WKW SCHOOL

2020

CI6206 Internet Programming

MVC

Wong Twee Wee

Ver1.1

WEB DEVELOPMENT IN JAVA

- In the beginning… servlet -only solution
  - Problem
    - Web developers may not be good designers
  - Solution
    - Designers work on HTML UI, developers work on functionality
    - UI then handed to developers for integration
  - Problems
    - Generating UI using servlets
    - Maintaining/updating the UI

Moral: Too much HTML mixed with Java code

WEB DEVELOPMENT IN JAVA

- Next step – JavaServer Pages
  - Parsing only done once
  - Standard template language
  - Easy to mix logic and content
    - Designers code HTML UI
    - Developers add dynamic aspects
  - Problem
    - Maintenance

Moral: Too much Java mixed with HTML code

SERVLETS AND JSP

- Servlet-only solution works well when
  - There is not much output
  - Format/layout of page is highly variable
  - Lots of processing needed
- JSP-only solution works well when
  - Output is mostly character data
  - Format/layout mostly fixed

What if we have a mix of requirements?

DESIGN PATTERNS

- In software engineering, a **design pattern** is a general reusable solution to a commonly occurring problem within a given context in software design.
- Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.
  - https://en.wikipedia.org/wiki/Software_design_pattern

Other design patterns : http://www.tutorialspoint.com/design_pattern/

BENEFITS OF DESIGN PATTERNS

- Design patterns enable large-scale reuse of software architectures and also help document systems
- Patterns explicitly capture expert knowledge and design tradeoffs and make it more widely available
- Patterns help improve developer communication
- Pattern names form a common vocabulary

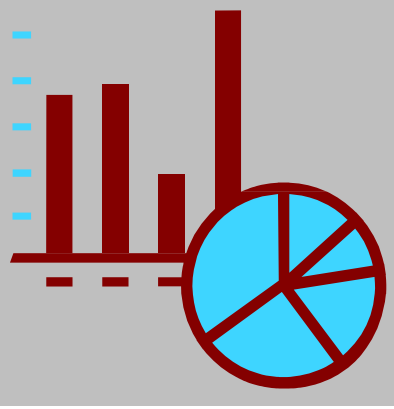
MODEL-VIEW-CONTROLLER PATTERN

- Problem
  - Applications need to support multiple types of users with multiple types of interfaces
  - Different applications may need to be developed
  - Non-interface-specific code is duplicated in each application
  - Difficult to determine what to duplicate since interface-specific and non-interface-specific code are intertwined

MODEL-VIEW-CONTROLLER PATTERN

- Solution
  - MVC pattern separates functionality, presentation and control logic
  - Allows multiple views to share the same data model
  - Makes supporting multiple clients easier to implement, test, and maintain

	A	B
1	Sales	Region
2	$5,968	North
3	$69,784	South
4	$85,721	East
5	$6,684	West



BENEFITS OF MVC

- Decoupling views and models
- Reduces the complexity of your design
- Makes code more flexible
- Makes code more maintainable

