CLOUD DEVELOPMENT

Web Application Architecture

DESIGN ISSUES FOR WEB APPS

Construction and testing	• How do we build a web application? • What technology should we choose?
Re-use	• Can we use standard components?
Scalability	• How will our web application cope with large numbers of requests?
Security	• How do we protect against attack, viruses, malicious data access, denial of service?
Different data views	• User types, individual accounts, data protection

DESIGN ISSUES FOR WEB APPS

Need to change the look-and-feel without changing the core/logic

Need to present data under different contexts (e.g., powerful desktop, web, mobile device).

Need to interact with/access data under different contexts (e.g., touch screen on a mobile device, keyboard on a computer)

Need to maintain multiple views of the same data (list, thumbnails, detailed, etc.)

3-Tier Architecture

A three-tier architecture allows separation of business logic and data access.

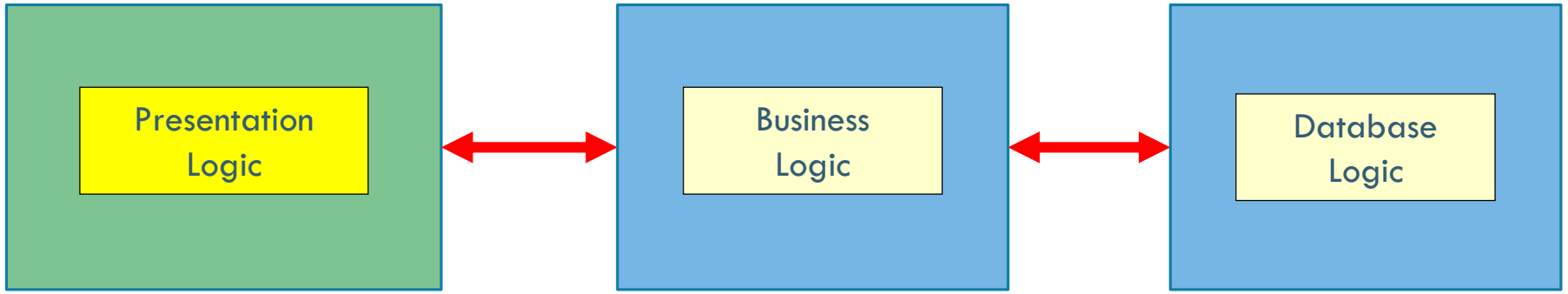
- In addition to the two-tier separation of GUI and business logic

- Tier 1: Presentation (Generation of HTML output)
- Tier 2: Application/ Business (Business Logic)
- Tier 3: Data Storage/ Management (DBMS)

