PRACTICAL MANUAL

For TEACHER's REFERENCE

SUBJECT: CLOUD COMPUTING

M.Sc. I.T. Part I, Semester I

Choice Based

with effect from the academic year

2019 – 2020

Practical No: 01

Aim: Write a program for implementing Client Server communication model using TCP.

Practical 1A: A client server based program using TCP to find if the number entered is prime.

Code:

1. **tcpServerPrime.java**

```java
import java.net.*;
import java.io.*;
class tcpServerPrime
{
        public static void main(String args[])
        {
                try
                {
                        ServerSocket ss = new ServerSocket(8001);
                        System.out.println("Server Started...............");
                        Socket s = ss.accept();


                        DataInputStream in = new DataInputStream(s.getInputStream()); int
                        x= in.readInt();


                        DataOutputStream otc = new DataOutputStream(s.getOutputStream()); int
                        y = x/2;


                        if(x ==1 || x ==2 || x ==3)
                        {
                                otc.writeUTF(x + "is Prime");
                                System.exit(0);
```

```
                }

                for(int i=2; i<=y; i++)

                {

                        if(x%i != 0)

                        {

                                otc.writeUTF(x + " is Prime");

                        }
                        else

                        {

                                otc.writeUTF(x + " is not Prime");

                        }


                }

        }

        catch(Exception e)

        {

                System.out.println(e.toString());

        }

    }

}
```

2. **tcpClientPrime.java**

```
import java.net.*;

import java.io.*;


class tcpClientPrime

{

        public static void main(String args[])
```

```
        {
                try
                {
                        Socket cs = new Socket("LocalHost",8001);
                        BufferedReader infu = new BufferedReader(new

InputStreamReader(System.in));

                        System.out.println("Enter a number : ");

                        int a = Integer.parseInt(infu.readLine());

                        DataOutputStream out = new

DataOutputStream(cs.getOutputStream());

                        out.writeInt(a);

                        DataInputStream in = new DataInputStream(cs.getInputStream());
                        System.out.println(in.readUTF()); cs.close();
                }
                catch(Exception e)
                {
                        System.out.println(e.toString());
                }
        }
}
```

Output:

Practical 1B: A client server TCP based chatting application.

Code:

1. **ChatServer.java**

```java
import java.net.*;

import java.io.*;

class ChatServer
{
        public static void main(String args[])
        {
                try
                {
                        ServerSocket ss = new ServerSocket(8000);
                        System.out.println("Waiting for client to connect..");
                        Socket s = ss.accept();

                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

                        DataOutputStream out = new DataOutputStream(s.getOutputStream());
                        DataInputStream in = new DataInputStream(s.getInputStream()); String
                        receive, send;

                        while((receive = in.readLine()) != null)

                        {
```

```
                    if(receive.equals("STOP"))

                            break;

                    System.out.println("Client Says : "+receive);

                    System.out.print("Server Says : ");

                    send = br.readLine();

                    out.writeBytes(send+"\n");

            }

            br.close();

            in.close();

            out.close();

            s.close();

        }

        catch(Exception e)

        {

                e.printStackTrace();

        }

    }

}
```

2. **ChatClient.java**

```
import java.net.*;

import java.io.*;

class ChatClient

{

    public static void main(String args[])

    {

            try

            {
```

```
Socket s = new Socket("Localhost",8000);
BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
DataOutputStream out = new DataOutputStream(s.getOutputStream());
DataInputStream in = new DataInputStream(s.getInputStream()); String
msg;

System.out.println("To stop chatting with server type STOP");
System.out.print("Client Says: "); while((msg = br.readLine()) !=
null)

{
        out.writeBytes(msg+"\n");

        if(msg.equals("STOP"))

                break;

        System.out.println("Server Says : "+in.readLine());

        System.out.print("Client Says : ");

}
br.close();

in.close();

out.close();

s.close();

}

catch(Exception e)


{
e.printStackTrace();

}
}
}
```

Output:

Server: **Client:**

```
E:\Ds_Yugi>java ChatServer
Waiting for client to connect..
Client Says : Hii...
Server Says : Hello
Client Says : How are you?
Server Says : Fine, thank you..
```

```
E:\Ds_Yugi>java ChatClient
To stop chatting with server type STOP
Client Says: Hii...
Server Says : Hello
Client Says : How are you?
Server Says : Fine, thank you..
Client Says : STOP
```

Practical No: 02

Aim: Write a program for implementing Client Server communication model using UDP.

Practical 2A: A client server based program using UDP to find if the number entered is even or odd.

Code:

1. **udpServerEO.java**

/*Program which finds entered number is even or odd */

import java.io.*;

import java.net.*;

public class udpServerEO

{

      public static void main(String args[])

      {

          try

          {

```
DatagramSocket ds = new DatagramSocket(2000);
byte b[] = new byte[1024];

DatagramPacket dp = new DatagramPacket(b,b.length);
ds.receive(dp);

String str = new String(dp.getData(),0,dp.getLength());
System.out.println(str);

int a= Integer.parseInt(str);

String s= new String();

if (a%2 == 0)

        s = "Number is even";

else

        s = "Number is odd";


byte b1[] = new byte[1024];

b1 = s.getBytes();

DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1000);

ds.send(dp1);

    }

    catch(Exception e)

    {

        e.printStackTrace();

    }

  }

}
```

2. **udpClientEO.java**

/*Program which finds entered number is even or odd*/

import java.io.*;

```java
import java.net.*;

public class udpClientEO
{
        public static void main(String args[])
        {
                try
                {
                        DatagramSocket ds = new DatagramSocket(1000);
                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

                        System.out.println("Enter a number : ");

                        String num = br.readLine();

                        byte b[] = new byte[1024];

                        b=num.getBytes();

                        DatagramPacket dp = new
DatagramPacket(b,b.length,InetAddress.getLocalHost(),2000);

                        ds.send(dp);

                        byte b1[] = new byte[1024];

                        DatagramPacket dp1 = new DatagramPacket(b1,b1.length);
                        ds.receive(dp1);

                        String str = new String(dp1.getData(),0,dp1.getLength());
                        System.out.println(str);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Output:

Practical 2B: A client server based program using UDP to find the factorial of the entered number.

Code:

1. **udpServerFact.java**

```
/*Program which calculate factorial of a number*/

import java.io.*;

import java.net.*;

public class udpServerFact

{

        public static void main(String args[])

        {

                try

                {

                        DatagramSocket ds = new DatagramSocket(2000);
                        byte b[] = new byte[1024];
```

```
                    DatagramPacket dp = new DatagramPacket(b,b.length);
                    ds.receive(dp);

                    String str = new String(dp.getData(),0,dp.getLength());
                    System.out.println(str);

                    int a= Integer.parseInt(str);

                    int f = 1, i;

                    String s= new String();

                    for(i=1;i<=a;i++)

                    {

                            f=f*i;

                    }

                    s=Integer.toString(f);

                    String str1 = "The Factorial of " + str + " is : " + f;
                    byte b1[] = new byte[1024]; b1 = str1.getBytes();

                    DatagramPacket dp1 = new

DatagramPacket(b1,b1.length,InetAddress.getLocalHost(),1000);

                    ds.send(dp1);

            }

            catch(Exception e)

            {

                    e.printStackTrace();

            }

        }

}
```

2. **udpClientFact.java**

/*Program which calculate factorial of a number*/

import java.io.*;

```java
import java.net.*;
public class udpClientFact
{
        public static void main(String args[])
        {
                try
                {
                        DatagramSocket ds = new DatagramSocket(1000);
                        BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
                        System.out.println("Enter a number : ");
                        String num = br.readLine();
                        byte b[] = new byte[1024];
                        b=num.getBytes();
                        DatagramPacket dp = new
DatagramPacket(b,b.length,InetAddress.getLocalHost(),2000);
                        ds.send(dp);
                        byte b1[] = new byte[1024];
                        DatagramPacket dp1 = new DatagramPacket(b1,b1.length);
                        ds.receive(dp1);
                        String str = new String(dp1.getData(),0,dp1.getLength());
                        System.out.println(str);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
        }
}
```

Output:

Practical 3C: A program to implement simple calculator operations like addition, subtraction, multiplication and division.

Code:

1. **RPCServer.java**

```
import java.util.*;

import java.net.*;

class RPCServer

{

        DatagramSocket ds;

        DatagramPacket dp;

        String str,methodName,result;

        int val1,val2;

        RPCServer()

        {
```

```
try
{
        ds=new DatagramSocket(1200);

        byte b[]=new byte[4096];

        while(true)
        {
                dp=new DatagramPacket(b,b.length);

                ds.receive(dp);

                str=new String(dp.getData(),0,dp.getLength());

                if(str.equalsIgnoreCase("q"))
                {
                        System.exit(1);
                }
                else
                {

                        StringTokenizer st = new StringTokenizer(str," "); int
                        i=0;

                        while(st.hasMoreTokens())
                        {
                                String token=st.nextToken();

                                methodName=token;

                                val1 = Integer.parseInt(st.nextToken());

                                val2 = Integer.parseInt(st.nextToken());
                        }
                }
                System.out.println(str);

                InetAddress ia = InetAddress.getLocalHost();
```

```
                    if(methodName.equalsIgnoreCase("add"))

                    {

                            result= "" + add(val1,val2);

                    }

                    else if(methodName.equalsIgnoreCase("sub"))

                    {

                            result= "" + sub(val1,val2);

                    }

                    else if(methodName.equalsIgnoreCase("mul"))

                    {

                            result= "" + mul(val1,val2);


                    }

                    else if(methodName.equalsIgnoreCase("div"))

                    {

                            result= "" + div(val1,val2);

                    }

                    byte b1[]=result.getBytes();

                    DatagramSocket ds1 = new DatagramSocket();
                    DatagramPacket dp1 = new

DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
                    System.out.println("result : "+result+"\n");
                    ds1.send(dp1);
            }
        }
        catch (Exception e)

        {

            e.printStackTrace();

        }
```

```
        }
        public int add(int val1, int val2)
        {
                return val1+val2;
        }
        public int sub(int val3, int val4)
        {
                return val3-val4;
        }


        public int mul(int val3, int val4)
        {
                return val3*val4;
        }
        public int div(int val3, int val4)
        {
                return val3/val4;
        }
        public static void main(String[] args)
        {
                new RPCServer();
        }
}
```

2. **RPCClient.java**

```
import java.io.*;
import java.net.*;
class RPCClient
```

```java
{
        RPCClient()
        {
                try
                {
                        InetAddress ia = InetAddress.getLocalHost();

                        DatagramSocket ds = new DatagramSocket();

                        DatagramSocket ds1 = new DatagramSocket(1300);

                        System.out.println("\nRPC Client\n");

                        System.out.println("Enter method name and parameter like add 3
4\n");

                        while (true)
                        {
                                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

                                String str = br.readLine();

                                byte b[] = str.getBytes();

                                DatagramPacket dp = new
DatagramPacket(b,b.length,ia,1200);

                                ds.send(dp);

                                dp = new DatagramPacket(b,b.length);

                                ds1.receive(dp);

                                String s = new String(dp.getData(),0,dp.getLength());
                                System.out.println("\nResult = " + s + "\n");
                        }
                }
                catch (Exception e)
                {
```

```
                    e.printStackTrace();

            }

      }

      public static void main(String[] args)

      {

            new RPCClient();

      }

}
```

Output:

```
C:\WINDOWS\system32\cmd.exe - java RPCClient

E:\Ds_Yugi>javac RPCClient.java

E:\Ds_Yugi>java RPCClient

RPC Client

Enter method name and parameter like add 34

sub 10 8

Result = 2

mul 27 2

Result = 54

add 20 7

Result = 27

div 10 2
```

```
C:\WINDOWS\system32\cmd.exe - java RPCServer

E:\Ds_Yugi>javac RPCServer.java

E:\Ds_Yugi>java RPCServer
sub 10 8
result : 2

mul 27 2
result : 54

add 20 7
result : 27

div 10 2
result : 5
```

, square root, cube and cube root of the entered number.

Code:

1. **RPCNumServer.java**

```java
import java.util.*;

import java.net.*;

import java.io.*;

class RPCNumServer

{

      DatagramSocket ds;
```

```
DatagramPacket dp;

String str,methodName,result;

int val;

RPCNumServer()

{

        try

        {

                ds=new DatagramSocket(1200);

                byte b[]=new byte[4096];

                while(true)

                {

                        dp=new DatagramPacket(b,b.length);

                        ds.receive(dp);

                        str=new String(dp.getData(),0,dp.getLength());
                        if(str.equalsIgnoreCase("q")) {

                                System.exit(1);

                        }

                        else

                        {

                                StringTokenizer st = new StringTokenizer(str," ");

                                int i=0;

                                while(st.hasMoreTokens())

                                {

                                        String token=st.nextToken();

                                        methodName=token;

                                        val = Integer.parseInt(st.nextToken());

                                }

                        }
```

```
                    System.out.println(str);

                    InetAddress ia = InetAddress.getLocalHost();

                    if(methodName.equalsIgnoreCase("square"))

                    {

                            result= "" + square(val);

                    }

                    else if(methodName.equalsIgnoreCase("squareroot"))

                    {

                            result= "" + squareroot(val);

                    }

                    else if(methodName.equalsIgnoreCase("cube"))

                    {

                             result= "" + cube(val);

                    }

                    else if(methodName.equalsIgnoreCase("cuberoot"))

                    {

                            result= "" + cuberoot(val);

                    }

                    byte b1[]=result.getBytes();

                    DatagramSocket ds1 = new DatagramSocket();

                    DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
                    System.out.println("result : "+result+"\n");
                    ds1.send(dp1);

              }

          }

          catch (Exception e)

          {
```

```java
                e.printStackTrace();

        }

}

public double square(int a) throws Exception

{

        double ans;

        ans = a*a;

        return ans;

}

public double squareroot(int a) throws Exception

{

        double ans;

        ans = Math.sqrt(a);

        return ans;

}

public double cube(int a) throws Exception

{

        double ans;

        ans = a*a*a;

        return ans;

}

public double cuberoot(int a) throws Exception

{

        double ans;

        ans = Math.cbrt(a);

        return ans;

}
```

```
        public static void main(String[] args)

        {

                new RPCNumServer();

        }

}
```

2. **RPCNumClient.java**

```java
import java.io.*;

import java.net.*;

class RPCNumClient

{

        RPCNumClient()

        {

                try

                {

                        InetAddress ia = InetAddress.getLocalHost();
                        DatagramSocket ds = new DatagramSocket();
                        DatagramSocket ds1 = new DatagramSocket(1300);
                        System.out.println("\nRPC Client\n");

                        System.out.println("1. Square of the number - square\n2. Square root of
the number - squareroot\n3. Cube of the number - cube\n4. Cube root of the number -
cuberoot");

                        System.out.println("Enter method name and the number\n");

                        while (true)

                        {

                                BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

                                String str = br.readLine();

                                byte b[] = str.getBytes();
```

DatagramPacket dp = new

DatagramPacket(b,b.length,ia,1200);

        ds.send(dp);

        dp = new DatagramPacket(b,b.length);

        ds1.receive(dp);

        String s = new String(dp.getData(),0,dp.getLength());
        System.out.println("\nResult = " + s + "\n");

    }

  }

  catch (Exception e)

  {

    e.printStackTrace();

  }

}

public static void main(String[] args)

{

  new RPCNumClient();

}

}

Output:

C:\WINDOWS\system32\cmd.exe - java RPCNumServer

E:\Ds_Yugi>javac RPCNumServer.java

E:\Ds_Yugi>java RPCNumServer
square 7
result : 49.0

squareroot 25
result : 5.0

cube 3
result : 27.0

