LABORATORY RECORD

NAME :	SUSHMA S
REG. NO :	2013242030
SEMESTER :	IX
DEGREE :	M.Sc.Information Technology(5 Yrs. Integ.)
COURSE CODE & TITLE :	XT9506 – Service Oriented Architecture Laboratory

DEPARTMENT OF MATHEMATICS
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI – 600 025

DEPARTMENT OF MATHEMATICS
COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI – 600 025

BONAFIDE CERTIFICATE

NAME :	SUSHMA S
REG. NO :	2013242030
SEMESTER :	IX
DEGREE :	M.Sc.Information Technology (5 Yrs. Integ.)
COURSE CODE & TITLE :	XT9506 – Service Oriented Architecture Laboratory

Certified to be the bonafide record work done by the above student in Service Oriented Architecture Laboratory course during the IX semester of the academic year 2017-2018 submitted for the Practical Examination held on _____________

Lab Course Instructor Head of the Department

INDEX

#	Date	Title of the lab exercise	Page	Signature
1	05-07-2017	STUDY OF TECHNOLOGIES IN SOA	1	
2	12-07-2017	SERVICE SCENARIO - A DISCUSSION	7	
3	19-07-2017	WORKING WITH TECHNOLOGIES IN SOA i) XML PARSING ii) DOM	9	
4	26-07-2017	XML-RPC	15	
5	02-08-2017	CREATING A WEB SERVICE USING JAVA	18	
6	09-08-2017	CONSUMING A JAVA-WS USING JAVA CLASS, SERVLET AND JSP	24	
7	16-08-2017	CREATING A WEB SERVICE USING .NET	33	
8	23-08-2017	CONSUMING A .NET-WS USING .NET	36	
9	30-08-2017	HETEROGENEOUS WEB SERVICES (JAVA-WS WITH .NET CLIENT)	40	
10	13-09-2017	HETEROGENEOUS WEB SERVICES (.NET-WS WITH JAVA CLIENT)	44	
11	20-09-2017	XML ENCRYPTION	47	
12	27-09-2017	XML DECRYPTION	50	

Ex. No. : 1 Date : 05-07-2017

Title : STUDY OF TECHNOLOGIES IN SOA

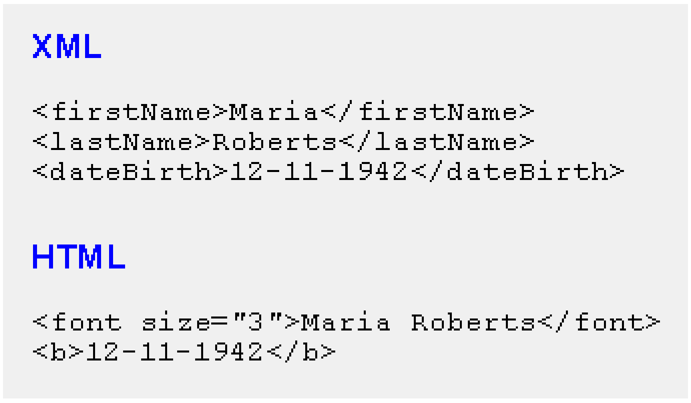
AIM

To perform a study of various technologies in SOA namely XML, SOAP, WSDL and UDDI.

TECHNOLOGIES

XML

Extensible Markup Language (XML) is used to describe data. The XML standard is a flexible way to create information formats and electronically share structured data via the public Internet, as well as via corporate networks. XML code is similar to Hypertext Markup Language (HTML). Both XML and HTML contain markup symbols to describe page or file contents. HTML code describes Web page content (mainly text and graphic images) only in terms of how it is to be displayed and interacted with.



XML

```
<firstName>Maria</firstName>
<lastName>Roberts</lastName>
<dateBirth>12-11-1942</dateBirth>
```

HTML

```
<font size="3">Maria Roberts</font>
<b>12-11-1942</b>
```

Figure 1.1 XML vs. HTML

XML data is known as self-describing or self-defining, meaning that the structure of the data is embedded with the data, thus when the data arrives there is no need to pre-build the structure to store the data; it is dynamically understood within the XML.

The basic building block of an XML document is an element, defined by tags. An element has a beginning and an ending tag. All elements in an XML document are contained in an outermost element known as the root element. XML can also support nested elements, or elements within elements. This ability allows XML to support hierarchical structures. Element names describe the content of the element, and the structure describes the relationship between the elements.

An XML document is considered to be "well formed" (that is, able to be read and understood by an XML parser) if its format complies with the XML specification, if it is properly marked up, and if elements are properly nested. XML also supports the ability to define attributes for elements and describe characteristics of the elements in the beginning tag of an element.

SOAP

SOAP is an XML-based messaging protocol. It defines a set of rules for structuring messages that can be used for simple one-way messaging but is particularly useful for performing RPC-style (Remote Procedure Call) request-response dialogues.

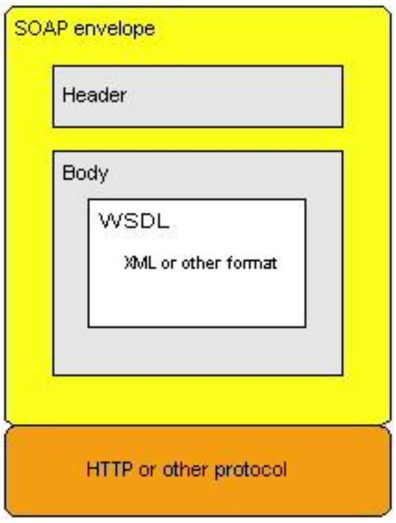
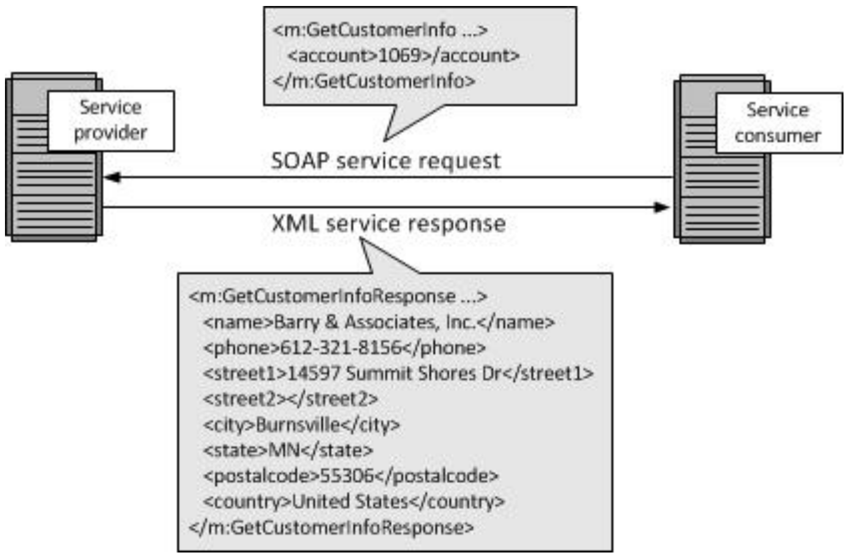


Figure 1.2 SOAP envelope

SOAP provides the envelope for sending Web Services messages over the Internet/Internet. The SOAP envelope contains two parts:

- An optional header providing information on authentication, encoding of data, or how a recipient of a SOAP message should process the message.
- The body that contains the message. These messages can be defined using the WSDL specification.

SOAP commonly uses HTTP, but other protocols such as Simple Mail Transfer Protocol (SMTP) may be used. SOAP can be used to exchange complete documents or to call a remote procedure. The following figure (Figure 1.3) illustrates using SOAP for Web Services.



```
<m:GetCustomerInfo ...>
  <account>1069>/account>
</m:GetCustomerInfo>
```

```
<m:GetCustomerInfoResponse ...>
  <name>Barry & Associates, Inc.</name>
  <phone>612-321-8156</phone>
  <street1>14597 Summit Shores Dr</street1>
  <street2></street2>
  <city>Burnsville</city>
  <state>MN</state>
  <postalcode>55306</postalcode>
  <country>United States</country>
</m:GetCustomerInfoResponse>
```

Figure 1.3 SOAP in Web Services

WSDL

Web Services Description Language (WSDL) is a format for describing a Web Services interface. It is a way to describe services and how they should be bound to specific network addresses. WSDL has three parts:

- Definitions
- Operations
- Service bindings

Definitions are generally expressed in XML and include both data type definitions and message definitions that use the data type definitions. These definitions are usually based upon some agreed upon XML vocabulary.

Operations describe actions for the messages supported by a Web service. There are four types of operations:

- One-way: Messages sent without a reply required
- Request/response: The sender sends a message and the received sends a reply.
- Solicit response: A request for a response. (The specific definition for this action is pending.)
- Notification: Messages sent to multiple receivers. (The specific definition for this action is pending.)

Operations are grouped into port types. Port types define a set of operations supported by the Web service. Service bindings connect port types to a port. A port is defined by associating a network address with a port type. A collection of ports defines a service. This binding is commonly created using SOAP.

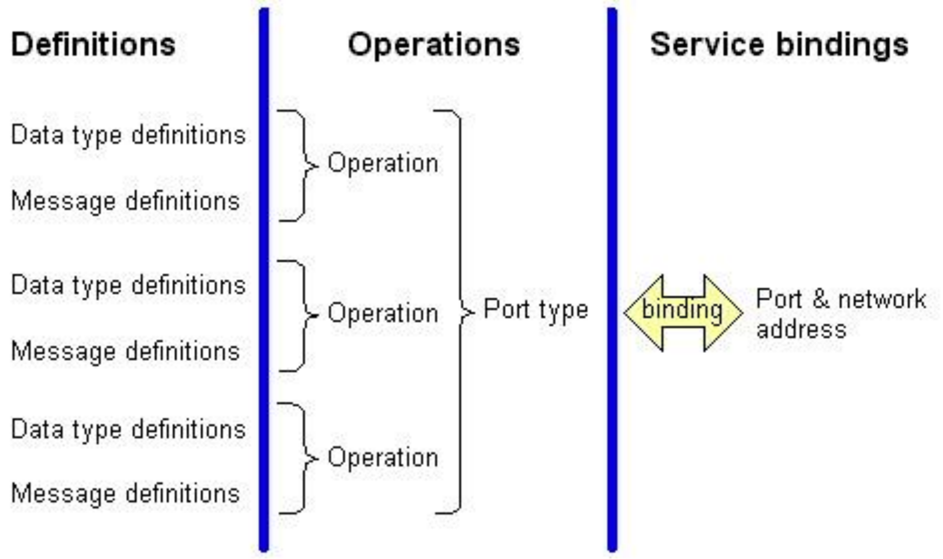


Figure 1.4 Relationship between different parts of WSDL

UDDI

Universal Description, Discovery, and Integration (UDDI) provides the definition of a set of services supporting the description and discovery of (1) businesses, organizations, and other Web Services providers, (2) the Web Services they make available, and (3) the technical interfaces which may be used to access those services. The idea is to "discover" organizations and the services that organizations offer, much like using a phone book or dialing information.

UDDI was first developed by UDDI.org and then transferred to OASIS. UDDI.org was comprised of more than 300 business and technology leaders working together to enable companies and applications to quickly, easily, and dynamically find, and use Web Services.

UDDI is based on a common set of industry standards, including HTTP, XML, XML Schema, and SOAP. It provides an infrastructure for a Web Services-based software environment for both publicly available services and services only exposed internally within an organization. The UDDI Business Registry system consists of three directories:

- UDDI white pages: basic information such as a company name, address, and phone numbers, as well as other standard business identifiers like Dun & Bradstreet and tax numbers.
- UDDI yellow pages: detailed business data, organized by relevant business classifications. The UDDI version of the yellow pages classifies businesses according to the newer NAICS (North American Industry Classification System) codes, as opposed to the SIC (Standard Industrial Classification) codes.
- UDDI green pages: information about a company's key business processes, such as operating platform, supported programs, purchasing methods, shipping and billing requirements, and other higher-level business protocols.

The UDDI technical architecture consists of three parts:

UDDI Data Model UDDI Data Model is an XML Schema for describing businesses and web services. The data model is described in detail in the "UDDI Data Model" chapter.

UDDI API Specification It is a specification of API for searching and publishing UDDI data.

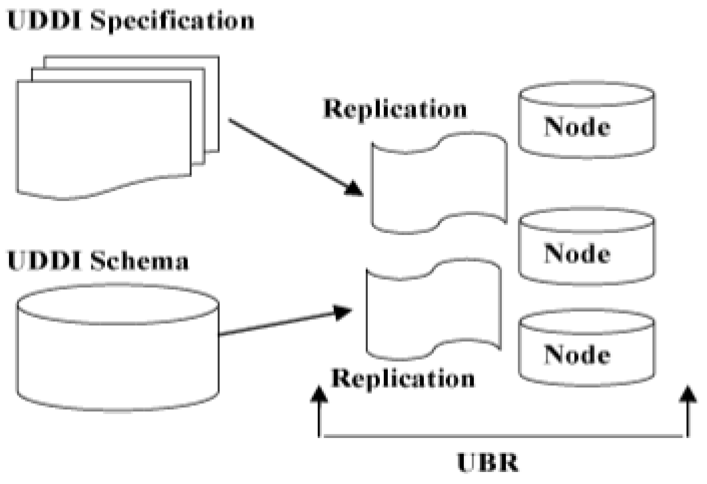


Figure 1.5 UDDI Technical Architecture

UDDI Cloud Services These are operator sites that provide implementations of the UDDI specification and synchronize all data on a scheduled basis.

The UDDI Business Registry (UBR), also known as the Public Cloud, is a conceptually single system built from multiple nodes having their data synchronized through replication.

RESULT

A detailed study has been performed on the various technologies of Service Oriented Architecture.

Ex. No. : 2 Date : 12-07-2017

Title : SERVICE SCENARIO – A DISCUSSION

AIM

To discuss in detail, the scenarios involved in the Emotion Helper Web Service.

WEB SERVICE DESCRIPTION

Emotion Helper is a Web Service that is developed for users feeling a plethora of emotions but do not know how to overcome or enhance them. This service helps them channelize their emotions in a better way.

The service takes two strings - emotion and reason as inputs and returns a string – response as result. The reason should be chosen from a dropdown menu, and there are a set of image buttons for the user to select an emotion. Based on the reason and emotion selected by the user, the server selects a response from an associated XML file.

This application intends to make the user feel better by the displaying an appropriate response.

List of emotions:

Angry, Sad, Scared, Stressed, Happy, Enthusiastic

Reasons for the emotions Angry and Sad:

Friend, Family, Unfulfilled expectation, Unfair treatment, Unexpected situation, Loss of a loved one

Reason for the emotions Scared and Stressed:

Fear of loss of loved one, Family, Work Pressure, Exam or Placement Stress, Commitments, Sickness and health.

Reason for the emotions Happy and Enthusiastic:

Achievement, Gifts given or received, Charity and good deeds, Vacation, events, parties, Family and friends, Good food

The User Interface looks like Figure 2.1. Dropdowns containing reasons are placed above the corresponding emotions. When the user selects an emotion (represented by an emoji image button) the response is displayed to him.

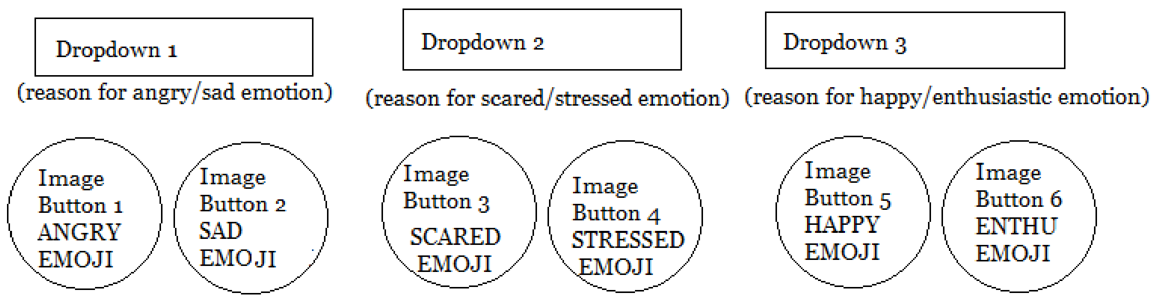

Dropdown 1
(reason for angry/sad emotion)
Dropdown 2
(reason for scared/stressed emotion)
Dropdown 3
(reason for happy/enthusiastic emotion)
Image Button 1 ANGRY EMOJI
Image Button 2 SAD EMOJI
Image Button 3 SCARED EMOJI
Image Button 4 STRESSED EMOJI
Image Button 5 HAPPY EMOJI
Image Button 6 ENTHU EMOJI

Figure 2.1 User Interface design of Emotion Helper Service

RESULT

The Web Service is described and its inputs and outputs are discussed.

Ex. No. : 3 - (i)   Date : 19-07-2017

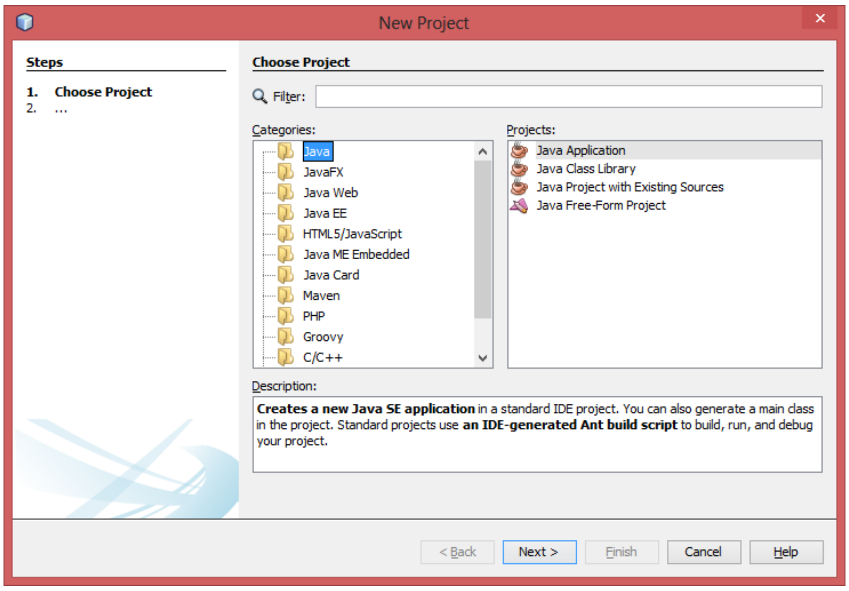
Title : WORKING WITH TECHNOLOGIES IN SOA – XML PARSING

AIM

To work with XML Parsers in Java.

