UNIVERSITÀ DI PISA

Programmazione di reti
Corso B

6 Dicembre 2016
Lezione 11

Contenuti

- Progetto
- Servizi web
  - SOAP vs REST
  - Cliente REST in Java

Progetto: *TextTwist*

- Implementare un gioco di rete: i giocatori ricevono un set di lettere e devono formare delle parole con quelle lettere. Vince la partita il giocatore che ha trovato le parole più lunghe e numerose. I giocatori accumulano punteggi partita dopo partita, con una classifica universale disponibile.

- 2 componenti:

  - `TwistServer` - riceve richieste dai clienti, memorizza gli utenti e il loro punteggio generale, gestisce le partite. Il server deve essere in grado di gestire più partite alla volta.

  - `TwistClient` - gestisce input dal cliente e comunica col *server,* gestisce la partita lato cliente. Twist client può avere solo una partita attiva alla volta.

Registrazione, *login* e *logout*

- Funzionalità implementate usando **RMI**
- Quando viene avviato `TwistClient`, l'utente può fare o *login*, o la registrazione.
- Per la registrazione, l'utente invia ***username* e *password*** e riceve la conferma dal server, che inserisce l'utente nella lista di utenti.
- Per il *login*, l'utente invia ***username* e *password*,** insieme ad altri dati, se necessari (dipende dal design dell'applicazione). Il server risponde con una conferma.
- Per fare *logout*, l'utente invia la richiesta e il *server* conferma.
- L'utente può **chiudere l'applicazione senza fare *logout*.** In questo caso, quando il server si accorge, il ***logout* avviene automaticamente sul *server***. La prossima volta che l'applicazione viene avviata, si mostrano le due opzioni: *login* e register.

Richiesta nuova partita

- Ogni partita viene iniziata da un giocatore
- Il giocatore invia al server una lista di altri giocatori (username) con cui vuole giocare (almeno un altro giocatore), su una connessione TCP
- Il server memorizza l'invito e lo invia ad ogni giocatore usando una callback RMI, messa a disposizione da ogni utente online.
- Il server conferma l'inoltro al giocatore che ha iniziato la partita e chiude la connessione TCP. Se qualche utente non è raggiungibile (online), si invia un messaggio di errore e la partita non viene avviata.

Setup nuova partita

- Server aspetta delle risposte per 7 minuti.
- Clienti visualizzano l’invito e lo accettano o rifiutano, inviando un messaggio al server su TCP.
- Se tutti i clienti accettano entro i 7 minuti, la partita può essere iniziata. Altrimenti viene cancellata.

Avvio nuova partita

- Il server invia ai clienti, usando le stesse connessioni TCP su cui ha ricevuto le risposte, una lista di lettere.

- I clienti ricevono le lettere, chiudono la connessione, e notificano gli utenti che il gioco è pronto.

- I clienti mostrano le parole quando gli utenti confermano. Da questo momento gli utenti hanno 2 minuti per inserire delle parole.

- Dopo 2 minuti gli utenti vengono fermati e le parole vengono inviate al server usando UDP.

Invio risultati

- Dopo aver inviato le lettere, il server aspetta per 5 minuti le parole su UDP.

- Una volta arrivate tutte, calcola i punteggi e invia la classifica a ogni utente, su un canale multicast impostato per questa partita

- Se dopo 5 minuti i clienti non inviano niente, ricevono 0 punti.

- I clienti ricevono il multicast e lo mostrano sullo schermo direttamente. Dopo la ricevuta è possibile iniziare un'altra partita, o usare le altre funzionalità del TwistClient.

Lettere e punteggi

- Sarà messo a disposizione un dizionario
- Le parole devono essere parte del dizionario e contenere solo le lettere a disposizione per essere validi
- Una parola valida di lunghezza N riceve N punti
- Per scegliere le lettere iniziale, un modo semplice sarebbe scegliere una parola lunga dal dizionario e permutare le lettere

Visualizzare clasifica

- Fuori una partita, gli utenti possono scegliere di consultare la classifica generale, inviando una richiesta TCP al server.

- Il server risponde con la lista ordinata di utenti ed il loro punteggio generale e chiude la connessione.

- La lista di utenti deve essere persistente: gli username e i punteggi non si devono perdere dopo un crash del server.

Aspetti generali

- Le funzionalità sono definite, però lo sviluppatore deve fare delle scelte di implementazione per arrivare al risultato voluto.

