ORACLE Workforce Development Partner

DIPLOMADO
Java SE 8 & JEE7 Developer

Curso Oficial ORACLE University

PRESENTACIÓN

BBVA Next Technologies, se encuentra en constante crecimiento y actualización de conocimiento en los talentos contratados, por lo cual se ha visto en la necesidad de promover el servicio de capacitación en las distintas tecnologías que la operación demanda. El esquema que ha promovido internamente bajo sus políticas establecidas se basa en el apoyo económico que recibe cada profesional de TI, no solo para las áreas Desarrollo y programación de Aplicaciones JAVA sino áreas de soft skills entre otras. La necesidad de contar con personal certificado provee una mayor oportunidad de atraer nuevos negocios por lo que proponemos un sistema de entrenamiento oficial en el que se promueva sistemáticamente un path de curso a cumplir para mantener a las unidades de trabajo optimas en la necesidad de cubrir los requerimientos de los clientes finales.

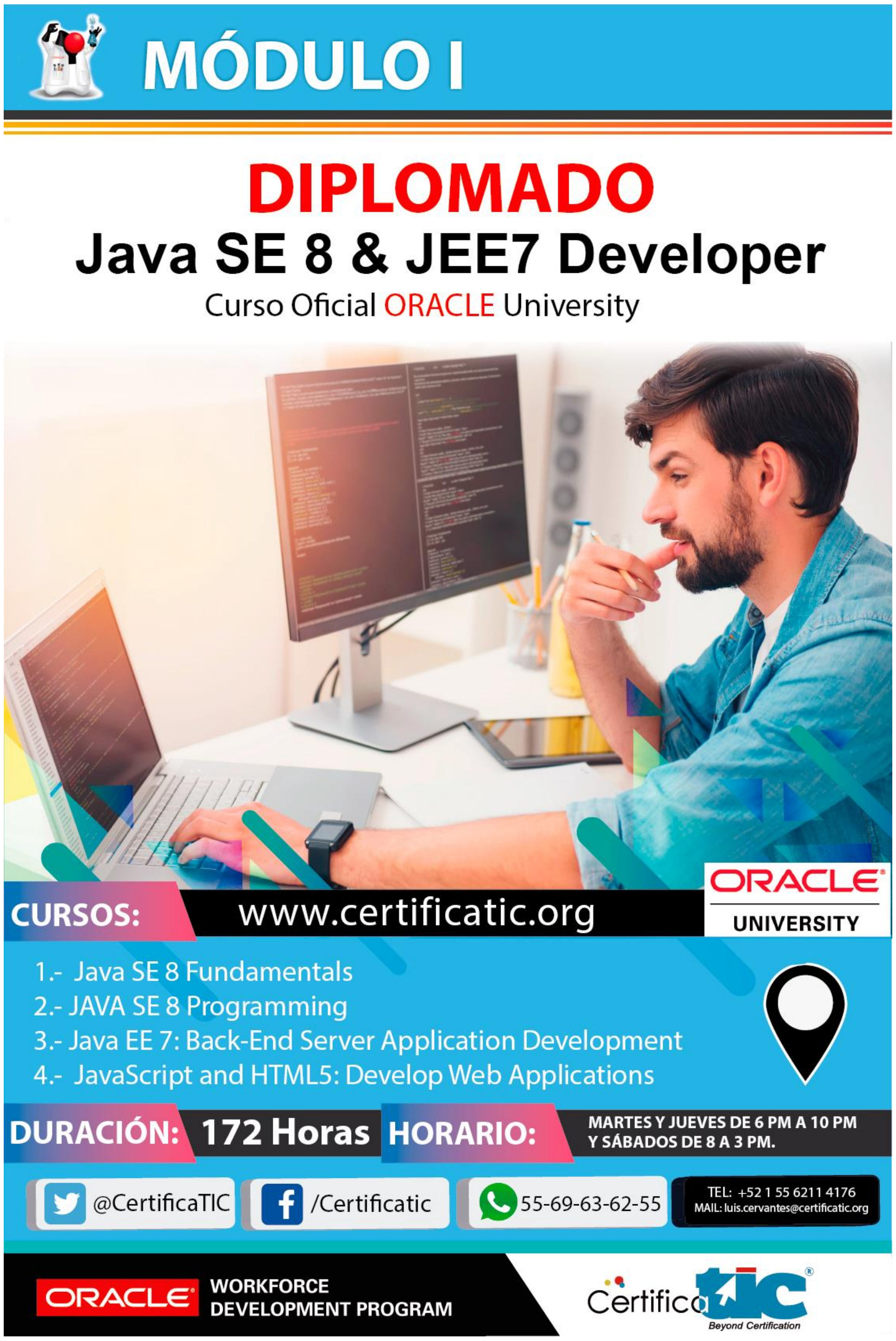
Por tal motivo Organización Educativa CertificaTIC, S.C. bajo sus regulaciones permite la entrega de capacitación oficial certificada de TI en Oracle a personas físicas por lo que expide la presente propuesta en un esquema de delivery en varias semanas:

OBJETIVO ACADÉMICO

El participante podrá adquirir los conocimientos necesarios para certificarse en tres Niveles bajo la estructura de dos Módulos. La capacitación se realizará en sitio en instalaciones del cliente en formato entre semana y sabatino de acuerdo a la logística interna dentro de una cadena de curso establecidos en el Path de Diplomado.