PROCEDURE

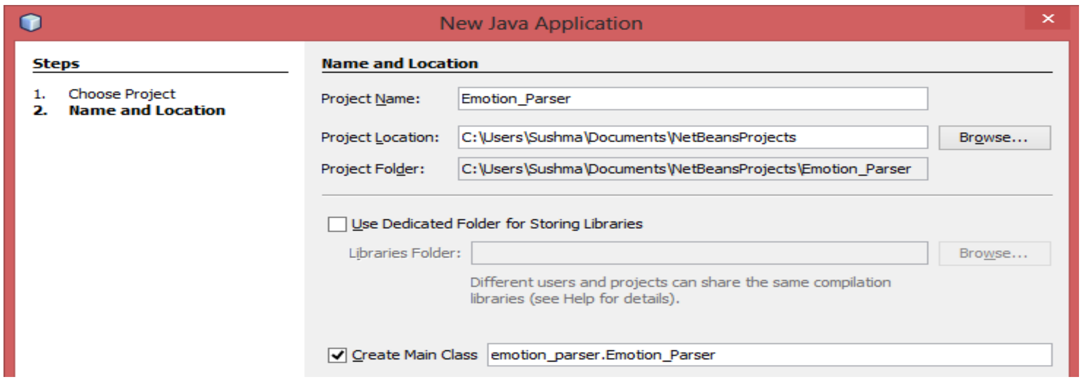
1. Open NetBeans and create a new Project. (File -> New ->Java Application)



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use **an IDE-generated Ant build script** to build, run, and debug your project.

Figure 3.1 New Java Project

2. Name the Project “Emotion_Parser” and click on Finish.



Project Name: Emotion_Parser

Project Location: C:\Users\Sushma\Documents\NetBeansProjects

Project Folder: C:\Users\Sushma\Documents\NetBeansProjects\Emotion_Parser

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Main Class emotion_parser.Emotion_Parser

Figure 3.2 Naming the project

3. To add the JDOM JAR file, Right click on Libraries under Project Node -> Select ‘Add JAR/Folder’ option and browse to select the JAR file.

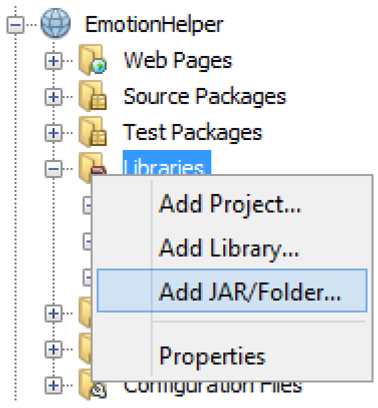
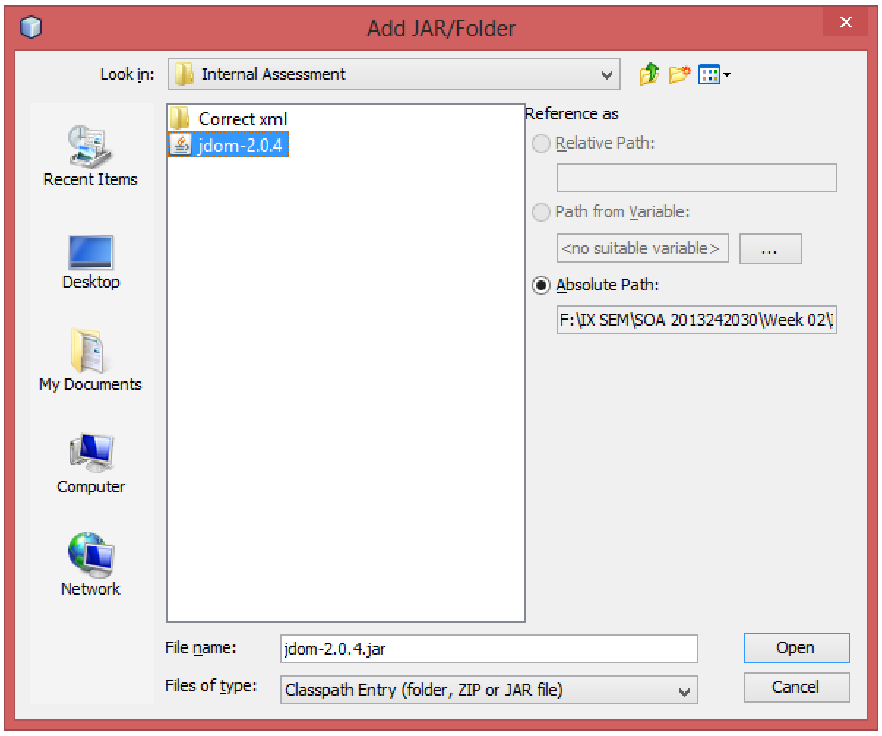


Figure 3.3 Adding JAR File



Look in: Internal Assessment

Reference as
Relative Path:
Path from Variable: <no suitable variable>
(●) Absolute Path: F:\IX SEM\SOA 2013242030\Week 02\

File name: jdom-2.0.4.jar
Files of type: Classpath Entry (folder, ZIP or JAR file)

Figure 3.4 Browsing and selecting JAR File

4. Create a UI using JFrame with image buttons and dropdowns for the user to select emotion and reason. Add action listeners for all the buttons to call the parser as in Code 3.1.

```
public class Emotion_Parser {
 JFrame f;
 String emotion = "", reason = "";
 public Emotion_Parser() //constructor
  {
   JButton angry = new JButton(new
   ImageIcon("src/images/Angry.jpg"));
   JComboBox AngrySadComboBox = new JComboBox(AngrySad);
   AngrySadComboBox.setBounds(30, 160, 135, 25);
   f.add(angry);
   angry.addActionListener(new ActionListener() {
     @Override
     public void actionPerformed(ActionEvent e) {
      emotion = "Angry";
      reason = AngrySadComboBox.getSelectedItem().toString();
      emoparse();
      }
    }

   );
```

Code 3.1 JFrame code for angry emotion

5. Create a new method emoparse() to implement the Parser logic.

6. In the method, create a Document from an input XML File.

7. Extract the root element. Traverse through its child elements.

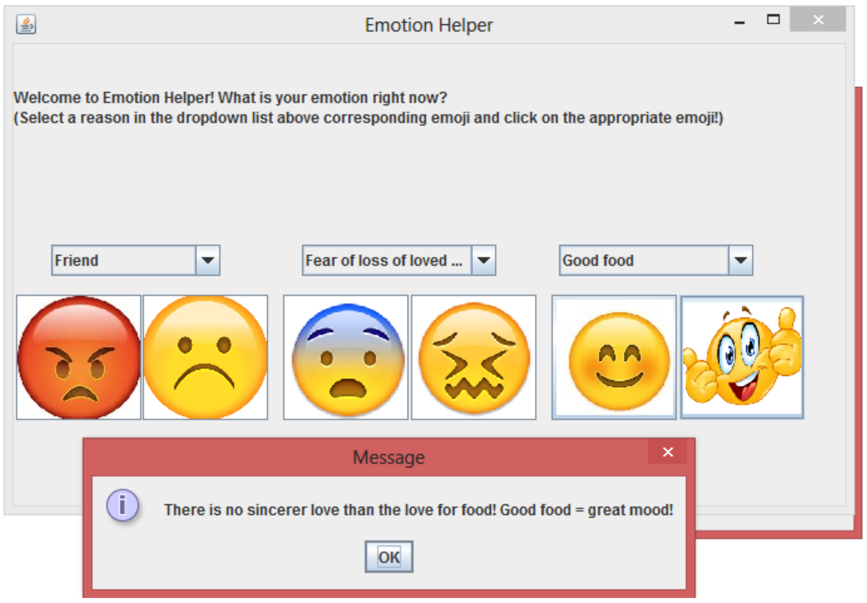
8. Extract the response for appropriate emotions by examining reasons in attributes as in Code 3.2. Display the response.

```
 File inputfile = new File("emohelp.xml");
SAXBuilder saxbuilder= new SAXBuilder();
Document doc = saxbuilder.build(inputfile);
Element rootelt = doc.getRootElement();
List<Element> rootchildren = rootelt.getChildren();

   for(int i = 0;i < rootchildren.size(); i++)
   {
       Element emo_detail = rootchildren.get(i);
       emotion1 = emo_detail.getChild("emotion").getText();
       //System.out.println("Emotion: " + emotion1);
```

```
                    if(emotion1.equals(emotion))
                    {
                        //JOptionPane.showMessageDialog(f, emotion);
                    List<Element> responses = emo_detail.getChildren();
                    for(int j=0;j< responses.size();j++)
                    {
                        Element resp = responses.get(j);
                        //System.out.println("Child: " + resp.getText());
                        if(resp.getName().equals("response"))
                        {
                          reason1 = resp.getAttributeValue("reason");
                           if(reason1.equals(reason))
                           {response = resp.getText();

                           JOptionPane.showMessageDialog(f, response);}

                        }
                      }
                   }
               }
```

Code 3.2 Code to parse XML file

OUTPUT


Emotion Helper

Welcome to Emotion Helper! What is your emotion right now?
(Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

There is no sincerer love than the love for food! Good food = great mood!

Figure 3.5 Response retrieved after parsing an XML file

RESULT

XML Parsing has been implemented successfully in Java.

Ex. No. : 3 - (ii) Date : 19-07-2017

Title : WORKING WITH TECHNOLOGIES IN SOA - DOM

AIM

To work with DOM to manipulate data in XML file

PROCEDURE

1. Create a new Java Application Project called Emotion DOM.
2. Add JDOM jar file.
3. Implement the parser logic and retrieve the data in emotion tag from an XML File.
4. Update the emotion by appending it with a string as follows

```
chld.getChild("emotion").setText(“new”);
```

Code 3.3 Setting the value of an XML node

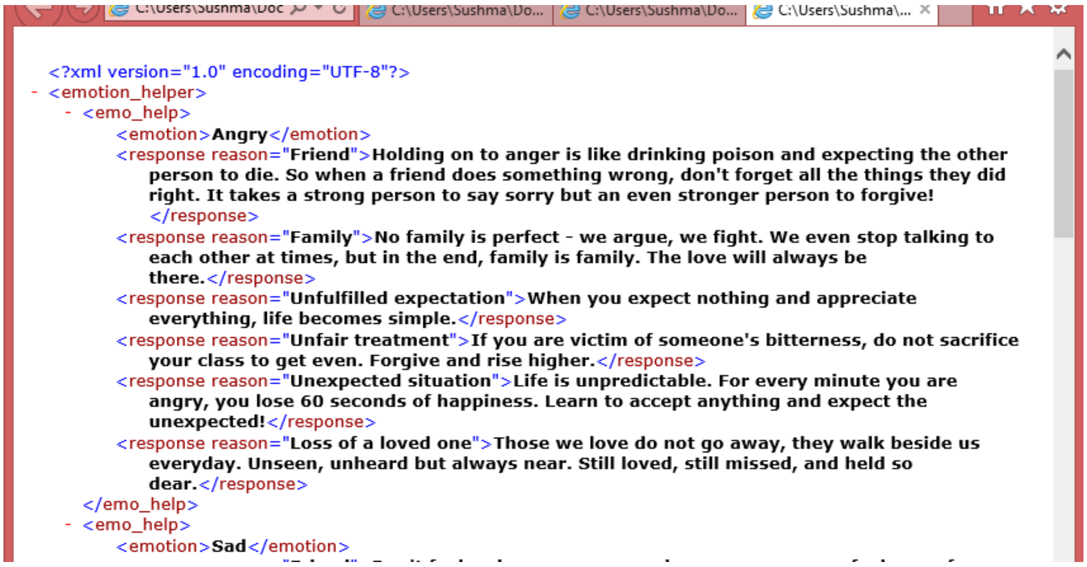
5. Replace the data in the XML file with the new updated value. Add the following code.

```
XMLOutputter xmloutput = new XMLOutputter();
xmloutput.setFormat(Format.getPrettyFormat());
xmloutput.output(doc, new FileWriter(inputfile));
```

Code 3.4 Updating the XML document

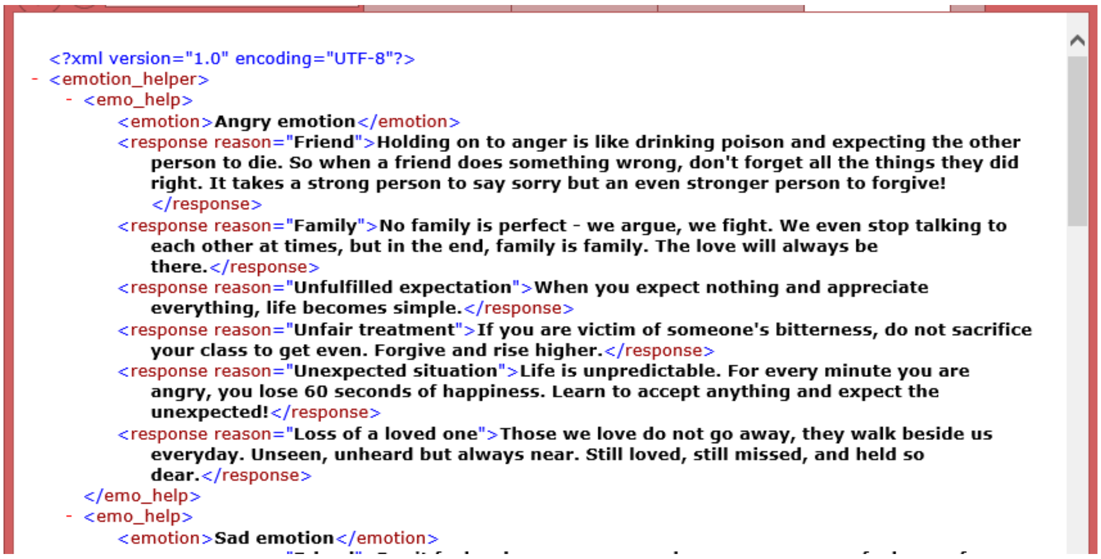
6. Open the XML file to find new values.

OUTPUT



```
  <?xml version="1.0" encoding="UTF-8"?>
- <emotion_helper>
  - <emo_help>
      <emotion>Angry</emotion>
      <response reason="Friend">Holding on to anger is like drinking poison and expecting the other
         person to die. So when a friend does something wrong, don't forget all the things they did
         right. It takes a strong person to say sorry but an even stronger person to forgive!
         </response>
      <response reason="Family">No family is perfect - we argue, we fight. We even stop talking to
         each other at times, but in the end, family is family. The love will always be
         there.</response>
      <response reason="Unfulfilled expectation">When you expect nothing and appreciate
         everything, life becomes simple.</response>
      <response reason="Unfair treatment">If you are victim of someone's bitterness, do not sacrifice
         your class to get even. Forgive and rise higher.</response>
      <response reason="Unexpected situation">Life is unpredictable. For every minute you are
         angry, you lose 60 seconds of happiness. Learn to accept anything and expect the
         unexpected!</response>
      <response reason="Loss of a loved one">Those we love do not go away, they walk beside us
         everyday. Unseen, unheard but always near. Still loved, still missed, and held so
         dear.</response>
    </emo_help>
  - <emo_help>
      <emotion>Sad</emotion>
```

Figure 3.5 Original XML File



```
  <?xml version="1.0" encoding="UTF-8"?>
- <emotion_helper>
  - <emo_help>
      <emotion>Angry emotion</emotion>
      <response reason="Friend">Holding on to anger is like drinking poison and expecting the other
         person to die. So when a friend does something wrong, don't forget all the things they did
         right. It takes a strong person to say sorry but an even stronger person to forgive!
         </response>
      <response reason="Family">No family is perfect - we argue, we fight. We even stop talking to
         each other at times, but in the end, family is family. The love will always be
         there.</response>
      <response reason="Unfulfilled expectation">When you expect nothing and appreciate
         everything, life becomes simple.</response>
      <response reason="Unfair treatment">If you are victim of someone's bitterness, do not sacrifice
         your class to get even. Forgive and rise higher.</response>
      <response reason="Unexpected situation">Life is unpredictable. For every minute you are
         angry, you lose 60 seconds of happiness. Learn to accept anything and expect the
         unexpected!</response>
      <response reason="Loss of a loved one">Those we love do not go away, they walk beside us
         everyday. Unseen, unheard but always near. Still loved, still missed, and held so
         dear.</response>
    </emo_help>
  - <emo_help>
      <emotion>Sad emotion</emotion>
```

Figure 3.6 Updated XML File

RESULT

XML document has been manipulated using DOM.

Ex. No. : 4 Date : 26-07-2017

Title : XML-RPC

AIM:

To implement XML RPC concept in Java for the Emotion Helper Service.

PROCEDURE

1. Open NetBeans and create a new Project. Name it ‘XmlRpcServer’.

2. Create a new Java class and name it ‘XmlRpcServer.java’.

3. Add JAR file for XMLRPC.

4. In the Java Class created, create a new function “EmotionHelper” with two parameters emotion(String), reason(String) and return type String, to be called remotely from the client.

```
public String EmotionHelper(String emotion,String reason)
    {  //code to parse
     return (response);
    }
```

Code 4.1 Function to parse XML document

5. For the emotion and reason specified in the arguments, retrieve and return the appropriate response by parsing the XML file “emohelp.xml”.

6. In the main function, an instance of the same server class is then associated with a handler that is accessible by the client.

7. The server is initialized by the port number (here: 80).

```
WebServer server = new WebServer(80);
server.addHandler("emotion", new XmlRpcServer());
System.out.println("Server is now accepting requests.. Halt
Program to stop");
server.start();
```

Code 4.2 RPC server logic

8. Similarly, create a new Project and an associated java class for implementing the client logic (“XmlRpcClientemotion.java”).

