Enterprise and Cloud Computing

Lecture 1: Introductory Lecture

Geraldine Gray
geraldine.gray@itb.ie
Ph. 8851083

Moodle enrolment key: cloud

Module Overview

The focus of this module is to provide you with an understanding of

developing (web based) applications

for the enterprise.

Module aims:

Understand the

theory and practice

associated with implementing

large-scale distributed information systems.

The module builds on

Enterprise Computing

Rich Web Apps

Web development (server)

Web development (client)

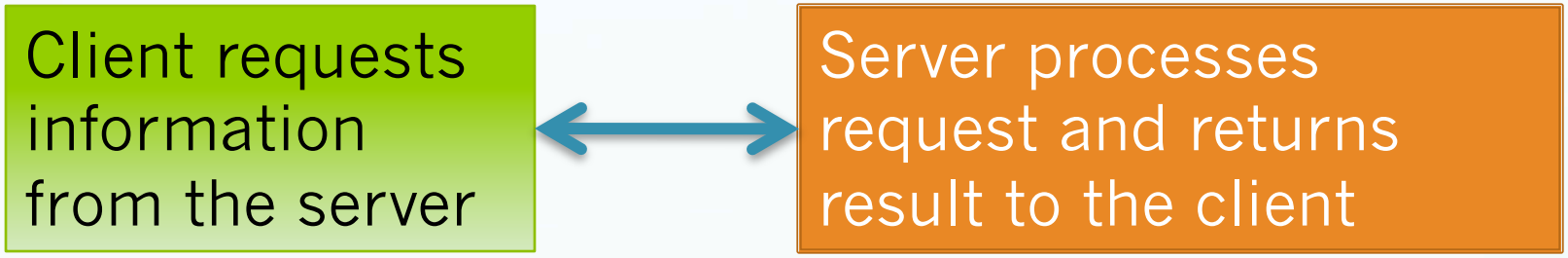
Objective for Lecture 1

- After this lecture, you should know the following:
  - What is a *n*-tier architecture
  - What Java EE means
  - What is an EJB, and the types of EJB's
  - What is an EJB container, and what is it responsible for.

  - The module learning outcomes and assessment schedule.

Large-scale distributed information systems

Simple, data based applications can run in a two tier, client-server architecture:

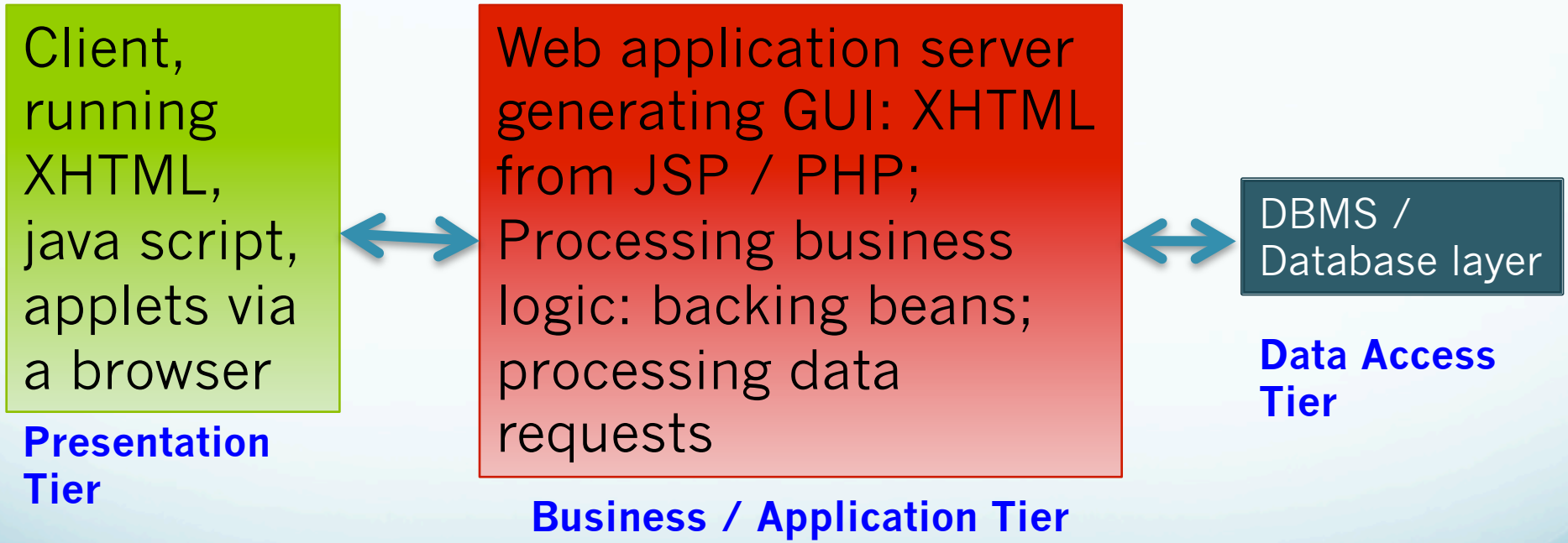


- The client makes a request for information to the server
- The server processes the request.

