1

Introduction to Web Services

Objectives

After completing this lesson, you should be able to do the following:

- Explain the need for web services
- Define web services
- Explain the characteristics of a web service
- Explain the use of both XML and JSON in web services
- Identify the two major approaches to developing web services
- Explain the advantages of developing web services within a Java EE

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- **Lesson 1: Introduction to Web Services** — You are here!
- Lesson 2: Creating XML Documents
- Lesson 3: Processing XML with JAXB
- Lesson 4: SOAP Web Services Overview
- Lesson 5: Creating JAX-WS Clients

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 6: Exploring REST Services
- Lesson 7: Creating REST Clients
- Lesson 8: Bottom Up JAX Web Services
- Lesson 9: Top Down JAX Web Services
- Lesson 10: Implementing JAX RS Web Services

Course Roadmap

Application Development Using Webservices [SOAP and Restful]

- Lesson 11: Web Service Error Handling
- Lesson 12: Java EE Security and Securing JAX WS

Course Objectives

After completing this course, you should be able to:

- Create XML documents by using namespace declarations and XML Schema
- Produce and consume XML and JSON content by using JAXB
- Create REST web service clients
- Create SOAP web service clients by using JAX-WS
- Create REST web services by using JAX-RS
- Create SOAP web services by using JAX-WS
- Handle errors in web service clients and services
- Secure web services by using WS-Security

Audience

The target audience includes:

- Java SE developers
- Java EE developers

Prerequisites

To successfully complete this course, you must have experience with:

- Developing applications using Java SE and Java EE
- Java SE 7 (ideally)

Course Schedule

Session	Module
Day 1	Lesson 1: Introduction to Web Services Lesson 2: XML Document Structure Lesson 3: XML Parsing with JAXB
Day 2	Lesson 4: SOAP Web Service: Overview Lesson 5: Creating JAX-WS Clients Lesson 6: RESTful Web Service: Overview
Day 3	Lesson 7: Creating REST Clients Lesson 8: Bottom-up JAX-WS Web Services Lesson 9: Top-down JAX-WS Web Services
Day 4	Lesson 10: Implementing JAX-RS Web Services
Day 5	Lesson 11: Web Service Error Handling Lesson 12: Security Overview

