

Per la comprensione del presente articolo è necessario aver letto e studiato l'articolo "Ping (in php)", http://www.maurziocozzetto.net/pdf/ping_php.pdf

Riscriviamo ora l'applicativo che fa il ping verso una macchina in JSP. Ovviamente seguiremo un approccio "Object Oriented". Scriveremo dapprima una versione "standard" del ping, con il codice java "immerso" nell'HTML. Poi una versione che sfrutta le *JSTL, Java Standard Tag Library*, delle note librerie java basate su XML progettate da *Sang Shin* (<http://www.javapassion.com>). Procediamo.

Scriveremo dapprima la classe *Ping*. Il codice della classe non presenta particolari problemi. Ecco qui di seguito il codice:

```
/*
 * Ping.java
 */

package it.itiscastelli.classi;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author maurizio
 */
public class Ping {
    String host;
    int counter;

    public Ping(String host, int counter) {
        this.host=host;
        this.counter=counter;
    }

    public boolean hostRaggiungibile() throws UnknownHostException, IOException {
        // se viene passato il nome di un host, getByName non fa nulla
        // altrimenti se viene passato un ip, getByName restituisce cmq il nome di un host
        InetAddress address = InetAddress.getByAddress(host);

        return address.isReachable(2000);
    }

    private String sistemaOperativo() {
        return System.getProperty("os.name");
    }

    public List<String> execPing() throws IOException {
        String sistema = this.sistemaOperativo();
        // l'oggetto Runtime rappresenta l'ambiente di esecuzione
        Runtime r = Runtime.getRuntime();
        // Process rappresenta il processo corrente
        Process p=null;
    }
}
```

```
    if (sistema.equals("Linux"))
        p = r.exec("ping -c "+this.counter+" "+this.host);
    else if (sistema.equals("Windows"))
        p = r.exec("ping -n "+this.counter+" "+this.host);

    // struttura dati che accoglie le righe risultato
    List<String> listaMessaggi = new ArrayList<String>();

    BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));

    // leggo una riga
    String inputLine=in.readLine();

    // continuo a leggere fino alla fine del file
    while (inputLine != null) {
        // memorizzo le righe nella struttura dati
        listaMessaggi.add(inputLine);
        // leggo un'altra riga
        inputLine=in.readLine();
    }

    // chiudo lo stream
    in.close();

    return listaMessaggi;
} // fine metodo

} // fine classe
```

Prima di scrivere l'applicativo, testiamo i metodi della classe scrivendo un applicativo Console Java. Noteremo un comportamento "insolito" del metodo *hostRaggiungibile()*.

Riportiamo il codice della classe Main:

```
/*
 * Main.java
 */

package it.mauriziocozzetto.main;

import it.mauriziocozzetto.classes.Ping;
import java.io.IOException;
import java.net.UnknownHostException;
import java.util.List;

/**
 *
 * @author maurizio
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws UnknownHostException, IOException {
        Ping p = new Ping("www.mauriziocozzetto.it",10);
        System.out.println(p.hostRaggiungibile());
    }
}
```

```
List<String> messaggi = p.execPing();

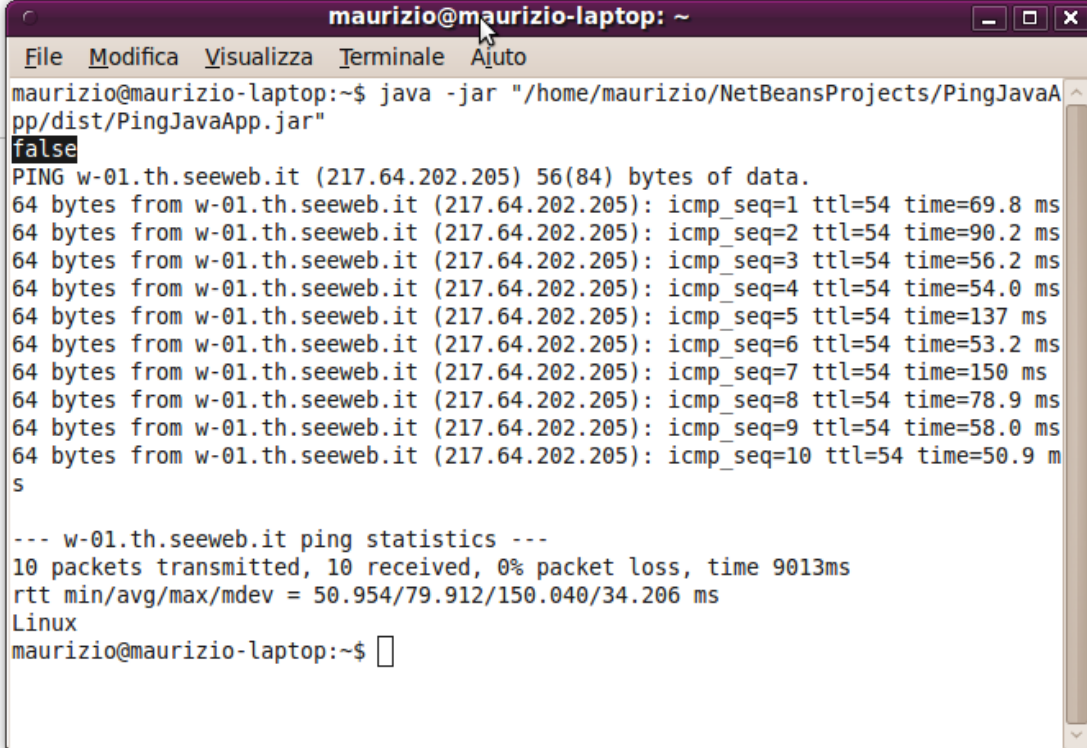
if (messaggi.size()==0)
    System.out.println("Non ci sono messaggi");
else
    for (String s : messaggi)
        System.out.println(s);

System.out.println(System.getProperty("os.name"));

}

} //
```