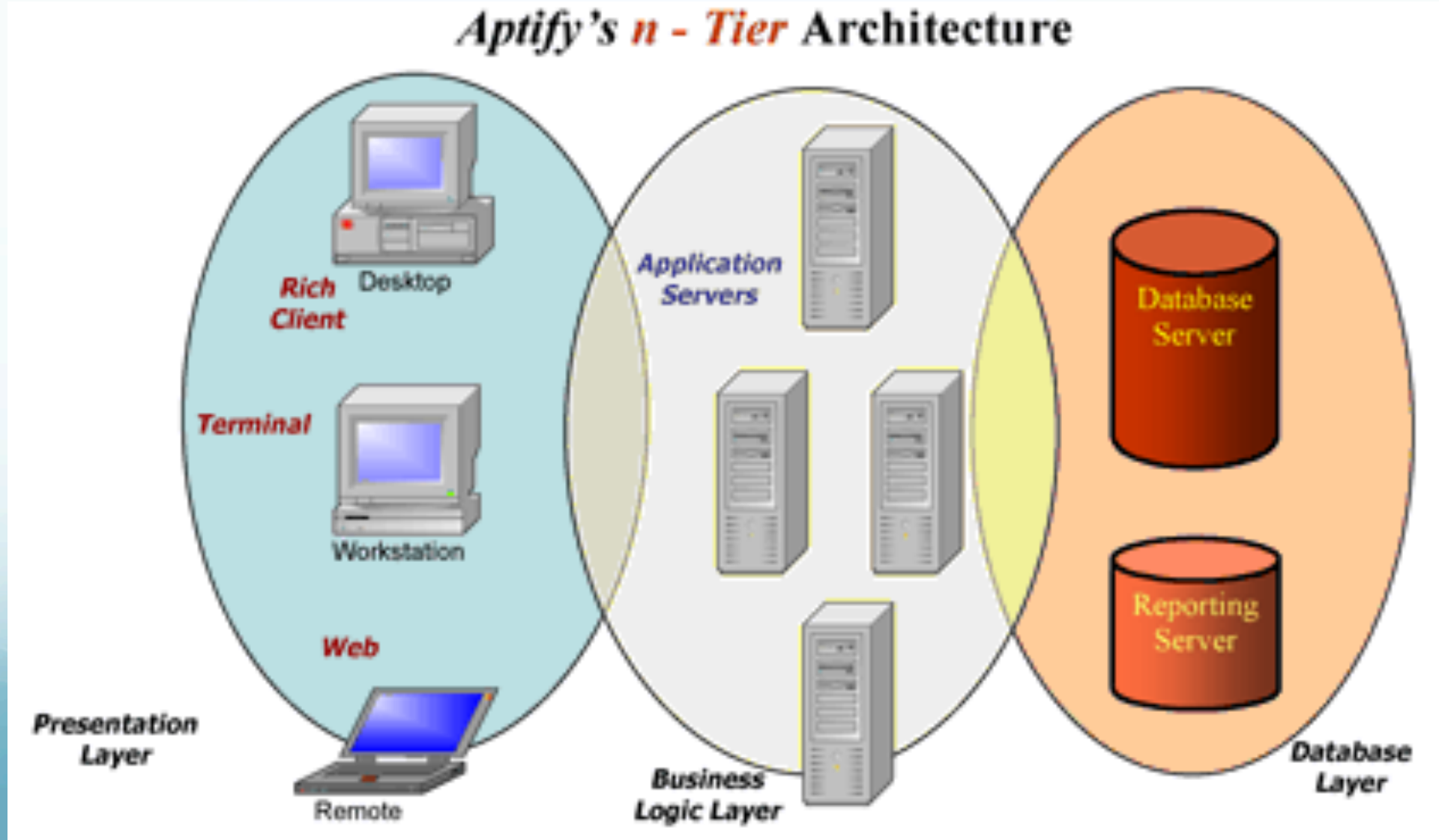
Increasingly, back end systems and infrastructure are growing as servers interact with each other to process requests. The objective is to provide a more streamlined, efficient architecture in which each server has a specific objective/role.