```
C:\WINDOWS\system32\cmd.exe - java RPCNumClient

1. Square of the number - square
2. Square root of the number - squareroot
3. Cube of the number - cube
4. Cube root of the number - cuberoot
Enter method name and the number

square 7

Result = 49.0

squareroot 25

Result = 5.0

cube 3

Result = 27.0

cuberoot 27

Result = 3.0
```

<u>**Practical No: 03**</u>

Aim: A multicast Socket example.

Practical 3: Write a program for multicast server and clients receiving the multicast packets.

Code:

1. **BroadcastServer.java**

```java
import java.net.*;

import java.io.*;

import java.util.*;

public class BroadcastServer

{

        public static final int PORT = 1234;

        public static void main(String args[])throws Exception {

                MulticastSocket socket;

                DatagramPacket packet;

                InetAddress address;

                // set the multicast address to your local subnet
                address = InetAddress.getByName("239.1.2.3");
                socket = new MulticastSocket();
                // join a Multicast group and send the group messages
                socket.joinGroup(address);
                byte[] data = null;
                for(;;)

                {
                        Thread.sleep(10000);
                        System.out.println("Sending "); String str
                        = ("This is Neha Calling...."); data =
                        str.getBytes();
                        packet = new DatagramPacket(data, str.length(),address,PORT);

                        // Sends the packet
                        socket.send(packet);
                } // end for

        } // end main
} // end class BroadcastServer
```

2. **BroadcastClient.java**

```java
import java.net.*;

import java.io.*;

public class BroadcastClient
```

```
{
        public static final int PORT = 1234;

        public static void main(String args[])throws Exception {

                MulticastSocket socket;

                DatagramPacket packet;

                InetAddress address;

                // set the mulitcast address to your local subnet
                address = InetAddress.getByName("239.1.2.3");
                socket = new MulticastSocket(PORT);
                //join a Multicast group and wait for a message
                socket.joinGroup(address); byte[] data = new
                byte[100];

                packet = new DatagramPacket(data,data.length);

                for(;;)

                {

                        // receive the packets
                        socket.receive(packet);

                        String str = new String(packet.getData()); System.out.println("Message
                        received from "+ packet.getAddress() + "

Message is : "+str);

                } // for

        } // main

} // end BroadcastClient
```

Output:

Practical No: 04

Aim: Write a program to show the object communication using RMI.

Practical 4A: A RMI based application program to display current date and time.

Code:

1. **InterDate.java**

```
import java.rmi.*;

public interface InterDate extends Remote

{

        public String display() throws Exception;

}
```

2. **ServerDate.java**

```
import java.rmi.*;

import java.rmi.server.*;

import java.util.*;

public class ServerDate extends UnicastRemoteObject implements InterDate {

        public ServerDate() throws Exception

        {

        }

        public String display() throws Exception

        {

                String str = "";

                Date d = new Date();

                str = d.toString();

                return str;

        }

        public static void main(String args[]) throws Exception {

                ServerDate s1 = new ServerDate();

                Naming.bind("DS",s1);

                System.out.println("Object registered.....");

        }

}
```

3. **ClientDate.java**

import java.rmi.*;

import java.io.*;

public class ClientDate

{

        public static void main(String args[]) throws Exception {

            String s1;

            InterDate h1 = (InterDate)Naming.lookup("DS");

            s1 = h1.display();

            System.out.println(s1);

        }

}

Output:

```
C:\WINDOWS\system32\cmd.exe - rmiregistry
E:\Ds_Yugi>javac ServerDate.java

E:\Ds_Yugi>javac ClientDate.java

E:\Ds_Yugi>rmic ServerDate
Warning: generation and use of skeletons and static stubs for JRMP
is deprecated. Skeletons are unnecessary, and static stubs have
been superseded by dynamically generated stubs. Users are
encouraged to migrate away from using rmic to generate skeletons and static
stubs. See the documentation for java.rmi.server.UnicastRemoteObject.

E:\Ds_Yugi>rmiregistry
```

```
C:\WINDOWS\system32\cmd.exe - java ServerDate

E:\Ds_Yugi>java ServerDate
Object registered.....
```

```
C:\WINDOWS\system32\cmd.exe

E:\Ds_Yugi>java ClientDate
Thu Jan 04 17:38:00 IST 2018
```

Practical 4B: A RMI based application program that converts digits to words, e.g. 123 will be converted to one two three.

Code:

1. **InterConvert.java**

```java
import java.rmi.*;

public interface InterConvert extends Remote
{
        public String convertDigit(String no) throws Exception;
}
```

2. **ServerConvert.java**

```java
import java.rmi.*;
import java.rmi.server.*;
public class ServerConvert extends UnicastRemoteObject implements InterConvert {
        public ServerConvert() throws Exception
        {
        }
        public String convertDigit(String no) throws Exception {
                String str = "";
                for(int i = 0; i < no.length(); i++)
                {
                        int p = no.charAt(i);
                        if( p == 48)
                        {
                                str += "zero ";
                        }
                        if( p == 49)
                        {
                                str += "one ";
                        }
                        if( p == 50)
```

```
{
        str += "two ";
}
if( p == 51)
{
        str += "three ";
}
if( p == 52)
{
        str += "four ";
}
if( p == 53)
{
        str += "five ";
}
if( p == 54)
{
        str += "six ";
}
if( p == 55)
{
        str += "seven ";
}
if( p == 56)
{
        str += "eight ";
}
```

```
                    if( p == 57)

                    {

                            str += "nine ";

                    }

            }

            return str;

    }

    public static void main(String args[]) throws Exception {

            ServerConvert s1 = new ServerConvert();

            Naming.bind("Wrd",s1);

            System.out.println("Object registered....");

    }

}
```

3. **ClientConvert.java**

```
import java.rmi.*;

import java.io.*;

public class ClientConvert

{

    public static void main(String args[]) throws Exception {


            InterConvert h1 = (InterConvert)Naming.lookup("Wrd");
            BufferedReader br = new BufferedReader(new


InputStreamReader(System.in)); System.out.println("Enter
            a number : \t"); String no = br.readLine();

            String ans = h1.convertDigit(no);

            System.out.println("The word representation of the entered digit is : " +ans);

