RESTful Web Services With Java™ API for XML Web Services (JAX-WS)

Marc Hadley

Senior Staff Engineer
Sun Microsystems
http://www.sun.com/

TS-1222

Goal of This Talk

What You Will Learn

Learn how to use the Java™ API for XML Web Services (JAX-WS) APIs to consume and create RESTful Web Services

Agenda

Introduction
The JAX-WS HTTP Binding
The Dispatch<T> API
The Provider<T> API
Advanced Topics
Conclusion
Additional Resources

Agenda

Introduction

The JAX-WS HTTP Binding

The Dispatch<T> API

The Provider<T> API

Advanced Topics

Conclusion

Additional Resources

REST?

- REpresentational State Transfer (REST)
  - Term coined by Roy Fielding in his PhD thesis
  - Architectural style of the Web
  - Resources are first class objects
  - Resources are addressable (URLs)
  - Interact with representations of resources
  - State is maintained within a resource representation
  - Small set of methods that can be applied to any resource (HTTP methods)
- Scalable, low cost of coordination

REST vs. RPC

- Different architectural styles
  - Underlying technologies can support either style
- RPC
  - Few endpoints, many methods (few nouns,many verbs)
- REST
  - Many resources, few methods (many nouns,few verbs)

RESTful Web Services?

- Based on existing Web architecture
  - Use HTTP as application layer protocol rather than transport
  - Able to leverage Web infrastructure like caches
- CRUD semantics

	SQL	HTTP
Create	INSERT	PUT
Read	SELECT	GET
Update	UPDATE	POST
Delete	DELETE	DELETE

RESTful WS vs. “SOAP” WS

- SOAP WS toolkits encourage an RPC or message-oriented architecture
  - Use HTTP as a transport (tunneling)
  - Single endpoint, many custom methods
  - Exposed resources not “on the Web”
  - Lose advantages of Web infrastructure
- RESTful WS—Put the Web back in Web Services
  - Based on Web architecture
  - Resources are “on the Web”
  - Few methods (CRUD) but many resources
  - Take advantage of Web infrastructure

JAX-WS?

- Java API for XML Web Services
- Follow on to Java API for XML-based Remote Procedure Calls (JAX-RPC), less RPC-centric, protocol agnostic
- Uses Java Architecture for XML Binding (JAXB) for XML to Java technology data binding
- Final, included in Java SE 6 and Java EE 5
- Programming model supports SOAP+WSDL style Web Services and RESTful Web Services
- No standard for RESTful service description so code generation not supported

Agenda

Introduction

The JAX-WS HTTP Binding

The Dispatch<T> API

The Provider<T> API

Advanced Topics

Conclusion

Additional Resources

Choosing the HTTP Binding—Client

```
URI nsURI = new URI(namespace);
QName serviceName = new QName(name,nsURI.toString());
QName portName = new QName(port,nsURI.toString());
Service s = Service.create(serviceName);
URI address = new URI(...);
s.addPort(portName, HTTPBinding.HTTP_BINDING,
    address.toString());
```

Configuring the HTTP Binding

- Set of standard properties
  - Endpoint address
  - HTTP method
  - URI path and query parameters
  - HTTP authentication
  - HTTP session (cookies, URL rewriting)
  - HTTP headers
  - HTTP response code
- Set properties in a runtime message context, each exchange has a separate context

Configuring the HTTP Binding—Client

```
Service s = ...;
Dispatch<Source> d = s.createDispatch(portName,
    Source.class, Service.Mode.PAYLOAD);
Map<String, Object> requestContext =
    d.getRequestContext();
requestContext.put(MessageContext.HTTP_REQUEST_METHOD,
    new String("GET"));
```

Agenda

Introduction

The JAX-WS HTTP Binding

The Dispatch<T> API

The Provider<T> API

Advanced Topics

Conclusion

Additional Resources

Dispatch<T> Interface

- Dynamic, low level API
- Methods
  - T invoke(T msg)
  - Response<T> invokeAsync(T msg)
  - Future<?> invokeAsync(T msg, AsyncHandler<T> h)
  - void invokeOneWay(T msg)
- Supported types for T
  - javax.xml.transform.Source
  - javax.activation.DataSource
  - javax.xml.soap.SOAPMessage
  - Object—when using JAXB

Dispatch<T> With Java API for XML Processing (JAXP) (Yahoo API)

```
Service s = ...;
URI address = new URI("http", null,
    "api.search.yahoo.com", 80,
    "/NewsSearchService/V1/newsSearch",
    "appid=jaxws_restful_sample&query=java",
    null);
s.addPort(portName, HTTPBinding.HTTP_BINDING,
    address.toString());Dispatch<Source> d =
Dispatch<Source> d = s.createDispatch(portName,
    Source.class, Service.Mode.PAYLOAD);
Map<String, Object> requestContext =
    d.getRequestContext();
requestContext.put(MessageContext.HTTP_REQUEST_METHOD,
    new String("GET"));
Source result = d.invoke(null);
```

Dispatch<T> With JAXP (Cont.)

```
Source result = d.invoke(null);
DOMResult domResult = new DOMResult();
Transformer trans =
TransformerFactory.newInstance().newTransformer();
trans.transform(result, domResult);

XPathFactory xpf = XPathFactory.newInstance();
XPath xp = xpf.newXPath();
xp.setNamespaceContext(new NSResolver("yn", nsURI.toString()));
NodeList resultList = (NodeList)xp.evaluate(
    "/yn:ResultSet/yn:Result",
    domResult.getNode(), XPathConstants.NODESET);
int len = resultList.getLength();
for (int i=0;i<resultList.getLength;i++) {
    String title = xp.evaluate("yn:Title", resultList.item(i));
    String click = xp.evaluate("yn:ClickUrl",
        resultList.item(i));
    System.out.printf("[%d] %s (%s)\n",i,title,click);
}
```