Workforce
Development Partner

MODULO I -172 Horas



MÓDULO I

DIPLOMADO
Java SE 8 & JEE7 Developer
Curso Oficial ORACLE University

CURSOS: www.certificatic.org

ORACLE UNIVERSITY

1.- Java SE 8 Fundamentals
2.- JAVA SE 8 Programming
3.- Java EE 7: Back-End Server Application Development
4.- JavaScript and HTML5: Develop Web Applications

DURACIÓN: 172 Horas HORARIO: MARTES Y JUEVES DE 6 PM A 10 PM Y SÁBADOS DE 8 A 3 PM.

@CertificaTIC /Certificatic 55-69-63-62-55 TEL: +52 1 55 6211 4176 MAIL: luis.cervantes@certificatic.org

ORACLE WORKFORCE DEVELOPMENT PROGRAM

CertificaTIC Beyond Certification

Organización Educativa CertificaTIC S.C.
RFC: OEC140529BK9
Tel: +52 (55) 67232060 | WhatsApp: +52 1 5569636255| contacto@certificatic.org
www.certificatic.org

ORACLE UNIVERSITY

Workforce
Development Partner

MODULO II -142 Horas

MÓDULO II

DIPLOMADO
Java SE 8 & JEE7 Developer
Curso Oficial ORACLE University

CURSOS: www.certificatic.org

UNIVERSITY

5.- Java EE 6: Develop Web Services with JAX-WS & JAX-RS
6.- Java Design Patterns
7.- Spring FrameWork Core 4

DURACIÓN: 142 Horas **HORARIO:** MARTES Y JUEVES DE 6 PM A 10 PM Y SÁBADOS DE 8 A 3 PM.

@CertificaTIC | /Certificatic | 55-69-63-62-55 | TEL: +52 1 55 6211 4176 MAIL: luis.cervantes@certificatic.org

WORKFORCE DEVELOPMENT PROGRAM

Beyond Certification

DURACIÓN: 314 Horas

REQUISITOS ACADÉMICOS

❑ Egresados de cualquier carrera de Tecnología de Informática interesados en desarrollarse como programadores y desarrolladores de software.

INCLUYE:

Material oficial de Oracle University * (eKit´s) de los siguientes cursos:

MODULO I. Curso presencial de 172 horas.

- Java SE 8 Fundamentals*
- JAVA SE 8 Programming*
- Java EE 7: Back-End Server Application Development*
- JavaScript and HTML5: Develop Web Applications*

MAS 2 Voucher de certificación por persona con vigencia de 6 meses para presentar examen:

1. Oracle Certified Associate, Java SE 8 Programmer Certification. Oracle Certified.
2. Professional, Java SE 8 Programmer Certification.

- Evaluación Inicial y Evaluación Final

MODULO II. Curso presencial de 142 horas.

- Java EE 6: Develop Web Services with JAX-WS & JAX-RS*
- Java Design Patterns*
- Spring FrameWork Core 4

MAS 1 Voucher de certificación por persona con vigencia de 6 meses para presentar examen:

3. Oracle Certified Expert, Java EE 6 Web Services Developer Certification.

- Evaluación Inicial y Evaluación Final

Workforce Development Partner

LAP TOP

1. Procesador: Intel(R) Core (TM) i5 o un procesador equivalente
2. Mínimo de memoria RAM: 8 GB
3. Mínimo de espacio en Disco 150 GB disponibles.

Conoce a tu Ethien Salinas: https://youtu.be/_7bIhg0YYiU

PROGRAMA (Continuar abajo)

MÓDULO I

DIPLOMADO

Java SE 8 & JEE7 Developer

Curso Oficial ORACLE University

Java SE 8 Fundamentals

Duration: 5 Days

What you will learn

This Java SE 8 Fundamentals training introduces you to object-oriented programming using the Java language. Through hands-on exercises, you'll begin to build a baseline of knowledge to propel your career in development.

Learn To:

Use Java programming language constructs to create a Java technology application.
Use decision and looping constructs and methods to dictate program flow.
Understand basic object oriented concepts such as inheritance, encapsulation, and abstraction.
Use and manipulate object references, and to write simple error handling code.
Use the new SE 8 java.time and java.time.format packages to format and print the local date and time.
Specify a data modification by passing a predicate lambda expression to the Collections class.

Benefits to You
By enrolling in this course, you'll expand your knowledge of Java SE 8, while building your Java skill set. You'll build a solid basis in the Java programming language upon which to base continued work and training.

Audience

Application Developers
Developer
Project Manager
System Administrator
Team Leader
Technical Administrator
Technical Consultant
Web Administrator

Course Objectives

Write Java code that uses variables, arrays, conditional and loop constructs

Manipulate primitive numeric data and string data using Java operators

Create Java classes and use object references

Access the fields and methods of an object

Manipulate text data using the methods of the String and StringBuilder classes

Use casting without losing precision or causing errors

Declare, override, and invoke methods

Access and create static fields and methods

Use classes from the java.time and java.time.format packages to format and print the local date and time

Encapsulate a class using access modifiers and overloaded constructors

Define and implement a simple class hierarchy

Demonstrate polymorphism by implementing a Java Interface

Use a Predicate Lambda expression as the argument to a method

Handle a checked exception in a Java application

Course Topics

What Is a Java Program?
Introduction to Computer Programs
Key Features of the Java Language
The Java Technology and Development Environment
Running/testing a Java program

Creating a Java Main Class
Java Classes
The main Method

Data In the Cart
Introducing variables
Working with Strings
Working with numbers
Manipulating numeric data

