

DEVELOPING WEB SERVICES WITH JAVA

DESIGN WEB SERVICE ENDPOINT

CONTENTS

- Web Service Endpoints
- Packaging and deployment
- Web Service Invocation
- Steps to create Web Services using WSDL on NetBeans
- Workshops
- Exercises

WEB SERVICE DESIGN DECISIONS

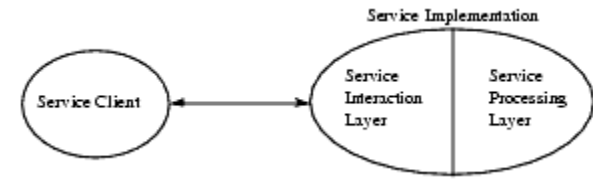
- **Decide whether and how to publish a Web Service**
 - The design of the service interface depends on:
 - The interface should reflect the type and nature of the calls that clients will make to use the service.
 - You should consider the type of endpoints you want to use--EJB service endpoints or JAX-RPC service endpoints--and when to use them.
 - The level of interoperability achieved
 - Since one reason for adding a Web service interface is to achieve interoperability, you must ensure that your design decisions do not affect the interoperability of the service as a whole.
 - The manner in which a service is published restricts the availability/visibility to clients.
- **How request are received**
 - A request made by a clients needs to be pre-processed for converting into an internal format.
 - This step helps conversion of request data into a format understandable by the business logic of the application.
- **Which protocol to use for delegating requests**
 - The pre-processed request information can be presented to the business logic in many ways.
 - Determining a fixed protocol may save processing time and reduce discrepancies.
- **How requests are processed**
 - A Web Service offers only an interface to the business logic.
 - This step helps in determining how the interface can be used to handle Web Service request.
- **Decide how responses are formulates and sent:** The response from a Web Service to the client application might have to be formatted or packaged such that the client application is able to understand it.
- **Determine how problems are reported**
 - Errors can occur in any application. Hence, deciding how to throw or handle exceptions or errors and on the system or service levels, is of utmost importance while designing a Web Service.
 - This step also includes formulating a plan for recovering from errors and exceptions.

WEB SERVICE DESIGN DECISIONS

- **After considering these steps, start designing your Web service by devising suitable answers to these questions:**
 - How will clients make use of your services? Consider what sort of calls clients may make and what might be the parameters of those calls.
 - How will your Web service receive client requests? Consider what kind of endpoints you are going to use for your Web service.
 - What kind of common preprocessing, such as transformations, translations, and logging, needs to be done?
 - How will the request be delegated to business logic?
 - How will the response be formed and sent back?
 - What kinds of exceptions will the service throw back to the clients, and when will this happen?
 - How are you going to let clients know about your Web service? Are you going to publish your service in public registries, in private registries, or some way other than registries?

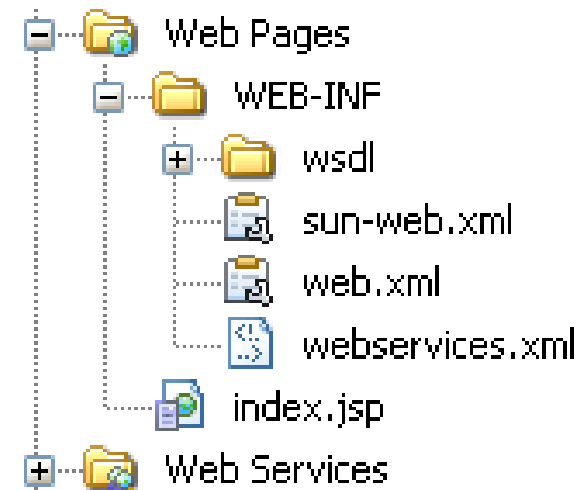
LAYERED VIEW OF A WEB SERVICES

- A Web Service implementation can be envisioned to consist of two parts
 - An interaction layer
 - Is made up of the service endpoint interface that the service exposes to the clients (JAX-RPC or EJB)
 - Contains the logic for delegating requests to the business logic and formulating responses
 - A processing layer: consists of the business logic that is used to process client requests
- Dividing the service implementation into layers help to:
 - Get clarity on the division of responsibilities
 - Designate a single location for all request processing logic
 - Describe existing business logic as a Web Service