Large-scale distributed information systems

2nd & 3rd year project work typically develops 3-tier web based applications:



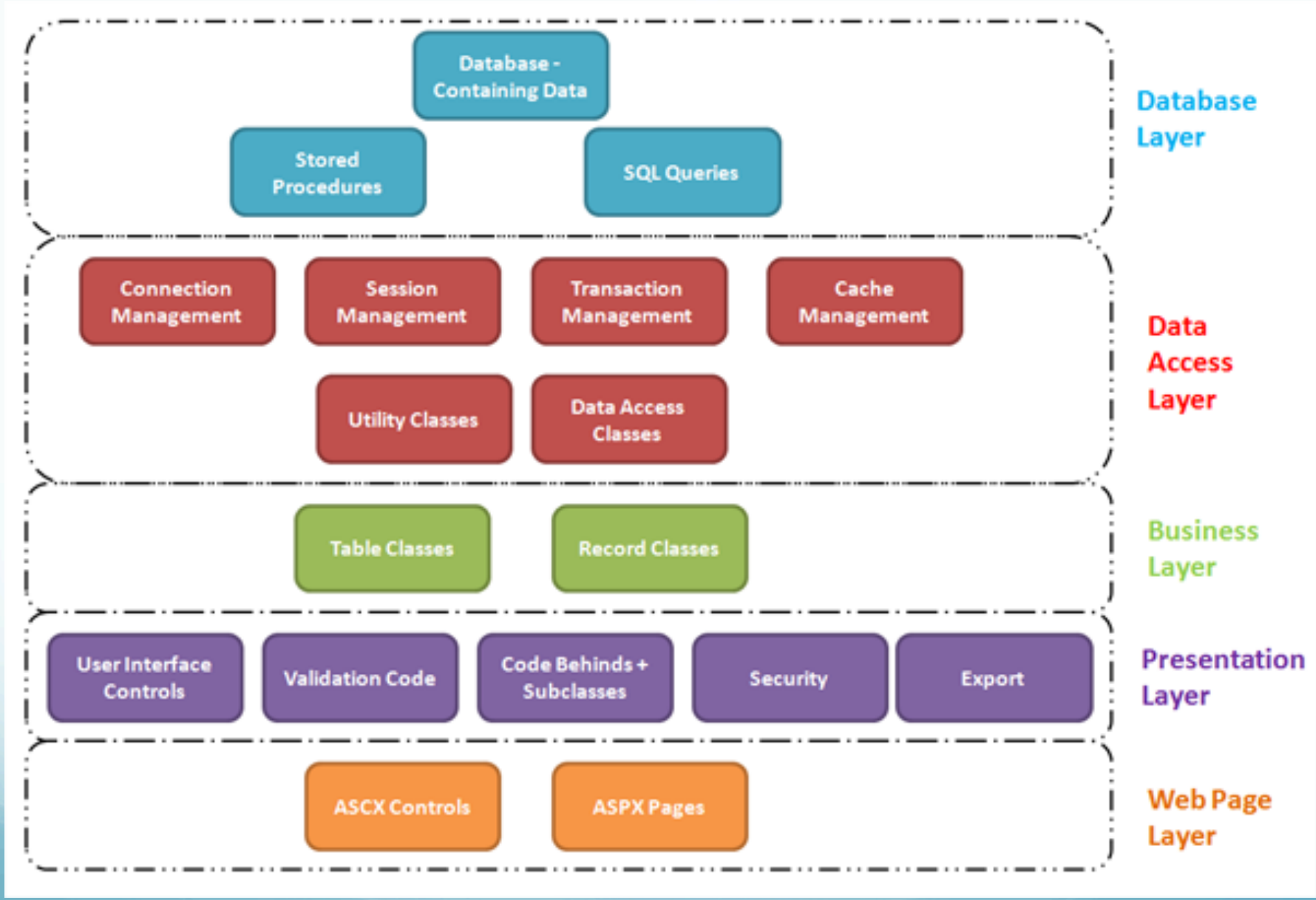
Enterprise Applications are typically *n*-tier architectures



N-tier architecture

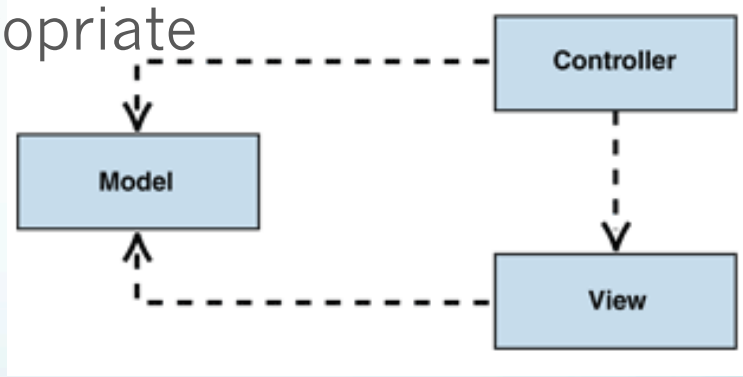
- The presentation layer must support a variety of client types
- The web server & business logic layer can be distributed across a number of physical servers, each implementing different aspects of the implementation.
- The database layer can be supporting a variety of database implementations distributed over the network

These three tiers can be subdivided into many components, but the overall architecture is the same. The image below is an ASP.NET architecture



