The J2EE platform (now called Java Platform, Enterprise Edition - Java EE) is essentially a distributed application server environment. It is a Java environment that provides the following:

- a runtime infrastructure for hosting applications,
- a set of Java extension APIs to build applications.

**Objectives of J2EE**

**J2EE architecture**

**J2EE container**

**J2EE application**

**Java Server Pages**

**Example implementations**

- JBoss: Open Source
- IBM Websphere: proprietary

J2EE is continuously extended by new technologies, e.g., integrating the support for Web Services.

The idea of J2EE is to provide a standardized programming model for the realization of distributed applications at the organizational level.

Java-based, but with interfaces to legacy applications, for example through Corba, component-based.

network-oriented: supporting Web Services.

**J2EE consists of 2 components**

- a runtime infrastructure for applications.
- a set of **Java extension APIs** to build applications. Examples are Enterprise Java Beans (EJB), Java Servlets, JavaServer Pages (JSP), RMI via Internet-Inter-ORB Protocol (RMI-IIOP), Java Naming and Directory Interface (JNDI), Java Transaction API and Java Mail.
A J2EE platform consists of the J2EE application server (runtime environment), one or several J2EE containers, and the database storage.

A typical J2EE platform has one or several containers. A J2EE container has two principal tasks: runtime environment for managing application components, to provide access to J2EE APIs.

Available APIs of the J2EE platform:
- RMI/IIOP: Remote Method Invocation (via IIOP)
- JNDI: Java Naming and Directory Interface
- JTA: Java Transaction API
- JDB: Java Database Connectivity Extension
- JMS: Java Message Service
- Java Mail
- JAF: Java Beans Activation Framework

Examples for application components: Java Servlets, Java Server Pages, Enterprise JavaBeans.

J2EE supports the following general containers:
- Web container: Java Servlets, JSP pages
- EJB container: Enterprise Java Bean components
- Applet container: Java applets
- Application container: Standard Java applications

A J2EE application consists of several modules, each of which again contains several application components. Modules and application components are listed in an archive file:
- EAR (Enterprise archive)
- WAR (Web archive)
- JAR (Java archive)

[Diagram of J2EE application structure]
JavaServer Pages technology uses XML-like tags and scriptlets written in the Java programming language to encapsulate the logic that generates the content for the Web page.

```html
<%-- Comment: <%-- Comment --%>
Declaration: <%! int x = 0; %>  
Expression: <%= expression %>
```

Scriptlets <contain Java Code

```html
<% code fragments %>
  <% if (value.getName().length != 0) { %>
    <h2>The value is: <%= value.getName()%></h2>
  <% } else { %>
    <h2>Value is empty</h2>
  <% } %>
```

Implicit objects available to JSP:

- request
- response
- session
- out
- page

The J2EE platform (now called Java Platform, Enterprise Edition - Java EE) is essentially a distributed application server environment. It is a Java environment that provides the following:

- a runtime infrastructure for hosting applications,
- a set of Java extension APIs to build applications.

**Objectives of J2EE**

- **J2EE architecture**
- **J2EE container**
- **J2EE application**

**Java Server Pages**

**Example implementations**

- **JBoss**: Open Source advanced middleware for J2EE based distributed applications
- **IBM WebSphere**: proprietary integration and application infrastructure software; provides J2EE support

J2EE is continuously extended by new technologies, e.g., integrating the support for Web Services.

**Issues**

This section focuses on the following issues:

- Discussion of basic aspects of distributed systems.
- Transparency as a key concept of distributed systems.
- How do distributed components cooperate? Thus, we discuss models of cooperation among components of distributed applications.

What is the client-server model?

**System Models**

- Transparency

**Paradigms for distributed applications**

- **Client-server model**

**Middleware**

Middleware is defined as a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers.

- hides the complexity of the communication between two or more systems or services.

Examples are CORBA, Java RMI, DCOM (Microsoft's Distributed Component Object Model).

Middleware services are e.g.: communication facilities, naming of remote entities (objects), persistence (distributed file system), distributed transactions, facilities for security.
A distributed system can be described in form of descriptive models.

**Architectural model**
- defines the interaction between components and the mapping onto the underlying network.

**Software layers**

**System architectures**

**Interaction model**

**Failure model**

**Security model**

The following sections of the course will discuss in more detail various aspects of these system models.

---

Deals with the placement of components across a network of computers and the functional roles they assume during interaction.

- client-server model.
- proxy servers.
- peer processes.
- community of software agents.

Pattern of distributing data and workload:

- Everybody is the same
  - plays the same role
  - (for SW agents)
  - (improve performance, increase availability)