9. Create a User Interface for the user to select emotion and reason. Add the parameters to a Vector.

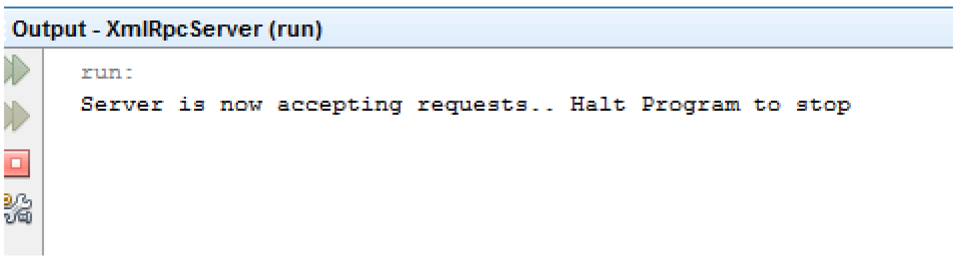
10. The function server.execute(Handler,Vector) in the client sends the request to the server.

```
XmlRpcClient server = new XmlRpcClient("http://localhost/RPC2");
Vector params = new Vector();
params.add(emotion);
params.add(reason);
Object result = server.execute("emotion.EmotionHelper", params);
String response = result.toString();
JOptionPane.showMessageDialog(f, response);
```

Code 4.3 RPC Client logic

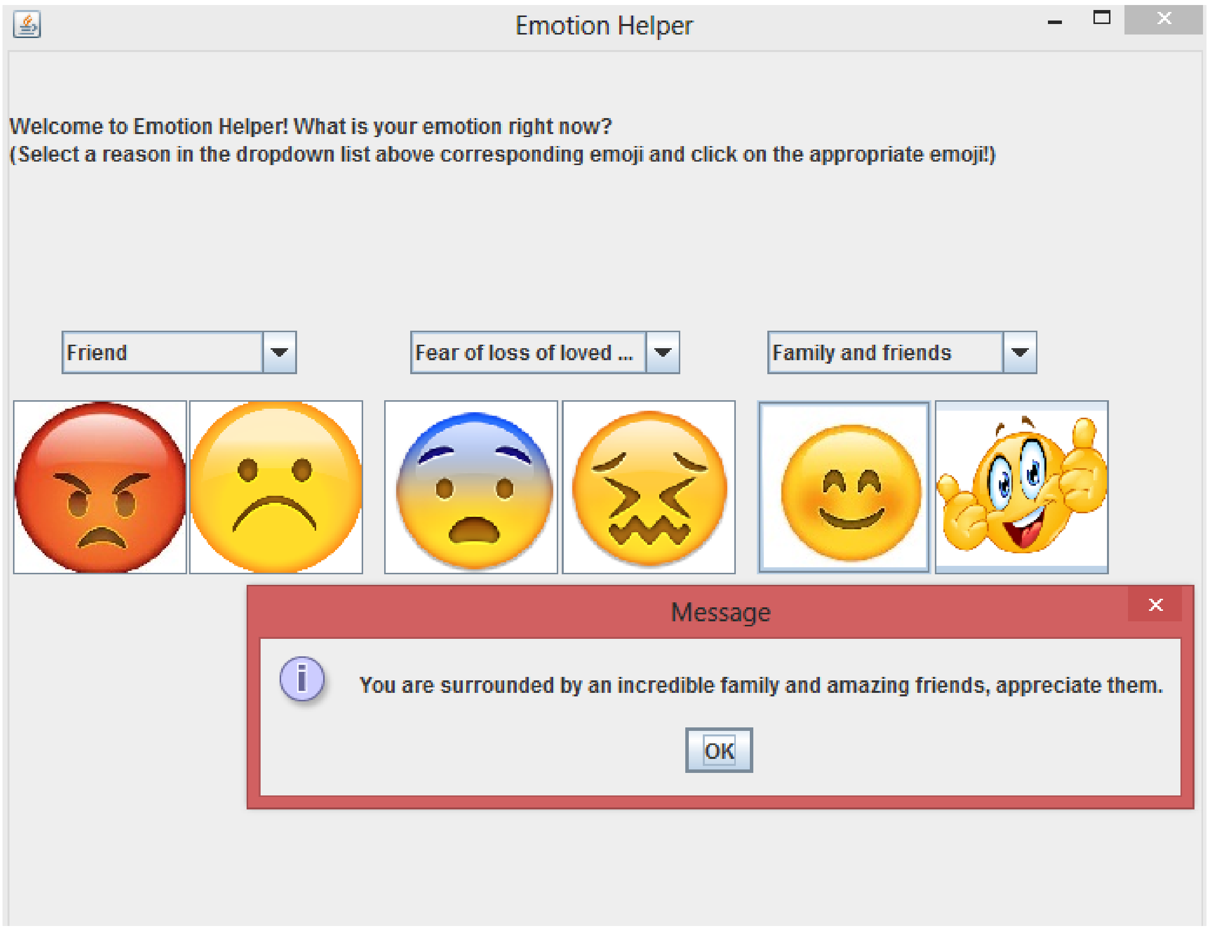
11. The return value of this procedure call on the server is always an Object which is casted to String and displayed.

OUTPUT



```
run:
Server is now accepting requests.. Halt Program to stop
```

Figure 4.1 RPC Server



Welcome to Emotion Helper! What is your emotion right now?
(Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

You are surrounded by an incredible family and amazing friends, appreciate them.

Figure 4.2 RPC Client

RESULT

XML – RPC has been successfully implemented in Java.

Ex. No. : 5 Date : 02-08-2017

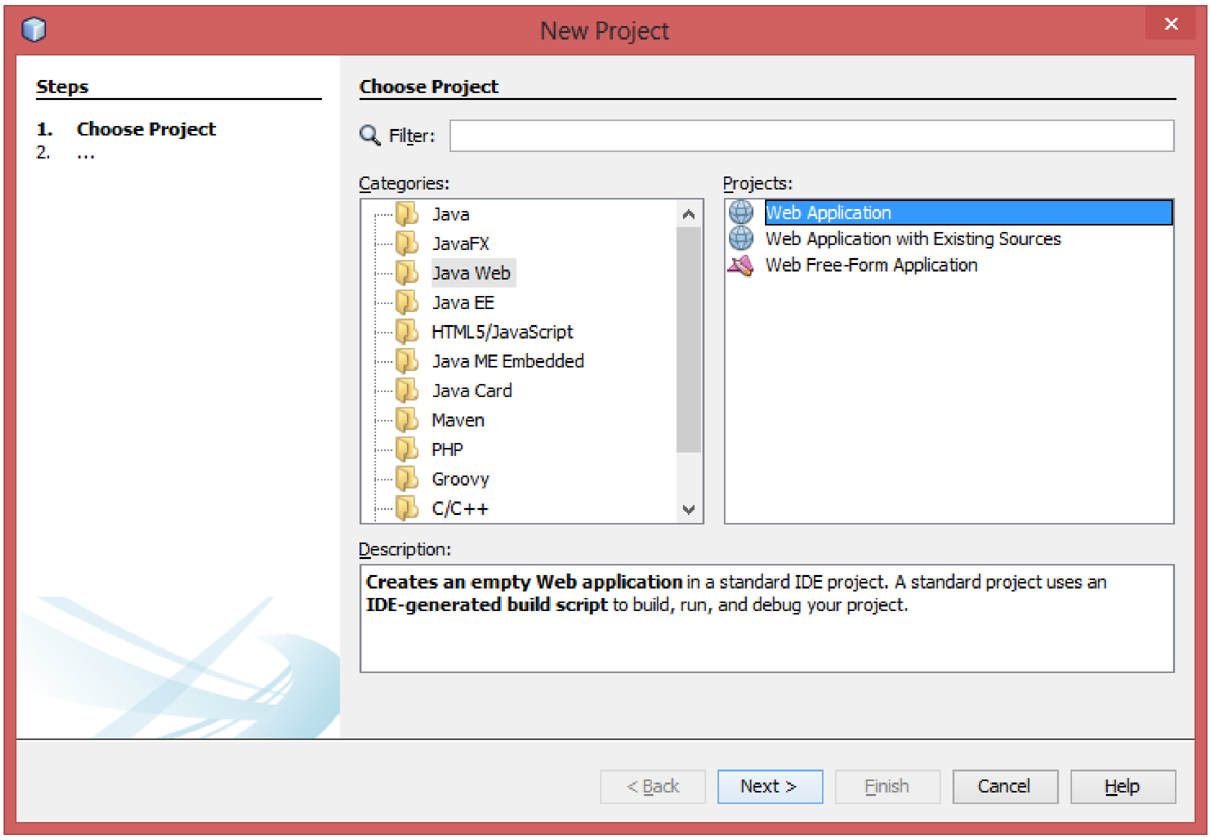
Title : CREATING A WEB SERVICE USING JAVA

AIM

To create an Emotion Helper Web Service using Java.

PROCEDURE

1. Open NetBeans. Create a new Project by selecting File -> New Project -> Java Web (Category) -> Web Application. Name the project “Emotion Helper”.



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

Figure 5.1 Creating a Web Application in Java

2. Right Click on Project Name in File explorer -> New -> Web Service to create a new Web Service and name it “EmotionHelper”. Specify a package name (emotion.help).

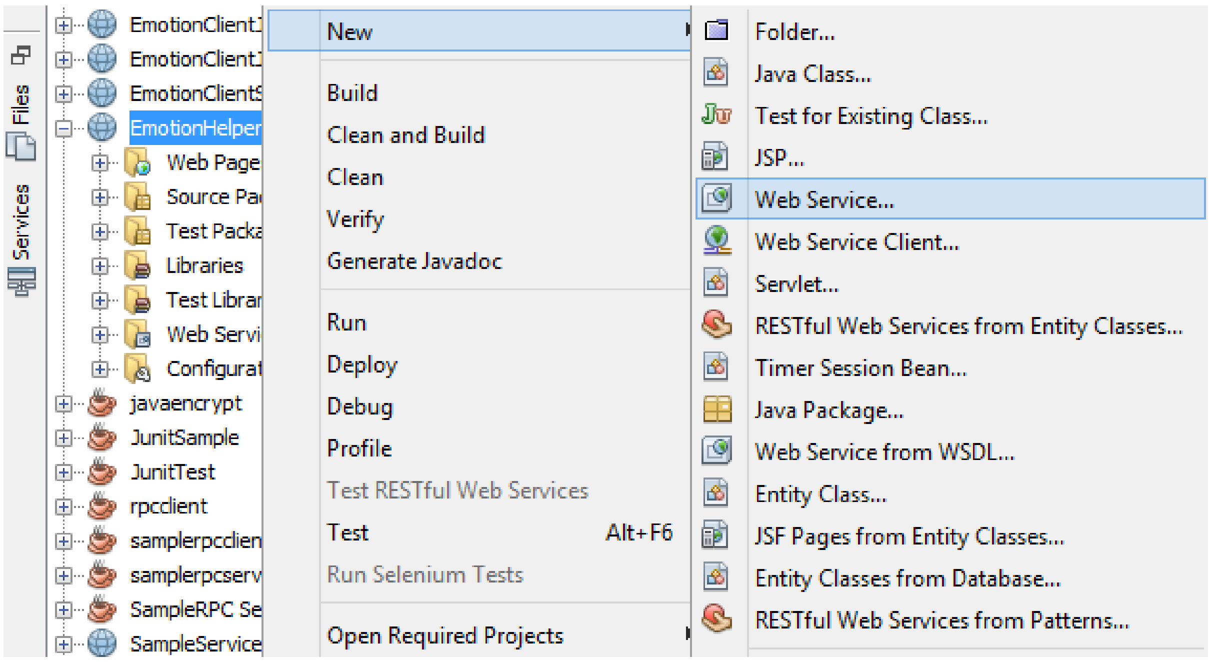
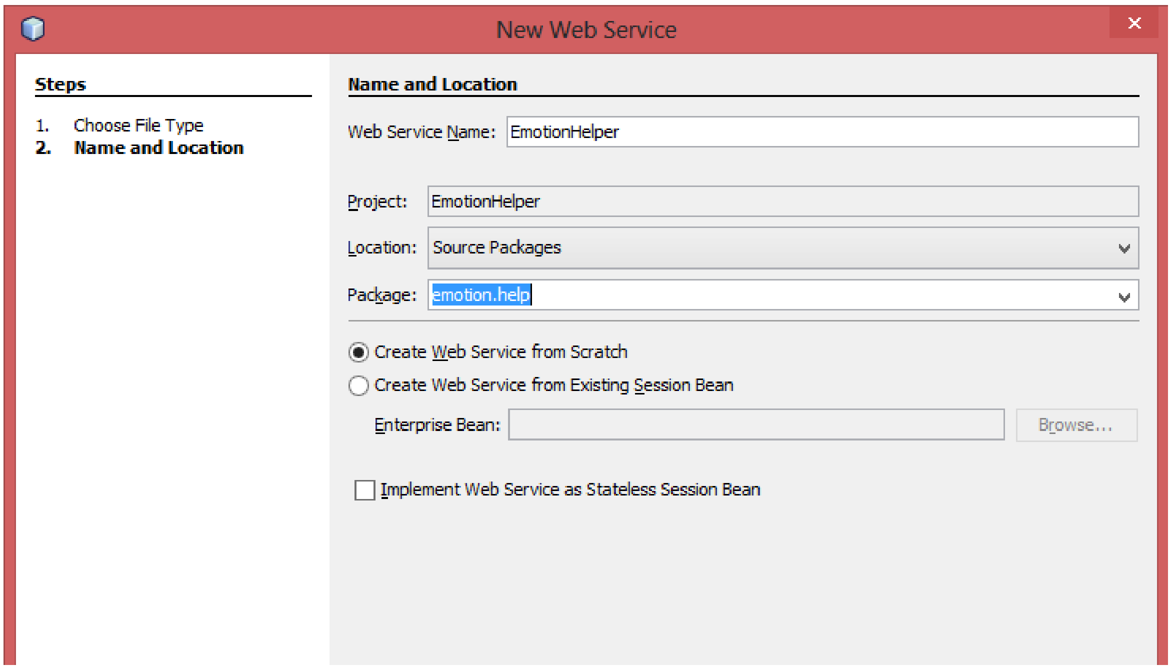


Figure 5.2 Creating a new Web Service

3. Select option ‘Create a Web Service from scratch’ -> Finish.


New Web Service

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Web Service Name: EmotionHelper

Project: EmotionHelper

Location: Source Packages

Package: emotion.help

(●) Create Web Service from Scratch

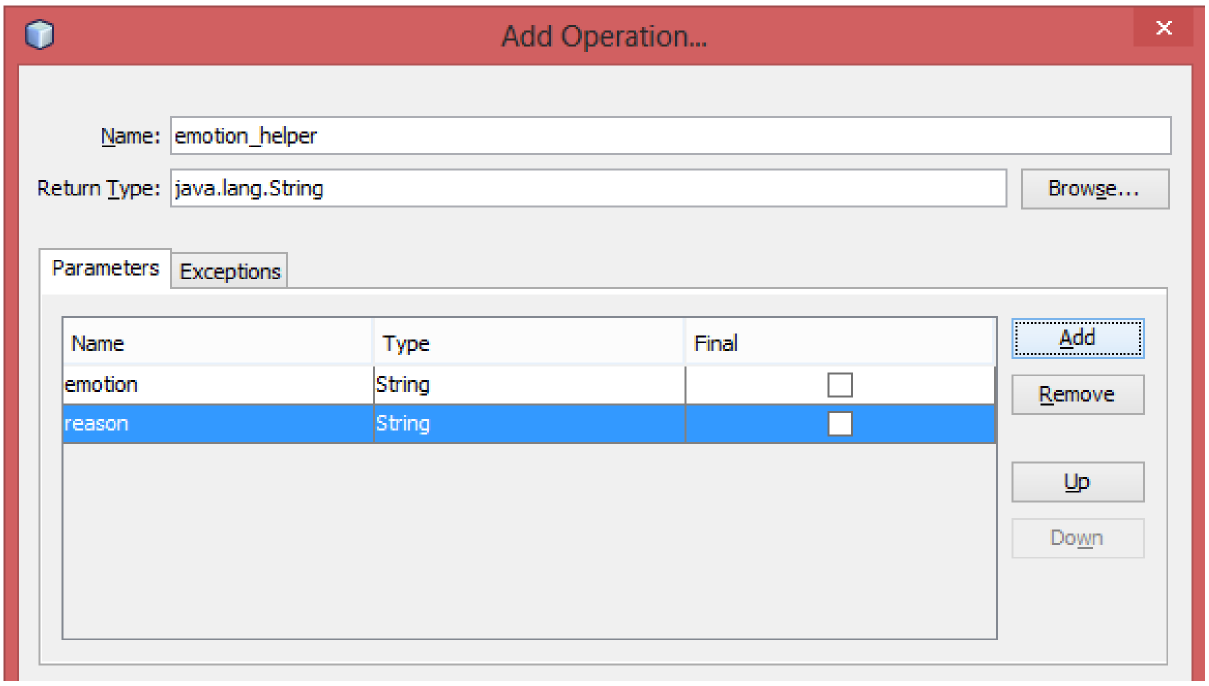
( ) Create Web Service from Existing Session Bean

Enterprise Bean: Browse...

