Software Platforms

LM in Computer Engineering

Massimo Maresca

Let us recap what Web Service is about

https://docs.oracle.com/javaee/7/tutorial/webservices-intro.htm

Web services are client and server applications that communicate over the World Wide Web's (WWW) HyperText Transfer Protocol (HTTP). As described by the World Wide Web Consortium (W3C), web services provide a standard means of interoperating between software applications running on a variety of platforms and frameworks. Web services are characterized by their great interoperability and extensibility as well as their machine-processable descriptions, thanks to the use of XML. Web services can be combined in a loosely coupled way to achieve complex operations. Programs providing simple services can interact with each other to deliver sophisticated added-value services.

Big vs REST WS

"Big" Web Services

- JAX-WS provides the functionality for "big" web services. Big web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

RESTful Web Services

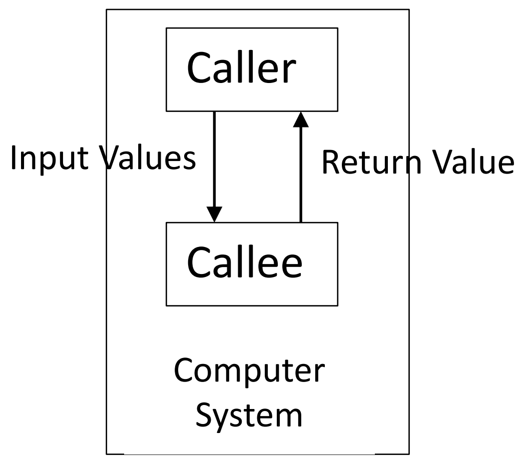
- JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service-API definitions.
- Project Jersey is the production-ready reference implementation for the JAX-RS specification. Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

FROM SOAP TO REST – SOAP RECAP

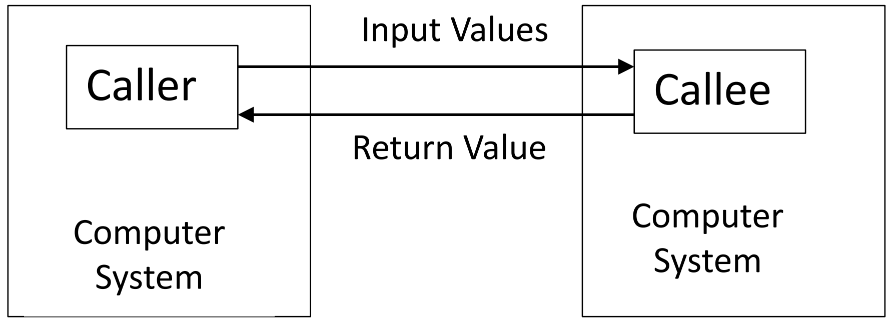
- Let us first recap what SOAP WS means
  - RPC/RMI
  - Language independency
  - Transparency
- What does RPC means ? Call – Response with service delocalization.
- Support of Client-Server “channels” to hide delocalization.
- SOAP as a Transport Protocol (Data/Control Plane)
- WSDL as an agnostic Interface Description Language (Management Plane)

SOAP-WS (BIG-WS) Recap

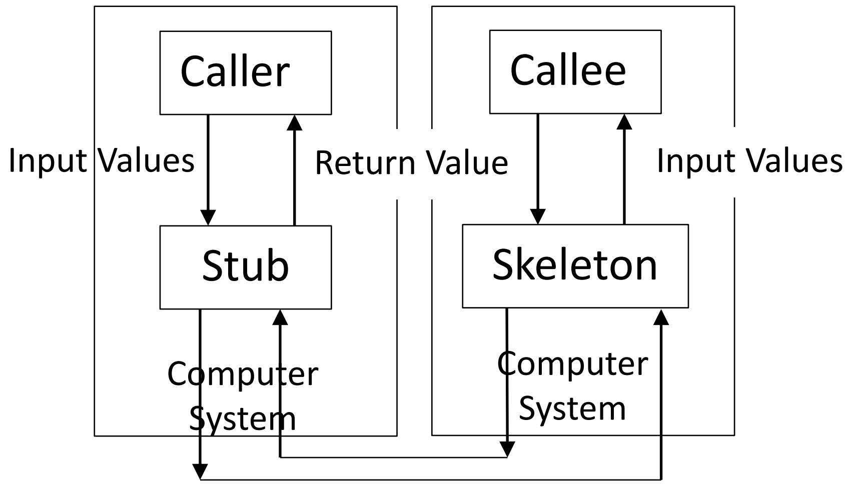
Local Procedure Call



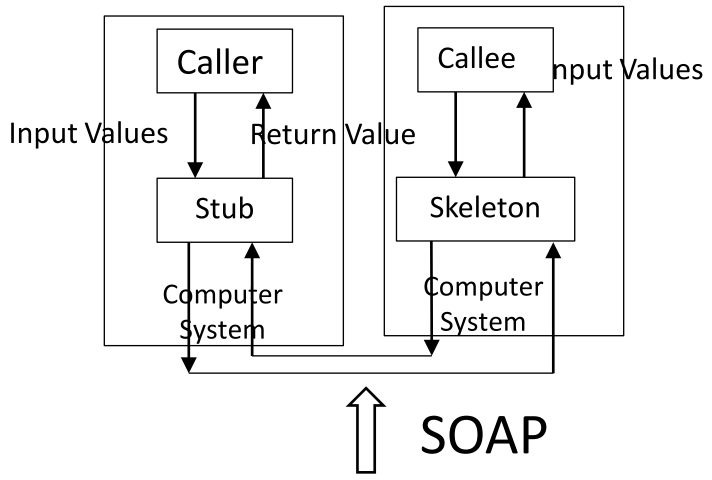
Remote Procedure Call/Method Invocation



RPC/RMI Architecture



Language Independency



Web Service Structure

- A link to Web Service Creation
  https://docs.oracle.com/javaee/7/tutorial/jaxws001.htm
- The starting point for developing a JAX-WS web service is a Java class annotated with the javax.jws.WebService annotation. The @WebService annotation defines the class as a web service endpoint.
- Examples of Web Server platforms:
  - Apache Tomcat + Axis 2
  - Jboss Wildfly
  - Sun Glassfish
  - Oracle WebLogic

