Recommendations for Web Service Development

Nazrul Islam, Sameer Tyagi, Satish Viswanatham

Sun Microsystems, Inc.
glassfish.dev.java.net

Session TS-9263

Goal of This Talk

Discuss how to build and manage Web Services easily in Java EE 5 and some design options

Agenda

Ease of Development

Web Services BluePrints/Patterns

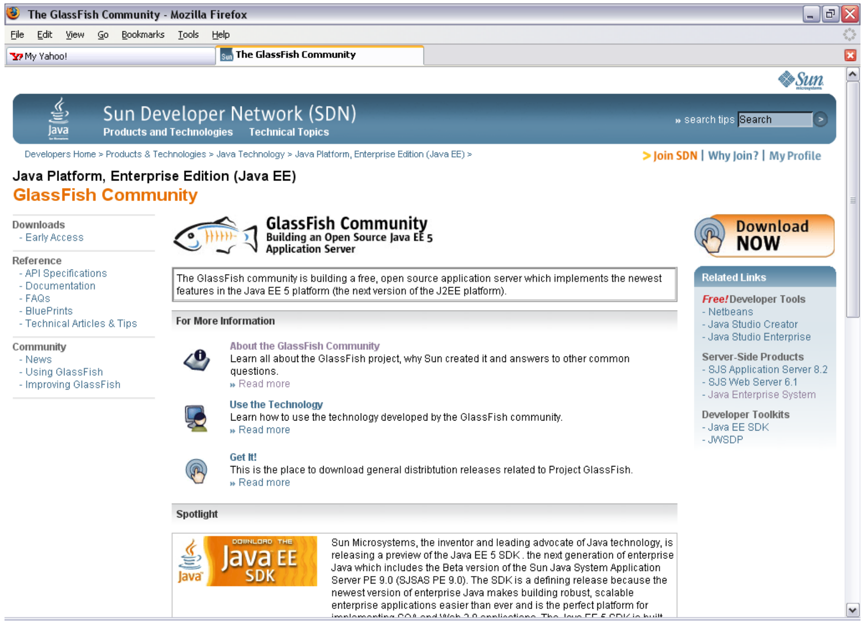
Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

Project GlassFish^{SM} Community


The GlassFish Community - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
My Yahoo!
The GlassFish Community
Sun Developer Network (SDN)
Products and Technologies
Technical Topics
» search tips
Search
Developers Home > Products & Technologies > Java Technology > Java Platform, Enterprise Edition (Java EE) >
> Join SDN | Why Join? | My Profile
Download NOW

Java Platform, Enterprise Edition (Java EE)

GlassFish Community

Downloads
- Early Access

Reference
- API Specifications
- Documentation
- FAQs
- BluePrints
- Technical Articles & Tips

Community
- News
- Using GlassFish
- Improving GlassFish

GlassFish Community
Building an Open Source Java EE 5 Application Server

The GlassFish community is building a free, open source application server which implements the newest features in the Java EE 5 platform (the next version of the J2EE platform).

For More Information

About the GlassFish Community
Learn all about the GlassFish project, why Sun created it and answers to other common questions.
» Read more

Use the Technology
Learn how to use the technology developed by the GlassFish community.
» Read more

Get It!
This is the place to download general distribtution releases related to Project GlassFish.
» Read more

Spotlight

DOWNLOAD THE Java EE SDK

Sun Microsystems, the inventor and leading advocate of Java technology, is releasing a preview of the Java EE 5 SDK . the next generation of enterprise Java which includes the Beta version of the Sun Java System Application Server PE 9.0 (SJSAS PE 9.0). The SDK is a defining release because the newest version of enterprise Java makes building robust, scalable enterprise applications easier than ever and is the perfect platform for implementing SOA and Web 2.0 applications. The Java EE 5 SDK is built

Related Links

Free! Developer Tools
- Netbeans
- Java Studio Creator
- Java Studio Enterprise

Server-Side Products
- SJS Application Server 8.2
- SJS Web Server 6.1
- Java Enterprise System

Developer Toolkits
- Java EE SDK
- JWSDP

Building a Java™ EE 5 Platform-based Open Source Application Server

java.sun.com/javaee/GlassFish

- Simplifying Java application development with Java EE 5 technologies
- Includes JWSDP, EJB 3.0, JSF 1.2, JAX-WS and JAX-B 2.0
- Supports > 20 frameworks and apps
- Open source CDDL license
- Basis for the Sun Java System Application Server PE 9
- Free to download and free to deploy
- Over 2,200 members and 200,000 downloads

Agenda

Ease of Development

Web Services BluePrints/Patterns

Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

What Changed for the Web Services?