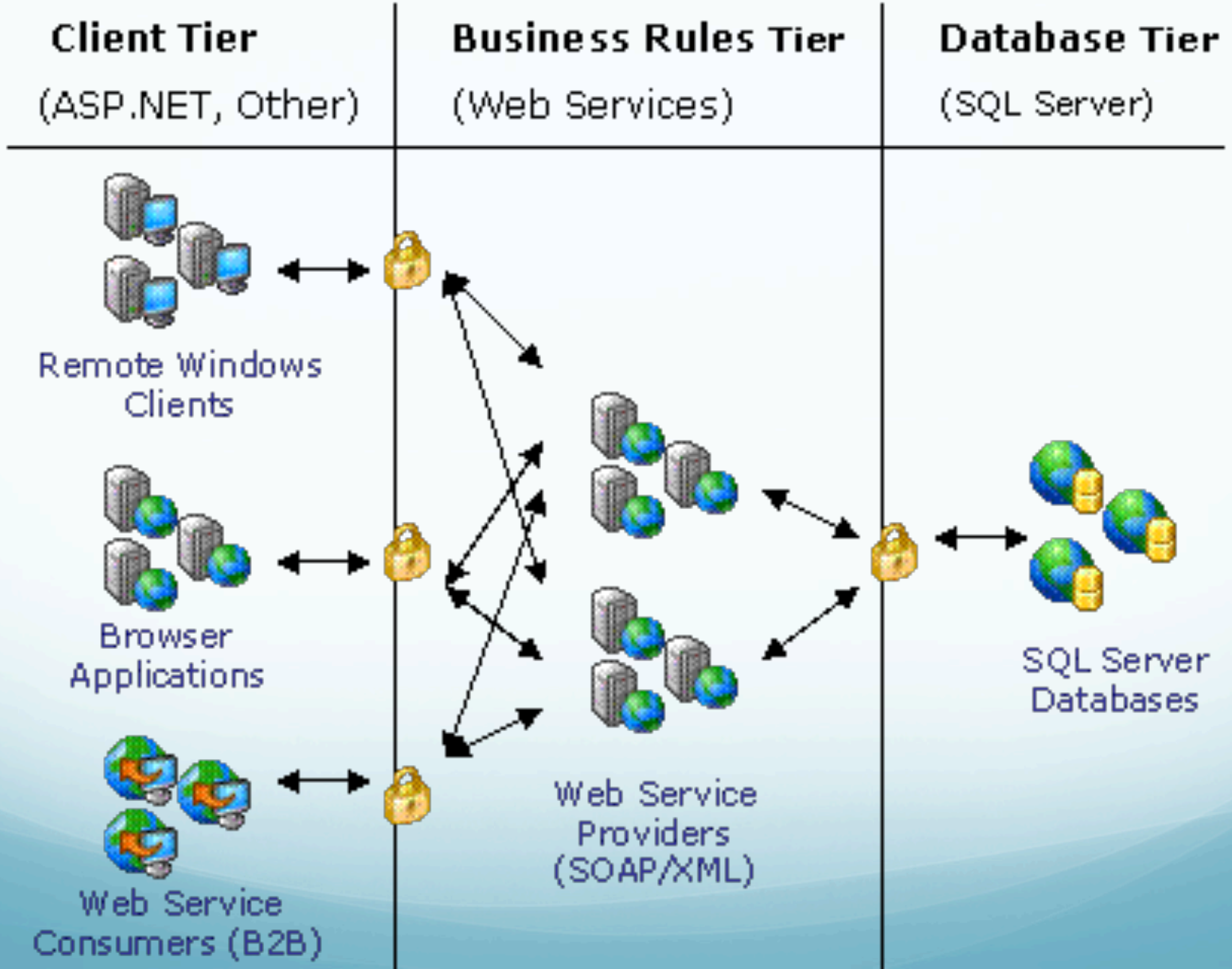
Examples of common architectures:

- MVC architectural pattern – Model View Controller
  - **Model** represents the underlying data for the application, and processes data related requests
  - **View** – renders the model in a form suitable for interaction
  - **Controller** – interprets user input, and informs the model and view to change as is appropriate



- SOA – service oriented architecture:
  - User Interface
  - Enterprise Service Bus (Middleware Layer)
  - Back-End Services
  - Web Services

Service Oriented Architecture - .NET



Module Learning outcomes - knowledge

Having successfully completed this module, the student will be able to :

- Discuss in detail all services provided by an application server.
- Explain the Java EE architecture and its components, and discuss current developments in Java EE.
- Understand integration issues that arise when using multiple architectures, and current solutions.
- Discuss the range of standards supporting persistence for object oriented application development.
- Explain the role and significance of cloud computing in application development.

Module learning outcomes – skill

Having successfully completed this module, the student will be able to:

- Design, implement and test a distributed application running on an application server, with an implementation based on, or equivalent to, the Java EE architecture.
- Use an IDE that supports Java EE development.

