Web Applications Developments

ITWT413

DR. HALA SHAARI

Course Structure

- Lecture (Sunday& Tuesday)
- Time: 12:30-14:00
- Microsoft Teams Code (goi2d67)

(Lectures, labs, announcements, References):

- Grading :
  - 25% Midterm exam.
  - 25% Assignments
  - 25% Group Project (Groups of two).
  - 25% Final Exam.

Lecture Agenda

- Introduction to Enterprise Systems
- Overview of Java EE
- Creating an Application
- Data Layer
- Logic Layer
- Creating the Interface

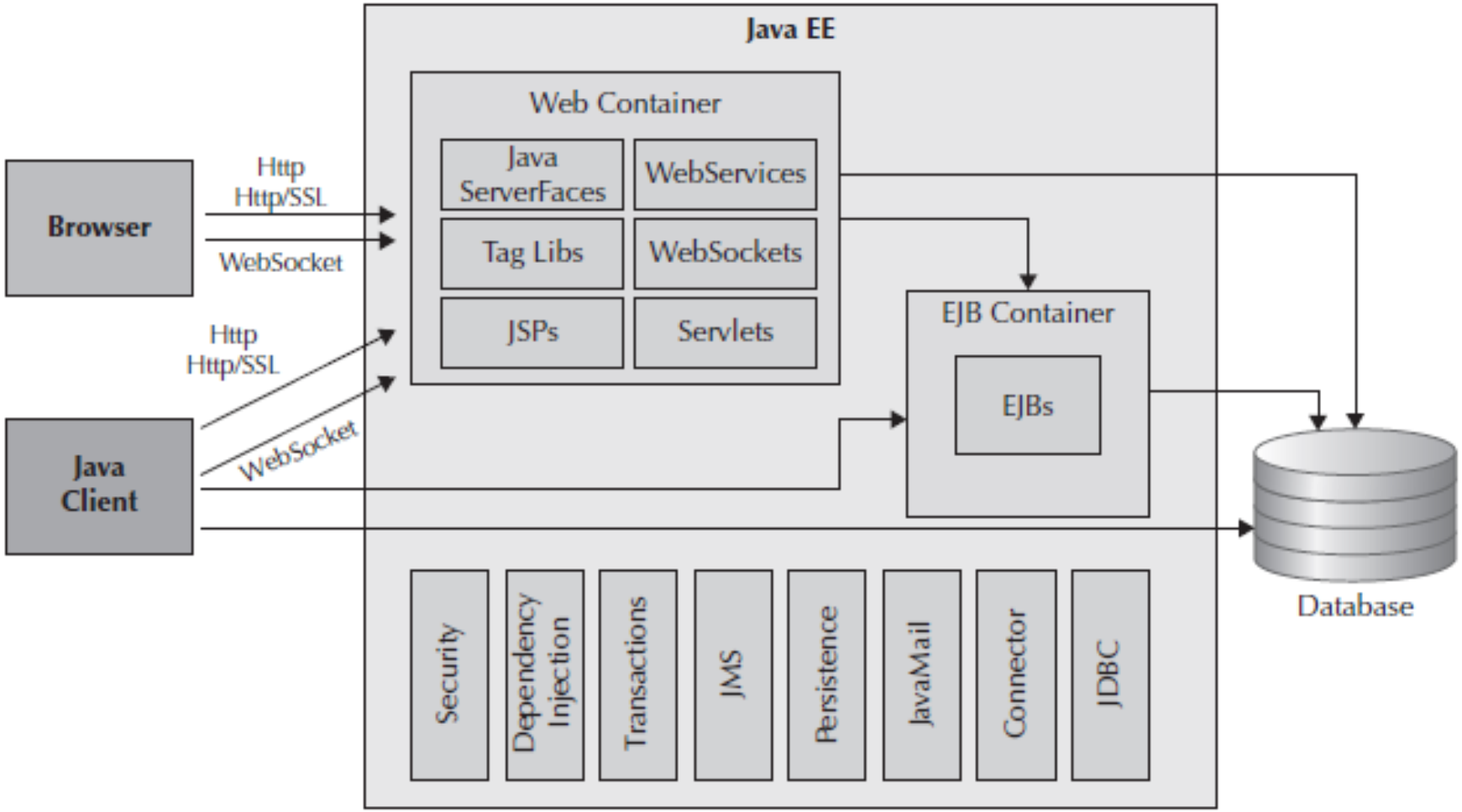
1. Introduction to Enterprise Systems

Definition of Enterprise Systems (ES)