- Significantly revised and simplified
- JAX-RPC 2.0 renamed to JAX-WS 2.0
  - Breaks compatibility with JAX-RPC 1.1
- Key features
  - Simplified programming model with annotations and dependency injection
  - Uses JAXB 2.0 for type-mappings
  - Portable runtime artifacts
  - Can generate annotated JAX-WS and JAXB code from WSDL and XSD

JAXB 2.0 Is Now Bi-Directional

- 1.0: Schema → Java only
  - JAXB is for compiling schema
  - Don’t touch the generated code
- 2.0: Java → XML + schema compiler
  - JAXB is about persisting POJOs to XML
  - Annotations for controlling XML representation
  - Modify the generated code to suit your taste

J2EE™ 1.4 Platform-Based Web Service

Code Written by Developer/Deployer

```
package endpoint;
import java.rmi.*;
public class HelloServiceImpl
            implements HelloServiceSEI
{

    public String sayHello(String
param)
        throws
java.rmi.RemoteException {
        return "Hello " + param;
    }
}
package endpoint;
import java.rmi.*;
public interface HelloServiceSEI
            extends java.rmi.Remote {
    public String sayHello(String
param)
        throws
java.rmi.RemoteException;
}
```

```
<?xml version='1.0' encoding='UTF-8' ?>
<webservices xmlns='http://java.sun.com/xml/ns/j2ee'
version='1.1'>
  <webservice-description>
    <webservice-description-name>
     HelloService</webservice-description-name>
    <wsdl-file>
     WEB-INF/wsdl/HelloService.wsdl</wsdl-file>
    <jaxrpc-mapping-file>
     WEB-INF/HelloService-mapping.xml
     </jaxrpc-mapping-file>
    <port-component xmlns:wsdl-port_ns='urn:HelloService/wsdl'>
      <port-component-name>HelloService</port-component-name>
      <wsdl-port>wsdl-port_ns:HelloServiceSEIPort</wsdl-port>
      <service-endpoint-interface>
       endpoint.HelloServiceSEI</service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>WSServlet_HelloService</servlet-link>
      </service-impl-bean>
    </port-component>
  </webservice-description>
</webservices>
<?xml version='1.0' encoding='UTF-8' ?>
<configuration
    xmlns='http://java.sun.com/xml/ns/jax-rpc/ri/config'>
  <service name='HelloService'
      targetNamespace='urn:HelloService/wsdl'
      typeNamespace='urn:HelloService/types'
      packageName='endpoint'>
    <interface name='endpoint.HelloServiceSEI'
        servantName='endpoint.HelloServiceImpl'>
    </interface>
  </service>
</configuration>
}
```

Ease of Development

Web Service in Java EE 5 Platform

```
package server;
import javax.jws.WebService;
@WebService
public class HelloImpl {

    public String sayHello(String name) {
        return "Hello, " + name + "!";
    }
}
```

Ease of Development

Development and Deployment of POJO

- Compile POJO to auto-deploy directory
  - javac -classpath $AS_HOME/lib/javaee.jar -d $AS_HOME/domains/domain1/autodeploy HelloImpl.java
- Annotations are processed and appropriate deployment descriptors are generated automatically

Ease of Development

Web Services as First-Class Objects

- Auto-discovery of Web Services
- View Web Service meta data
    - URI, endpoint type, descriptors, and associated information
- Auto generated test forms—ping
    - Shows operations and parameter values
    - Supports Java APIs for XML Web Services (JAX-WS) standard
- Web Services as Java Business Integration (JBI) service providers by default

Ease of Development

Operational Statistics and Content Visualization

- Number of requests, throughput, response time (average, min, max), number of SOAP faults
- Message trace—SOAP messages for a Web-service endpoint are displayed
  - SOAP request, response, HTTP headers, response time, Size of request, response, SOAP fault, Client IP address and user principle
- You can also configure the number of messages that are kept in memory (25 by default)
- LOW (statistics), HIGH (statistics + content visualization), OFF (none)

Ease of Development

Call Flow/Root Cause Analysis

- Track processing time of a request in each of the major container (Web, EJB™ architecture, JDBC™ software, and ORB)
- The flow data often reveal performance bottlenecks
- Measure performance in live environment
  - Using filter (IP, user principle) you can collect information on particular request types

