ADVANCE JAVA

The dictionary meaning of **advance** is a forward movement or a development or improvement and the meaning of improve means thing that makes something better. All in all, we have to improve our basic knowledge to master in that particular field.

[Java](https://www.javatpoint.com/java-tutorial) is divided into two parts i.e. **Core Java (J2SE)** and **Advanced Java (JEE)**. The core Java part covers the fundamentals (data types, functions, operators, loops, thread, exception handling, etc.) of the Java programming language. It is used to develop general purpose applications. Whereas **Advanced Java** covers the standard concepts such as database connectivity, networking, Servlet, web-services, etc. In this section, we will discuss **what is advance Java, its benefit, uses, topics of advance Java**, and the **difference between core Java and advance Java.**

Advance Java

It is a part of Java programming language. It is an advanced technology or advance version of Java specially designed to develop web-based, network-centric or enterprise applications. It includes the concepts like [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), JDBC, [RMI](https://www.javatpoint.com/RMI), [Socket programming](https://www.javatpoint.com/socket-programming), etc. It is a specialization in specific domain.

Most of the applications developed using advance Java uses tow-tier architecture i.e. Client and Server. All the applications that runs on Server can be considered as advance Java applications.

Why advance Java?

* It simplifies the complexity of a building n-tier application.
* Standardizes and API between components and application sever container.
* JEE application Server and Containers provides the framework services.

Benefits of Advance Java

The four major benefits of advance Java that are, network centric, process simplification, and futuristic imaging standard.

* JEE (advance Java) provides libraries to understand the concept of **Client-Server architecture** for web- based applications.
* We can also work with web and application servers such as **Apache Tomcat** and **Glassfish** Using these servers, we can understand the working of HTTP protocol. It cannot be done in core Java.
* It is also important understand the advance Java if you are dealing with trading technologies like **Hadoop, cloud-native** and **data science**.
* It provides a set of services, **API** and **protocols**, that provides the functionality which is necessary for developing **multi-tiered** application, web-based application.
* There is a number of advance Java frameworks like, **Spring, Hibernate, Struts,** that enables us to develop secure **transaction-based** web applications such as banking application, inventory management application.

Difference between Core Java and Advance Java

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Core Java** | **Advance Java** |
| **Used for** | It is used to develop general purpose application. | It is used to develop web-based applications. |
| **Purpose** | It does not deal with database, socket programming, etc. | It deals with socket programming, DOM, and networking applications. |
| **Architecture** | It is a single tier architecture. | It is a mute-tier architecture. |
| **Edition** | It is a Java Standard Edition. | It is a Java Enterprise Edition. |
| **Package** | It provides java.lang.\* package. | It provides java.servlet.\* package. |

Advance Java Topics or Syllabus

**1. Basics of a Web application**

* What is a web application?
* What is a web client and web server?
* How do client and server communicate?
* HTTP protocol basics
* HTML language basics
* What is a TCP/IP port, URL?
* Need for a Web Container

**2. Web Container and Web Application Project Set up**

* To set up Tomcat Container on a machine
* To set up a Servlets JSP project in Eclipse
* To configure dependency of Servlet JSP APIs
* Web application project structure

**3. Servlets**

* What are Servlets?
* What can they do? Why are they needed?
* How do Servlets look in code?
* HTTP Methods; GET, POST, PUT, DELETE, TRACE, OPTIONS
* GET/POST request; differences between the two
* Servlet Lifecycle
* Servlet Context and Servlet Config
* Forwarding and Redirection of requests

**4. Session Management**

* What is a session?
* Why is it required?
* How to get a session?
* Session information passing between client and server
* Session information passing mechanisms - Cookies, Rewriting
* How to destroy a session

**5. JSPs**

* Introduction to JSP and need for JSPs
* Basic HTML tags
* JSP Lifecycle

**6. JSP Elements**

* Scriptlets
* Expressions
* Declarations
* Significance of above elements and fitment into the JSP Lifecycle
* What are Directives in JSP?
* Page Directive
* Include Directives
* Taglib Directive

**7. JSP Tag library**

* JSP Standard Actions
* Expression Language
* JSTL basics and it's usage
* Need for Custom Tag Library
* Custom Tag Library implementation

Struts Framework (version 2.x)

**1. Basics of MVC**

* What is MVC?
* MVC Type1 and Type2 architecture
* Why Struts framework?
* Struts 1 overview
* Struts 1 and Struts 2 comparison

