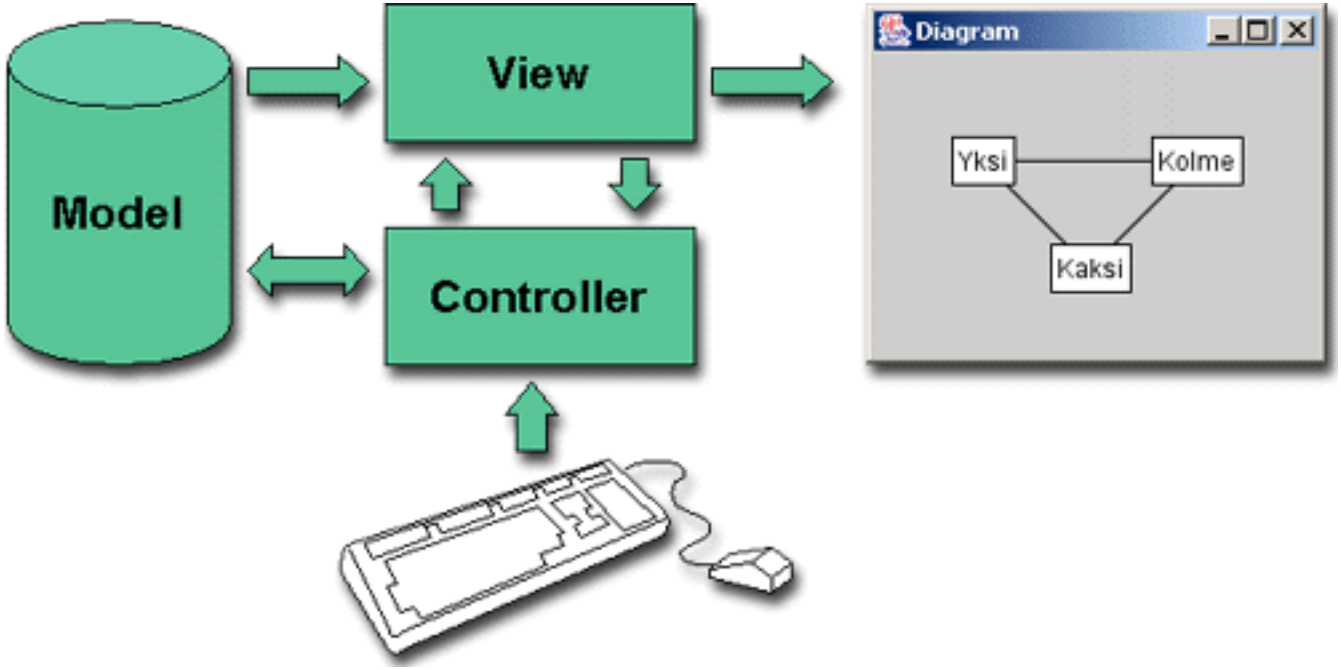
Benefits of MVC Architecture

- Improved maintainability
  - Due to modularity of software components
- Promotes code reuse
  - Due to OO approach (e.g., subclassing, inheritance)
  - e.g., extending a base EventListener class
- Model independence
  - Designers can enhance and/or optimize model without changing the view or controller (the secret!)
- Pluggable look and feel
  - New L&F without changing model
  - Multiple views use the same data

MVC: The Pros and Cons

- Pro:
  - Multiple views of same model
  - Synchronized views
  - Pluggable V & C and "look and feel"
- Con:
  - Complexity for simple interactions
  - Potentially excessive updates/messages
  - Tight coupling, in practice (V-C, VC-M)

