Describe REST Web services with WSDL 2.0

A how-to guide

Lawrence Mandel (lmandel@ca.ibm.com)   29 May 2008
Rational Software Developer
IBM

At their core, Web services define a mechanism for machine-to-machine interaction using a network and XML. A key component of a Web service is a formal description with Web Services Description Language (WSDL). Until recently there was no formal language to describe REpresentational State Transfer (REST) Web services—now there's WSDL 2.0. This article introduces you to REST and WSDL 2.0, and walks you through creating a WSDL 2.0 description of a REST Web service.

Introduction

The term *Web services* is typically associated with operation- or action-based services using SOAP and the WS* standards, such as WS-Addressing and WS-Security. The term *REST Web services* generally refers to a resource-based Web services architecture that uses HTTP and XML. Each of these architectural Web service styles has its place, but until recently, the WSDL standard didn't equally support both styles. The WSDL 1.1 HTTP binding was inadequate to describe communications with HTTP and XML, so there was no way to formally describe REST Web services with WSDL. The publication of WSDL 2.0, which was designed with REST Web services in mind, as a World Wide Web Consortium (W3C) recommendation means there is now a language to describe REST Web services.

REST

REST is an architectural style that treats the Web as a resource-centric application. Practically, this means each URL in a RESTful application represents a resource. The URLs are also easy to understand and remember. For example, a bookstore might define the URL http://www.bookstore.com/books/ for a list of books it sells and the URL http://www.bookstore.com/books/0321396855/ for details about a specific book with the ISBN 0321396855. This is in contrast to action-centric applications, which typically have long, cryptic URLs describing actions to perform, such as http://www.bookstore.com/action/query?t=b&id=11117645532&qp=0321396855. Query parameters are used to filter the results. Using the same bookstore example, specifying

the subject parameter restricts the book list to books about a specific subject. For example, the URL http://www.bookstore.com/books/?subject=computers/eclipse returns a list of books about the Eclipse platform.

Dr. Roy Fielding coined the term REST in his Ph.D. dissertation, where he referred to "hypermedia as the engine of application state." This means that a resource is expected to contain hyperlinks. These hyperlinks are the method by which a transition can take place that changes the resource state or transfers to another resource. While hyperlinks are commonplace in (X)HTML applications meant to be used by humans, they have not typically been used in XML, which is meant to be consumed by machines. Like (X)HTML, REST Web services make use of hyperlinks in XML.

Traditional Web applications access resources using HTTP GET or POST operations. In contrast, RESTful applications access resources following the create, read, update, and delete (CRUD) style using the full range of HTTP verbs (POST, GET, PUT, and DELETE).

There's one more key component of a REST application: RESTful applications should be stateless. This means in a REST application no session state is stored on the server. All of the information needed to satisfy the request is carried in the request message itself. A client can therefore cache a representation of a resource, which can significantly improve the application's performance, where a service explicitly allows it. To learn more about REST, see the Resources section at the end of this article.

Returning other content types

Web services generally return data as XML, but there are other content types that are useful to service consumers. For example, it's generally preferable for Asynchronous JavaScript + XML (Ajax) applications to receive JavaScript Object Notation (JSON) data, and human consumers likely want to receive the data as HTML, which can be rendered in their browsers.

In the HTTP world, the selection of the data format is known as *content type negotiation*. In content type negotiation, the client specifies the preferable content types and those that are acceptable, then the service responds with the most appropriate content type. This means client A can request the data from a Web service as XML, and client B can request the data as JSON or some other type.

To give clients the ability to request a content type, construct your service so that it makes use of the built-in HTTP Content-Type header: `Content-Type: text/html`.

An alternative approach for applications that don't understand the Content-Type header is to specify a parameter, such as `cType`, on the URL, as in `http://www.bookstore.com/books/?cType=application/xml`.

WSDL and REST

A WSDL description contains all the details of a Web service, including:

- The service's URL.
- The communication mechanisms it understands.
- What operations it can perform.
- The structure of its messages.

Clients can use these details to interact with a service.

No doubt, one significant reason why REST Web services have to this point not made use of WSDL is that the WSDL 1.1 HTTP binding was inadequate to describe them.

WSDL 2.0 was declared a W3C recommendation in June 2007. This second version of WSDL was created to address issues with WSDL 1.1, many of which had been identified by the Web Services Interoperability (WS-I) organization. In addition, WSDL 2.0 has good support for HTTP bindings.