[ ] Implement Web Service as Stateless Session Bean

Figure 5.3 Naming a Web Service

4. In the design pane, click on ‘Add operation’ and give the operation a name. Specify a return type.



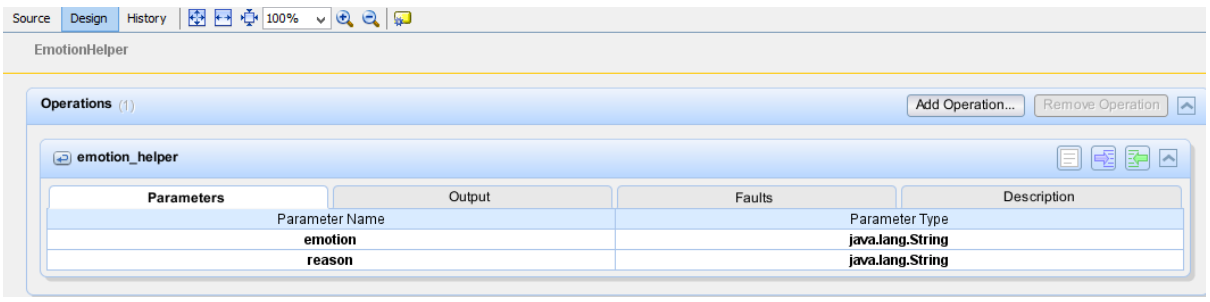
Name: emotion_helper

Return Type: java.lang.String

Name	Type	Final
emotion	String	☐
reason	String	☐

Figure 5.4 Adding a new Operation

5. Add parameters and specify their return types -> Click ‘OK’



Parameter Name	Parameter Type
emotion	java.lang.String
reason	java.lang.String

Figure 5.5 Design pane of operations

6. Go to source pane and add code for the operation logic to implement the Emotion Helper.

```
@WebService(serviceName = "EmotionHelper")
public class EmotionHelper {

    /**
     * Web service operation
     */
    @WebMethod(operationName = "emotion_helper")
       public String emotion_helper(@WebParam(name = "emotion")
   String emotion, @WebParam(name = "reason") String reason) {
        //TODO write your implementation code here:
       String response = "",emotion1="",reason1="";
               try
       {
        File inputfile = new File("emohelp.xml");
        SAXBuilder saxbuilder= new SAXBuilder();
        Document doc = saxbuilder.build(inputfile);
        Element rootelt = doc.getRootElement();
        List<Element> rootchildren = rootelt.getChildren();

           for(int i = 0;i < rootchildren.size(); i++)
           {
               Element emo_detail = rootchildren.get(i);
               emotion1 =emo_detail.getChild("emotion").getText();
               //System.out.println("Emotion: " + emotion1);
               if(emotion1.equals(emotion))
               {

               List<Element> responses = emo_detail.getChildren();
               for(int j=0;j< responses.size();j++)
               {
                   Element resp = responses.get(j);

                   if(resp.getName().equals("response"))
                   {
                      reason1 = resp.getAttributeValue("reason");
                       if(reason1.equals(reason))
                       {response = resp.getText();

                      }

                   }
               }

             }
        }}
     catch(IOException | JDOMException e)
     {
         System.err.println(e.getMessage());
         return(e.getMessage());
     }

            return (response);
    }
}
```

Code 5.1 Emotion Helper Web Service logic

7. Right click on service -> Deploy

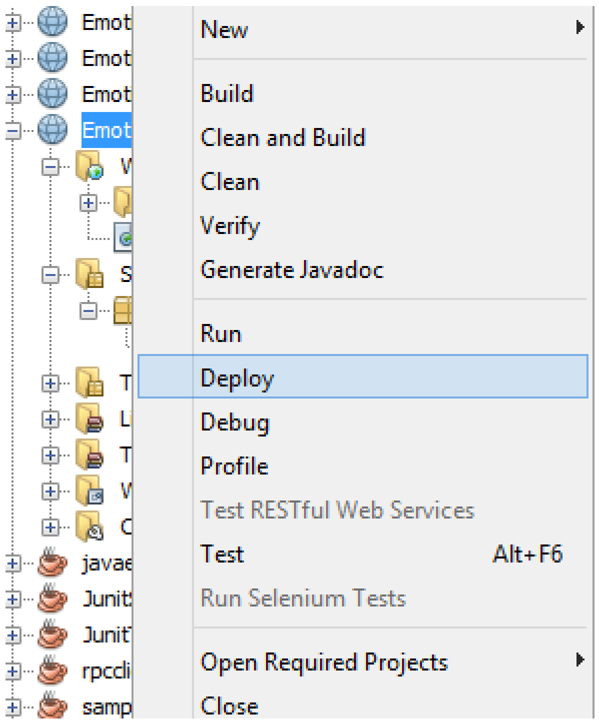


Figure 5.6 Deploying a Web Service

8. In the Files explorer under 'Web Services' section of project, Right Click on Web Service – >Test Web Service to test the web service (Service will open in a new browser window.)

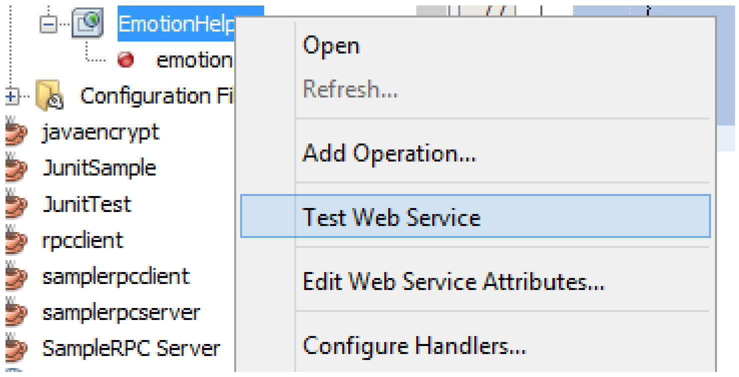
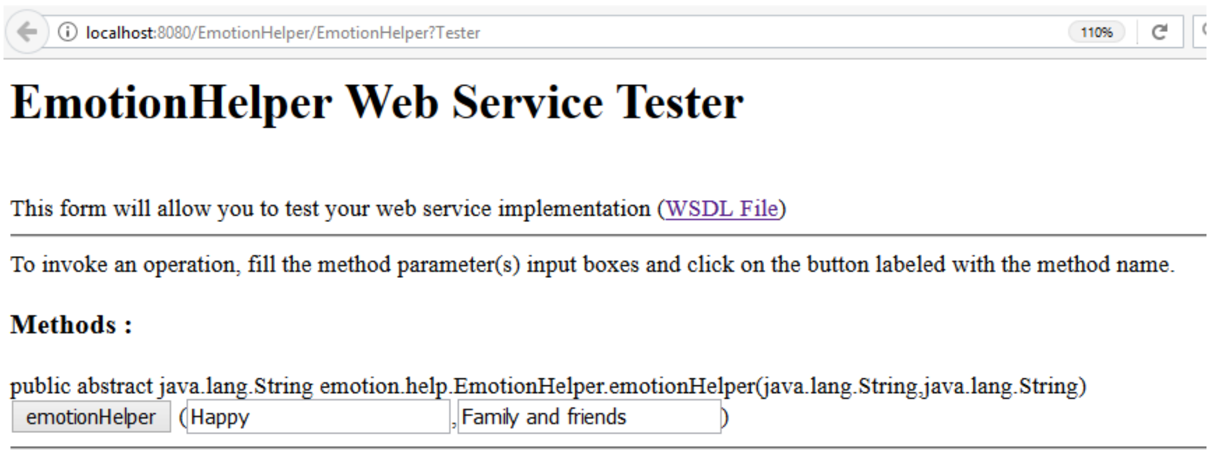


Figure 5.7 Testing a Web Service

9. Invoke the web service by specifying the parameters.

OUTPUT



localhost:8080/EmotionHelper/EmotionHelper?Tester

EmotionHelper Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

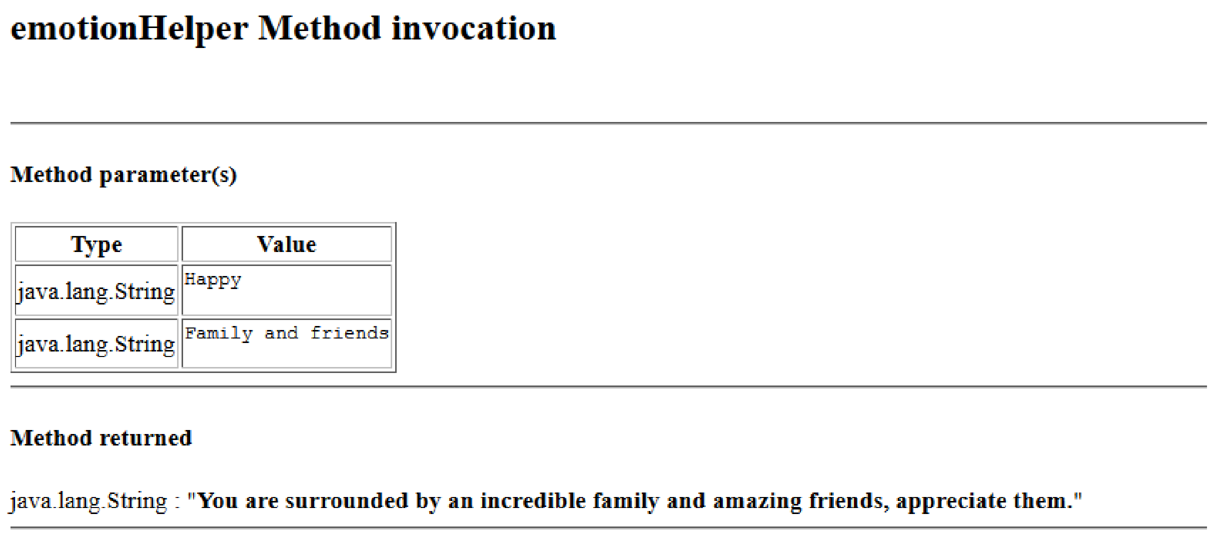
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

public abstract java.lang.String emotion.help.EmotionHelper.emotionHelper(java.lang.String,java.lang.String)

emotionHelper (Happy , Family and friends)

Figure 5.8 Web Service Tester



emotionHelper Method invocation

Method parameter(s)

Type	Value
java.lang.String	Happy
java.lang.String	Family and friends

Method returned

java.lang.String : "**You are surrounded by an incredible family and amazing friends, appreciate them.**"

Figure 5.9 Invocation of emotionHelper operation

RESULT

Emotion Helper Web Service has been successfully created.

Ex. No. : 6 Date : 09-08-2017

Title : CONSUMING A JAVA-WS USING JAVA CLASS, SERVLET AND JSP

AIM

To consume a Web Service using Java Class, Servlet and JSP

PROCEDURE

Steps to consume a Web Service using Java Client

1. Open Netbeans. File -> New Project -> Java -> Java Application

2. Name the project “EmotionClientJava”.

3. Right Click on Project Node in Projects pane -> New -> Web Service Client.

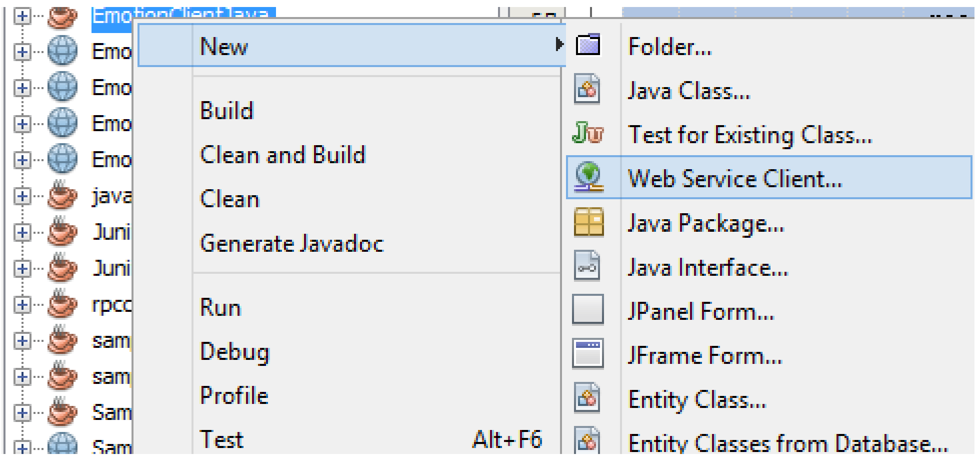


Figure 6.1 Creating a new Web Service Client

4. Browse and select Emotion Helper Web Service -> OK -> Finish.

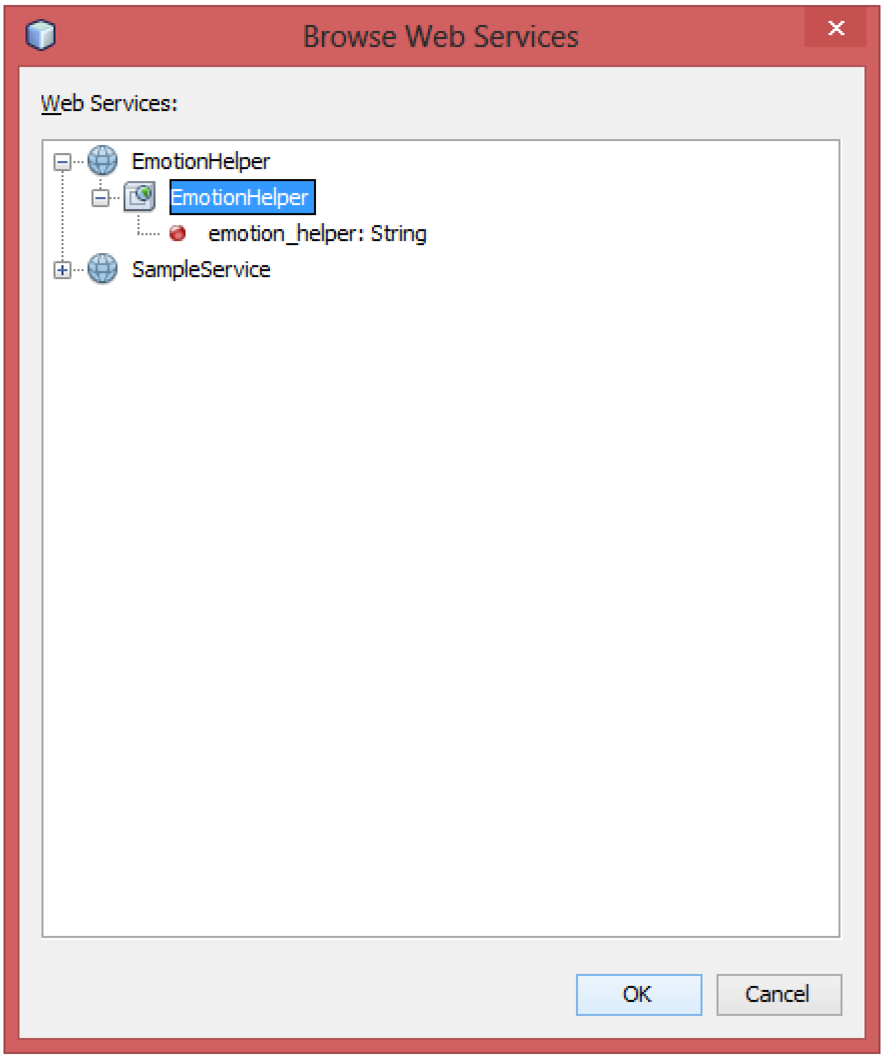
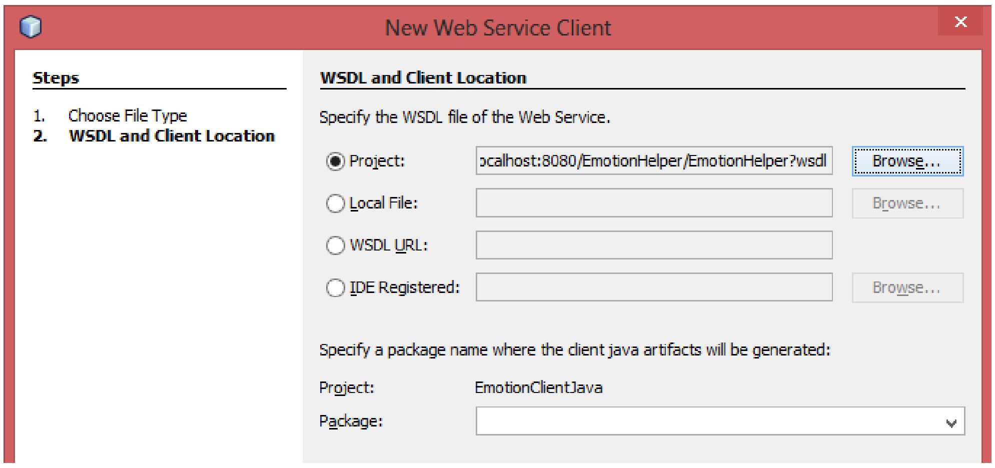


Figure 6.2 Browsing Web Service to add Service Reference



Specify the WSDL file of the Web Service.

(● Project:) ocalhost:8080/EmotionHelper/EmotionHelper?wsdl

Local File:

WSDL URL:

IDE Registered:

Specify a package name where the client java artifacts will be generated:

Project: EmotionClientJava

Package:

Figure 6.3 Selecting the location of Service

5. Create a User Interface for the user to select emotion and reason using Java Swing.

6. Under ‘Web Services References’ in the client application, drag the emotion_helper operation to the Java code to invoke the service by passing emotion and reason as parameters.

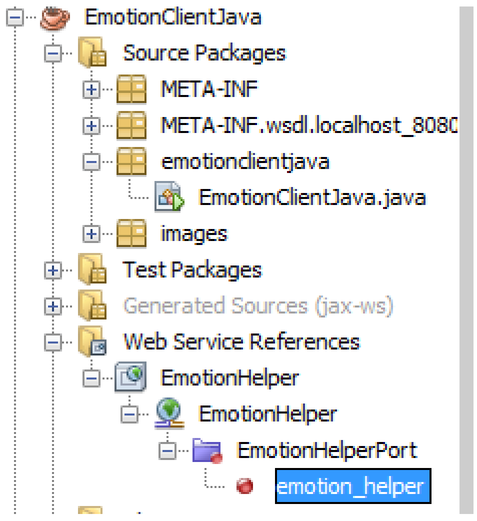


Figure 6.4 Web Service References

```
private static String emotionHelper(java.lang.String emotion,
java.lang.String reason) {
        emotion.help.EmotionHelper_Service service = new
emotion.help.EmotionHelper_Service();
        emotion.help.EmotionHelper port =
service.getEmotionHelperPort();
        return port.emotionHelper(emotion, reason);
    }
```

Code 6.1 Service invocation in client

7. Value returned from service (response) is displayed. Run the client program.

Steps to consume Web Service using Servlet

1. Open Netbeans. File -> New Project -> Java Web -> Java Web Application.

2. Name the project “EmotionClientServlet”.

3. Right Click on Project Node in Project pane -> New -> Web Service Client.

4. Browse and select Emotion Helper Web Service -> OK -> Finish.

5. Right click on the Project Node and select New -> Servlet.

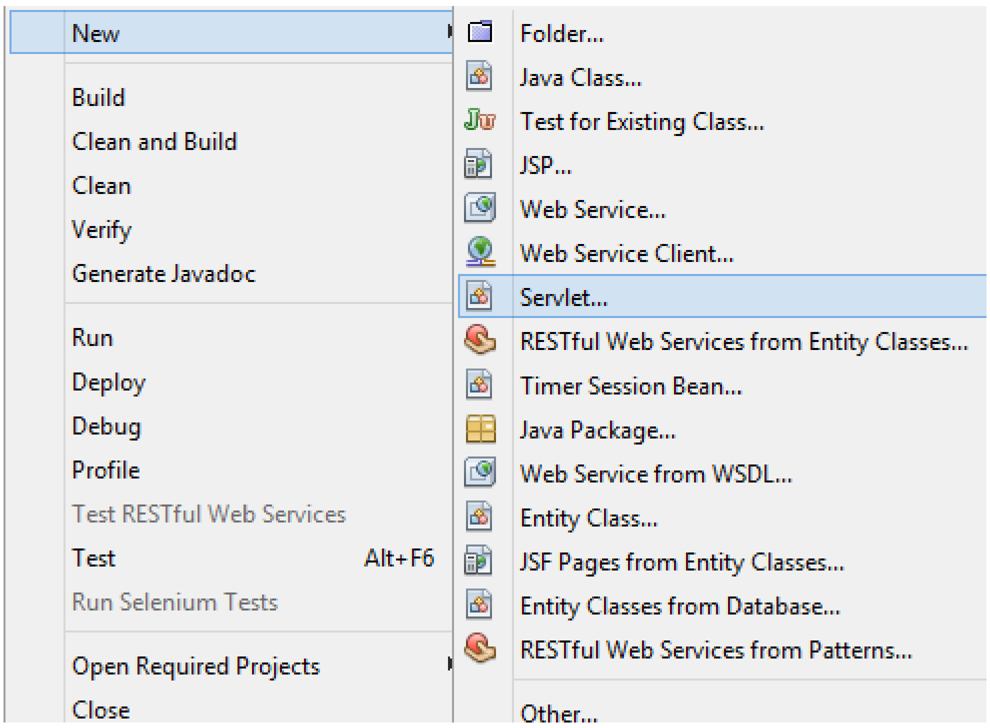


Figure 6.5 Adding a new servlet

6. Drag the emotion_helper operation (listed under Web Services References) to the Java code in the servlet to invoke the service by passing emotion and reason as parameters.