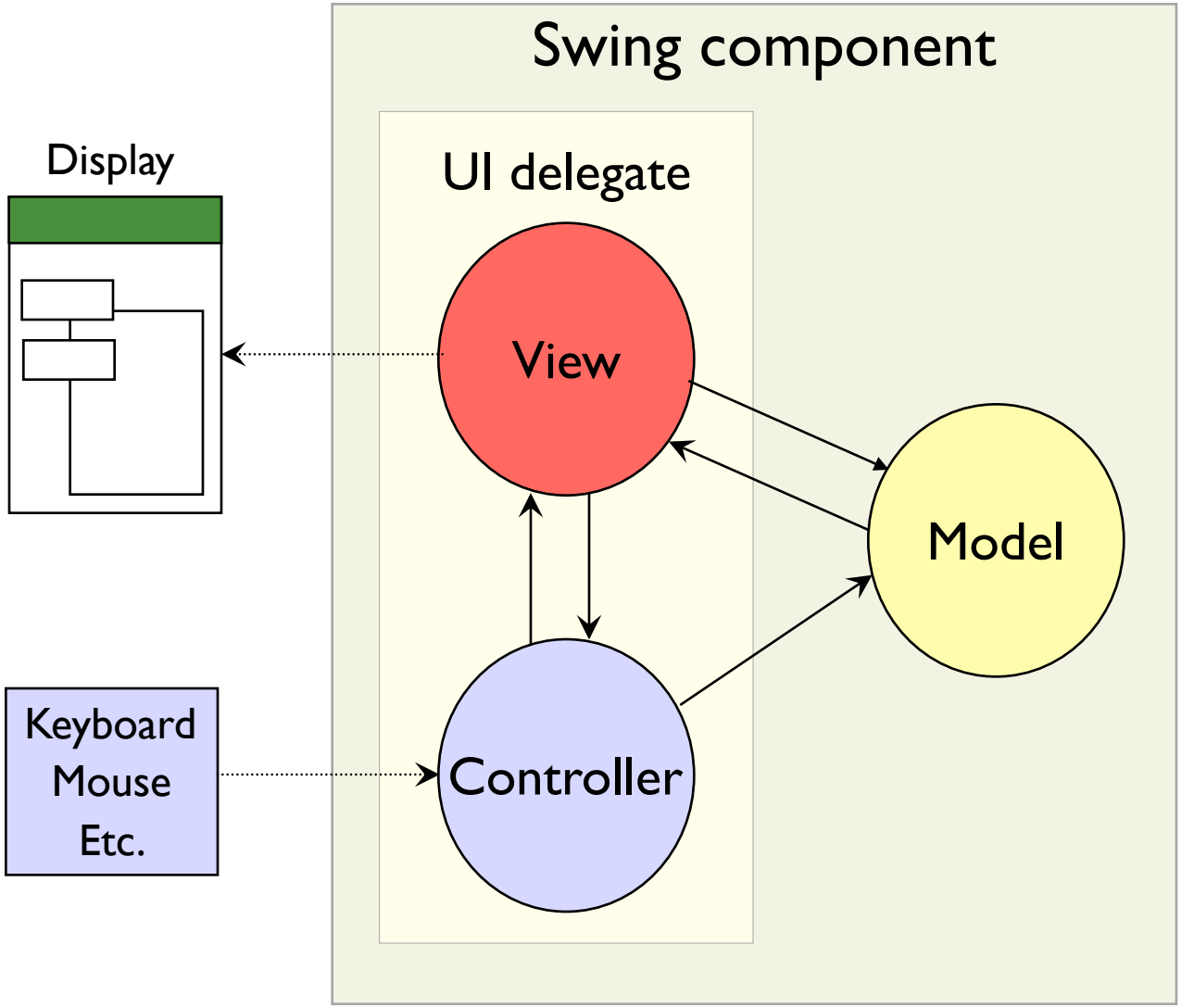
MVC Example – Swing



VC – Swing Views and Controllers

- In Swing, the term look and feel (L&F) is common
  - The look is the view
  - The feel is the controller
- In practice, the view and controller parts of MVC are very tightly connected
- Swing combines the view and controller into a single entity known as a UI delegate
- Advantage
  - Combining the view and controller allows the appearance and behavior (L&F) of a component to be treated as a single unit, thus facilitating changes to the UI; otherwise, combination of correct model and viewers needed for each L&F
- This is known as pluggable look and feel

MVC and Swing



M – Swing Models

- In Swing, many models exist as interfaces
  - e.g., ButtonModel, BoundedRangeModel, ComboBoxModel, ListModel, ListSelectionModel, TableModel, Document
- The interface is implemented in model classes
- Usually there is a default model class that is automatically associated with a component
  - e.g., DefaultButtonModel implements ButtonModel
  - e.g., AbstractDocument implements Document (PlainDocument is a subclass of AbstractDocument)