    }
```

}

Output:

Practical No: 05

Aim: Show the implementation of web services.

What Are Web Services?

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

Types of Web Services:

On the conceptual level, a service is a software component provided through a network-accessible endpoint. The service consumer and provider use messages to exchange invocation request and response information in the form of self-containing documents that make very few assumptions about the technological capabilities of the receiver.

On a technical level, web services can be implemented in various ways. The two types of web services can be distinguished as "big" web services and "RESTful" web services.

1) **"Big" Web Services:**

In Java EE 6, JAX-WS provides the functionality for "big" web services. Big web services use XML messages that follow the Simple Object Access Protocol (SOAP) standard, an XML language defining a message architecture and message formats. Such systems often contain a machine-readable description of the operations offered by the service, written in the Web Services Description Language (WSDL), an XML language for defining interfaces syntactically.

The SOAP message format and the WSDL interface definition language have gained widespread adoption. Many development tools, such as NetBeans IDE, can reduce the complexity of developing web service applications.

A SOAP-based design must include the following elements.

▫ A formal contract must be established to describe the interface that the web service offers. WSDL can be used to describe the details of the contract, which may include messages, operations, bindings, and the location of the web service. You may also process SOAP messages in a JAX-WS service without publishing a WSDL.

▫ The architecture must address complex nonfunctional requirements. Many web service specifications address such requirements and establish a common vocabulary for them. Examples include transactions, security, addressing, trust, coordination, and so on.

▢   The architecture needs to handle asynchronous processing and invocation. In such cases, the infrastructure provided by standards, such as Web Services Reliable Messaging

(WSRM), and APIs, such as JAX-WS, with their client-side asynchronous invocation support, can be leveraged out of the box.

2)   **RESTful Web Services:**

In Java EE 6, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. REST is well suited for basic, ad hoc integration scenarios. RESTful web services, often better integrated with HTTP than SOAP-based services are, do not require XML messages or WSDL service–API definitions.

Project Jersey is the production-ready reference implementation for the JAX-RS specification. Jersey implements support for the annotations defined in the JAX-RS specification, making it easy for developers to build RESTful web services with Java and the Java Virtual Machine (JVM).

Because RESTful web services use existing well-known W3C and Internet Engineering Task Force (IETF) standards (HTTP, XML, URI, MIME) and have a lightweight infrastructure that allows services to be built with minimal tooling, developing RESTful web services is inexpensive and thus has a very low barrier for adoption. You can use a development tool such as NetBeans IDE to further reduce the complexity of developing RESTful web services.

A RESTful design may be appropriate when the following conditions are met.

▢   The web services are completely stateless. A good test is to consider whether the interaction can survive a restart of the server.

▢   A caching infrastructure can be leveraged for performance. If the data that the web service returns is not dynamically generated and can be cached, the caching infrastructure that web servers and other intermediaries inherently provide can be leveraged to improve performance. However, the developer must take care because such caches are limited to the HTTP GET method for most servers.

▢   The service producer and service consumer have a mutual understanding of the context and content being passed along. Because there is no formal way to describe the web services interface, both parties must agree out of band on the schemas that describe the data being exchanged and on ways to process it meaningfully. In the real world, most commercial applications that expose services as RESTful implementations also distribute so-called value-added toolkits that describe the interfaces to developers in popular programming languages.

◻ Bandwidth is particularly important and needs to be limited. REST is particularly useful for limited-profile devices, such as PDAs and mobile phones, for which the overhead of headers and additional layers of SOAP elements on the XML payload must be restricted.

◻ Web service delivery or aggregation into existing web sites can be enabled easily with a RESTful style. Developers can use such technologies as JAX-RS and Asynchronous JavaScript with XML (AJAX) and such toolkits as Direct Web Remoting (DWR) to consume the services in their web applications. Rather than starting from scratch, services can be exposed with XML and consumed by HTML pages without significantly refactoring the existing web site architecture. Existing developers will be more productive because they are adding to something they are already familiar with rather than having to start from scratch with new technology.

Deciding Which Type of Web Service to Use:

Basically, you would want to use RESTful web services for integration over the web and use big web services in enterprise application integration scenarios that have advanced quality of service (QoS) requirements.

◻ **JAX-WS**: addresses advanced QoS requirements commonly occurring in enterprise computing. When compared to JAX-RS, JAX-WS makes it easier to support the WS-* set of protocols, which provide standards for security and reliability, among other things, and interoperate with other WS-* conforming clients and servers.

◻ **JAX-RS**: makes it easier to write web applications that apply some or all of the constraints of the REST style to induce desirable properties in the application, such as loose coupling (evolving the server is easier without breaking existing clients), scalability (start small and grow), and architectural simplicity (use off-the-shelf components, such as proxies or HTTP routers). You would choose to use JAX-RS for your web application because it is easier for many types of clients to consume RESTful web services while enabling the server side to evolve and scale. Clients can choose to consume some or all aspects of the service and mash it up with other web-based services.

Practical 5A: Implementing "Big" Web Service.

1) **<u>Creating a Web Service</u>**

A. **<u>Choosing a Container</u>:**

1. Choose File > New Project. Select Web Application from the Java Web.
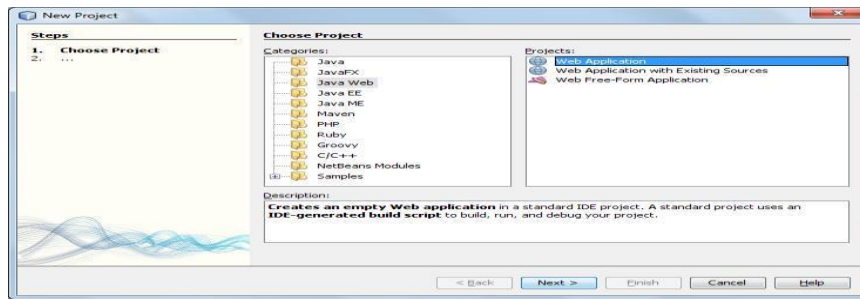
2. Name the project CalculatorWSApplication. Select a location for the project. Click Next.
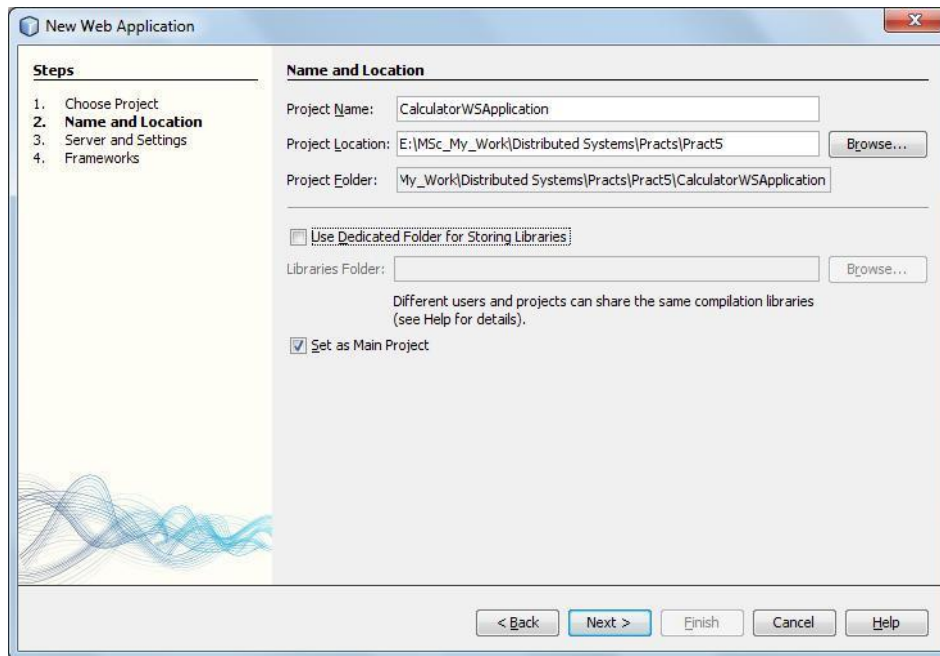
3. Select your server and Java EE version and click Finish.

B. **Creating a Web Service from a Java Class**

1. Right-click the CalculatorWSApplication node and choose New > Web Service.
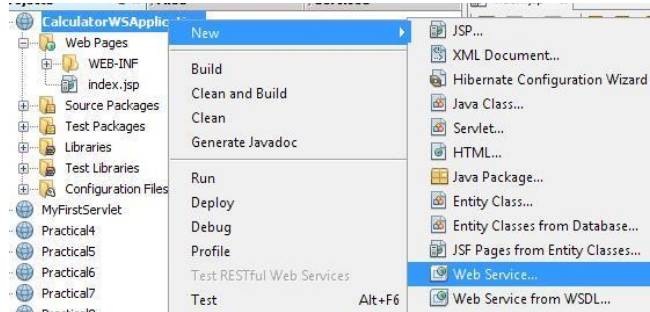
2. Name the web service CalculatorWS and type org.me.calculator in Package. Leave Create Web Service from Scratch selected. If you are creating a Java EE 6 project on GlassFish or WebLogic, select Implement Web Service as a Stateless Session Bean.
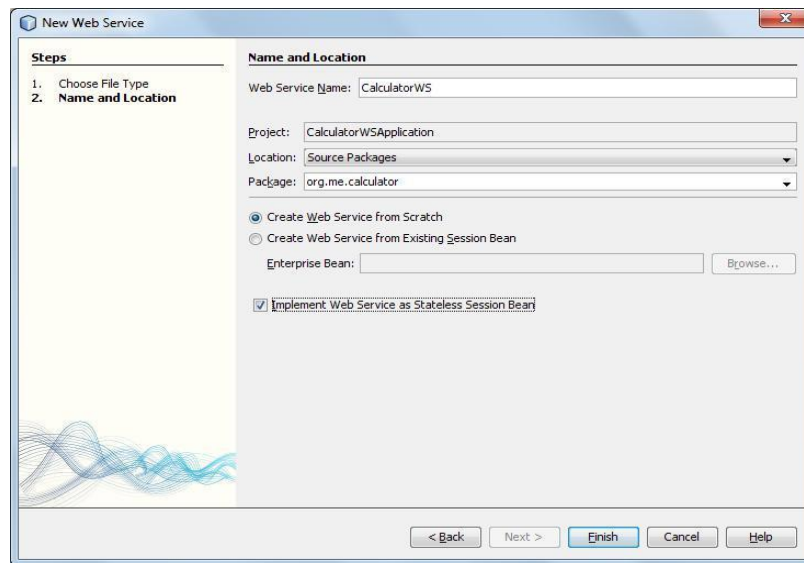
3. Click Finish. The Projects window displays the structure of the new web service and the source code is shown in the editor area.

2) **Adding an Operation to the Web Service**

The goal of this exercise is to add to the web service an operation that adds two numbers received from a client. The NetBeans IDE provides a dialog for adding an operation to a web service. You can open this dialog either in the web service visual designer or in the web service context menu.

A. **<u>To add an operation to the web service</u>:**

1. Change to the Design view in the editor.

2. Click Add Operation in either the visual designer or the context menu. The Add Operation dialog opens.
3. In the upper part of the Add Operation dialog box, type add in Name and type int in the Return Type drop-down list.

4. In the lower part of the Add Operation dialog box, click Add and create a parameter of type int named i.

5. Click Add again and create a parameter of type int called j. You now see the following:
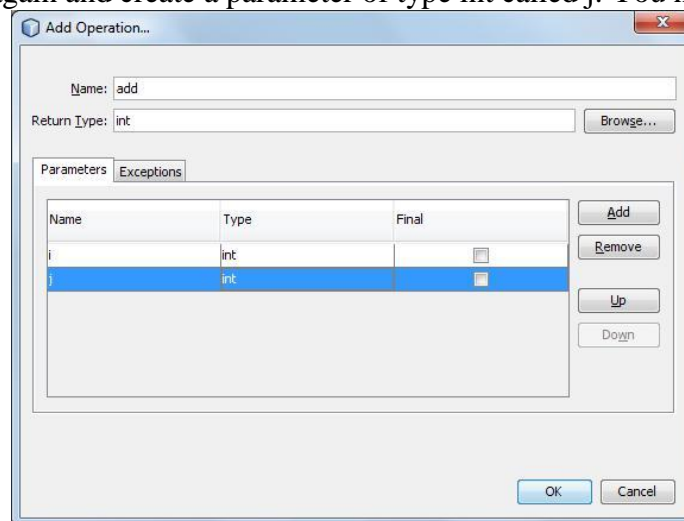
6. Click OK at the bottom of the Add Operation dialog box. You return to the editor.
7. The visual designer now displays the following:
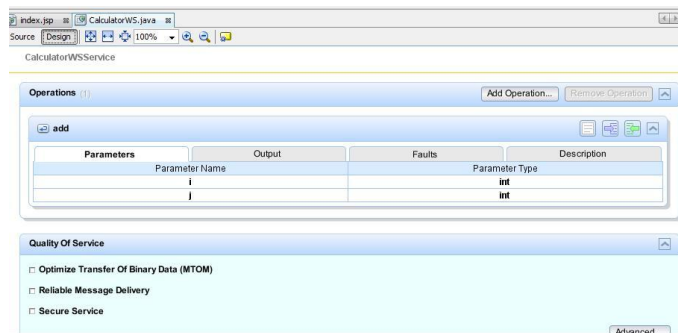
8. Click Source. And code the following.

@WebMethod(operationName = "add")

  public int add(@WebParam(name = "i") int i, @WebParam(name = "j") int j)

{

    int k = i + j;

    return k;

  }

3) Deploying and Testing the Web Service

After you deploy a web service to a server, you can use the IDE to open the server's test client, if the server has a test client. The GlassFish and WebLogic servers provide test clients.

A. To test successful deployment to a GlassFish or WebLogic server:

1. Right-click the project and choose Deploy. The IDE starts the application server, builds the application, and deploys the application to the server
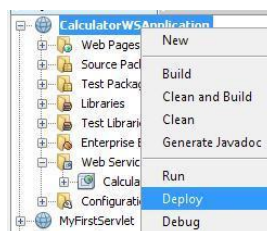
2. In the IDE's Projects tab, expand the Web Services node of the CalculatorWSApplication project. Right-click the CalculatorWS node, and choose Test Web Service.
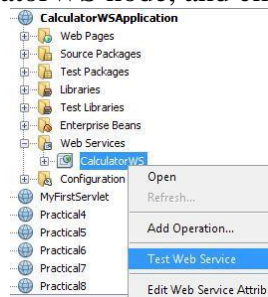
3. The IDE opens the tester page in your browser, if you deployed a web application to the GlassFish server.
4. If you deployed to the GlassFish server, type two numbers in the tester page, as shown below:
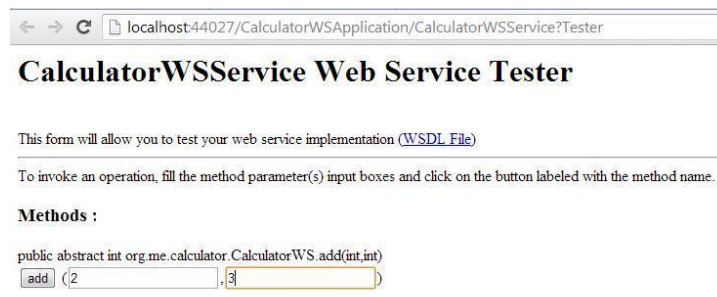
5. The sum of the two numbers is displayed:

4) **Consuming the Web Service**

Now that you have deployed the web service, you need to create a client to make use of the web service's add method.

1. **Client: Java Class in Java SE Application**
1. Choose File > New Project. Select Java Application from the Java category. Name the project CalculatorWS_Client_Application. Leave Create Main Class selected and accept all other default settings. Click Finish.\
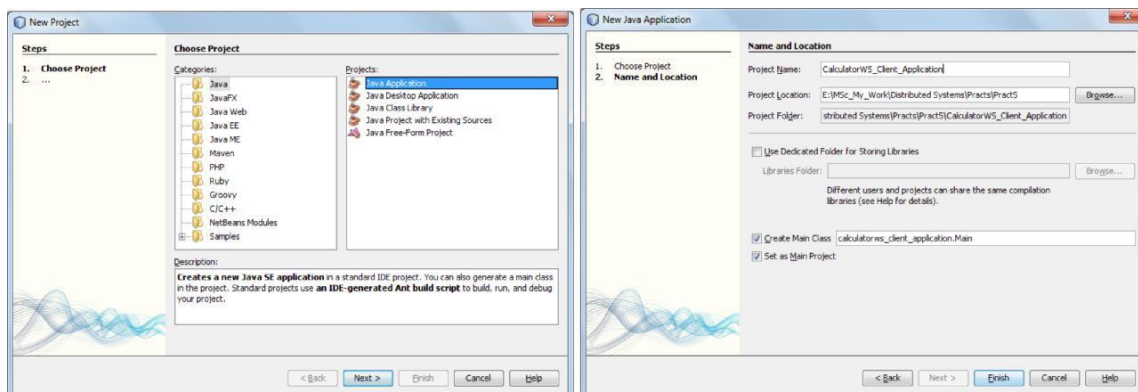
2. Right-click the CalculatorWS_Client_Application node and choose New > Web Service Client. The New Web Service Client wizard opens.
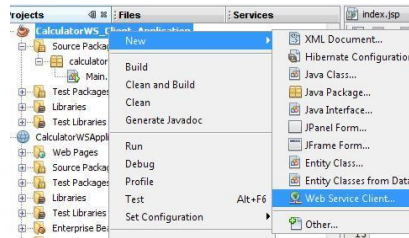
3. Select Project as the WSDL source. Click Browse. Browse to the CalculatorWS web service in the CalculatorWSApplication project. When you have selected the web service, click OK.
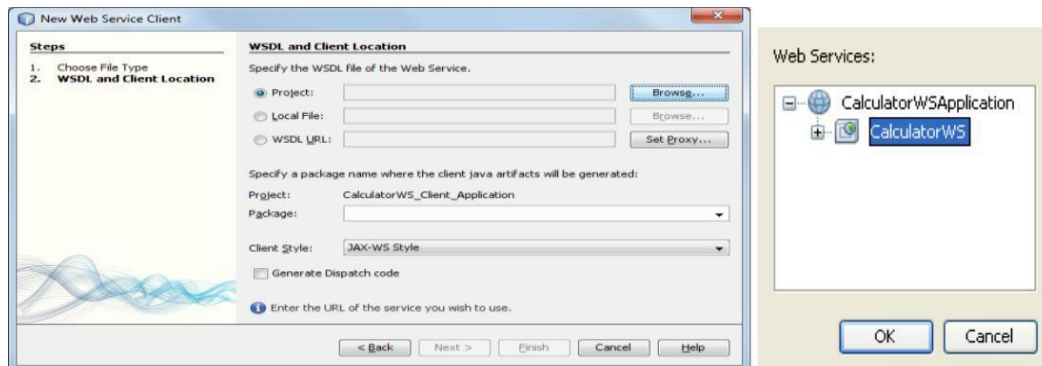
4. Do not select a package name. Leave this field empty.
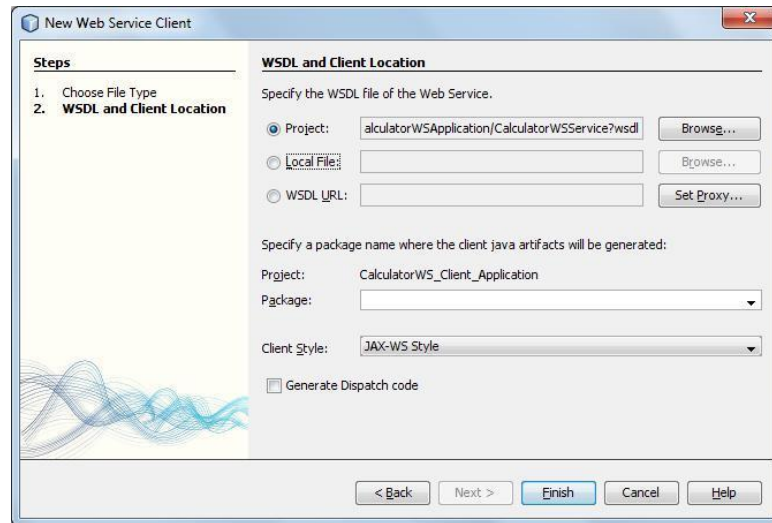
5. Leave the other settings at default and click Finish. The Projects window displays the new web service client, with a node for the add method that you created:
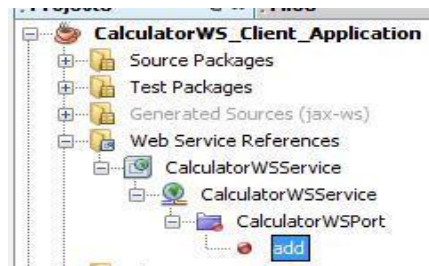
6. Double-click your main class so that it opens in the Source Editor. Drag the add node below the main() method.
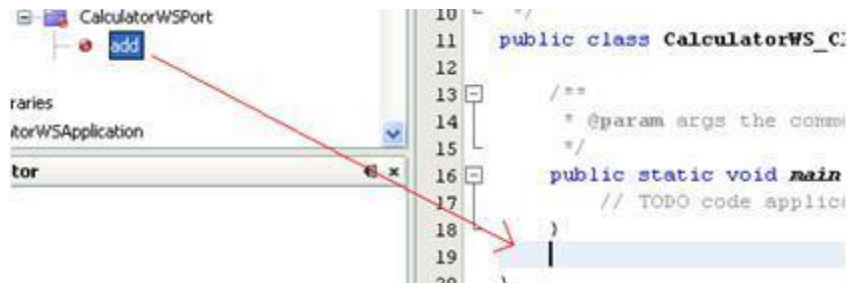
You now see the following:

```
public static void main(String[] args)

{

    // TODO code application logic here

}

private static int add(int i, int j)

{

        org.me.calculator.CalculatorWS_Service service = new
org.me.calculator.CalculatorWS_Service();

        org.me.calculator.CalculatorWS port = service.getCalculatorWSPort();
        return port.add(i, j);

}
```

7. In the main() method body, replace the TODO comment with code that initializes values for i and j, calls add(), and prints the result.

```
public static void main(String[] args)

{

    int i = 3;

    int j = 4;

    int result = add(i, j);

    System.out.println("Result = " + result);

}
```

8. Surround the main() method code with a try/catch block that prints an exception.