DEMO

Web Services Monitoring

Agenda

Ease of Development

Web Services BluePrints/Patterns

Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

Web Services BluePrints/Patterns

- Prefer Java language type parameters that have standard type mapping
  - For example, use Java technology arrays instead of ArrayList and Collection
- Handling non-standard type parameters
  - Extensible type mapping not standard
  - Avoid as much as possible
- Two types of Web service requests
  - Short processing time → synchronous response
  - Long processing time → asynchronous response

Web Services BluePrints/Patterns

Choice of the Interface Endpoint Type

- JAX-WS service endpoint in the Web tier
    - A JAX-WS service endpoint has to handle concurrent client access on its own
    - Transactional context is unspecified
    - There is also no declarative means to automatically start the transaction
- JAX-WS service endpoint in the Enterprise JavaBeans™ (EJB) architecture tier
    - An EJB architecture-based service endpoint is implemented as a stateless session bean, multi-threaded access is handled by the EJB architecture-based container
    - Runs in the transaction context of the EJB architecture-based container
    - Can declaratively demarcate transactions

Agenda

Ease of Development

Web Services BluePrints/Patterns

Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

Strategies for Document-Based WS

Using XML in the SOAP Body

- Interoperability
- Validate against schema if XML docs are used
- Better performance than encoded formatting styles
- Service interface clearly describes the types of documents expected; This makes the WSDL file easier for clients to understand
- Can not use custom bindings or binding frameworks directly
- Endpoint receives object representation; if you want the XML, you have to reconstruct it

Strategies for Document-Based WS

Using String in the SOAP Body

- Simple, same as writing a "hello world" application
- Simple to develop clients
- No issues with interoperability
- Schema validation offered by the runtime cannot be used
- Service interface is not descriptive because the document type is just a general string
- Memory intensive: The entire XML document is read into memory as a string for each request

Strategies for Document-Based WS

Switching Off Data Binding

- Integration with third-party frameworks
- The XML document is received in its entirety
- Building RESTful services
- The behavior may be implementation specific
- Loss of business context: The payload context is not described in the WSDL

Strategies for Document-Based WS

Using the xsd:any Element in WSDL

- The mapping of the xsd:any element has been standardized to map to a SOAPElement
- Element is named in the WSDL
- Can be used with binding frameworks
- Schemas can evolve independently
- Need to manipulate low level SOAPElement objects
- Schemas defining the documents are not referenced directly
- Schemas need to be negotiated out of band

Strategies for Document-Based WS

Using Base 64-Encoded or Raw Bytes in the SOAP Body

- This may be useful when the XML contains characters or declarations that are not supported either by the SOAP message infoset or by the runtime implementation; examples of these are Document Type Definition (DTD) declarations, binary data, locale-specific character encoding, and so on
- Interoperability: Both parties need to know out of band what the data is
- Increased size: Base64 increases the size by 33%

Strategies for Document-Based WS

Using Message Attachments in the SOAP Message

- Useful for documents that might conform to schemas such as a DTD
- Useful for large documents (can be compressed and decompressed)
- Additional facilities can be built on the attachments using handlers
- Interoperability: Not all vendors support attachments