7. Right click on the Project Node and select New -> JSP, to create a UI using Drop-downs and Image Buttons in a form for the user to select the emotion and reason. Specify the name of the servlet in the form action.

```
<form action="emotionservlet" name="emoform" method="post">
```

Code 6.2 Form action in JSP

8. If there are any errors in the servlet, right click on the project node and, “Clean and Build”.

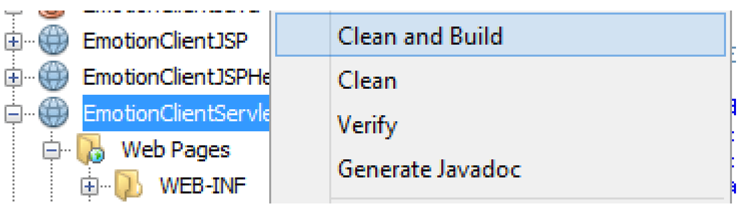


Figure 6.6 Clean and Build

9. Add code that calls method invoking the service. Print the response returned by the Web Service. Run the client program.

```
String result = emotionHelper(emotion,reason);
out.println("Message for you:<br><br>"+ result);
```

Code 6.3 Invoking the Web Service in the client

Steps to consume Web Service using JSP

1. Open Netbeans. File -> New Project -> Java Web -> Java Web Application.

2. Name the project “EmotionClientJSP”.

3. Right Click on Project Node in Project pane -> New -> Web Service Client.

4. Browse and select Emotion Helper Web Service -> OK -> Finish.

5. Right click on the Project Node and select New -> JSP, to create a UI using Drop-downs and Image Buttons in a form for the user to select the emotion and reason. Specify form action.

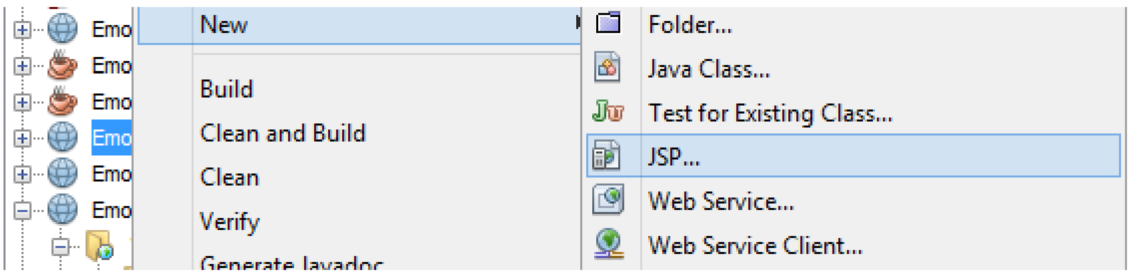


Figure 6.7 Adding a new JSP

```
<form action="respons.jsp" name="emoform" method="post">
```

```
  What is your emotion right now? (Select a reason in the dropdown
list above corresponding emoji and click on the appropriate emoji!)

<br><br>
            &nbsp; &nbsp; &nbsp; &nbsp;
            <select name="AngrySadDropdown">
                <option>Friend</option>
                <option>Family</option>
                <option>Unfulfilled expectation</option>
                <option>Unfair treatment</option>
                <option>Unexpected situation</option>
                <option>Loss of a loved one</option>
            </select>
            &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp; &nbsp;
&nbsp; &nbsp;
             <select name="ScaredStressedDropdown">
                <option>Fear of loss of loved one</option>
                <option>Family</option>
                <option>Work Pressure</option>
                <option>Exam or Placement Stress</option>
                <option>Commitments</option>
                <option>Sickness and health</option>
            </select>
            &nbsp; &nbsp; &nbsp; &nbsp;&nbsp; &nbsp; &nbsp; &nbsp;
&nbsp;

            <select name="HappyEnthuDropdown">
                <option>Achievement</option>
                <option>Gifts given or received</option>
                <option>Charity and good deeds</option>
                <option>Vacation, events, parties</option>
                <option>Family and friends</option>
                <option>Good food</option>
            </select>

            <br><br><br><br>
            <input type="submit" id="Angry" name="Angry"
style="background-image: url('\images/Angry.jpg'); border: solid 0px
#000000; width: 102px; height: 102px;" value=" " />
            <input type="submit" id="Sad" name="Sad"
style="background-image: url('\images/Sad.jpeg'); border: solid 0px
#000000; width: 102px; height: 102px;" value=" "/>
            &nbsp; &nbsp; &nbsp; &nbsp;

            <input type="submit" id="Scared" name="Scared"
style="background-image: url('\images/Scared.jpg'); border: solid
0px #000000; width: 98px; height: 98px;" value=" "/>
            <input type="submit" id="Stressed" name="Stressed"
style="background-image: url('\images/stressed.jpeg'); border: solid
0px #000000; width: 101px; height: 101px;" value=" "/>
            &nbsp; &nbsp; &nbsp; &nbsp;
            <input type="submit" id="Happy" name="Happy"
style="background-image: url('\images/Happy.jpg'); border: solid 0px
#000000; width: 93px; height: 92px;" value=""/>
            <input type="submit" id="Enthusiastic"
name="Enthusiastic" style="background-image:
url('\images/Enthu.jpeg'); border: solid 0px #000000; width: 106px;
height: 89px;" value=" "/>
        </form>
```

Code 6.4 JSP code for UI

6. Drag the emotion_helper operation (listed under Web Services References) to the JSP below the heading tags to invoke the service by passing emotion and reason as parameters.

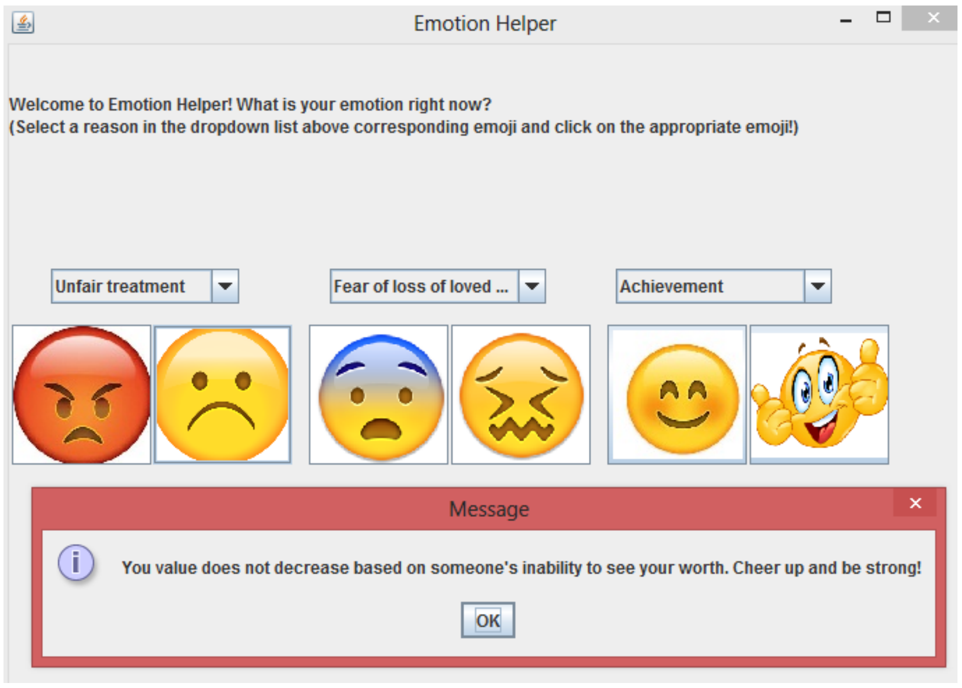
```
<%  String emotion = "", reason = "";
            if (request.getParameter("Angry") != null){
                emotion = "Angry";
                reason = request.getParameter("AngrySadDropdown");
            }
            else if (request.getParameter("Sad") != null){
                emotion = "Sad";
                reason = request.getParameter("AngrySadDropdown");
            }
            else if (request.getParameter("Scared") != null){
                emotion = "Scared";
                reason =
request.getParameter("ScaredStressedDropdown");
            }
            else if (request.getParameter("Stressed") != null){
                emotion = "Stressed";
                reason =
request.getParameter("ScaredStressedDropdown");
            }
            else if (request.getParameter("Happy") != null){
                emotion = "Happy";
                reason = request.getParameter("HappyEnthuDropdown");
            }
            else if (request.getParameter("Enthusiastic") != null){
                emotion = "Enthusiastic";
                reason = request.getParameter("HappyEnthuDropdown");
            }



          try {
            emotion.help.EmotionHelper_Service service = new
      emotion.help.EmotionHelper_Service();
            emotion.help.EmotionHelper port =
      service.getEmotionHelperPort();
```

```
        java.lang.String result = port.emotionHelper(emotion, reason);
        out.println("Message for you: "+ result);
    } catch (Exception ex) {

    }
```

Code 6.5 JSP Code to invoke the service

7. Store and display the response returned by the Web Service. Run the client project.

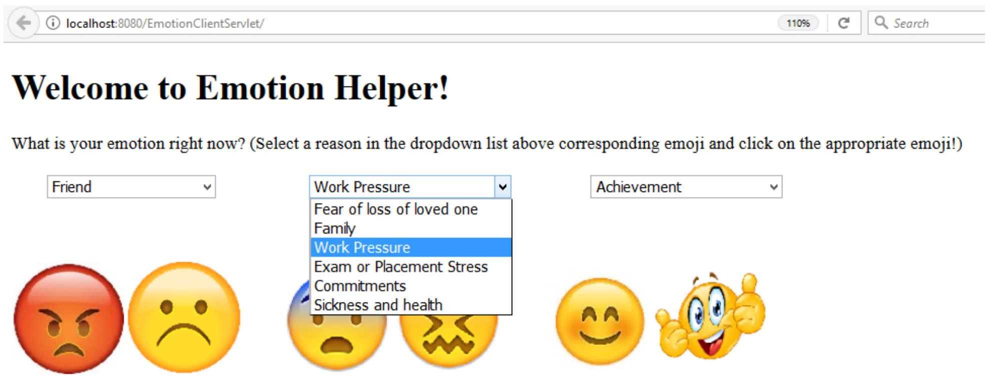
OUTPUT



Welcome to Emotion Helper! What is your emotion right now?
(Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

You value does not decrease based on someone's inability to see your worth. Cheer up and be strong!

Figure 6.8 Consuming Web Service in Java


localhost:8080/EmotionClientServlet/

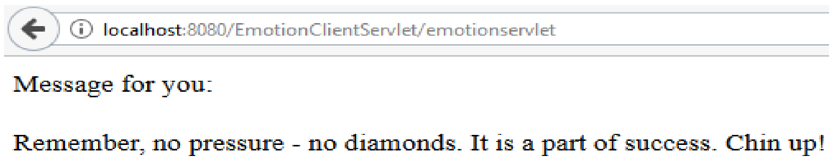
Welcome to Emotion Helper!

What is your emotion right now? (Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

Friend | Work Pressure | Achievement

Fear of loss of loved one
Family
Work Pressure
Exam or Placement Stress
Commitments
Sickness and health

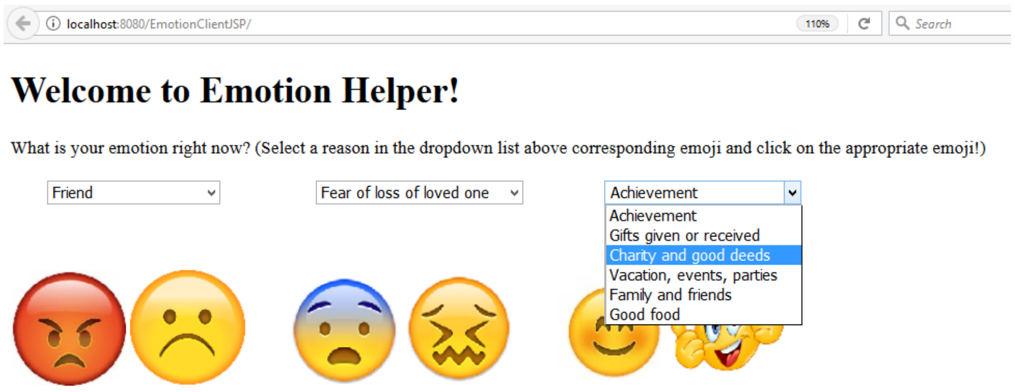
Figure 6.9 Consuming Web Service in Servlet – Page 1


localhost:8080/EmotionClientServlet/emotionservlet

Message for you:

Remember, no pressure - no diamonds. It is a part of success. Chin up!

Figure 6.10 Consuming Web Service in Servlet – Page 2


localhost:8080/EmotionClientJSP/

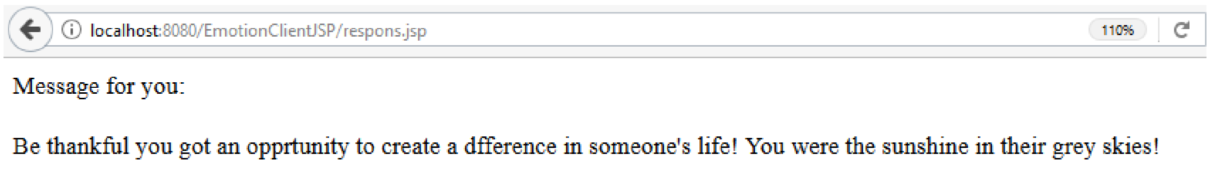
Welcome to Emotion Helper!

What is your emotion right now? (Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

Friend | Fear of loss of loved one | Achievement

Achievement
Gifts given or received
Charity and good deeds
Vacation, events, parties
Family and friends
Good food

Figure 6.11 Consuming Web Service in JSP – Page 1


localhost:8080/EmotionClientJSP/respons.jsp

Message for you:

Be thankful you got an opprtunity to create a dfference in someone's life! You were the sunshine in their grey skies!

Figure 6.12 Consuming Web Service in JSP – Page 2

RESULT

The Web Service has been consumed using Java class, servlet and JSP.

Ex. No. : 7 Date : 16-08-2017

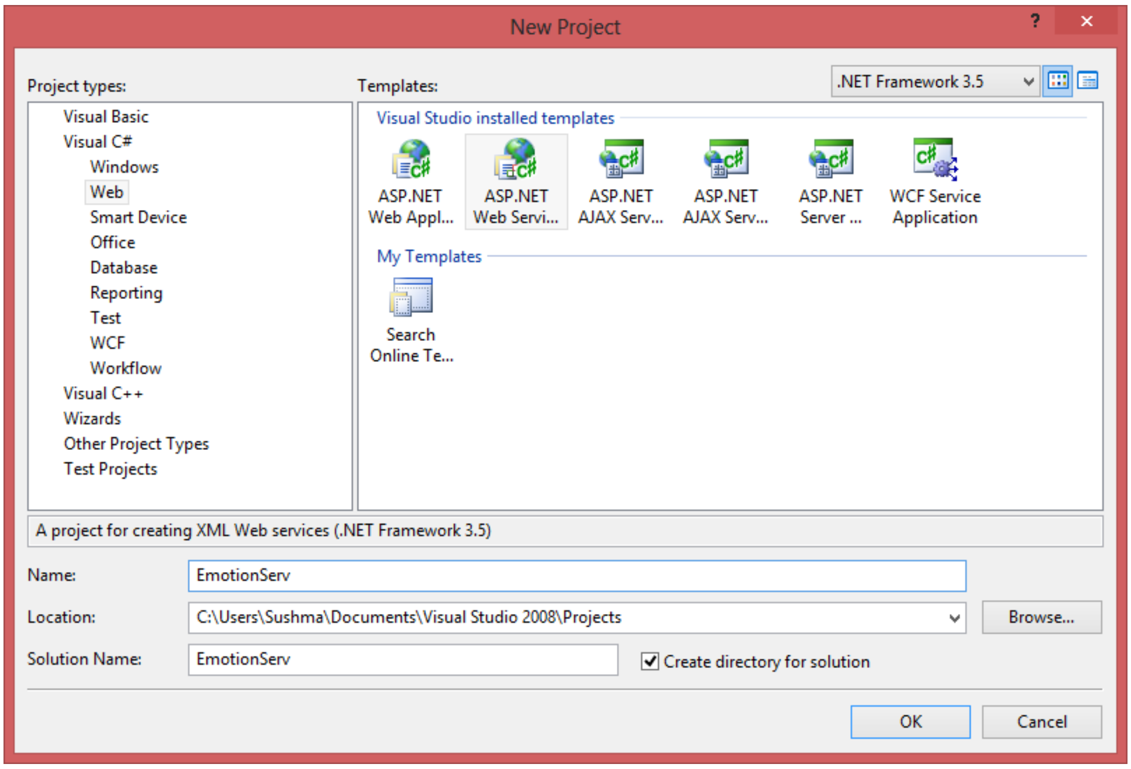
Title : CREATING A WEB SERVICE USING .NET

AIM

To create a Web Service in .NET

PROCEDURE

1. Open Visual Studio. Create a new Project. Choose Web under Visual C# and select ASP .NET Web Service Application. Name the project ‘EmotionServ’.



