|  |  |
| --- | --- |
| **Roll No.:** A059 | **Name:** Chinmay Parikh |
| **Prg/Yr/Sem:** B.Tech(I.T.)/4th /7 | **Batch:** A3 |
| **Date of Experiment:** 23/8/2014 | **Date of Submission:** 30/8/2014 |

**Aim**: Implementing Java Web Service JEE6 Application (Interest Calculation) using JAX-WS API and accessing through a Web Client.

1. **Design Scenario –**

a. Create Java Web Application with a Web Service called Interest Calculation using JAXRPC.

b. Create Client Web Service to call the Interest Service.

You can use the JAX-RPC programming model to develop SOAP-based web service clients and endpoints. JAX-RPC enables clients to invoke web services developed across heterogeneous platforms. Likewise, JAX-RPC web service endpoints can be invoked by heterogeneous clients. JAX-RPC requires SOAP and WSDL standards for this cross-platform interoperability.

JAX-RPC lets people develop a web service endpoint using either a Servlet or Enterprise JavaBeans (EJB) component model. The endpoint is then deployed on either the Web or EJB container, based on the corresponding component model. Endpoints are described using a Web Services Description Language (WSDL) document. (This WSDL document can be published in a public or private registry, though this is not required). A client then uses this

WSDL document and invokes the web service endpoint.

JAX-RPC stands for Java API for XML-based RPC. It's an API for building Web services and clients that used remote procedure calls (RPC) and XML. Often used in a distributed client/server model, an RPC mechanism enables clients to execute procedures on other systems.

In JAX-RPC, a remote procedure call is represented by an XML-based protocol such as SOAP. The SOAP specification defines envelope structure, encoding rules, and a convention for representing remote procedure calls and responses. These calls and responses are transmitted as SOAP messages over HTTP.

1. **Detailed Steps for creating a web service & web service client in Net Beans 7.0**

*Steps –*

Part 1

1. Start NetBeans 7.0 by clicking on it and start a new project File->New Project. Select the New Project as Java Web -> Web Application.
2. Click on Next and specify the application name as Jax-RPC.
3. Create the Java Interface :

Select the project Jax-RPC and right click on it. Select New->Java Interface. Name it JAX1.

1. Rewrite the source code for JAX1.java interface to Create a Web Service Endpoint Interface.
2. Select the project JAX-RPC and right click it. Select New->Web Service. Call it JAXImpl in package pack1.
3. Rewrite the source code for JAXImpl.java to Create a Web Service Endpoint Implementation.
4. Select the source packages of Jax-RPC project and right click on it. Select New->Java Class. Call it Publisher.
5. Rewrite the source code of Publisher.java to Create a Endpoint Publisher.
6. Build & Deploy the project Jax-RPC Project. Select the Web Service JAXImpl and right click on it and select Test Web Service. This displays the Tester with the WSDL Document. Store the WSDL URL for later use.

Part 2

1. Build & Deploy the project Jax-RPC Project. Select the Web Service JAXImpl and right click on it and select Test Web Service. This displays the Tester with the WSDL Document. Store the WSDL URL for later use.
2. Now drag the interest property into the index.jsp pageof the JAX-RPCClient project, in the body section.
3. Run the project. This will display the jsp page in IE with the interest output.
4. **Questions –**
5. **Distinguish between JAX-WS and JAX-RPC API.   
   Ans:**

\* SOAP 1.2 : JAX-RPC and JAX-WS support SOAP 1.1. JAX-WS also supports SOAP 1.2.

\* XML/HTTP : The WSDL 1.1 specification defined an HTTP binding, which is a means by which you can send XML messages over HTTP without SOAP. JAX-RPC ignored the HTTP binding. JAX-WS adds support for it.

\* WS-I's Basic Profiles : JAX-RPC supports WS-I's Basic Profile (BP) version 1.0. JAX-WS supports BP 1.1. (WS-I is the Web services interoperability organization.)

\* New Java features : JAX-RPC maps to Java 1.4. JAX-WS maps to Java 5.0. JAX-WS relies on many of the features new in Java 5.0.Java EE 5, the successor to J2EE 1.4, adds support for JAX-WS, but it also retains support for JAX-RPC, which could be confusing to today's Web services novices.

\* The data mapping model : JAX-RPC has its own data mapping model, which covers about 90 percent of all schema types. Those that it does not cover are mapped to javax.xml.soap.SOAPElement. JAX-WS's data mapping model is JAXB. JAXB promises mappings for all XML schemas.

\* The interface mapping model : JAX-WS's basic interface mapping model is not extensively different from JAX-RPC's; however: JAX-WS's model makes use of new Java 5.0 features. JAX-WS's model introduces asynchronous functionality.

\* The dynamic programming model : JAX-WS's dynamic client model is quite different from JAX-RPC's. Many of the changes acknowledge industry needs: It introduces message-oriented functionality. It introduces dynamic asynchronous functionality. JAX-WS also adds a dynamic server model, which JAX-RPC does not have.

