Model View Controller Pattern

V. Paúl Pauca

Department of Computer Science
Wake Forest University

CSC 331-631
Fall, 2013

Design Patterns

Differ in granularity and level of abstraction

Classification of Design Patterns I

Classified by:

- purpose (what a pattern does), and
- scope (applies primarily to classes or to objects)

Classification of Design Patterns II

Purpose

- Creational: concerned with process of object creation
- Structural: deal with composition of objects or classes
- Behavioral: characterize ways of interaction between classes/objects and distribution of responsibility

Scope

- Class: deal with relationships between classes and subclasses (compile time, static)
- Object: deal with object relationships (run time, dynamic)

Classification of Design Patterns III

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method	Adapter	Interpreter Template method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Command Iterator Mediator Memento Observer State Strategy Visitor

Gamma *et al., 1994*

Should do your own research about most current classification

Model-View-Controller Pattern I

T. Reenskaug, 1979, “Applications Programming in Smalltalk-80: How to use Model-View-Controller”

Google says:

Model-View-Controller Pattern II

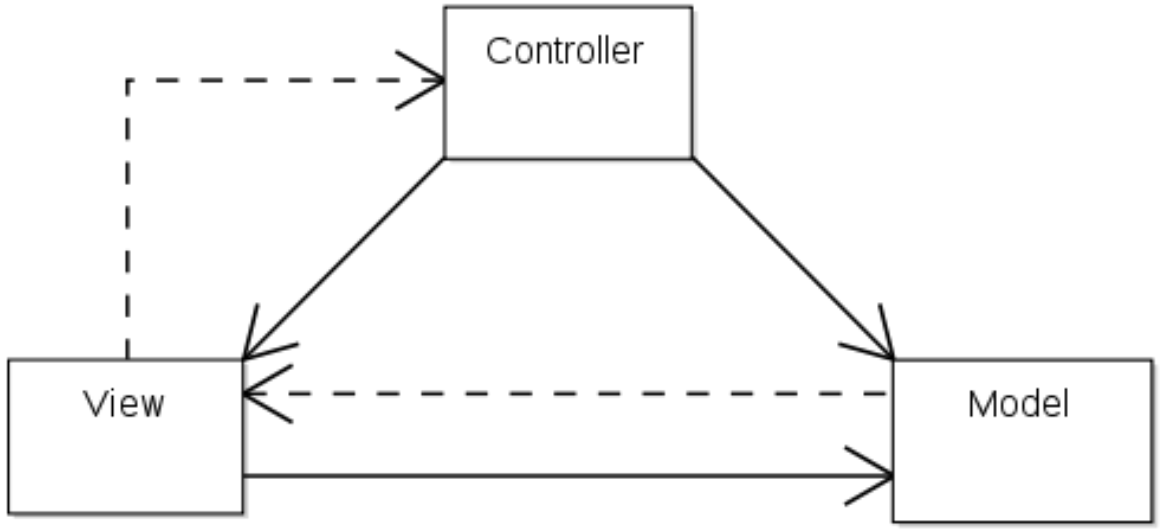
Applications

- GUI frameworks
  - Model-view-presenter in .NET framework
  - Model-view-controller-connector in Xforms
  - Cocoa: IB constructs views, connects them to controllers via outlets and actions
  - GTK+
  - MFC: document/view architecture
  - Java Swing
- Web-based frameworks
  - Implemented by web content management systems
  - ABAP, Actionscript, ASP, ColdFusion, Flex, Groovy, Java, PHP, Ruby, etc. (too many to list)

The Model-View-Controller (MVC) Pattern

- **Problem:** Applications requiring user interface and underlying data access
- **Solution:** MVC decouples the data (model) from the user interface (view), reducing architectural design complexity
- **Consequences:** MVC increases design flexibility and promotes reusability. Facilitates independent development, testing, and maintenance.

