Pemrograman Berbasis Web

Pertemuan 12

Nori Wilantika
Lya Hulliyyatus Suadaa
Yeni Rimawati

Politeknik Statistika STIS
Prodi DIV Komputasi Statistik

12

Web Framework

Web Framework

- A software framework that is designed to support the development of web applications

- Provide a standard way to build and deploy web applications on the World Wide Web

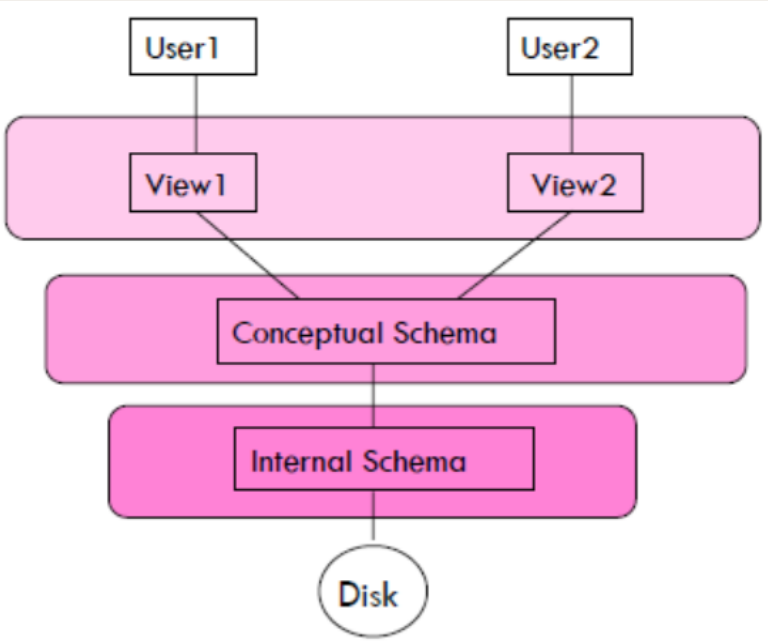
- Features:
  - Libraries for database access
  - Templating framework
  - Session management
  - Security, etc.

- Often promote code reuse

(Wikipedia)

Framework Architecture

Database Architecture with Views



What users see

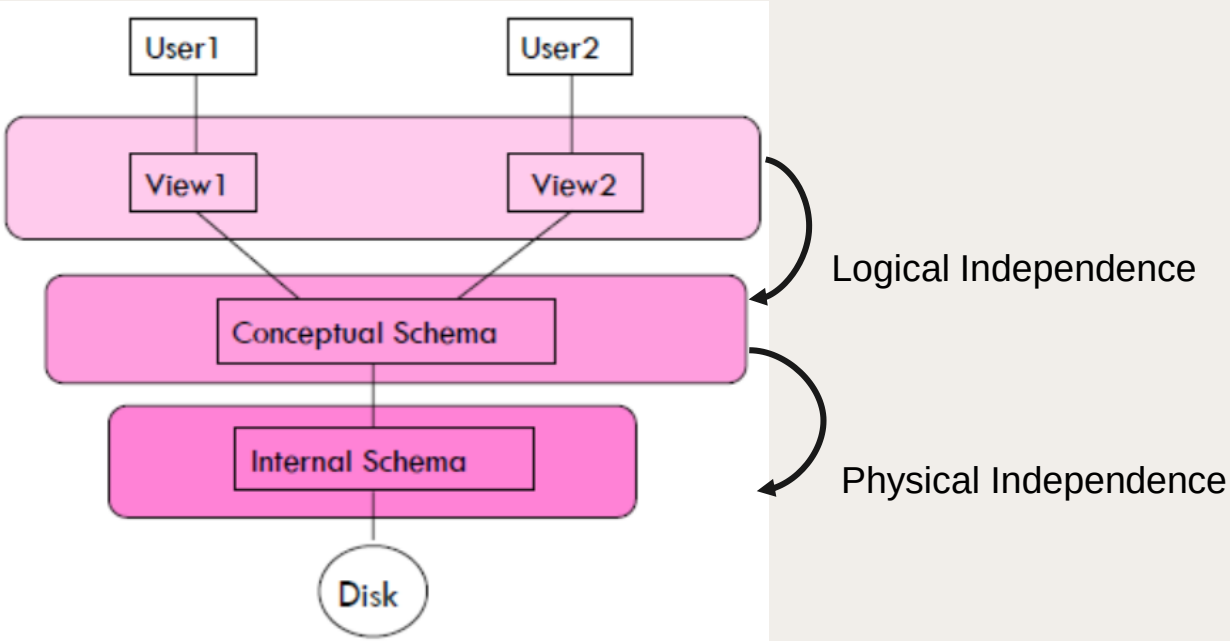
Tables and links

Files on disk

Each level is independent

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

Database Architecture with Views



Each level is independent

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

Database Architecture with Views

- Logical Independence: The ability to change the logical schema without changing the external schema or application programs
  - Can add new fields, new tables without changing views
  - Can change structure of tables without changing view

- Physical Independence: The ability to change the physical schema without changing the logical schema
  - Storage space can change
  - Type of some data can change for reasons of optimization

