Chapter 1

JUMPSTART

1.1 What Are the Next Frontiers?

1.1.1 The Web Services Phenomenon

In June of 2003, there were more than 113 printed books on Web Services available at Amazon.com, compared to 2 printed books in September of 2001, 45 printed books in March of 2002, and 86 printed books in February of 2003. Do a search on the Web, and you will be faced with a huge collection of articles and Web sites devoted to Web Services. Additionally, there are a large number of printed books available to the Web Services audience about an almost identical subject: WSDL-UDDI-SOAP technology. With all of these resources available, how does one choose the best of the bunch?

Scanning through a few titles on the subject, readers may notice that while there are abundant printed materials on Web Services concepts and SOAP programming. There is a general lack of emphasis on the Quality of Services (or so-called "ilities" such as scalability and availability) and real-life, full-scale implementations of Web Services in the printed media. It is not easy to differentiate market hype from a reliable technology implementation, but this book intends to do so—differentiate real-

life examples from the market hype technology, define the Quality of Services for successful Web Services implementation, and discuss how it is applied to legacy mainframe interoperability and Business-to-Business integration.

1.1.2 The Web Services Evolution

Web Services technology and associated vendor products have evolved greatly over the past few years, from SOAP 1.0 to SOAP 1.2. New related technologies emerge from time to time, such as the earlier releases of the Java Web Services Developer Pack™ (including JAX Pack) and Apache Axis. Java™ has become the de facto lingua franca of Web Services. Previously, developers needed to build their own application infrastructure tools to handle security, exceptions, and logging. They also needed to write programming routines to handle message acknowledgement and exception handling routines to ensure SOAP messages were reliably exchanged. Nowadays, new Web Services tools such as Java Web Services Developer Pack™ and Apache Axis provide many built-in Web Services security features, SOAP messaging capabilities, and SOAP server infrastructure tools. These technologies have almost completely changed the previous Web Services programming model and landscape.

As more developers become familiar with these new enabling Web Services technologies, they need to assemble different infrastructure and application components together in order to provide a complete Web Services solution. Most literature today addresses how to build SOAP programs or lab prototypes, but does not address how to implement a complete Web Services solution to interoperate with legacy mainframe applications or to integrate with a Business-to-Business application. There are also many design elements and considerations for the cross-platform interoperability (such as mainframe interoperability), integration experience (such as cross-enterprise integration), and best practices (for example, architecture patterns) that constitute the bigger picture of Web Services technology implementation. These elements are the extended frontiers of Web Services technology toward a total business solution. This book attempts to explore these frontiers by discussing a structured framework for mainframe interoperability, cross-enterprise integration, and some design best practices.

1.2 How This Book Is Different

Although there are a number of recent books on the topic of Web Services, much of the information currently available is purely marketing hype about what Web Services can do, with little practical information for developers and architects. To emphasize how this book differs from others, I have identified five distinctive features. I also expand on three key ideas, and finish this section by presenting an overview of its unique values.

1.2.1 Five Distinctive Features of This Book

In addition to the list of technical subjects covered in the table of contents, there are five distinctive features in this book. These features are designed to address different needs of the developer and architect audience. They also help differentiate this book from similar Web Services books.

The Frontiers of Web Services

The frontiers of Web Services technology that describe how to build a total business solution include mainframe and legacy systems interoperability and cross-enterprise integration (for example, SOAP-JMS binding). The book also introduces how to use some cool developer toolkits (such as the Java Web Services Developer Pack™ and Apache Axis) for building Web Services prototypes. This goes beyond a textbook on SOAP programming or a concept book on Web Services technology. These technology areas are helping to bring Web Services implementations closer to reality.

Real-Life Examples

Current, publicly available Web Services implementations are analyzed throughout the book. Real-life examples and business scenarios make the technology contextual and make it easier for developers and architects to reapply appropriate Web Services technology in their own business environment. They differentiate the book from others that focus primarily on the technology details. Different use cases or business scenarios applying Web Services, will be outlined to help establish a business case for implementing Web Services.