DEPLOYMENT DESCRIPTOR STRUCTURE

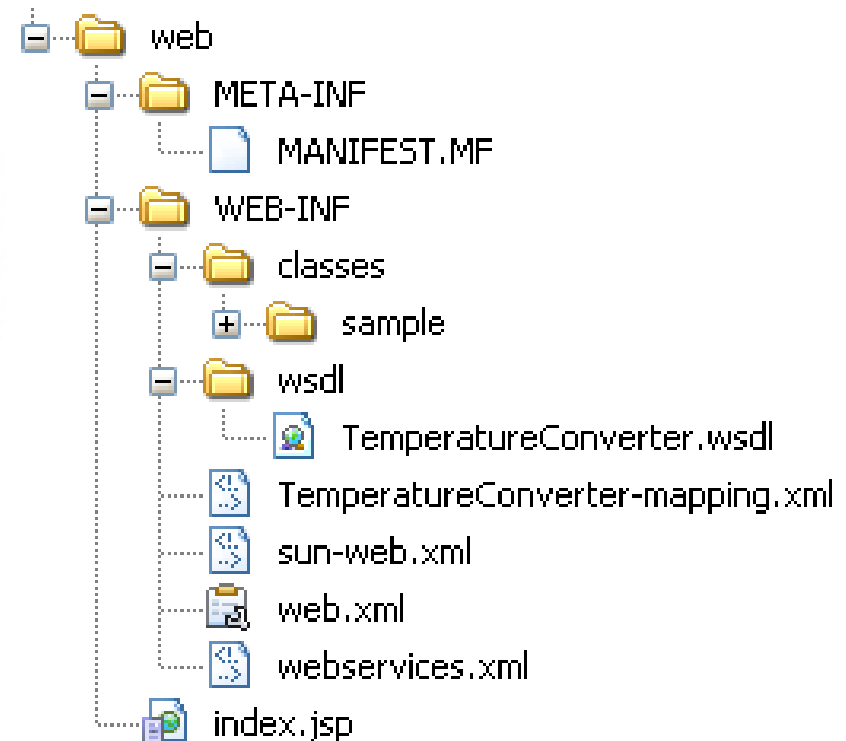
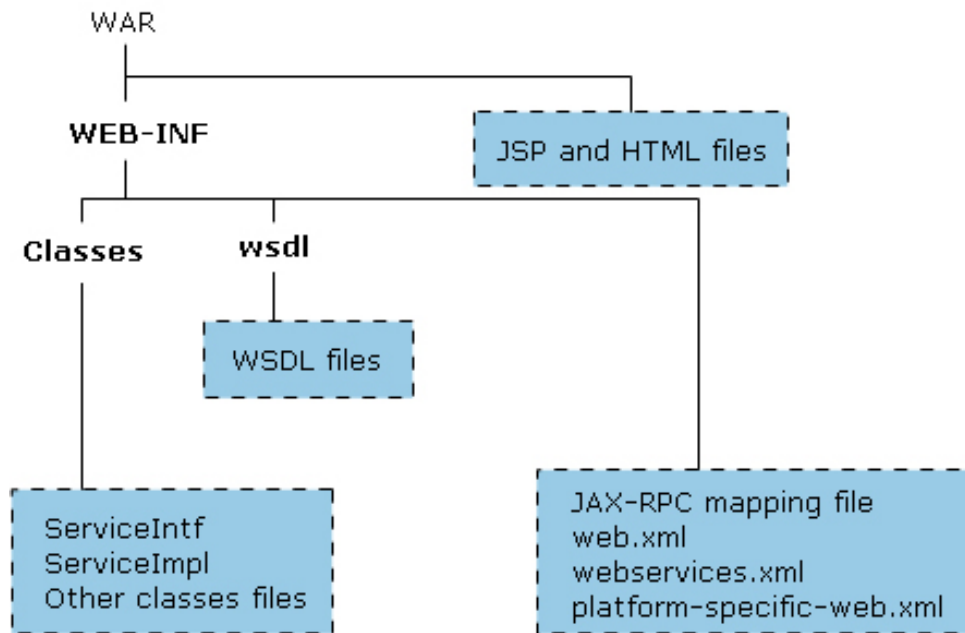
```
<?xml version='1.0' encoding='UTF-8' ?>
<webservices
  xmlns='http://java.sun.com/xml/ns/j2ee' version='1.1'>
  <webservice-description>
    <webservice-description-name>
      CurrencyConverterBean
    </webservice-description-name>
    <wsdl-file>
      WEB-INF/wsdl/CurrencyConverterBean.wsdl
    </wsdl-file>
    <jaxrpc-mapping-file>
      WEB-INF/CurrencyConverterBean-mapping.xml
    </jaxrpc-mapping-file>
    <port-component
      xmlns:wsdl-port_ns='urn:CurrencyConverterBean/wsdl'>
      <port-component-name>CurrencyConverterBean
    </port-component-name>
    ...
  </port-component>
</webservice-description>
</webservices>
```