"Enterprise System is software applications that have cross-organizational capabilities as opposed to department or group-specific programs."

-By Neil Kokemuller

- Enterprise systems enable collaboration, communication, and data sharing.
- Features: availability, scalability, performance, security, manageability, data integrity.
- Interoperability is crucial for communication between systems.


Java EE
Web Container
Java ServerFaces
WebServices
Tag Libs
WebSockets
JSPs
Servlets
EJB Container
EJBs
Browser
Http
Http/SSL
WebSocket
Http
Http/SSL
Java Client
WebSocket
Database
Security
Dependency Injection
Transactions
JMS
Persistence
JavaMail
Connector
JDBC

1.1 Features of Enterprise Systems

- Availability: Systems must handle increased demand without downtime.
- Scalability: Systems must adapt to organizational changes.
- Performance: Improves business workflow efficiency.
- Security: Ensures data confidentiality and system availability.
- Manageability: Complexity management to avoid system failures.
- Data Integrity: Protects data from corruption or loss.
- Interoperability: Facilitates system communication and collaboration.

1.2 Technologies

- Java EE: API set for enterprise application development.
- .NET: Another common technology for enterprise systems.
- Focus in this course is on Java EE.

2. Overview of Java EE

- Java EE provides infrastructure for creating and maintaining enterprise systems.

- Includes APIs like JPA for persistence, JTA for transactions, and JAAS for security.
  - JTA: Java Transaction API provides a standard interface for demarcating transactions. The Java EE architecture provides a default auto commit to handle transactions commits and rollbacks.
  - JPA: Java Persistence API is a solution for persistence. Persistence uses an object/relational mapping approach to bridge the gap between an object-oriented model and a relational database. JPA can also be used as a query language.
  - Security Services: Java Authentication and Authorization Services (JAAS) enables services to authenticate and enforce access controls upon users