A project for creating XML Web services (.NET Framework 3.5)

Name: EmotionServ

Location: C:\Users\Sushma\Documents\Visual Studio 2008\Projects

Solution Name: EmotionServ

☑ Create directory for solution

Figure 7.1 Creating a Web Service in .NET using Visual Studio

2. Modify the Web Method created in EmotionServ.asmx.cs, such that it takes two strings emotion and reason as input and returns a string response as output. Name it EmotionHelper().

3. Add code to parse XML file and retrieve appropriate response for the emotion and reason received as parameters.

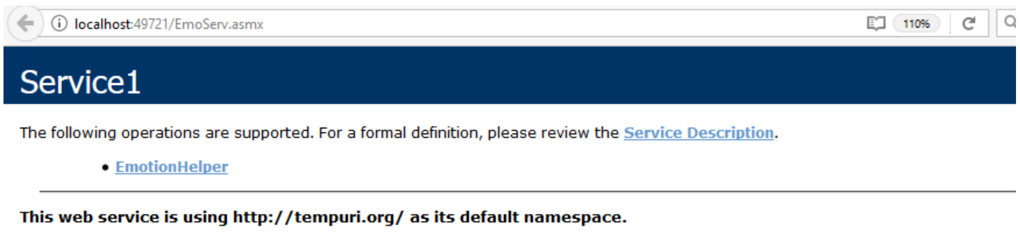
```
[WebMethod]
        public string EmotionHelper(string emotion, string reason)
        {

            XmlDocument xmldoc = new XmlDocument();
            String emotion1 = "", response = "";
            xmldoc.Load("emohelp.xml");
           XmlNodeList emohelps =
      xmldoc.DocumentElement.SelectNodes("/emotion_helper/emo_help");
            foreach (XmlNode emohelp in emohelps)
            {
                emotion1 = emohelp.FirstChild.InnerText;
                if (emotion1.Equals(emotion))
                {
                    foreach (XmlNode responses in emohelp)
                    {

                        if (responses.Name.Equals("response") &&
responses.Attributes["reason"].Value.Equals(reason))
                        {
                            response = responses.InnerText; break;
                        }
                    }
                }
            }

            return (response);
        }}
```

Code 7.1 Web Service logic in C#

4. Return the response to the client.

5. Run the project to test the Web Service.

OUTPUT



localhost:49721/EmoServ.asmx

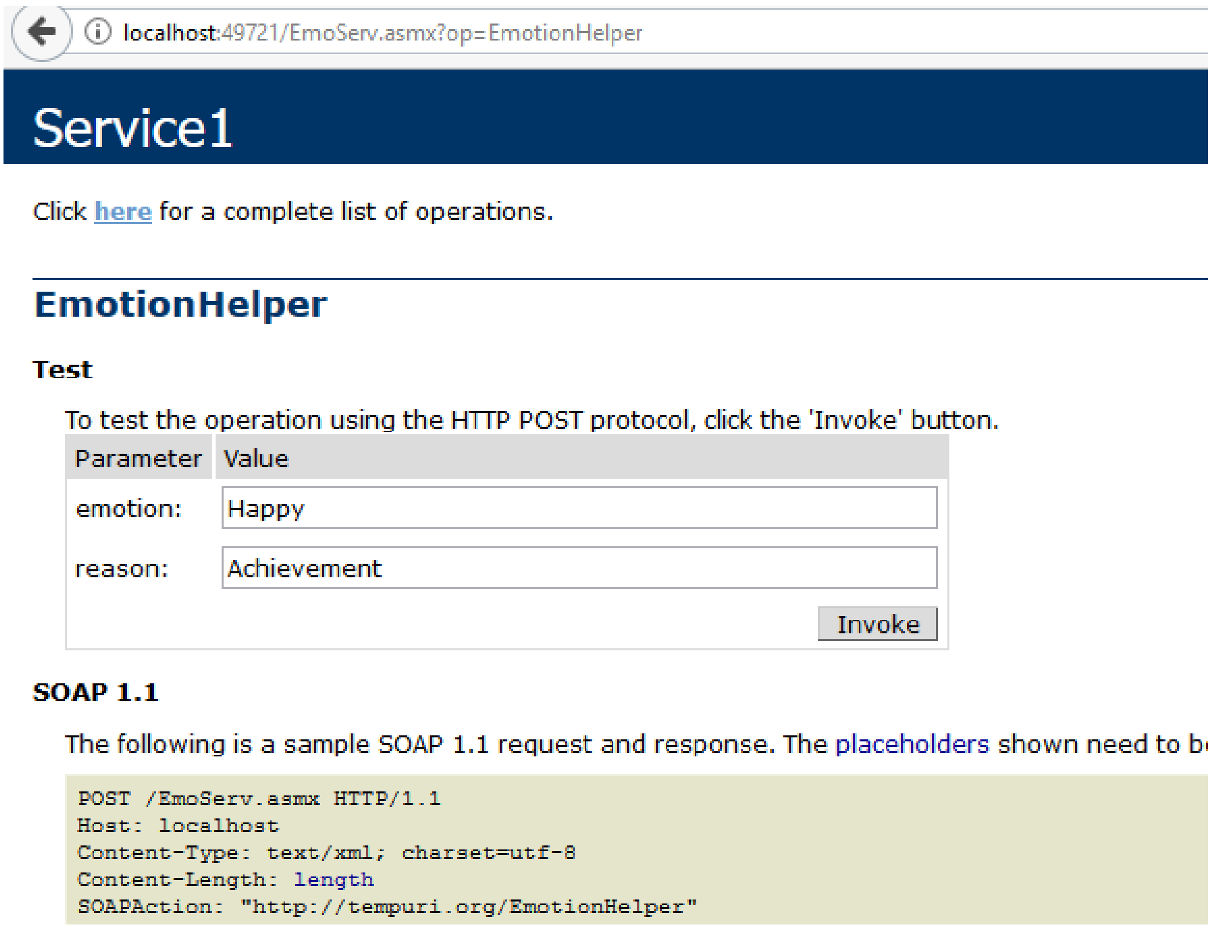
Service1

The following operations are supported. For a formal definition, please review the Service Description.

- EmotionHelper

This web service is using http://tempuri.org/ as its default namespace.

Figure 7.2 Web Service in .NET


localhost:49721/EmoServ.asmx?op=EmotionHelper

Service1

Click here for a complete list of operations.

EmotionHelper

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
emotion:	Happy
reason:	Achievement

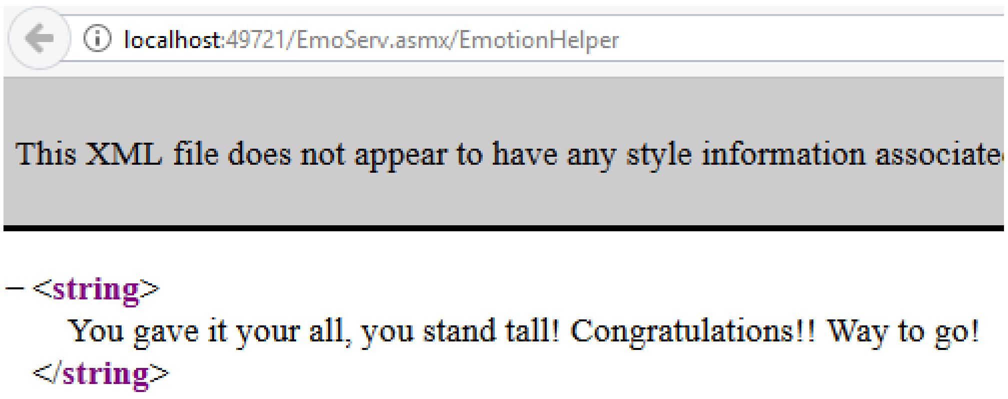
Invoke

SOAP 1.1

The following is a sample SOAP 1.1 request and response. The placeholders shown need to b

```
POST /EmoServ.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/EmotionHelper"
```

Figure 7.3 Web Service invocation in .NET


localhost:49721/EmoServ.asmx/EmotionHelper

This XML file does not appear to have any style information associate

```
– <string>
    You gave it your all, you stand tall! Congratulations!! Way to go!
  </string>
```

Figure 7.4 Web Service invocation result in .NET

RESULT

Web Service has been created in .NET using Visual Studio.

Ex. No. : 8 Date : 23-08-2017

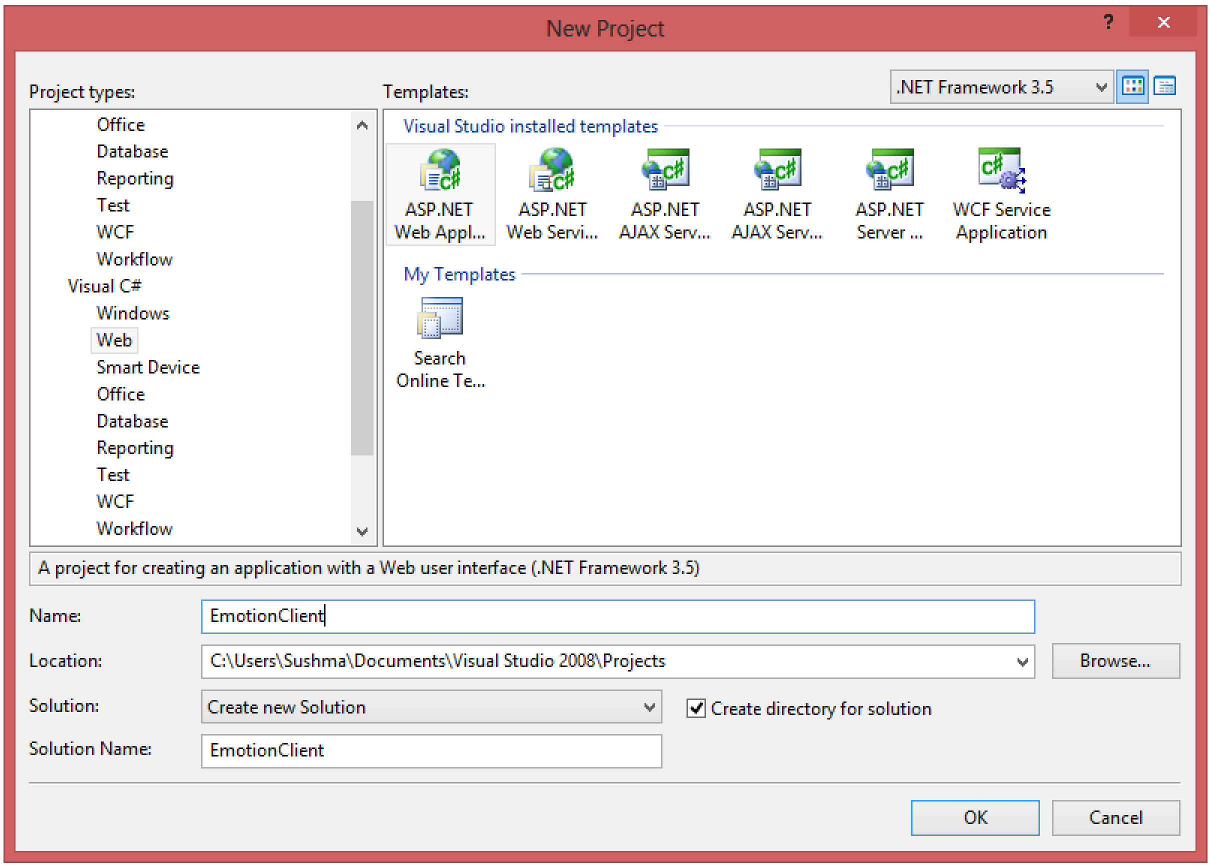
Title : CONSUMING A .NET-WS USING .NET

AIM

To consume a Web Service using .NET

PROCEDURE

1. Open Visual Studio. Create a new Project. Choose Web under Visual C# and select ASP .NET Web Application. Name the project ‘EmotionServ’.



A project for creating an application with a Web user interface (.NET Framework 3.5)

Name: EmotionClient

Location: C:\Users\Sushma\Documents\Visual Studio 2008\Projects

Solution: Create new Solution ☑ Create directory for solution

Solution Name: EmotionClient

Figure 8.1 Creating a WS Client in .NET

2. Right click on the project in Solution Explorer and select ‘Add Web Reference’.

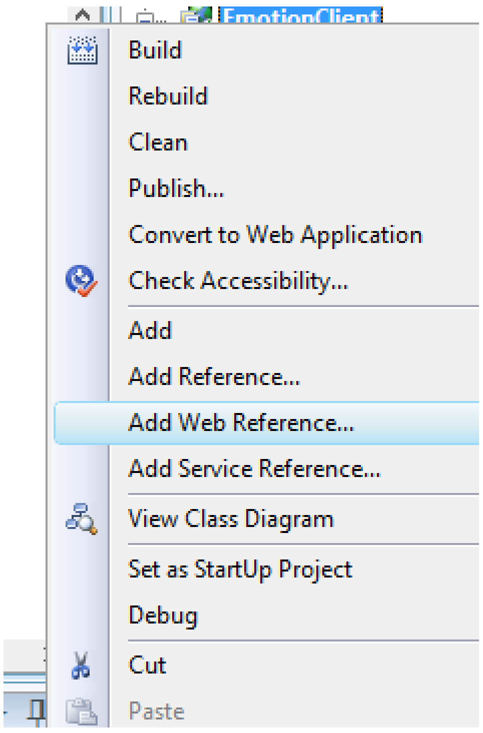
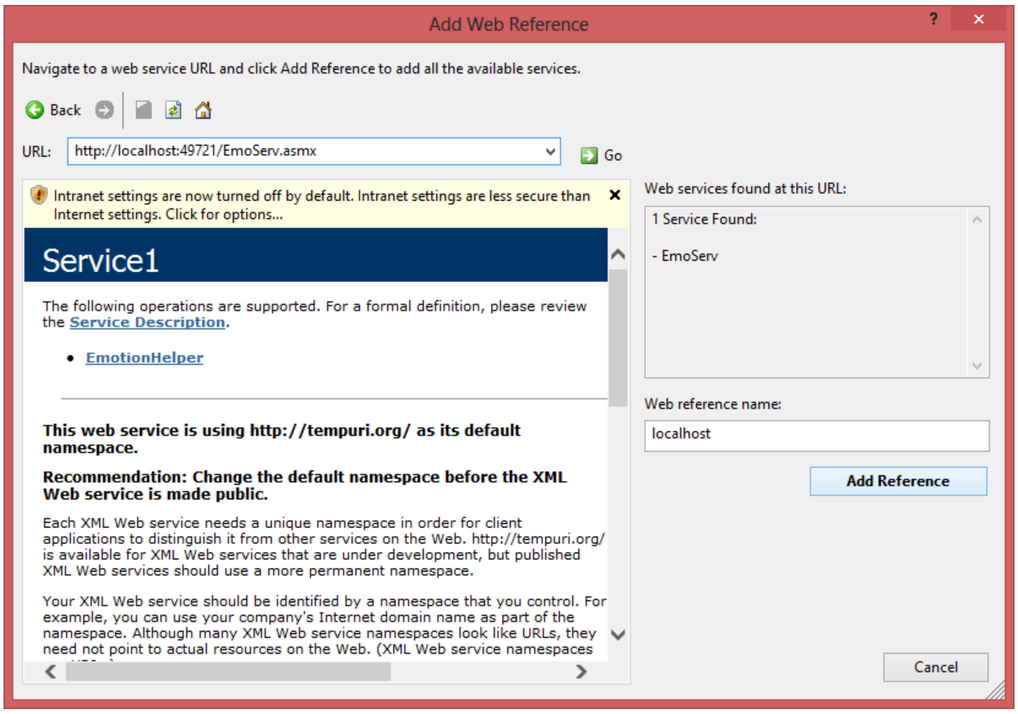


Figure 8.2 Adding a Web Reference .NET

3. Type the URL of the Web Service when run on the browser and select go. The Web Service will be displayed and its operations listed. Specify Web Reference name as ‘localhost’ and select ‘Add reference’.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:49721/EmoServ.asmx

Intranet settings are now turned off by default. Intranet settings are less secure than Internet settings. Click for options...

Service1

The following operations are supported. For a formal definition, please review the Service Description.

- EmotionHelper

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces

Web services found at this URL:

1 Service Found:

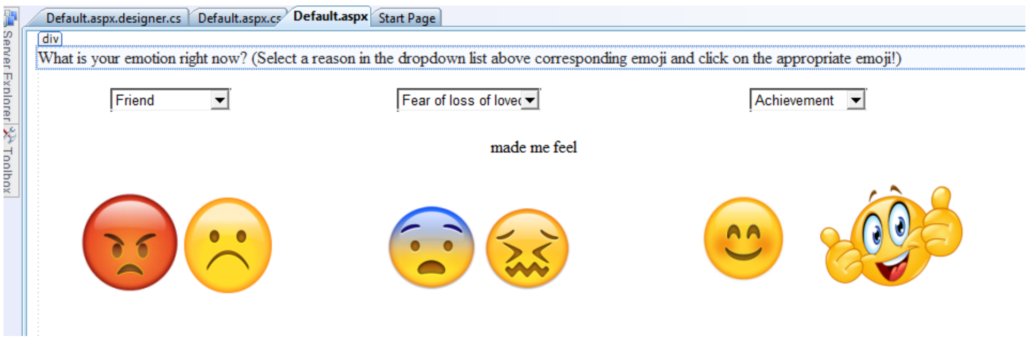
- EmoServ

Web reference name:

localhost

Figure 8.3 Web Reference in .NET

4. In the designer of default.aspx add image buttons for each emotion and dropdowns for selecting the reason.



What is your emotion right now? (Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

Friend | Fear of loss of loved | Achievement

made me feel

Figure 8.4 Designer in .NET

5. Add Click events as follows for each of the image buttons so that the method to invoke the Web Service will be called.

```
protected void SadEmoji_Click(object sender,
ImageClickEventArgs e)
   {
```

```
            emotion = "Sad";
            reason = DropDownList1.SelectedValue;
            emotionhelp(emotion, reason);
```

Code 8.1 Calling method to invoke WS

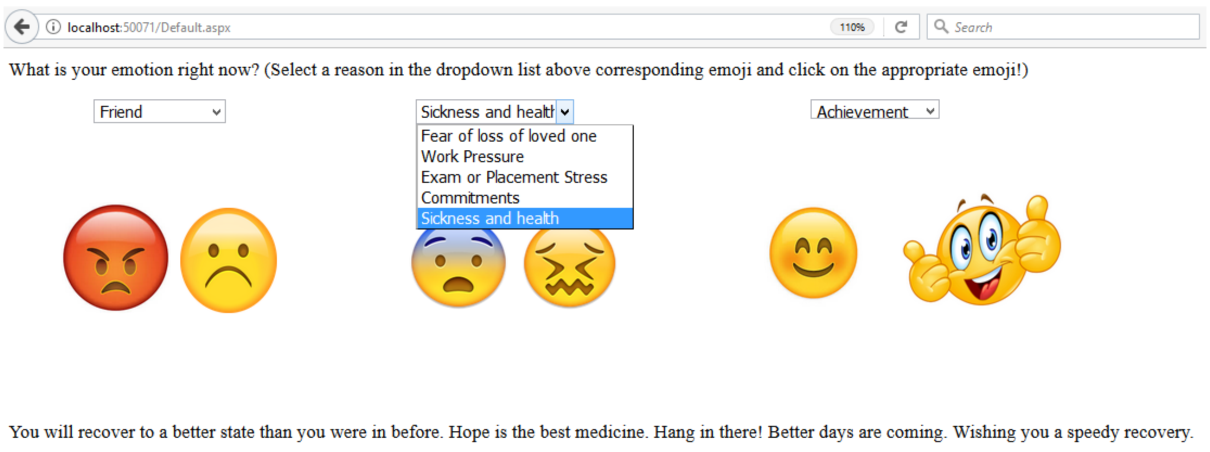
6. Create a method to invoke the Web Service and add the following code to it.

```
protected void emotionhelp(string emotion, string reason)
        {
            localhost.Service1 ws = new localhost.Service1();
            Label1.Text = ws.EmotionHelper(emotion, reason);

      }
```

Code 8.2 Invoking Web Service

7. Run the client project to consume the service.

OUTPUT



What is your emotion right now? (Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

You will recover to a better state than you were in before. Hope is the best medicine. Hang in there! Better days are coming. Wishing you a speedy recovery.