Course Environment

Classroom PC

Core Apps

- JDK 8
- NetBeans 8
- Java EE6
- WebLogic

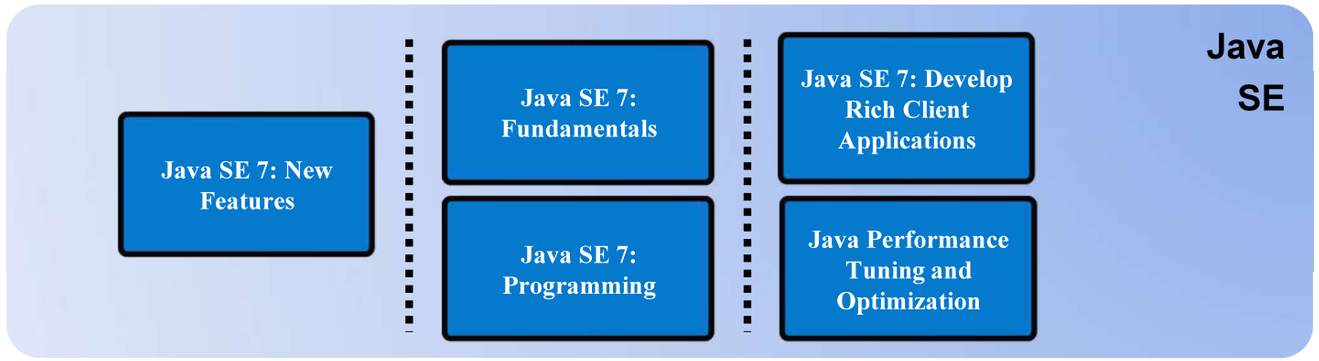
Practice Files

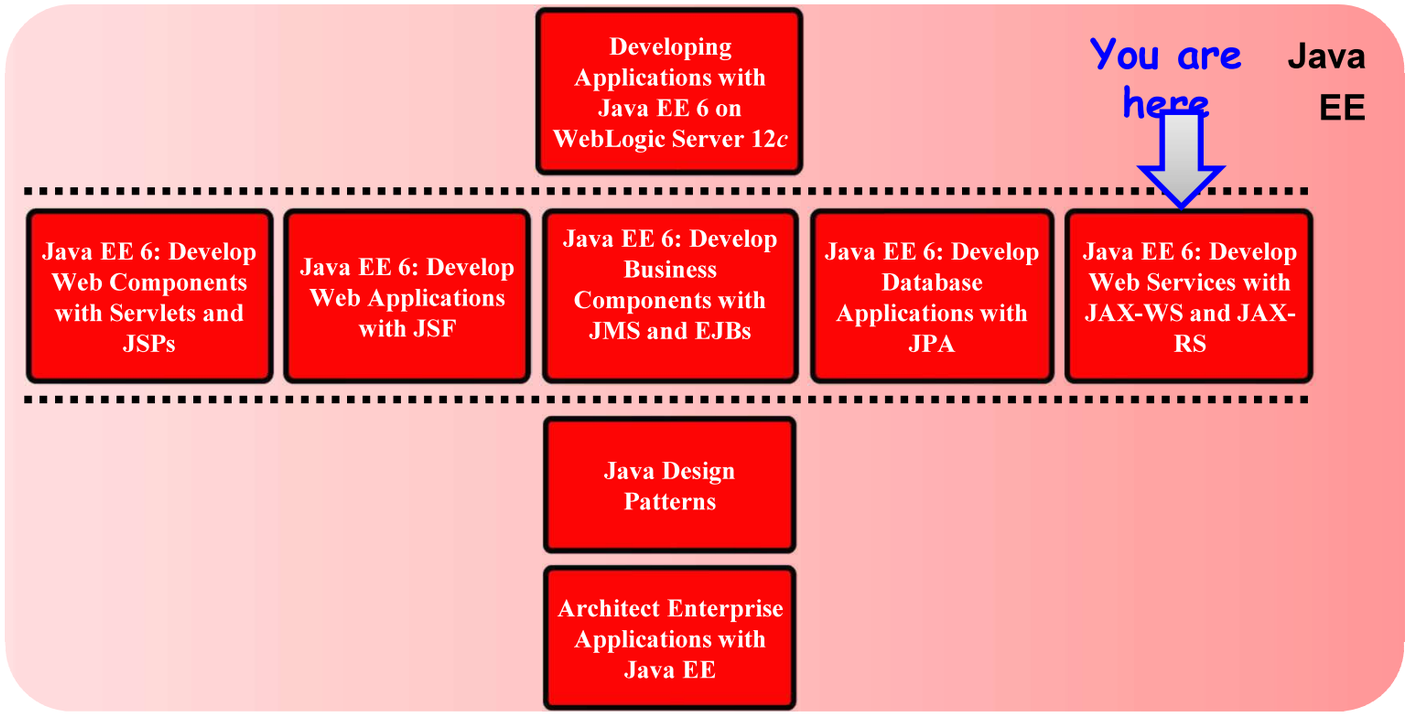
- Example folders
- Practice folders

Additional Tools

- Firefox
- RESTClient extension for Firefox
- cURL
- jQuery
- Activity Guide
- Java API Documentation and Java Language Specification
- Java EE Specifications

Java Curriculum





Need for Web Services

A computer program calling a subroutine located on a different machine on a network is not a new development.

- RPC from Sun MicroSystems was an early example of cross-system execution.
- Other examples include .Net Remoting, CORBA/IIOP, and RMI.
- In all cases, the need is the same:
  - Run an operation on a remote machine (or even on the same machine but in a different address space).
- Many of the remoting technologies suffer from problems:
  - Platform-specific (CPU architecture or programming language)
  - Complexity

Web Service: Definition

- The World Wide Web Consortium (W3C) defines a web service as follows:
- “A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards.”
- **Note:** Over the years, web services have evolved beyond this definition.

Characteristics of Web Services

- Platform neutral (both CPU architecture and programming language independent)
  - A platform-neutral data-interchange format is needed (text instead of binary).
- Client-server architecture
  - A server has a set of available operations.
  - A client can request the execution of an operation on a server.
- “Web” services are most likely to use HTTP as a transport protocol.
  - In theory, both SOAP and RESTful services are not tied to HTTP. In practice, you almost always use HTTP.
- The use of a service must be described.