Level 8, BSc hons: can learn state of the art technologies, and manage ones own learning

CA – 40%

- CA will be broken down as follows:
  - Project: Full Java EE implementation., value: 25%
  - Class test/lab exercise on cloud computing. value: 10%
  - Weekly MCQ tests/Lab work – 5%

- Late assessments will be marked down by one grade.

What do you currently know?

- JSP
- Servlets
- Annotations
- JSF
- XHTML
- Web application container
- Tomcat / Glassfish
- Web service access
- Cloud computing
- Database normalisation
- JDBC
- ORM tools
- Persistence tools
- Javascript
- AJAX
- Java EE
- Backing beans

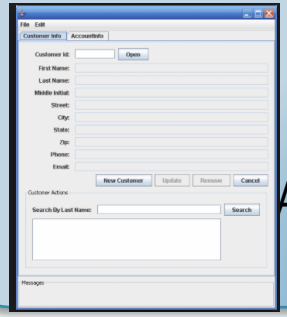
Introduction to Java EE

What is Java EE

- Java EE, Java Enterprise Edition (formerly J2EE) is a suit of API's designed to make Enterprise Applications easier to build.
- Java EE consists of a number of tiers, generally a:
  - Client tier
  - Middle tier
    - Web tier
    - Business tier
  - Data tier / Enterprise Information

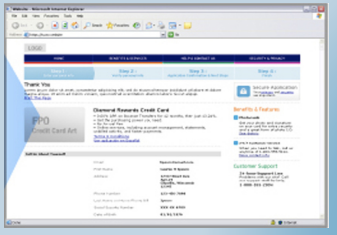
Java EE tiers

Java EE Application 1



Application Client

Java EE Application 2



Web Pages

Client Tier

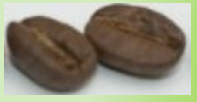
Client Machine

Java Server Faces pages

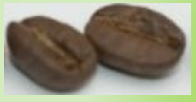
Web Tier

Java EE server(s)

Enterprise Beans



Enterprise Beans



Business Tier

Data Tier

Database server(s)

Client tier & Data tier

- The client tier consists of the clients' machine.
  - This contains either an application running on the client machine, or the dynamic HTML pages that are viewed by a browser on his machine. The client tier generally runs on a different machine from the Java EE Server.

- Information / Data tier
  - The database(s) storing the information relevant to the application. These may be distributed across the enterprise, and running on various DMBS implementations (MySQL, SQL Server, Oracle), not all of them relational databases (OODBMS-objectstore, XML databases etc.).

Web tier

- The Web Tier contains components that generate the web front end for the application, and run in a web container (e.g. Tomcat). It is responsible for:
  - Generating dynamic content
  - Collecting input from user
  - Controlling the flow between screens / pages
  - Maintaining state during a user session
  - May perform some basic logic
- Components are usually Java based, such as **Java Servlets, JSP** files, or developed using a framework such as **Java Servers Faces (JSF)**.
- The presentation layer can run on a separate machine from other components of the application.

Business Tier

- The Business Tier contains the Java code that implements the functionality of the application including:
  - Reading/Writing/Updating data in the database
  - Implementing the business logic for the application, e.g. e-commerce site, payroll, financial services etc.
- This layer consists of:
  - Managed beans that store variable information for a JSF page (XHTML form) and manages page navigation
  - Enterprise Java Beans (**EJBs).** EJB components are deployed in an **EJB container** which provides **security**, **transaction management** and the **ability to scale** the application.

What's an EJB?

- **A bean** is a programming model/java class that allows the developer to focus on its business purpose.
- An **Enterprise Java Bean** (EJB) is a java program that runs within an **EJB container** and so can use functionality provided by the EJB container.
- The **EJB container** is responsible for making the objects distributed, and for managing services such as transactions, persistence, concurrency, security and resource management

Types of EJB's

- There are three types of Enterprise JavaBean server-side components
  - **JPA Entities** (remember entities in an ERD?)
    - Model business concepts that can be expressed as nouns
    - Customer, Product, Order, Piece of Equipment, Patient, Perscription
    - An entity bean is an object representing a row in a database table
  - **Session Beans**
    - Mangages processes or task, i.e. the business transactions - verbs
    - Ordering a product, making a reservation, prescribing medication
    - Often modeling the relationship between entity beans
    - A session bean represents everything that happens during the course of a transaction, including updating entity beans in the database

Entities.

Has a persistent state

Processes.

Does not have a persistent state

Types of EJB's

- **Message Beans**
  - Handles asynchronous communication
  - It's a JMS (java messaging service) listener
  - Accepts messages from any other entity in the JEE architecture.

Subsequent lectures will look at enterprise beans in more detail. The following slides give more information on the EJB container . . .

What is an EJB container?