MODEL-VIEW-CONTROLLER PATTERN

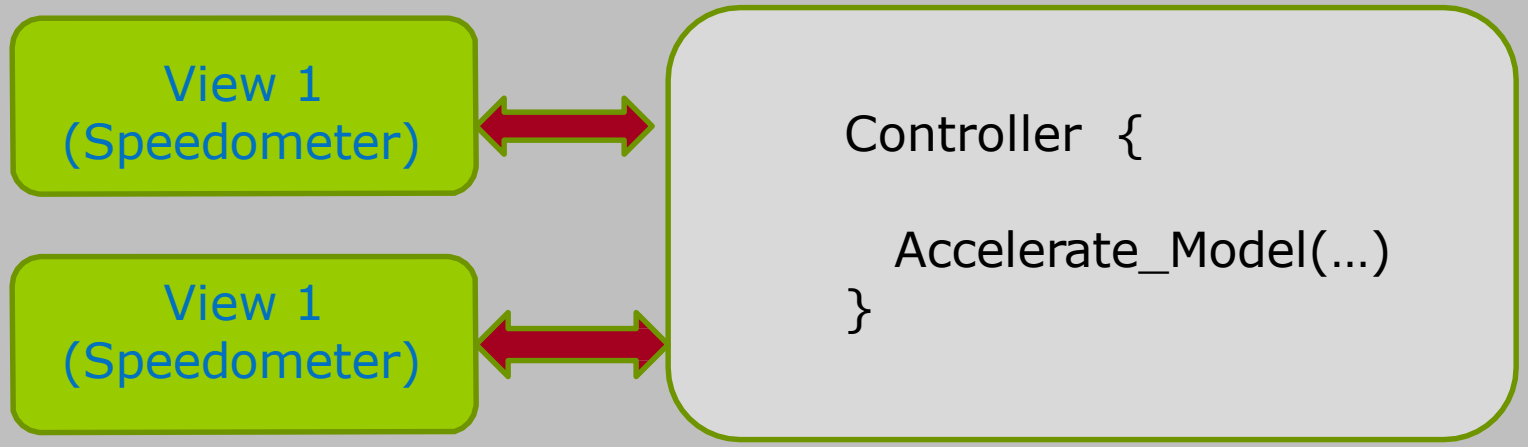
- Participants and Responsibilities
  - Model
    - Manages the behaviour and data of the application
  - View
    - Renders contents of a model to UI
    - Responsible for maintaining consistency in its presentation when the model changes
  - Controller
    - Mechanism by which model and view communicate
    - Processes user inputs and generates a response by operating on model objects (View → Model)

MODEL-VIEW-CONTROLLER PATTERN

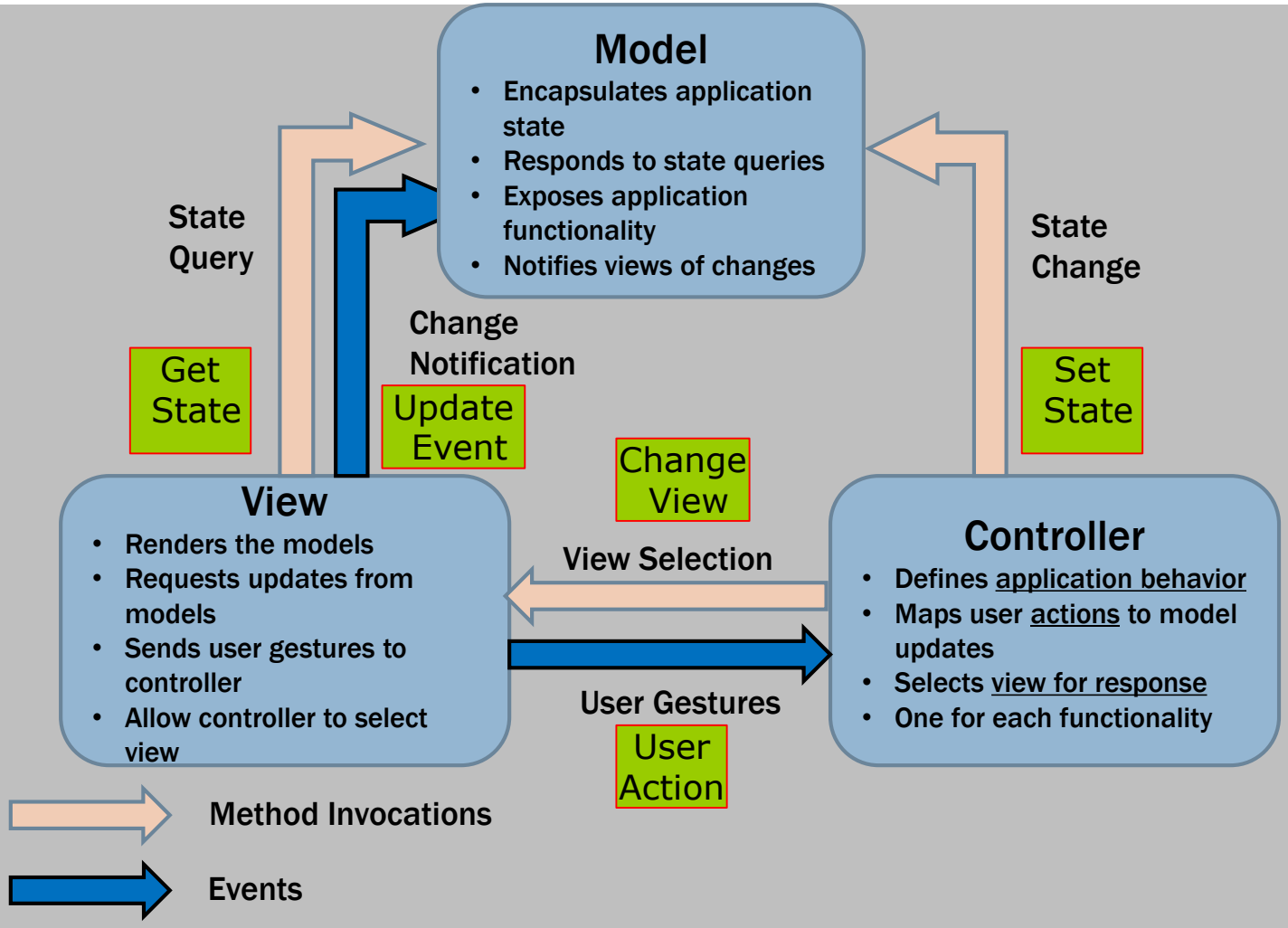
- Analogy (Automobile)
  - Speed of the car affected by the accelerator pedal (Controller)
  - Speed is manifested by the engine (Model)
  - Speed is shown by the speedometer (View)