Managing Multiple Items
Working with Conditions
Working with a List of Items
Processing a list of items

Describing Objects and Classes
Working with objects and classes
Defining fields and methods
Declaring, Instantiating, and Initializing Objects
Working with Object References
Doing more with Arrays
Introducing the NetBeans IDE
Introducing the Soccer League Use Case

Manipulating and Formatting the Data in Your Program
Using the String Class
Using the Java API Docs
Using the StringBuilder Class
More about primitive data types
The remaining numeric operators
Promoting and casting variables

Creating and Using Methods
Using methods
Method arguments and return values
Static methods and variables
How Arguments are Passed to a Method
Overloading a method

Using Encapsulation
Access Control
Encapsulation
Overloading constructors

More on Conditionals
Relational and conditional operators
More ways to use if/else constructs
Using Switch Statements
Using the NetBeans Debugger

More on Arrays and Loops
Working with Dates
Parsing the args Array
Two-dimensional Arrays
Alternate Looping Constructs
Nesting Loops
The ArrayList class

Using Inheritance
Overview of inheritance
Working with subclasses and superclasses
Overriding methods in the superclass
Introducing polymorphism
Creating and extending abstract classes

Using Interfaces
Polymorphism in the JDK foundation classes
Using Interfaces
Using the List Interface
Introducing Lambda expressions

Handling Exceptions
Handling Exceptions: An overview
Propagation of exceptions
Catching and throwing exceptions
Handling multiple exceptions and errors

Java SE 8 Programming

Duration: 5 Days

What you will learn

This Java SE 8 Programming training covers the core language features and Application Programming Interfaces (API) you will use to design object-oriented applications with Java Standard Edition 8 (Java SE 8) Platform.

Learn To:

Create Java technology applications with the latest JDK Technology
Develop your object-oriented skills
Identify good practices in the use of the language to create robust Java application
Use Lambda expressions in Java applications
Store and manipulate data using collections
Manipulate files, directories and file systems
Connect to databases using standard SQL queries through JDBC
Create high-performance multi-threaded applications

Benefits to You
You can use this course to further develop your skills with the Java language and prepare for the Oracle Certified Professional, Java SE 8 Programmer Exam!

Audience

Developer
Java Developers
Java EE Developers

Related Training

Required Prerequisites

Java SE 8 Fundamentals

Course Objectives

Creating high-performing multi-threaded applications

Creating Java technology applications that leverage the object-oriented features of the Java language, such as encapsulation, inheritance, and polymorphism

Implementing input/output (I/O) functionality to read from and write to data and text files and understand advanced I/O streams

Executing a Java technology application from the command line

Manipulating files, directories and file systems using the JDK NIO.2 specification

Creating applications that use the Java Collections framework

Performing multiple operations on database tables, including creating, reading, updating and deleting using both JDBC and JPA technology

Searching and filter collections using Lambda Expressions

Implementing error-handling techniques using exception handling

Using Lambda Expression concurrency features

Course Topics

Java Platform Overview
Defining how the Java language achieves platform independence
Differentiating between the Java ME, Java SE, and Java EE Platforms
Evaluating Java libraries, middle-ware, and database options
Defining how the Java language continues to evolve

Java Syntax and Class Review
Creating simple Java classes
Creating primitive variables
Using operators
Creating and manipulate strings
Using if-else and switch statements
Iterating with loops: while,do-while,for,enhanced for
Creating arrays
Using Java fields, constructors, and methods

Encapsulation and Subclassing
Using encapsulation in Java class design
Modeling business problems using Java classes
Making classes immutable
Creating and use Java subclasses
Overloading methods

Overriding Methods, Polymorphism, and Static Classes
Using access levels: private, protected, default, and public.
Overriding methods
Using virtual method invocation
Using varargs to specify variable arguments
Using the instanceof operator to compare object types
Using upward and downward casts
Modeling business problems by using the static keyword

Implementing the singleton design pattern

Abstract and Nested Classes
Designing general-purpose base classes by using abstract classes
Constructing abstract Java classes and subclasses
Applying final keyword in Java
Distinguish between top-level and nested classes

Interfaces and Lambda Expressions
Defining a Java interface
Choosing between interface inheritance and class inheritance
Extending an interface
Defaulting methods
Anonymous inner classes
Defining a Lambda Expression

Collections and Generics
Creating a custom generic class
Using the type inference diamond to create an object
Creating a collection by using generics
Implementing an ArrayList
Implementing a TreeSet
Implementing a HashMap
Implementing a Deque
Ordering collections

Collections Streams, and Filters
Describing the Builder pattern
Iterating through a collection using lambda syntax
Describing the Stream interface
Filtering a collection using lambda expressions
Calling an existing method using a method reference
Chaining multiple methods together
Defining pipelines in terms of lambdas and collections

Lambda Built-in Functional Interfaces
Listing the built-in interfaces included in java.util.function
Core interfaces - Predicate, Consumer, Function, Supplier
Using primitive versions of base interfaces
Using binary versions of base interfaces

Lambda Operations
Extracting data from an object using map
Describing the types of stream operations
Describing the Optional class
Describing lazy processing
Sorting a stream
Saving results to a collection using the collect method
Grouping and partition data using the Collectors class

Exceptions and Assertions
Defining the purpose of Java exceptions

Using the try and throw statements
Using the catch, multi-catch, and finally clauses
Autoclose resources with a try-with-resources statement
Recognizing common exception classes and categories
Creating custom exceptions
Testing invariants by using assertions