Design Patterns and Best Practices

Some Web Services design patterns and when-to-use architecture principles are discussed. The Web Services design patterns and best approaches address the different needs of infrastructure architects, J2EE developers, security architects, and integration architects. These best practices are accumulated and extracted from past customer engagement experience. The emphasis will be on building Quality of Services (the so-called "ilities") for reliable, available, and scalable Web Services. This can be the basis for customizing a Web Services development and deployment cookbook.

Web Services Technology Market Space

A sampler of major Web Services vendor architectures and products is supplied in Appendix B. It provides a quick overview and a handy guide to the Web Services technology market space. A list of URLs and other resources is also provided because these usually provide more up-to-date information on vendors and products. These resources can assist developers and architects who want to do a tool selection and vendor assessment.

Paper and Pencil Lab Exercise

Guided hands-on labs are offered to build complete examples incrementally. Developers and architects can start with basic skills in Web Services technologies such as SOAP, WSDL, and JAXM in earlier chapters, and then Web Services development tools and security add-on tools in later chapters. These exercises help build up the skills needed to develop a complete prototype in Chapter 8, Web Services in Action: Case Study.

1.2.2 Three Key Ideas

There are three key ideas conveyed in this book, and they are backed up by technical details of various Web Services technologies.

Technology With Business

Technology is an enabling tool to collaborate with business; it does not rule over business. What is the business case for Web Services? Web Services technology can tie business benefits and service level to a company's bottom-line goals. This book will provide ideas and examples on how to establish a business case for Web Services solutions. It helps developers and architects collaborate with business, ties the technology solutions to their business environment, and identifies the benefits to their bottom-line goals.

The Big Picture

SOAP and UDDI are only parts of the big picture of Web Services technology. Web Services applications do not begin and end with a SOAP program or a UDDI look-up. Web Services solutions using ebXML provide a richer set of messaging and workflow functionality for Business-to-Business integration (B2Bi), and they have become more visible in the industry recently. Quality of Services, or so-called "ilities" such as scalability and reliability, becomes a key challenge to developers and implementation managers. Thus, it is important to look at different components of Web Services technology solutions and at various aspects of designing and scaling the Web Services solutions. This book examines different aspects of Web Services technology. It intends to present a bigger picture of Web Services technology, instead of focusing on solely WSDL, UDDI, and SOAP.

What Makes Web Services a Killer Application

What makes a solution a killer application is dependent on whether it creates "stickiness" (good user experience), ease of use, and the flexibility to implement and integrate. There is much market hype about Web Services being killer applications. The underlying enabling technology of WSDL, UDDI, and SOAP is not new. Web Ser-

vices applications may be relatively simple to implement and can address existing technology challenges. But there is a fine distinction between the Web Services phenomenon and the Web Services market hype. This book discusses business scenarios where Web Services solutions can become killer applications, and clarifies any market hype by exploring what can be done (the capabilities of Web Services technology based on its associated strengths) and cannot be done (the weaknesses of Web Services technology).

1.2.3 Inside the Big Ideas

This book is about what Web Services technology is and how different technology components can be assembled to build solutions. The following paragraphs summarize some key technology components and relate them to the big ideas in the book.

Web Services 101

The first few chapters provide a refresher course on Web Services technology basics, including WSDL, UDDI, SOAP, and ebXML technology. It also explains what makes the WSDL-UDDI-SOAP technology more popular than CORBA—it is easier to understand and implement. WSDL-UDDI-SOAP technology was heavily marketed by Microsoft in the early days (before it became open). Today, WSDL-UDDI-SOAP technology is not the only Web Services technology; ebXML is one alternative. C# and Java are not the only programming languages for Web Services; we also have Perl (for example, SOAP-Lite is a package on top of Perl to implement Web Services) and some others. The paper and pencil exercises at the end of the first few chapters provide building blocks to build simple synchronous and asynchronous Web Services clients using Java or SOAP-Lite. Chapter 8, Web Services in Action: Case Study, brings these building blocks together as a solution and reinforces the big picture of Web Services technology discussed in the previous sections.

