Web Services & REST

By :

Sujata Batra

Sujata Batra

Service –Oriented Architecture

- A means of developing distributed systems where the components are stand-alone services.
- Services may execute on different computers from different service providers.
- Standard protocols have been developed to support service communication and information exchange.

Services or components?

- A service can be defined as:
  - A loosely-coupled, reusable software component that encapsulates discrete functionality which may be distributed and programmatically accessed. A web service is a service that is accessed using standard Internet and XML-based protocols.
- A critical distinction between a service and a component as defined in component-based software engineering is that services are independent.
- Services do not have a ‘requires’ interface.
- Services rely on message-based communication with messages expressed in XML.

Benefits of SOA

- Services can be provided locally or outsourced to external providers.
- Services are language-independent.
- Investment in legacy systems can be preserved.
- Inter-organizational computing is facilitated through simplified information exchange.

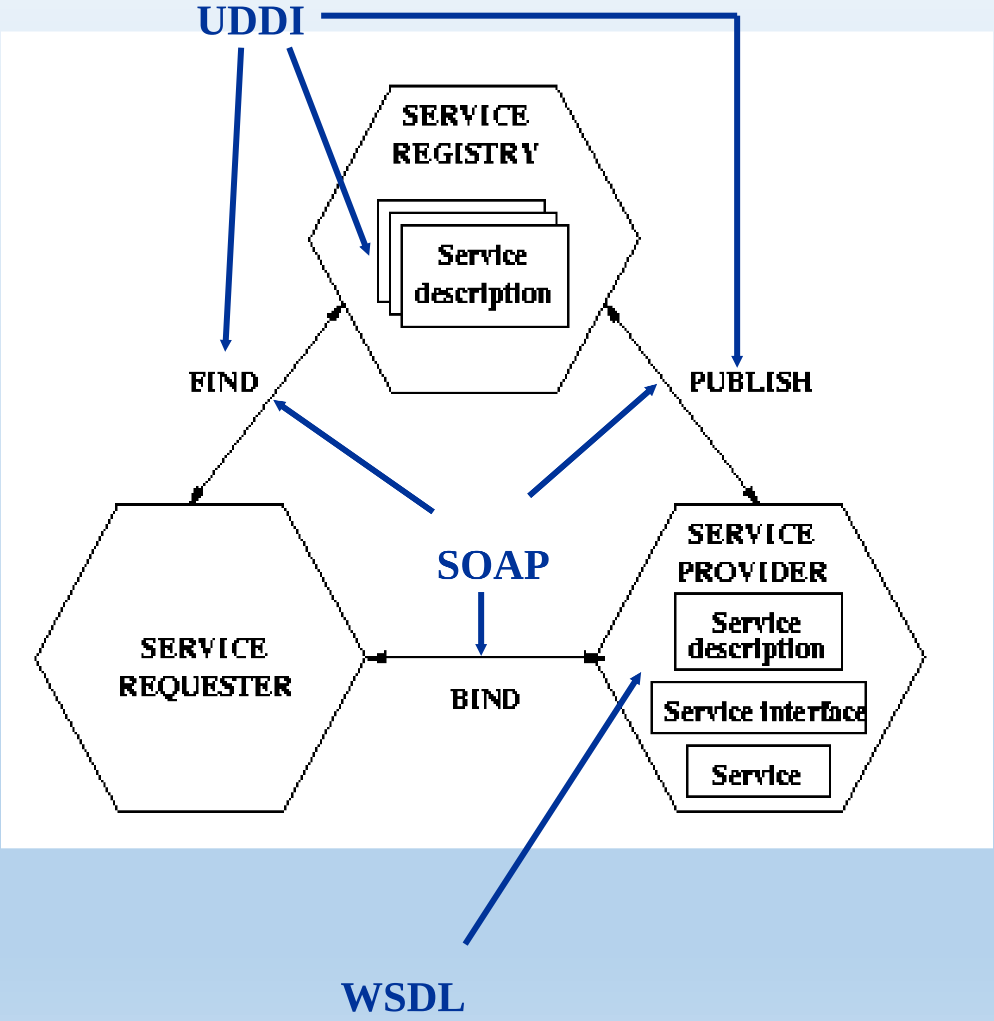
SOA and Web Services

- Service-oriented architecture is an architectural style
    - Derived from the client-server architectural style.
    - Clients (service consumers or requesters) and servers (service providers) connected by a service “bus”.
    - Services defined using formal interfaces (contracts).
    - Service bus supports point-to-point and messaging styles of communication.
    - Support for system qualities, e.g., security and transaction management.
- Web services
    - Services provided in a SOA deployed over the web.