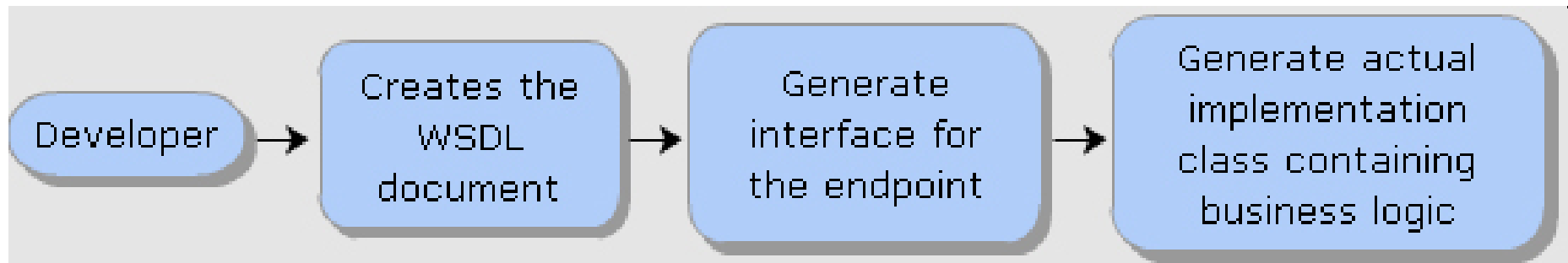
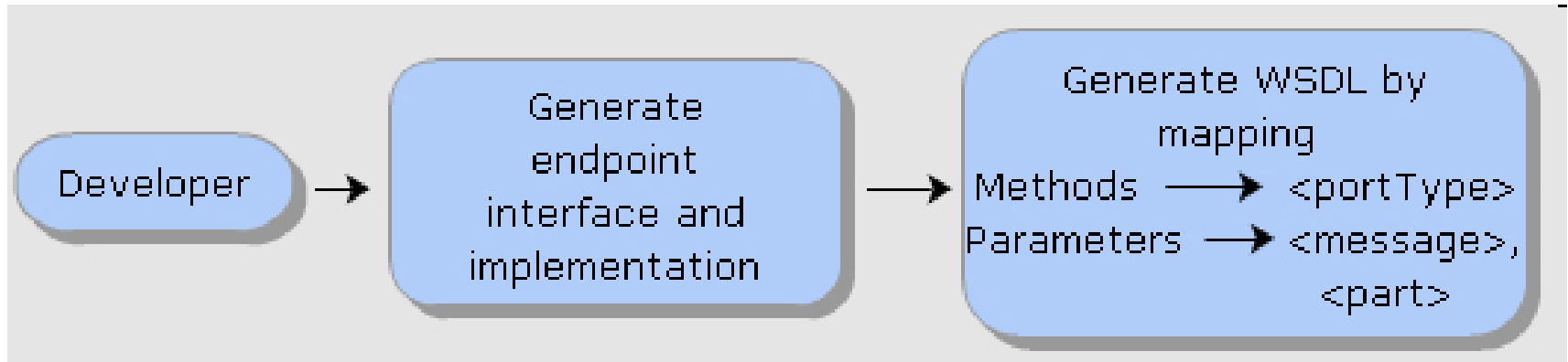
DEPLOYMENT DESCRIPTOR STRUCTURE (cont)

- **Web Services:** the webservice element is root element of the deployment descriptor file. It can declare one or more webservice-description elements.
- **Web Service-description:** webservice-description element binds J2EE endpoints to their WSDL port definitions. There is different webservice-description element for each endpoint.
- **Web Service-description-name:** is used to describe the name of the Web Service.
- **wsdl-file:** specifies the exact location where the WSDL document can be found in the J2EE archive file for the endpoint.
- **port-component:** describes the mapping between a specific JSE or EJB endpoint to a specific port element in the WSDL document.
 - **port-component-name:** identifies a unique name for a particular JSE or EJB endpoint. This name is used by the JAX-RPC mapping file to map a specific J2EE endpoint with the service name.
 - **wsdl-port:** is used to specify a single WSDL port definition in the WSDL file specified in the WSDL file element.
 - **service-endpoint-interface:** is used to declare a fully qualified name for the type of interface exposed by the client.
- **jaxrpc-mapping-file:** specifies the exact location of the JAX-RPC mapping file. This file maps the J2EE endpoint and its corresponding WSDL file.