The Big Picture

Many Web Services books discuss WSDL, UDDI, and SOAP technology. Nevertheless, they have limited coverage on implementing Web Services, especially about how Web Services handles end-to-end security, mainframe or legacy systems interoperability, and cross-enterprise integration. There are Quality of Services design patterns available for enhancing performance and scalability of Web Services. Chapter 2, The Web Services Phenomenon and Emerging Trends, provides examples and business scenarios about partnering technology and business to produce end-to-end solutions using Web Services technology. Chapter 3, Web Services Technology Overview, introduces different technology components such as SOAP and ebXML (the parts), and discusses how these components can be integrated and interoperated to become a solution (the big picture) in the subsequent chapters on Web Services architecture, mainframe interoperability, cross-enterprise integration, and security. This book

emphasizes a vendor-independent architecture framework and reusable Web Services patterns, which can help create end-to-end solutions based on past experience and best practices easily.

Web Services Architecture

There are many published Web Services architectures available today. Yet most of them are vendor product architectures, rather than a generic Web Services reference architecture (such as W3C's Web Services architecture). It is important to differentiate vendor product architectures from a Web Services architecture framework and methodology. A generic Web Services architecture provides a repeatable and consistent way to design and deploy scalable, reliable Web Services, independent of the underlying vendor products. Chapter 4, Web Services Architecture and Best Practices, introduces a vendor-independent architecture framework to design Web Services and to bring different technology pieces together in a big, complete picture. It also discusses some best practices of delivering Web Services solutions with Quality of Services. Appendix B summarizes various Web Services architectures based on different vendor products, which can be good reference materials for the vendor-independent Web Services architecture framework depicted in Chapter 4.

Mainframe Interoperability

Many business functions or killer applications today are still provided by mainframe or legacy systems, which can be wrapped as a business service (using EJB or XML-RPC) for reuse by Open Systems (such as Java front-ends) and interoperability with other systems (such as Business-to-Business integration). This will require the use of some Java classes (such as CICS or VSAM connectivity jar files provided by Java Connector Architecture products) on the mainframe. This book introduces new underlying integration technology concepts using Web Services and discusses different alternatives on how to expose business functionality provided by mainframe or legacy systems. On an IBM z/OS mainframe, legacy systems running on an MVS or a VSE operating system can be invoked and interoperated from the Unix services of the same machine. It opens up new opportunities to integrate with legacy systems, as opposed to the traditional integration approach using proprietary middleware.

For customers who have less flexibility in maintaining legacy systems, legacy Cobol applications on MVS or VSE can be also cross-compiled as Java byte codes running under the Java Virtual Machine (JVM) of the same mainframe. EAI products can be used as the underlying adapter to interoperate with the mainframe as Web Services. This book also introduces other interoperability options, such as transcoding Cobol codes to Java components. It discusses the rationale and benefits of each interoperability option, as well as the constraints that are essential during the design stage.

Web Services technologies enable the reuse of business functionality provided by mainframes and legacy systems. They help protect past investments of business functionality developed on legacy and proprietary platforms and ease building killer ap-

plications based on existing customer and account data kept by these legacy systems. Killer applications create user stickiness by aggregating useful and timely customer and account information from different data sources that may run on legacy systems using Web Services as the technology enabler.

Web Services and EAI

EAI, B2Bi, and Web Services are technologies for integrating business corporations. EAI are traditional middleware products (for example, Message Oriented Middleware). B2Bi is specific for integrating business corporations with workflow. Some people generalize that Web Services technology is another EAI and can replace traditional EAI middleware. This is a misconception. There is a close synergy between Web Services and EAI technology. This book clarifies the relationship. It confirms that Web Services can be used for lightweight integration, but it cannot replace EAI. In addition, it discusses various patterns showing how Web Services technology is used for B2Bi, and how it would differentiate from EAI technology.

Web Services Security

Web Services security is probably the most fast-changing aspect in Web Services technology. There are a few challenges in this technology. Much of today's Web Services literature has covered SOAP message level security and the general security requirements addressed by SOAP security only. The infrastructure level security for a Web Services infrastructure or appliance is not well covered. Another challenge is that there is no cohesive coverage to bring the technology pieces of authentication (such as Liberty Alliance), authorization (such as SAML, XACML), traceability (such as tracking a SOAP message), data privacy/confidentiality/data integrity (such as XML encryption), availability (such as making SOAP server resilient), and non-repudiation (such as XKMS, digital signature) together under an end-to-end security framework.