- An **EJB container** provides the run time environment for Enterprise Java Beans (EJBs) supporting:
  - Distributed objects
  - Transaction management
  - Persistence
  - Concurrency
  - Security
  - Resource Management and Instance Handling

Distributed objects

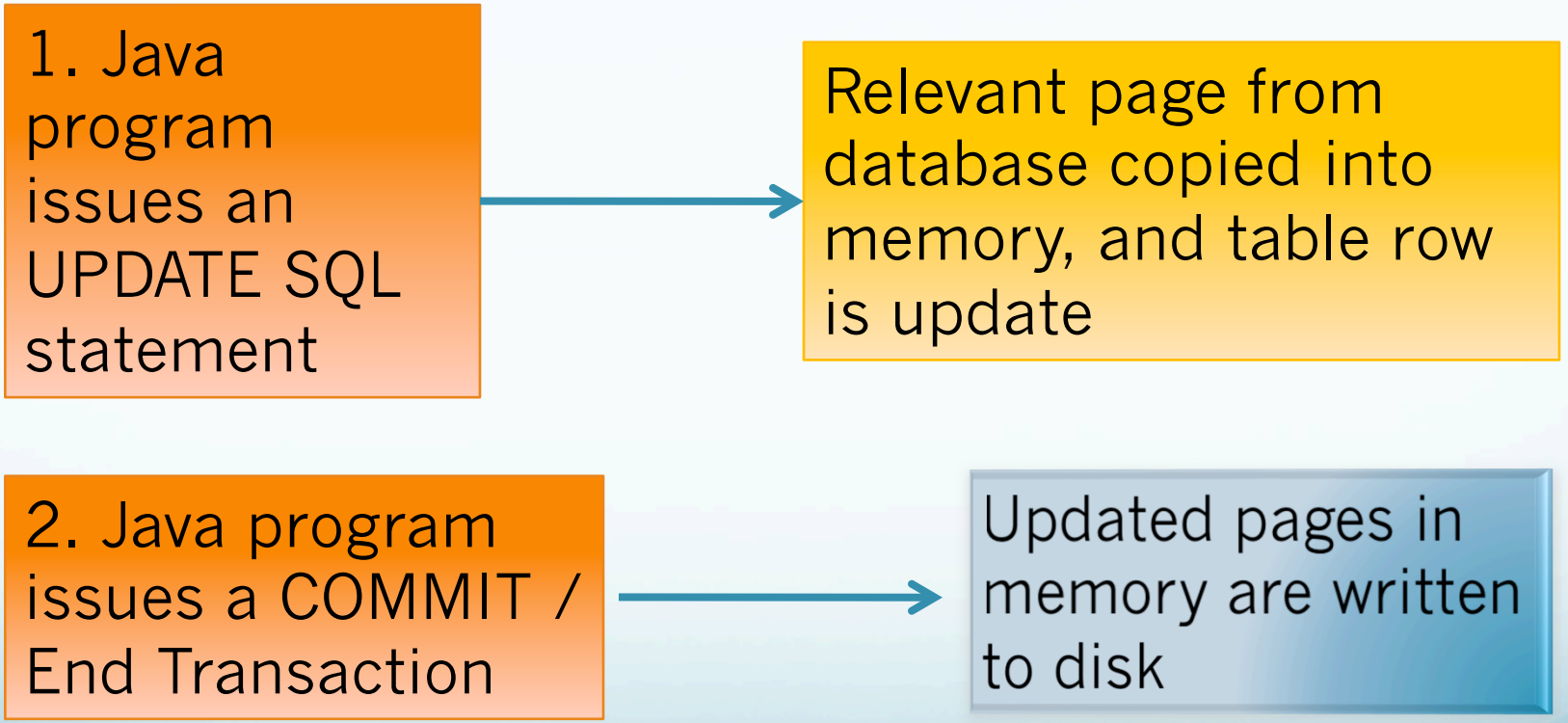
- Supporting distributed objects means providing location transparency.
  - If object A needs to call a method in object B, is does not need to know where B is located, just that B exists.
  - JNDI (Java Naming and Directory Interface) provides a naming and directory service for objects written in Java.
    - It defines how objects should be named
    - Maintains a directory of where objects are located
    - Methods calls using a JNDI name can access an object with out knowing its location as the JNDI directory tracks the location.
- An EJB container implements a JNDI directory service.

Transactions

- **Transactions** are a collection of database changes that comprise of a **business transaction**.
  - For example, if you want to move €100 from your savings account to your deposit account, the transaction would be as follows:
    - Begin transactions
    - savings_balance = savings_balance – 100
    - deposit_balance = deposit_balance + 100
    - End transaction/Commit
  - Managing transactions ensures either ALL updates are completed, or NO updates are completed.

