Jakarta EE

Table of Contents

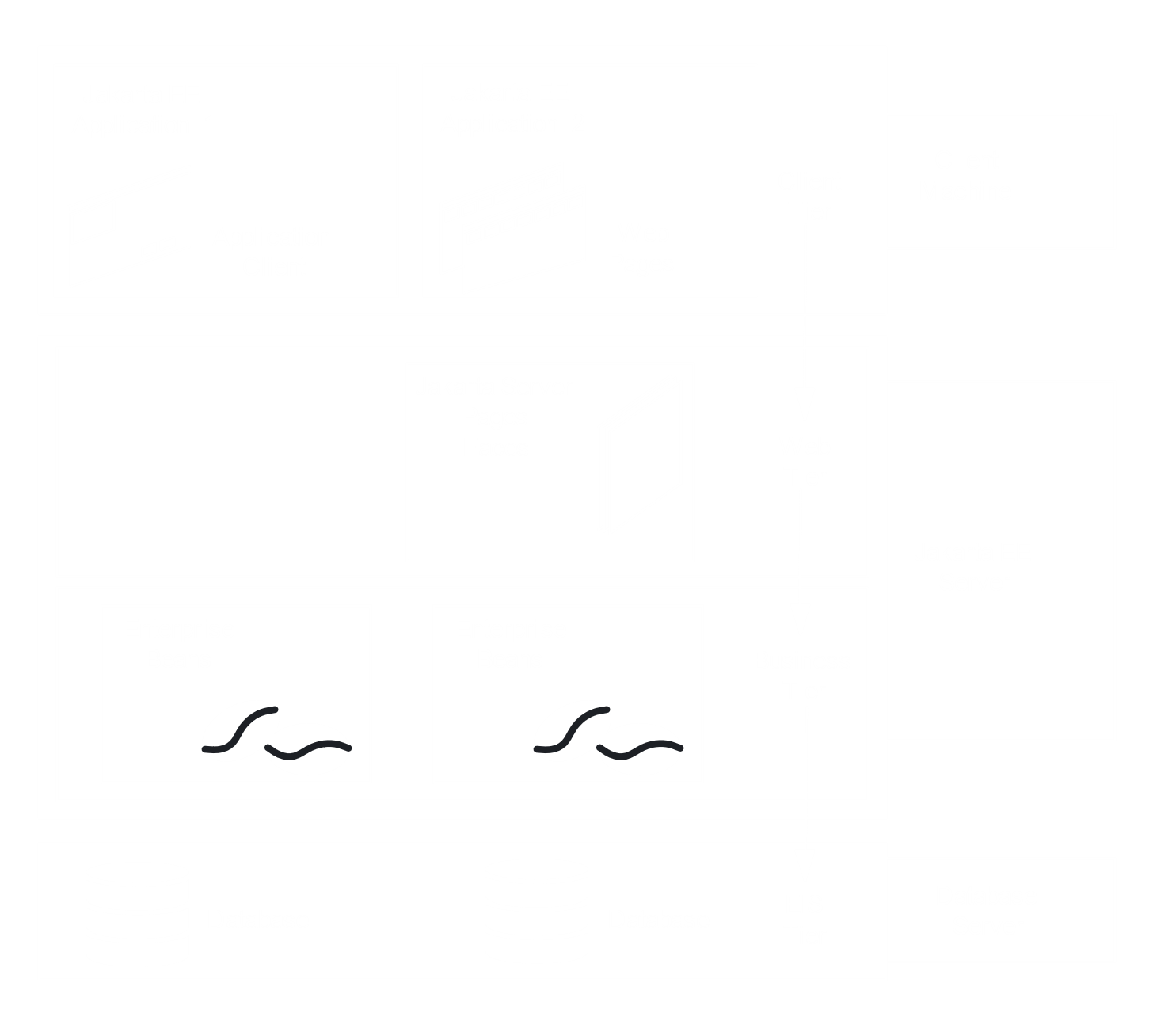
[Tiered Components 2](#_Toc100618646)

[Enterprise Application Technologies 3](#_Toc100618647)

**Java Enterprise Extensions** were originally part of the core JDK. As part of Java 2, it was rebranded to **Java 2 Platform Enterprise Edition** (J2EE). This was later rebranded again to **Java Platform Enterprise Edition** (Java EE). Finally, Oracle gave away the rights to the Eclipse foundation, who renamed it to **Jakarta EE** (JEE) and made it open source.

JEE are a set of specifications. The JEE platform uses a **multi-tiered application model** for enterprise applications. Application logic is divided into components and **components** are installed on machines depending on which tier the component belongs to.

## Tiered Components



**Client tier** components work on the client machine. They allow the user to interact with the application the display information to them. Several types of clients are supported by JEE, including HTML, Java, etc.

**Web Tier** components receive user input from the client tier components and generate responses. These include things like servlets, JSP pages, etc.

**Business Tier** components handle the business logic. These are usually implemented using EJB components from an EJB container. The EJBs take data from the client, process it and send it to EIS tier to store. It can also do the opposite, retrieving data from storage to send to the client.

The **EIS Tier** handles information systems, including database systems, transaction processing systems, legacy systems and resource planning systems. This is the point at which the application integrates with non-JEE systems.

## Enterprise Application Technologies

* **Enterprise Java Beans** (EJBs) – Standard building blocks which contain the business logic for enterprise applications.
* **JEE Connector Architecture** – Architecture for connecting the JEE platform to enterprise information systems.
* **Java Message Service API** (JMS) – Specification for an API for enterprise messaging systems.
* **Java Persistence API** (JPA) – Persistence model for mapping Java objects to a relational database and vice versa.
* **Java Transaction API** (JTA) – Allows applications to perform distributed transactions, i.e. transactions that access and update data on multiple networked computer resources. Also used to write JDBC drivers, EJB containers and hosts.
* **JavaMail** – Protocol and platform-independent framework to compose, write and read emails.
* **Java Servlets** – Server-side API for handling HTTP requests and responses.
* **JavaServer Pages** (JSP) – Text documents defining how dynamic content can be added to static HTML or markup pages which are then compiled to servlets.
* **JavaServer Pages Standard Tag Library** (JSTL) – Encapsulates core functionality common to JSP applications.
* **JavaServer Faces** (JSF) – An API framework that allows building of UI elements, managing their state, handling events, etc. following the MVC architecture.
* **Java API for RESTful Web Services** (JAX-RS) – API for RESTful web service development.
* **Java API for XML-Based RPC** (JAX-RPC) – APIs for XML-based RPC standards.
* **Java Architecture for XML Binding** (JAXB) – APIs and tools to automatically map XML representations and Plain Old Java Objects (POJOs).
* **SOAP with Attachments API for Java** (SAAJ) – Provides a standard way to send XML documents over the internet from the Java platform.