ENDPOINT PACKAGE STRUCTURE



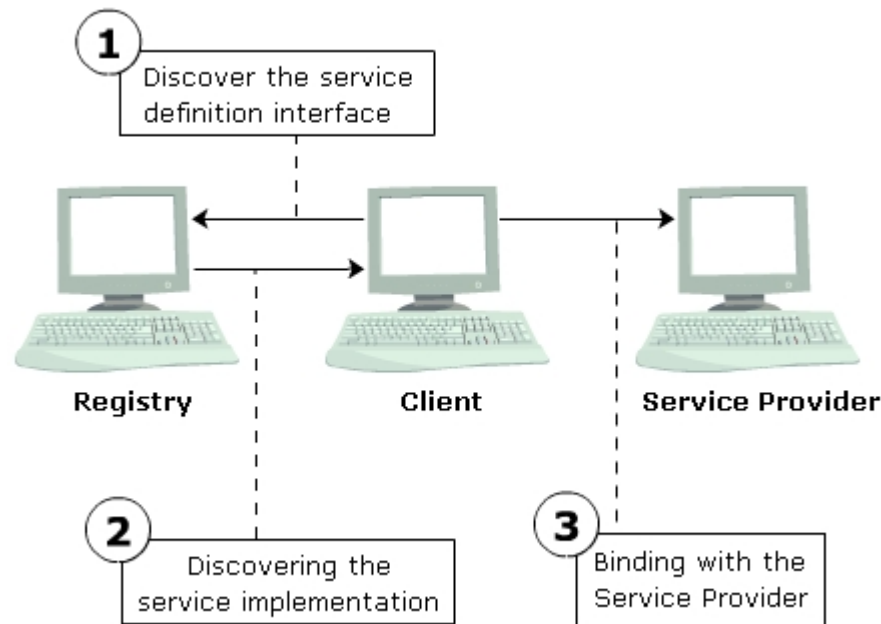
DEPLOYMENT PROCESS



PUBLISHING WEB SERVICES

- **A Web Service is published in a registry** to make it available to clients.
- **Publishing a Web Service involves** making the details about the Web Service, such as interfaces, methods, parameters and the service location etc. available to clients. This description is made available in the WSDL document which is published in the registry. A registry may hold only the Web Service's WSDL description or it may also optionally hold the XML schemas referenced by the service description.
- **Undeploying a Web Service involves disabling and removing** a service endpoint from the web container. All the associated files are removed from the server and other server resources if used are freed.

WEB SERVICE INVOCATION



- Discover the Service Definition Interface (SDI)
 - A client must know the parameters required and the return types of a Web Service's methods to make a valid invocation.
 - This process of determining method signatures is known as discovering the service definition interface.
- Discover the Service Implementation: the process of locating the actual Web Service's address is known as discovering the service implementation.
- Bind with the Service Provider
 - A client must bind to the specific location of the service to start invoking methods on it.
 - This binding can be performed when a client is developed or deployed (static binding) or at runtime (dynamic binding).
 - The type of binding, that is, static binding or dynamic binding depends on whether the client is designed for use with a specific service or usable with all services.

SDI

- The three ways in which a SDI can be obtained from the Service Provider by a client are:
 - **Direct:** a Web Service client can directly retrieve the service description from the provider by using email, FTP, etc.
 - **HTTP GET Request:** a client can obtain the service description from the provider over the web-page by using a HTTP GET request.
 - **Dynamic Discovery:**
 - The service descriptions are stored in local or public registries like UDDI or ebXML.
 - A client looks up a Web Service from these registries at runtime using a specialized set of API's.
 - This is the most commonly used method of communicating amongst Web Services and clients nowadays

LOCATING & BINDING

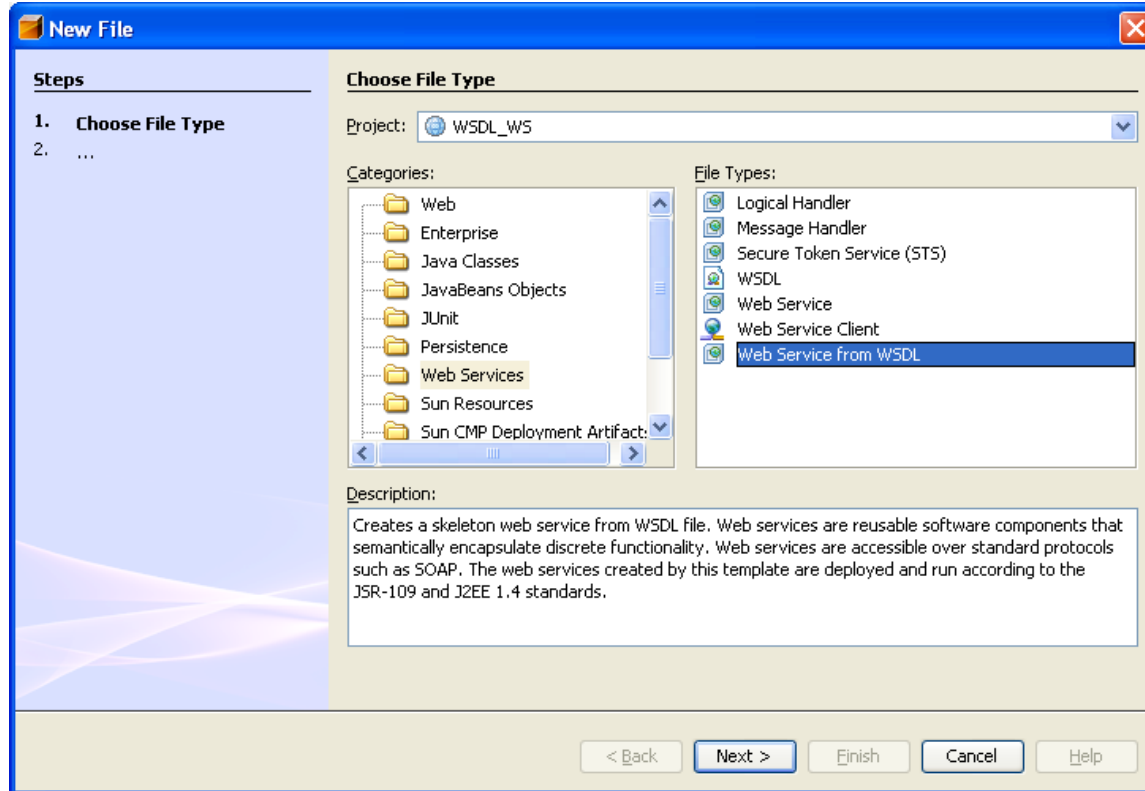
- Locating Service Implementation
 - Web Service clients query public or private registries for looking up Web Service descriptions. These queries take the form of well-formatted XML messages which are transmitted using standard protocols, such as SOAP or XML-RPC.
 - Some common criteria used to find a service are service response time, accuracy of results, supported protocols amongst others. When the appropriate service is located, the actual location of the service is returned to the requestor. Thus, the actual service implementation is obtained by the service requestor.
- Binding
 - After locating the service implementation the client creates a message to be sent to the service provider.
 - This message is sent to the provider by using the network protocols specified in the WSDL documents.
 - Finally, the client of a Web Service makes calls to the Web Service using the API specified in the WSDL document.