LESSON: Keep the VIEW (what the user sees) independent of the MODEL (domain knowledge)

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

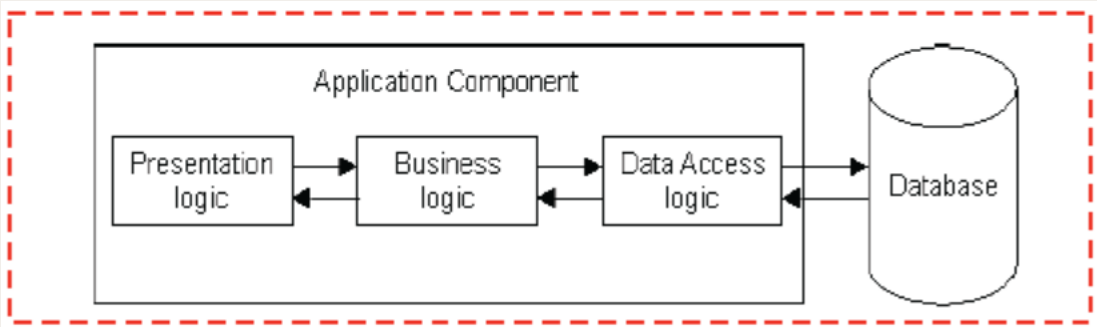
Significance of “Tiers”

- N-tier architectures have the same components
  - Presentation
  - Business/Logic
  - Data

- N-tier architectures try to separate the components into different tiers/layers
  - Tier: physical separation
  - Layer: logical separation

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

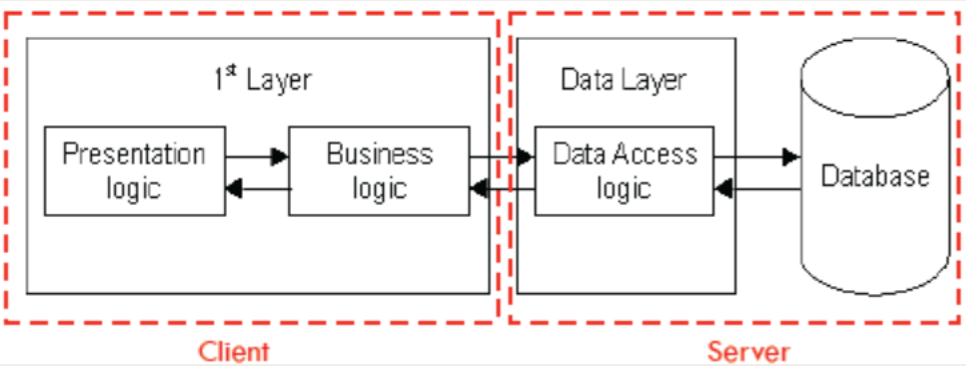
1-Tier Architecture



- All 3 layers are on the same machine
  - All code and processing kept on a single machine
- Presentation, Logic, Data layers are tightly connected
  - Scalability: Single processor means hard to increase volume of processing
  - Portability: Moving to a new machine may mean rewriting everything
  - Maintenance: Changing one layer requires changing other layers

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

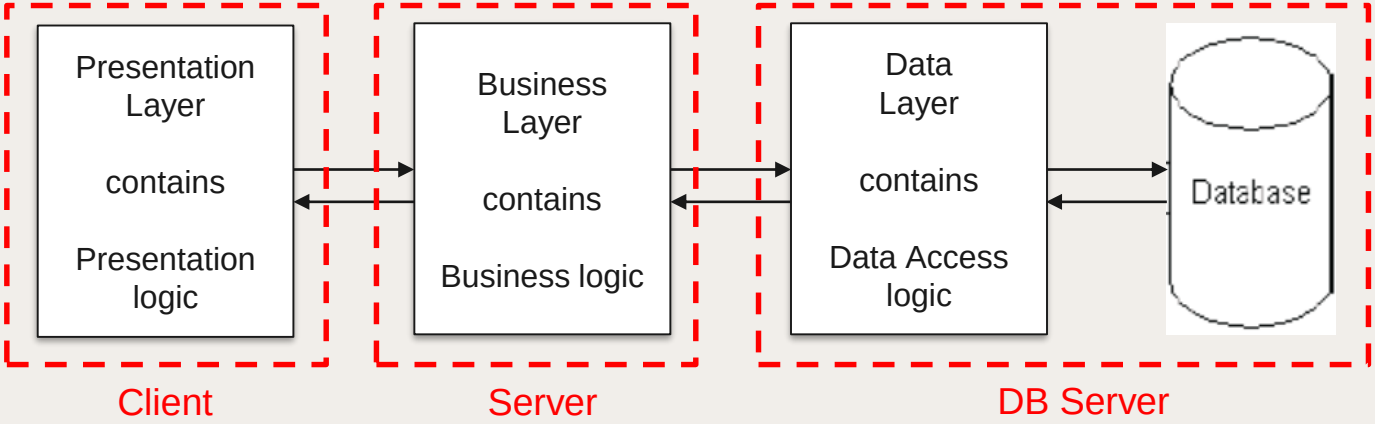
2-Tier Architecture



- Database runs on Server
  - Separated from client
  - Easy to switch to a different database
- Presentation and logic layers still tightly connected
  - Heavy load on server
  - Potential congestion on network
  - Presentation still tied to business logic