**2. Struts 2 Architecture**

* Architecture Diagram explanation of following components:
* Components of Model, Views and Controller in Struts Framework
* Interceptors
* Model/Action classes
* Value Stack
* OGNL
* Introduction to configurations; framework and application architecture
* Declarative and Annotations configuration approaches

**3. Struts 2 set up and first Action class**

* Download JAR files
* Struts 2 project build up and Configuration files
* To build Action class
* To intercept an HTTP request via Struts2 framework using Action class
* Defining data and business logic in Action class
* Preparing and Forwarding control to Views

**4. Struts 2 Interceptors**

* What are Interceptors
* Responsibilities of an Interceptor
* Mechanism of Interceptor calling in Struts 2
* Defining Interceptors
* Defining Interceptor stacks
* Defining Custom Interceptors

**5. Struts 2 Tag Library**

* Introduction to tag library of Struts 2 and it's usage

**6. Struts 2 Validations**

* Validations using Validateable interface
* Workflow interceptor mechanism for validations
* Validations using Validateable interface
* Validation Framework introduction and architecture
* Validating user input with above two mechanisms

**7. Struts 2 Tiles Frameworks**

* Introduction to Tiles in a page
* Struts2 Tiles framework introduction
* Defining tiles.xml file
* Configuring pages for tiles
* A complete Tiles example with Struts2

Hibernate Framework (version 3.x)

**1. Introduction**

* What is ORM principle?
* Why ORM?
* ORM implementations

**2. Hibernate Architecture**

* Introduction to Hibernate
* Hibernate Architecture
* What are Persistent classes?

**3. Hibernate CRUD**

* Setting up Hibernate project
* Configuring all JARs and XML files
* Setting up connection to DB using Hibernate
* Performing basic CRUD operations using Hibernate API
* Object Identity; Generator type classes
* Using SQL with Hibernate
* Using HQL
* Using Criteria queries

**4. Mapping Collections and Associations**

* To define sets, mas, lists in Hibernate
* Association Mappings:
  1. One to one
  2. One to many
  3. Many to one
  4. Many to many
* Hibernate Caching
* What is caching?
* What are the types of caching in Hibernate?
* Explanation of various caching mechanisms in Hibernate

**5. Using Hibernate Annotations**

* Sample example of using Hibernate Annotations

Spring Framework (version 3.x)

**1. Introduction to spring**

* What is Spring?
* Spring Architecture explanation and all it's components

**2. Introduction to all modules of Spring**

* Spring Bean Factory
* Spring Application Context
* Spring DI
* Spring Integration; Spring messaging, Spring JMS
* Spring MVC
* Spring DAO

**3. Setting up spring**

* Setting up of Spring framework
* Download JARs
* Configure XML files

**4. Dependency Injection**

* What is Dependency Injection?
* How is it implemented using Spring Framework?
* Bean Wiring mechanisms in Spring

**5. Spring AOP**

* What is Spring AOP?
* Implementation of Spring AOP

Spring Boot Framework (Version 2.x)

**1. Introduction**

* Spring Boot Introduction
* Spring Boot Version
* Spring vs Spring Boot vs Spring MVC
* Spring Boot Architecture

**2. Creating Project**

* Spring Initializr
* Download & Install STS IDE
* Spring Boot Example
* Spring Boot CLI
* Spring Boot Example-STS

**3. Project Components**

* Annotations
* Dependency Management
* Application Properties
* Starters
* Starter Parent
* Starter Web
* Starter Data JPA
* Starter Actuator
* Starter Test
* Devtools
* Multi Module Project
* Packaging
* Auto-Configuration

**4. Tool Suite**

* Hello World Example
* Project Deployment Using Tomcat

**5. Spring Boot AOP**

* What is AOP?
* AOP Before Advice
* AOP After Advice
* AOP Around Advice
* After Returning Advice
* After Throwing Advice

**6. Spring Boot Database**

* JPA
* JDBC
* H2 Database
* Crud Operations

**7. Spring Boot View**

* Thymeleaf View

**8. Spring Boot Caching**

* What is Caching?
* Cache Provider
* EhCaching

**9. Spring Boot Misc**

* Run Spring Boot Application
* Changing Port
* Spring Boot Rest Example

**Web Services: REST and SOAP**

* Logging Framework: Splunk, Log4J, SLF4j
* Version-control system + repository hosting service: Git + Github