1. **Explain the WS\* Framework.**

**Ans:** WS\* framework is characterized by:

* An abstract (vendor-neutral) existence defined by standards organizations and implemented by (proprietary) technology platforms
* Core building blocks that include Web services, service descriptions, and messages
* A communications agreement centered around service descriptions based on WSDL
* A messaging framework comprised of SOAP technology and concepts
* A service description registration and discovery architecture sometimes realized through UDDI
* A well-defined architecture that supports messaging patterns and compositions
* A second generation of Web services extensions (also known as the WS-\* specifications) continually broadening its underlying feature-set

1. **Explain the format of WSDL document.**

**Ans:** Assuming the service provides a single publicly available function, called sayHello. This function expects a single string parameter and returns a single string greeting. For example if you pass the parameter world then service function sayHello returns the greeting, "Hello, world!".

Content of HelloService.wsdl file –

<definitions name="HelloService"

targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"

xmlns="http://schemas.xmlsoap.org/wsdl/"

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"

xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<message name="SayHelloRequest">

<part name="firstName" type="xsd:string"/>

</message>

<message name="SayHelloResponse">

<part name="greeting" type="xsd:string"/>

</message>

<portType name="Hello\_PortType">

<operation name="sayHello">

<input message="tns:SayHelloRequest"/>

<output message="tns:SayHelloResponse"/>

</operation>

</portType>

<binding name="Hello\_Binding" type="tns:Hello\_PortType">

<soap:binding style="rpc"

transport="http://schemas.xmlsoap.org/soap/http"/>

<operation name="sayHello">

<soap:operation soapAction="sayHello"/>

<input>

<soap:body

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

namespace="urn:examples:helloservice"

use="encoded"/>

</input>

<output>

<soap:body

encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"

namespace="urn:examples:helloservice"

use="encoded"/>

</output>

</operation>

</binding>

<service name="Hello\_Service">

<documentation>WSDL File for HelloService</documentation>

<port binding="tns:Hello\_Binding" name="Hello\_Port">

<soap:address

location="http://www.examples.com/SayHello/">

</port>

</service>

</definitions>

*Definition:* HelloService

*Type:* Using built-in data types and they are defined in XMLSchema.

*Message:*

sayHelloRequest : firstName parameter

sayHelloresponse: greeting return value

Port Type: sayHello operation that consists of a request and response service.

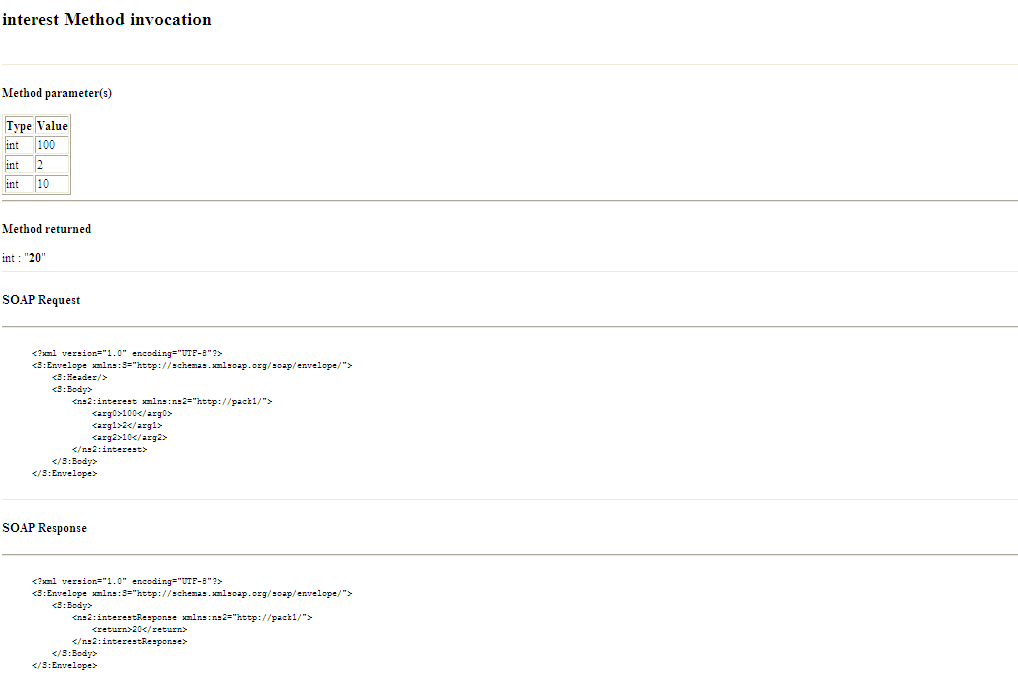
*Binding:* Directions to use the SOAP HTTP transport protocol.

*Service:* Service available at http://www.examples.com/SayHello/.

*Port:* Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

1. **Printouts –**

Part 1 –

Part 2 –



1. **Conclusion –** Hence we have learnt about JAVA Web Services and implemented a simple interest calculation program. We understood the difference between JAX-WS and JAX-RPC and accessed it through Web Client.