This book clarifies the roles of different Web Services security technologies; HTTPS and digital signatures are the building blocks to providing authentication and non-repudiation. The recent WS-Security specification does not specify how network transport, infrastructural, or application security should be handled. The notion of XML Trust Service attempts to provide a broader perspective of different aspects of XML Web Services security, such as key registration and authentication (XKMS), entitlement and identity (SAML), and fine-grained data access rights (XACML).

This book also discusses the outlook of recent Web Services security initiatives led by major vendors, and their underlying technology. WS-Security used to be Microsoft-proprietary. IBM has recently partnered with Microsoft and VeriSign to support it under the "Web Services Security Roadmap" (*http://www-106.ibm.com/developerworks/webservices/library/ws-secmap/*). This initiative supersedes previous IBM and Microsoft XML Security specifications. Sun and many other vendors now also support WS-Security. Security continues to be a problem area for Microsoft platforms, despite their commitment to address it. Microsoft PASSPORT has been criticized by Microsoft

opponents for its design objectives and implementation, which will lock in customers and intrude on their privacy.

Project Liberty (*http://www.projectliberty.org/*) is a market response to PASSPORT and provides a choice of network identity solutions. It provides a platform-independent architecture for Single Sign-on and network identity management to support authentication and authorization with SAML. There is a misconception that Liberty Alliance is competing with WS-Security. This book clarifies that misconception and discusses the role of Liberty in identity management, which complements the message-level and application-level security provided by the WS-Security specification.

Finally, this book proposes a security framework to design end-to-end Web Services security. It addresses security at different levels, from network level, infrastructure level, message level, to application level. It reinforces the idea of the big picture, bringing different security technologies together.

The Frontiers of Web Services Technology

Web Services technology initiatives are fast-changing. The frontiers of Web Services technology also extend to wireless Web Services (such as enabling the mobile device to support SOAP messaging via kSOAP and J2ME). There are generally one or more events every week. The challenge to many developers and architects is how to manage these fast-changing technologies. We have seen the convergence of Web Services Security from SOAP-SEC, WS-Security, and various security specifications, followed by the convergence of XLANG and WSFL into BPEL4WS (Business Process Execution Language for Web Services, or *http://www.106.ibm.com/developerworks/library/ws-bpel/*). We also see the recent Web Services Choreography Interface (or WSCI) specification, which defines business process orchestration for Web Services similar to BPEL4WS. This book suggests some strategies to manage these new emerging technologies by reviewing the forthcoming standards specification and partnering with technology thought leaders to reduce implementation risks.

There are "no new things under the sun." New technology and tools emerge from time to time. This book identifies and analyzes several factors that foster the fast-changing Web Services technologies. It is important to understand the basics, the limitations, and the future direction of Web Services to prepare for the next frontiers. Apache Axis is a next-generation SOAP engine. It is based on IBM's Web Services Toolkit and has included utilities to generate WSDL. It is a good penetration strategy for vendors to incorporate their technology into Open Sources. It is also a good tactic for developers and architects to keep track of some leading Open Source tools for Web Services because they may be embedded into commercial Web Services products soon.

The developer dictates the market. Web Services products that can win developer support and dominate the developer desktop may become the next dominant Web Services technologies. The success of Windows-based product has led the .NET marketing strategy. Sun's Java Web Services Developer Pack™ (JWSDP) is an all-in-one developer kit, a response to Web Services market. All-in-one will become a key element for future developer kits.

Apart from watching new products and tools, a list of forthcoming Web Services specifications (such as JSRs) are identified and may be the next frontiers of Web Services technology.

1.2.4 Values of the Book

This book focuses on Sun's Web Services technology (for example, JWSDP and JAX) with a Sun ONE architecture and J2EE flavor. The majority of Web Services books available today introduce WSDL-UDDI-SOAP programming exercises and concepts. This book introduces the frontiers of Web Services technology and the steps for designing the entire application with scalability and availability. Experience says that putting the technology to use in real-life examples can make learning more effective, so this book focuses on doing that.

It provides some program codes and small hands-on labs for illustration, but it does not replicate other beginner's Web Services books or SOAP programming-level books. Thus, it is a good accompaniment for other Web Services books. This book also includes extensive pointers to URLs and to other Web Services resources.