  How about 2 speedometers ; 1 for the driver and 1 for the passenger ? (Different views but on the same Data)

ANALOGY (AUTOMOBILE)

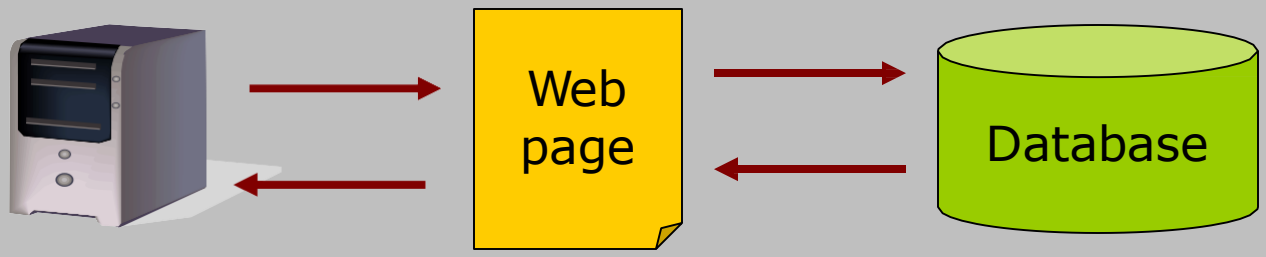


MODEL-VIEW-CONTROLLER PATTERN



WEB APPLICATIONS AND MVC

- Model 1
  - Traditional approach to Web development
  - Page-centric and decentralized
  - Web pages provide content
  - Next page to display determined by hyperlinks or request parameters on current page
  - All processing done within page (e.g. JSP or servlet)



WEB APPLICATIONS AND MVC

- Model 1
  - Advantages
    - Lightweight, simple design
    - Good for small, static applications
    - Suitable for Web applications that have very simple page flow
  - Problems
    - Hard to maintain
    - Design does not promote the division of labor

WEB APPLICATIONS AND MVC

- Model 2
  - Introduces a controller
  - Centralizes logic for dispatching requests to the next Web page based on
    - Request URL
    - Input parameters
    - Application state
  - Pros and cons
    - Easier to maintain and extend
    - More complex