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

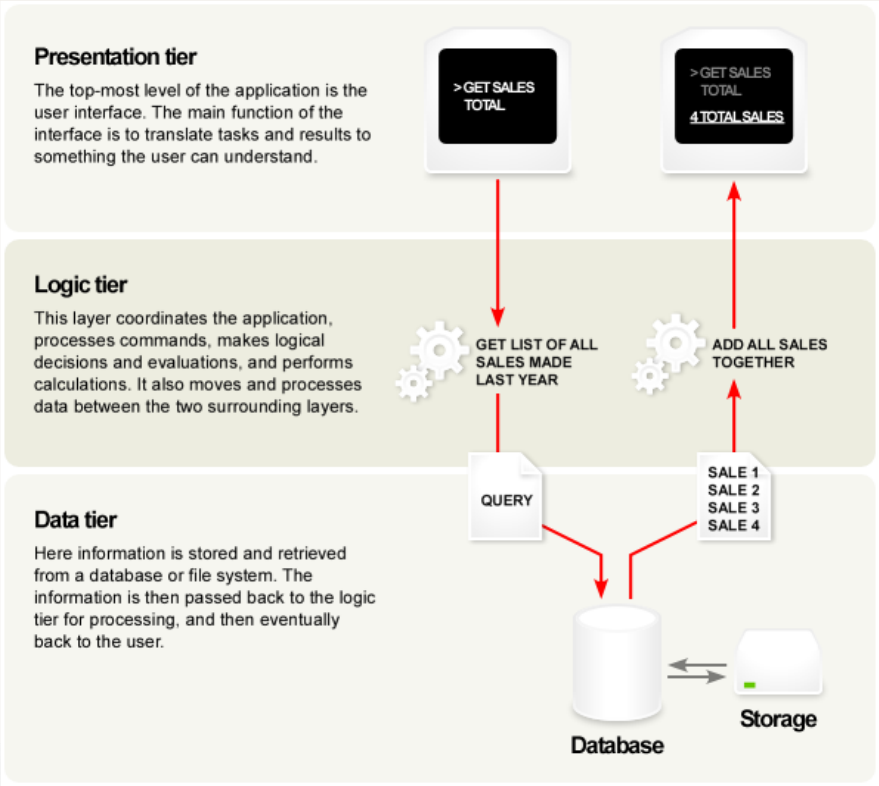
3-Tier Architecture



- Each layer can potentially run on a different machine
- Presentation, logic, data layers disconnected

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

A Typical 3-tier Architecture

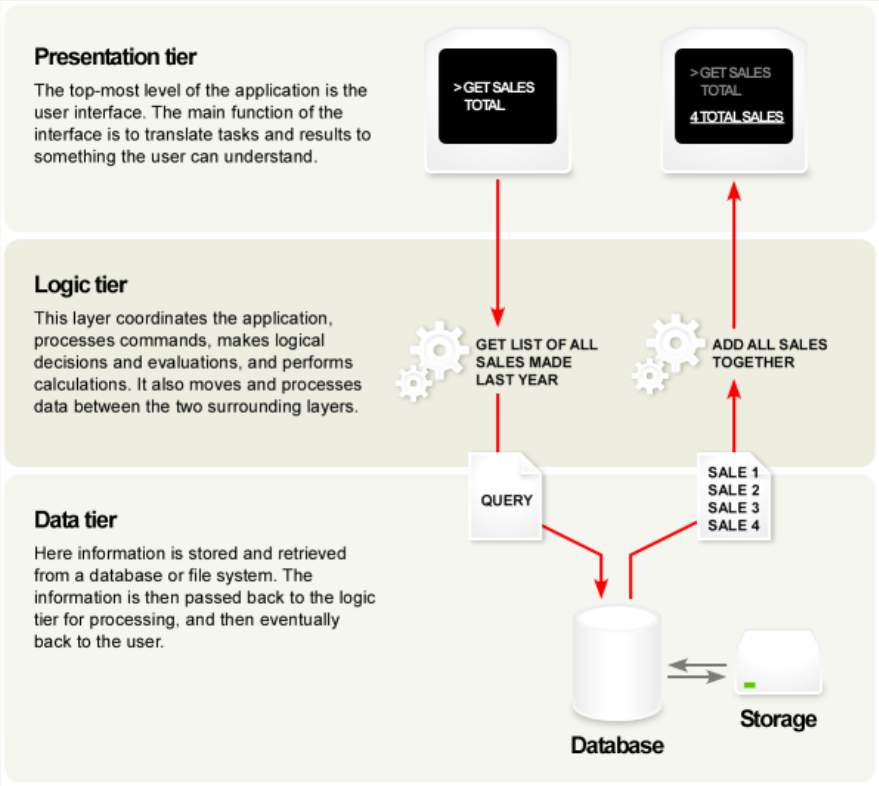


Architecture Principles

- Client-server architecture
- Each tier (Presentation, Logic, Data) should be independent and should not expose dependencies related to the implementation
- Unconnected tiers should not communicate
- Change in platform affects only the layer running on that particular platform