1.3 Bringing the Pieces Together

1.3.1 Service Requester–Service Provider Relationship

Web Services technology can be described in terms of a Service Requester–Service Provider relationship (refer to Figure 1–1). The Service Provider runs business services from their systems locally and remotely. Business services provided can be found in a Service Registry. In order to register and publish the business service in the Service Registry, the Service Provider defines (authors) service description and configuration information (such as configuration files or WSDL—Web Services Description Language) and then codes the implementation (Service Implementation). The Service Implementation may be from existing legacy system functionality via Remote Procedure Calls or new applications.

The Service Requester is a consumer of business services. This can be the end-user (as in Business-to-Consumer) or server (as in Business-to-Business scenario). The Service Requester finds the business services from the Service Registry via a Service Proxy (such as an Apache SOAP server). Upon a successful search, the Service Registry, which may be provided by the same Service Provider or by a public Service Registry node, fetches the appropriate service description (for example, WSDL) and returns the service end-points (that is where the business service is located) to the Service Requester. Then the Service Requester can "bind" the business service to the actual service end-point or location.

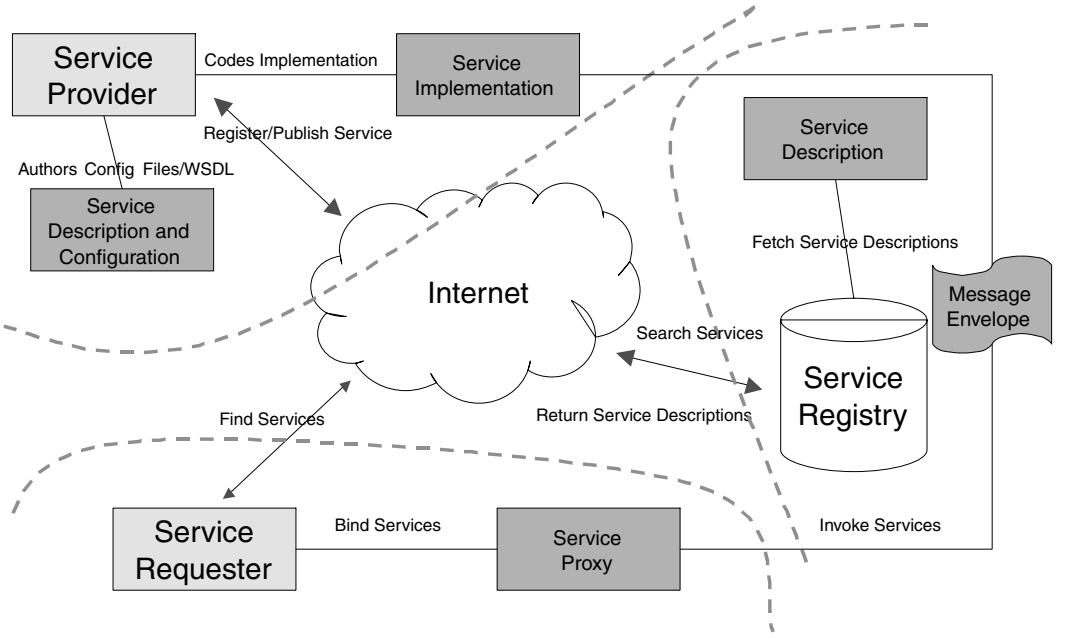

Figure 1–1 Web Services Consumer–Service Provider Relationship

In summary, the Service Provider uses WSDL to describe the business service and configuration to implement business services. Service descriptions (such as WSDL) are then published in the Service Registry (either UDDI or ebXML).

The Service Provider also uses SOAP technology (such as SOAP Proxy) to wrap existing legacy system functionality as reusable business services. The Service Requester discovers business services dynamically via a SOAP Proxy from the Service Registry, and binds the business services to the actual URI (Universal Resource Identifier) locally or remotely. The business services are encapsulated in XML messages using a SOAP message envelope. This enables easier data manipulation and processing using XML-based products and applications.

1.3.2 Software Tools

Table 1–1 provides a list of the software tools that you will need to work with the examples and labs throughout this book.