STEPS TO CREATE WS USING WSDL ON NETBEANS

- **Requirement:** Server is running, the WSDL, the Services is deployed on running Server
- **Step 1:** Creating Web Application
- **Step 2:** Adding Web Services to Web Application using the WSDL
- **Step 3:** Implementing the Methods/ Operations on the generated Web Services
- **Step 4:** Build the Web Services and running

STEPS TO CREATE WS USING NETBEANS

- **Step 2:** Adding Web Services to Web Application using the WSDL



- Choose the Web Services in Categories, and Web Services from WSDL in File Type
- Click Next Button

STEPS TO CREATE WS USING NETBEANS

- **Step 2:** Adding Web Services to Web Application using the WSDL (cont)
 - Note: the package name must reference namespace targetNamespace in wsdl file (ex: http://ws.sample/)

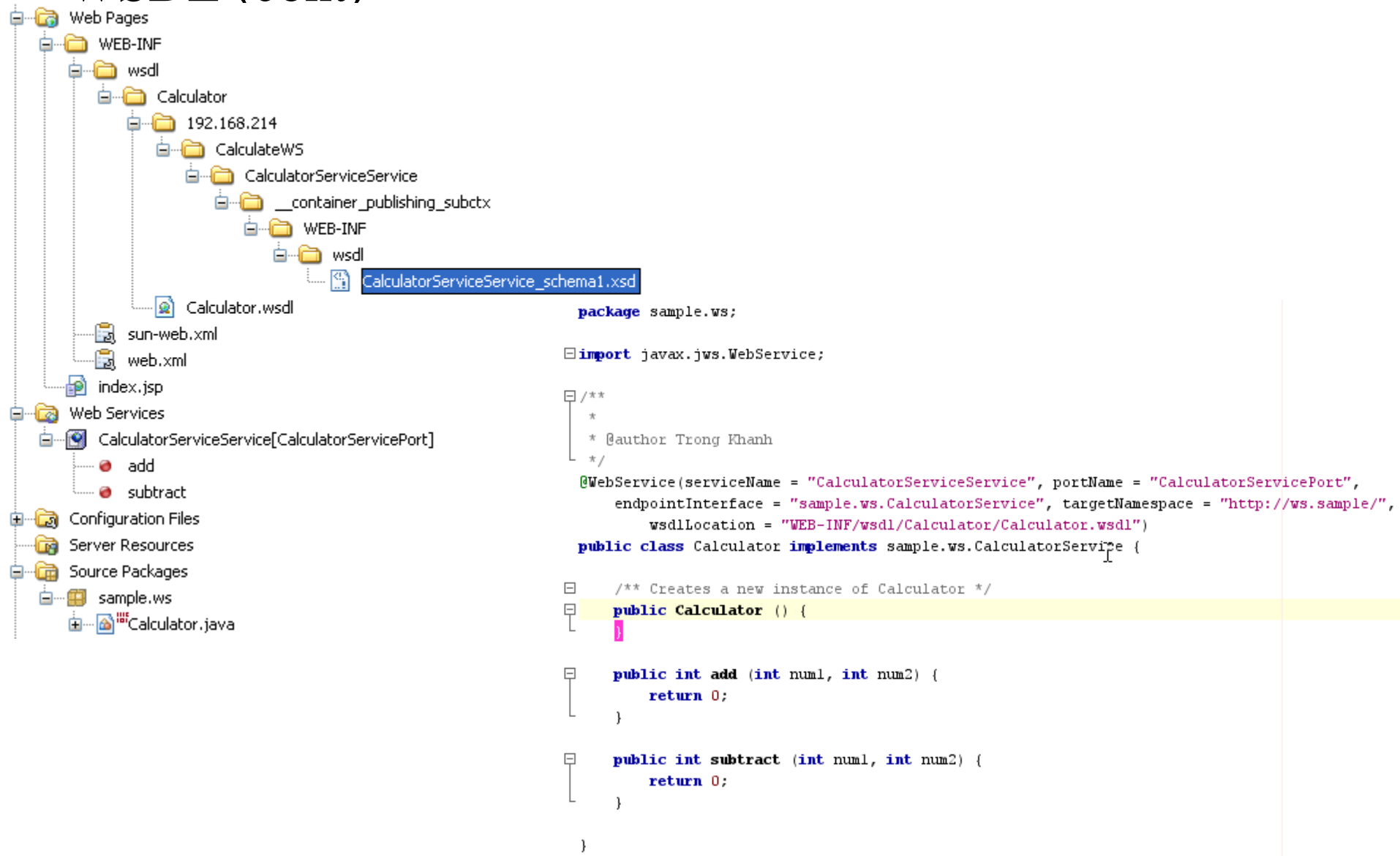
The screenshot shows the 'New Web Service from WSDL' dialog box in NetBeans. The 'Steps' pane on the left indicates the current step is '2. Name and Location'. The 'Name and Location' section contains the following fields and controls:

- Web Service Name:** A text field containing 'Calculator', highlighted with a red box and an arrow pointing to the text 'Type Web Service Name'.
- Project:** A text field containing 'WSDL_WS'.
- Location:** A dropdown menu set to 'Source Packages'.
- Package:** A text field containing 'sample.ws', highlighted with a red box and an arrow pointing to the text 'Type package Name'.
- Local WSDL File:** A text field containing 'G:\Laptrinh\Servlet\WSDL_WS\Calculator.wsdl', highlighted with a red box. To its right is a 'Browse...' button, also highlighted with a red box and an arrow pointing to the text 'Browse to the wsdl file location'.
- Web Service Port:** A text field containing 'CalculatorServiceService#CalculatorServicePort', with a 'Browse...' button to its right.

At the bottom of the dialog, there are four buttons: '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Finish' button is highlighted with a red box and an arrow pointing to the text 'Click Finish button'.

STEPS TO CREATE WS USING NETBEANS

- **Step 2:** Adding Web Services to Web Application using the WSDL (cont)



The screenshot displays the NetBeans IDE interface. On the left, the 'Project Explorer' shows the project structure. The 'Web Pages' folder contains 'WEB-INF', which includes 'wsdl' and 'Calculator'. The 'wsdl' folder contains '192.168.214', which contains 'CalculateWS', which contains 'CalculatorServiceService'. The 'CalculatorServiceService' folder contains '__container_publishing_subctx', which contains 'WEB-INF', which contains 'wsdl', which contains 'CalculatorServiceService_schema1.xsd'. The 'Calculator' folder contains 'Calculator.wsdl'. The 'Web Services' folder contains 'CalculatorServiceService[CalculatorServicePort]', which has 'add' and 'subtract' operations. The 'Configuration Files' folder contains 'sun-web.xml' and 'web.xml'. The 'Server Resources' folder contains 'index.jsp'. The 'Source Packages' folder contains 'sample.ws' and 'Calculator.java'.

```
package sample.ws;

import javax.ws.WebService;

/**
 * @author Trong Khanh
 */
@WebService(serviceName = "CalculatorServiceService", portName = "CalculatorServicePort",
    endpointInterface = "sample.ws.CalculatorService", targetNamespace = "http://ws.sample/",
    wsdlLocation = "WEB-INF/wsdl/Calculator/Calculator.wsdl")
public class Calculator implements sample.ws.CalculatorService {

    /** Creates a new instance of Calculator */
    public Calculator () {

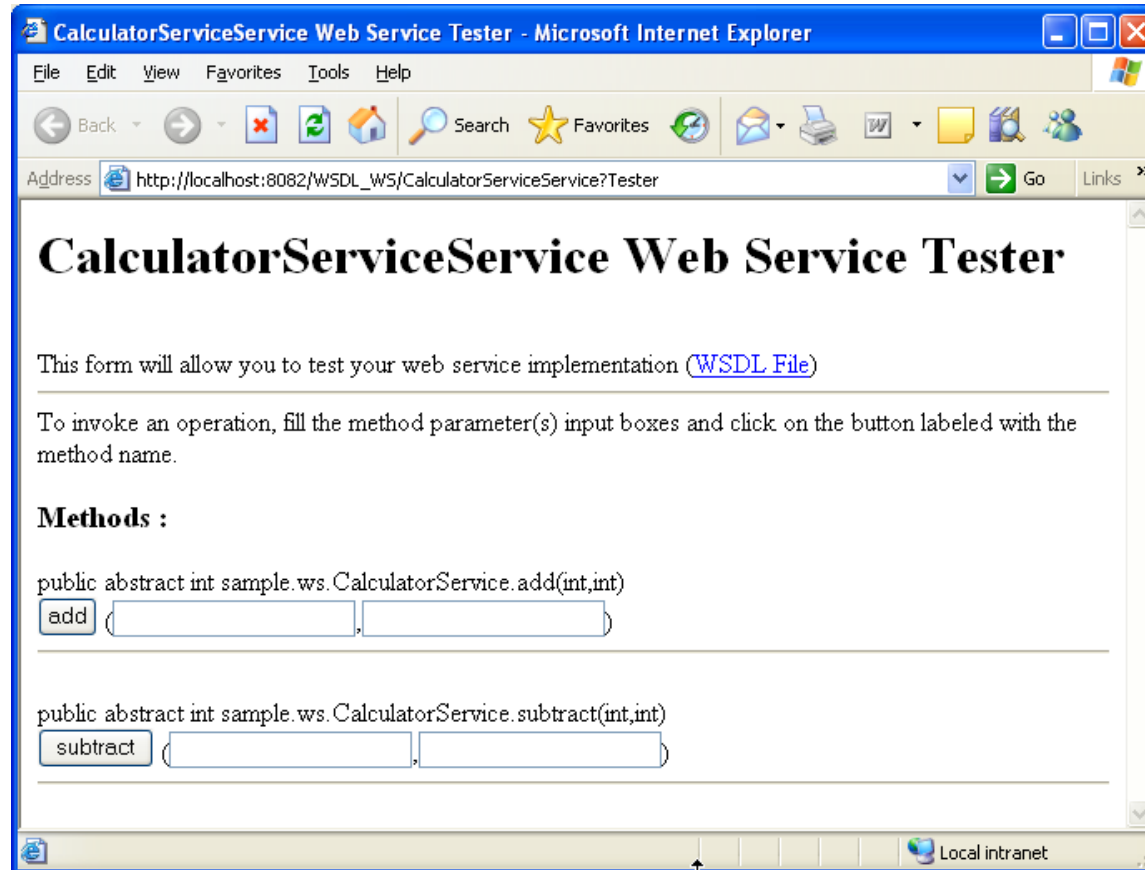
    }

    public int add (int num1, int num2) {
        return 0;
    }

    public int subtract (int num1, int num2) {
        return 0;
    }
}
```