```
public static void main(String[] args)

{

    try

    {

        int i = 3;
```

```
    int j = 4;

    int result = add(i, j);

    System.out.println("Result = " + result);

} catch (Exception ex) {

    System.out.println("Exception: " + ex);

}

}
```

9. Right-click the project node and choose Run.

The Output window now shows the sum:

compile:

run:

Result = 7

BUILD SUCCESSFUL (total time: 1 second)

Practical 5B: Implementing Web Service that connects to MySQL database.

⬚ **Building Web Service:-**

⬚ JAX-WS is an important part of the Java EE platform.

⬚ JAX-WS simplifies the task of developing Web services using Java technology.

⬚ It provides support for multiple protocols such as SOAP, XML and by providing a facility for supporting additional protocols along with HTTP.

⬚ With its support for annotations, JAX-WS simplifies Web service development and reduces the size of runtime files.

⬚ Here basic demonstration of using IDE to develop a JAX-WS Web Service is given.

⬚ After creating the web service, create web service clients that use the Web service over a network which is called consuming a web service.

⬚ The client is a servlet in a web application.

⬚ Let's build a Web Service that returns the book name along with its cost for a particular ISBN.

⬚ To begin building this service, create the data store. The server will access the data stored in a MySQL table to serve the client.

1) **Creating MySQL DB Table**

create database bookshop;

use bookshop;

? **Create a table named Books that will store valid books information**

create table books(isbn varchar(20) primary key, bookname varchar(100), bookprice varchar(10));

? **Insert valid records in the Books table**

insert into books values("111-222-333","Learn My SQL","250");

insert into books values("111-222-444","Java EE 6 for Beginners","850"); insert

into books values("111-222-555","Programming with Android","500"); insert

into books values("111-222-666","Oracle Database for you","400");

insert into books values("111-222-777","Asp.Net for advanced programmers","1250");

2) **Creating a web service**

   i. **Choosing a container**

   ▪ Web service can be either deployed in a Web container or in an EJB container.

   ▪ If a Java EE 6 application is created, use a Web container because EJBs can be placed directly in a Web application.

   ii. **Creating a web application**

   ▪ To create a Web application, select File - New Project.

⮚ New Project dialog box appears. Select Java Web available under the Categories section and Web Application available under the Projects section. Click Next.

⮚ New Web Application dialog box appears. Enter BookWS as the project name in the Project Name textbox and select the option Use Dedicated Folder for Storing Libraries.

⮚ Click Next. Server and Settings section of the New Web Application dialog box appears. Choose the default i.e. GlassFish v3 Domain as the Web server, the Java EE 6 Web as the Java EE version and the Context Path.

⮚ Click –Finish

⮚ The Web application named BookWS is created.

iii. **Creating a web service**

⮚ Right-click the BookWS project and select New -> Web Service as shown in diagram.
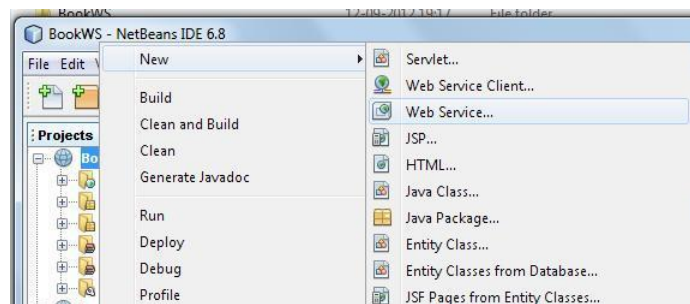
⮚ New Web Service dialog box appears. Enter the name BookWS in the Web Service Name textbox, webservice in the Package textbox, select the option Create Web Service from scratch and also select the option implement web service as a stateless session bean as shown in the diagram.
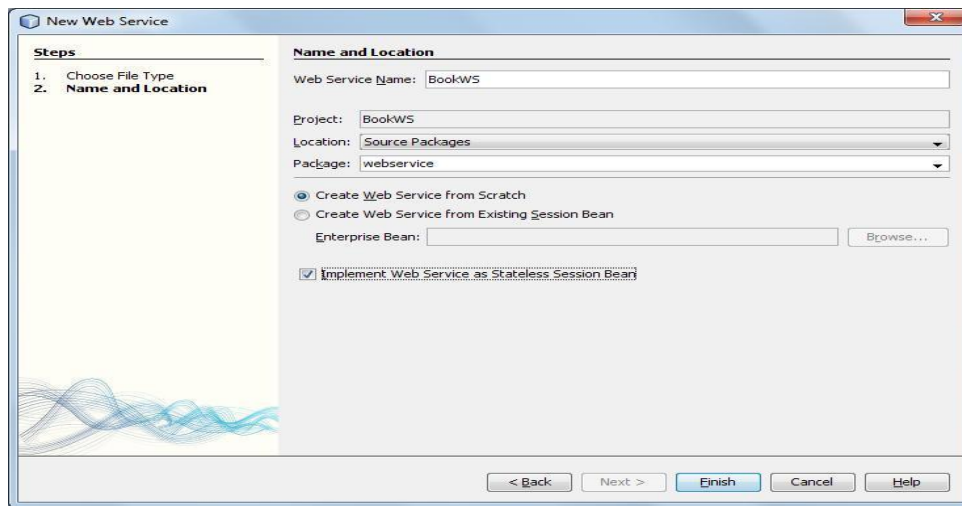
⬜ Click Finish.

⬜ The web service in the form of java class is ready.

3) **Designing the web service**

Now add an operation which will accept the ISBN number from the client to the web service.

i. **Adding an operation to the web service**

⬜ Change the source view of the BookWS.java to design view by clicking Design available just below the name of the BookWS.java tab.

⬜ The window changes as shown in the diagram.

⬜ Click Add Operation available in the design view of the web service.

⬜ Add Operation dialog appears. Enter the name getBookDetails in the Name textbox and java.lang.String in the Return Type textbox as shown in the diagram.

⬜ In Add Operation dialog box, click Add and create a parameter of the type String named isbn as shown in the diagram.
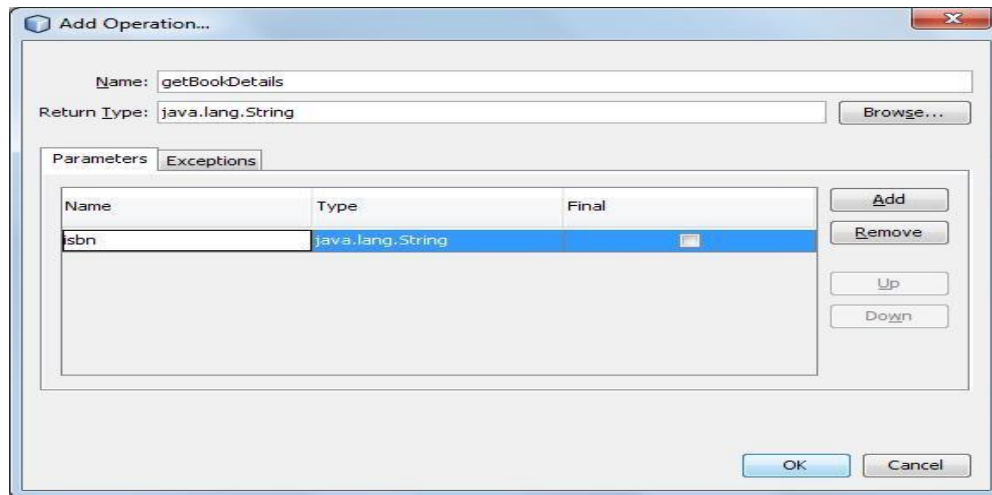
* Click Ok. The design view displays the operation added as shown in the diagram.
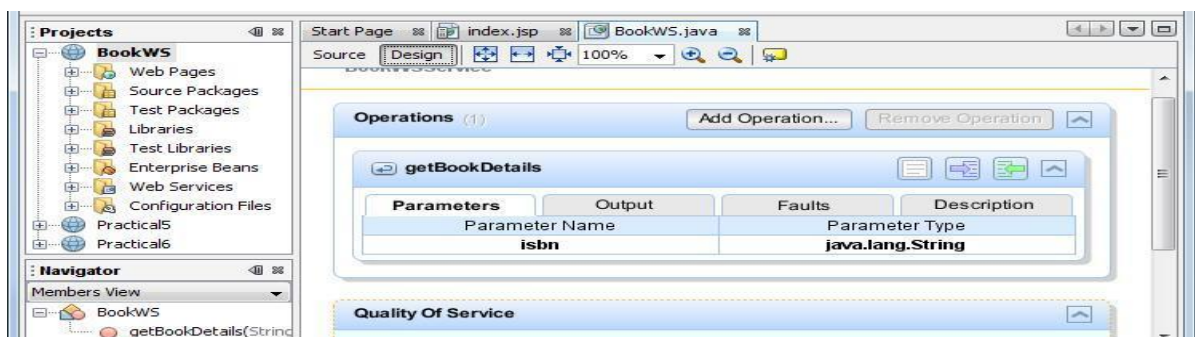
* Click Source. The code spec expands due to the operation added to the web service as shown in the diagram.
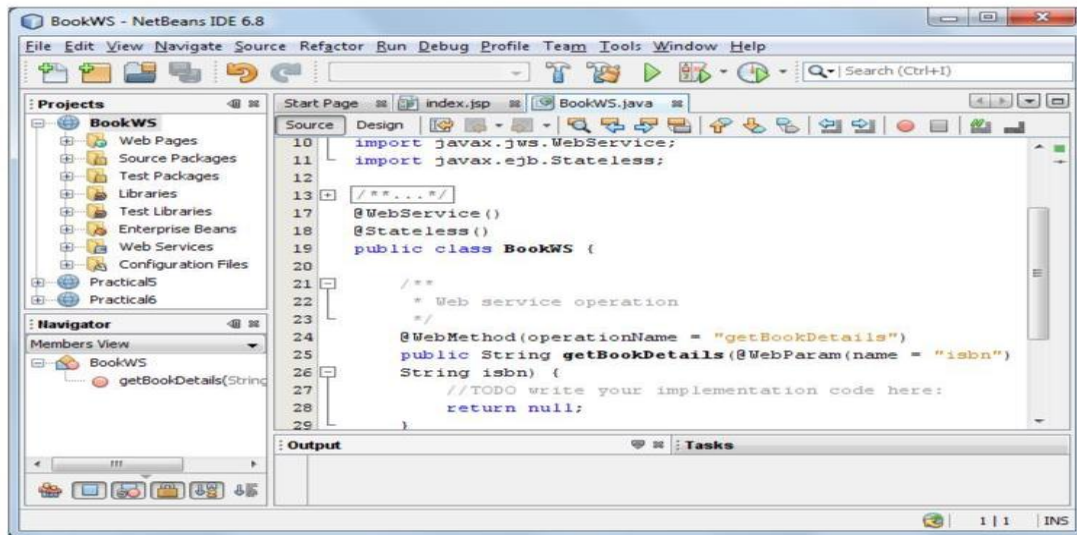
⬜ Modify the code spec of the web service BookWS.java.

Code Spec

package webservice;

import java.sql.*;

import javax.jws.WebMethod;

import javax.jws.WebParam;

import javax.jws.WebService;

import javax.ejb.Stateless;

@WebService()

@Stateless()

public class BookWS {

  /**

   * Web service operation */
  @WebMethod(operationName = "getBookDetails") public
  String getBookDetails(@WebParam(name = "isbn") String
  isbn) {

    //TODO write your implementation code here:

```
        Connection dbcon = null;

        Statement stmt = null;

        ResultSet rs = null;

        String query = null;

        try

        {

            Class.forName("com.mysql.jdbc.Driver").newInstance();

            dbcon                                                        =
DriverManager.getConnection("jdbc:mysql://localhost/bookshop","root","123");

            stmt = dbcon.createStatement();

            query = "select * from books where isbn = '" +isbn+ "'"; rs
            = stmt.executeQuery(query); rs.next();

            String bookDetails = "<h1>The name of the book is <b>" +rs.getString("bookname") +
"</b> and its cost is <b>" +rs.getString("bookprice") + "</b></h1>.";

            return bookDetails;

        }

        catch(Exception e)


        {

            System.out.println("Sorry failed to connect to the database.." + e.getMessage());

        }

        return null;

    }

}
```

4) **Adding the MySQL connector**

   - We need to add a reference of MySQL connector to our web service. It is via this
     connector that our web service will be able to communicate with the database.

☐ Right click on the libraries and select Add JAR/Folder as shown in the diagram.
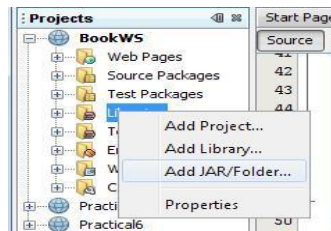
☐ Choose the location where mysql-coonector-java-5.1.10-bin is located, select it and click on open as shown.
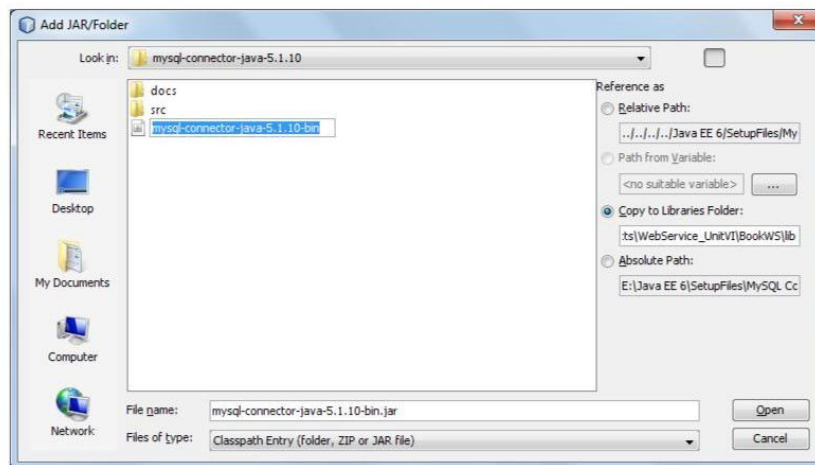
5) **Deploying and testing the web service**

☐ When a web service is deployed to a web container, the IDE allows testing the web service to see if it functions as expected.

☐ The tester application provided by GlassFish, is integrated into the IDE for this purpose as it allows the developer to enter values and test them.

☐ No facility for testing whether an EJB module is deployed successfully is currently available.

▫ To test the BookWS application, right click the BookWS project and select Deploy as shown in the diagram.
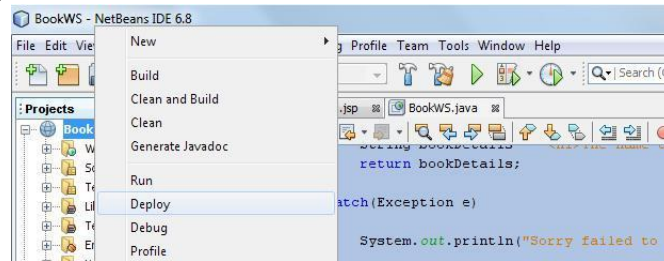
▫ The IDE starts the server, builds the application and deploys the application to the server.

▫ Follow the progress of these operations in the BookWS (run-deploy) and GlassFish v3 Domain tabs in the Output view.

▫ Now expand the web services directory of the BookWS project, right-click the BookWS Web service and select Test web service as shown in the diagram.
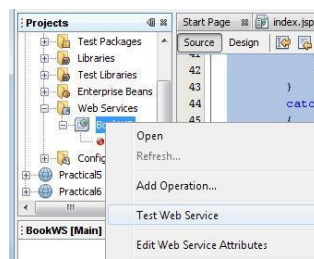
▫ The IDE opens the tester page in the web browser, if the web application is deployed using GlassFish server as shown in the figure.

☐ Enter the ISBN number as shown in the diagram.

☐ Click getBookDetails. The book name and its cost are displayed as shown in the diagram.

← → C 🔲 localhost:35637/BookWS/BookWSService?Tester

getBookDetails Method invocation

Method parameter(s)

Type	Value
java.lang.String	111-222-333

Method returned

java.lang.String : **"The name of the book is Learn My SQL and its cost is 250<."**

SOAP Request

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:getBookDetails xmlns:ns2="http://webservice/">
            <isbn>111-222-333</isbn>
        </ns2:getBookDetails>
    </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
```

6) **Consuming the web service**

⬚ Once the web service is deployed, the next most logical step is to create a client to make use of the web service's getBookDetails() method.

  i.   **Creating a web application**

⬚ To create a web application, select File -> New Project.

⬚ New project dialog box appears, select java web available under the categories section and web application available under the projects section. Click Finish.

⬚ New web application dialog box appears. Enter BookWSServletClient as the project name in the Project Name textbox and select the option Use Dedicated Folder for Storing Libraries.

⬚ Click Next. Server and settings section of the new web application, dialog box appears. Choose the default i.e. GlassFish v3 Domain as the web serevr, the Java EE 6 web as the Java EE version and the context path.

