<<Java Web 服务>>

图书基本信息

书名：<<Java Web 服务>>

13位ISBN编号：9787564119270

10位ISBN编号：7564119276

出版时间：2010-1

出版时间：东南大学

作者：卡林

页数：297

前言

This is a book for programmers interested in developing Java web services and Java clients against web services, whatever the implementation language. The book is a code- driven introduction to JAX-WS (Java APl for XML-Web Services), the framework of choice for Java web services, whether SOAP-based or REST-style. My approach is to interpret JAX-WS broadly and, therefore, to include leading-edge developments such as the Jersey project for REST-style web services, officially known as JAX-RS (Java APl for XML-RESTful Web Services). JAX-WS is bundled into the Metro Web Services Stack, or Metro for short. Metro is part of core Java, starting with Standard Edition 6 (hereafter, core Java 6). However, the Metro releases outpace the core Java releases. The current Metro release can be down- loaded separately from https://wsit, dev.java.net. Metro is also integrated into the Sun application server, GlassFish. Given these options, this book's examples are deployed in four different ways: Core Java only This is the low-fuss approach that makes it easy to get web services and their clients up and running. The only required software is the Java software development kit (SDK), core Java 6 or later. Web services can be deployed easily using the Endpoint, HttpServer, and HttpsServer classes. The early examples take this approach. Core Java with the current Metro release This approach takes advantage of Metro features not yet available in the core Java bundle. In general, each Metro release makes it easier to write web services and clients. The current Metro release also indicates where JAX-WS is moving. The Metro release also can be used with core Java 5 if core Java 6 is not an option. Standalone Tomcat This approach builds on the familiarity among Java programmers with standalone web containers such as Apache Tomcat, which is the reference implementation. Web services can be deployed using a web container in essentially the same way as are servlets, JavaServer Pages (JSP) scripts, and JavaServer Faces (JSF) scripts. A standalone web container such as Tomcat is also a good way to introduce container-managed security for web services. GlassFish This approach allows deployed web services to interact naturally with other enterprise components such as Java Message Service topics and queues, a JNDI (Java Naming and Directory Interface) provider, a backend database system and the @Entity instances that mediate between an application and the database sys- tem, and an EJB (Enterprise Java Bean) container. The EJB container is important because a web service can be deployed as a stateless Session EJB, which brings advantages such as container-managed thread safety. GlassFish works seamlessly with Metro, including its advanced features, and with popular IDEs (Integrated Development Environment) such as NetBeans and Eclipse. An appealing feature of JAX-WS is that the API can be separated cleanly from deploy- ment options. One and the same web service can be deployed in different ways to suit different needs. Core Java alone is good for learning, development, and even lightweight deployment. A standalone web container such as Tomcat provides additional support. A Java application server such as GlassFish promotes easy integration of web services with other enterprise technologies. Code-Driven Approach My code examples are short enough to highlight key features of JAX-WS but also re- alistic enough to show off the production-level capabilities that come with the JAX-WS framework. Each code example is given in full, including all of the import statements. My approach is to begin with a relatively sparse example and then to add and modify features. The code samples vary in length from a few statements to several pages of source. The code is deliberately modular. Whenever there is a choice between con- ciseness and clarity in coding, I try to opt for clarity. The examples come with instructions for compiling and deploying the web services and for testing the service against sample clients. This approach presents the choices that JAX-WS makes available to the programmer but also encourages a clear and thor- ough analysis of the JAX-WS libraries and utilities. My goal is to furnish code samples that can serve as templates for commercial applications. JAX-WS is a rich API that is explored best in a mix of overview and examples. My aim is to explain key features about the architecture of web services but, above all, to illus- trate each major feature with code examples that perform as advertised: Architecture without code is empty; code without architecture is blind. My approach is to integrate the two throughout the book. Web services are a modern, lightweight approach to distributed software systems, that is, systems such as email or the World Wide Web that require different software com- ponents to execute on physically distinct devices. The devices can range from large servers through personal desktop machines to handhelds of various