Figure 8.5 Web service consumption in .NET

RESULT

Web Service has been consumed using a .NET client.

Ex. No. : 9 Date : 30-08-2017

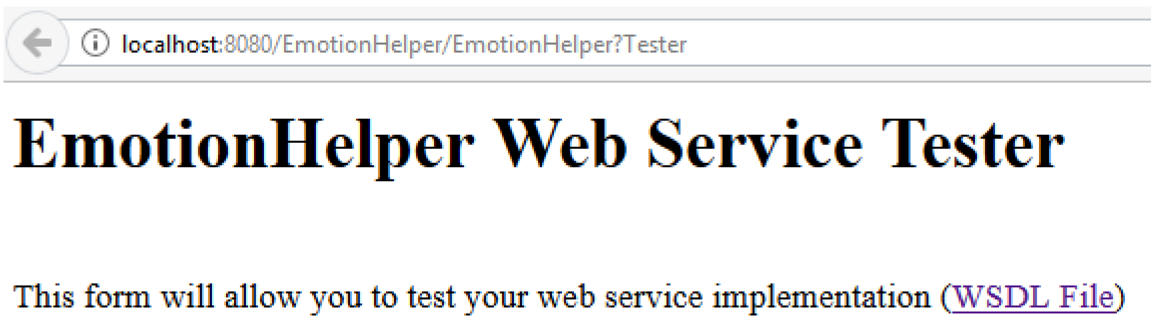
Title : HETEROGENEOUS WEB SERVICES (JAVA-WS WITH .NET CLIENT)

AIM

To create a Web Service in Java and consume it in .NET

PROCEDURE

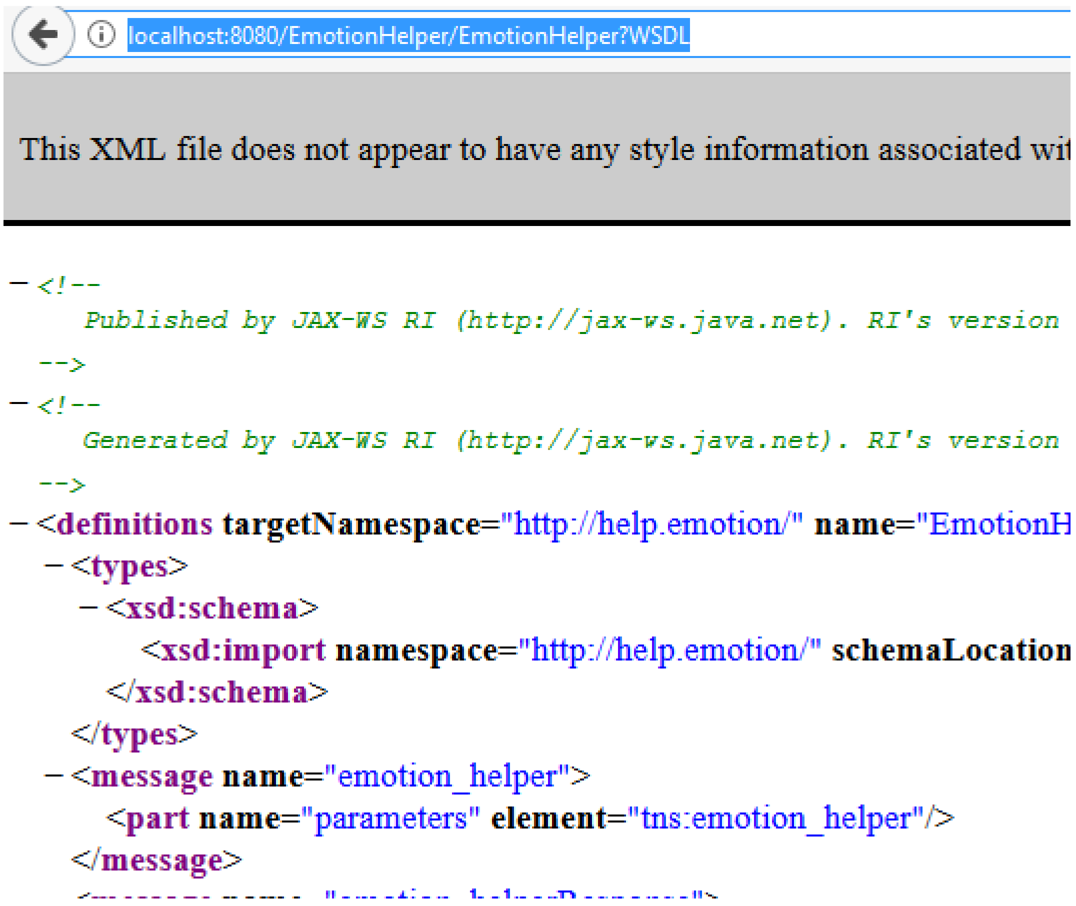
1. Create a Web Service in Java to implement Emotion Helper and copy the URL of its WSDL file.


localhost:8080/EmotionHelper/EmotionHelper?Tester

EmotionHelper Web Service Tester

This form will allow you to test your web service implementation (WSDL File)

Figure 9.1 Java Web Service Tester



localhost:8080/EmotionHelper/EmotionHelper?WSDL

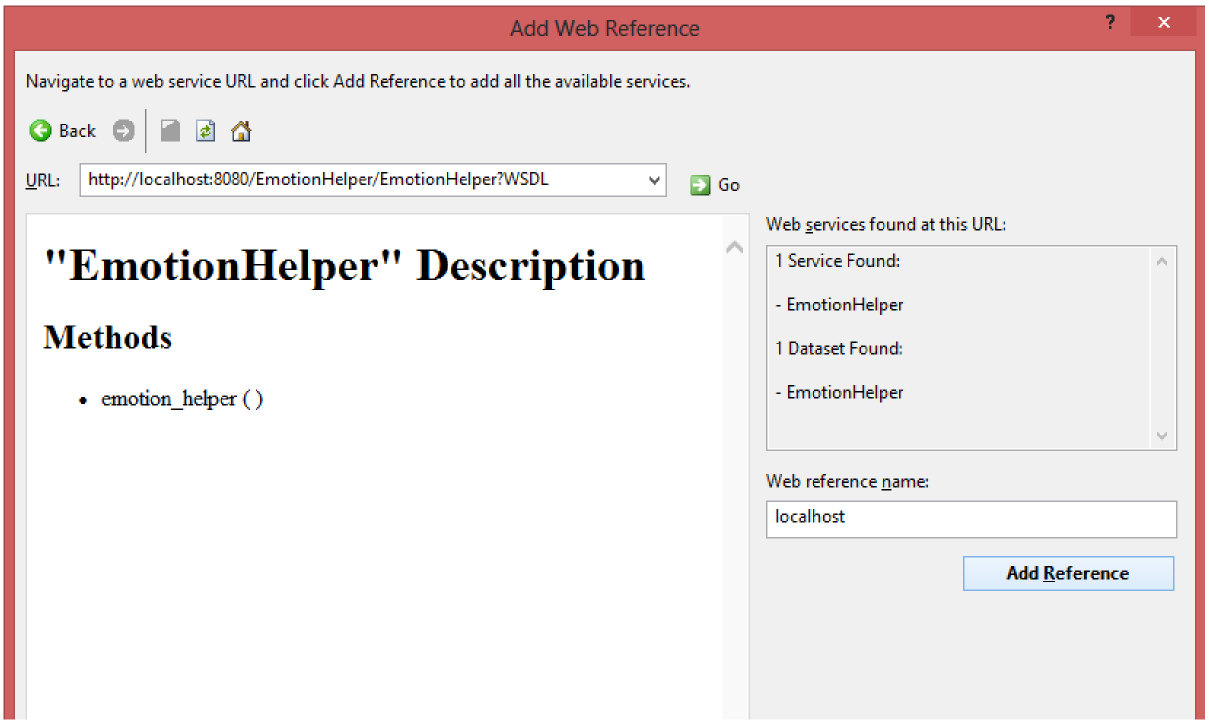
This XML file does not appear to have any style information associated wi

```
− <!--
    Published by JAX-WS RI (http://jax-ws.java.net). RI's version
  -->
− <!--
    Generated by JAX-WS RI (http://jax-ws.java.net). RI's version
  -->
− <definitions targetNamespace="http://help.emotion/" name="EmotionH
  − <types>
    − <xsd:schema>
        <xsd:import namespace="http://help.emotion/" schemaLocation
      </xsd:schema>
    </types>
  − <message name="emotion_helper">
      <part name="parameters" element="tns:emotion_helper"/>
    </message>
```

Figure 9.2 WSDL file of Web Service

2. Create a new ASP.NET Web Application Project (Visual C#) in Visual Studio and name it ‘EmotionCientHetero’.

3. Right click project to add a Web Reference. Paste the URL of the Java Web Service’s WSDL file and click on Go. Specify Web Reference name as ‘localhost’.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: http://localhost:8080/EmotionHelper/EmotionHelper?WSDL

"EmotionHelper" Description

Methods

- emotion_helper ()

Web services found at this URL:

1 Service Found:

- EmotionHelper

1 Dataset Found:

- EmotionHelper

Web reference name: localhost

Figure 9.3 Adding Web Reference in client

4. Create a User Interface with the designer and add Click events for all the image buttons.

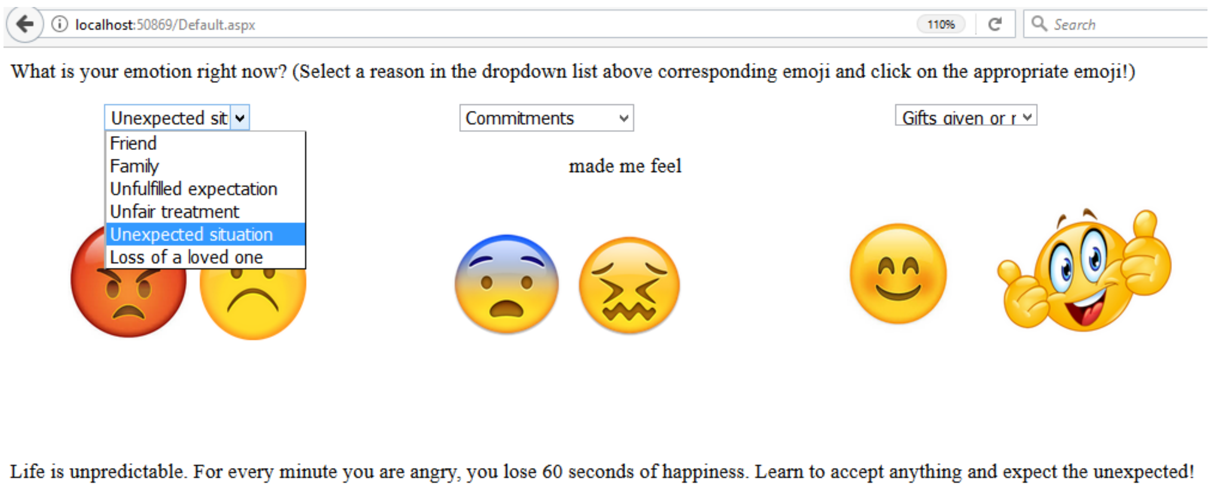
5. Create a method to invoke the Web Service. Add the following code to it.

```
protected void emotionhelp(string emotion, string reason)
        {
            localhost.EmotionHelper ws = new
localhost.EmotionHelper();
            Label1.Text = ws.emotion_helper(emotion, reason);
     }
```

Code 1.1 Invoking Web Service in C#

6. Run the client project to consume the Java Web Service heterogeneously in .NET.

OUTPUT


localhost:50869/Default.aspx

What is your emotion right now? (Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

made me feel

Life is unpredictable. For every minute you are angry, you lose 60 seconds of happiness. Learn to accept anything and expect the unexpected!

Figure 9.4 Consuming a Java WS in .NET

RESULT

Java Web Service has been successfully consumed in .NET

Ex. No. : 10 Date : 13-09-2017

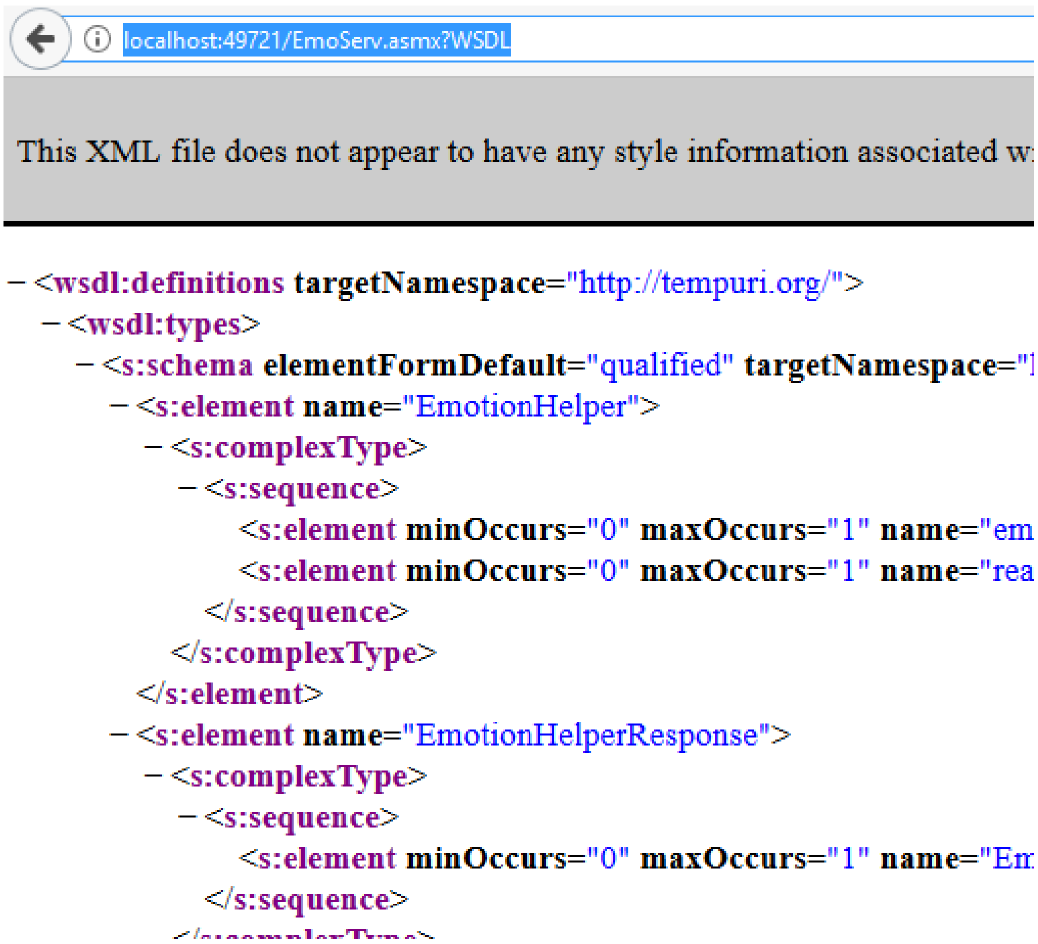
Title : HETEROGENEOUS WEB SERVICES (.NET-WS WITH JAVA CLIENT)

AIM

To create a web service in .NET and consume it in JAVA.