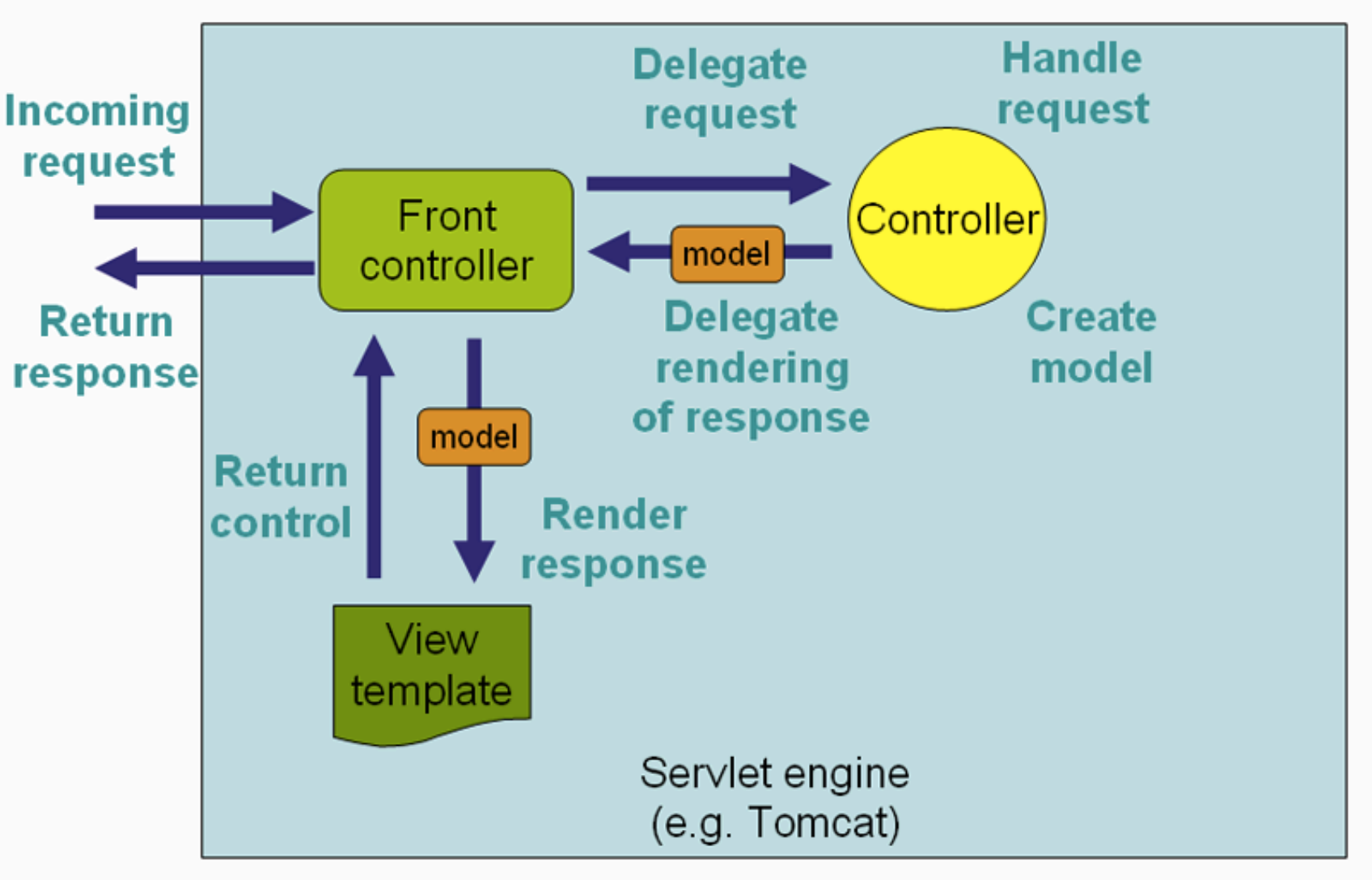
TableModel Example

DemoTable

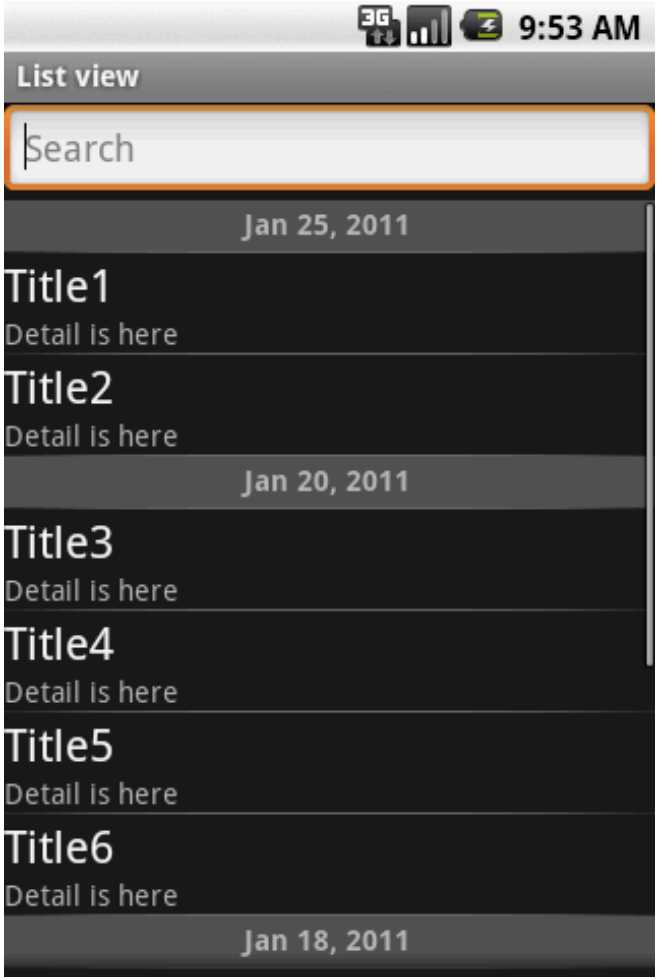
Performance	Time	Athlete	Nation	Place	Date
100 METRES	9.79	Maurice Greene	USA	Athína	16 06 1999
200 METRES	19.32	Michael Johns...	USA	Atlanta GA	01 08 1996
400 METRES	43.18	Michael Johns...	USA	Sevilla	26 08 1999
800 METRES	1:41.11	Wilson Kipketer	DEN	Köln	24 08 1997
1000 METRES	2:11.96	Noah Ngeny	KEN	Rieti	05 09 1999
1500 METRES	3:26.00	Hicham El Gu...	MAR	Roma	14 07 1998
