
MONASH University
MONASH
INFORMATION
TECHNOLOGY
FIT5192 Lecture 2:
Enterprise Architecture Concepts and
Java EE Application Architecture
GROUP
OF EIGHT
AUSTRALIA

MONASH University

This Lecture

- Look at the different architecture approaches towards enterprise applications.
- Learn the breakdown of Java EE platform and the features it offers developers in building rich web enterprise applications.
- Introduce some important terms and concepts.
- Find out how to approach Java EE development and the process of assembly & deployment.

GROUP OF EIGHT AUSTRALIA

Enterprise Software

- Software that is built to support business activities in a stable and reliable environment.
  - The more complex the requirements, often the more demanding the software becomes.
  - Usually the software then becomes supported by multiple systems using different programming backends.
  - Often require access to data from different sources and incorporate such data to complete the operations required

MONASH University

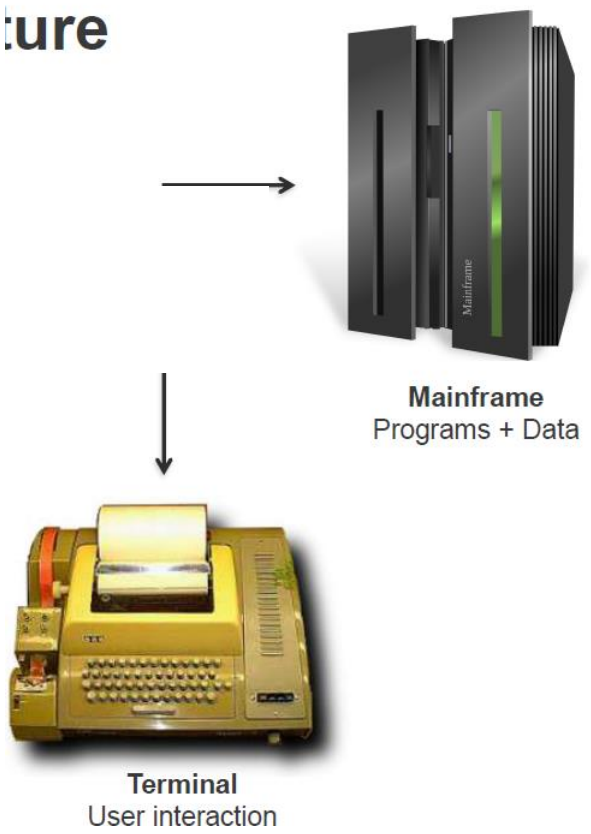
Tier-based Architecture Approaches

Multi-tier Architecture

- Often referred to as n-tier architecture.
- A client-server architecture
- It aims to provide a model that governs developers to build systems that are flexible, maintainable, reusable and scalable.
- It segregates a system into “tiers” based on the responsibilities of the components.
- Components on tiers are often located in different machines, even though it is not necessary.

Single-Tier Architecture

- Mainframe houses the central processing unit and memory.
- Every program and data was stored in a singlez “powerful” machine.
- Users could access this centralized computer by means of terminals.



Pros & Cons of Single-Tier Architecture

Pros:

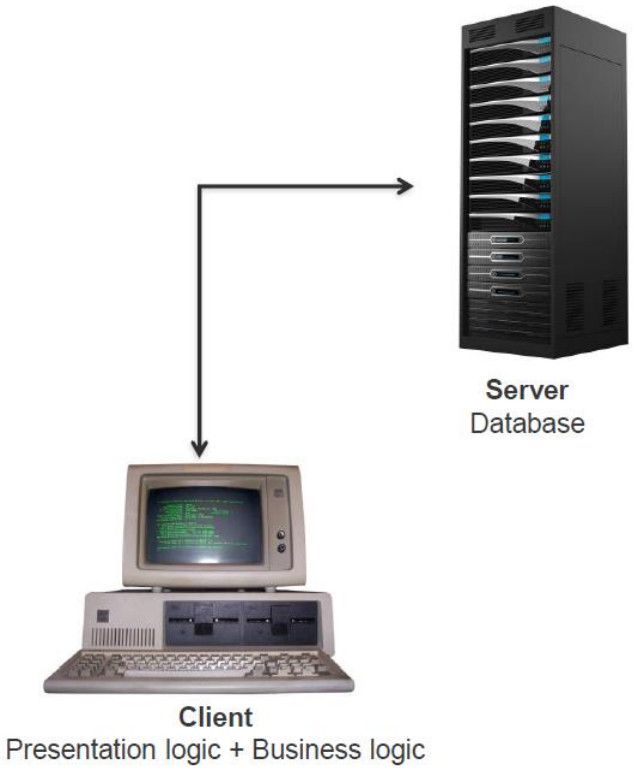
- It is simpler to develop as applications don't need to handle any network protocols.
- It has less overheads. Problems such as synchronization of faraway data, network failure, bogus data from a server is not a concern.
- It has faster response time as requests and response don't need to transfer across network.