WSDL is an XML language to formally describe a Web service. Consider a Web service's WSDL description its API contract with clients. The WSDL description specifies the address, allowable communication mechanisms, interface, and message types of a Web service. In short, a WSDL description provides all the information a client needs to use a Web service.

WSDL's usefulness extends beyond its use as an API contract. Being a formal definition, WSDL can be consumed by Web services tools to perform actions, such as:

- Generate client and service stubs in various languages.
- Publish a Web service.
- Dynamically test a Web service.

The majority of Web services tools include support for WSDL 1.1, and support for WSDL 2.0 is growing. The Apache Web services project contains two subprojects that currently support WSDL 2.0. Woden is a Java™-based WSDL 2.0 validating parser. The project also contains an XSL Transformation (XSLT) WSDL 2.0 pretty printer that provides a more human-readable form of a WSDL document. Axis2, also from Apache, is a popular Web service runtime engine capable of generating Java client and server stubs from a WSDL 2.0 document.

Describe a REST Web service with WSDL 2.0

The remainder of this article takes you through the steps needed to create a WSDL 2.0 description for a REST Web service using the following simple example scenario.

You run a bookstore, which has a creative URL: http://www.bookstore.com. You've previously created two REST Web services:

- The **book list service** retrieves the list of the books that you sell in your store.
- The **book details service** retrieves the details about a specific book.

The information is returned in XML documents. Take a look at the details about the services:

The URL of the book list service is http://www.bookstore.com/books/. This list of books you sell is quite large, as you're an established retailer. So you've provided the following list of query parameters that clients can use to filter the results:

- Author
- Language

- Publisher
- Subject
- Title

For example, the URL http://www.bookstore.com/books/?subject=computers/eclipse returns the list of computer books about Eclipse, as shown in Listing 1.

Listing 1. Response from the book list service

```
<booklist:bookList xmlns:booklist="http://www.bookstore.org/booklist/xsd"
  xmlns:book="http://www.bookstore.org/book/xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bookstore.org/booklist/xsd booklist.xsd
                      http://www.bookstore.org/book/xsd book.xsd">

  <booklist:book url="http://www.bookstore.com/books/0321442598"
    title="BIRT: A Field Guide to Reporting"/>
  <booklist:book url="http://www.bookstore.com/books/0321205758"
    title="Contributing to Eclipse: Principles, Patterns, and Plug-Ins"/>
  <booklist:book url="http://www.bookstore.com/books/0321245873"
    title="Eclipse AspectJ: Aspect-Oriented Programming with AspectJ and the..."/>
  <booklist:book url="http://www.bookstore.com/books/0321288157"
    title="Eclipse Distilled"/>
  <booklist:book url="http://www.bookstore.com/books/0131425420"
    title="Eclipse Modeling Framework"/>
  <booklist:book url="http://www.bookstore.com/books/0321334612"
    title="Eclipse Rich Client Platform: Designing, Coding, and Packaging Java..."/>
  <booklist:book url="http://www.bookstore.com/books/0321396855"
    title="Eclipse Web Tools Platform: Developing Java Web Applications"/>
  <booklist:book url="http://www.bookstore.com/books/032142672X"
    title="Eclipse: Building Commercial-Quality Plug-Ins (2nd Edition)"/>
  <booklist:book url="http://www.bookstore.com/books/0321443853"
    title="Integrating and Extending BIRT"/>
  <booklist:book url="http://www.bookstore.com/books/0321268385"
    title="Official Eclipse 3.0 FAQs"/>
  <booklist:book url="http://www.bookstore.com/books/0321256638"
    title="Official Eclipse 3.0 FAQs"/>
</booklist:bookList>
```

The URL of the book details service is http://www.bookstore.com/books/*ISBN_NUMBER*, where *ISBN_NUMBER* should be replaced with the ISBN for a specific book. For example, the URL http://www.bookstore.com/books/0321396855 returns the details of my book, *Eclipse Web Tools Platform*, as shown in Listing 2 (see Resources for a link to the book's Web site).