Java Date/Time API
Creating and manage date-based events
Creating and manage time-based events
Combining date and time into a single object
Working with dates and times across time zones
Managing changes resulting from daylight savings
Defining and create timestamps, periods and durations
Applying formatting to local and zoned dates and times

I/O Fundamentals
Describing the basics of input and output in Java
Read and write data from the console
Using streams to read and write files
Writing and read objects using serialization

File I/O (NIO.2)
Using the Path interface to operate on file and directory paths
Using the Files class to check, delete, copy, or move a file or directory
Using Stream API with NIO2

Concurrency
Describing operating system task scheduling
Creating worker threads using Runnable and Callable
Using an ExecutorService to concurrently execute tasks
Identifying potential threading problems
Using synchronized and concurrent atomic to manage atomicity
Using monitor locks to control the order of thread execution
Using the java.util.concurrent collections

The Fork-Join Framework
Parallelism
The need for Fork-Join
Work stealing
RecursiveTask
RecursiveTask

Parallel Streams
Reviewing the key characteristics of streams
Describing how to make a stream pipeline execute in parallel
List the key assumptions needed to use a parallel pipeline
Defining reduction
Describing why reduction requires an associative function
Calculating a value using reduce
Describing the process for decomposing and then merging work
Listing the key performance considerations for parallel streams

Database Applications with JDBC
Defining the layout of the JDBC API
Connecting to a database by using a JDBC driver
Submitting queries and get results from the database
Specifying JDBC driver information externally
Performing CRUD operations using the JDBC API

Localization
Describing the advantages of localizing an application
Defining what a locale represents
Read and set the locale by using the Locale object
Building a resource bundle for each locale
Calling a resource bundle from an application
Changing the locale for a resource bundle

Java EE 7: Back-End Server Application Development

Duration: 5 Days

What you will learn

The Java EE 7: Back-End Server Application Development training teaches you how to build and deploy enterprise applications that comply with Java Platform, Enterprise Edition 7 Full Profile. Learn to develop applications with the following technologies: Enterprise JavaBeans (EJB), Java Persistence API (JPA), JDBC, Java Transaction API (JTA), Contexts and Dependency Injection (CDI), Java Message Service (JMS), Bean Validation, Batch API, Timer services, Java EE Concurrency and more.

Learn To:

Use Java EE 7 technologies to create, read, update and delete database records using both JDBC and JPA technologies.
Create a flexible component model using EJB and CDI technology.
Create SOAP-based and XML web services.
Develop the business and integration tiers of an enterprise application.
Understand how those components responsible for: interacting with other systems through web services and message queues.
Become proficient with database access and manipulation using transactions.
Provide timer, concurrency and batch services.
Develop expertise using Java Enterprise Edition 7, the latest version of the Java platform for development of enterprise applications.

Benefits to You

When you walk away from this course, you will have developed the knowledge and skills to read and write messages to systems that may or may not be developed using Java with Java Message Service create batch services to process thousands of jobs in parallel. This interactive, hands-on training is an excellent follow-up course to the Java EE 7: Front-end Application Development training.

Audience

Application Developers
Developer
J2EE Developer
Java Developers
Java EE Developers
System Integrator

Related Training

Required Prerequisites

Understand OO principles

Basic understanding of database concepts and SQL syntax

Experience with Java SE

Java SE 8 Programming

Suggested Prerequisites

Java EE 7: Front-end Web Application Development

Java SE 7 or 8 programmer certification

Course Objectives

Apply dependency injection using CDI

Apply the batch API to the problem of processing thousands of jobs in parallel

Create and apply Timer services

Create and use web services in enterprise applications

Develop enterprise components using EJB

Use JDBC in an enterprise environment

Use JMS to communicate between various enterprise systems

Use JPA to persist entities and create, read, update and delete database records

Course Topics

Java Platform, Enterprise Edition

The Java EE Platform
The needs of enterprise application developers
Java EE specifications
A comparison of services and libraries
Java EE application tiers and architecture

Enterprise Development Tools and Applications

The purpose of an application server
Properties of Java EE components
The development process of Java EE applications
Configuring and deploying Java EE applications

Java Beans, Annotations and Logging

Java SE features in Java EE applications
Creating POJO JavaBeans components
Using logging
Using common Java annotations
Developing custom annotations
The role of annotations in Java EE applications

XML Programming with JAXB
The benefits of XML
XML namespaces and schemas
Java XML APIs
The Java XML Binding API (JAXB)
Reading and writing XML documents with JAXB
xjc: the JAXB binding compiler
JAXB annotations

SOAP Web Services with JAX-WS
Overview of SOAP
Overview of WSDL files
Comparing WSDL-first and code-first design approaches
Writing a JAX-WS web service
Generating WSDL from a Java class
Creating JAX-WS web service clients

Java Naming and Directory (JNDI) Services
What is JNDI?
Naming service concepts
Directory service concepts
JNDI packages
Using JNDI to look up JDBC and EJB components in Java EE

The EJB Component Model
The role EJB components play in Java EE appplications
The role of the EJB container
EJB changes in Java EE 7
Local, distributed and no-client EJB client access views
EJB Session types
Stateless, Stateful and Singleton EJBs
Session bean packaging and deploying

Contexts and Dependency Injection
What is dependency injection?
Using Qualifiers
The beans.xml file and Alternatives
Using Producers and Disposers
Using Interceptors
Using Events and Stereotypes

Java Message Service
What is the Java Message Service?
Why do we need JMS?
JMS Overview