Cons:

- It does not support remote/distributed access of data resources.
- Monolithic manner of the code causes higher maintenance.
- It does not support load balancing.
- It does not support heterogeneous environments.

Two-Tier Architecture

- In the 1980s, most enterprise applications followed a two-tier architecture approach (client / server).
- Business and presentation logic is mixed together – the client application (fat client) contains the presentation logic, the application navigation, the business logic and the database access.



Pros & Cons of Two-Tier Architecture

Pros:

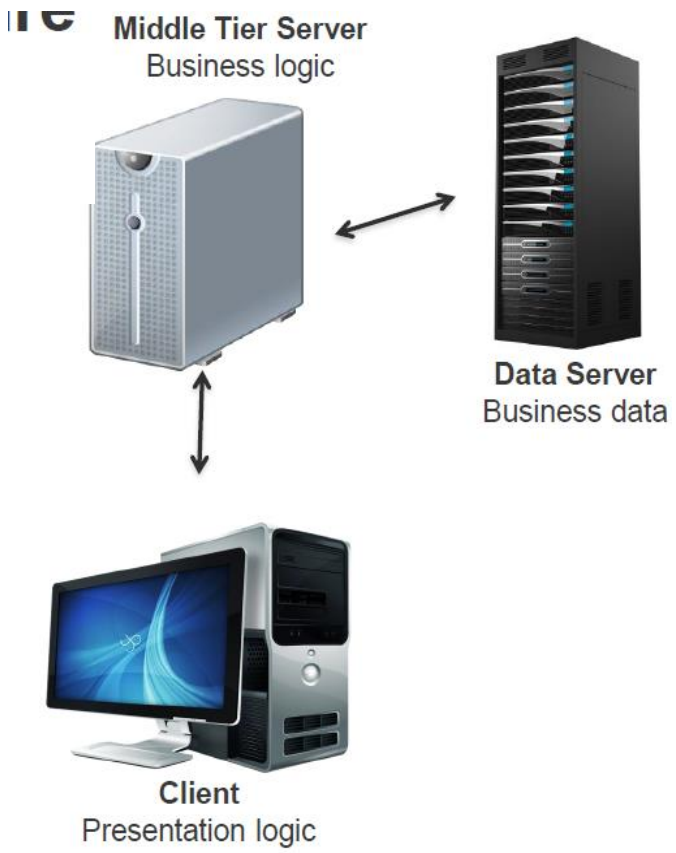
- It is simpler to develop as business logic is constrained to a single, non-distributed system.
- Data can be shared across multiple systems.

Cons:

- It is expensive to change. When business rules are modified, the whole client applications needs to be changed tested and redistributed. Even when the user interface remains unchanged.
- It is difficult to control software version and re-distribute new version as client beholds most of the business logics.
- It has limited scalability as the number of clients that can access the database is limited.
- It has minimal logic sharing since the application logic is coupled with the client.
- It does not support heterogeneous environments.

Three-Tier Architecture

- Separates the presentation logic from the business logic.
  - Tier 1: the client contains the presentation logic (thin client).
  - Tier 2: the middle tier provides business processes logic and data access.
  - Tier 3: the data server provides the business data.