PROCEDURE

1. Create a Web Service in .NET to implement Emotion Helper and copy the URL of its WSDL file.



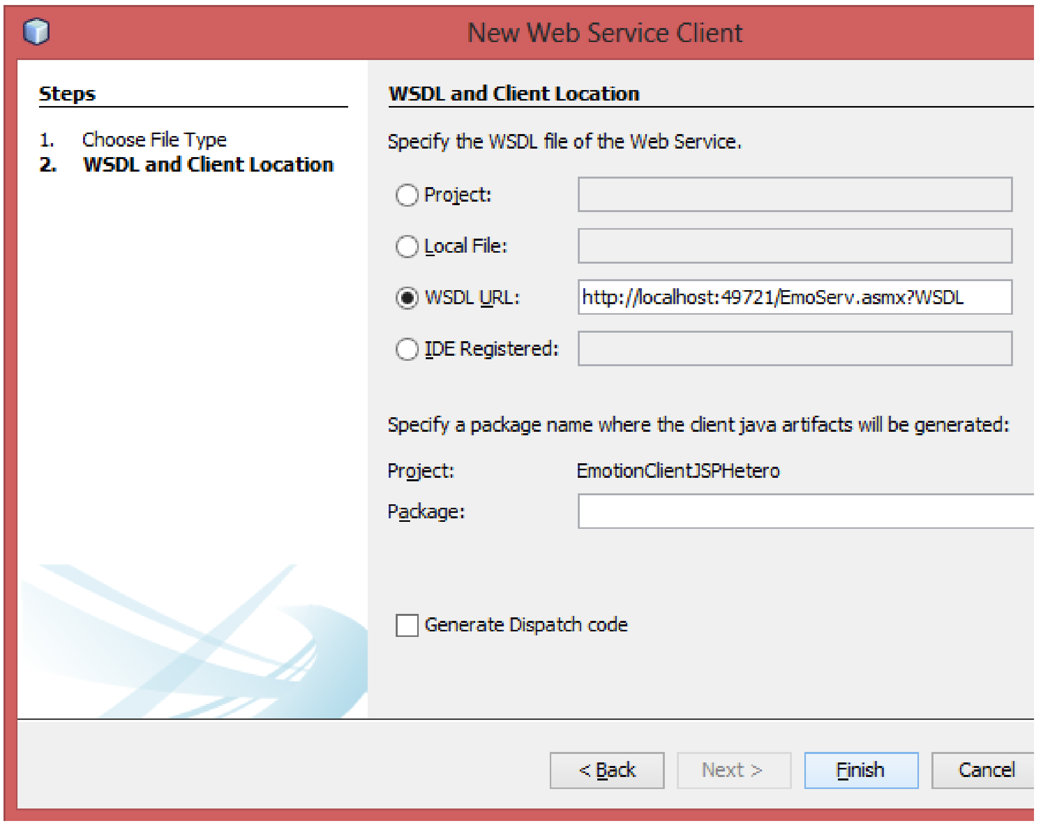
```
localhost:49721/EmoServ.asmx?WSDL

This XML file does not appear to have any style information associated w

– <wsdl:definitions targetNamespace="http://tempuri.org/">
  – <wsdl:types>
    – <s:schema elementFormDefault="qualified" targetNamespace="
      – <s:element name="EmotionHelper">
        – <s:complexType>
          – <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="em
              <s:element minOccurs="0" maxOccurs="1" name="rea
            </s:sequence>
          </s:complexType>
        </s:element>
      – <s:element name="EmotionHelperResponse">
        – <s:complexType>
          – <s:sequence>
              <s:element minOccurs="0" maxOccurs="1" name="Em
            </s:sequence>
```

Figure 10.1 WSDL File of Web Service

2. Create a new Web Application Project in Java to consume the Web Service created and name it ‘EmotionClientJSPHetero’.

3. Right click on the Project Node and select Add Web Service Client.

4. Paste the WSDL file URL and click on finish.



Specify the WSDL file of the Web Service.

Project:

Local File:

WSDL URL: http://localhost:49721/EmoServ.asmx?WSDL

IDE Registered:

Specify a package name where the client java artifacts will be generated:

Project: EmotionClientJSPHetero

Package:

☐ Generate Dispatch code

Figure 10.2 Adding Web Service Reference in Java

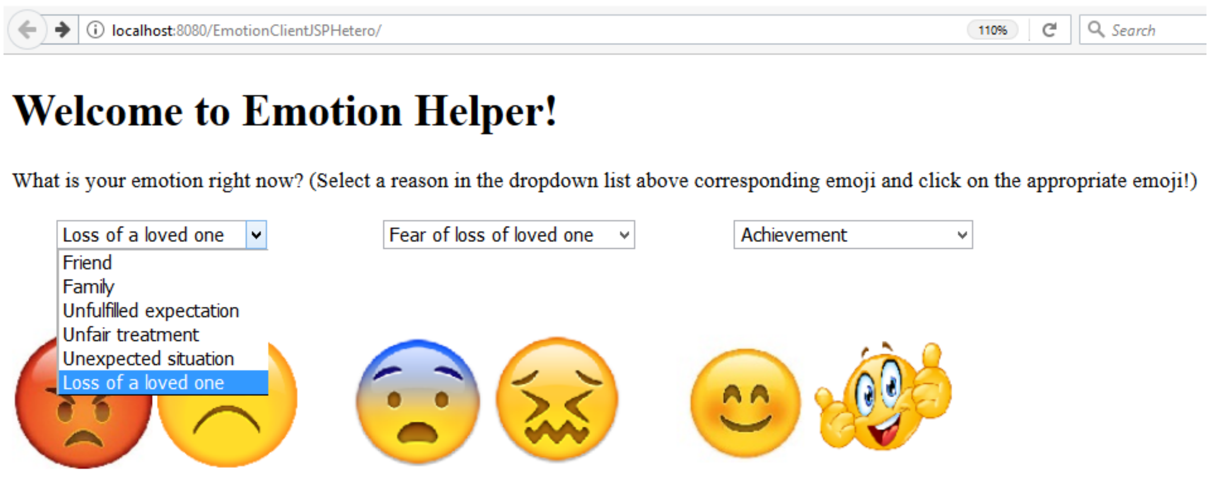
5. Create a User Interface to receive emotion and reason as input from the user.

6. Invoke the Web Service using JSP by dragging the operation listed under Web Service References in Project pane.

7. Display the value returned by the Web Service.

8. Run the client project to consume the .NET Web Service heterogeneously in Java.

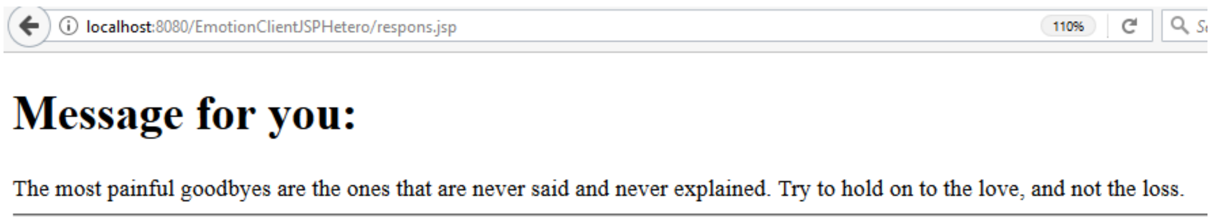
OUTPUT


localhost:8080/EmotionClientJSPHetero/

Welcome to Emotion Helper!

What is your emotion right now? (Select a reason in the dropdown list above corresponding emoji and click on the appropriate emoji!)

Figure 10.3 Heterogeneous JSP Client – Page 1


localhost:8080/EmotionClientJSPHetero/respons.jsp

Message for you:

The most painful goodbyes are the ones that are never said and never explained. Try to hold on to the love, and not the loss.

Figure 10.4 Heterogeneous JSP Client – Page 2

RESULT

.NET Web Service has been consumed in Java using a JSP Client.

Ex. No. : 11   Date : 20-09-2017

Title : XML ENCRYPTION

AIM

To encrypt data present in an XML file.

PROCEDURE

1. Create a Java Application Project and name it ‘EmotionEncryption’.
2. Create a new method called encrypt that takes in a string as input, encrypts using an encryption algorithm and returns the encrypted word. Add the following code

```
public static String encrypt(String word)
    {
        String encword = "";
        for(int i = 0;i<word.length();i++)
        {
            char ch = word.charAt(i);
            ch = (char) (ch + 5);
            encword += ch;
        }


        return encword;
    }
```

Code 11.1 Method to encrypt a word

3. Create a method to parse the XML file, find the required tag and replace it with the encrypted data. This method invokes the encrypt() method.

```
public static void parser(String inputfile)
        {String enc = "";
           try
           {

           File input = new File(inputfile);
               SAXBuilder saxbuilder = new SAXBuilder();
               Document doc = saxbuilder.build(input);
```

```
             Element rootelt = doc.getRootElement();
             System.out.println(rootelt.getName());

            List<Element> listofelts = rootelt.getChildren();

            for (int i=0;i<listofelts.size();i++)
    {

        Element chld = listofelts.get(i);

        enc = encrypt(chld.getChild("emotion").getText());
                chld.getChild("emotion").setText(enc);

    }
            XMLOutputter xmloutput = new XMLOutputter();
            xmloutput.setFormat(Format.getPrettyFormat());
            xmloutput.output(doc, new FileWriter(inputfile));
        }
        catch(Exception e)
        {
            System.out.println(e);
        }

     }
```

Code 11.2 Method to parse and modify XML file

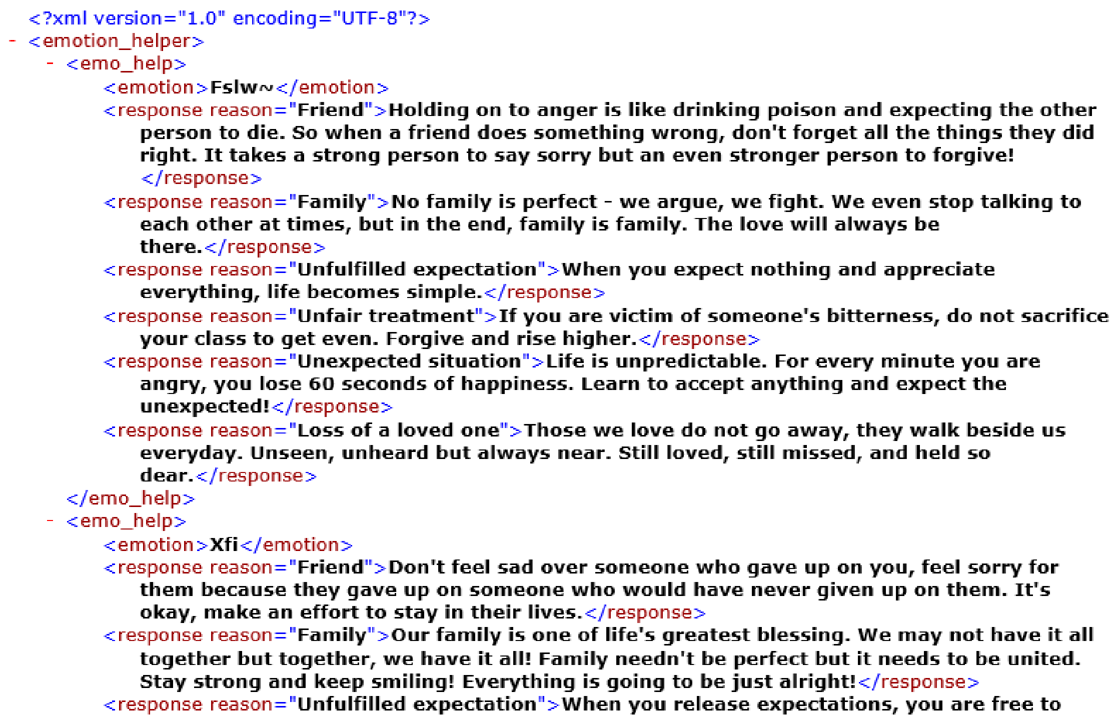
4. Invoke the parser method in the main function as follows

```
parser("emohelp.xml");
```

Code 11.2 Invoking the parser method

5. Run the project and open the XML file. It will contain encrypted values in the data in emotion tag.

OUTPUT



```
  <?xml version="1.0" encoding="UTF-8"?>
- <emotion_helper>
  - <emo_help>
        <emotion>Fslw~</emotion>
        <response reason="Friend">Holding on to anger is like drinking poison and expecting the other
            person to die. So when a friend does something wrong, don't forget all the things they did
            right. It takes a strong person to say sorry but an even stronger person to forgive!
            </response>
        <response reason="Family">No family is perfect - we argue, we fight. We even stop talking to
            each other at times, but in the end, family is family. The love will always be
            there.</response>
        <response reason="Unfulfilled expectation">When you expect nothing and appreciate
            everything, life becomes simple.</response>
        <response reason="Unfair treatment">If you are victim of someone's bitterness, do not sacrifice
            your class to get even. Forgive and rise higher.</response>
        <response reason="Unexpected situation">Life is unpredictable. For every minute you are
            angry, you lose 60 seconds of happiness. Learn to accept anything and expect the
            unexpected!</response>
        <response reason="Loss of a loved one">Those we love do not go away, they walk beside us
            everyday. Unseen, unheard but always near. Still loved, still missed, and held so
            dear.</response>
    </emo_help>
  - <emo_help>
        <emotion>Xfi</emotion>
        <response reason="Friend">Don't feel sad over someone who gave up on you, feel sorry for
            them because they gave up on someone who would have never given up on them. It's
            okay, make an effort to stay in their lives.</response>
        <response reason="Family">Our family is one of life's greatest blessing. We may not have it all
            together but together, we have it all! Family needn't be perfect but it needs to be united.
            Stay strong and keep smiling! Everything is going to be just alright!</response>
        <response reason="Unfulfilled expectation">When you release expectations, you are free to
```

Figure 11.1 XML file after encrypting data in emotion tag

RESULT

Data in XML file has been encrypted and the XML file has been updated with the encrypted value.

Ex. No. : 12   Date : 27-09-2017

Title : XML DECRYPTION

AIM

To decrypt data present in an XML file.

PROCEDURE

1. Create a Java Application Project and name it ‘EmotionDecryption’.

2. Create a new method called decrypt that takes in a string as input, decrypts using the reverse of the algorithm it used to encrypt the word and returns the decrypted word. Add the following code.

```
public static String decrypt(String word)
    {
        String decword = "";
        for(int i = 0;i<word.length();i++)
        {
            char ch = word.charAt(i);
            ch = (char) (ch - 5);
            decword += ch;
        }


        return decword;
    }
```

Code 12.1 Method to decrypt a word

3. Create a method to parse the XML file, find the required tag and replace it with the decrypted data. This method invokes the decrypt() method.

```
public static void parser(String inputfile)
        {String dec = "";
           try
           {

           File input = new File(inputfile);
               SAXBuilder saxbuilder = new SAXBuilder();
               Document doc = saxbuilder.build(input);
```

```
             Element rootelt = doc.getRootElement();
             System.out.println(rootelt.getName());

            List<Element> listofelts = rootelt.getChildren();

            for (int i=0;i<listofelts.size();i++)
     {

        Element chld = listofelts.get(i);

        dec = decrypt(chld.getChild("emotion").getText());
                  chld.getChild("emotion").setText(dec);

     }
            XMLOutputter xmloutput = new XMLOutputter();
            xmloutput.setFormat(Format.getPrettyFormat());
            xmloutput.output(doc, new FileWriter(inputfile));
        }
        catch(Exception e)
        {
            System.out.println(e);
        }

      }
```

Code 12.2 Method to parse and update XML document

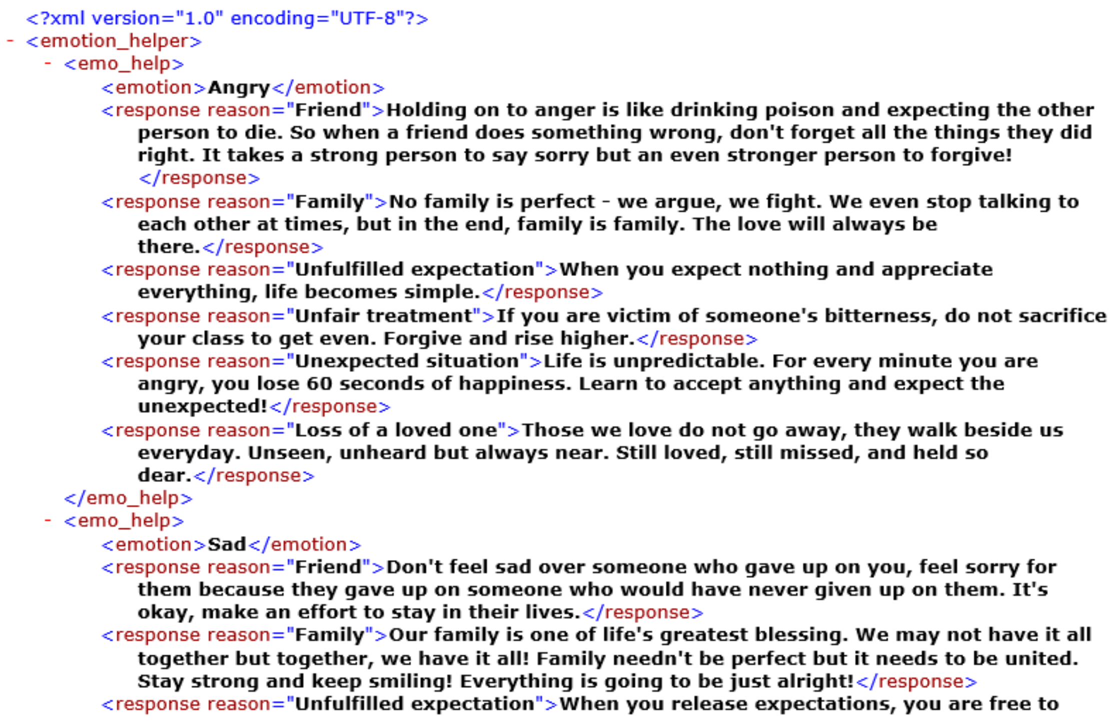
4. Invoke the parser method in the main function as follows

```
parser("emohelp.xml");
```

Code 12.3 Invocation of parser method

5. Run the project and open the XML file. It will contain decrypted values in the data in emotion tag.

OUTPUT



```
  <?xml version="1.0" encoding="UTF-8"?>
- <emotion_helper>
  - <emo_help>
        <emotion>Angry</emotion>
        <response reason="Friend">Holding on to anger is like drinking poison and expecting the other
           person to die. So when a friend does something wrong, don't forget all the things they did
           right. It takes a strong person to say sorry but an even stronger person to forgive!
           </response>
        <response reason="Family">No family is perfect - we argue, we fight. We even stop talking to
           each other at times, but in the end, family is family. The love will always be
           there.</response>
        <response reason="Unfulfilled expectation">When you expect nothing and appreciate
           everything, life becomes simple.</response>
        <response reason="Unfair treatment">If you are victim of someone's bitterness, do not sacrifice
           your class to get even. Forgive and rise higher.</response>
        <response reason="Unexpected situation">Life is unpredictable. For every minute you are
           angry, you lose 60 seconds of happiness. Learn to accept anything and expect the
           unexpected!</response>
        <response reason="Loss of a loved one">Those we love do not go away, they walk beside us
           everyday. Unseen, unheard but always near. Still loved, still missed, and held so
           dear.</response>
    </emo_help>
  - <emo_help>
        <emotion>Sad</emotion>
        <response reason="Friend">Don't feel sad over someone who gave up on you, feel sorry for
           them because they gave up on someone who would have never given up on them. It's
           okay, make an effort to stay in their lives.</response>
        <response reason="Family">Our family is one of life's greatest blessing. We may not have it all
           together but together, we have it all! Family needn't be perfect but it needs to be united.
           Stay strong and keep smiling! Everything is going to be just alright!</response>
        <response reason="Unfulfilled expectation">When you release expectations, you are free to
```

Figure 12.1 XML file after decryption

RESULT

The encrypted XML has been decrypted and encrypted data has been replaced with the decrypted data.