Key standards

- SOAP (Simple Object Access Protocol)
  - A message exchange standard that supports service communication.
- WSDL (Web Service Definition Language)
  - This standard allows a service interface and its bindings to be defined.
- UDDI (Universal Description Discovery and Integration)
  - Defines the components of a service specification that may be used to discover the existence of a service.

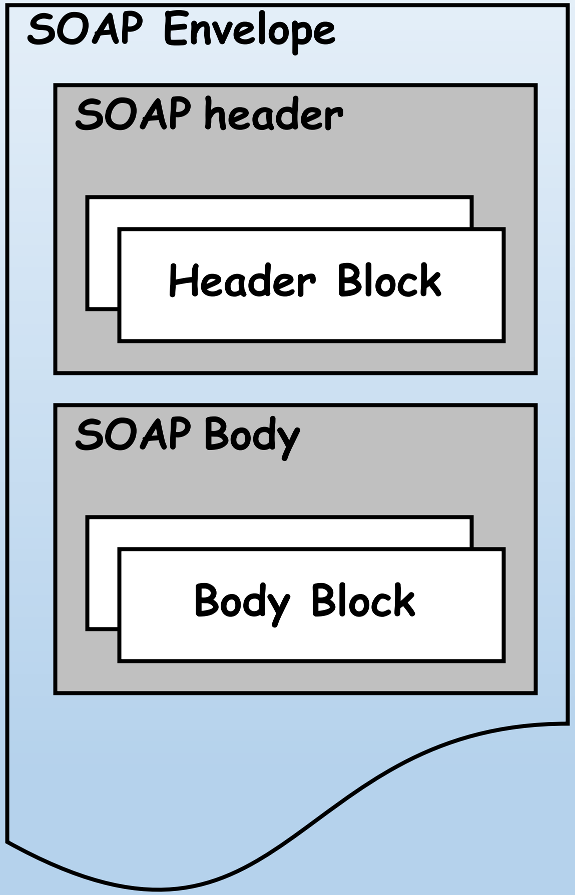
Service-oriented architecture

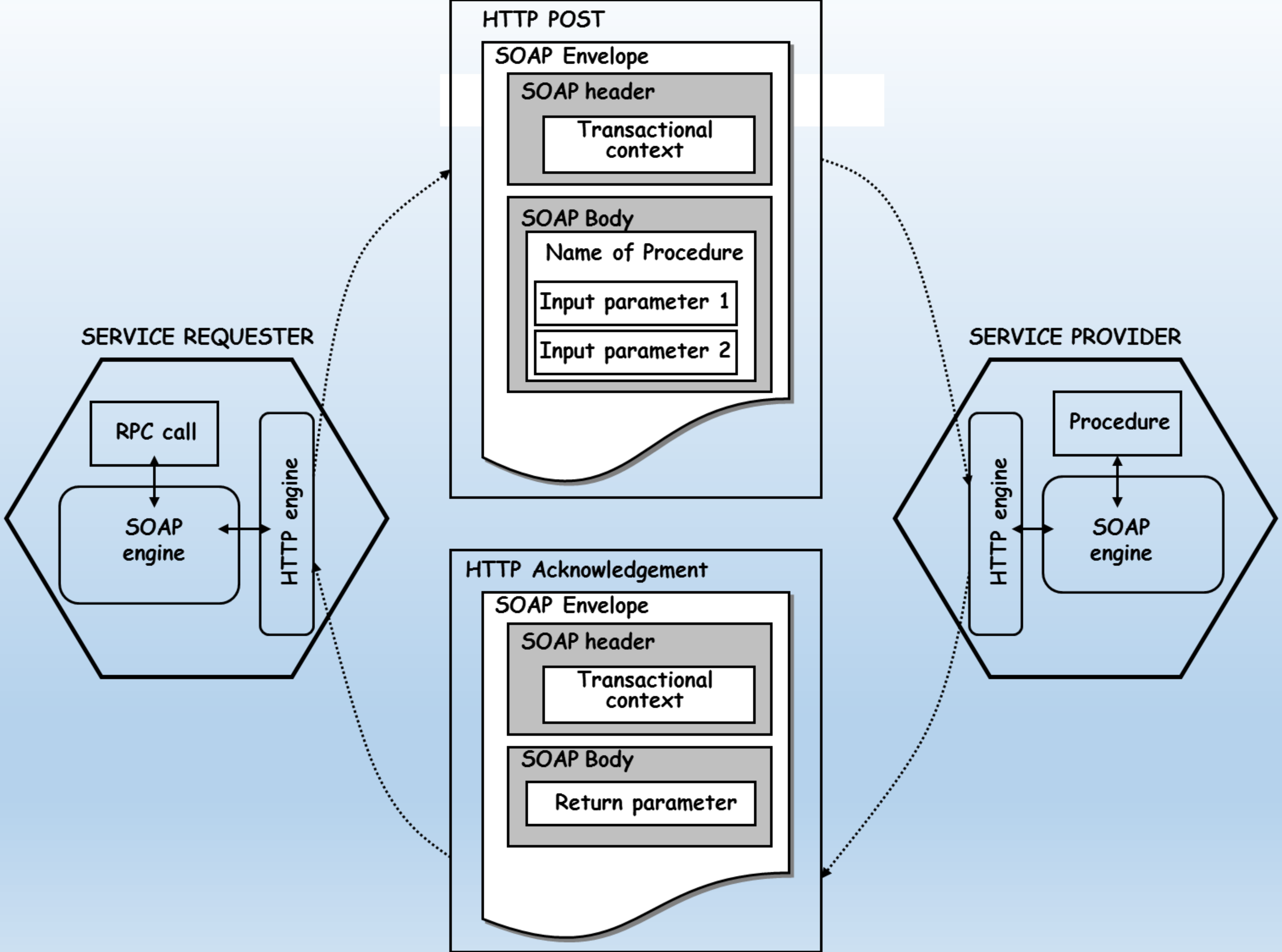


Sujata Batra

SOAP messages

- SOAP is based on message exchanges
- Messages are seen as envelops where the application encloses the data to be sent
- A message has two main parts:
- header: which can be divided into blocks
- body: which can be divided into blocks
- SOAP does not say what to do with the header and the body, it only states that the header is optional and the body is mandatory
- Use of header and body, however, is implicit. The body is for application level data. The header is for infrastructure level data





HTTP POST

SOAP Envelope

SOAP header

Transactional context

SOAP Body

Name of Procedure

Input parameter 1

Input parameter 2

SERVICE REQUESTER

RPC call

SOAP engine

HTTP engine

SERVICE PROVIDER

Procedure

HTTP engine

SOAP engine

HTTP Acknowledgement

SOAP Envelope

SOAP header

Transactional context

SOAP Body

Return parameter

REST

REST stands for Representational State Transfer

- It is an architectural ***pattern*** for developing web services as opposed to a ***specification***.
- REST was first described by Roy Fielding in 2000
- REST web services communicate over the HTTP specification, using HTTP vocabulary:
  - Methods (GET, POST, etc.)
  - HTTP URI syntax (paths, parameters, etc.)
  - Media types (xml, json, html, plain text, etc)
  - HTTP Response codes.

Rest

- Representational
  - Clients possess the information necessary to identify, modify, and/or delete a web resource.
- State
  - All resource state information is stored on the client.
- Transfer
  - Client state is passed from the client to the service through HTTP.

Resource

- In a REST based architecture you typically have a REST server which provides access to the resources
  - and a REST client which accesses and modify the REST resources.
- Every resource should support the HTTP common operations.
  - Resources are identified by URI

Content negotiation

- REST allows that resources have different representations,
  - e.g. text, xml, json etc.
  - The rest client can ask for specific representation via the HTTP protocol .

Http-Rest

- The **HTTP request** is sent *from the client*.
  - Identifies the location of a **resource**.
  - Specifies the **verb**, or HTTP **method** to use when accessing the resource.
  - Supplies optional **request headers** (name-value pairs) that provide additional information the server may need when processing the request.
  - Supplies an optional **request body** that identifies additional data to be uploaded to the server (e.g. form parameters, attachments, etc.)