Se lanciamo l'applicativo (per esempio da shell), otteniamo, come risultato dell'esecuzione del metodo *hostRaggiungibile()*, il valore **false** e questo è piuttosto "strano", visto e considerato che si ottengono i risultati del ping.

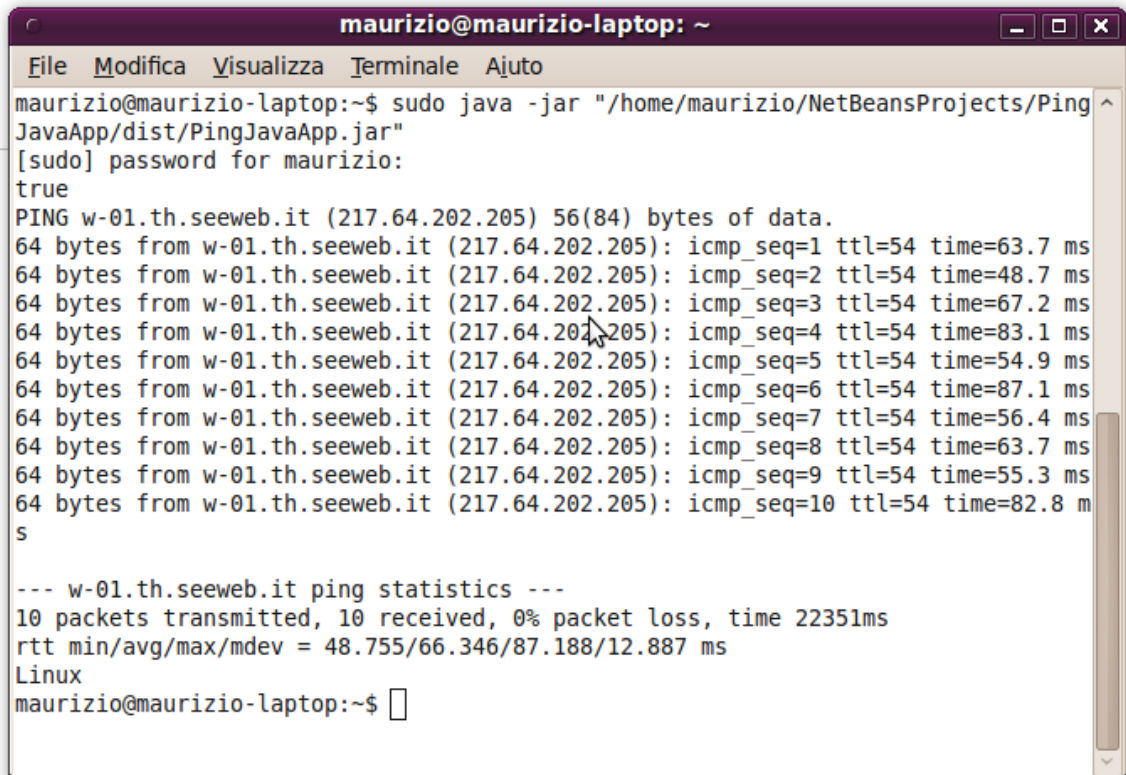


```
maurizio@maurizio-laptop: ~
File Modifica Visualizza Terminale Aiuto
maurizio@maurizio-laptop:~$ java -jar "/home/maurizio/NetBeansProjects/PingJavaApp/dist/PingJavaApp.jar"
false
PING w-01.th.seeweb.it (217.64.202.205) 56(84) bytes of data.
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=1 ttl=54 time=69.8 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=2 ttl=54 time=90.2 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=3 ttl=54 time=56.2 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=4 ttl=54 time=54.0 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=5 ttl=54 time=137 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=6 ttl=54 time=53.2 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=7 ttl=54 time=150 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=8 ttl=54 time=78.9 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=9 ttl=54 time=58.0 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=10 ttl=54 time=50.9 ms
s
--- w-01.th.seeweb.it ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 50.954/79.912/150.040/34.206 ms
Linux
maurizio@maurizio-laptop:~$
```