Point-to-point messaging architecture
Publish/subscribe messaging architecture
Message producers and consumers
Queues and topics
Durable vs. non-durable subscriptions

Message-driven Beans
The life cycle of a message-driven bean
Creating a message-driven bean
Creating life cycle handlers for message-driven beans
Configuring a message-driven bean

Java EE Concurrency
Concurrency in Java EE
Asynchronous EJBs
Managed Executors

JDBC in Java EE Environments
Overview of the JDBC API
Using CDI to inject a JDBC resource in a Java EE component
The Data Access Object pattern

Transactions in Java EE Environments
What are transaction semantics?
Comparing programmatic and declarative transaction scoping
Using JTA to scope transactions programmatically
Implementing a container-managed transaction policy using declarations
Controlling container-managed transaction propagation

Java Persistence API
Object-relational mapping
Entities and the entity manager
Persistence contexts and persistence units
Create, read, update and delete operations with JPA
Create typed queries in JPA with JPQL

Bean Validation with JPA
What is Bean Validation?
JPA lifecycle phases where validation takes place
Using the built-in validation constraints
Creating a custom bean validation constraint
Programmatic validation by injecting a Validator
Using validation groups

Timer and Batch Services
What are timer services?
Programmatic and automatic timers
What is Batch processing?
Jobs, steps and chunks
Batch examples

Security

Authentication, authorization and confidentiality
Apply Java EE security using deployment descriptors
Creating users and groups and mapping them to roles
Defining possible web service attack vectors

JavaScript and HTML5: Develop Web Applications

Duration: 4 Days

What you will learn

This JavaScript and HTML5 course teaches you how to code application logic in web applications using JavaScript and how to create HTML5 pages to parse and send data using HTML5 forms. Create and modify the Document Object Model(DOM), create responsive layouts with CSS3, store local data with JSON, and draw on HTML5 canvas. Students will add interactive behaviors to web pages creating better user experiences and add dynamic data using AJAX, REST and WebSocket with JavaScript.

Learn To:

Code application logic using JavaScript to control user interactions and display data.
Create applications with HTML5 forms to send data to services.
Debug and inspect web applications and styles using browser's tools.
Create design templates and standards using CSS and JavaScript that adapt to different devices including mobile with Media Queries and Responsive Design.
Read and validate data from HTML5 forms using JavaScript.
Parse, modify, and validate data using Javascript API.
Add interactivity in HTML5 forms using events and DOM modification.
Store and send JavaScript Object data to services, local storage or across different pages and HTML5 elements using JavaScript Object Notation.
Draw on HTML5 canvas using JavaScript.
Store user data in web applications using HTML5 Local Storage
Create JavaScript code to retrieve and display dynamic data from REST services using AJAX.
Create JavaScript code to interact with WebSocket for real-time communication.
Create jQuery code to animate elements, handle DOM, events, or AJAX responses.

Benefits to You

This course will prepare any web developer with enough JavaScript, HTML5 and CSS3 knowledge to build complex and modern sites and for those looking to develop Java EE front-end web applications.

Audience

Application Developers
Developer
Forms Developer
J2EE Developer
Java Developers
Java EE Developers
Team Leader

Technical Consultant

Related Training

Required Prerequisites

Basic experience in any programming language

Basic knowledge of web concepts

Course Objectives

Create and run an HTML5 applications in NetBeans

Write JavaScript code to use variables, objects, functions and arrays

Create HTML5 forms to request information and process it

Write JavaScript functions for HTML5 events

Manipulate HTML5 elements through DOM

Use the JavaScript API

Store objects by using the JSON API, Cookies, and Local Storage

Style HTML documents with CSS3

Use Media Queries and media data to adapt the web page to different screen sizes

Create closures, prototypes, and modules in JavaScript

Create a Canvas, intervals, Drag and Drop interactions, and implement mouse gestures in HTML5

Use AJAX to consume RESTful Web Services

Identify the required Back-End technologies for REST and WebSocket with Java EE7

Use Selectors and DOM manipulators to handle documents with jQuery

Handle events and AJAX server responses with jQuery

Course Topics

Introduction
Knowing the objectives of the course
Setting up the Environment

Web Application Essentials

Creating HTML5 Applications in NetBeans
Running HTML pages and analizing them by using the browser's development tools
Separating CSS and JavaScript content from HTML pages
Running HTML5 Applications in NetBeans
Practice: Creating HTML5 Web Applications with NetBeans 8
Practice: Separating JavaScript and CSS Resources

JavaScript Fundamentals
Writing JavaScript code to declare variables, objects, functions and arrays
Writing JavaScript Arrays to store data
Defining JavaScript Objects as a key-value store
Accessing the properties of an object
Practice: Writing JavaScript code to pass tests in Jasmine

Combining HTML5 and JavaScript in Web Applications
Creating HTML5 Documents
Creating HTML5 Forms to request information and process it
Validating HTML5 form input
Writing JavaScript functions for HTML5 events
Manipulating HTML5 elements through DOM
Practice: Writing JavaScript code to modify document elements

The JavaScript API
Validating user input with JavaScript and Regular Expressions
Handling multiple values with JavaScript Collections
Manipulating Dates with the JavaScript Date API
Practice: Creating a meal-divider application
Practice: Calculating the total based on the age

Web Application Data
Converting Objects to JSON Strings
Parsing JSON Strings into JavaScript Objects
Storing Objects by using the JSON API, Cookies, and Local Storage
Practice: Saving user input using JSON and Local Storage
Practice: Restoring saved data when page loads