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

A Typical 3-tier Architecture

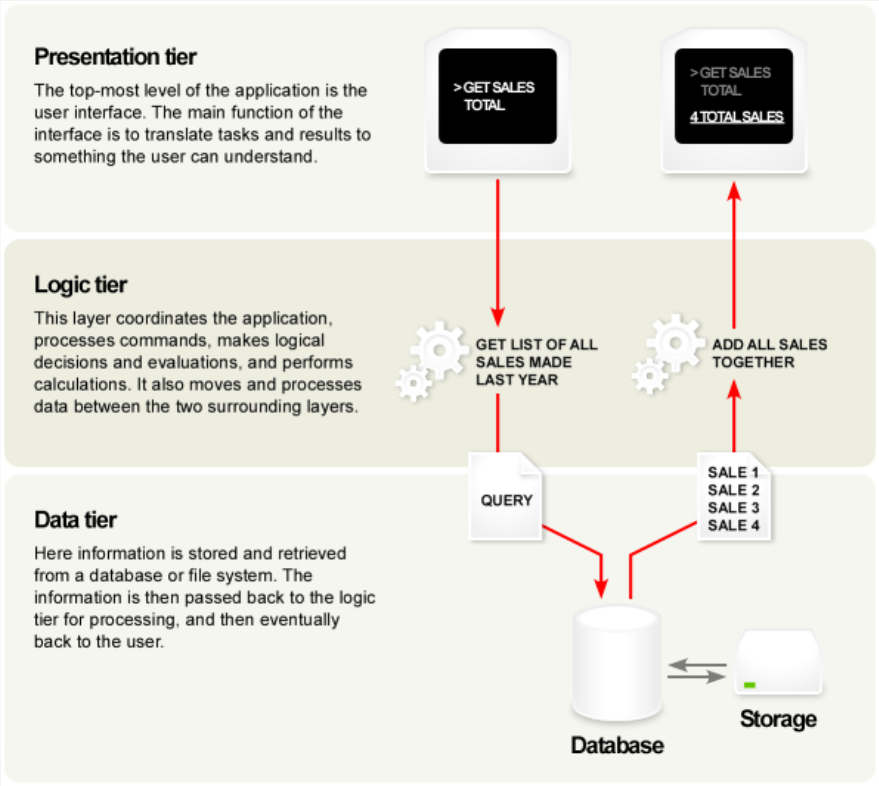


Presentation Tier

- Provides user interface
- Handles the interaction with the user
- Sometimes called the GUI or client view or front-end
- Should not contain business logic or data access code

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

A Typical 3-tier Architecture

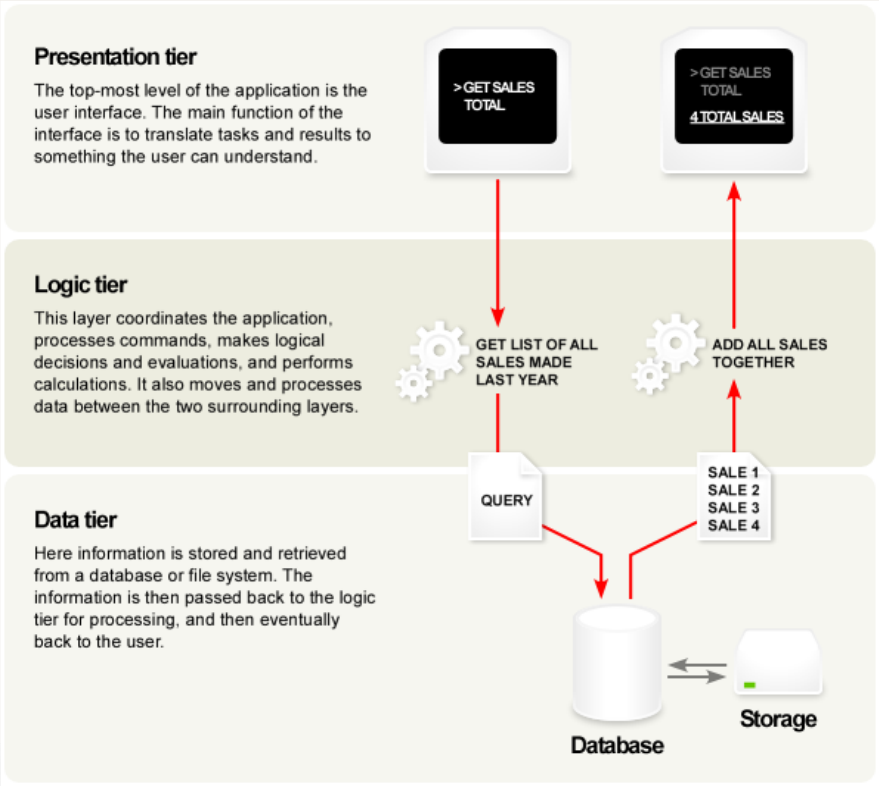


Logic Tier

- The set of rules for processing information
- Can accommodate many users
- Sometimes called middleware/back-end
- Should not contain presentation or data access code