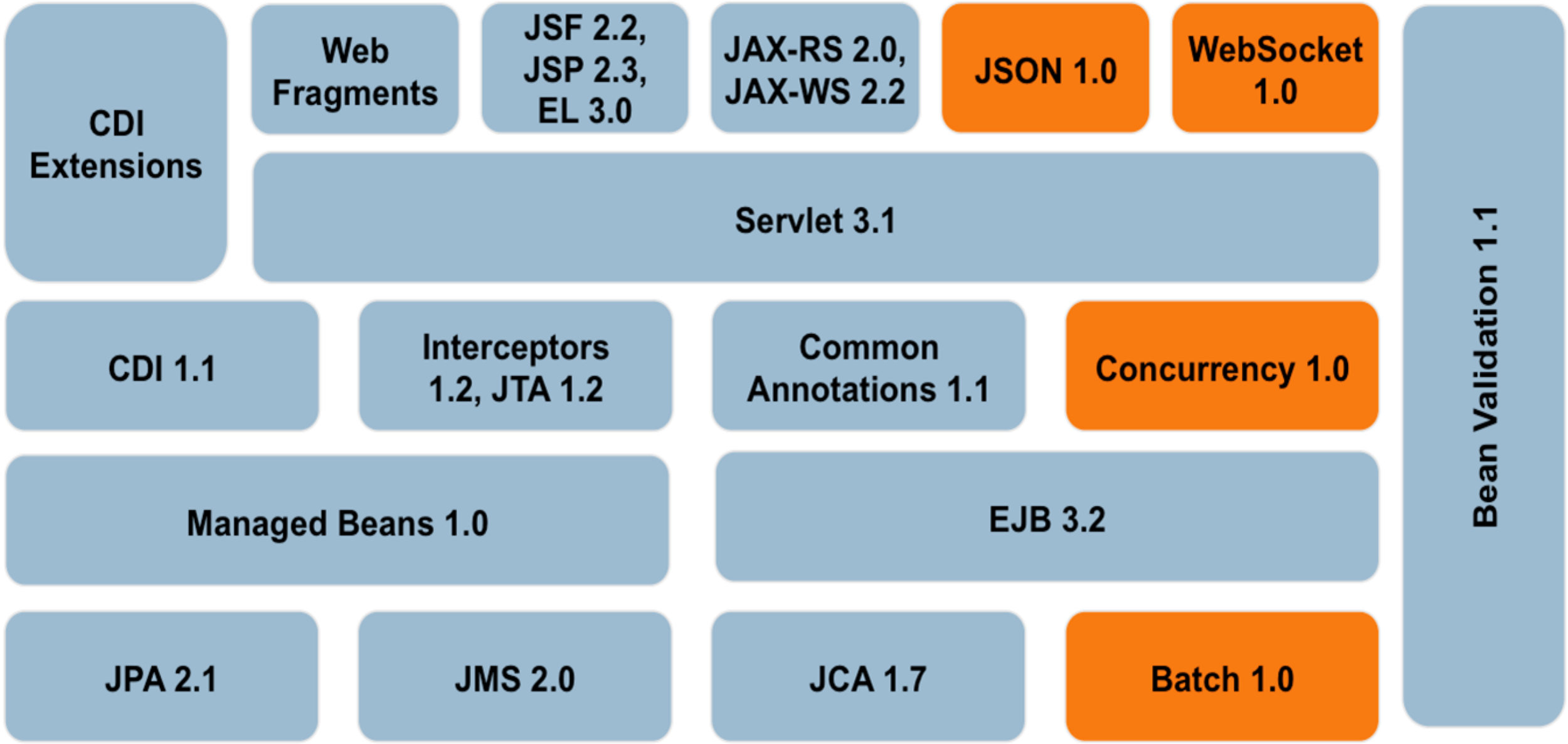
2.1 Understanding Java EE

- Java EE is a set of standards for server-side applications.

- Ensures compatibility with any compliant Java EE server.

