

Java Web Application Development Programme

Java, browser interfaces, Jakarta web technologies, persistence and Spring

FA-0790

Software Development · DevOps, Cloud & Infrastructure · Data & Analytics

Introduction

Develop foundations for Java web applications across core Java, browser interfaces, server-side web technologies, persistence and Spring. The modular programme also introduces dashboard communication and cloud services, with an application-project block. Use a mutually compatible supported JDK, Spring, Hibernate and servlet environment; older Java11/JavaEE examples serve as migration background, not a compatible baseline for current Spring.

Learning outcomes

- Write core Java and JDBC examples using object-oriented concepts, collections, exceptions and I/O.
- Build browser interfaces using HTML, CSS and JavaScript and apply introductory UX/dashboard principles.
- Use JSP/Servlet concepts in a compatible Jakarta web environment.
- Map relational data with Hibernate/JPA and apply Spring transaction and MVC concepts.
- Create layered Spring Boot applications with REST, security, logging and actuator fundamentals.
- Describe cloud services, identity/access and database choices and apply learning in a supervised project.

Prerequisites

Basic computer and file-handling skills and a prepared development environment. Prior programming and database familiarity are helpful for the combined programme. Current Spring Boot4 requires Java17 or later and compatible Spring7/Servlet6.1components; use the selected lab's tested dependency versions rather than mixing older Java11/JavaEE libraries.

Training outline

1. Core Java — 4-day module

Java technology overview

- Components that make up the Java programming language
- Java Features
- Java development tools
- Different deployment architectures
- Java language syntax
- Construct statements in Java

Compiling and Running a Simple Program

- Java Platform
- Setting Up Your Computer
- Writing a Program
- Compiling the Program
- Interpreting and Running the Program

Basics I: Flow Control in Java

Decision making flows

- if-else blocks
- switch-case blocks

Iterative flows

- while loop
- do-while loop
- for loop

Java development environment

- Java runtime environment and modern JDK distribution
- JDK
- JVM

- Classpath
- Path

Object Oriented Programming using Java

- Class, method, properties
- Creating Objects
- Constructors
- Constructor Overloading
- Method Overloading
- The this keyword
- Packages
- Access specifiers
- import keyword
- Life Cycle of an Object
- Java memory management
- Garbage Collection

Inheritance and Polymorphism

- Inheritance in Java
- Inheriting classes
- Overriding methods
- Creating Abstract classes, interfaces and methods

Other Class Features

- Create static variables, methods, and initializers
- Create final classes, methods, and variables
- Create and use enumerated types
- Use the static import statement
- Arrays of primitives
- Arrays of objects
- Enhanced for loop

Classes in Java

- Object class
- Overview of java.lang and java.util packages
- String class
- Date class
- Calendar class
- StringBuffer class
- Using string classes

- Scanner

Exceptions / Error Handling

- Define exceptions
- Use try, catch, throw, throws and finally statements
- Describe exception categories
- User Defined exceptions

Collections and Generics Framework

- Collections framework
- Legacy collection classes
- Comparable and Comparator interfaces
- Use generic collections
- Use type parameters in generic classes

I/O Fundamentals

- Command-line arguments
- Properties class
- Serialize and deserialize objects
- Distinguish readers and writers from streams
- Describe files and file I/O

Java Database Connectivity (JDBC)

- Introduction to JDBC API
- Understanding JDBC Interfaces

- Connection
- Statement
- PreparedStatement
- CallableStatement

ResultSet

- Create, Read, Update & Delete Queries
- Fetching of results
- Assessment

Date/Calendar and legacy collections are compatibility context; current date/time APIs are compared where appropriate. Use parameterised JDBC queries and carefully controlled serialization examples.

2. HTML, CSS, JavaScript and UI/UX – 5-day module

HTML5 Overview

- History of HTML5

- The HTML5 vision
- HTML living-standard specifications and their development
- What is part of HTML5?

Using HTML5 Today

- When can I use these features?

Using HTML5 in browsers that do not support it

- Detecting native availability of HTML5 features
- Emulation

HTML5 Markup

- HTML5 page structure
- HTML5 DOCTYPE

HTML5 markup

- Structural elements
- Semantic elements
- Deprecated elements

HTML5 Forms

- HTML5 form elements
- Building and using HTML5 forms

HTML5 Audio and Video

- The audio and video elements
- Understanding audio and video
- Audio and video containers
- Audio and video codecs

HTML5 Geo-location

- Geo-location overview
- User Privacy
- Location information sources

CSS3

- CSS selectors
- Borders
- Rounded corners
- Drop shadows
- Backgrounds
- Text effects
- Adding shadows to text
- Colours and Opacity

- RGBA and HSL values and alpha channel opacity
- Setting opacity
- Transitions and transforms
- Using transitions
- Using transforms
- UI Properties
- Appearance
- Box-sizing
- Resize
- Outline-offset
- Generated Content Properties
- Content
- Media Queries
- HTML and a compatible supported jQuery version

HTML5 Canvas and SVG

Overview

- Canvas vs. SVG
- Canvas coordinates
- Context
- Pixel data

HTML5 Storage

Overview

- Local Storage
- Session Storage

HTML5 IndexedDB

Overview

- ObjectStore
- Transaction
- CRUD Operations

HTML5 Offline Web Applications

Offline Applications Overview

- Service worker and cache strategy; historical AppCache manifest comparison
- Online and Offline Events

JavaScript

- Introduction to JavaScript
- Data Types

- JavaScript dynamic typing and typeof operator
- Variables
- Operators
- == and === difference
- parsers
- Conditional/ Looping Constructs
- Arrays
- JSON
- String, Date, Number, Math objects
- undefined, null and NaN distinctions

Functions :

- Syntax
- Arguments
- Return values
- Variable scope
- Named functions
- Function expression
- Closures

JavaScript Built-In Objects

- window
- document
- history
- location
- navigator
- Regular Expressions
- JavaScript Http Requests

Web Storage

- Local
- Session
- Cookies

DOM Traversal

- querySelector
- querySelectorAll

HTML uses a living standard and feature detection. Offline application exercises use supported service-worker/cache approaches rather than an AppCache manifest.