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

A Typical 3-tier Architecture



Data Tier

- The physical storage layer for data persistence
- Manages access to DB or file system
- Sometimes called back-end
- Should not contain presentation or business logic code

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

The 3-Tier Architecture for Web Apps

- Presentation Layer
  - Static or dynamically generated content rendered by the browser (front-end)
- Logic Layer
  - A dynamic content processing and generation level application server, e.g., Java EE, ASP.NET, PHP, ColdFusion platform (middleware)
- Data Layer
  - A database, comprising both data sets and the database management system or RDBMS software that manages and provides access to the data (back-end)

3-Tier Architecture - Advantages

- Independence of Layers
  - Easier to maintain
  - Components are reusable
  - Faster development (division of work)
    - Web designer does presentation
    - Software engineer does logic
    - DB admin does data model

Design Problems & Decisions

- 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

Need for general and reusable solution: **Architectural & Design Patterns**

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

Design Pattern

- 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
- Types of design pattern: Creational, Structural, Behavioral
- NOT a finished design
  - the pattern must be adapted to the application
  - cannot simply translate into code

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

Architectural Pattern

- A **general** and **reusable solution** to a commonly occurring problem in the design of **overall structure of software**
- Architectural pattern is **higher level of abstraction** of software design,
- Example of architectural pattern: MVC, MVVM, ...

Architectural VS Design Pattern

Architectural Pattern

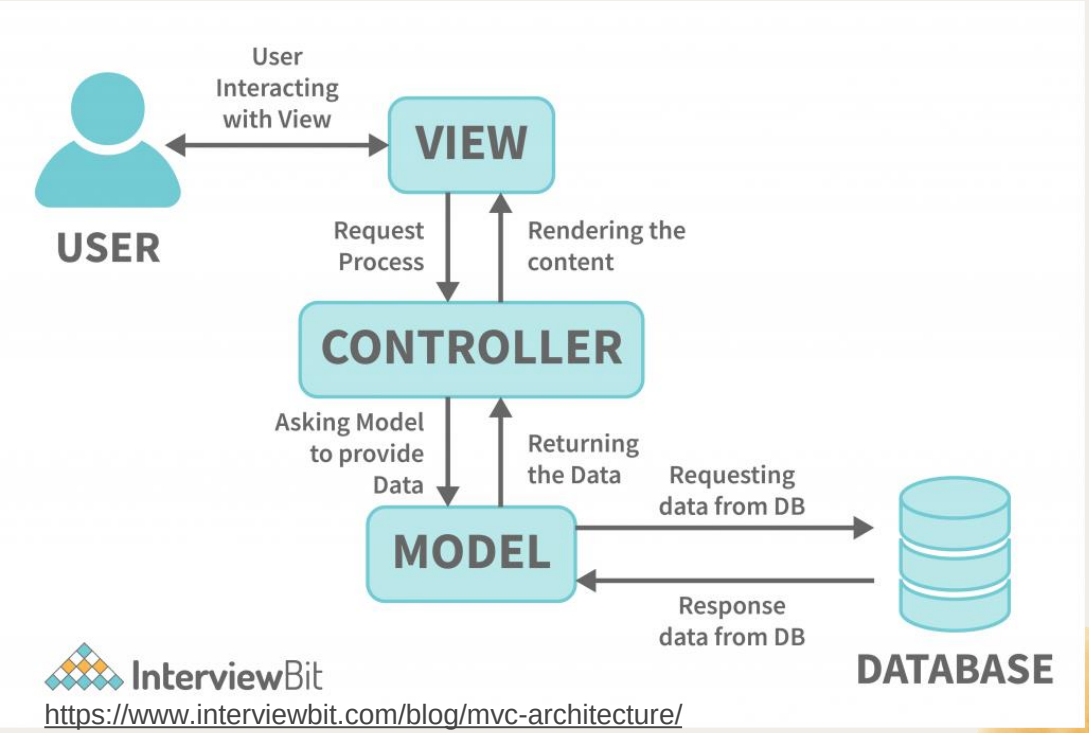
- Defined at higher level
  - Broader in scope than design pattern
- How component should behave & communicate in the system

Design Pattern

- Defined at granular level
- Detail solution relating to implementing certain components
  - Many design patterns can be implemented in one architectural pattern

MVC Pattern

- Pattern for graphical systems that promotes separation between model and view
- With this pattern the logic required for data maintenance (database, text file) is separated from how the data is viewed (graph, numerical) and how the data can be interacted with (GUI, command line, touch)