2.2 Java EE8 Specifications

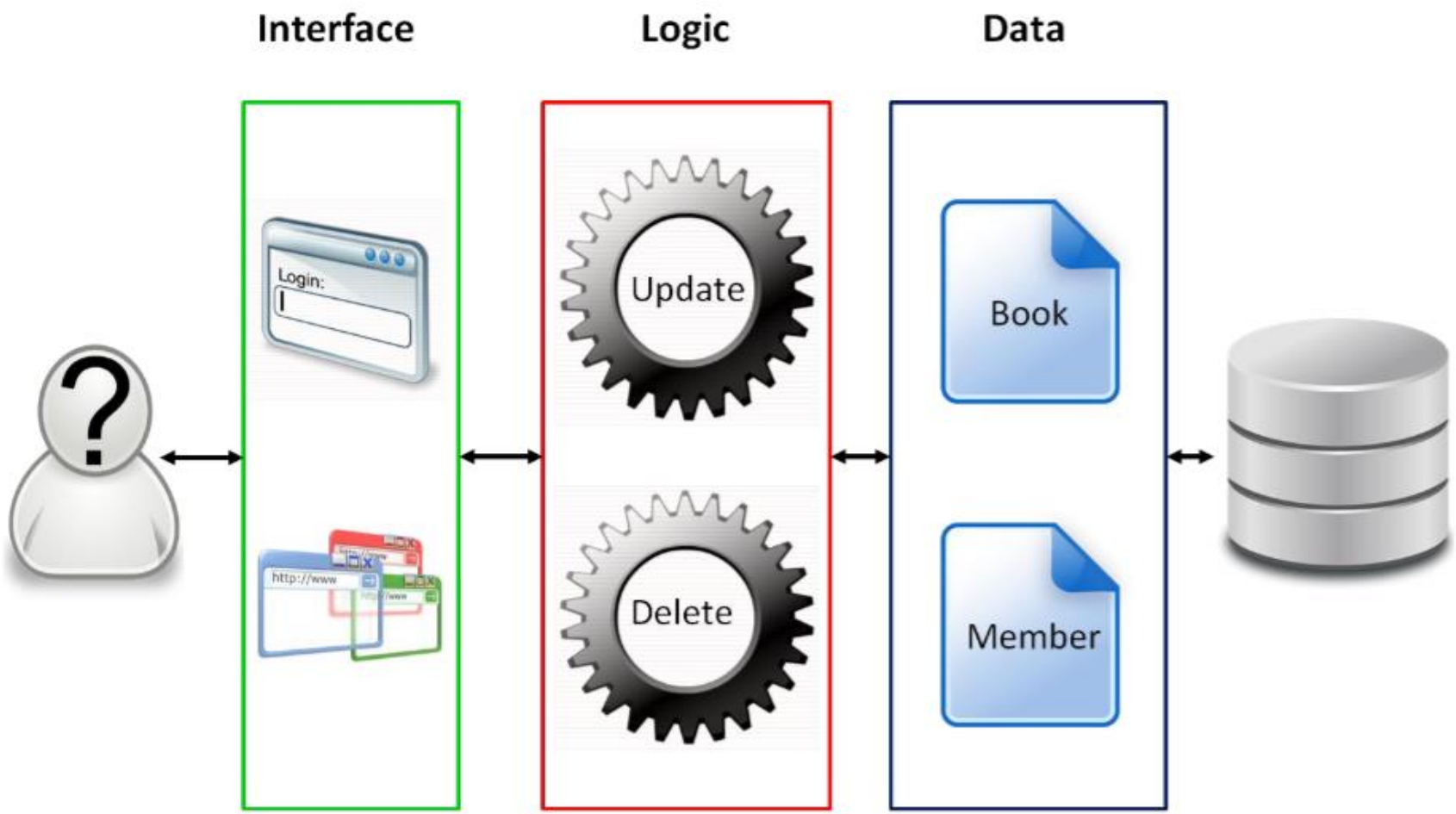
- Created by the Java Community Process (JCP).
- Java SE for desktop apps, Java EE for enterprise, Java ME for embedded systems.
- Java EE has 28 specifications covering various enterprise services.


CDI Extensions
Web Fragments
JSF 2.2, JSP 2.3, EL 3.0
JAX-RS 2.0, JAX-WS 2.2
JSON 1.0
WebSocket 1.0
Servlet 3.1
CDI 1.1
Interceptors 1.2, JTA 1.2
Common Annotations 1.1
Concurrency 1.0
Managed Beans 1.0
EJB 3.2
JPA 2.1
JMS 2.0
JCA 1.7
Batch 1.0
Bean Validation 1.1

3. Creating an Application

- Applications consist of UI, business logic, and data layers.

- Example: A Library System to show interactions between these layers.


Interface
Logic
Data
Login:
Update
Delete
Book
Member
http://www

4. Data Layer

- Handles persistent data using relational databases.

- JPA provides object-relational mapping for persistence.

4.1 JPA Overview

- JPA handles persistence via entities and manages data in relational databases.

- Includes components like Entity Manager API and JPQL for querying.
  - Object Relational Mapping, which is the mechanism to map objects to data stored in the database.
  - An Entity Manager API to perform database-related operations such as Create, Read, Update and Delete (CRUD) operations.
  - The Java Persistence Query Language (JPQL), which allows users to query and retrieve data easily.

4.2 Understanding Entities

- Entities are persistent objects stored in databases.

- Annotations are used to customize entity-to-table mapping.