Agenda

Ease of Development

Web Services BluePrints/Patterns

Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

RESTful Web Services

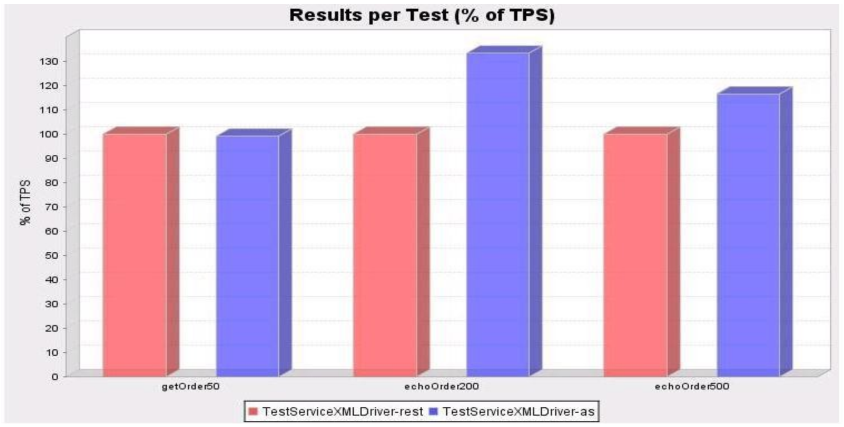
Representational State Transfer

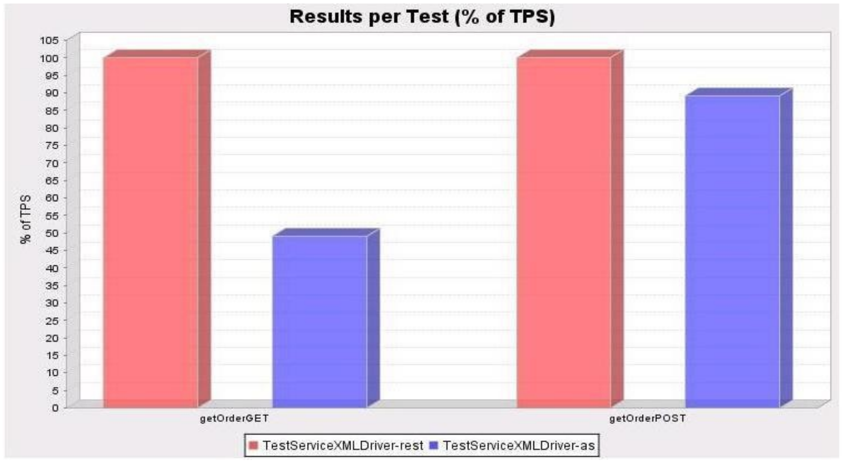
- REST is an arch style, not a standard
- Stateless—each request has all necessary information
- Employs standard HTTP methods such as GET, POST, PUT, DELETE
- Use logical URLs for all resources
- Design to reveal data gradually

RESTful Web Services

Representational State Transfer

- Simple to build with JAX-WS
- In the context of your application consider
  - Performance implications
  - Contract implications





Agenda

Ease of Development

Web Services BluePrints/Patterns

Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

Web Services Annotations

@WebService

- Marks a java class as web service implementation
  - Name—name of the WSDL <portType>
  - ServiceName—name of the WSDL <service>
  - WsdlLocation—location of pre-defined WSDL
  - TargetNamespace—XML namespace for WSDL and schema elements

Web Services Annotations

@SOAPBinding

- Default is DOCUMENT/LITERAL binding
  - Style—DOCUMENT or RPC
  - Use—LITERAL or ENCODED
  - ParameterStyle—WRAPPED or BARE

Web Services Annotations

@WebParam

- Depends on SOAPBinding
  - Name—name of wsdl:part in case of RPC
  - PartName—local name of element in case of DOCUMENT/BARE style
  - TargetNameSpace—namespace for this element, used only for DOCUMENT style
    - Defaults to namespace of web service
  - Mode—IN, INOUT, OUT holder types/RPC
  - Header—true or false
- Same rules apply for @WebResult

Web Services Annotations
@WebMethod

- Marks the method as web service operation
  - OperationName—name of the wsdl:operation
  - Action—SOAPAction header in case of SOAP
  - Exclude—by default all public methods are exposed

Web Services Annotations

@WebServiceRef

- No defaults
    - Name—jndi name of the resource
    - Type—java type of the resource
    - MappedName—product specific resource name
    - Value—service class name
    - WsdlLocation—location of wsdl
- wsdlLocation commonly used

Web Services Annotations

@WebServiceProvider

- Provider implementation class
  - portName—name of the WSDL wsdl:portName
  - ServiceName—name of the WSDL wsdl:service
  - WsdlLocation—location of pre-defined WSDL
- Also look at @ServiceMode—PAYLOAD(Source) or MESSAGE (SOAPMessage)

Web Services Annotations

Others

- @Stateless can used with @WebService
- @WebServiceClient—represents generated service interface, not client
- @WebEndpoint—ports with in the service
- @BindingType—default is SOAP 1.1/HTTP
- All annotations are in javax.jws.* or javax.jws.soap.*

Web Service Annotations

Example

```
@WebService( name="HelloWebService",
          targetNamespace="http://javaone.org/HelloWebService")
@SOAPBinding(style=SOAPBinding.Style.RPC,
use=SOAPBinding.Use.LITERAL)

public class HelloImpl {
   @WebMethod(action="urn:sayHello")
   @WebResult(name="greeting")
   public String sayHello(
         @WebParam(name="user")
         String name) {
      return "Hello "+name+"!";
   }
}

@WebServiceRef(wsdlLocation=”
http://localhost:8080/HelloImpl/HelloImplService?WSDL”)
static server.HelloImpl service;
```

