Enterprise Java Beans

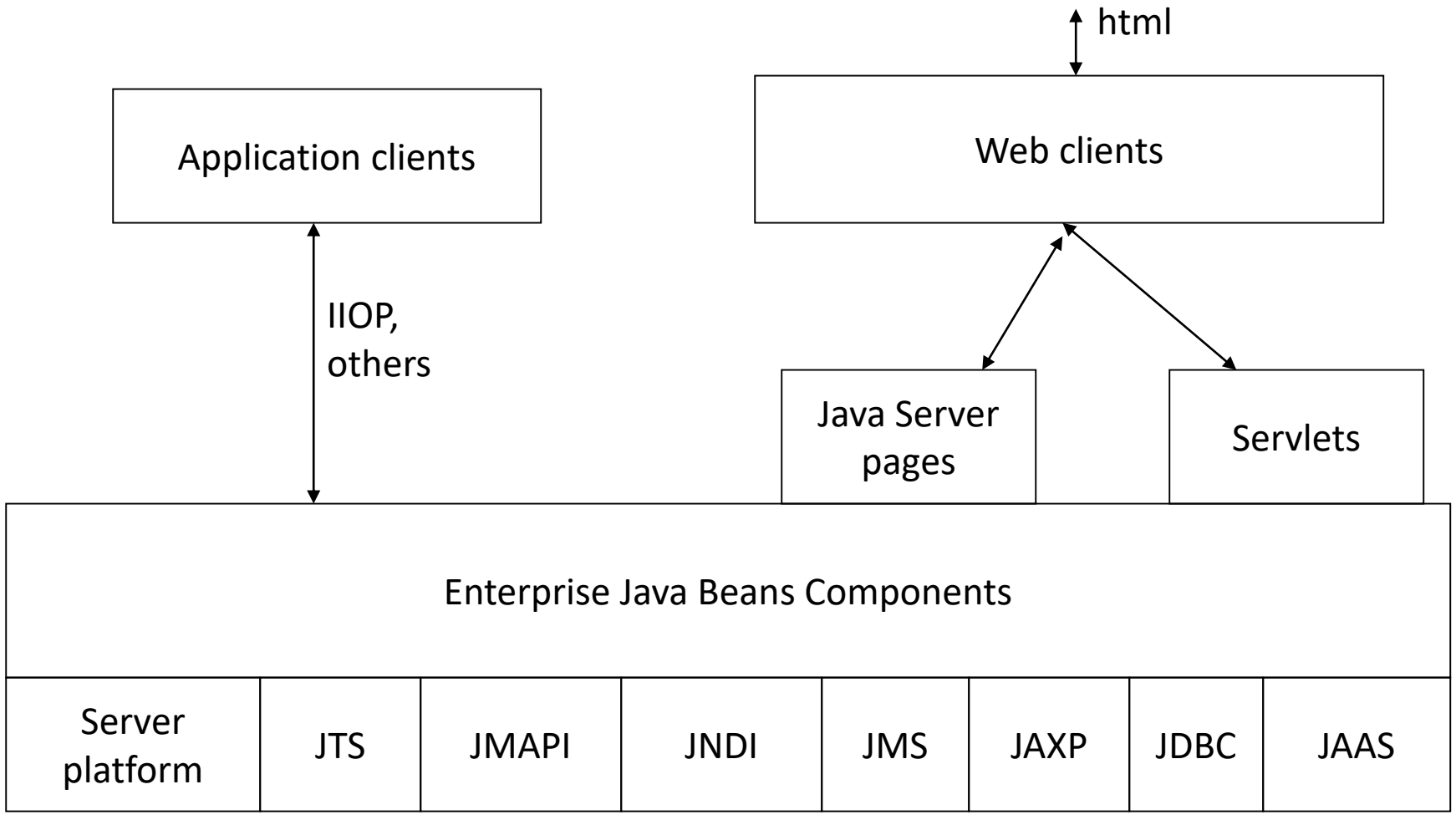
Swagat Ranjan Sahoo
Assistant Professor

Department of Computer Science and Engineering
GL Bajaj Institute of Technology and Management, Greater Noida-201306, India

Outline

- What is EJB
- Creating Java Beans
- Java Beans Properties
- Types of Beans
  - Session Beans
  - Entity Beans
  - Message-Driven Beans (MDB)

J2EE Technology Architecture



What is EJB?

- **Enterprise Java Beans (EJB)** is a server-side component architecture for modular construction of enterprise applications in Java.
- It contains the business logic of an application.
- At runtime, the application clients execute the business logic by invoking the enterprise bean's methods.
- It simplifies the development of distributed, transactional, secure, and scalable applications.
- EJB is part of the **Java EE (Jakarta EE)** platform.