types. Distributed systems are complicated because they are made up of networked components. There is nothing more frustrating than a distributed systems example that does not work as claimed because the debugging is tedious. My approach is thus to provide full, working examples together with short but precise instructions for getting the sample application up and running. Chapter-by-Chapter Overview The book has seven chapters, the last of which is quite short. Here is a preview of each chapter: Chapter 1, Java Web Services Quickstart This chapter begins with a working definition of web services, including the dis- tinction between SOAP-based and REST-style services. This chapter then focuses on the basics of writing, deploying, and consuming SOAP-based services in core Java. There are web service clients written in Peri, Ruby, and Java to underscore the language neutrality of web services. This chapter also introduces Java's SOAP APl and covers various ways to inspect web service traffic at the wire level. The chapter elaborates on the relationship between core Java and Metro.

<<Java Web 服务>>

内容概要

《Java Web 服务:构建与运行(影印版)》提供了对Java的API的一个全面介绍，包括针对XML Web服务的JAX-WS和针对RESTful Web服务的JAX-RS。
《Java Web服务：构建与运行》通过提供混合架构概述、完整的工作代码示例以及短而精确的编译、部署和执行应用程序的指示，采用明确实用的方法来处理这些技术。
你将学习如何从头开始编写Web服务以及集成现有服务到你的Java应用程序中。
有了这《Java Web 服务:构建与运行(影印版)》，你将：· 理解基于SOAP的和REST样式的服务的区别· 编写、部署和使用基于SOAP的核心Java服务· 理解Web服务描述语言（WSDL）服务契约· 认识SOAP消息的结构· 学习如何交付基于Java的RESTful Web服务和消耗商业RESTful服务· 了解对基于SOAP和基于REST的Web服务的安全要求· 学习如何在各种环境下部署JAX-WS服务不管是学生还是有经验的程序员，当你需要立即运用这些技术展开工作时，《Java Web服务：构建与运行》都是你需要的一本理想的简明指南。

作者简介

Martin Kalin, 德保罗大学计算机和数字媒体学院教授，拥有西北大学的博士学位。
他撰写过关于C语言、C++和Java的书，并参与开发过大型分布式系统中的进程调度和产品配置。

<<Java Web 服务>>

书籍目录

章节摘录

插图：The HTTP start line comes first and specifies the request method, in this case thePOST method, which is typical of requests for dynamic resources such as webservices or other web application code （for example, a Java servlet） as opposed torequests for a static HTML page. In this case, a POST rather than a GET requestis needed because only a POST request has a body, which encapsulates the SOAPmessage. Next comes the request URL followed by the HTTP version, in this case1.1, that the requester understands. HTTP 1.1 is the current version.Next come the HTTP headers, which are key/value pairs in which a colon （:） separates the key from the value. The order of the key/value pairs is arbitrary. The keyAccept occurs three times, with a MIME （Multipurpose Internet Mail Extensions） type/subtype as the value: text/xml, multipart/*, and application/soap. These threepairs signal that the requester is ready to accept an arbitrary XML response, aresponse with arbitrarily many attachments of any type （a SOAP message can havearbitrarily many attachments）, and a SOAP document, respectively. The HTTP keySOAPAction is often present in the HTTP header of a web service request and thekey's value may be the empty string, as in this case; but the value also might be thename of the requested web service operation.Two CRLF （Carriage Return Line Feed） characters, which correspond to two Java\n characters, separate the HTTP headers from the HTTP body, which is requiredfor the POST verb but may be empty. In this case, the HTTP body contains theSOAP document, commonly called the SOAP envelope because the outermost ordocument element is named Envelope. In this SOAP envelope, the SOAP body contains a single element whose local name is getTimeAsString, which is the name ofthe web service operation that the client wants to invoke. The SOAP request envelope is simple in this example because the requested operation takes noarguments.

媒体关注与评论

“《Java Web服务：构建与运行》有很多我发现非常实用的实例，从使用Amazon Associates Web服务到有很好的图解和安全说明以及加密密码的关于安全的一章，还包括使用证书的示例，这部分内容我还没在其他书中见过。
”　　——Greg Ostravich，Denver Java用户组主席

编辑推荐

《Java Web 服务:构建与运行(影印版)》是由东南大学出版社出版的。

版权说明