4.3 Creating Entity Class

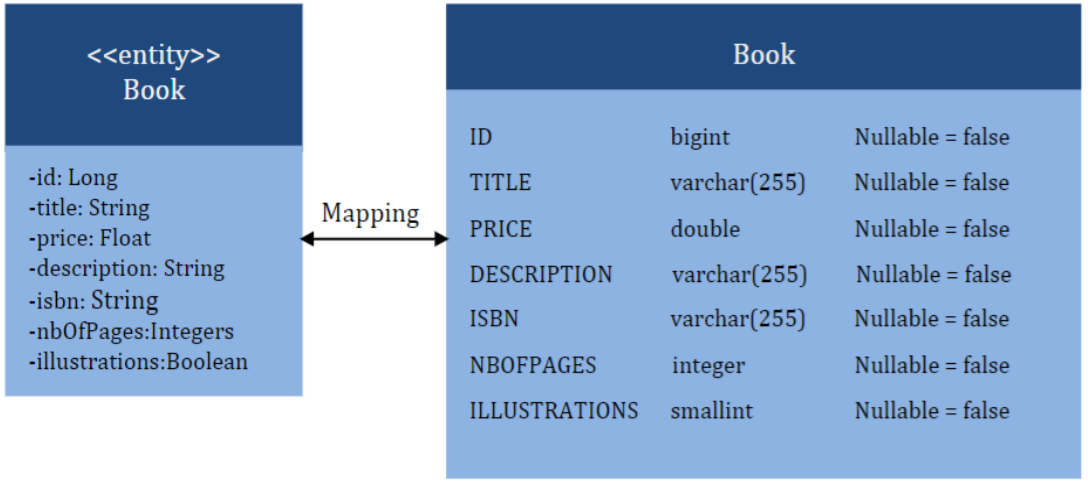
- Example: Book entity class with attributes like title, price, and description.

- Use annotations like @Entity, @Id, and @Column for mapping.

```
@Entity(name = "Book")
public class BookEntity implements Serializable {

        @Id
        @GeneratedValue(strategy = GenerationType.AUTO)
        private long Id;
        @column(nullable = false)
        private String title;
        private Float price;
        @column(length = 2000)
        private String description;
        private String isbn;
        private Integer nbOfPage;
        private Boolean illustrations;
        private int CopyNumber;

        .....
}
```



4.4 Entity Manager

- Entity Manager manages database operations like CRUD.

- Uses persistence context to track and manage entity changes.