Fig. 1 - Risultato dell'esecuzione del programma PingJavaApp

Il motivo è il seguente: occorre lanciare il programma con **privilegi di amministrazione** (in Ubuntu si può premettere al comando la parola riservata *sudo* che sta per *super-user do*).

Fatta questa prima prova, scriviamo ora la nostra Web Application Java. L'applicativo (*PingWebApp*) è composto da *due* pagine: la *prima* pagina è esattamente la stessa del programma *ping_php* (cioè la pagina statica *inserisciURL.html*), la seconda pagina (*ping.jsp*) invece è dinamica e sfrutta la classe *Ping*.



```

maurizio@maurizio-laptop: ~
File Modifica Visualizza Terminale Ajuto
maurizio@maurizio-laptop:~$ sudo java -jar "/home/maurizio/NetBeansProjects/Ping
JavaApp/dist/PingJavaApp.jar"
[sudo] password for maurizio:
true
PING w-01.th.seeweb.it (217.64.202.205) 56(84) bytes of data.
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=1 ttl=54 time=63.7 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=2 ttl=54 time=48.7 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=3 ttl=54 time=67.2 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=4 ttl=54 time=83.1 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=5 ttl=54 time=54.9 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=6 ttl=54 time=87.1 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=7 ttl=54 time=56.4 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=8 ttl=54 time=63.7 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=9 ttl=54 time=55.3 ms
64 bytes from w-01.th.seeweb.it (217.64.202.205): icmp_seq=10 ttl=54 time=82.8 m
s
--- w-01.th.seeweb.it ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 22351ms
rtt min/avg/max/mdev = 48.755/66.346/87.188/12.887 ms
Linux
maurizio@maurizio-laptop:~$

```

Fig. 2 - Risultato dell'esecuzione del programma con privilegi di amministrazione.

File inserisciURL.html

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Applicativo PingWebApp</title>
<style>
#blocco {
margin-top:40px;
margin-left:auto;
margin-right:auto;
width:480px;
}
p {
margin-top:0px;
margin-bottom:0px;
}
</style>
</head>
<body>
<div id="blocco">
<fieldset>
<legend>Ping</legend>
<form name="frmInserisci" action="ping.jsp" method="GET">
<p> Indirizzo IP o URL <input type="text" name="txtIp"/></p>
<p> Numero richieste ICMP <input type="text" name="txtCounter"/></p>
<p><input type="submit" name="btnSubmit" value="Invia" /> </p>
</form>
</fieldset>
</div>
</body></html>

```

File *ping.jsp*

```
<%--
  Document   : ping
  Created on : 8-lug-2009, 15.45.06
  Author    : maurizio
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="it.itiscastelli.classi.Ping" %>
<%@page import="java.util.List" %>

<!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>Ping in JSP</title>
    <style>
      #blocco {
        margin-top:40px;
        margin-left:auto;
        margin-right:auto;
        width:800px;
      }
      p {
        margin-top:0px;
        margin-bottom:0px;
      }
    </style>
  </head>
  <body>

