

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