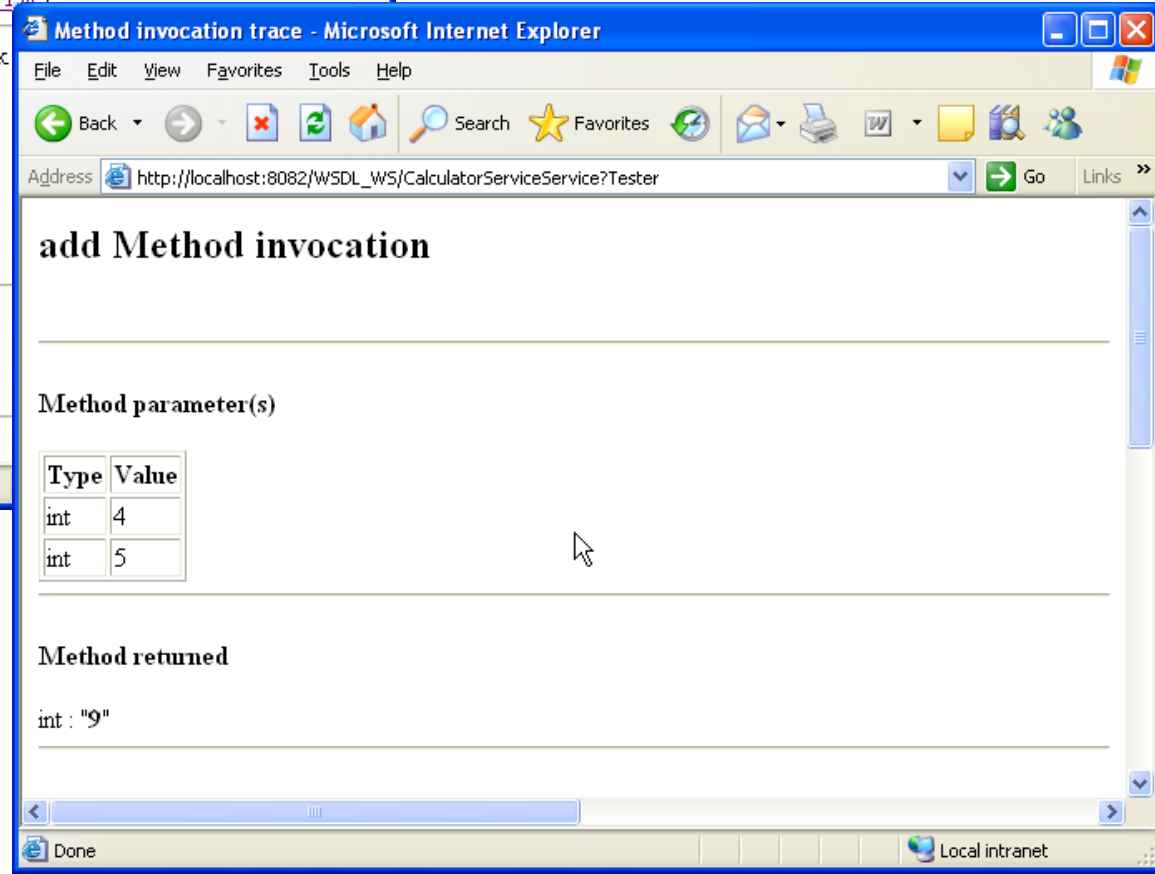
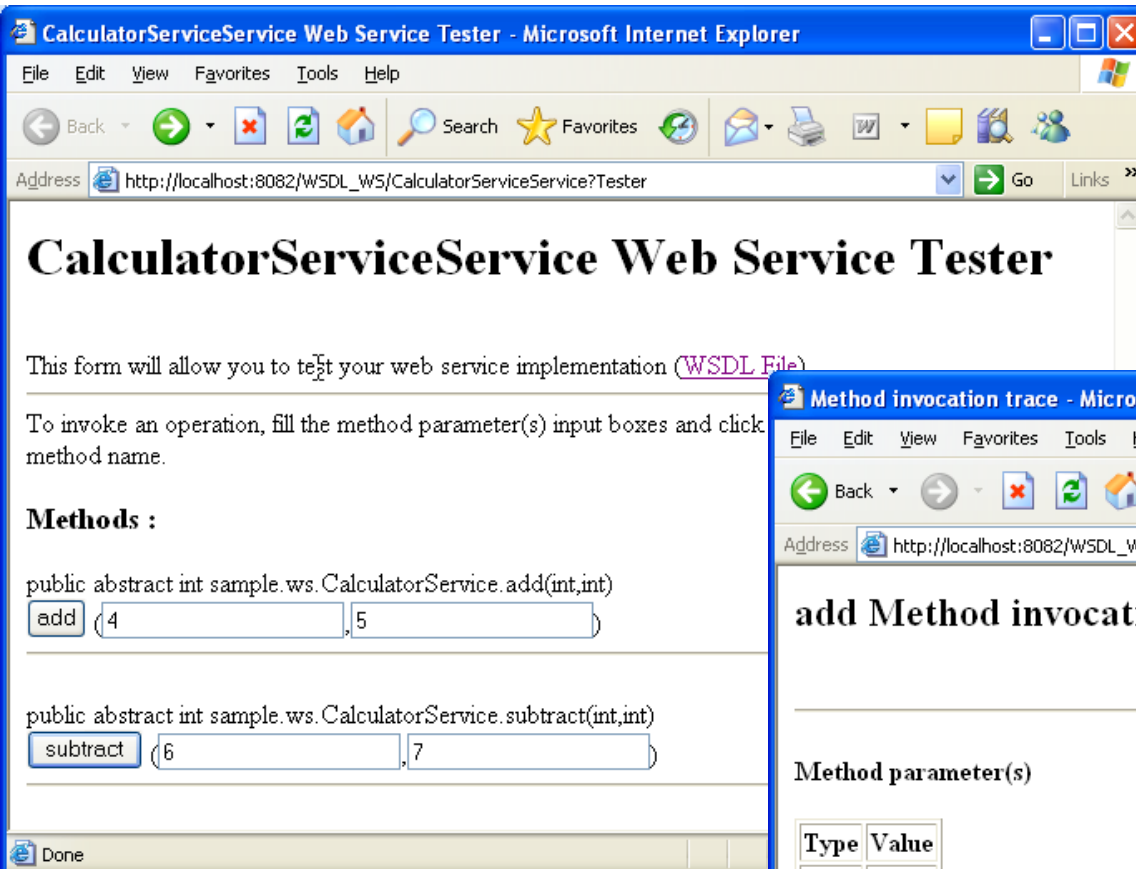
STEPS TO CREATE WS USING NETBEANS

- **Step 3:** Implementing the Methods/ Operations on the generated Web Services
- **Step 4:** Build and Run Project














STEPS TO CREATE WS USING NETBEANS

- **Step 4: Build and Run Project (cont)**



WORKSHOP ACTIVITIES

Designing Web Service Endpoint 		
Adding Sun Java System Application Server	 Show Me	 Let Me Try
Creating Web Application Project	 Show Me	 Let Me Try
Creating Web Service	 Show Me	 Let Me Try
Adding Operations to Web Service	 Show Me	 Let Me Try
Deploying and Testing Web Service	 Show Me	 Let Me Try

Building the WebService can convert from C to F degree and interaction using J2EE 1.4 on Sun Java Application Server System 9

EXERCISES

- Write the Web Service using WSDL of the exercises in first lesson