Pros & Cons of Three-Tier Architecture

Pros:

- It is easier to change or replace any tier without influencing the others tiers.
- It improves code reusability as similar business logic can be reused in many clients or applications.
- It has enhanced scalability/load balancing because of the separation of the application and database.
- Adequate security policies can be enforced within the server tiers without hindering the clients. In addition, access to data can now be granted on a service-by-service basis.
- It provides better fault recovery as redundant servers can be used to recover the system from network or server failures.

Cons:

- It is more difficult to develop as considerations need to be made for multi-threading, security, distribution, deployment and administration support.

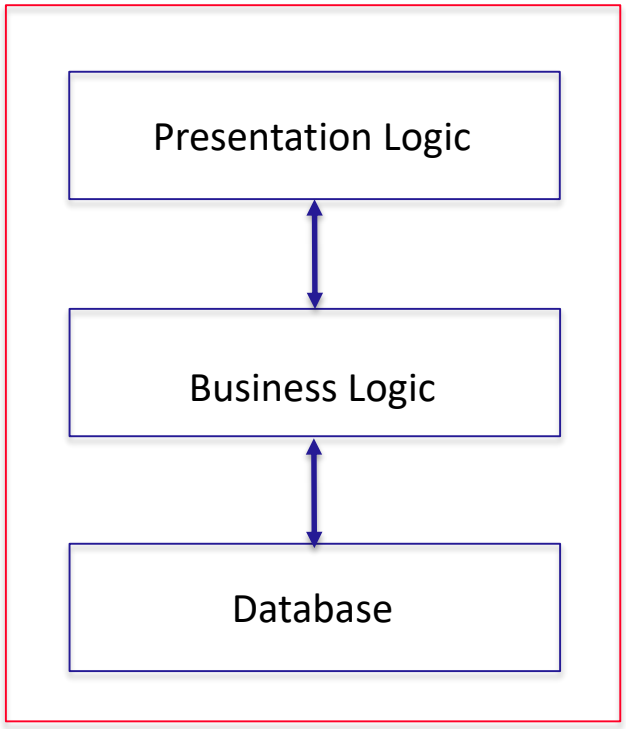
Example – Coffee Shop(1)

- Suppose I own a coffee shop in Caulfield and I do not have any branches.
- I give credit to my regular customers and I maintain their accounts in a software on my computer.
- Now what should my software do when a customer comes in with his member card issued by me?

Example – Coffee Shop(2)

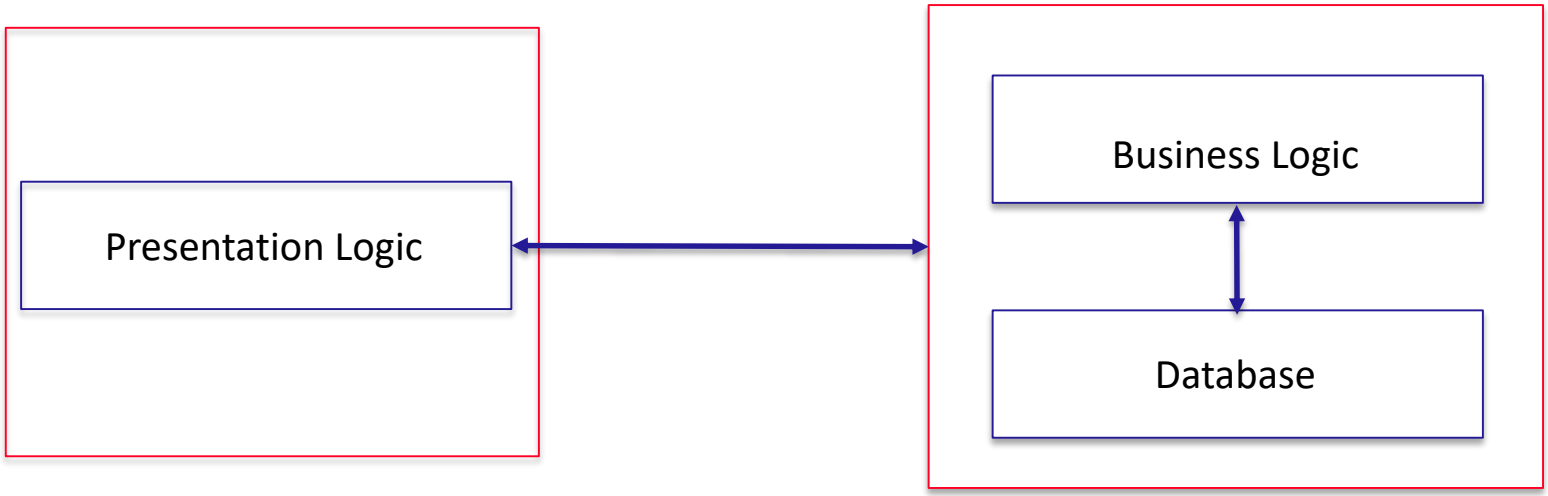
- The software should cater the at least the following requirements:
  - Presents a GUI to the employee to enter the customer ID and the order details
  - Some program with sufficient logic to read the ID and display the customer details such as his maximum amount of credit, the current balance and whether he has any overdue payment.
  - A database that maintains customers' details.
- Because I own only one coffee shop and I have a tight budget, all the above three requirements are implemented as one program and loaded into a single computer.