    <div id="blocco">
      <fieldset>
        <legend>Risultati</legend>
        <%
          if (request.getParameter("btnSubmit")!=null) {
            String host=request.getParameter("txtIp");

            if (host==null || host.equals(""))
              host="www.mauriziocozzetto.net";

            int counter=0;

            try {
              counter = Integer.parseInt(request.getParameter("txtCounter"););
            } catch(Exception e) {
              counter=4;
            }

            Ping p = new Ping(host,counter);

            List<String> messaggi = p.execPing();

            if (messaggi.size()==0)
              out.println("Non ci sono messaggi (probabilmente l'host "+host+" è
irraggiungibile o il nome è errato o il numero di richieste è nullo o non è un valore valido).");
```

```
        else
            for (String s : messaggi)
                out.println(s+"<br/>");
    } else {
        response.sendRedirect("inserisciURL.html");
    }

    %>
</fieldset>
</div>

</body>
</html>
```

Se non vengono inseriti i dati richiesti nelle due caselle testuali, il programma assume due valori di default: www.mauriziocozzetto.net per la casella *txtIp* e 4 per la casella *txtCounter*.

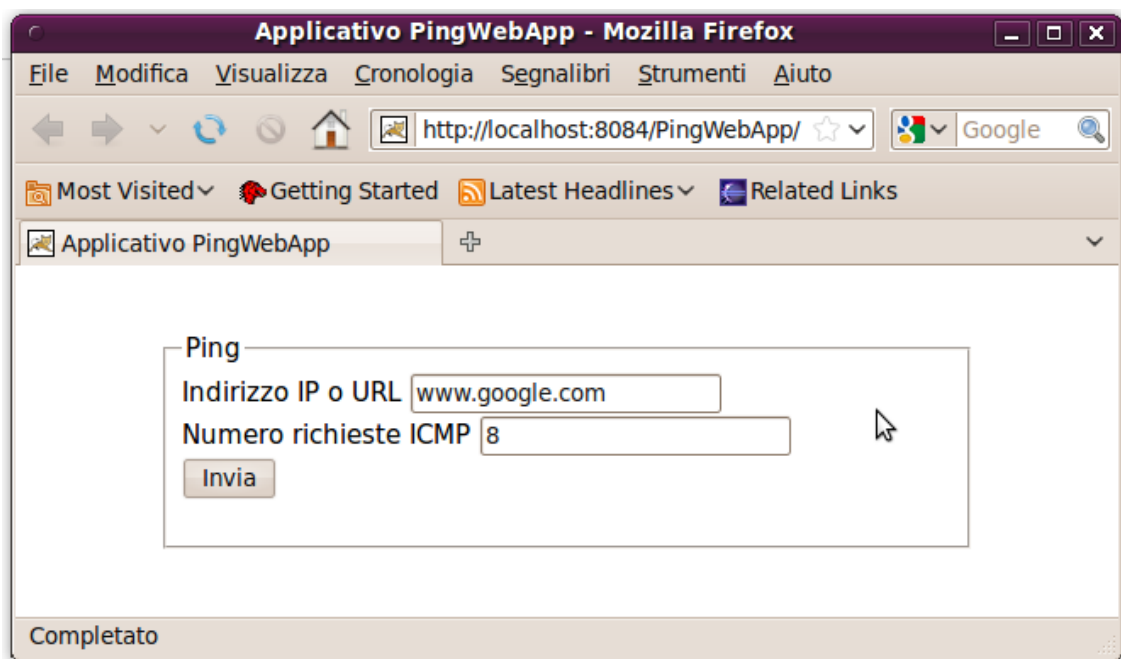
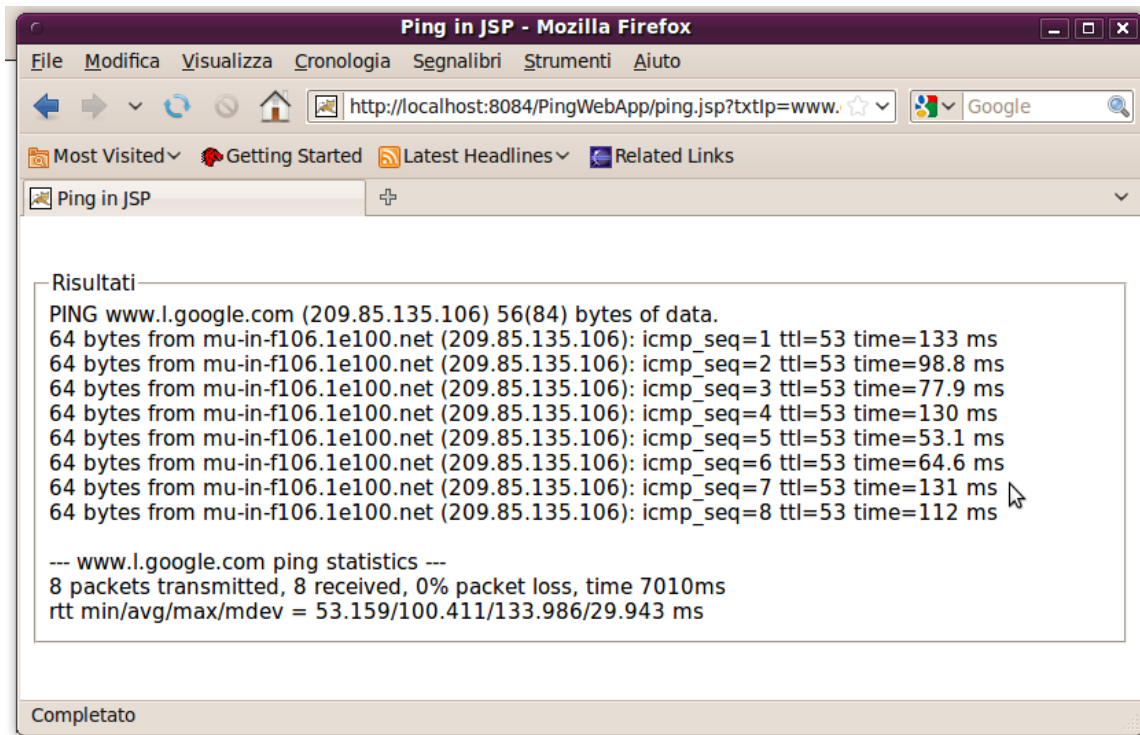


Fig. 3 - Pagina *inserisciURL.html*

Fig. 4 - Pagina *ping.jsp*

Riprogettiamo ora l'applicativo usando le *JSTL*. Il nuovo applicativo *PingJSTLWebApp* questa volta presenta delle importanti differenze col progetto *PingWebApp*. Innanzitutto la classe *Ping* va riprogettata in modo da poterla considerare un *JavaBean*.

Definizione di *JavaBean*