Listing 2. Response from the book details service

```
<book:book xmlns:book="http://www.bookstore.org/book/xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bookstore.org/book/xsd book.xsd">
  <book:title>Eclipse Web Tools Platform: Developing Java Web Applications</book:title>
  <book:author>
    <book:firstName>Naci</book:firstName>
    <book:lastName>Dai</book:lastName>
  </book:author>
  <book:author>
    <book:firstName>Lawrence</book:firstName>
    <book:lastName>Mandel</book:lastName>
  </book:author>
  <book:author>
    <book:firstName>Arthur</book:firstName>
    <book:lastName>Ryman</book:lastName>
```

```
  </book:author>
  <book:overview>
Discover Eclipse Web Tools Platform (WTP), the new end-to-end toolset for Java-based
Web development. The WTP seamlessly integrates all of the tools today's Java Web
developer needs. Eclipse WTP is both an unprecedented open source resource for
working developers and a powerful foundation for state-of-the-art commercial products.
This book offers in-depth descriptions of every tool included in WTP, introducing powerful
capabilities never before available in Eclipse. The authors cover the entire
Web-development process -- from defining Web application architectures and
development processes through testing and beyond. And if you're seeking to extend WTP,
this book provides an introduction to the platform's rich APIs. The book also
  </book:overview>
  <book:pages>752</book:pages>
  <book:publisher>Addison-Wesley Professional</book:publisher>
  <book:language>English</book:language>
  <book:isbn-10>0321396855</book:isbn-10>
  <book:isbn-13>978-0321396853</book:isbn-13>
  <book:price>54.99</book:price>
</book:book>
```

In the following sections, you learn how to create a WSDL 2.0 description of the book list service using the details outlined above. The book details service, while useful in the scenario, doesn't have a structurally different WSDL description, so it's not covered in the article. However, the book list and book details WSDL 2.0 descriptions and the sample documents from Listing 1 and Listing 2 are available in the Downloads section at the end of this article.

Before getting to the WSDL description of the book list service, let's take a look at the WSDL 2.0 basics.

The basics of WSDL 2.0

WSDL 2.0 is an XML language with the core namespace http://www.w3.org/ns/wsdl. The root element of a WSDL 2.0 document is the `description` element. There are four child elements of `description` that together encapsulate all of the details about a Web service:

- `types`
- `interface`
- `binding`
- `service`

A skeleton WSDL 2.0 document is shown in Listing 3.

Listing 3. Skeleton WSDL 2.0 document

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl">
  <wsdl:types/>
  <wsdl:interface/>
  <wsdl:binding/>
  <wsdl:service/>
</wsdl:description>
```

What's changed in WSDL 2.0?

The structure of a WSDL 2.0 document differs in some pretty noticeable ways from WSDL 1.1. The following list of differences isn't by any means complete, but it does contains a few of the more notable differences:

- The root element has changed from `definitions` to `description`.
- The `portType` element has been replaced with the `interface` element to better reflect its use.
- The `message` element no longer exists as a global element. Message descriptions are now encapsulated in the `interface` element.
- A binding is now reusable. It doesn't need to be associated with a specific interface. The association can be made in the service declaration.
- XML schemas, which in WSDL 1.1 could be imported in a number of ways, are imported using an `xsd:import` element as a direct child of the `types` element.

The `types` element contains all of the XML schema element and type definitions that describe the Web service's messages. WSDL 2.0 is open for the use of other type systems but practically is only used with XML schema.

The `interface` element defines the Web service operations, including the specific input, output, and fault messages that are passed, and the order in which they are passed.

The `binding` element defines how a client can communicate with the Web service. In the case of REST Web services, a binding specifies that clients can communicate using HTTP.

The `service` element associates an address for the Web service with a specific interface and binding.

WSDL 2.0 defines two other namespaces of interest with respect to REST Web services:

- HTTP namespace http://www.w3.org/ns/wsdl/http, which includes the HTTP binding elements
- WSDL extensions namespace http://www.w3.org/ns/wsdl-extensions, which includes definitions of three attributes: two used to associate a hyperlink in an XML document with a Web service description and the third to describe a Web service operation as safe

All of the WSDL elements discussed in this section are shown in more detail in the following sections.

Address the book list service

As a reminder, the URL for the book list service is http://www.bookstore.com/books/. To address the service, you use the WSDL `service` element, which requires at least one `endpoint` child element. The `endpoint` element's `address` attribute is used to specify the service's URL, as shown in Listing 4. The `endpoint` element is also used to associate a binding with the service with the `binding` attribute. The `service` element in turn associates an interface with the service with the `interface` attribute. You create the interface and binding in the following sections, so you can leave these attribute values blank for now.

The book list service's WSDL 2.0 document also requires a target namespace, so define the namespace http://www.bookstore.org/booklist/wsdl.

Listing 4. The book list service definition

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
   targetNamespace="http://www.bookstore.org/booklist/wsdl"
   xmlns:tns="http://www.bookstore.org/booklist/wsdl">

  <wsdl:service name="BookList" interface="">
    <wsdl:documentation>
       The bookstore's book list service.
    </wsdl:documentation>
    <wsdl:endpoint name="BookListHTTPEndpoint"
       binding=""
       address="http://www.bookstore.com/books/">
    </wsdl:endpoint>
  </wsdl:service>
</wsdl:description>
```

Talking HTTP with the book list service

The book list service's binding definition needs to specify that the service communicates using HTTP. To do this, specify the value http://www.w3.org/ns/wsdl/http for the `binding` element's `type` attribute.