Example – Coffee Shop(3)



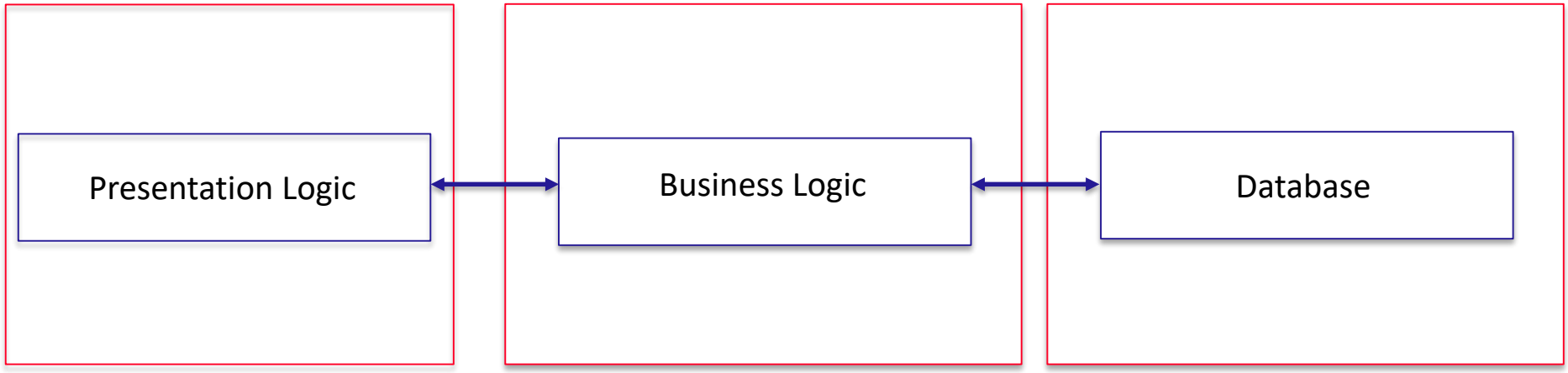
1-Tier Architecture

Example – Coffee Shop(4)



2-Tier Architecture
(Possible to have multiple clients)

Example – Coffee Shop(5)



3-Tier Architecture
(Possible to have multiple clients)