Style Applications using CSS3 and JavaScript
Applying CSS styles to HTML documents
Using CSS3 features to add dynamic styles to elements with events
Using Media Queries and media data to adapt to different screens
Using JavaScript to add and remove styles from elements
Practice: Writing CSS rules to style elements in the document

Advanced JavaScript
Defining Functions
Creating Closures and explaining Variable Scope
Writing JavaScript functions as modules
Creating Prototypes
Creating Drag-and-Drop interactions with JavaScript
Creating JavaScript Timers and Delays to create animations in HTML
Using the HTML5 Canvas Object to draw in pages
Practices: Creating a Canvas, intervals, Drag and Drop, and implementing Mouse Gestures

AJAX and WebSocket
Using AJAX with JavaScript to request data from an Application Server
Using AJAX to consume RESTful Web Services
Using AJAX calls to create "Server Push" interactions
Identifying alternatives to AJAX used in legacy code
Understanding AJAX Security
Using WebSocket to create Real-time Client/Server interactions
Identifying the required Back-End technologies for REST and WebSocket with Java EE7
Practices: Creating a Single-Page Application using RESTand a Tic-Tac-Toe Game Client with WebSocket

Developing Applications with jQuery
Adding jQuery and jQuery UI libraries to your projects
Using Selectors and DOM manipulators to handle documents
Handling Events with jQuery
Animating elements and Applying effects in the document
Handling AJAX server responses

MÓDULO II

DIPLOMADO

Java SE 8 & JEE7 Developer

Curso Oficial ORACLE University

Java EE 6: Develop Web Services with JAX-WS & JAX-RS

Duration: 5 Days

What you will learn

This Java EE 6 programming course covers the design and creation of SOAP and RESTful web services and clients. You'll use the NetBeans Integrated Development Environment (IDE) to develop JAX-WS and JAX-RS web services and deploy those services to Oracle WebLogic Server 12c. The majority of topics covered are portable across all application servers which support the Java EE 6 web service standards.

Learn To:

Create XML documents and XML schemas while using XML Namespaces.
Produce and consume JSON and XML using JAXB.
Understand WSDL files and the role they play in SOAP based web services and select either a top-down (WSDL first) or bottom-up (code first) approach to the development of SOAP web services.
Make calls to and implement web services based on SOAP standards using JAX-WS (Metro Stack).
Implement REST practices in the creation of web services with the JAX-RS specification (Jersey Stack).
Secure web services using Java EE Security standards, WS-Security extensions, and OAuth 1.0a.

Benefits to You

Java EE 6 technology facilitates cross-platform application development through the use of platform neutral network communication, supports HTML5 AJAX enabled applications and mobile clients by creating RESTful web services which use the JSON data-interchange format. Enrolling in this course will help you stay current on the latest Java EE 6 web service APIs.

Audience

J2EE Developer
Java Developers
Java EE Developers

Related Training

Required Prerequisites

Java SE7 Fundamentals

Java SE 7 Programming

Suggested Prerequisites
Java Design Patterns

Java SE 7: Develop Rich Client Applications

Oracle Certified Associate, Java SE 7 Programmer

Oracle Certified Professional, Java SE 7 Programmer

Tutorials available on the Oracle Learning Library

Course Objectives

Apply the JAX-RS API in the creation of RESTful Web Services

Secure Web Services using WS-Security, Jersey, and OAuth

Handle errors and exceptions in Web Services and clients

Create XML documents using namespace declarations and XML schema

Produce and consume XML and JSON content using JAXB

Create RESTful Web Service clients using the Jersey Client API

Understand the role of Web Services

Apply the JAX-WS API in the creation of SOAP Web Services and clients

Course Topics

An Introduction to Web Services
Explaining the need for web services
Defining web services
Explaining the characteristics of a web service
Explaining the use of both XML and JSON in web services
Identifying the two major approaches to developing web services
Explaining the advantages of developing web services within a Java EE container

XML
Describing the Benefits of XML
Creating an XML Declaration
Assembling the Components of an XML Document
Declaring and Apply XML Namespaces
Validating XML Documents using XML Schemas
Creating XML Schemas

JAXB
Listing the Different Java XML APIs
Explaining the Benefits of JAXB
Unmarshalling XML Data with JAXB
Marshalling XML Data with JAXB
Compiling XML Schema to Java
Generating XML Schema from Java Classes

Applying JAXB Binding Annotations
Creating External Binding Configuration Files

SOAP Web Services
SOAP message structure
Using WSDL files to define web services
WS-I Basic Profile and WS-Policy

Creating JAX-WS Clients
Using tools to generate JAX-WS client artifacts
Calling SOAP web services using JAX-WS in a Java SE environment
Calling SOAP web services using JAX-WS in a Java EE environment
Using JAXB Binding customization with a SOAP web service
Creating a JAX-WS Dispatch client
Creating a client that consumes a WS-Policy enhanced services (WS-MakeConnection)

RESTful Web Services
Describing the RESTful architecture and how it can be applied to web services
Designing a RESTful web service and identify resources
Navigating a RESTful web service using hypermedia
Selecting the correct HTTP method to use when duplicate requests must be avoided
Identifying Web Service result status by HTTP response code
Version RESTful web services

Creating RESTful Clients in Java
Using Java SE APIs to make HTTP requests
Using the Jersey Client APIs to make HTTP requests
Processing XML and JSON in a RESTful web service client