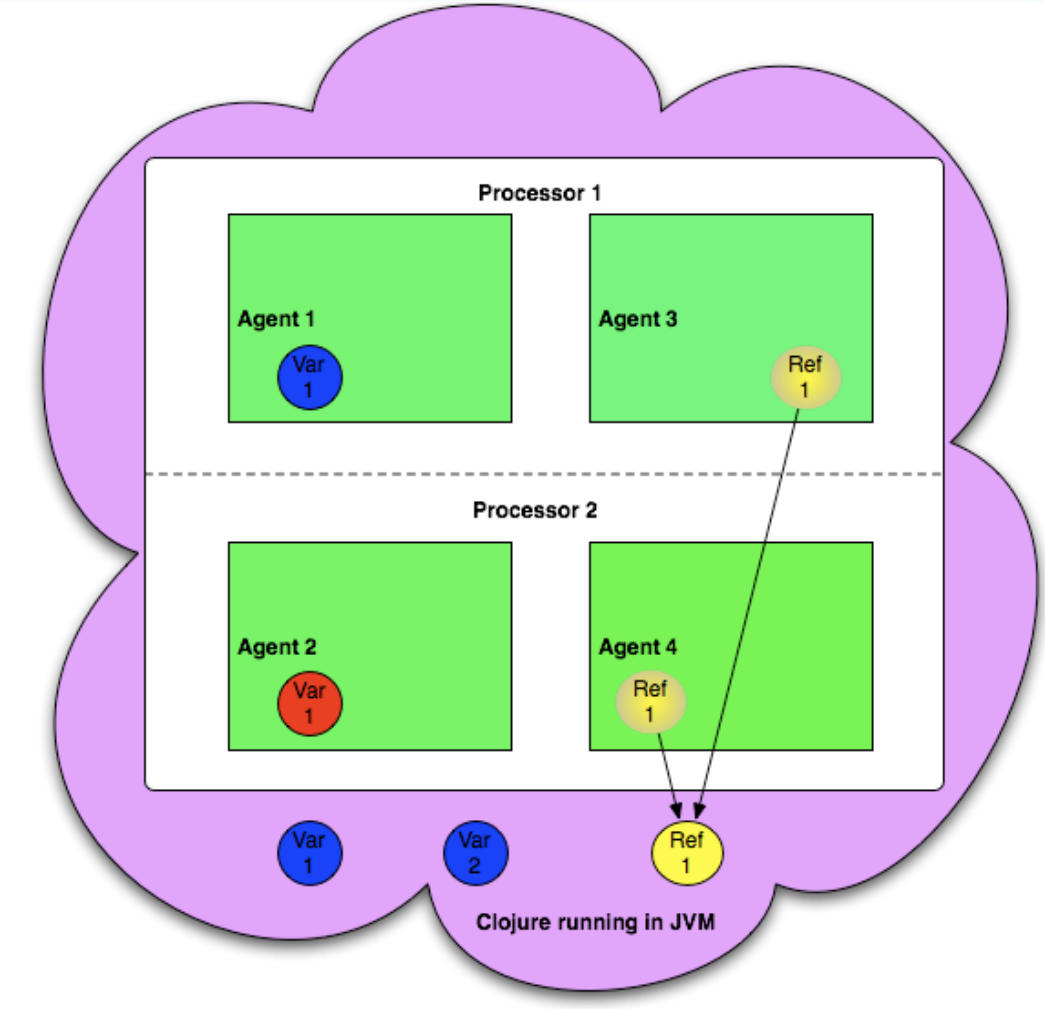
Persistence

- Persistence means changes to information / data persist past the life time of the program making the change, i.e. changes are recorded in a database.



Concurrency

- Concurrency:
  - Can handle multiple users.
  - Updates made by one user will never overwrite updates made by another user