MONASH
University

Java EE Platform

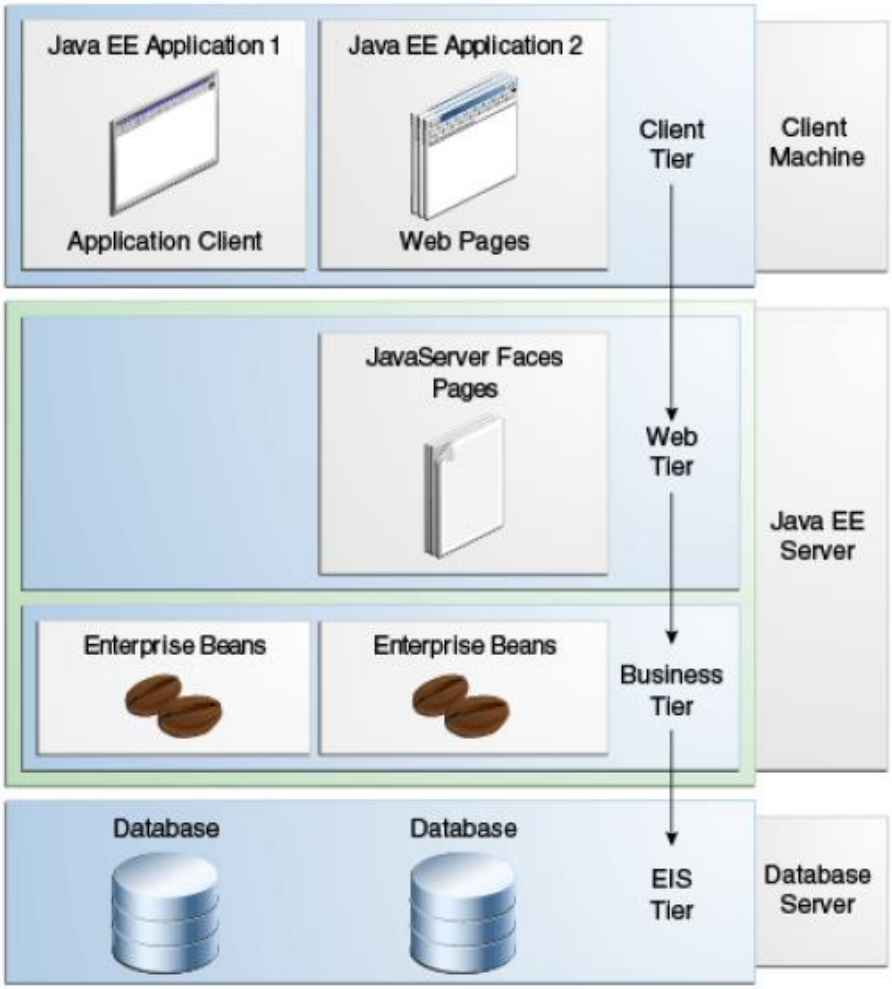
Java EE Architecture

- The Java EE platform uses a distributed multi-tiered architecture for enterprise applications.
- Application logic is divided into components according to function.
- These components are then installed on different machines depending on the tier to which they belong to.
  - Client-tier components run on the client machine.
  - Web-tier components run on the Web server/Java EE server.
  - Business-tier components run on the Java EE server.
  - Enterprise information system (EIS)-tier software runs on the EIS server.

Component-based Software

- Java EE applications consist of components.
- A Java EE component:
  - is a self-contained functional software unit that is assembled into a Java EE application
  - contains its related classes and files
  - can communicate with other components
- The Java EE specification defines the following components:
  - Application clients and applets are components on the client
  - Java Servlet, JavaServer Faces (JSF) and JavaServer Pages (JSP) technology components are web components that run on the web server/Java EE server
  - Enterprise Java Bean (EJB) components are business components that run on Java EE server

Multi-tiered Applications



Source: https://docs.oracle.com/javaee/7/tutorial/overview003.htm#BNAAY

Java EE Clients – Web Clients

- A Java EE client is usually either a web client or an application client
- Web client usually consists of two parts:
  - Dynamic web pages containing various types of markup language (e.g. HTML, XML, and etc) that are generated by web components running in the web tier
  - A web browser, which renders the pages received from the server
- A web client does not usually access database directly, execute complex business rules, or connect to legacy applications.
- The heavyweight operations are off-loaded to EJBs running on the Java EE server.

Java EE Clients – Application Clients

- An application client runs on a client machine.
- It usually has a graphical user interface (GUI) created using Swing or Abstract Window Toolkit (AWT) API, but a command line interface is also possible.
- It allows users to handle tasks that require a richer user interface.
- Application client directly access enterprise beans running in the business tier, but it is capable of opening an HTTP connection to communicate with a **Servlet** running in the web tier if needed

Applets

- An Applet is a small client program written in the Java programming language.
- It is typically embedded in a web page received from the web tier.
- It runs in the JVM installed in the web browser. The Java Plug-in and a security policy file is likely required.
- Due to these requirements, it has lost its popularity in recent years

Web Components

- Java EE web components are either servlets or web pages created using JavaServer Faces (JSF) technology and/or JSP technology.
- Servlets are Java programming language classes that dynamically process requests and constructs responses
- JSP pages are text-based documents that execute as servlets but allow a more natural approach to creating static content.
- JavaServer Faces technology builds on servlets and JSP technology and provides a user interface component framework for web applications.
- Applets, static HTML pages and other utility classes can be/are bundled with web components, but they are not considered as web component by Java EE specification.