Bottom-Up JAX-WS Web Services
Describing the benefits of Code First Design
Creating JAX-WS POJO Endpoints
Creating JAX-WS EJB Endpoints

Top-Down JAX-WS Web Services
Describing the benefits of WSDL First Design
Generating Service Endpoint Interfaces (SEIs) from WSDLs
Implementing Service Endpoint Interfaces
Customizing SEI Generation

JAX-RS RESTful Web Services
Download, Install, and Configure Jersey
Creating Application Subclasses
Creating Resource Classes
Creating Resource Methods, Sub-Resource Methods, and Sub-Resource Locator Methods
Producing and Consume XML and JSON content with JAX-RS

Web Service Error Handling
Describing how SOAP web services convey errors
Describing how REST web services convey errors
Returning SOAP faults
Returning HTTP error status codes

Mapping thrown Exceptions to HTTP status codes
Handling errors with SOAP clients
Handling errors with Jersey clients

Security Concepts
Explaining Authentication, Authorization, and Confidentiality
Applying Basic Java EE Security by using deployment descriptors (web.xml)
Creating users and groups and map them to application roles
Detailing possible web service attack vectors

WS-Security
Describing the purpose of WS-Policy, WS-SecurityPolicy, WS-Security
Configuring WebLogic Server for WS-Security
Applying WS-Policy to WebLogic JAX-WS Web Services
Signing and Encrypt SOAP Messages using WS-Security

Web Service Security with Jersey
Applying JSR-250 Security Annotations such as @RolesAllowed
Enabling an assortment of filters including the RolesAllowedResourceFilterFactory
Obtaining a SecurityContext and perform programmatic security
Authenticating using the Jersey Client API

OAuth 1.1a with Jersey
Describing the purpose of OAuth
Describing the request lifecycle when using OAuth
Creating OAuth enabled services using Jersey
Creating OAuth enabled clients using Jersey

Java Design Patterns

Duration: 4 Days

What you will learn

This Java Patterns course reviews common and emerging patterns specific to Java SDK and EE development. You'll learn the depth and evolution of pattern-based techniques in Java, with particular emphasis on Java EE 6 conventions.

Learn To:

Dinstinguish between Java EE 5 and Java EE 6 pattern-based features.
Implement relevant patterns in each tier of the Java EE environment.
Re-factor code to improve inter-tier communications.
Relate pattern-based development to an implementation architecture.
Apply object-oriented pronciples and design guidelines.
Implement well-known patterns to Java-specific code problems.

Lab Exercises

The lab exercises show you how to identify, apply and re-factor selected patterns into code, using a NetBeans or Eclipse IDE and the GlassFish Application Server v3. You'll also learn a subset of UML notation to expedite communicating through design instead of code.

Java Design Patterns

In design patterns, the responsibility of each component is identified by role. The conventions of design pattern documentation make it easier for development teams to communicate their programming intentions and provide a reference point for the entire Java development community.

Java-Based Frameworks

The Java language and popular Java-based frameworks incorporate more proven development practices into their programming interfaces with each major release. These practices, referred to as design patterns, document well-known names, code implementation and re-factoring techniques, and the risks and trade-offs associated with using them.

Audience

Application Developers
Architect
J2EE Developer
Java Developers
Java EE Developers

Related Training

Required Prerequisites

Experience with Java SE and Java EE development

Developing Applications for the Java EE 6 Platform

Course Objectives

Identify key design principles of object-oriented development

Apply Java-specific implementation techniques to well-known patterns

Use patterns to complete a Java application design

Use patterns to complete a web-tier application design

Use patterns to complete a business-tier application design

Use patterns to improve communication between Java EE tiers

Identify and refactor anti-patterns in working code

Using part of a sample architecture scheme, select design patterns for implementing the scheme

Course Topics

Reviewing Object-Oriented Principles in Java
Describe how OO concepts apply to Java
Describe how OO principles apply to Java
List the goals of an OO language
Interpret Unified Modeling Language (UML) notation and create UML diagrams
Identify selected design patterns

Reviewing Gang of Four Patterns
List key behavioral, creational and structural patterns
Apply the Facade pattern
Apply the Strategy pattern
Apply the Observer pattern
Apply the Composite pattern
Review the Model-View-Controller (MVC) patterns

Implementing Patterns in Java
Use implementation patterns designed for Java
List forces affecting class, state, and behavioral patterns
Describe how patterns, idioms and refactoring differ from each other

Exploring Changes in Java EE Technology
Describe the design goals of the Java EE model

Describe improvements in the Java EE 6 model

Implementing Integration Patterns
Describe design patterns for the integration tier
Review Java EE integration changes that apply design patterns
Identify use cases for applying integration tier patterns

Implementing Patterns in Business Components
Describe the role of an enterprise bean
Describe design patterns for the business tier

Implementing Infrastructural Patterns in Java EE
Describe the role of infrastructural Java EE patterns
Describe the Service Starter pattern
Describe the Singleton pattern
Describe the Bean Locator pattern
Describe the Resource Binder pattern

Implementing More Infrastructure Patterns
Describe how Java EE interceptors work
Describe the Dependency Injection Extender pattern
Describe the Payload Extractor pattern
Describe the Context Holder pattern
Describe the Thread Tracker pattern

Exploring Anti-Patterns
Describe the Law of Leaky Abstractions
Define AntiPatterns
Describe Integration Tier AntiPatterns
Describe Business Tier AntiPatterns
Describe Presentation Tier AntiPatterns

Selecting Patterns for Architecture
Define the roles of architect, designer, and developer
Describe the relationship between design patterns and architecture
List guidelines for applying patterns to an architectural solution

Temario curso

Desarrollo de Aplicaciones Empresariales con Spring Framework Core 4

I. Generales

El temario comprende las siguientes tecnologías.