⬚ Click Finish.
⬚ The web application named BookWSServletClient is created.

  ii. **Adding the web service to the client application**

⬚ Right-click the BookWSServletClient project and select New -> Web Service Client as shown in the diagram.
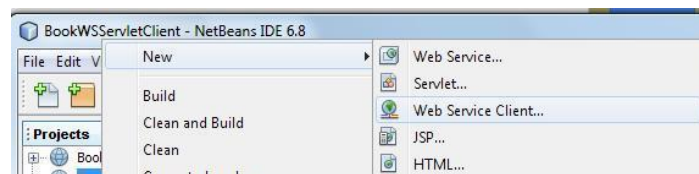
⏹ New Web Service Client dialog box appears. In the Project section, click Browse and browse through the web service which needs to be consumed. Click ok. The name of the web service appears in the New Web Service Client as shown in the diagram.
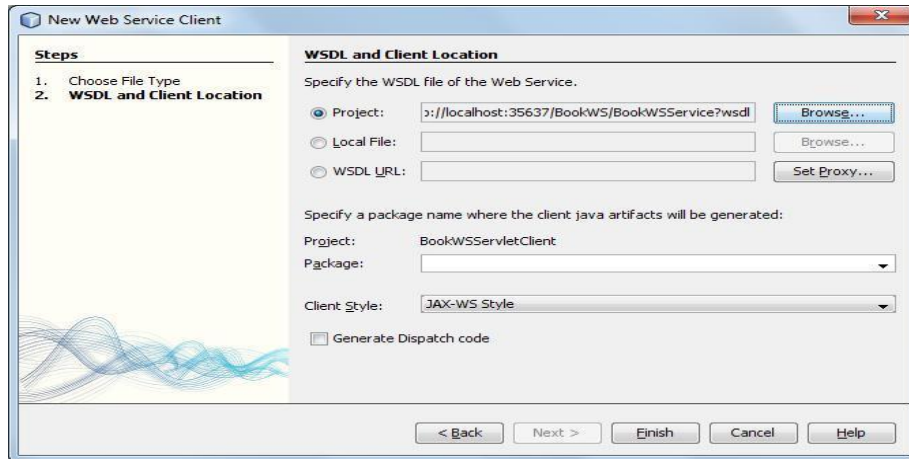
⏹ Leave the other settings as it is. Click Finish

⏹ The Web Service Reference directory is added to the BookWSServletClient application as shown in the diagram. It displays the structure of the newly created client including the getBookDetails() method created earlier.
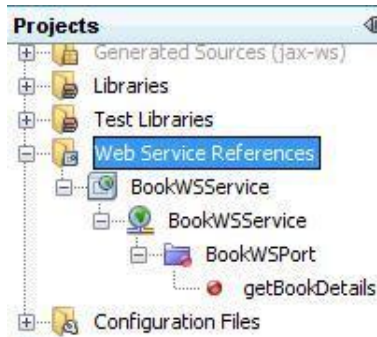
iii.   **Creating a servlet**

⬚ Create retreiveBookDetails.java using NetBeans IDE.
⬚ Right click source package directory, select New -> Servlet.

⬚ New Servlet dialog box appears. Enter retreiveBookDetails in the Class Name textbox and enter servlet in the package textbox.

⬚ Click Next. Configure Servlet Deployment section of the New Servlet dialog box appears. Keep the defaults.

⬚ Click Finish.
⬚ This creates the servlet named retreiveBookDetails.java in the servlet package.

⬚ retreiveBookDetails.java is available with the default skeleton created by the NetBeans IDE which needs to be modified to hold the application logic.

⬚ In the retreieveBookDetails.java source code, remove the following comments available in the body of the processRequest() method.

/*TODO output your page here*/

⬚ Replace the following code spec:

out.println("<h1>Servlet retreiveBookDetails at " + request.getContextPath () + "</h1>");

With the code spec of the getBookDetails() operation of the web service by dragging and dropping the getBookDetails operation as shown in the diagram.
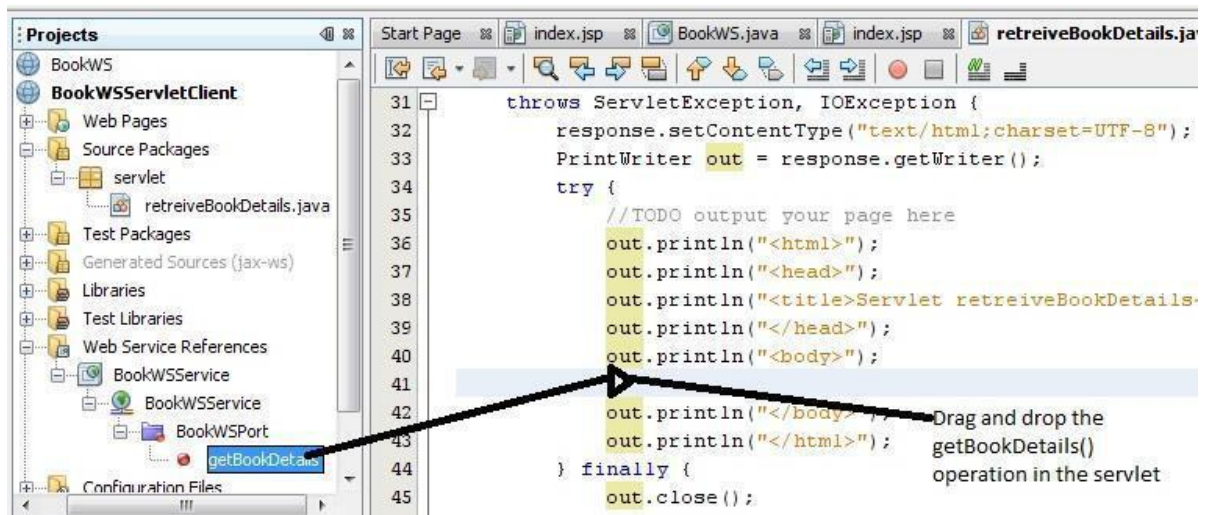
☐ The Servlet code spec changes as shown in the diagram

☐ The web service is instantiated by the @WebServiceRef annotation.

☐ Now change the following code spec:
java.lang.String isbn = "";
java.lang.String isbn = request.getParameter("isbn");

iv. **Creating an HTML form**

⬜ Once the web service is added and the servlet is created, the form to accept ISBN from the user needs to be coded.

⬜ Since NetBeans IDE by default [as a part of Web Application creation] makes available index.jsp file. Modify it to hold the following code spec.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%> <!DOCTYPE
HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

  "http://www.w3.org/TR/html4/loose.dtd">

<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>SOAP Cleint - Get Book Details</title>

  </head>

  <body bgcolor="pink">

    <form name="frmgetBookDetails" method="post" action="retreiveBookDetails">
      <h1>