Why Tomcat requires Axis 2

- Tomcat is just a servlet container. The implementation of the Http methods is left to the user.
- Considering that SOAP comes as a payload of a POST method a processing layer providing SOAP processing in a doPost() method is necessary.
- That is exactly the motivation behind Axis 2. When a service is requested Tomcat does not activate the service War, while it activates the Axis 2 War. In summary the Axis 2 doPost() method does what follows:
  - It extracts from the URI the name of the service,
  - It checks that the service exists in the repository, and if it does,
  - it processes the request (the Http POST message) and sends back the result.

FROM SOAP TO REST – RESTO INTRO

- Let us now move to REST:
  - **RESTful web services** are loosely coupled, lightweight web services that are particularly well suited for creating APIs for clients spread out across the internet. **Representational State Transfer (REST)** is an architectural style of client-server application centered around the **transfer** of **representations** of **resources** through requests and responses.

  https://docs.oracle.com/javaee/7/tutorial/jaxrs.htm

  https://download.oracle.com/otn-pub/jcp/jaxrs-2_1-final-eval-spec/jaxrs-2_1-final-spec.pdf
- REST is an architectural style based on web-standards and on the HTTP protocol. In a REST based architecture everything is a RESOURCE. A RESOURCE is accessed via a common interface based on the HTTP standard methods. In a REST based architecture a REST server provides access to the RESOURCES. A REST client can access and modify the REST RESOURCES.
- Every RESOURCE should support the HTTP common operations. Resources are identified by global IDs (which are typically URIs).
- REST allows that resources have different REPRESENTATION, e.g., text, XML, JSON etc. The REST client can ask for a specific representation via the HTTP protocol (content negotiation).

Principles

- Client Server (obvious)
- Stateless (Sessionless – supports scalability through load balancing)
- Accessible (URI)

Reference Specification vs Reference Implementation

- JSR 370: Java^{TM} API for RESTful Web Services (JAX-RS 2.1) Specification
  https://jcp.org/aboutJava/communityprocess/final/jsr370/index.html
- Jersey is the reference implementation. The idea is to give users the power to create Web Services in the form of POJO's (Plain Old Java Objects) taking advantage of run-time annotations.
  https://eclipse-ee4j.github.io/jersey/
- Jersey handles the HTTP requests, in terms of methods and media types, and dispathces the request to the appropriate methods.

From JSR 370

Goals

- POJO-based: The API will provide a set of annotations and associated classes/interfaces that may be used with POJOs in order to expose them as Web resources. The specification will define object lifecycle and scope.
- HTTP-centric: The specification will assume HTTPis the underlying network protocol and will provide a clear mapping between HTTP and URI elements and the corresponding API classes and annotations.
- Format independence: The API will be applicable to a wide variety of HTTP entity body content types. It will provide the necessary pluggability to allow additional types to be added by an application in a standard manner.
- Container: independence Artifacts using the API will be deployable in a variety of Web-tier containers. The specification will define how artifacts are deployed in a Servlet container.

From JSR 370

A JAX-RS application is packaged as a Web application in a .war file. The application classes are packaged in WEB-INF/classes or WEB-INF/lib and required libraries are packaged in WEB-INF/lib.

```
<servlet>
  <servlet-name>Service0</servlet-name>
  <servlet-class>
                  org.glassfish.jersey.servlet.ServletContainer
    </servlet-class>
  <init-param>
   <param-name>jersey.config.server.provider.packages</param-name>
   <param-value>it.unige.cipi</param-value>
  </init-param>
</servlet>
 <servlet-mapping>
  <servlet-name>Service0</servlet-name>
  <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
```

HTTP Methods

- The *PUT*, *GET*, *POST* and *DELETE* methods are typically used in REST based architectures. The following table gives an explanation of these operations.
  - GET defines a reading access of the resource without side-effects. The resource is never changed via a GET request.
  - PUT creates a new resource.
  - DELETE removes the resources.
  - POST updates an existing resource or creates a new resource.
- A RESTFul Web Service defines the base URI for the services, the supported MIME-types (XML, text, JSON, user-defined, …). It also defines the set of operations (POST, GET, PUT, DELETE) supported.

Examples

- Jersey Rest Service
  - Rest Service Command Line Project
  - Rest Service Eclipse Project
- Client based on Postman (https://www.getpostman.com/)
- Rest Client Jersey

JSON (example json <-> xml)

http://www.json.org/

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```

JSON Parsing with GSON

https://github.com/google/gson/blob/master/UserGuide.md#TOC-Overview

Main Functionalities:

- gson.fromJson(JsonString, Class); parses Json String into Class
- gson.toJson(Class); returns Json String

Tools ….

- Eclipse
- Tomcat
- Axis 2
- Jersey
- Ant
- Postman
- JSON
- GSON