JavaBeans

- Java classes that conform to the JavaBean standard:
  - has a default constructor.
  - allow access to properties only via accessors and mutators.
  - the name of the accessors and mutators must follow the JavaBeans guidelines.
- They are often used to manage the data flow between:
  - Java EE clients and components running on the Java EE server.
  - Server components and a database.
- They are considered Java EE components by the Java EE specification.

Business Components

- They contain the logic that solve the problems or meet the needs of a particular business domain.
- An enterprise bean is a server-side component that encapsulate such logic.
- They can only run on a Java EE server
- There are two types of EJB:
  - Session bean: performs a task for a client; it may implement a web service.
  - Message-driven bean: acts as a listener for a particular messaging type, such as the Java Message Service API

Enterprise Information System Tier (EIS)

- The enterprise information system tier handles EIS software and includes enterprise infrastructure systems, such as:
  - Enterprise Resource Planning (ERP)
  - Mainframe transaction processing
  - Database systems
  - Other legacy information systems

Java EE Containers

- A runtime environment that Java EE components run in.
- They provides the interface between a component and the low-level, platform-specific functionalities that support the component.
- Example of services provided by Java EE containers include:
  - EJB & servlet lifecycles
  - State management
  - Multithreading
  - Resource pooling and etc.
- Before a component can be executed, it must be assembled into a Java EE module and deployed into its container.

Assembly

- The assembly process involves specifying container settings for each component in the Java EE application and for the Java EE application itself.
- These settings customize the underlying support provided by the Java EE server.
- Example of configurable services include:
  - The Java EE security model lets developers configure a web component or enterprise bean so that the system resources are accessed only by authorized users.
  - The Java EE transaction model allow developers to specify relationships among methods that make up a single transaction so that all methods in one transaction are treated as a single unit.
  - JNDI lookup services provide a unified interface to multiple naming and directory services in the enterprise so that application components can access these services.
- As a result of the configurable services, the same component might behave differently in different environment.

Container Types

- The deployment process installs Java EE application components in the Java EE containers.
- There are 4 types of containers:

Container Type	Manage	Environment
EJB container	Enterprise Java Beans	Java EE server
Web container	Web pages, servlets	Java EE server
Application client container	Application client	Client machine
Applet container	Applet	Web browser + Java Plug-in on client machine

jar Files

- A type of archive files with the .jar extension (These are essentially zip files)
- During assembly, components are packed into:
  - JAR (Java Archive) file – a standard JAR file for packaging Java classes with .jar
  - It contains the Java modules and necessary deployment descriptors.
  - Deployment descriptions can be changed without the need to modify the source code of the application. They are processed at runtime.

war Files

- A type of archive files with the .war extension (These are essentially zip files)
- During assembly, components are packed into:
  - WAR (Web Archive) file – a standard WAR file for packaging a Java EE web components with a .war
  - It contains the Java web modules and necessary deployment descriptors.
  - Deployment descriptions can be changed without the need to modify the source code of the application. They are processed at runtime.

ear Files

- A type of archive files with the .ear extension (These are essentially zip files)
- During assembly, components are packed into:
  - EAR (Enterprise Archive) file – a standard JAR file for packaging a Java EE application with the extension .ear
  - It contains the Java EE modules and necessary deployment descriptors.
  - Deployment descriptions can be changed without the need to modify the source code of the application. They are processed at runtime.

Summary

- Today, we have looked at:
  - the evolution of enterprise software architecture and what the current practice is today.
  - the structure of Java EE platform and how it can help us build enterprise application specifically for the web.
- Next lecture: we will start working on enterprise application

MONASH University

See you in the Studio !

Readings

- ***Recommended background reading***:

Chapter 2: Context and Dependency Injection in Beginning Java EE 7, Antonio Goncalves