3-TIER ARCHITECTURE

Each layer can potentially run on a different machine

Presentation, logic, data layers disconnected

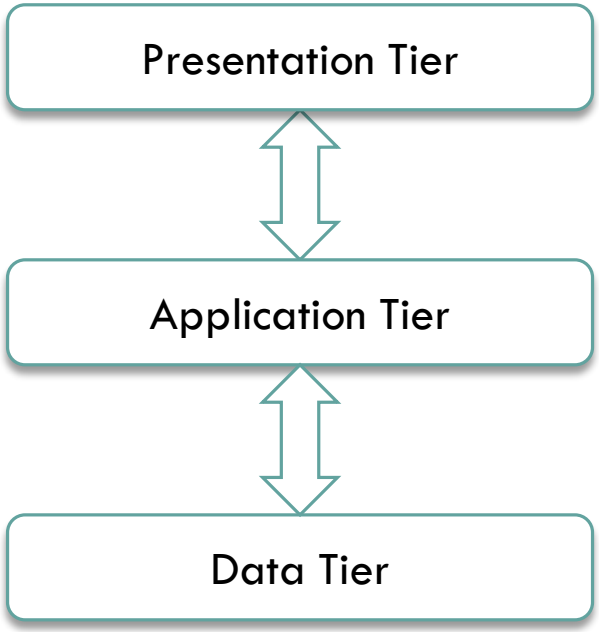


3-TIER ARCHITECTURE

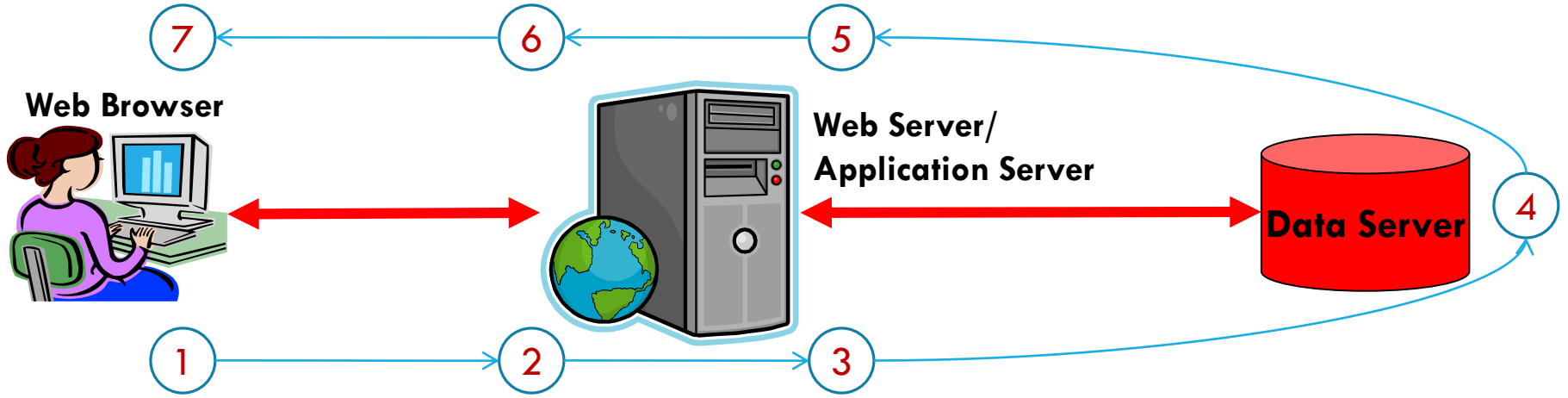
Information must flow in a sequential order between tiers

- The presentation tier may access the business tier, but must **never directly access the data tier**
- The middle Application (or Logic) coordinates all the information flow

Change in platform affects only the layer running on that particular platform



3-Tier Web Application – Example



Presentation Tier 1:

User Interface/ Interaction

Application Tier 2:

Business Logic –

functional algorithms, data access, etc.

Data Tier 3:

Information stored to and retrieved from Database

DESIGN PATTERNS

A general and reusable solution to a commonly occurring problem in the design of software

A template for how to solve a problem that has been used in many different situations

NOT a finished design

- The pattern must be adapted to the application
- Cannot simply translate into code

MODEL-VIEW-CONTROLLER (MVC)

An architectural pattern originating from Smalltalk (1979)

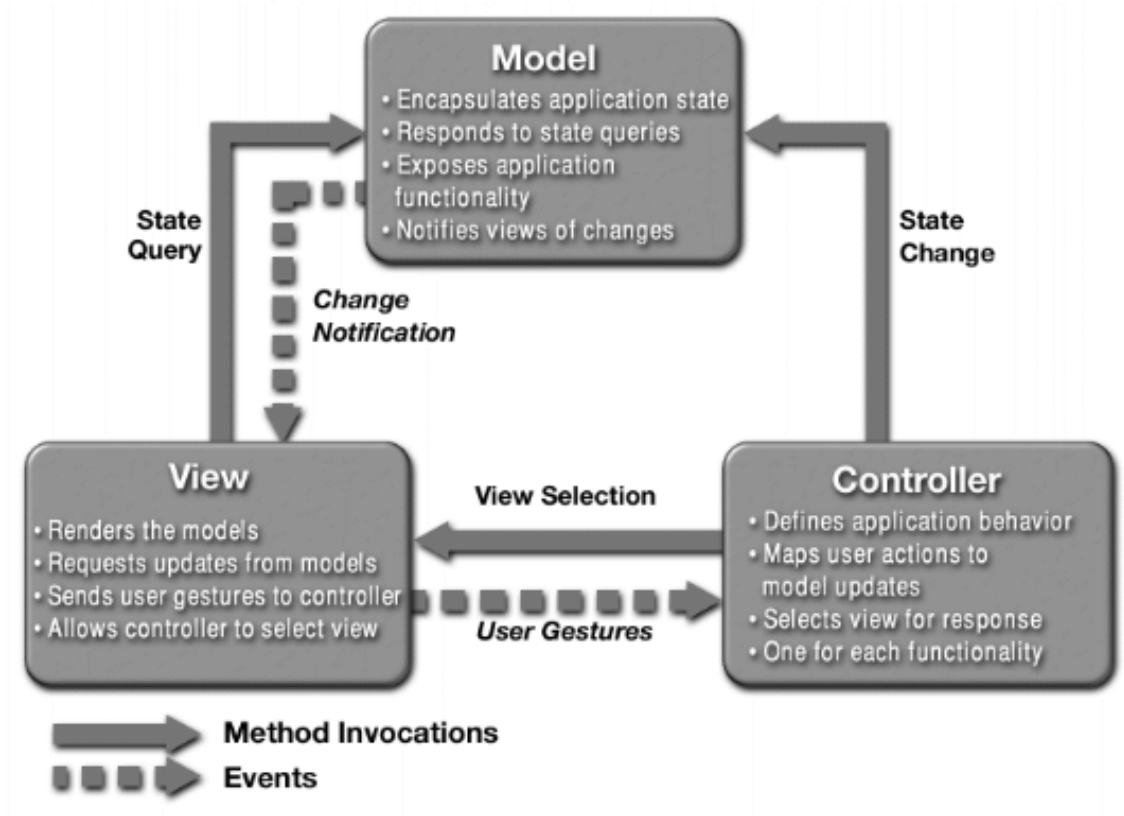
Separates data access and business logic from data presentation and user interaction so that:

- Changes to the user interface will not affect data handling
- Data can be reorganized without changing the user interface.
  - by introducing an intermediate component: the controller.

Breaks an application into the following three parts

- The controller
  - Handles requests, decides what logic to invoke – e.g. Business Logic which decide what page should apply
- The model
  - classes which model the data being displayed.
- The view
  - Output that the client sees.

MVC IN WEB APPLICATIONS



Model

- Contains domain-specific knowledge
- Records the state of the application
  - E.g., what items are in shopping cart
- Often linked to a database
- Independent of view
  - One model can link to different views

View

- Presents data to the user
- Allows user interaction
- Does no processing

Controller

- Defines how user interface reacts to user input (events)
- Receives messages from view (where events come from)
- Sends messages to model (tells what data to display)

Source: http://www.oracle.com/technetwork/articles/javase/index-142890.html#1

MVC IN WEB APPLICATIONS

Model

Represents data + database

View – User Interface

User interface (form)

Controller

Application code

- Display form
- Process form data

ADVANTAGES OF MVC (AND DESIGN PATTERNS IN GENERAL)

Modular

- Separation makes it easier to support different users and interface types in same application.
- Components are easily replaced

Reduced complexity in large applications.

- Maintainability
- Testability
- Reusability

EXAMPLE FLOW OF CONTROL IN MVC

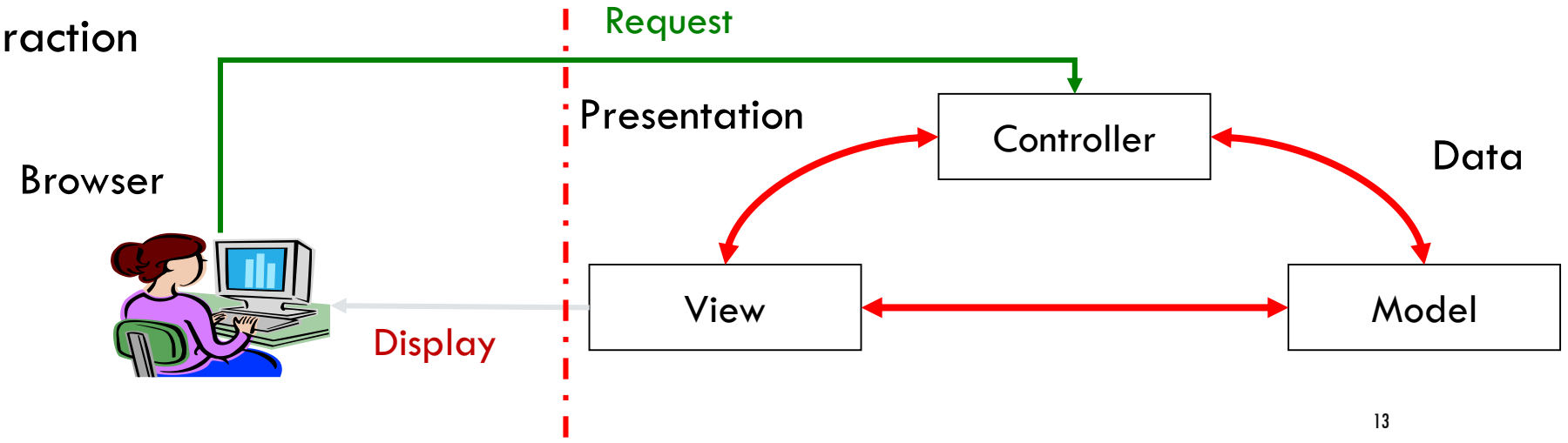
User interacts with the VIEW UI

CONTROLLER handles the user input (often a callback function attached to UI elements)