Http-Rest

Sample Client Requests:

- A typical client GET request:

```
GET /view?id=1 HTTP/1.1
User-Agent: Chrome
Accept: application/json
[CRLF]
```

- A typical client POST

```
POST /save HTTP/1.1
User-Agent: IE
Content-Type: application/x-www-form-urlencoded
[CRLF]
name=x&id=2
```

Http-Rest response

- The **HTTP response** is sent *from the server*.
  - Gives the **status** of the processed request.
  - Supplies **response headers** (name-value pairs) that provide additional information about the response.
  - Supplies an optional **response body** that identifies additional data to be downloaded to the client (html, xml, binary data, etc.)

Sample Server Responses:

```
HTTP/1.1 200 OK                Response Status
Content-Type: text/html
Content-Length: 1337
[CRLF]
<html>
  <!-- Some HTML Content. -->
</html>
```

Rest web service framework

A REST service framework provides a **controller** for routing HTTP requests to a request handler according to:

- The HTTP method used (e.g. GET, POST)
- Supplied path information (e.g /service/listItems)
- Query, form, and path parameters
- Headers, cookies, etc.

SOAP Vs REST

No.	SOAP	REST
1)	SOAP is a **protocol**.	REST is an **architectural style**.
2)	SOAP stands for **Simple Object Access Protocol**.	REST stands for **REpresentational State Transfer**.
3)	SOAP **can't use REST** because it is a protocol.	REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP.
4)	SOAP **uses services interfaces to expose the business logic**.	REST **uses URI to expose business logic**.
5)	**JAX-WS** is the java API for SOAP web services.	**JAX-RS** is the java API for RESTful web services.
6)	SOAP **defines standards** to be strictly followed.	REST does not define too much standards like SOAP.
7)	SOAP **requires more bandwidth** and resource than REST.	REST **requires less bandwidth** and resource than SOAP.
8)	SOAP **defines its own security**.	RESTful web services **inherits security measures** from the underlying transport.
9)	SOAP **permits XML** data format only.	REST **permits different** data format such as Plain text, HTML, XML, JSON etc.
10)	SOAP is **less preferred** than REST.	REST **more preferred** than SOAP.

Rest Frameworks

- Jersey
- RestEasy
- Apache CXF

Http Methods

- GET
- PUT
  - creates a new resource.
- DELETE
  - removes the resources.
- POST

Restful web service

- A RESTFul webservices are based on the HTTP methods and the concept of REST.
- A RESTFul webservice typically defines the base URI for the services,
- the supported MIME-types (XML, Text, JSON, user-defined,..)
- and the set of operations (POST, GET, PUT, DELETE) which are supported.

Jersey

- *Jersey* is the reference implementation for this specification.
- Jersey contains basically a REST server and a REST client.
- The core client can be used provides a library to communicate with the server.
- On the server side Jersey uses a servlet
  - which scans predefined classes to identify RESTful resources.
  - Via the web.xml configuration file for your web application,

Jersey Servlet

- This servlet analyzes the incoming HTTP request
- selects the correct class and method to respond to this request.
- This selection is based on annotations in the class and methods.
- Jersey servlet will be instructed via the web.xml to scan certain packages for data classes.
- JAX-RS supports the creation of XML and JSON via the Java Architecture for XML Binding (JAXB).

Annotation	Description
@PATH(your_path)	Sets the path to base URL + /your_path. The base URL is based on your application name, the servlet and the URL pattern from the web.xml" configuration file.
@POST	Indicates that the following method will answer to a HTTP POST request
@GET	Indicates that the following method will answer to a HTTP GET request
@PUT	Indicates that the following method will answer to a HTTP PUT request
@DELETE	Indicates that the following method will answer to a HTTP DELETE request

Annotation	Description
@Produces(MediaType.TEXT_PLAIN [, more-types])	@Produces defines which MIME type is delivered by a method annotated with @GET. In the example text ("text/plain") is produced. Other examples would be "application/xml" or "application/json".
@Consumes(type [, more-types])	@Consumes defines which MIME type is consumed by this method.
@PathParam	Used to inject values from the URL into a method parameter. This way you inject for example the ID of a resource into the method to get the correct object.