The binding can also optionally reference an interface. Leave the `interface` attribute blank; you create it in the next section. If an interface is associated with the binding, the `binding` element can optionally declare a child `operation` element that mirrors the interface `operation` element. You need to create a stub `operation` element and fill in this reference after creating the interface.

There are four HTTP communication verbs:

- GET
- PUT
- POST
- DELETE

The book list service is a read request and, therefore, communicates with HTTP GET. Set the GET verb on the `operation` element using the HTTP `method` attribute from the WSDL 2.0 HTTP namespace. To use this attribute, you first need to declare the namespace http://www.w3.org/ns/wsdl/http on the description element.

The book list service's HTTP binding declaration can be seen in Listing 5. Update the `endpoint` element's binding reference to reference the `tns:BookListHTTPBinding` now that you've declared the binding.

Listing 5. The book list binding definition

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
   targetNamespace="http://www.bookstore.org/booklist/wsdl"
   xmlns:tns="http://www.bookstore.org/booklist/wsdl"
   xmlns:whttp="http://www.w3.org/ns/wsdl/http">

  <wsdl:binding name="BookListHTTPBinding"
    type="http://www.w3.org/ns/wsdl/http"
    interface="">
    <wsdl:documentation>
       The RESTful HTTP binding for the book list service.
    </wsdl:documentation>
```

```
      <wsdl:operation ref="" whttp:method="GET"/>
  </wsdl:binding>

  <wsdl:service name="BookList" interface="">
    <wsdl:documentation>
       The bookstore's book list service.
    </wsdl:documentation>
    <wsdl:endpoint name="BookListHTTPEndpoint"
       binding="tns:BookListHTTPBinding"
       address="http://www.bookstore.com/books/">
    </wsdl:endpoint>
  </wsdl:service>
</wsdl:description>
```

Define the book list service operation

So far you've learned how to address and communicate with the book list Web service. Next you specify the book list service operation, which describes what the book list service does.

The `interface` element and its child `operation` element are used to define a service's operations. In the case of the book list service, you define a single operation, `getBookList`, that responds to requests with a book list.

Next, specify three attributes on the `operation` element:

- **pattern:** Used to specify the message exchange pattern (MEP) for the operation. The MEP defines the sequence of messages in the operation and their direction. In this case, specify the value http://www.w3.org/ns/wsdl/in-out to indicate that the service receives one input message—the request for the book list—and sends one output message—the book list. To support this MEP, specify `input` and `output` child elements of the `operation` element. These elements are used to reference the XML schema elements that define the message structures, which are created in the next section.
- **style:** Used to specify additional information about an operation. Specify the value http://www.w3.org/ns/wsdl/style/iri, which places restrictions on the `input` element content, such as requiring that it only use XML schema elements.
- **wsdlx:safe:** From the WSDL extensions namespace, this attribute declares that this operation is idempotent. This type of operation doesn't modify the resource and can therefore be called many times with the same results. To make use of this element, declare the WSDL extensions namespace http://www.w3.org/ns/wsdl-extensions on the `description` element.

You can find the predefined MEPs, styles, and the `safe` attribute definition in "WSDL 2.0 Part 2: Adjuncts" (see Resources for a link).

Check out the book list service's interface declaration in Listing 6. You can update the `service` and `binding` elements' interface references and the binding `operation` element's interface operation reference now that you've declared the interface and operation.

Listing 6. The book list interface definition

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
   targetNamespace="http://www.bookstore.org/booklist/wsdl"
   xmlns:tns="http://www.bookstore.org/booklist/wsdl"
   xmlns:whttp="http://www.w3.org/ns/wsdl/http"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">
```

```
 <wsdl:interface name="BookListInterface">
   <wsdl:operation name="getBookList"
     pattern="http://www.w3.org/ns/wsdl/in-out"
     style="http://www.w3.org/ns/wsdl/style/iri"
     wsdlx:safe="true">
    <wsdl:documentation>
        This operation returns a list of books.
    </wsdl:documentation>
    <wsdl:input element=""/>
    <wsdl:output element=""/>
   </wsdl:operation>
 </wsdl:interface>

 <wsdl:binding name="BookListHTTPBinding"
    type="http://www.w3.org/ns/wsdl/http"
    interface="tns:BookListInterface">
   <wsdl:documentation>
      The RESTful HTTP binding for the book list service.
   </wsdl:documentation>
   <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
 </wsdl:binding>

 <wsdl:service name="BookList" interface="tns:BookListInterface">
   <wsdl:documentation>
      The bookstore's book list service.
   </wsdl:documentation>
   <wsdl:endpoint name="BookListHTTPEndpoint"
      binding="tns:BookListHTTPBinding"
      address="http://www.bookstore.com/books/">
   </wsdl:endpoint>
 </wsdl:service>
</wsdl:description>
```