Ricordo che un *JavaBean* è una "normale" classe Java con le seguenti proprietà: la classe deve implementare *java.io.Serializable* cioè la classe è *persistente* (l'interfaccia *java.io.Serializable* non ha metodi), deve disporre almeno del costruttore *nulla* (senza argomenti), i suoi attributi devono essere *privati*, e per ogni attributo deve esistere un *getter* e un *setter*. I nomi dei metodi (che *non* restituiscono valori booleani) devono avere il prefisso **get**, mentre i metodi che restituiscono valori booleani devono avere il prefisso **is** (altrimenti non si possono usare nelle pagine *JSTL*).

La classe *Ping* diventa allora:

```

/*
 * Ping.java
 */

package it.itiscastelli.classi;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author maurizio
 */

```

```
public class Ping implements java.io.Serializable {
    private String host;
    private int counter;

    // il costruttore di default (senza argomenti) DEVE essere presente
    // pena un errore in fase di esecuzione
    public Ping() {
        //
    }

    public Ping(String host, int counter) {
        this.host=host;
        this.counter=counter;
    }

    // per i metodi che restituiscono un boolean è OBBLIGATORIA la parola "is"
    public boolean isHostRaggiungibile() throws UnknownHostException, IOException {
        // se viene passato il nome di un host, getByName non fa nulla
        // altrimenti se viene passato un ip, getByName restituisce cmq il nome di un host
        InetAddress address = InetAddress.getByHost(getHost());

        return address.isReachable(2000);
    }

    /**
     * @return the host
     */
    public String getHost() {
        return host;
    }