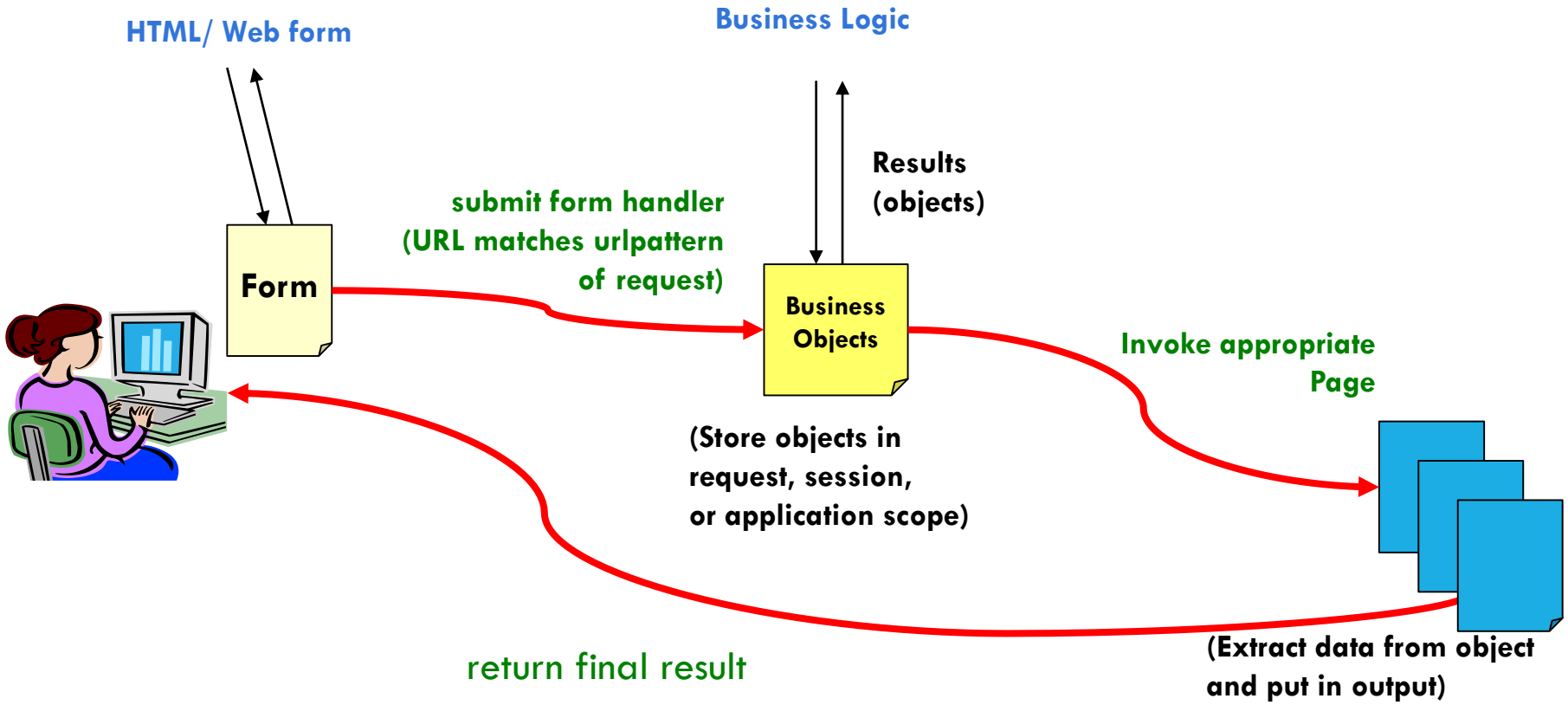
CONTROLLER updates the MODEL

VIEW uses MODEL to generate new UI

UI waits for user interaction



Flow of Control in MVC



3-TIER AND MVC COMPARED

Communication

- 3-tier: The presentation layer never communicates directly with the data layer-only through the logic layer (linear topology)
- MVC: All layers communicate directly

Usage

- 3-tier: Typically used in web applications where the client, application logic and data tiers run on physically separate platforms
- MVC: Historically used on applications that run on a single graphical workstation

REFERENCE

Java SE Application Design With MVC

- http://www.oracle.com/technetwork/articles/javase/index-142890.html#1

Software Architecture (Bijlsma, Heeren, Roubtsova, Stuurman)

- https://www.researchgate.net/publication/278847459_A_Bijlsma_BJ_Heeren_EE_Roubtsova_S_Stuurman_Software_Architecture