Define the book list service operation messages

The book list Web service has two messages: an input message and an output message. You need to describe specific message structures so that clients know what message to send to the service and what message to expect from the service.

WSDL 2.0 supports multiple type systems for describing the message content, but XML schema is the only one in use. This section doesn't cover the details of XML schema. XML schema is used in many other applications, like WSDL 1.1, and there are many good articles about it. This section highlights how to use XML schema for the book list REST Web service and how to use additional attributes defined by WSDL 2.0 to annotate a schema attribute.

To create the two messages for the book list REST Web service, you need to create two global elements:

- `getBookList` represents the input message. It contains a sequence of elements, including each of the query parameters you allow on the service, namely author, title, publisher, subject, and language. The content of the `getBookList` element is restricted to elements only, because you specified the IRI style for the interface operation.
- `bookList` represents the output message. It contains a sequence of `book` elements. Each `book` element contains `title` and `url` attributes. The `title` attribute should be self explanatory. The `url` attribute is a link to a book details REST Web service, which returns the details for the specific book.

Your definition of the `url` attribute includes two attributes from the WSDL extensions namespace. The attributes `wsdlx:interface` and `wsdlx:binding` identify the specific WSDL 2.0 interface and binding for the service. Tools can use this semantic information to automatically discover the service. To make use of these attributes, specify the WSDL extensions namespace on the `schema` element. Also, include the book details service namespace from its WSDL 2.0 description to reference the interface and binding for the service.

The XML schema for the book list service is shown in Listing 7. You can get the book details service description in the Downloads section.

Listing 7. XML schema for the book list service

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.bookstore.org/booklist/xsd"
   xmlns:tns="http://www.bookstore.org/booklist/xsd"
   xmlns:booksvc="http://www.bookstore.org/book/wsdl"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
   elementFormDefault="qualified">

  <element name="getBookList" type="tns:getBookListType">
   <annotation>
     <documentation>
        The request element for the book list service.
     </documentation>
    </annotation>
  </element>

  <element name="bookList" type="tns:bookListType">
    <annotation>
      <documentation>
        The response element for the book list service.
      </documentation>
    </annotation>
  </element>