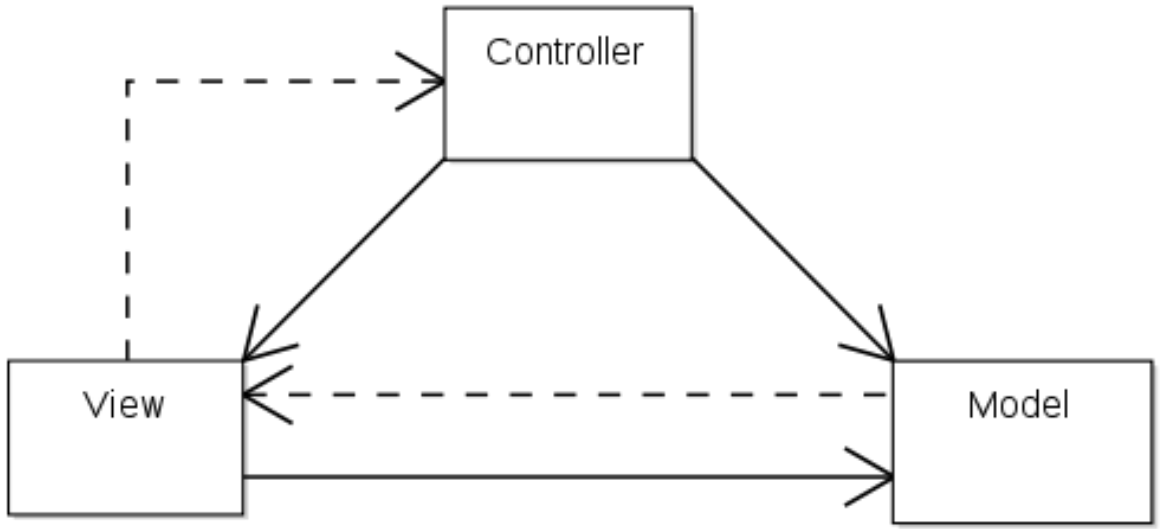
Implementation I



- _____ encapsulates the data to be used
- _____ displays the data and interacts with the user
- _____ decouples the view from the model
- _____ has a `View` and a `Model`
- _____ has a _____ (needs data to display)

Dashed arrows indicate use of observer or delegation

Implementation II



- `Model` encapsulates the data to be used
- `View` displays the data and interacts with the user
- `Controller` decouples the view from the model
- `Controller` has a `View` and a `Model`
- `View` has a `Model` (needs data to display)

Dashed arrows indicate use of observer or delegation

Components

- Model
  - Domain-specific representation of data
    Models encapsulate the data access
  - When model changes state, it notifies its associated views

Components

- Model
  - Domain-specific representation of data
    Models encapsulate the data access
  - When model changes state, it notifies its associated views
- View
  - Renders model in a user interface
  - Many-to-one relationship with models

Components

- Model
  - Domain-specific representation of data
    Models encapsulate the data access
  - When model changes state, it notifies its associated views
- View
  - Renders model in a user interface
  - Many-to-one relationship with models
- Controller
  - Receives input and initiates a response by calling model objects

Dynamic Behavior of MVC

(Event Driven behavior) order the following actions:

- Controller notifies the model of the user action
- User generates event through user interface, e.g. mouse click, touch events
- View changes itself as a result of a query or notification by model or controller
- Controller handles the input via a registered handler or callback, i.e. turns event into user action

MVC in .NET

- Defined in `System.Web.Mvc` namespace
- Supports part of the `System.Web` namespace

MVC in .NET

- Defined in `System.Web.Mvc` namespace
- Supports part of the `System.Web` namespace

Models

- Often retrieve and store model state in a database
- E.g. a `Product` object retrieves info from a database and writes info to a `Products` table in SQL server

MVC in .NET

- Defined in `System.Web.Mvc` namespace
- Supports part of the `System.Web` namespace

Models

- Often retrieve and store model state in a database
- E.g. a `Product` object retrieves info from a database and writes info to a `Products` table in SQL server

Views

- Display the application's user interface
- Created from the data model
- E.g. Edit view of a `Products` table

MVC in .NET

- Defined in `System.Web.Mvc` namespace
- Supports part of the `System.Web` namespace

Models

- Often retrieve and store model state in a database
- E.g. a `Product` object retrieves info from a database and writes info to a `Products` table in SQL server

Views

- Display the application's user interface
- Created from the data model
- E.g. Edit view of a `Products` table

Controllers

- Handle user interaction, work with the model, select views for display
- E.g. handles query-string values, passes values to the model

MVC in Web Applications I

Sample app: a web forum, (from `perlmonks.org/?node_id=402070`)

Model

- Handles state of the application
- Does not know anything about HTML, web servers, etc.
- E.g. `Class::DBI` objects representing threads, users and postings