1 MILE	3:43.13	Hicham El Gu...	MAR	Roma	07 07 1999
2000 METRES	4:44.79	Hicham El Gu...	MAR	Berlin	07 09 1999
3000 METRES	7:20.67	Daniel Komen	KEN	Rieti	01 09 1996
5000 METRES	12:39.36	Haile Gebrsel...	ETH	Helsinki	13 06 1998
10000 METRES	26:22.75	Haile Gebrsel...	ETH	Hengelo	01 06 1998

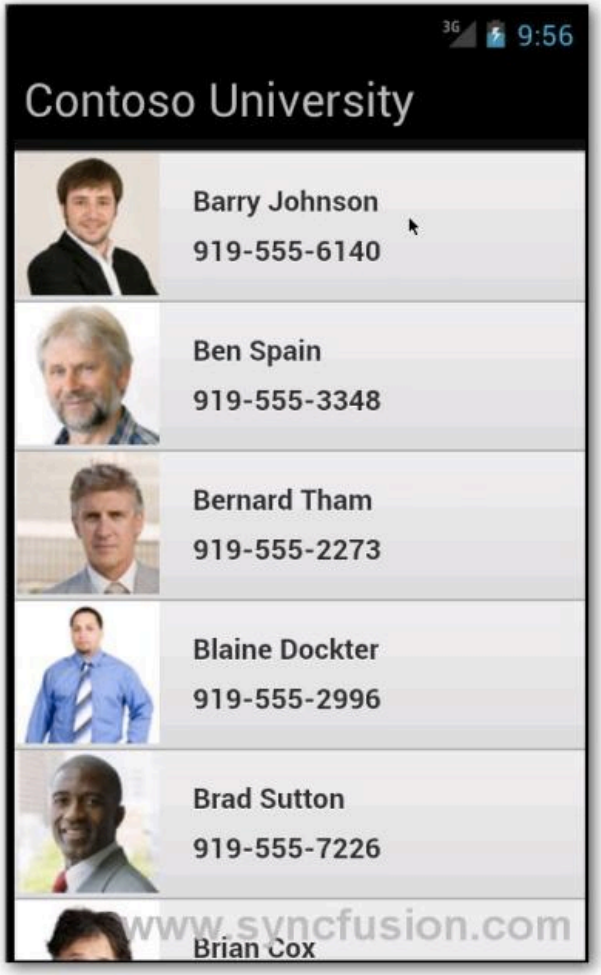
Instead of passing the data directly to the `JTable` object, we create a data model, pass the data to a `TableModel`, then pass the model to the `JTable` object