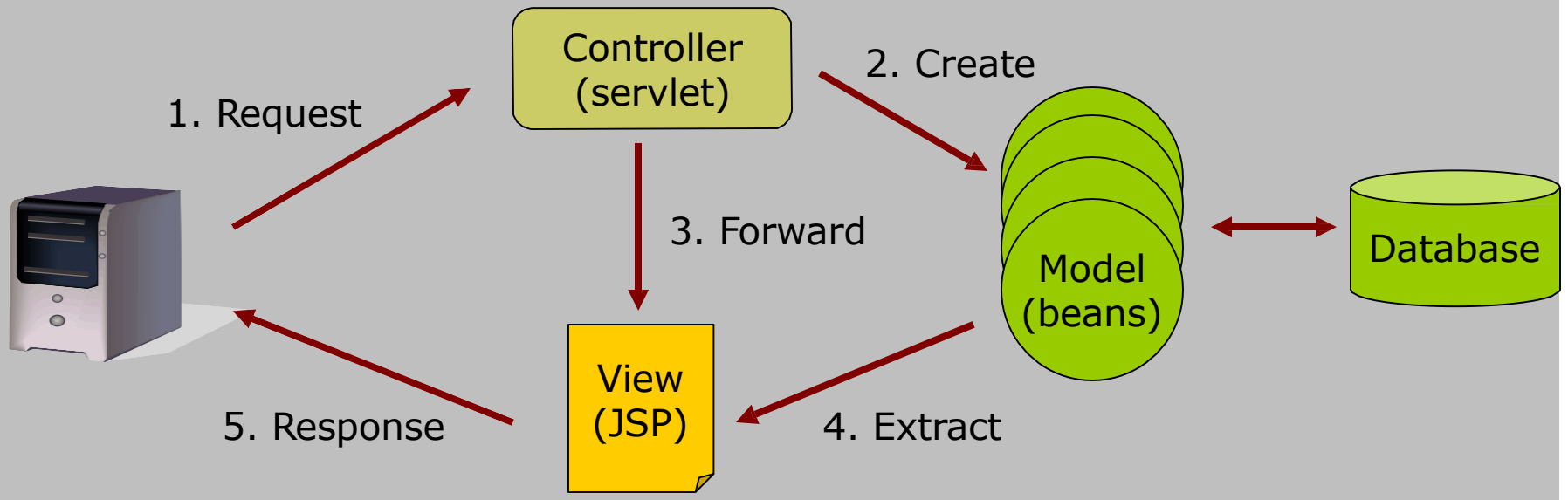
WEB APPLICATIONS AND MVC

- The model is the data and business/domain logic for your application
- The view is typically HTML generated by your application
- The controller receives HTTP requests and decides which domain objects to use to carry out specific tasks

http://www.tutorialspoint.com/design_pattern/mvc_pattern.htm

WEB APPLICATIONS AND MVC

- Model 2 components in Java
  - Model – JavaBeans
    - Reusable
  - View – JSP
    - No logic
  - Controller – servlet
    - No output



IMPLEMENTING MVC IN JAVA

- Defining beans to represent data (model)
  - See previous lecture notes
- Write controller servlet
  - First point of contact with user
  - Read request headers and form parameters
  - Process request
  - Instantiate and populate beans

IMPLEMENTING MVC IN JAVA

- Store beans for use by view
  - Use `setAttribute` on `HttpServletRequest`, `HttpSession` or `ServletContext` object
  - When do we use each one?
- Forward client's request to view
  - Use `RequestDispatcher.forward()`
- Extract data from beans
  - Bean location dependent on controller

IMPLEMENTING MVC IN JAVA

- Controller example

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    String data = request.getParameter("formdata");
    ...

    BeanObject beanObject = new BeanObject();
    ...

    request.setAttribute("beandata", beanObject);