Table 1–1 List of Software Tools Used in This Book

Software Tools	Description	Where to Download
Java Web Services Developer Pack	Also known as JWSDP, this is an all-in-one development toolkit with Apache Tomcat 4.1.2, Apache SOAP 2.2, and JAX packs. This is the core component for this book.	*http://java.sun.com/webservices/downloads/webservicespack.html*
Axis	New generation Apache SOAP engine. This is a good complementary product to JWSDP, and is essential to run many of the samples in this book.	*http://xml.apache.org/axis/index.html*
Trust Services Integration Kit	Also known as TSIK, this product provides XKMS and WS-security support. This component is essential to run the Case Study sample programs.	*http://www.xmltrustcenter.org/developer/verisign/tsik/download.htm*
jSAML Toolkit	A pre-Liberty SAML-based toolkit for implementing Single Sign-on. You'll need to register to download this tool. Select download from the "Product Downloads" on the left hand column at this Web site.	*http://www.netegrity.com/products/index.cfm?leveltwo=JSAML&levelthree=download*
SOAP-Lite	This product lets you access SOAP from a Perl client. Requires Perl version 5.0 or above. This is instrumental to illustrate that we do not require a Java client to invoke Web Services.	*http://www.soaplite.com/*

1.4 Tour of the Book

The map in Figure 1–2 shows the conceptual structure of the book, which is targeted for two different purposes: strategy, or management perspective, and technical orientation.

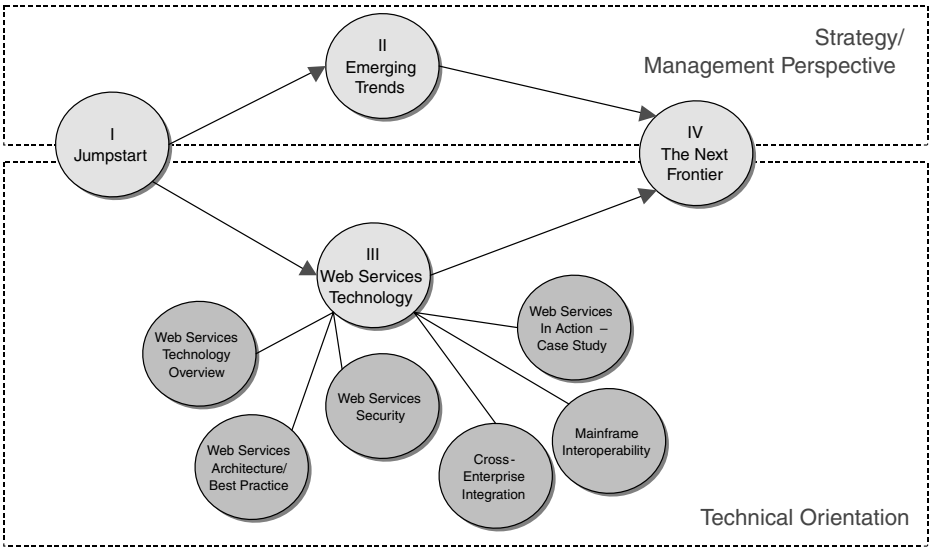

Figure 1–2 Tour of the Book

Chapter 2: The Web Services Phenomenon and Emerging Trends This chapter provides an impact analysis of the major business drivers of the changing economic landscape in banking, capital markets, and B2B Exchanges. New intermediaries now challenge traditional business. Services have become more competitive with new mergers and consolidations. There is a growing awareness of the importance of a reliable and scalable business and solution architecture. This chapter provides business scenarios for implementing Web Services, illustrated by establishing a business case for Web Services.

Chapter 3: Web Services Technology Overview This section gives a refresher hands-on lab for various Web Services enabling technology (including UDDI, SOAP, and ebXML), technology stack, and the latest development tools. It outlines how to enable your applications for Web Services in four steps. A sampler of the market space, existing Web Services vendors, and guidelines for choosing appropriate tools are discussed.

Chapter 4: Web Services Architecture and Best Practices Several vendors' Web Services architectures are discussed in this chapter. A discussion of what architecture framework is suitable to build different large-scale business solutions is offered. A service-based reference architecture is depicted, followed by some architecture principles and design patterns for best practices.

Chapter 5: Mainframe Integration and Interoperability This chapter introduces different ways for legacy mainframe COBOL applications to interoperate with open systems, compares the pros and cons of using Web Services, and discusses when to use Web Services. An overview of existing market products for enabling mainframe interoperability is also offered.