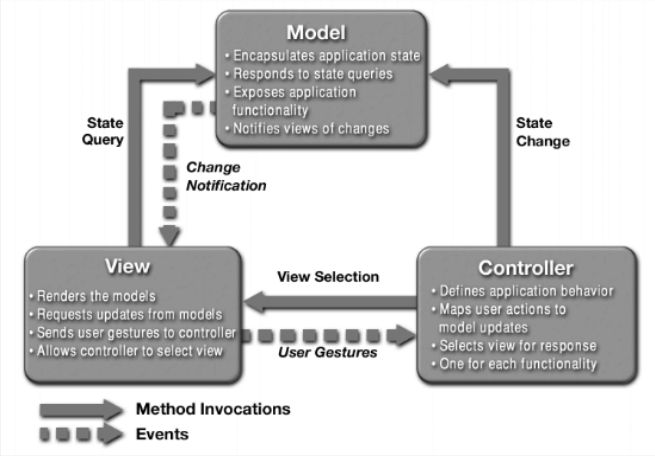


https://www.interviewbit.com/blog/mvc-architecture/

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

MVC Pattern

- Model
  - manages the behavior and data of the application domain
  - responds to requests for information about its state (usually from the view)
  - follows instructions to change state (usually from the controller)
- View
  - renders the model into a form suitable for interaction, typically a user interface (multiple views can exist for a single model for different purposes)
- Controller
  - receives user input and initiates a response by making calls on model objects
  - accepts input from the user and instructs the model and viewport to perform actions based on that input



(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

MVC for Web Application

- Model
  - database tables (persistent data)
  - session information (current system state data)
  - rules governing transactions
- View
  - HTML
  - CSS style sheets
  - server-side templates
- Controller
  - client-side scripting
  - http request processing
  - business logic/preprocessing

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

3-tier Architecture vs. MVC Architecture

- 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 (triangle topology)

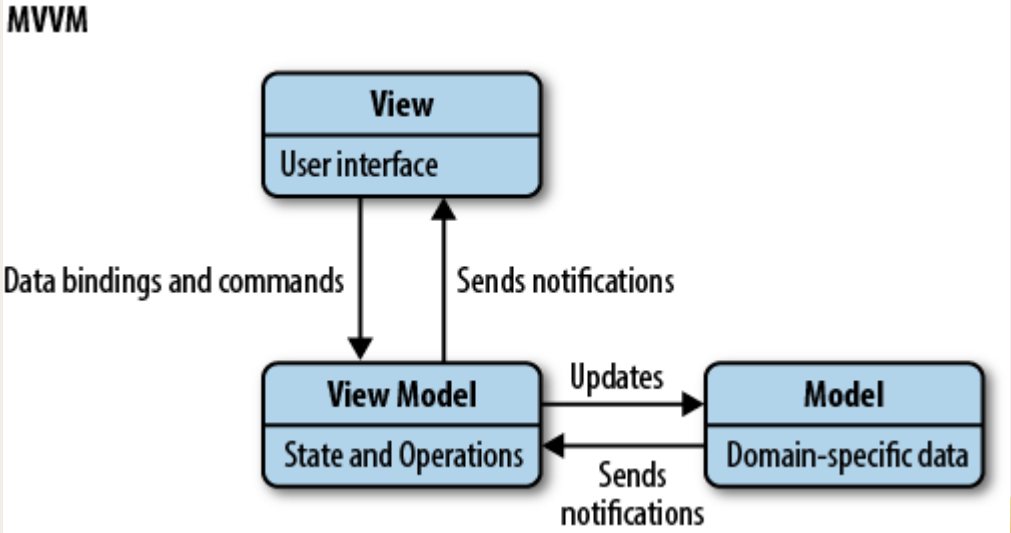
- Usage
  - 3-tier: Mainly used in web applications where the client, middleware and data tiers ran on physically separate platforms
  - MVC: Usually implemented in client-server architecture. The Controller and Model in MVC have a loose dependency on either a Service or Data layer/tier.

(Manos Papagelis @CSC309: Programming on the Web, Toronto University)