1. **Introducción a Spring Framework**
2. **Spring Core**
3. **Spring AOP**
4. **Spring JDBC - Transaction**
5. **Spring ORM – Hibernate 4**
6. **Fundamentos Spring MVC y Spring Security**
7. **Fundamentos Spring REST**

II. Temario

Introducción a Spring Framework 4

i. ¿Qué es Spring Framework?
   a. POJOs
   b. JavaBeans
   c. Spring Beans
ii. Motivación de Spring Framework
iii. Arquitectura y Módulos principales
iv. Proyectos Spring.io
v. Programación orientada a interfaces
vi. Instalación ambiente de desarrollo

vii. Práctica 1. Definición y uso de Interfaces

Spring Core

i. Conceptos
    a. Inversión de Control
    b. Inyección de Dependencias
    c. Inversión de Dependencias
ii. Contenedor de IoC
    a. BeanFactory
    b. ApplicationContext
    c. Tipos de configuración de Beans
iii. Configuración de Beans con XML
    a. Definición de Beans
    b. Inyección de Dependencias
iv. Práctica 2. Inyección de Dependencias Bean Factory
v. Práctica 3. Inyección de Dependencias Application Context
vi. Bean Scopes
    a. Práctica 4. Bean Scopes
vii. Ciclo de vida de Beans
    a. Inicialización Lazy
    b. Práctica 5. Init – Destroy ciclo de vida
    c. Práctica 6. Definición lazy de Beans
viii. Definición de Beans heredada (Templates)
    a. Práctica 7. Definición de beans heredada
ix. Bean Post Processors

a. Práctica 8. Bean post processors

x. Beans internos

a. Práctica 9. Beans Internos

xi. Inyección de Colecciones y Arreglos

a. Práctica 10. Inyección de Colecciones y Arreglos.

xii. Namespace p, c y util

xiii. Autowiring

a. Práctica 11. Autowiring

xiv. Práctica 12. Convertidor número a letra configuración XML

xv. Configuración con @Anotaciones.

a. Namespace context

b. @Required, @Autowired y @Qualifier

    i. Práctica 13. @Required, @Autowired y @Qualifier

c. Anotaciones JSR 250 @Resource, @PostConstruct y @PreDestroy

    i. Práctica 14. @Resource, @PostConstruct y @PreDestroy

d. Component scan

e. Anotaciones estereotipo @Component, @Service, @Repository, @Controller

    i. Práctica 15. @Component, @Service, @Repository, @Controller

f. Anotaciones JSR 330 @Inject y @Named

    i. Práctica 16. @Inject y @Named

xvi. Configuración con Clases Java.

a. Anotaciones @Configuration, @Bean e @Import

    i. Práctica 17. @Configuration, @Bean e @Import

xvii. Práctica 18. Convertidor número a letra configuración @Anotaciones

vi. Spring AOP con XML

   a. Dependencias
   b. Configuración
      i. Aop config
   c. Deficinión de pointcut
      i. Expresiones de Pointcut
   d. Tipos de Advice
      i. Before advice
      ii. After advice
      iii. After returning advice
      iv. After throwing advice
      v. Around advice
   e. Práctica 23. Spring AOP usando configuración XML

vii. Spring AOP con @Anotaciones

   a. Dependencias
      i. @AspectJ
   b. Configuración
      i. Aop aspectj-autoproxy
   c. Deficinión de pointcut
      i. Expresiones de Pointcut
   d. Tipos de Advice
      i. Before advice
      ii. After advice
      iii. After returning advice
      iv. After throwing advice

v. Around advice

e. Práctica 24. Spring AOP usando configuración @Anotaciones

f. Comparativa Spring AOP XML y AOP @Anotaciones

g. Práctica 25. Spring AOP Adivinador

Spring JDBC - Transaction

i. API JDBC

   a. Implementación API JDBC

ii. ¿Por qué usar Spring JDBC?

iii. Filosofía de Acceso a Datos

   a. Patrón de diseño DAO

   b. DataSource

iv. Manejo de excepciones

v. Spring Jdbc Templates

   a. JdbcTemplate

   b. NamedParameterJdbcTemplate

vi. Jdbc DAO Support

vii. Uso JdbcTemplate

   a. Execute

   b. Query

   c. QueryFor

   d. Update

   e. BatchUpdate

viii. Jdbc Callbacks

   a. RowMapper

d. Forward y SendRedirect (Servlet API)

vii. Validaciones
viii. Práctica 29. Implementación aplicación CRUD
ix. Spring Security Introducción
x. ¿Qué es Spring Security?
xi. Módulos Spring Security
xii. Configuración Spring Security
xiii. Implementando Seguridad en Spring MVC
xiv. Práctica 30. Formulario Login básico

Fundamentos Spring REST

i. ¿Qué es REST?
ii. Principios de REST
iii. HTTP REST code statsus
iv. Headers
v. Produces y Consumes
vi. @ResponseBody y @RequestBody
vii. Servicios RESTful con Spring
    a. Configuración Dependencias
    b. Implementación REST
        i. GET
        ii. POST
        iii. PUT
        iv. DELETE
viii. Práctica 31. Implementación Servicios RES