  <complexType name="getBookListType">
    <sequence>
      <element name="author" type="string" minOccurs="0" maxOccurs="unbounded"/>
      <element name="title" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="publisher" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="subject" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="language" type="string" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="bookListType">
    <sequence>
      <element name="book" type="tns:bookType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="bookType">
    <attribute name="title" type="string"/>
    <attribute name="url" type="anyURI"
       wsdlx:interface="booksvc:BookInterface"
       wsdlx:binding="booksvc:BookHTTPBinding"/>
  </complexType>
</schema>
```

To reference the input and output elements declared in the XML schema, you have to import the schema into your WSDL document. To import a schema, you use a schema `import` element in

the types section, as shown in Listing 8. You also need to add references to the `getBookList` and `bookList` elements in the interface operation's `input` and `output` elements, and add the XML schema book list schema's namespace declarations to the `description` element.

Listing 8 shows the complete WSDL 2.0 description of the book list REST Web service. Get the description of this service and the book details REST Web service in the Downloads section.

Listing 8. WSDL 2.0 description of the book list REST Web service

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
   targetNamespace="http://www.bookstore.org/booklist/wsdl"
   xmlns:tns="http://www.bookstore.org/booklist/wsdl"
   xmlns:whttp="http://www.w3.org/ns/wsdl/http"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:msg="http://www.bookstore.org/booklist/xsd">
  <wsdl:documentation>
     This is a WSDL 2.0 description of a sample bookstore service
     listing for obtaining book information.
  </wsdl:documentation>

  <wsdl:types>
    <xs:import namespace="http://www.bookstore.org/booklist/xsd"
       schemaLocation="booklist.xsd"/>
  </wsdl:types>

  <wsdl:interface name="BookListInterface">
    <wsdl:operation name="getBookList"
       pattern="http://www.w3.org/ns/wsdl/in-out"
       style="http://www.w3.org/ns/wsdl/style/iri"
       wsdlx:safe="true">
      <wsdl:documentation>
         This operation returns a list of books.
      </wsdl:documentation>
      <wsdl:input element="msg:getBookList"/>
      <wsdl:output element="msg:bookList"/>
    </wsdl:operation>
  </wsdl:interface>

  <wsdl:binding name="BookListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http"
     interface="tns:BookListInterface">
    <wsdl:documentation>
       The RESTful HTTP binding for the book list service.
    </wsdl:documentation>
    <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
  </wsdl:binding>

  <wsdl:service name="BookList" interface="tns:BookListInterface">
    <wsdl:documentation>
       The bookstore's book list service.
    </wsdl:documentation>
    <wsdl:endpoint name="BookListHTTPEndpoint"
       binding="tns:BookListHTTPBinding"
       address="http://www.bookstore.com/books/">
    </wsdl:endpoint>
  </wsdl:service>
</wsdl:description>
```

Summary

In this article you learned about REST and how WSDL 2.0. REST Web services use HTTP and XML for communication. RESTful applications are resource-centric as opposed to action-centric.

The value in describing REST Web services in a formal way is using the description as a formal contract between clients and service providers, and support for tools. WSDL 2.0 supports the description of REST Web services. You walked through the description of the book list REST Web service using WSDL 2.0 and XML schema.

Hopefully you'll use the steps shown here for authoring the book list service's WSDL 2.0 description to author descriptions for your own REST Web services.

Acknowledgements

Thanks to Christopher Ferris, John Kaputin, and Anne James for reviewing this article and providing useful and detailed feedback.

Downloads

Description	Name	Size
bookstore Sample WSDL 2.0 descriptions	bookstore-sample.zip	5KB

Resources

Learn

- Read "How I Explained REST to My Wife" by Ryan Tomayko for a good introduction to REST.
- Read "RESTful SOA using XML" (developerWorks, Feb 2008) for a good overview of REST in practice and implementing REST Web services.
- Check out Roy Fielding's dissertation "Architectural Styles and the Design of Network-based Software Architectures," the definitive source about REST.
- "WSDL 2.0: Primer" provides a good introduction to the features of WSDL 2.0.
- The "WSDL 2.0 Part 1: Core Language" recommendation provides the specifics of WSDL 2.0 and the use of XML schema with WSDL 2.0.
- The "WSDL 2.0 Part 2: Adjuncts" recommendation provides the specifics of HTTP, SOAP 1.2, and MEP usage with WSDL 2.0.
- Read Lawrence Mandel's book *Eclipse Web Tools Platform: Developing Java Web Applications*.
- The SOA and Web services zone on IBM® developerWorks hosts hundreds of informative articles and introductory, intermediate, and advanced tutorials on how to develop Web services applications.
- Play in the IBM SOA Sandbox! Increase your SOA skills through practical, hands-on experience with the IBM SOA entry points.
- The IBM SOA Web site offers an overview of SOA and how IBM can help you get there.
- Stay current with developerWorks technical events and webcasts.
- Browse for books on these and other technical topics at the Safari bookstore.
- Check out a quick Web services on demand demo.

Get products and technologies

- Download Apache Woden for a Java-based WSDL 2.0 validating parser.
- Download Apache Axis2 for a Java-based Web services run time capable of generating and running Web services based on WSDL 2.0.
- Innovate your next development project with IBM trial software, available for download or on DVD.

Discuss

- Participate in the discussion forum for this content.
- Discuss REST and Web services on the Best Practices for SOA and Web Services and Web Site Development with Open Source Software forums.
- Get involved in the developerWorks community by participating in developerWorks blogs.

About the author

Lawrence Mandel

Lawrence Mandel, a software developer at the IBM Toronto Lab, is currently working on a reporting solution for IBM Rational. He leads the Apache Woden project, which is developing a WSDL 2.0 validating parser and related tools. Lawrence is coauthor of the book Eclipse Web Tools Platform: Developing Java Web Applications.