        ISBN : <input type="text" name="isbn"/><br><br>
      </h1>

      <input type="submit" value="Submit"/>
    </form>

  </body>

</html>
```

v.   **Building the Web Application**

⬜ Build the web application.
⬜ Right click BookWSServletClient project and select Build.

⬜ Once the Build menu item is clicked the details about the compilation and building of the BookWSServletClient Web application appears in the output – BookWSServletClient (dist) window.

vi.   **Running the Application**

⬜ Once the compilation and building of the web application is done run the application.
⬜ Right click the BookWSServerCleint project and select run.

⬜ Once the run processing completes in NetBeans IDE a web browser is automatically launched and the BookWSServletCleint application is executed as shown in the diagram.

⬜ Enter the ISBN as shown in the diagram
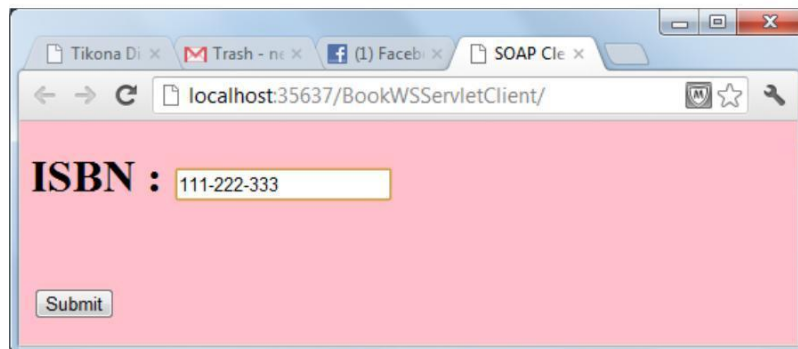
⬚ Click Submit. The book name and its cost are displayed as shown in the diagram.

Practical: 06

Aim: Implement Xen virtualization and manage with Xen Center

- Install **XenServer** in VMware Workstation and select Guest operating system as Linux.
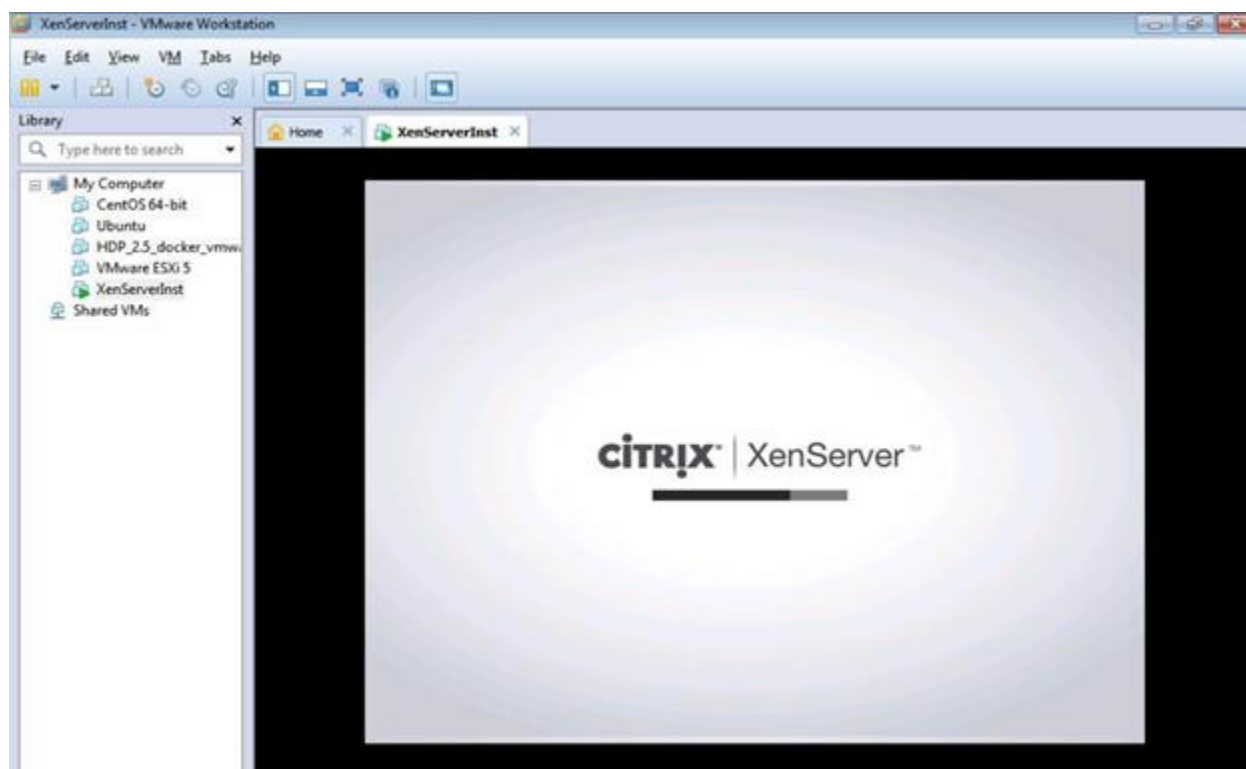
Note IP Address – " 192.168.124.137" ping it from command prompt.

- Now Install Citrix App if not installed
- Now Open Citrix XenCenter – and Click and **Add Server.**

- Fill IP address copied from Installation and User name as "root" and Password as "root123" which we had given during installation and Click on Add.
- Then click on Ok
- Now Click on New Storage
- Select Window File Sharing (CIFS) and click on next
- Uncheck Auto generate option Click on Next.
- Provide the path of shared windows XP image and enter local pc credential , click on Finish
- Click on New VM – and Windows XP SP3
- Select ISO file and click on next –
- Click on Next –
- Uncheck – Start the new VM and click on create now
- Now Right click on Windows XP and Start -
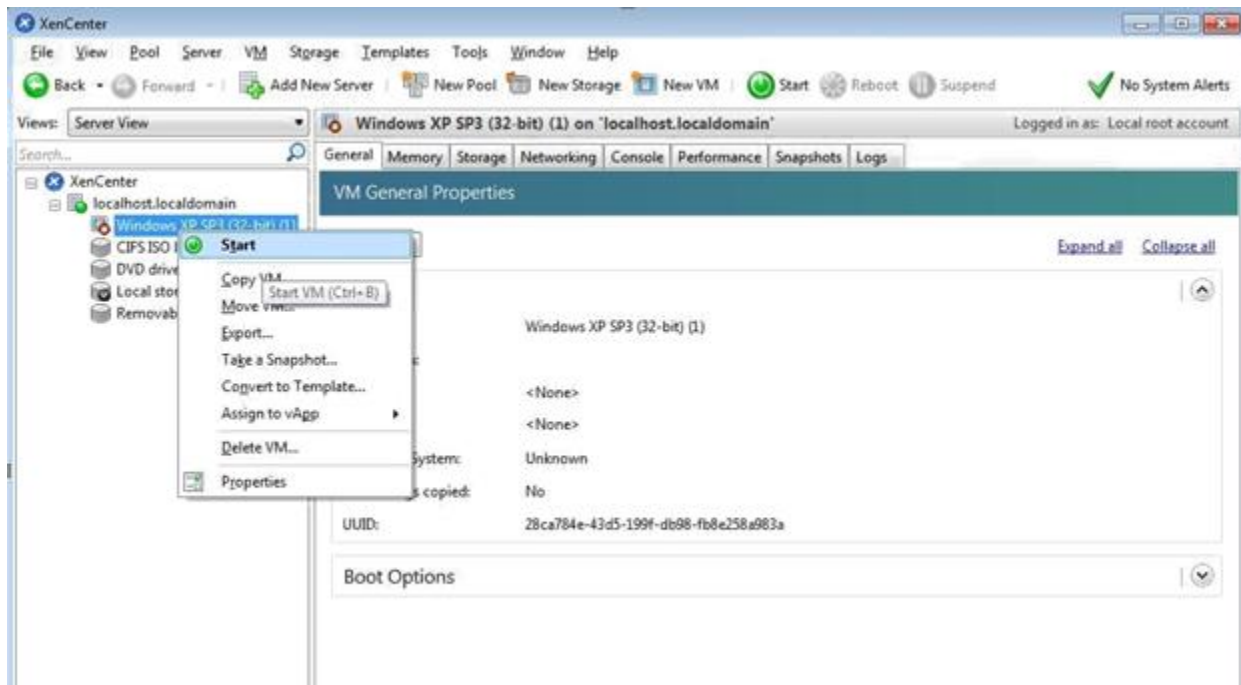
Installation is successful and virtual node has been created if we get below Welcome screen of Windows XP machine.

<div align="center">
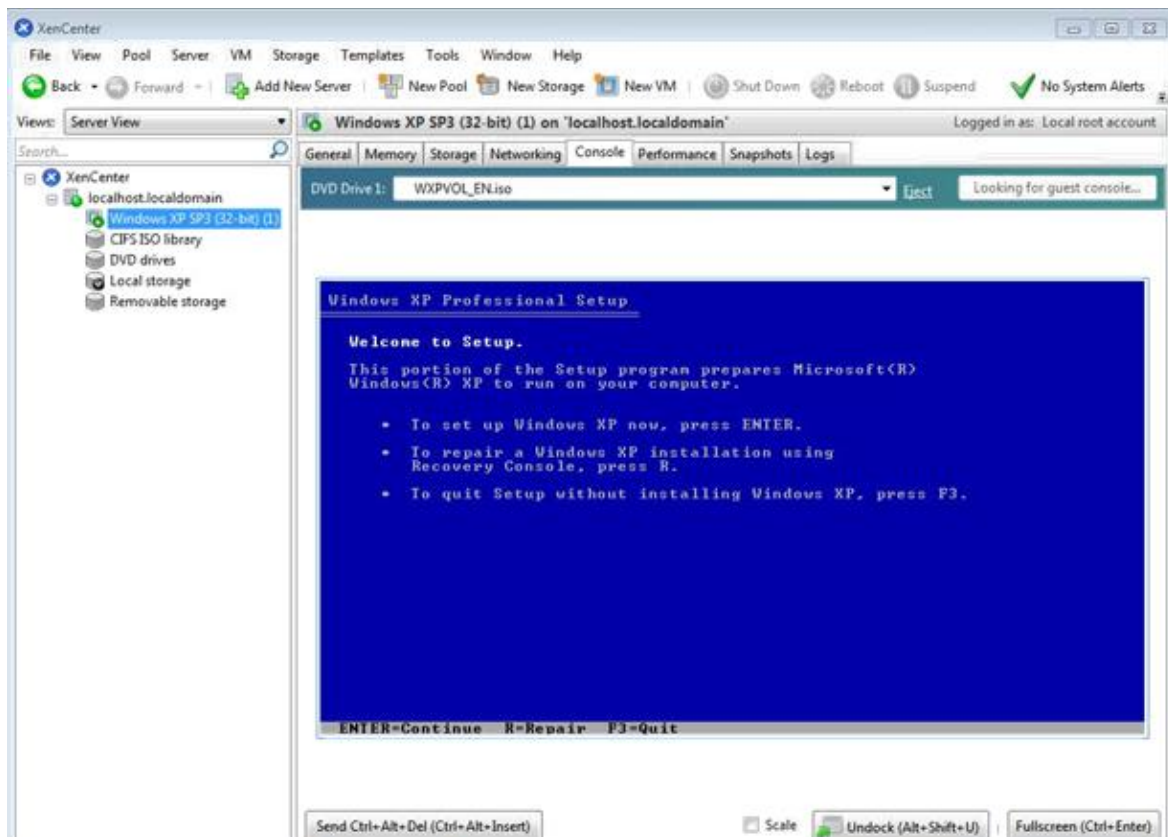