Chapter 6: Enterprise and Cross-Enterprise Integration XML Web Services has some competitive advantages in enterprise and cross-enterprise integration. Different e-Business enterprise and cross-enterprise integration design patterns are discussed. A comparison is made between Web Services-based and EAI-based integration, and when to use each.

Examples will be elaborated in Chapter 8, Web Services in Action: Case Study.

Chapter 7: Web Services Security End-to-end Web Services security is the key to securing financial transactions. Most solutions focus security in specific tiers and layers but do not put it in an end-to-end perspective. Web Services security is often viewed as insecure due to lack of understanding. This chapter discusses the key components of PKI-based and Web Services security infrastructure, buy versus build (for example, should we outsource digital certificate management to an external trust service provider?), what to use, and when to use. It will also suggest ways to enable leveraging existing security infrastructure as Web Services.

Several Web Services security initiatives are discussed in this chapter, such as XML security (XKMS, SAML, XACML,WS-Security) and Project Liberty.

Chapter 8: Web Services in Action: Case Study This chapter goes through a case study of how Web Services is designed and implemented with the real-life example of a Foreign Exchange service. It illustrates a small-scale Use Case requirements analysis and a SunTone architecture design. The demonstration program is based on JWSDP to illustrate synchronous and asynchronous Web Services with secure message services.

Chapter 9: The Next Frontiers This is an industry survey of different emerging technology variants and how they may impact different industries over the next few years.

Appendices

Appendix A—Resources and References

Appendix B—Some Web Services Vendors

Appendix C—Demo Environment Set Up

1.5 Special Features

1.5.1 Building a Complete Example

Throughout the book, a Foreign Exchange trading Web application example will be built. To build a complete application (see Figure 1–3), you can follow the examples and concepts in the book to wrap an existing business service as Web Services (for example, using JWSDP's wsdeploy or Axis), publish it in a Service Registry (for example, using JAXR), expose the business service via a Message Provider (for example, using JAXM), and/or parse data from the business partners using JAXP and XSLT.

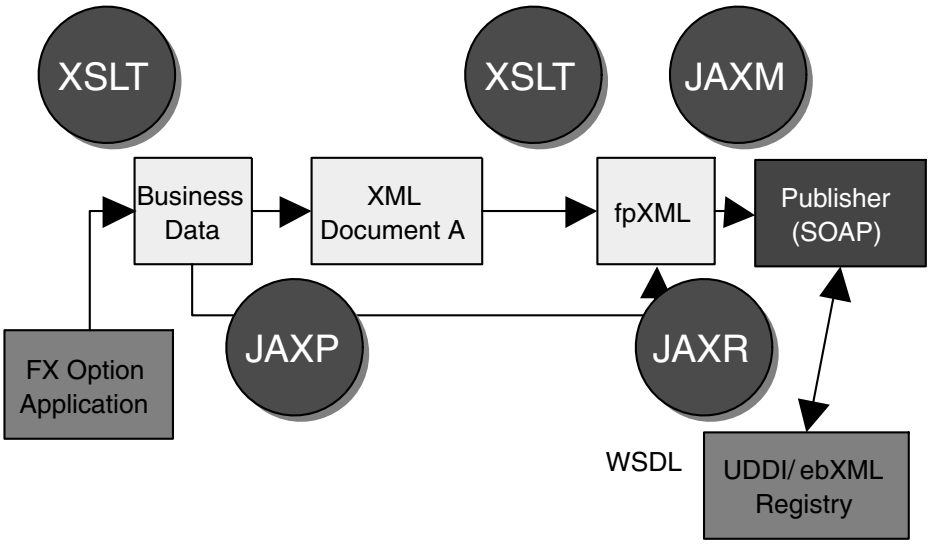

Figure 1–3 Building a Complete Example

1.5.2 Chapter Overview

The "Chapter Overview" highlights the main ideas of each chapter for both quick preview and recapitulation.

For Example:

- Web Services, units of business services and components, is positioned as the next IT strategy for enterprises. The enabling technologies are built on top of Java and XML. They are also referenced as "Java Web Services" or "XML Web Services."