    /**
     * @param host the host to set
     */
    public void setHost(String host) {
        this.host = host;
    }

    private String sistemaOperativo() {
        return System.getProperty("os.name");
    }

    // i metodi DEVONO avere la parola "get"
    // ma in JSTL si richiamano senza
    public List<String> getExecPing() throws IOException {
        String sistema = this.sistemaOperativo();

        Runtime r = Runtime.getRuntime();

        Process p=null;

        if (sistema.equals("Linux"))
            p = r.exec("ping -c "+this.getCounter()+" "+this.host);
        else if (sistema.equals("Windows"))
            p = r.exec("ping -n "+this.getCounter()+" "+this.host);

        // struttura dati che accoglie le righe risultato
        List<String> listaMessaggi = new ArrayList<String>();
    }
}
```

```
BufferedReader in = new BufferedReader(new InputStreamReader(p.getInputStream()));

// leggo una riga
String inputLine=in.readLine();

// continuo a leggere fino alla fine del file
while (inputLine != null) {
    // memorizzo le righe nella struttura dati
    listaMessaggi.add(inputLine);
    // leggo un'altra riga
    inputLine=in.readLine();
}

// chiudo lo stream
in.close();

return listaMessaggi;
}

/**
 * @return the counter
 */
public int getCounter() {
    return counter;
}

/**
 * @param counter the counter to set
 */
public void setCounter(int counter) {
    this.counter = counter;
}
}
```

Ricordiamoci poi di includere nel progetto NetBeans le librerie *JSTL*.

Segue per brevità, visto che la pagina *inserisciURL.html* è la stessa, codice della pagina *ping.jsp*.

File ping.jsp

```
<%--
  Document   : ping
  Created on : 8-lug-2009, 15.45.06
  Author    : maurizio
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!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>Ping in JSTL</title>
```

```

<style>
  #blocco {
    margin-top:40px;
    margin-left:auto;
    margin-right:auto;
    width:800px;
  }
  p {
    margin-top:0px;
    margin-bottom:0px;
  }
</style>
</head>
<body>

<div id="blocco">
  <fieldset>
    <legend>Risultati</legend>
    <c:choose>
      <c:when test="{!empty param.btnSubmit}">
        <c:set var="host" value="{param.txtIp}" />

        <c:if test="{empty host}">
          <c:set var="host" value="www.mauriziocozzetto.net" />
        </c:if>

        <c:set var="counter" value="{param.txtCounter}" />

        <c:if test="{empty counter}">
          <c:set var="counter" value="4" />
        </c:if>

        <jsp:useBean id="p" class="it.itiscastelli.classi.Ping" />

        <jsp:setProperty name="p" property="host" value="{host}" />

        <c:catch var="ex">
          <jsp:setProperty name="p" property="counter" value="{counter}" />
        </c:catch>

        <c:if test="{ex!=null}">
          <jsp:setProperty name="p" property="counter" value="4" />
        </c:if>

        <c:if test="{empty p.execPing}">
          <c:out value="Non ci sono messaggi (probabilmente l'host {host} è
irraggiungibile o il nome è errato o il numero di richieste è nullo o non è un valore valido)." />
        </c:if>

        <c:forEach var="s" items="{p.execPing}">
          <c:out value="{s}" /><br/>
        </c:forEach>

      </c:when>
      <c:otherwise>
        <jsp:forward page="inserisciURL.html" />
      </c:otherwise>
    </c:choose>

```