Practical: 07

</div>

Aim: Implement virtualization using VMWare ESXi Server and managing with vCenter

Steps:

Install **ESXi iso** in VMWare workstation.
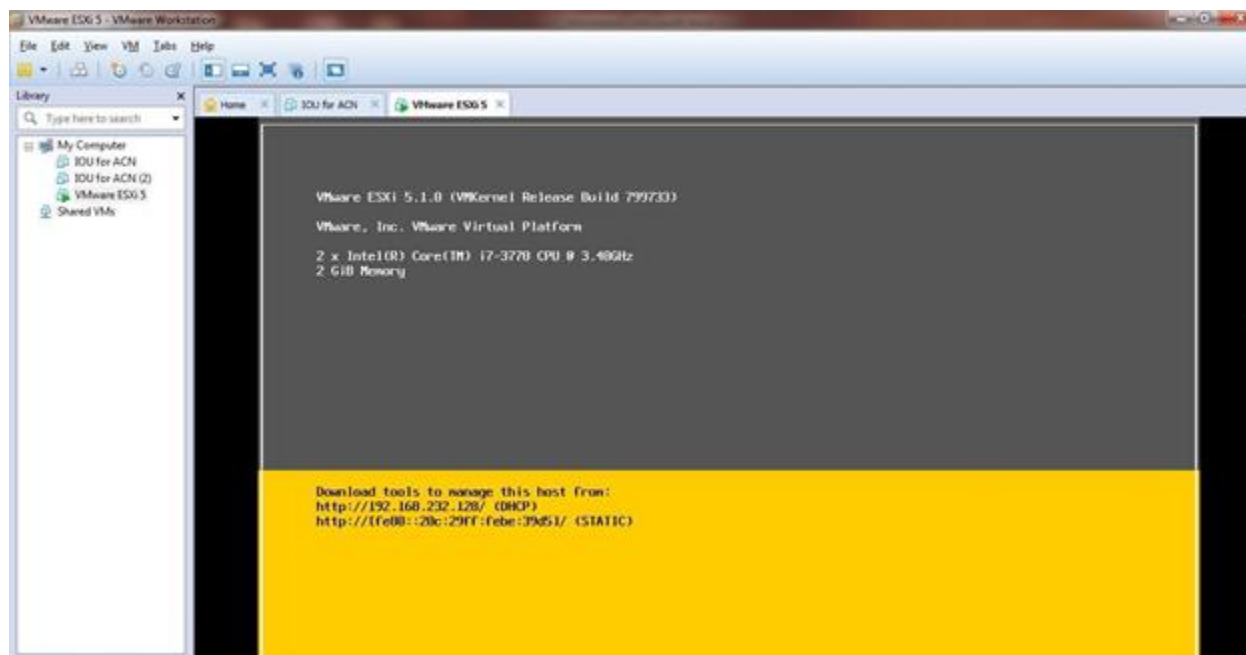
Install VMware vSphere Client

In vSphere create new **Virtual Machine.** Install Windows XP iso file and open it.

Practical: 08

Aim: Implement Windows Hyper V virtualization

First we have to uninstall vmware software if already installed on computer because the VMware Workstation installer does not support running on a Hyper-V virtual machine. after uninstalling vmware we can proceed to next step go to control panel and click on uninstall a program.
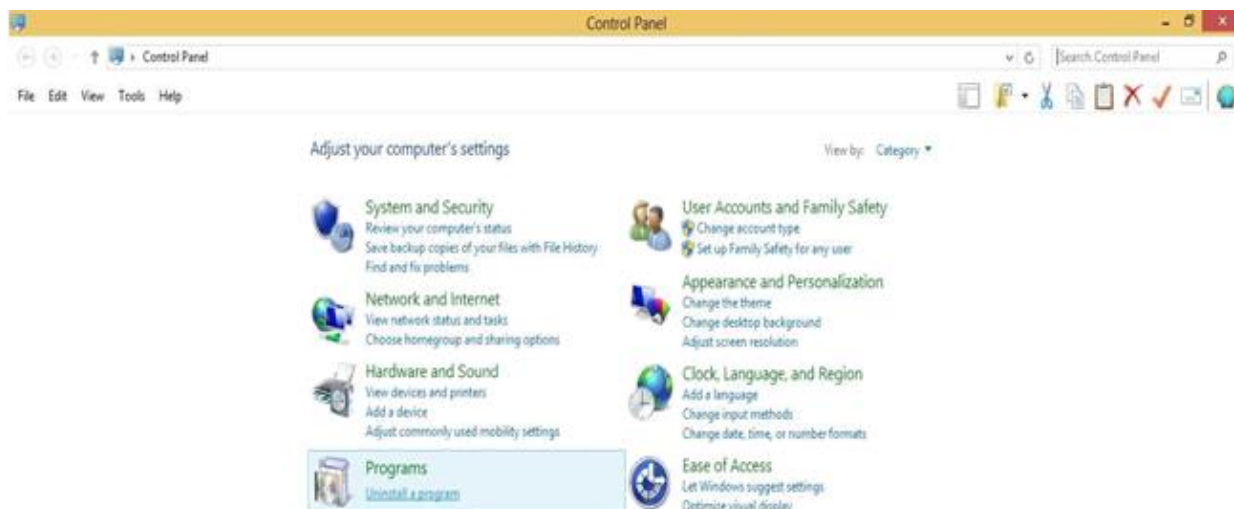
Click on Turn windows features on or off.

Now in windows features check on Hyper-V option.
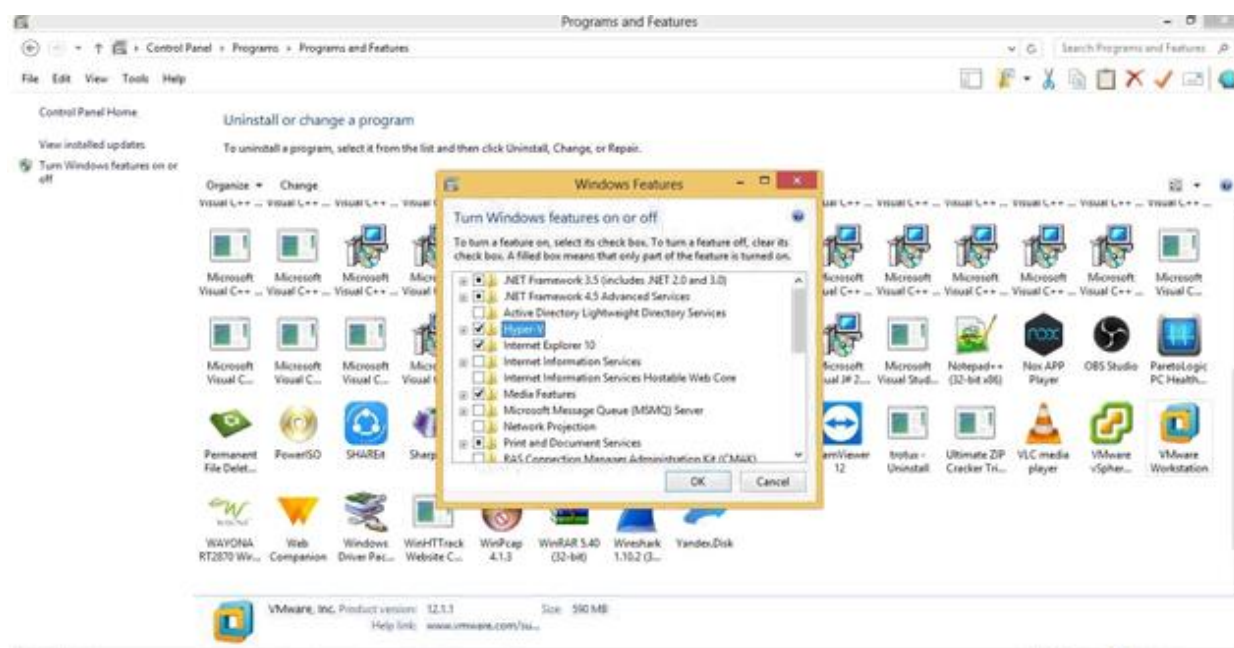
After Restart **Search for hyper-v manager in search box and click on that.**
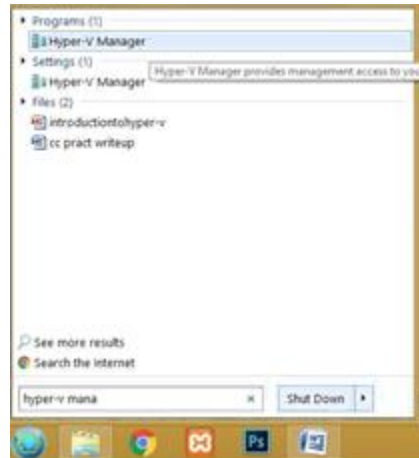
for creating virtual machine first we have to create virtual switch click on virtual switch manager option.
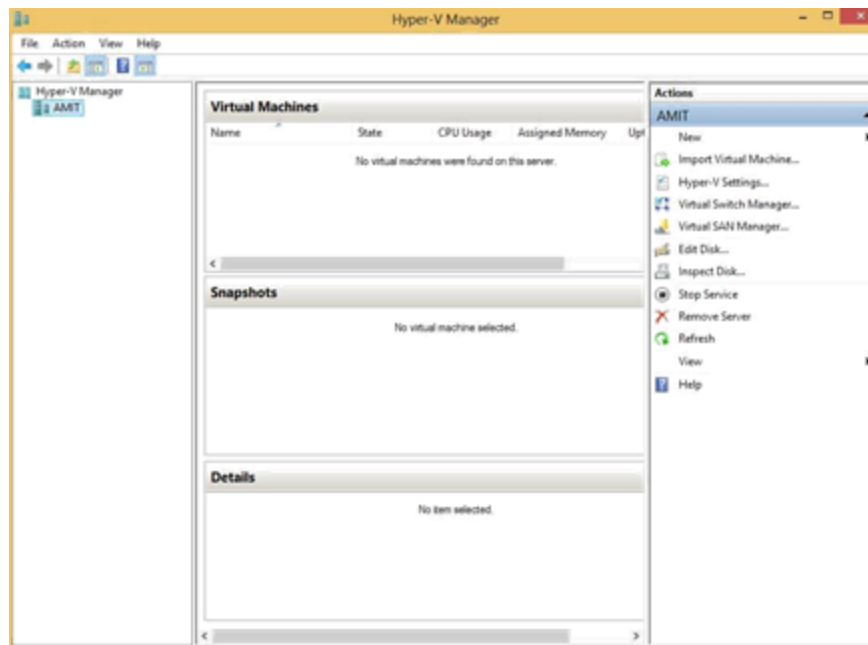
Select External as a connection type and then click on create virtual switch.

Create new Virtual switch and install windows XP .iso
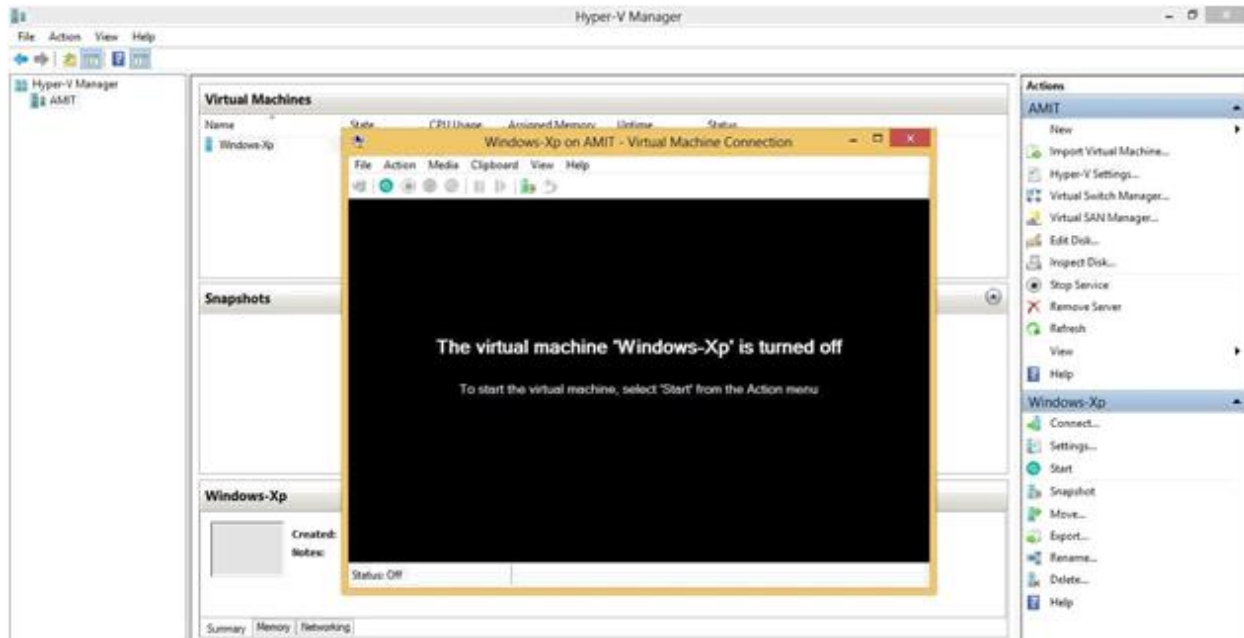
and virtual machine will start.
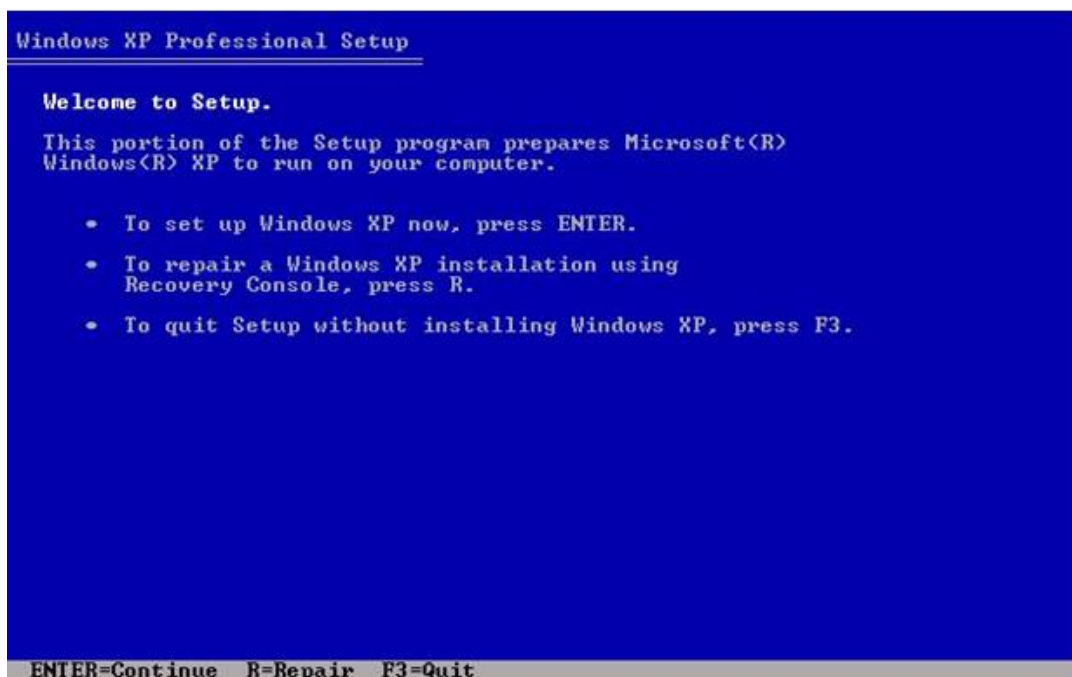
Practical: 09

 Aim: Develop application for Microsoft Azure.

Step 1:

To develop an application for Windows Azure on Visual Studio install the
"Microsoft Azure SDK for .NET (VS 2010) – 2.8.2.1"

Step2:

Turn windows Features ON or OFF:

Go to Control panel and click on programs.

Turn Windows features on or off.

Step3:

Now, Start the visual studio 2010 and Go To File->New->Project

Expand Visual C#-> Select Cloud
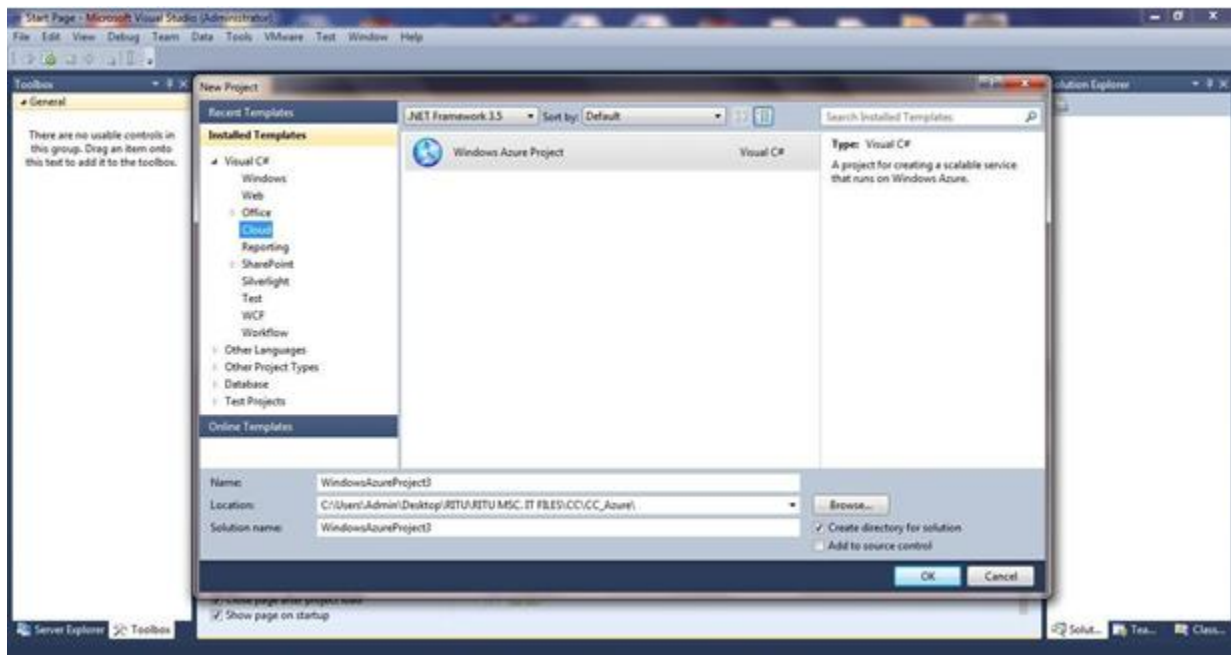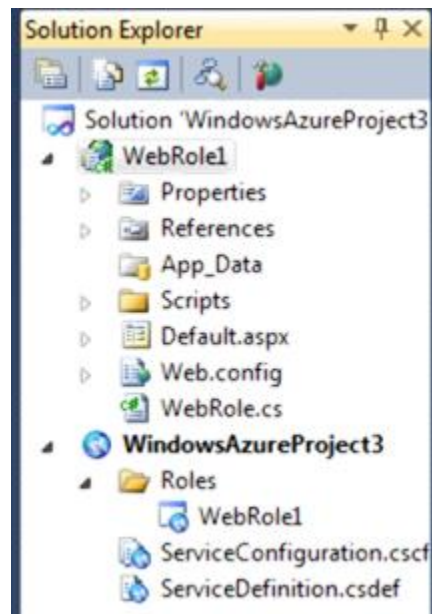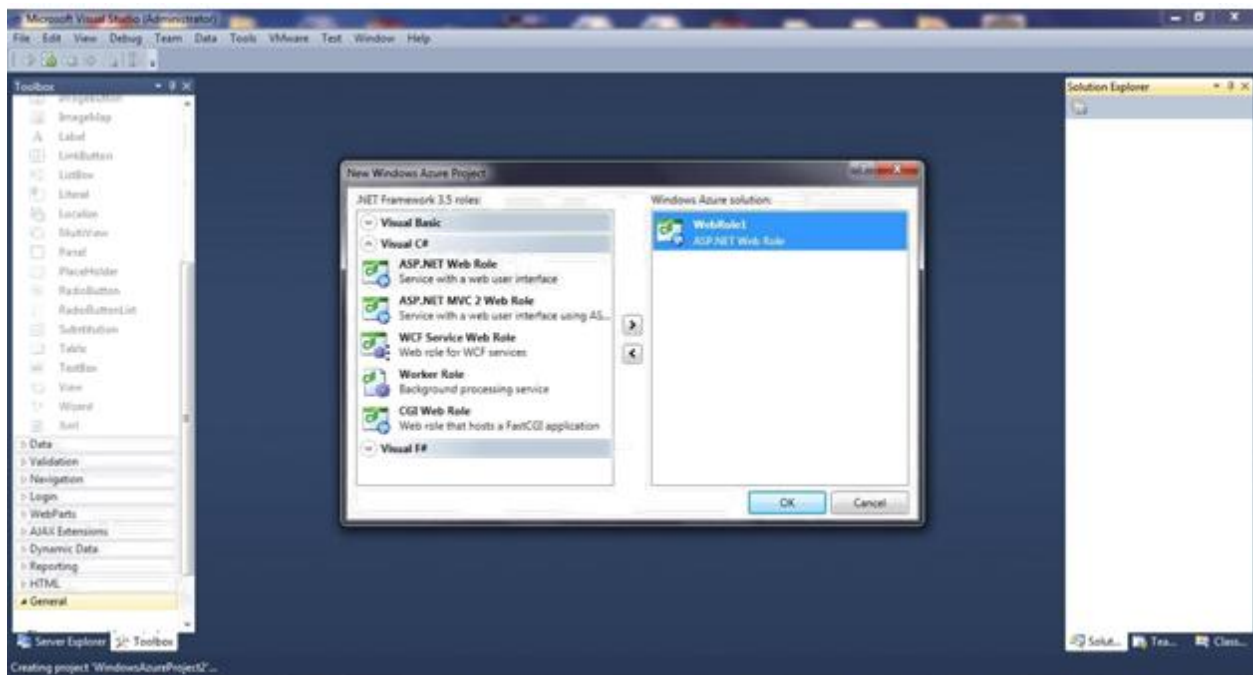
Right Click on WebRole1>>ADD>>New Item
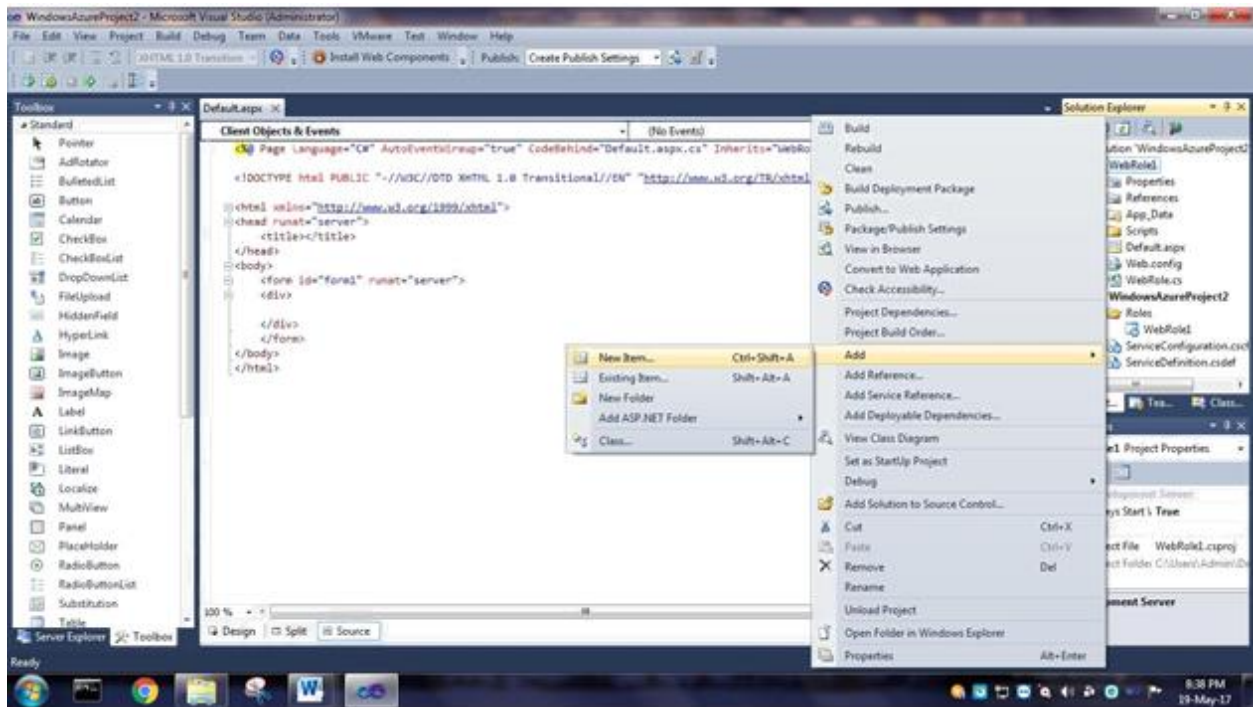
Add a New web Form. Give it a name. Click Add
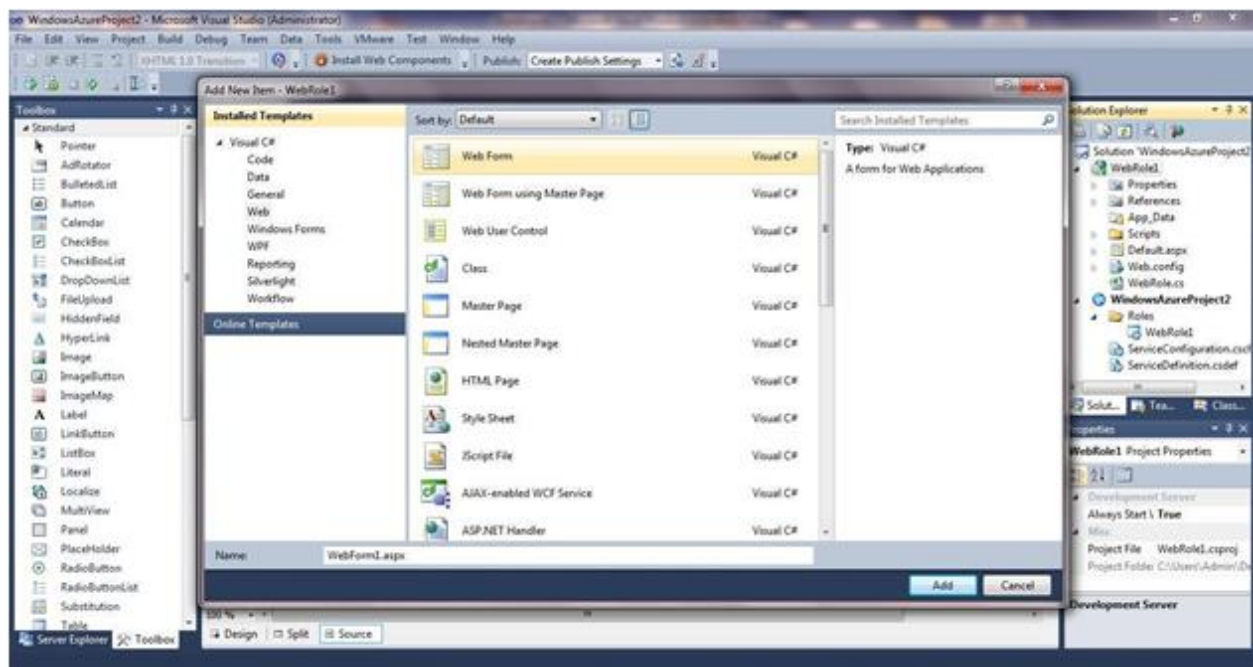

Deploy the project:
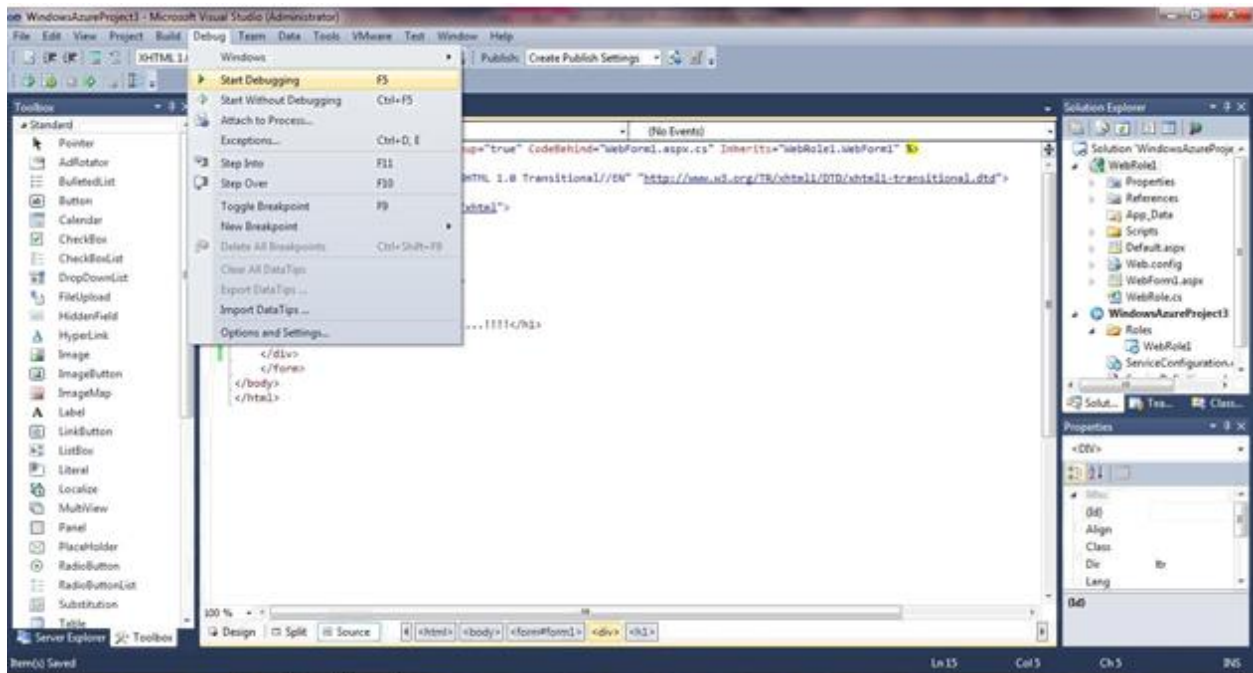
Run Project
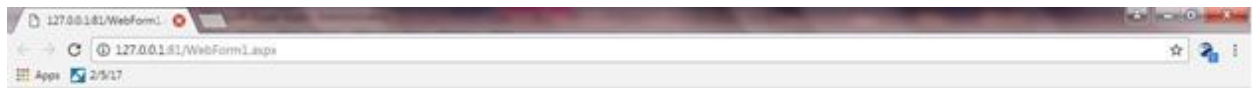