MVVM Pattern

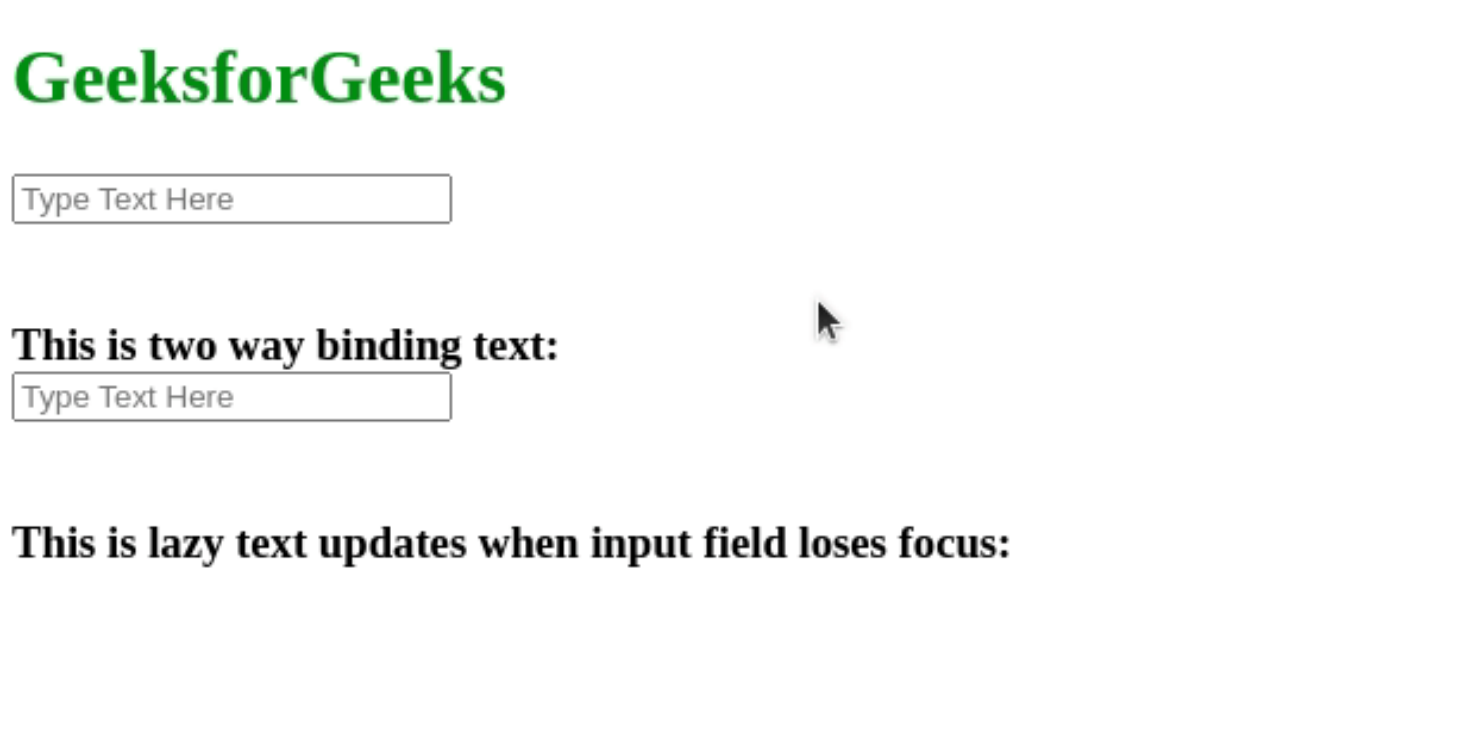
Model-View-ViewModel (MVVM) is a pattern that separates objects into three distinct groups:

- Models hold application data. They're usually structs or simple classes.
- Views display visual elements and controls on the screen. They're typically subclasses of UIView.
- View Models transform model information into values that can be displayed on a view. They're usually classes, so they can be passed around as references.



(https://www.raywenderlich.com/34-design-patterns-by-tutorials-mvvm)

Example



https://www.geeksforgeeks.org/how-a-view-model-works-in-vue-js/

Web Framework Application

Web Framework Application

- Front End Framework: focuses on the visual elements of a website or app that a user will interact with (the client side).
  - CSS framework
  - Javascript framework

- Back End Framework: focuses on the side of a website users can’t see (the server side).
  - PHP framework

CSS Framework:
Bootstrap

- Bootstrap was created at Twitter in mid-2010 by Mark Otto and Jacob Thornton.
- Prior to being an open-sourced framework, Bootstrap was known as Twitter Blueprint.
- It served as the style guide for internal tools development at the company for over a year before its public release, and continues to do so today.

https://getbootstrap.com/

CSS Framework:
Materialize

- Materialize CSS was created and designed by Google.

- Materialize CSS is also known as Material Design.
  - combines the classic principles of successful design along with innovation and technology.

- Google's goal is to develop a system of design that allows for a unified user experience across all their products on any platform.

https://materializecss.com/

CSS Framework:
Foundation

- Foundation was formerly maintained by ZURB from 2008 and used internally on all client projects and ZURB sites and apps.

- Foundation 2.0 is released to the public as an open source project!

- Developers are encouraged to participate in the project and make their own contributions to the platform.

https://get.foundation/

CSS Framework:
Bulma

- Bulma is an open-source, responsive CSS framework based on Flexbox. (without Javascript)

- Bulma has been highly adopted within the Laravel community, which has helped with its increasing popularity.

https://bulma.io/

Javascript Framework:
Vue.js

- Vue is a framework and ecosystem that covers most of the common features needed in front-end development.

- It was created by Evan You in 2014 as a personal side project.

- Today, Vue is actively maintained by a team of both full-time and volunteer members from all around the world, where Evan serves as the project lead.

https://vuejs.org/

Javascript Framework:
React.js

- React.js is a free and open-source front-end JavaScript library for building user interfaces based on UI components.

- It is maintained by Meta (formerly Facebook) and a community of individual developers and companies

- React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

https://reactjs.org/

Javascript Framework:
Ember.js

- Ember.js was developed by Yehuda Katz and initially released on in December 2011.
- Ember follows MVVM (Model-View-Viewmodel) model architecture.
- Many master companies like Microsoft, LinkedIn, Netflix, Apple Music, etc. are currently using it.

https://emberjs.com/

Javascript Framework:
Node.js

- Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser.

- Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

https://nodejs.org/

PHP Framework:
Codeigniter

- CodeIgniter is a powerful PHP framework with a very small footprint and clear documentation.
- Almost everything is set in CodeIgniter. Just connect to database!
- Encouraging MVC design pattern.

https://codeigniter.com/

PHP Framework:
Yii

- Yii is a feature-rich PHP framework with extensive documentation and helpful starting guide

- It comes with an extremely powerful class code generator known as Gii to ease out the process of object-oriented programming and rapid prototyping, which provides a web-based interface, helping the programmer generate the desired code interactively.

https://www.yiiframework.com/

PHP Framework:
Laravel

- Laravel is an extensive PHP framework known for its elegant syntax with plenty of features and built-in functions to customize complex apps with additional security and speed when compared to other frameworks.

- Its functions include user authentication, session management, and caching.

- It is easy to begin with Laravel due to the availability of extensive documentation and active community.

https://laravel.com/

PHP OOP

```php
class Fruit {
  public $name;
  public $color;

  function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  function get_name() {
    return $this->name."<br>";
  }
  function get_color() {
    return $this->color."<br>";
  }
  public function intro() {
    echo "The fruit is {$this->name}
    and the color is {$this->color}.<br>";
  }
}
```

```php
$apple = new Fruit("Apple", "red");
echo $apple->get_name();
echo $apple->get_color();
$apple->intro();
echo "<br><br>";
```

```php
// Berry is inherited from Fruit
class Berry extends Fruit {
  public function berryIntro() {
    echo "This is a berry fruit <br>";
  }
}
```

```php
$strawberry = new Berry("Strawberry", "red");
echo $strawberry->get_name();
echo $strawberry->get_color();
$strawberry->intro();
$strawberry->berryIntro();
```

Apple
red
The fruit is Apple and the color is red.

Strawberry
red
The fruit is Strawberry and the color is red.
This is a berry fruit

DESTRUCTORS

```php
<?php
class Fruit {
  public $name;
  public $color;

  function __construct($name, $color) {
    $this->name = $name;
    $this->color = $color;
  }
  function __destruct() {
    echo "The fruit is {$this->name} and the color is {$this->color}.";
  }
}

$apple = new Fruit("Apple", "red");
?>
```

The fruit is Apple and the color is red.

VISIBILITY

- **public** accessible everywhere
- **private** accessible only within the same class
- **protected** accessible only within the class itself and by inheriting and parent classes

```php
<?php
class Fruit {
  public $name;
  protected $color;
  private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '300'; // ERROR
?>
```

- For properties, a visibility declaration is required
- For methods, a visibility declaration is optional; by default, methods are public
- Accessing a private or protected property / method outside its visibility is a **fatal error**

STATIC

- Class properties or methods can be declared **static**
  Static properties or methods can be called directly - without creating an instance of the class first.
- Static class properties **cannot** be accessed via an instantiated class object, but static class methods can
- Static class properties and methods are accessed (via the class) using the scope resolution operator ::
- Static class method have no access to **$this**

```
class Employee {
  static $totalNumber = 0;
  public $name;

  function __construct($name) {
    $this->name = $name;
    Employee::$totalNumber++;
} }
$e1 = new Employee("Ada");
$e2 = new Employee("Ben");
echo Employee::$totalNumber  # prints 2
```

CONSTANT

- Class constants can be useful if you need to define some constant data within a class.
- Constants cannot be changed once it is declared.

```
vis const identifier = value;
```

- Classes can have their own constants and constants can be declared to be public, private or protected. By default, class constants are public
- Class constants are case-sensitive. However, it is recommended to name the constants in all uppercase letters.

```
class MyClass {
  const SIZE = 10;
}
echo MyClass::SIZE;   # prints 10
$o = new MyClass();
echo $o::SIZE;        # prints 10
```

INTROSPECTION FUNCTIONS

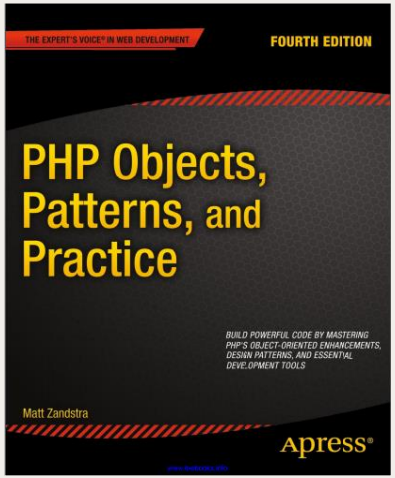
- There are functions for inspecting objects and classes:

class_exists(string class)	returns TRUE if a class class exists
get_class(object obj)	returns the name of the class to which an object obj belongs
is_a(object obj, string class)	returns TRUE if obj is an instance of class named class
method_exists(object obj,string method)	returns TRUE if obj has a method named method
property_exists(object obj,string property)	returns TRUE if obj has a property named property
get_object_vars(object)	returns an array with the accessible non-static properties of object mapped to their values
get_class_methods(class)	returns an array of method names defined for class

REFERENSI

Materi disadur dari:

1. PHP Objects, Patterns, and Practice, 4th Edition (Matt Zandstra)



2. Manos Papagelis @CSC309: Programming on the Web, Toronto University

Thanks!

Do you have any questions?

wilantika@stis.ac.id
lya@stis.ac.id
yeni.rima@stis.ac.id