Role of EJB Development

- Bean developer: Develops bean component
- Application assembler: composes EJBs to form applications
- Deployer: deploys EJB applications within an operation environment
- System administrator: Configures and administers the EJB computing and networking infrastructure
- EJB Container Provider and EJB server provider: Vendors specializing in low-level services such as transactions and application management

Enterprise Java Beans

- Deployable unit of code.
- At run-time, an enterprise bean resides in an EJB container.
- An EJB container provides the deployment environment and runtime environment for enterprise beans including services such as security, transaction, deployment, concurrency etc.
- Process of installing an EJB in a container is called EJB deployment.

Business Entities Process and Rules

- EJB Applications organize business rules into components.
- Components typically represent a business entity or business process.
- Entity: is an object representing some information maintained in the enterprise. Has a “state” which may be persistent.
- Example: Customer, Order, Employee,
- Relationships are defined among the entities: dependencies.

Creating Java Beans

JavaBeans are reusable components that follow specific design conventions. These are simpler compared to EJBs and are used for local, lightweight operations.

Steps to Create a JavaBean:

- **Create a Java class:** A JavaBean is essentially a standard Java class.
- **Add a no-argument constructor:** This ensures the bean can be instantiated easily.
- **Define properties:** Properties are private variables in the class.
- **Provide getter and setter methods:** These methods allow external code to read or modify the bean's properties.
- **Make the class serializable:** By implementing the java.io.Serializable interface, the bean's state can be saved and restored.

Creating Java Beans

Example JavaBean:

```
import java.io.Serializable;
public class Employee implements Serializable {
    private int empId;
    private String empName;
    // No-argument constructor
    public Employee() {}
    // Getter and Setter for empId
    public int getEmpId() {
        return empId;
    }
    public void setEmpId(int empId) {
        this.empId = empId;
    }

    // Getter and Setter for empName
    public String getEmpName() {
        return empName;
    }
    public void setEmpName(String empName) {
        this.empName = empName;
    }
}
```

Java Beans Properties

A JavaBean property is an attribute of the JavaBean that follows a naming pattern and is accessed through getter and setter methods.

Types of Properties:

1.Read-Only Property: Has only a getter method.

Example:

```
public String getName() {
    return name;
}
```

Java Beans Properties

Types of Properties:

2.Write-Only Property:

Has only a setter method.

Example:

```
public void setName(String name) {
   this.name = name;
}
```

Java Beans Properties

Types of Properties:

3.Read-Write Property: Has both getter and setter methods.

Example:

```
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
```

Types of Beans

EJBs are server-side components used in enterprise applications.

There are three primary types of beans in EJB:

i. Session Beans

ii. Entity Beans (Deprecated)

iii. Message-Driven Beans (MDB):

Types of Beans

i.Session Beans

- Session beans perform operations for a client.
- They are short-lived and not persisted.
- Session beans are of two types:
    1. Stateful Session Beans
    2. Stateless Session Beans

Types of Beans

1.Stateful Session Bean

- Maintains state for a client across multiple method calls.
- A single instance is created per client.
- Example Use Cases: Shopping cart, user session.

Example:

```
import javax.ejb.Stateful;
import java.util.ArrayList;
import java.util.List;

@Stateful
public class ShoppingCart {
    private List<String> items = new ArrayList<>();

    public void addItem(String item) {
        items.add(item);
    }

    public List<String> getItems() {
        return items;
    }
}
```

Types of Beans

2.Stateless Session Bean

- Does not maintain a client-specific state.
- A single instance can serve multiple clients.
- Example Use Cases: Performing calculations, email services.

Example

```
import javax.ejb.Stateless;

@Stateless
public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}
```

Types of Beans

ii.Entity Beans

- Represents persistent data stored in a database.
- Used in older Java EE versions; now replaced by **JPA (Java Persistence API)**.
- Example Use Case: Managing database records such as employees or products.

Example

```
import javax.persistence.Entity;
import javax.persistence.Id;
public class Employee {
    private int id;
    private String name;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

Types of Beans

iii. Message-Driven Beans (MDB)

- Used for processing messages asynchronously.
- Acts as a listener for JMS (Java Messaging Service).
- Example Use Case: Processing incoming emails or notifications.

Stateful vs Stateless Beans

Feature	Stateful Session Bean	Stateless Session Bean
State Maintenance	Maintains conversational state.	No state maintained between calls.
Instance Per Client	One instance per client.	Shared instance for multiple clients.
Performance	Relatively slower due to state handling.	Faster as no state management is needed.
Use Case	Shopping cart, user sessions.	Simple tasks like calculations.

Thank you!