Dispatch<T> With JAXB (Yahoo API)

```
Service s = ...;
URI address = new URI(...);
s.addPort(portName, HTTPBinding.HTTP_BINDING,
    address.toString());
JAXBContext jbc = JAXBContext.newInstance(
    "com.yahoo.search");
Dispatch<Object> d = s.createDispatch(portName,
    jbc, Service.Mode.PAYLOAD);
Map<String, Object> requestContext =
    d.getRequestContext();
requestContext.put(MessageContext.HTTP_REQUEST_METHOD,
    new String("GET"));
ResultSet rs = (ResultSet)d.invoke(null);
```

Dispatch<T> With JAXB (Cont.)

```
ResultSet rs = (ResultSet)d.invoke(null);
for (ResultType r: rs.getResult()) {
    System.out.printf("%s: (%s)\n",r.getTitle(),r.getClickUrl());
}
```

DEMO

Dispatch<Source> With JAXP and JAXB (Yahoo API)

Agenda

Introduction

The JAX-WS HTTP Binding

The Dispatch<T> API

The Provider<T> API

Advanced Topics

Conclusion

Additional Resources

Provider<T> Interface

- Dynamic, low level service API
- 1 Method
  - T invoke(T request)
- Supported types for T
  - javax.xml.transform.Source
  - javax.activation.DataSource
  - javax.xml.soap.SOAPMessage
- Service implementing Provider<T> can be deployed in a Java EE container or published via JAX-WS Endpoint API

Example Provider

```
@WebServiceProvider
@ServiceMode(value=Service.Mode.PAYLOAD)
public class MyProvider implements Provider<Source> {
    public Source invoke(Source source) {
        String replyElement = new String("<p>hello world</p>");
        StreamSource reply = new StreamSource(
                new StringReader(replyElement));
        return reply;
    }

    public static void main(String args[]) {
        Endpoint e = Endpoint.create( HTTPBinding.HTTP_BINDING,
                new MyProvider());
        e.publish("http://127.0.0.1:8084/hello/world");
        Thread.sleep(10000); // leave running for 10 seconds
        e.stop();
    }
}
```

Provider<T> Types

- javax.xml.transform.Source
  - PAYLOAD or MESSAGE mode
  - Error if message not XML
- javax.activation.DataSource
  - MESSAGE mode
  - Useful for handling binary data or MIME packages, Java Mail API also useful here
- javax.xml.soap.SOAPMessage
  - MESSAGE mode
  - Good for dynamic SOAP-based services
- JAXB Object not directly supported

Provider With JAXB

```
public Source invoke(Source s) {
    JAXBContext jc = JAXBContext.newInstance(...);
    Unmarshaller u = jc.createUnmarshaller();
    RequestType request =
        (RequestType)u.unmarshall(s);
    ReplyType reply = processRequest(request);
    return new JAXBSource(jc,reply);
}
```

DEMO

Provider<Source> Using Endpoint API

Agenda

Introduction

The JAX-WS HTTP Binding

The Dispatch<T> API

The Provider<T> API

Advanced Topics

Conclusion

Additional Resources

Handlers in the HTTP Binding

- Only javax.xml.ws.handler.LogicalHandler supported
- Access to entire message if XML media type or root part of multipart/related if that is an XML media type
- Access to HTTP headers, method, status code, path and query parameters via MessageContext
- Handler exceptions are converted to HTTP 500 status code
  - Can throw HTTPException to control HTTP status code

Working With Non-XML Payloads

- Non-XML payloads are supported using the javax.activation.DataSource interface
- Dispatch<DataSource> used for HTTP client
- Provider<DataSource> used for HTTP server
- DataSource can be used for any media type including binary data like JPEG
- E.g., to emulate POSTing a HTML form:
  - Use media type application/x-www-form-urlencoded
  - Format content as “param1=value1&param2=value2”
  - Use HTTP POST method

DataSource Interface

- InputStream getInputStream()
  - Get a stream to read the data from
- String getName()
  - Get the name of the object—context dependent
- String getContentType()
  - The media type of the data, application/octet-stream if not known

DEMO

Working With Binary Data

Agenda

Introduction

The JAX-WS HTTP Binding

The Dispatch<T> API

The Provider<T> API

Advanced Topics

Conclusion

Additional Resources

Conclusion

- JAX-WS can be used to produce and consume RESTful Web services
- Dispatch<T> is client-side interface
- Provider<T> is server-side interface
- Developer can choose abstraction:
  - XML + XPath using JAXP
  - Java based objects using JAXB
  - SOAP messages using SAAJ
  - Arbitrary data using DataSource
- Not limited to XML payloads

Agenda

Introduction
The JAX-WS HTTP Binding
The Dispatch<T> API
The Provider<T> API
Advanced Topics
Conclusion
Additional Resources

For More Information

See

- http://www.jcp.org/en/jsr/detail?id=224 (JAX-WS specification)
- https://jax-ws.dev.java.net/ri-download.html (JAX-WS reference implementation)
- http://en.wikipedia.org/wiki/Representational_State_Transfer (REST information)
- http://today.java.net/pub/au/59 (Speakers weblog)
- http://java.sun.com/javaee/glassfish/ (Free open-source Java EE Application Server incl JAX-WS)

Q&A

RESTful Web Services With JAX-WS

Marc Hadley

Senior Staff Engineer
Sun Microsystems
http://www.sun.com/

TS-1222