MVC Example – Spring

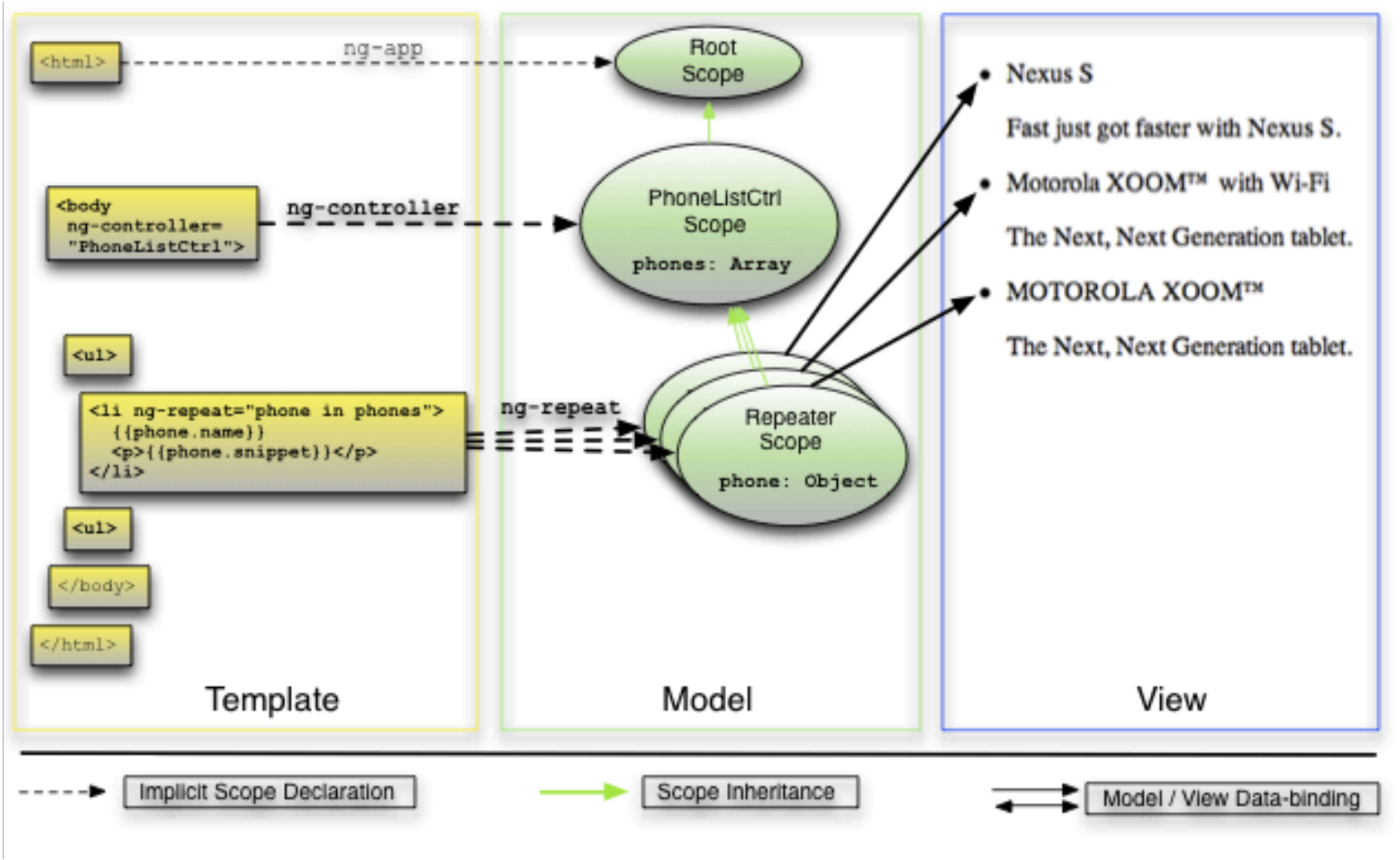


MVC Example – Android





MVC Example - AngularJS



· https://angularjs.org/