    RequestDispatcher dispatcher =
                  request.getRequestDispatcher("/WEB-INF/xxx.jsp");
    dispatcher.forward(request, response);
}
```

read and process request parameters

create and populate bean

store bean

forward request

IMPLEMENTING MVC IN JAVA

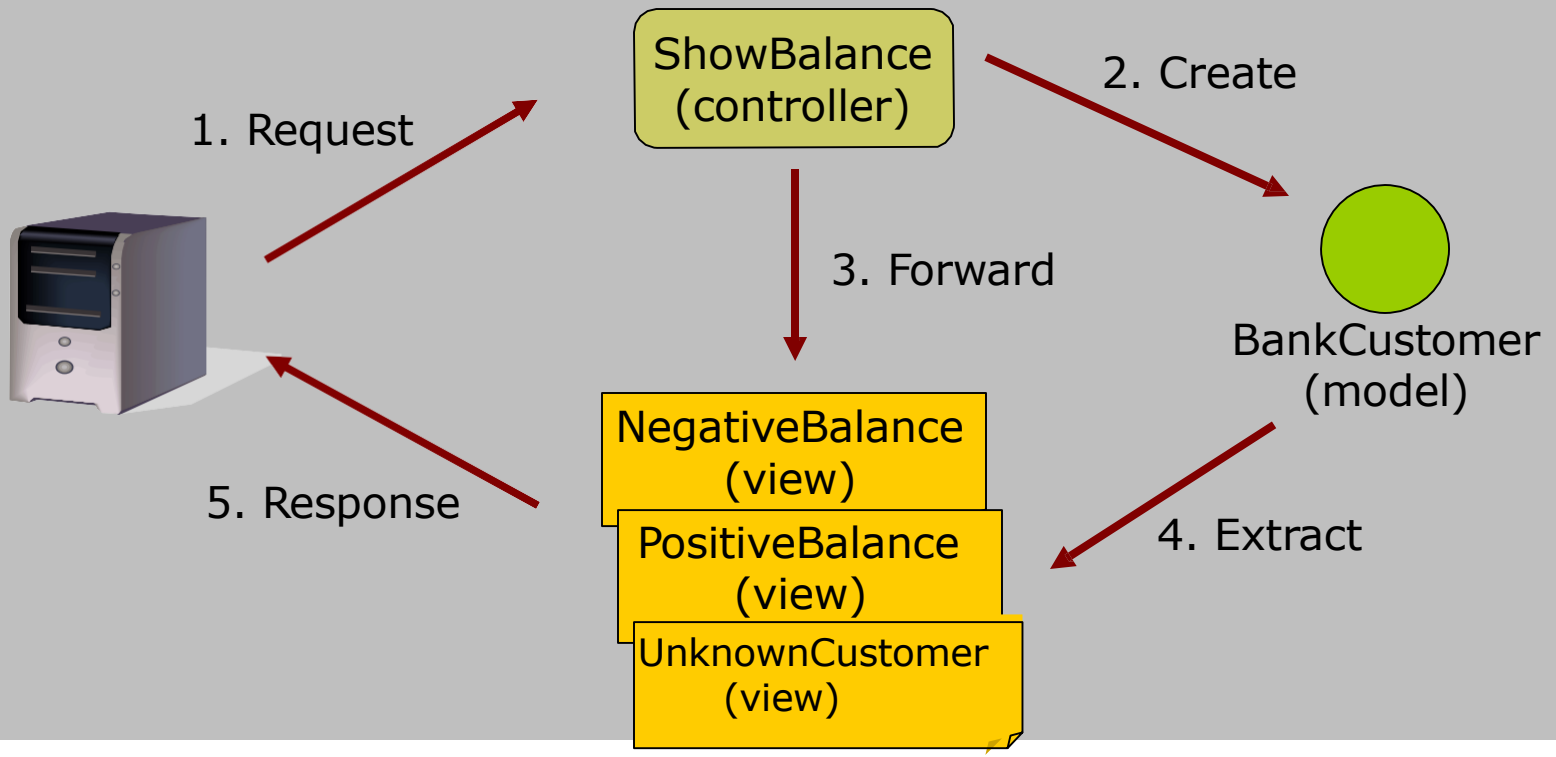
- Example: Session-based data sharing

```
BeanObject value = new BeanObject();
...
session.setAttribute("bean", value);
RequestDispatcher dispatcher =
        request.getRequestDispatcher("/WEB-INF/SomePage.jsp");
dispatcher.forward(request, response);
```

```
<jsp:useBean id="key" type="somePackage.BeanObject"
        scope="session" />
<jsp:getProperty name="key" property="someProperty" />
```

IMPLEMENTING MVC IN JAVA

- The bank application
  - Display customer's bank balance
  - Different Web pages depending on amount



IMPLEMENTING MVC IN JAVA

- Example: Calendar application
  - View list of events and enter new event
  - Model
    - `ScheduleDb` (maintains list of `ScheduleItem`s)
  - View
    - `MVCScheduleView.jsp`
    - `MVCScheduleEntryView.jsp`
  - Controller
    - `MVCViewSchedulerController`
    - `MVCScheduleEntryController`
    - `MVCSaveEntryContoller`

IMPLEMENTING MVC IN JAVA

- Example: Calendar application
  - View list of events and enter new event
  - Model
    - `ScheduleDb` (maintains list of `ScheduleItem`s)
  - View
    - `MVCScheduleView.jsp`
    - `MVCScheduleEntryView.jsp`
  - Controller
    - `MVCViewSchedulerController`
    - `MVCScheduleEntryController`
    - `MVCSaveEntryContoller`