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("chapter02PU");
EntityManager em = emf.createEntityManager();
em.persist(book);
```

5. Logic Layer

- Contains business logic, implemented with Enterprise Java Beans (EJB).
- EJB simplifies database operations and scalability.

5.1 EJB Overview

- EJB encapsulates business logic for server-side applications.

- Manages persistence through the Entity Manager API.

5.2 EJB Container

- EJB Container provides services like transaction management and security.
- In Enterprise Java Bean technology, there is one component called the EJB Container, which is responsible for managing Enterprise JavaBeans.
- The following list is some of the important service:
  - Security
  - Transaction management
  - Remote accessibility
  - Resource and life cycle management
  - Clustering and load-balancing
  - Concurrent requests support

-It also ensures availability and scalability by managing bean instances dynamically.

5.3 Benefits of EJB

- Simplicity: Focus on business logic without worrying about infrastructure.
- Light-weight Client: Minimal client-side code required.
- Portability: Beans are portable and can be moved between servers.

5.4 Classifications of EJB

- Session Beans: Handle business logic (can be stateful or stateless).

- Message-Driven Beans: Process messages asynchronously.

5.5 Functionalities Provided by the Container

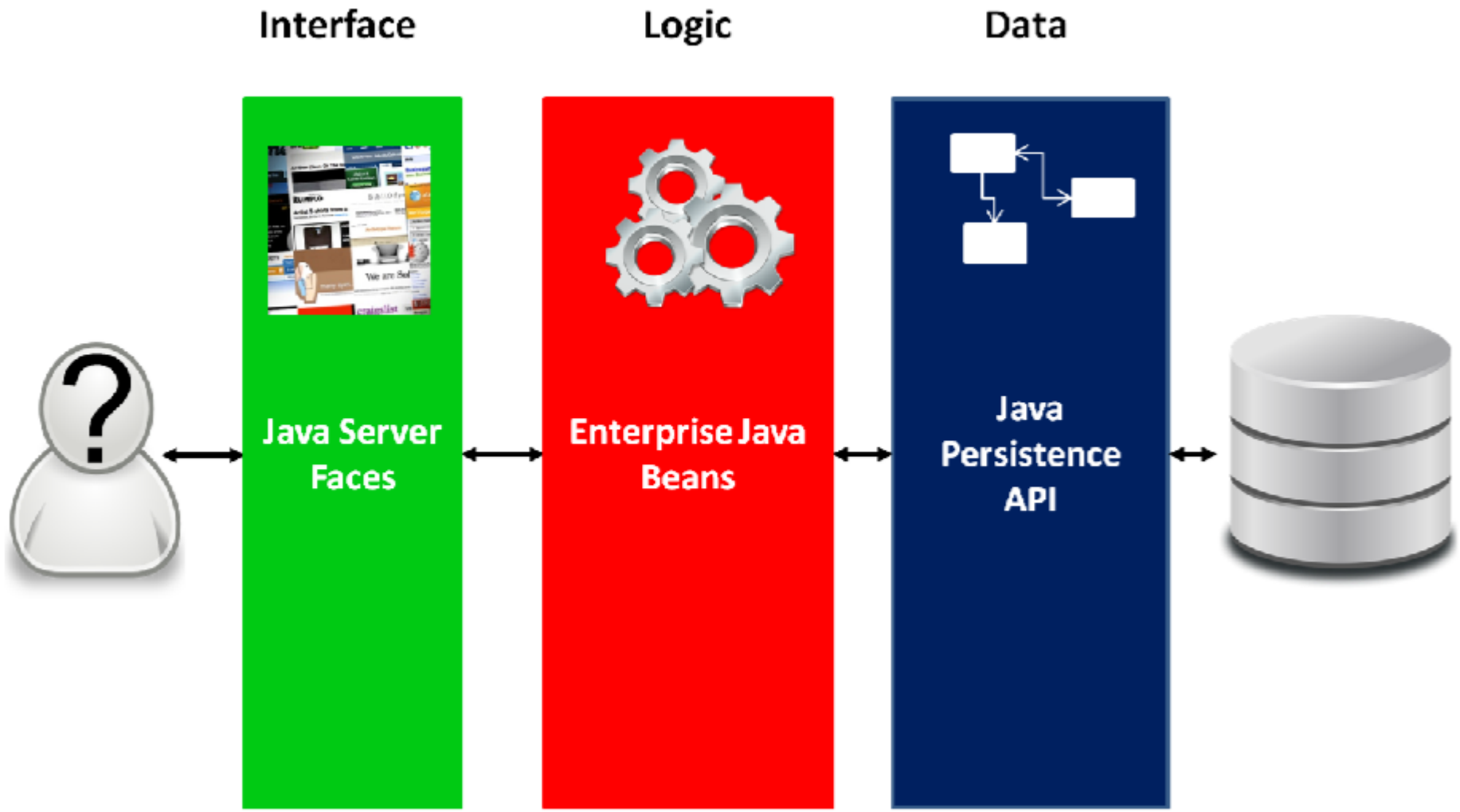
- Security: Role-based access control using annotations like @RolesAllowed.

- Transactions: Automatic rollback in case of failures.

6. Creating the Interface

- Java Server Faces (JSF) is used for building UI in web applications.

- JSF provides reusable components and libraries for UI development.


Interface
Logic
Data
Java Server Faces
Enterprise Java Beans
Java Persistence API

7. Conclusion

- Java EE simplifies the development of scalable enterprise systems.

- This chapter provides a basic guide to building enterprise applications.

Questions?