



# JAX-WS and related APIs: The Web Services Foundation

**Santiago Pericas-Geertsen**  
**Sun Microsystems**

[Santiago.PericasGeertsen@sun.com](mailto:Santiago.PericasGeertsen@sun.com)



**UNLOCK  
OPPORTUNITY**

What will you open?

**SUN TECH DAYS 2006-2007**  
A Worldwide Developer Conference

# About the speaker

- Ph.D. in Computer Science from Boston University
- Staff Engineer at Sun for 4.5 years
- Participated in the following projects:
  - > XSLTC, Fast Infoset / Fast Schema, JAX-RPC, JAX-WS, WS Performance, JAXP
  - > Currently tech lead for JAXP

# What is JAX-WS 2.0?

- Next generation JAX-RPC
- Core technology for building next generation distributed systems on the Web
- Cornerstone for SOA on the Java platform
- Two types of APIs:
  - > Strongly typed or Java based (higher level)
  - > Loosely typed or XML based (lower level)

# What's new in JAX-WS 2.0?

- Embrace POJO concepts via annotations
- Descriptor-free programming
- Layered architecture
- Protocol and transport independence
- Integrated data binding via Java Architecture for XML Binding (JAXB) 2.0
- Part of Java SE 6 and Java EE 5 platforms

# Interoperability

- Standards-compliant
  - > W3C/WS-I SOAP 1.1/1.2, WSDL 1.1, BP 1.0/1.1
- Foundation for full WS-\* stack
  - > WSIT (a.k.a., Project Tango)
  - > Interoperability with Windows Communication Foundation (a.k.a., Indigo)
- 100% W3C XML Schema support in JAXB 2.0

# Takeaways from this Talk

Learn How to Write a ...

	Java API	XML API
Web Service	✓	✓
Web Service Client	✓	✓

*RESTful*

# A Web Service using the Java API

# Building Strategies

- Starting from a **WSDL** file
  - > Generate classes using `wsimport`
  - > Implement WS interface
  - > Build and deploy
- Starting from a **POJO**
  - > Annotate POJO
  - > Build and deploy
    - > WSDL file generated automatically

# Web Service from a POJO

- Write a POJO implementing the service
- Add `@WebService` to it
- Optionally, inject a `WebServiceContext`
- Deploy the application
- Point your clients at the WSDL
  - > e.g. `http://myserver/myapp/MyService?WSDL`

# Calculator Service

`@WebService`

```
public class Calculator {  
    public int add(int a, int b) {  
        return a+b;  
    }  
}
```

- Public methods become web methods
- Or use `@WebMethod` annotation
- Reasonable defaults for names

# Web Service Client

- Generate annotated classes/interfaces from WSDL
  - > WSDL is generated when starting from POJOs
- Call new on Service class
- Get a proxy using a getPort() method
- Invoke any remote operations

# Calculator Service Client

```
CalculatorService svc =  
    new CalculatorService();  
Calculator proxy = svc.getCalculatorPort();  
int answer = proxy.add(35, 7);
```

- No need to use factories!
- Use of dynamic proxies to hide complexity
- XML is completely hidden from programmer

# Calculator Service Demo using Netbeans

# **A RESTful Web Service using the XML API**

# RESTful Web Services

- REpresentational State Transfer (REST)
  - > Term coined by Roy Fielding in his PhD thesis
- Architectural style of the Web
- Resources are addressable, first class objects
- 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)

# To REST or not to REST?

- To REST:
  - > Put the **Web** back in Web Services
  - > Based on Web architecture
  - > Resources are **on** the Web
  - > Few methods (CRUD) but many resources
- Not to REST:
  - > Typically: SOAP and HTTP POST
  - > Use HTTP as a transport (tunneling)
  - > Single endpoint, many custom methods
  - > Exposed resources **not on** the Web

# RESTful Client for Google Base

- Google Base?
  - > CRUD content to be indexed by Google
  - > Attach attributes to content to help search engine
  - > Web or **web services** interface
- Query Google Base for content using “sun+tech+days”
  - > Results returned as an Atom feed

# JAX-WS Dispatch<T> API

- Client-side XML API in JAX-WS
- Dispatch<T> for T:
  - > javax.xml.transform.Source (JAXP)
  - > javax.activation.DataSource (ACTIVATION)
  - > javax.xml.soap.SOAPMessage (SAAJ)
  - > Object using JAXB (†)
- Methods for one-way and asynch calls available

(†) More of a strongly typed than an XML API

# Google Base Client

```
URI address = new URI("http", null,  
    "www.google.com", 80,  
    "/base/feeds/snippets",  
    "bq=sun+tech+days", null);  
Dispatch<Source> d =  
    createDispatch(address);  
Source result = d.invoke(null);  
outputResult(result);
```

# Google Base Client: createDispatch()

```
Dispatch<Source> createDispatch(URI uri) {
    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"));
    setUpHTTPHeaders(d);
    return d;
}
```

# Google Base Client Demo

# RESTful Service

- The dual of Dispatch<T> is Provider<T>
- Implementation method:
  - > T invoke(T request)
  - > T ranges over same set of types
- Use **@WebServiceProvider** annotation

# Google Base Service

```
@WebServiceProvider
```

```
@ServiceMode(value=Service.Mode.PAYLOAD)
```

```
@BindingType(value=HTTPBinding.HTTP_BINDING)
```

```
public class GoogleBaseService  
    implements Provider<Source> {
```

```
    @Resource
```

```
    protected WebServiceContext context;
```

```
    public Source invoke(Source source) {
```

```
        ...
```

```
    }
```

```
}
```

# Google Base Service: invoke()

```
public Source invoke(Source source) {  
    Map<String, List<String>> query =  
        getQueryParams();  
    return createResponse(query);  
}
```

# Google Base Service: getQueryParams()

```
protected Map<String, List<String>>
  getQueryParams()
{
  MessageContext msgCtx =
    context.getMessageContext();
  String queryString =(String)
msgCtx.get(MessageContext.QUERY_STRING);
  return parseQueryString(queryString);
}
```

# Takeaways review

We learned how to write

	Java API	XML API
Web Service	✓	✓
Web Service Client	✓	✓

*RESTful*

# Conclusions

- Annotations reduce need for descriptors
- APIs at different levels of abstraction
- Clean integration with JAXB 2.0
- Support for traditional and RESTful WS
- Fully integrated into Netbeans
- Part of Java EE 5 and Java SE 6 platforms

# References

- Project Glassfish (AS):
  - > <https://glassfish.dev.java.net/>
- JAX-WS:
  - > <https://jax-ws.dev.java.net/>
- WADL (RESTful WS):
  - > <https://wadl.dev.java.net/>
- Blogs:
  - > <http://blogs.sun.com/theaquarium>

# Q & A



# JAX-WS and related APIs: The Web Services Foundation

**Santiago Pericas-Geertsen**  
**Sun Microsystems**

[Santiago.PericasGeertsen@sun.com](mailto:Santiago.PericasGeertsen@sun.com)



**UNLOCK  
OPPORTUNITY**

What will you open?

**SUN TECH DAYS 2006-2007**  
A Worldwide Developer Conference