UX and dashboard communication

- UX Fundamentals

Implementation strategies for better user experience;

- Prioritization Matrix
- Define centralized Metrics
- Test and Scalability

- Principles of human perception in relation to different visualizations

Data Classification and Representation

- Data Classification
- Encoding with examples
- Encoding with colors and order
- Designer Guidelines

- Principles of user psychology and Behavior

- Dashboarding and communication

Compare current compatible Cognos Analytics, Power BI and Qlik capabilities

- UI Best Practise across BI Tools
- Hands-on Power BI or Qlik
- Collaboration and communication with multiple teams spread across

- Advanced-chart and extension examples in the selected BI environment; legacy Cognos10/11 context

Actionable ways for improving existing dashboards

- Will be done with Instructor Lab Examples

BI products are compared using compatible available versions; hands-on work selects Power BI or Qlik rather than mastering every named product/version.

3. JSP and Servlets — 3-day module

- JSP & Servlets

JavaEE history and current JakartaEE web concepts

- Review Web Development Concepts
- Review History of Server-Side Development

Creating JSP Pages

- Basic syntax
- Using JSTL
- Using Unified Expression Language (EL)

Processing JSP Pages

- Request/response processing
- Validating, transforming, encapsulating and manipulating data
- Managing application and user state data with Sessions and Cookies
- Handling application and HTTP errors

Design Patterns

- Model-View-Controller
- Front Controller
- Business Delegate

Servlets

- Creation and usage
- Filters, Listeners, and Servlet Mappings
- Asynchronous servlets
- Web service endpoint

Web Application Security

- JSON-P
- Testing and Validating Web Application
- Packaging for Production Deployment

Current Jakarta namespaces, container versions and template/tag-library versions must match. JavaEE naming is historical migration context. JSON-P is a JSON-processing API, not an authentication control.

4. Hibernate and Spring — 4-day module

Introduction to Hibernate

- Benefits of Hibernate as ORM Framework
- Architecture of Hibernate Framework
- Understanding Persistence lifecycle Getting started with Hibernate ORM quickly
- Mapping tables to beans
- Mapping properties to columns
- Setting up the user POJO object
- Create a simple CRUD Application using Hibernate ORM Relationships
- Simple Association (one to one)
- Basic Collection mapping (one-to-many)
- Simple unidirectional mapping (many-to-one)
- Collection (many to many)
- Relationship mapping examples: one-to-one, one-to-many, many-to-one and many-to-many
- Working with queries

- Named queries
- Examples: Using queries and named queries Transaction Support • Local transaction, global transaction
- Transaction API
- Isolation levels
- Examples: Using Transaction API
- Locking Examples
- Examples: Implementing optimistic locking with versioning

More mapping concepts

- Type system
- Mapping collections of value types
- Three ways to map inheritance
- Examples: Re-implementing roles as a typed collection
- Examples: Re-implementing Employee as a separate table
- Working with compound keys
- Examples: Working with compound keys Advanced Queries
- Working with the Query API
- Working with Binding Parameters
- Grouping
- Sub-queries
- Optimizing queries
- Native SQL queries
- Examples: Using advanced query support
- Examples: Using native SQL query support Spring support for ORM and Hibernate ORM Framework
- Spring Introduction
- How spring simplifies Hibernate ORM framework development
- Legacy HibernateTemplate comparison and current Hibernate/JPA integration
- Using spring DAO support
- Using spring's transaction support
- Using spring DAOs
- Using declarative transactions
- Examples: Spring DAO Injection Caching Support
- Caching Overview
- Caching Architecture
- API to control cache

- Compatible cache-provider examples, including Ehcache

Spring Web MVC

- Overview
- Creating Spring Web MVC project
- Configure JSP, JSTL and Thymeleaf
- The Model object
- The ModelAndView object
- The ModelAndView object
- Read URI path variables
- Read query string parameters
- Read Form Data with @ModelAttribute
- JSON

Aspect Oriented Programming

- Overview
- Decoupling Repositories from Controllers
- AspectJ Dependencies and .pom / Spring Bean Configuration
- Working with Spring Profiles

Spring AOP Hands-on

- Aspect, Advice,
 - Pointcut,
 - JoinPoint,
 - Annotations,
 - XML Configuration
- Best Practices for Logging

Use current compatible Hibernate/JPA and Spring integration. Older HibernateTemplate/DAO examples are legacy context; choose current session/EntityManager and declarative transaction approaches for the prepared exercises.

5. Spring Boot fundamentals — 5-day module

- Project creation using supported Spring tooling/Initializr or a compatible CLI.
- POM, starters, application entry point and test structure.
- Auto-configuration, properties/YAML, external/custom settings and multiple configuration files.
- Embedded server choices and version compatibility.
- Data access with Spring Data JPA/Hibernate; service and REST controller layers.
- Spring Security integration, logging and actuator health/metrics.

6. Cloud technical essentials — 1-day module

- Introduction and History of AWS, Azure, GCP, DO
- Foundational Services
- Security, Identity, and Access Management
- Databases: RDS, DynamoDB
- Management Tools

Compare provider-specific services and responsibility boundaries at an introductory level.

7. Application project — 2-day block

Apply the programme's development concepts in a supervised application project.