Practical: 10

Aim: Develop application for Google App Engine

- Open Eclipse Luna. Go to **Help Menu Install New Software…**
- In **Install** window Click on the **"Add"** button besides the **Work with** textbox. **Add Repository** window appears. Enter the **Location** as **"https://dl.google.com/eclipse/plugin/4.4"** and click on "**OK**" button.
- From the available softwares select the required softwares and tools as shown in the below image for the **GAE**. Then click on the **"Next"** button.
- In the **Install Details** window click on **"Next"** button.

- In the Next Window "Review the Items to be Installed" then click on **"Next"**

- In the next window for Review Licenses select the option **"I accept……"** and click on **"Finish"**
- button.
- After Installation you will get option to **"Restart Eclipse",** click on **Yes.** So that the software you selected gets updated…

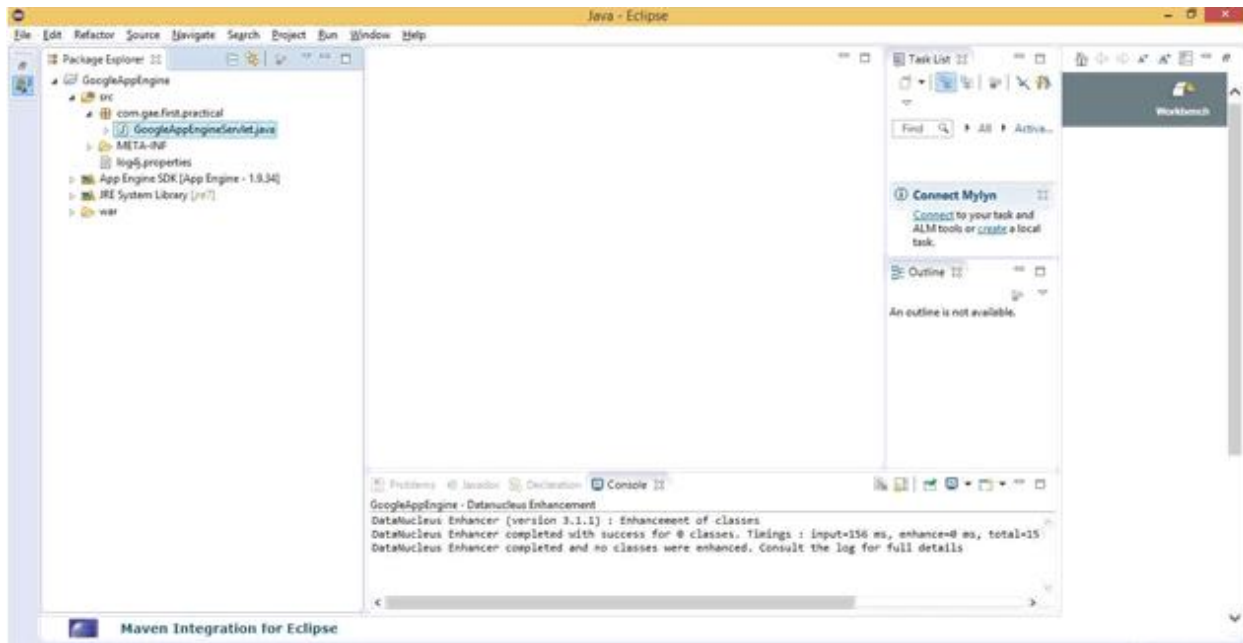- Now, go to **File Menu_New_Other.**
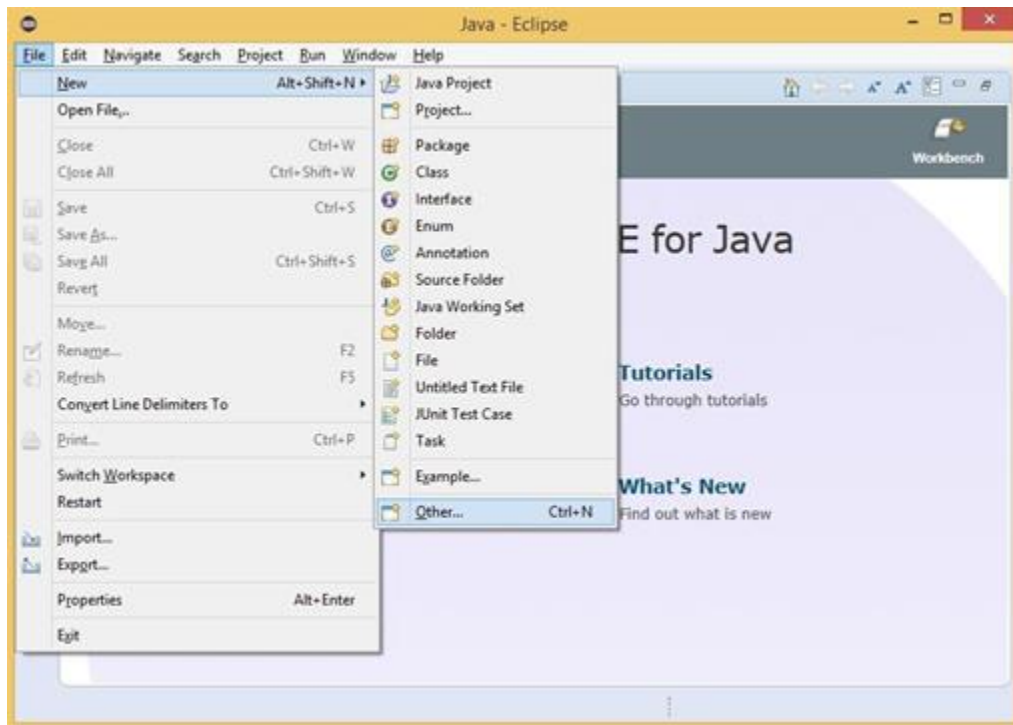
In the New window select **Google_Web Application Project** and click on **"Next"** button.
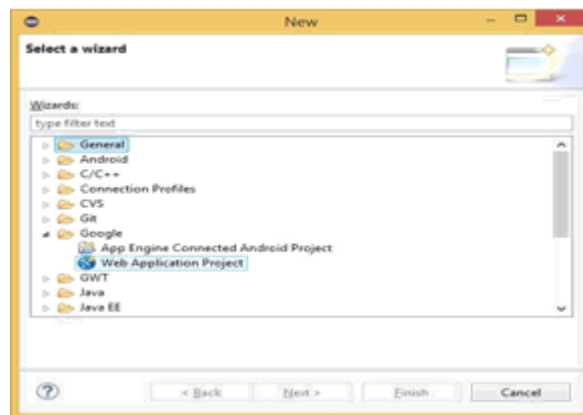

Enter the details for the new Web application project. Deselect the **Use Google Web Toolkit** option under the section **Google SDKs**. Click on the **"Finish"** button.
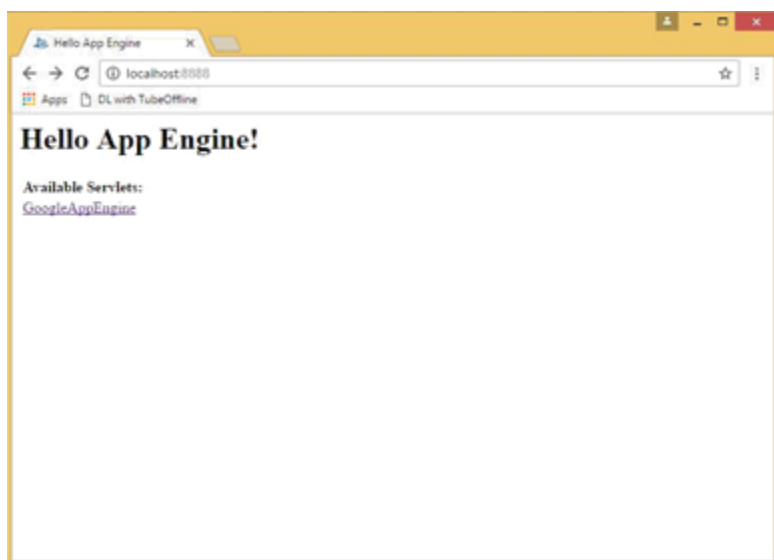
From the **Package Explorer** open the **.java** file (Here it is **"Google_App_EngineServlet.java"**).

Edit the file as required (Unedited file too can be used. Here the editing is done to "what should be displayed" on the browser). **Save** the file. Click on the **Run** option available on the Tools bar.
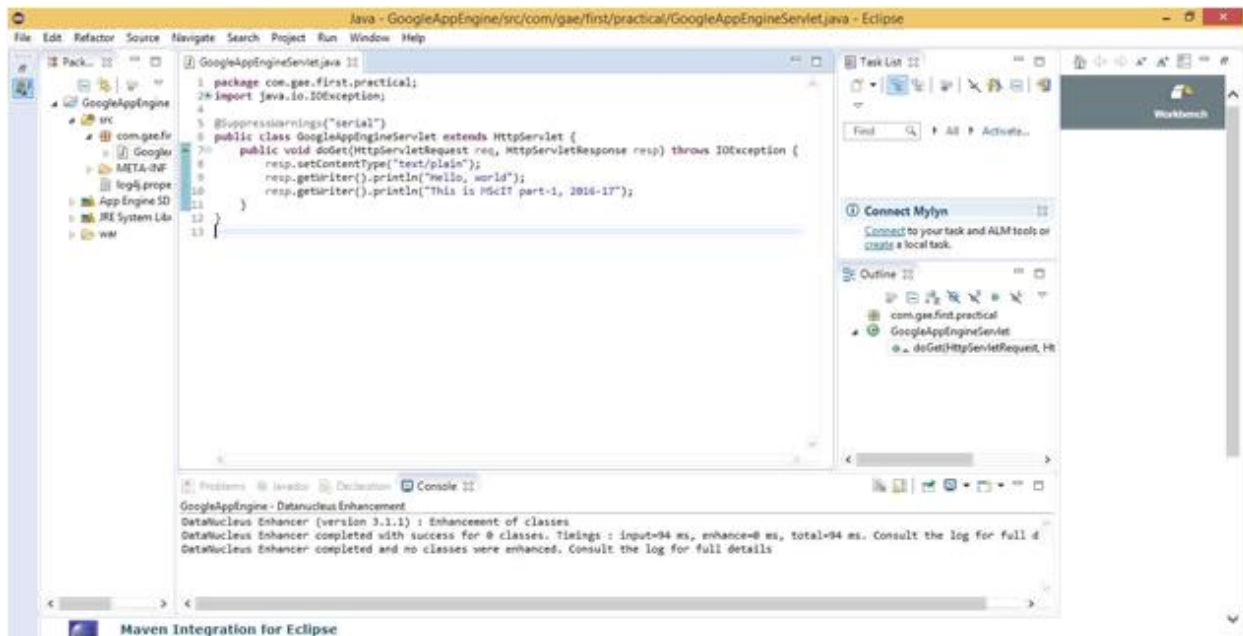
In the browser (Here, Google Chrome) type the address as **"localhost:8888" which is "Default"**.

In **localhost:8888** the link to the **Google_App_EngineServlet.java** file as **Google_App_Engine** is displayed. Click on this link. It will direct you to **"localhost:8888/Google_App_Engine"**.
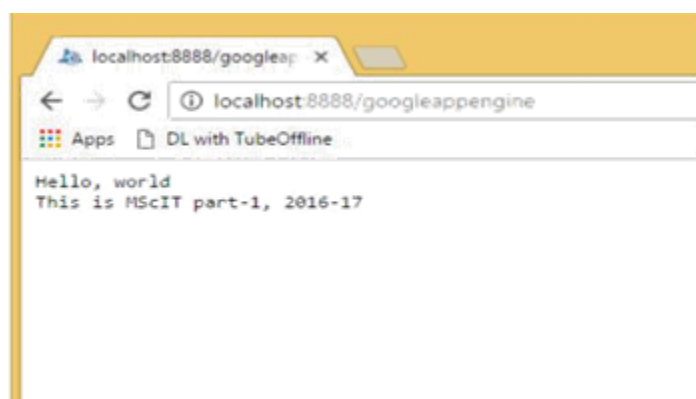
The **output text entered** in the **java** program is **displayed as the output** when clicked the link "Google_App_Engine".