Web Services Annotations

Commonly Used Annotations

- @WebService
- @WebMethod
- @OneWay
- @WebResult
- @WebParam
- @HandlerChain
- @SOAPBinding
- @WebServiceRef
- @BindingType
- @RequestWrapper
- @ResponseWrapper
- @ServiceMode
- @WebEndpoint
- @WebFault
- @WebServiceClient
- @WebServiceProvider

Agenda

Ease of Development

Web Services BluePrints/Patterns

Strategies for Document-Based Web Services

RESTful Web Services

Web Services Annotations

Web Services in Enterprise

Programmatic Support for Web Services Management

Java Management Extensions (JMX™) API in Project GlassFish

```
try {
    // acs is object of type AppserverConnectionSource.
    final DomainRoot domainRoot = acs.getDomainRoot();

    Map m = dr.getWebServiceMgr().getWebServiceEndpointKeys();

    System.out.println("Number of web services " + m.keySet().size());
    System.out.println("Fully qualified names...");
    for (Iterator iter = m.keySet().iterator(); iter.hasNext();) {
                        System.out.println((String)iter.next() + "\n"));
    }
 } catch(...) {
 }
```

Web Services in Enterprise

Clustering Web Services

- Bind the URL of the Target Service programmatically at run time
  - This is often determined at or before deployment, so it is not necessary to do at runtime
- Bind the URL of the Target Service at deployment time
  - This is done in the runtime deployment descriptor of the client calling the service
- Use @WebServiceRef + service url from descriptor

Web Services in Enterprise

Externalizing Web Service Endpoint Addresses

- Service/dispatch interface—get URI from descriptor
- Bind the URL of the Target Service at build time (WSDL)
  - Usually the URL is of the load balancer

```
<service name="StringPurchaseOrderService">
    <port name="PurchaseOrderServiceSEIPort"
     binding="tns:PurchaseOrderServiceSEIBinding">
      <soap:address location="http://lbhost
     :8080/webservice/StringPurchaseOrderServiceBean"/>
   </port>
</service>
```

Web Services in Enterprise

Service Virtualization

- For a fine-grained control of Web-service request and responses, XSLT rules can applied to each of the Web-service endpoints
- Applying XSLT can mean a longer processing time for the Web services
- Transformation rules can act as proxies for the Web service with different dialects

Web Services in Enterprise

Performance

- Fast Infoset encoding (binary format) improves performance by two to four times for larger Web services
- Fast Infoset can be enabled with no application code change
  - Client side system property **`com.sun.xml.rpc.client.ContentNegotiation=pessimistic`**

Web Services in Enterprise

Self Management

- Management rules comprise an event and an optional action (Mbean)
  - Events—Lifecycle events, monitor events, log events, trace events, timer events, notification events
- Using the monitoring statistics, you can trigger alerts or configure application server to perform management tasks

Web Services in Enterprise

Security and Audit

- Web service security
  - username/password, X509, SAML token profile
- WS-I Basic Security Profile
- Integration with Access Manager
  - Single sign-on for Web services
- Audit module records requests and responses for non-repudiation
  - Audit module can be customized by implementing `com.sun.appserv.security.AuditModule`

Web Services in Enterprise

Governance

- Effectively advertise Web services through registries
- We support both ebXML and UDDI registries
- Easy to configure, publish, and un-publish features

DEMO

Summary

- It is easy to develop, deploy, and manage Web Services on Java EE 5
- Download the Java EE 5 SDK and try!
- Java BluePrints Solutions Catalog

  http://bpcatalog.dev.java.net

For More Information

- TS-1194 Java™ API for XML Web Services (JAX-WS) 2.0
- BOF-2593 Implementing High-Performance Web Services with Next-Generation Java™ Technology APIs
- Java API for XML Web Services
  http://java.sun.com/webservices/jaxws/
- Web Services Management in Project GlassFish
  https://glassfish.dev.java.net/javaee5/ws-mgmt/wsmgmthome.html
- AMX
  https://glassfish.dev.java.net/javaee5/amx/

Q&A

Recommendations for Web Service Development

Nazrul Islam, Sameer Tyagi, Satish Viswanatham

Sun Microsystems, Inc.
glassfish.dev.java.net

Session TS-9263