Service Descriptions

Web service clients are coded to call operations present on a remote server. The available operations must be discovered in order for the client to call them.

- Web Services Description Language (WSDL): Similar to a Java interface. A list of all operations, parameters, and return types.
  - Primarily for SOAP services
  - WSDL 2.0 can support REST services.
- Web Application Description Language (WADL): An XML description of REST services operations.
- Human readable: Developers review HTML pages describing available URLs and review sample code.
- Self-describing: Given a staring URL, you discover the “rest.”

XML and JSON in Web Services

To facilitate a platform-neutral exchange of data, a general purpose data-interchange format is needed.

- Extensible Markup Language (XML): Used by both SOAP and REST web services
  - A large number of processing libraries exist to support XML in almost every language.
  - Java developers can use SAX, DOM, StAX, and JAXB.
  - SOAP web service rely on XML but libraries hide a large portion of the XML work.
- JavaScript Object Notation (JSON): Used by REST web services
  - A subset of JavaScript
  - Less verbose than XML
  - Support is evolving in other languages.

Approaches to Developing Web Services

There are two main types of web services:

- SOAP: Heavily standards based, evolved over the years
  - XML data is transmitted across HTTP using a POST request.
  - The XML data lists the method and parameters to call.
  - An XML response is generated containing the method return data.
  - It is closely tied to formal interface definitions (WSDL).
    - WSDL > Code (top-down development)
    - Code > WSDL (bottom-up development)
- REST: Lightweight, less formal approach
  - Uses HTTP operations as method names (GET, POST, and so on)
  - Different URLs for different “resources” (similar to objects)
  - Parameters and return data in either XML or JSON

Web Services Implementations

- SOAP web services are supported by JAX-WS.
  - JAX-WS is part of Java SE, but a production-grade HTTP server is not.
  - All the extra WS-* features are not included.
  - The Metro project is a popular “complete” JAX-WS implementation that is a GlassFish project.
  - WebLogic Web Services Stack is now based on Metro.
- RESTful web services are supported by JAX-RS.
  - Not part of Java SE
  - JAX-RS covers implementing only services, not clients.
  - Jersey is a production-quality implementation that is also the reference implementation.

Web Services Within Java EE Containers

Web services rely on HTTP communication and therefore need production-grade HTTP connection handling—a web container.

- Tomcat, GlassFish, and WebLogic are all examples of web containers.
- Only Java EE Full Profile servers support JAX-WS and JAX-RS out of the box.
- Metro and Jersey can be added to a basic web container like Tomcat.
- Web services simply provide a platform-neutral wrapper around business logic.
  - If your logic needs transactions, security, or any of the other features provided by a Java EE application server, then you will most likely end up creating web service classes, which delegate to EJBs that use JPA.

Web Service Tools

Web service testing tools are commonly used during the development of web services, primarily as general-purpose clients.

- Application servers may have integrated testing tools—GlassFish and WebLogic have web applications that generate forms to call SOAP operations.
- Command-line clients like cURL are great for testing and also allow shell scripts to function as web service clients.
- Because web browsers have extensive support for HTTP, they can not only function a basic testing tools but also commonly have extensions that provide greater HTTP control to the end user.

Which protocol is typically used by web services?

a. RPC (Open Network Computing Remote Procedure Call)
b. RMI-IIOP (Remote Method Invocation over the Internet Inter-Orb Protocol)
c. Distributed Component Object Model (DCOM)
d. Hypertext Transfer Protocol (HTTP)

Which of the following are common characteristics of a web service?

a. A text-based data interchange format
b. A peer-to-peer architecture
c. Support for the Java Message Service (JMS) API
d. A client accessible service description

Resources

Topic	Website
Web Services Architecture	http://www.w3.org/TR/ws-arch/
Metro	http://metro.java.net/
Jersey	http://jersey.java.net/
The Java EE 6 Tutorial	http://docs.oracle.com/javaee/6/tutorial/doc/

Summary

In this lesson, you should have learned how to:

- Explain the need for web services
- Define web services
- Explain the characteristics of a web service
- Explain the use of both XML and JSON in web services
- Identify the two major approaches to developing web services
- Explain the advantages of developing web services within a Java EE container

Practice 1: Overview

This practice covers the following topics:

- Configuring NetBeans to Control WebLogic Server
- Creating and Deploying Web Service Sample Applications
- Web Service Testing