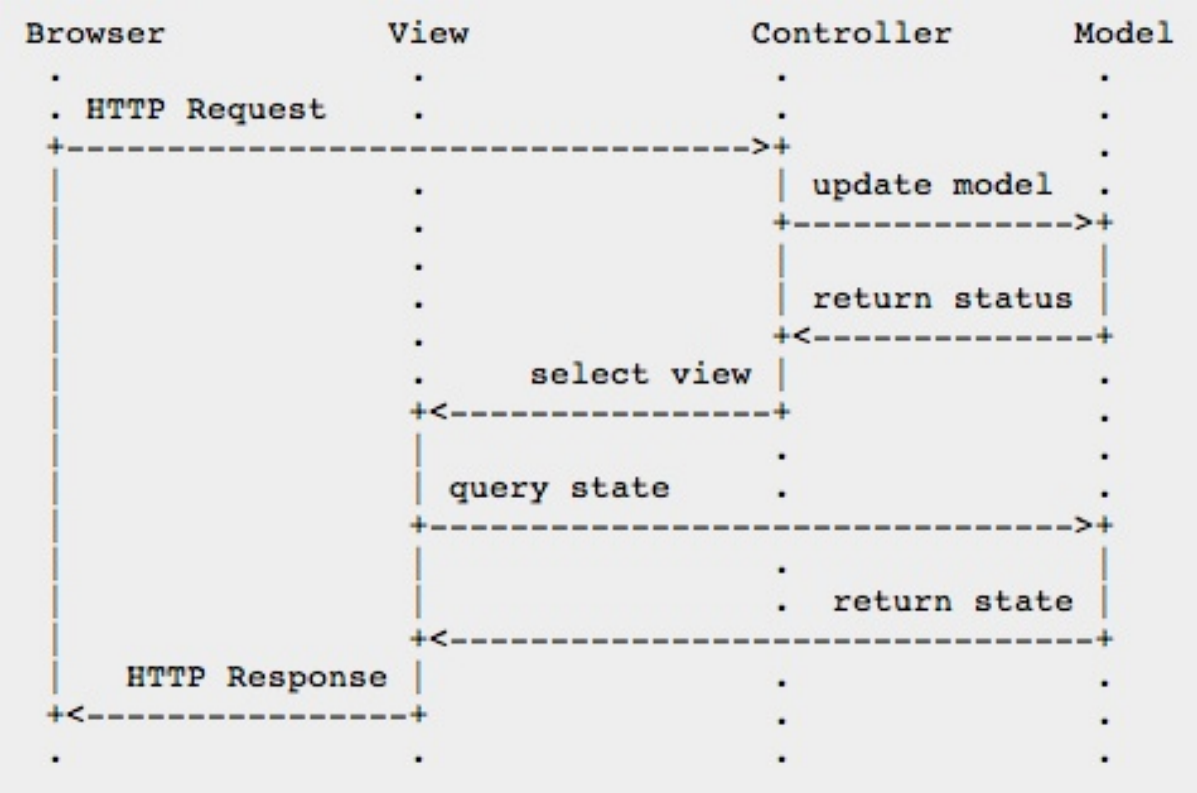
View

- Implemented as a template that renders a HTML page
- Cannot change state of the model
- E.g. templates to render login page, posting page, etc.

Controller

- Receives HTTP requests and translates them into actions for the model
- Selects appropriate view to handle the response

MVC in Web Applications II

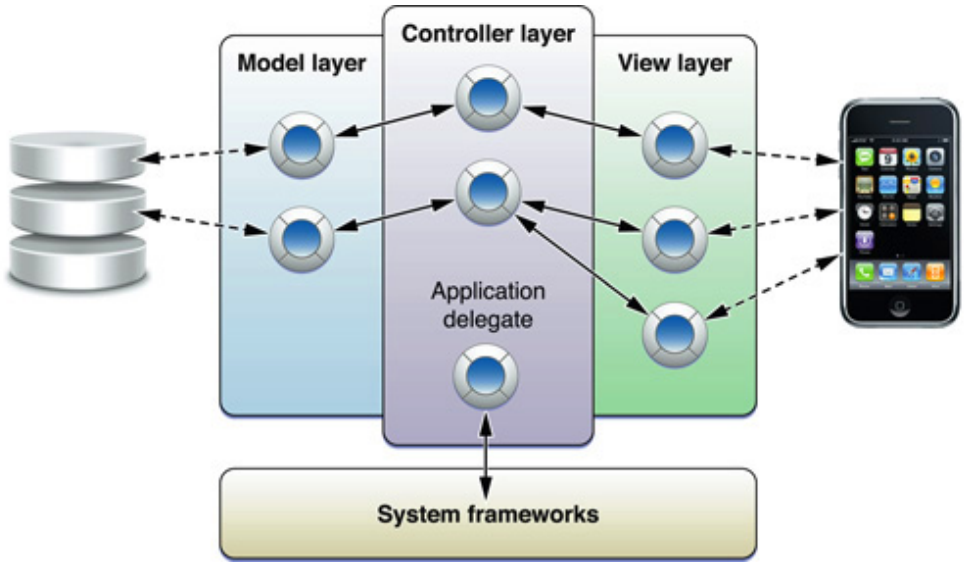


MVC in Web Applications III

Benefits

- *Separates request from pages*
  No coupling between user requests and HTML pages
- *Views are simple*
  Changing the page layout does not involve touching the application logic
  Less chance of adding bugs if layout is changed later
- *Shields the model implementation*
  Can change the model implementation without touching the user interface, as long as the model API doesn't change

MVC in XCode



- The `View` object is typically embedded in a XIB
- The `Model` often uses CoreData

ViewController Example