- The Web Services technology stack includes the transport layer (such as SOAP), a service description language (such as WSDL), and transaction routing and service negotiation (such as BP).
- Web Services has many associations with CORBA. Some perceive Web Services technology is a reincarnation of Application Service Provider and CORBA.
- Microsoft's version of Web Services is .NET, with C# as one key development language. Sun, IBM, and other technology vendors are using Java as the key underlying technology.

1.5.3 Objectives

The "Objectives" section identifies the learning objectives of each chapter to help you appreciate why and how the technology is used.

For Example:

- To identify the root and derivation of Web Services technologies
- To have an appreciation of the underlying Web Services technologies, their benefits, and when to use them
- To establish some guidelines in choosing a Web Services developer tool and platform in the marketplace
- To illustrate four steps to enable a simple application for Web Services
- To illustrate what Web Services solutions are available on the market, along with their benefits and limitations

1.5.4 Best Practices and Pitfalls

The "Best Practices and Pitfalls" section suggests industry-best practices and pitfalls for the specific Web Services technology. You can use this section as a checklist for good design practice or pitfalls to avoid.

For Example:

Pitfalls

- Do not try to have too many dependencies or items that require longer lead time (for example, do not try to SOAP-enable a mainframe platform).
- Do not start implementation without senior management support.
- Do not start a pilot using a mission-critical functionality.
- Do not involve a big team in the pilot. Start with a small team.

1.5.5 Case Study

The "Case Study" section illustrates how the technology is used, its critical success factors, and its implementation pitfalls.

For Example:

Challenges

A medium-size listed financial institution in Hong Kong wants to reduce its operating costs associated with supporting legacy systems on a mainframe. The corporation has been innovative and reputable in providing securities trading, retail banking, and credit card services to the local market.

The CEO is concerned about implementing new technology for technology's sake and believes that not all legacy systems need to be migrated to UNIX, as they are fairly stable.

Is it relevant to introduce Web Services technology to meet its challenge?

1.5.6 Paper and Pencil

The "Paper and Pencil" section provides some study questions and simple hands-on exercises for practice.

For Example:

Hands-on Lab. Download the Web Services Developer Pack from *http://java.sun.com.* Install and configure it according to the Installation Guide. Based on the sample sniplet tradeOrder.xml in the text below (see Figure 1–4), write your first JAXM-SOAP program with your favorite editor or your favorite Integrated Developer Environment to send the following XML message:

```
    <trade>
  <tradeHeader>
    <partyTradeIdentifier>
      <partyReference href = "#XYZ"/>
    </partyTradeIdentifier>
    <partyTradeIdentifier>
      <partyReference href = "#ABC"/>
    </partyTradeIdentifier>
    <tradeDate>2002-01-15</tradeDate>
   </tradeHeader>
```

Figure 1–4 Trade Order in XML (tradeorder.xml)

```
    <fxSimpleOption>
      <productType>Nondeliverable Option</productType>
      <buyerPartyReference href = "#XYZ"/>
      <sellerPartyReference href = "#ABC"/>
      <expiryDateTime>
         <expiryDate>2002-04-09</expiryDate>
         <hourMinuteTime>1000</hourMinuteTime>
         <businessCenter>USNY</businessCenter>
      </expiryDateTime>
      <exerciseStyle>European</exerciseStyle>
     </fxSimpleOption>
    </trade>
```

Figure 1–4 Trade Order in XML (tradeorder.xml)—*continued.*

1.5.7 Resources and References

The "Resources and References" section lists useful and relevant articles, magazines, books, and URLs for further reading. URLs provide a frequently updated link to major Web Services resources where books may become outdated.

For Example:

Web Services Resource Web Sites

http://www.webservices.org

http://www.theserverside.com/home/index.jsp

Web Services Overview

John Hagel III and John Seely Brown. *"Your Next IT Strategy."* Harvard Business Review, October 2001.

IBM Web Services Overview Team. *"Web Services Architecture Overview."* IBM, September 2000. *http://www-106.ibm.com/developerworks/library/w-ovr/*

Lawrence Wilkes. *"Web Services—Right Here, Right Now."* IBM, 2002. *http://www.3.ibm.com/software/solutions/webservices/pdf/cbdi.pdf*