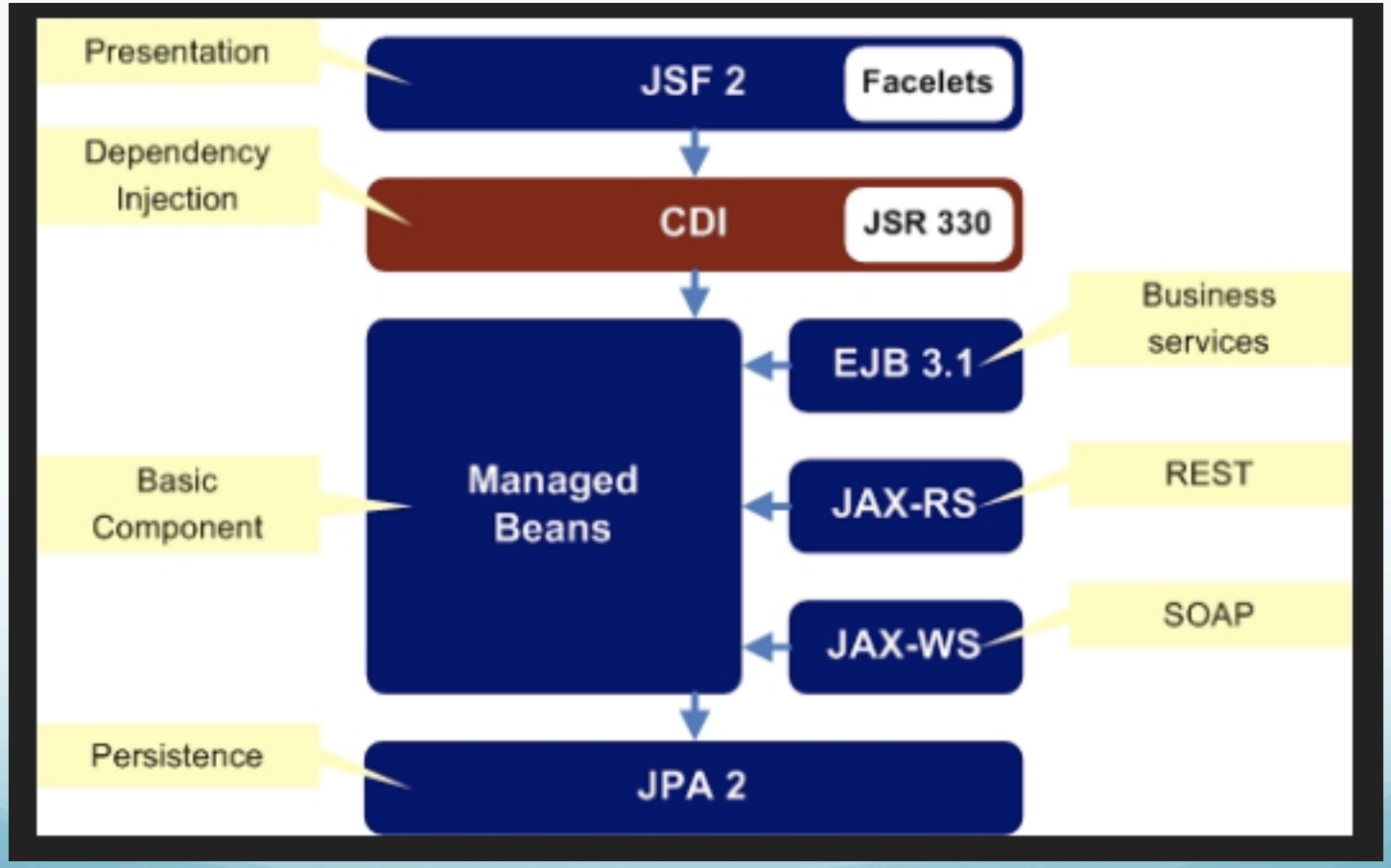
Security

- **Authentication**: Validates the identity of the user.
  - Can user logon and password, ID cards, swipe cards, security certificates,e tc.
- **Access Control**: applies security policies that regulate what a specific user can do and can not do within the system
  - Control access to subsystems, data or business objects
  - Control behaviour – updates, deletes, read only
- **Secure communication** between client and server
  - Could use a dedicated network line
  - More common to use Encryption - SSL

Resource Management & Instance Pooling

- Instance pooling is managing a collection of bean instances so they are quickly accessible at runtime
  - The EJB server creates serveral instances of the bean, and then holds on to them until they are needed
  - As clients make business method requests, bean instances from the pool are assigned to the EJB object associated with the client
  - When the EJB object doesn't need the instance any more it is return to the pool
  - Instances are selected arbitrarily from the pool
- The EJB server maintains instances pools of every type of bean deployed

The technologies and frameworks used in each tier



Java EE technologies at a glance

- **Web Services technologies**
  - JAX-WS
  - JAX-RPC
  - JAXB
  - SOAP, SAAJ
  - Web Services Metadata
- **Web App Technologies**
  - Java Server Faces, visual java server faces
  - JSP & JSP standard Tag library
  - Servlets
- **Enterprise App Technologies**
  - Annotations
  - EJB's
  - JavaMail
  - JMS
  - Java Persistence API
  - Java Transaction API
  - JEE connector architecture (connecting to information source)
- **Management & Security**
  - Application deployment
  - Application mgmt
  - Java Authorisation

Application Frameworks

- An application framework is a software framework that is designed to support the development of applications.
- The purpose of the framework is to reduce development time by providing libraries of commonly used functionality, for example: database access routines.
- Frameworks tend to support particular architectures such as MVC (model view controller) or SOA (service oriented architecture)
- Examples include:
  - JavaServer Faces
  - Struts
  - Spring
  - Hibernate

Implementing JEE based applications

- IDE: **Netbeans** - includes Glassfish application server, Derby database.
- The glassfish application server incudes a web container and an EJB container. It supports a number of frameworks including:
  - JSF – java server faces
  - JPA - java persistence API
  - SPRING (MVC application framework)

Why learn these technologies?

Take a look at http://www.computerjobs.ie searching under

Spring or JEE