INTRODUCTION TO JAVA EE AND ENTERPRISE DEVELOPMENT

OBJECT ORIENTED PROGRAMMING I

Sercan Külcü | Object Oriented Programming I | 10.01.2023

Contents

Introduction

Java Enterprise Edition (Java EE), formerly known as Java 2 Enterprise Edition (J2EE), is a set of specifications and APIs for building enterprise-class applications using Java. Java EE includes a range of APIs for developing web, enterprise, and distributed applications.

Java EE Architecture

The Java EE architecture consists of a set of services and APIs for building enterprise applications. The Java EE architecture is based on the following key components:

Java Servlets: Java Servlets provide a standard way to generate dynamic web content using Java.

JavaServer Pages (JSP): JavaServer Pages provide a way to generate dynamic web pages using Java code.

JavaServer Faces (JSF): JavaServer Faces is a framework for building user interfaces for web applications.

Enterprise JavaBeans (EJB): Enterprise JavaBeans provide a way to encapsulate business logic in a server-side component.

Java Message Service (JMS): Java Message Service is a standard way to send and receive messages between distributed applications.

Java Persistence API (JPA): Java Persistence API provides a standard way to access and manipulate data in a database.

Java Transaction API (JTA): Java Transaction API provides a standard way to manage transactions in a distributed environment.

Java EE Containers

A Java EE container is a runtime environment that provides support for the deployment and execution of Java EE applications. The Java EE container provides a range of services, including:

Component management: The container manages the lifecycle of components, such as Servlets, EJBs, and JSPs.

Resource management: The container provides access to resources, such as databases and JMS queues.

Security management: The container provides security features, such as authentication and authorization.

Transaction management: The container provides transaction management features, such as rollback and commit.

Java EE Application Development

Developing Java EE applications typically involves the following steps:

Design the application: Define the requirements and design the application architecture.

Choose the development environment: Choose a development environment, such as Eclipse or IntelliJ IDEA, and install the necessary plugins.

Develop the application: Write the code for the application, using the appropriate Java EE APIs and frameworks.

Test the application: Test the application using unit tests, integration tests, and system tests.

Deploy the application: Deploy the application to a Java EE container, such as Apache Tomcat or JBoss.

Best Practices for Java EE Development

When developing Java EE applications, it's important to follow best practices to ensure that your application is efficient, secure, and reliable.

Here are some best practices for Java EE development:

Use design patterns and frameworks to promote modular and maintainable code.

Use dependency injection to promote loose coupling and testability.

Use exception handling and logging to promote robustness and reliability.

Follow security best practices, such as using secure communication protocols and validating user input.

Conclusion

Java EE provides a powerful and flexible platform for developing enterprise applications using Java. By following best practices and using the Java EE APIs and frameworks judiciously, you can build robust, scalable, and efficient enterprise applications in Java.