```

</fieldset>
</div>
</body></html>

```

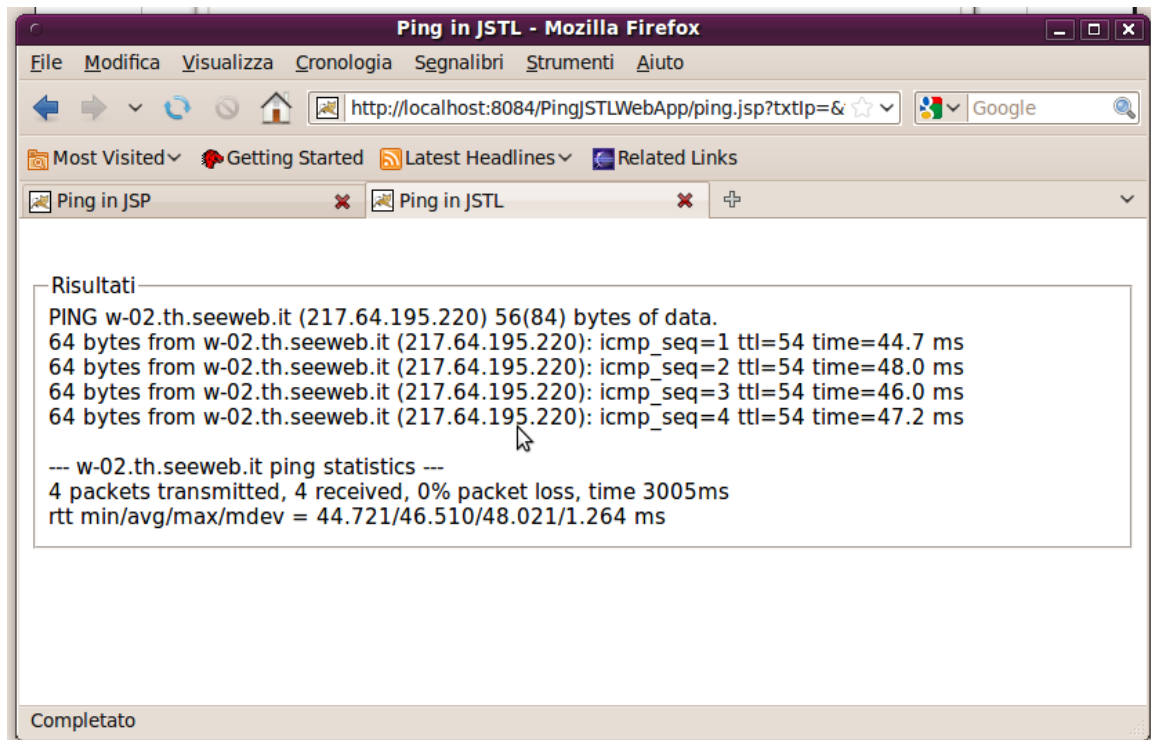


Fig. 5 - Risultato dell'esecuzione del Ping

La direttiva `<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>` consente di usare la libreria **core** delle JSTL (che contiene tutte le **strutture di controllo** fondamentali). Per un approfondimento delle JSTL, si leggano gli articoli:

<http://www.mauriziocozzetto.it/wp/archives/232>,
<http://www.mauriziocozzetto.it/wp/archives/235>,
<http://www.mauriziocozzetto.it/wp/archives/236>.

L'oggetto **param** è un oggetto predefinito ed è l'equivalente dell'oggetto **request**.

```
<jsp:useBean id="p" class="it.itiscastelli.classi.Ping" />
```

equivale a

```
Ping p = new Ping();
```

Il costruttore è richiamato *senza* argomenti.

Per istanziare un oggetto p usando il costruttore con due argomenti

```
Ping p = new Ping(host, counter);
```

occorre prima istanziare l'oggetto usando il costruttore *"nullo"*, poi usare i *setter*.

Quindi

```
Ping p = new Ping();
```

```
....
```

```
p.setHost(host);
```

```
p.setCounter(counter);
```

in JSTL diventa

```
<jsp:useBean id="p" class="it.itiscastelli.classi.Ping" />  
<jsp:setProperty name="p" property="host" value="{host}"/>  
<jsp:setProperty name="p" property="counter" value="{counter}"/>
```

Per richiamare il metodo **getExecPing()** in JSTL si scrive **execPing**.

La struttura selettiva *ad un ramo*

```
if (cond) {  
    ....  
}
```

è tradotta come

```
<c:if test="{cond}">  
    ....  
</c:if>
```

dove l'espressione *{cond}* è detta *Expression Language (EL)*.

La struttura selettiva *a due rami*

```
if (cond) {  
    ....  
} else {  
    ....  
}
```

è tradotta come

```
<c:choose>  
    <c:when test="{cond}">  
        ....  
    </c:when>  
    <c:otherwise>  
        ....  
    </c:otherwise>  
</c:choose>
```

e traduce anche la *struttura di selezione multipla*.

La struttura ciclica *forEach*

```
for (Oggetto o : collection) {  
    ....  
}
```

diventa

```
<c:forEach var="o" items="{collection}">  
    ....  
</c:forEach>
```