- Se una funzionalità non è definita al 100%, c'è spazio per le vostre idee.

- Si deve usare la tecnologia specificata per ogni funzionalità (RMI, TCP, UDP)

- Si può usare IO o NIO (bonus)

Interfaccia

- Testuale: di base.
    - Meno interattiva: gli inviti ad una partita non vengono mostrarti subito all'utente, perché questo potrebbe essere già in un'altra partita. Viene mostrata solo una notifica veloce (che interferirà con l'input). E' l'utente che deve guardare la lista di inviti per accettare la partita.
- Grafica (bonus):
    - Gli inviti ad una partita possono essere visualizzati subito quando arrivano (usando un'area diversa dell'interfaccia grafica).
    - L'interfaccia grafica non è la parte principale del progetto - deve solo facilitare l'interazione con le funzionalità di `TextTwist -` non perdete troppo tempo con l'interfaccia

Sottomissione

- **Solo se superati esami: Architettura e Sistemi Operativi**

- File di codice e *input*, configurazione, etc.

- Relazione in formato PDF con informazioni su: *design*, classi, *thread*, strutture dati, sincronizzazione.

- 2 eseguibili (jar): uno per cliente, uno per server.

- **Solo formato elettronico (per corso B)**.

Sottomissione

- Sul sito Moodle entro il *deadline* (alle **23:55!!!!**)

- Ci sarà un *deadline* per ogni appello di esame, 7 giorni prima dello scritto (**12 gennaio** e **2 febbraio** per i primi 2 appelli)

- Si può sottomettere il progetto per il *deadline* di un'appello e discuterlo in un appello successivo, o fare lo scritto in un appello successivo. L'unico vincolo è: se si vuol fare lo **scritto nell'appello n**, si deve superare la discussione del **progetto almeno nell'appello n**.

- Non si deve fare l'iscrizione all'esame se si intende fare solo il progetto. **L'iscrizione all'esame si fa per lo scritto** (se si vuole andare allo **scritto** all'appello **n** si fa l'iscrizione per l'appello **n**, a prescindere del fatto che il progetto si discute nell'appello **n-i o n**)

- Valutazione del modulo di laboratorio - preferibilmente prima della discussione

Discussione

- La sottomissione del progetto riceverà (su Moodle) un voto: 0 se non ammesso alla discussione, 1 se ammesso.

- Se ammesso, riceverete (su Moodle) un *link* a un poll *Doodle* dove potete prendere appuntamento per la discussione.

- Durante la discussione si **presenta il progetto** (dimostrazione del funzionamento + risposta a delle domande sul codice) e si risponde ad altre **domande** relative agli **argomenti studiati**.

- Per ogni appello ci sarà un giorno per la discussione (o 2 giorni se necessario). Per i primi due appelli la discussione si terrà nei giorni **16/17 gennaio**, **6/7 febbraio**

- **Portare student ID e laptop (se possibile).**

Discussione - bibliografia

- Le *slide*, gli esempi e gli appunti
- Documentazione Java 8
- ***Multithreading*:**
  - *Java concurrency tutorial*: https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html
  - Oaks & Wong, “Java Threads”, O’Reilly
  - Lewis & Berg, “Multithreaded programming with java technology”, Sun Microsystems
- ***Socket*:**
  - *Java networking tutorial*: https://docs.oracle.com/javase/tutorial/networking/index.html
  - Java NIO documentation
  - Harold, “Java Network Programming”, O’Reilly
- **RMI:**
  - *Java RMI tutorial*: https://docs.oracle.com/javase/tutorial/rmi/
  - Architettura RMI : https://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/guide/rmi/spec/rmi-arch.doc.html
- **Servizi web:**
  - Java web services tutorial: http://docs.oracle.com/javaee/6/tutorial/doc/bnayk.html

Come usare le costanti (configurazione)

- 4 possibilità - dalla meno alla più flessibile (e consigliabile):

    1. non definite come variabili e usate ‘hard coded’ direttamente nel codice: ogni volta che dobbiamo cambiare il valore, dobbiamo cercare tutti gli utilizzi della costante e cambiarla. Molto poco flessibile, **non consigliata**!

    2. definite come variabili `final static` in una classe : ogni volta che vogliamo cambiare un valore, cambiamo solo la definizione. Però comunque dobbiamo ricompilare il codice - non suggerito per applicazioni che vengono poi distribuite a clienti (il .jar non si ricompila facilmente!)

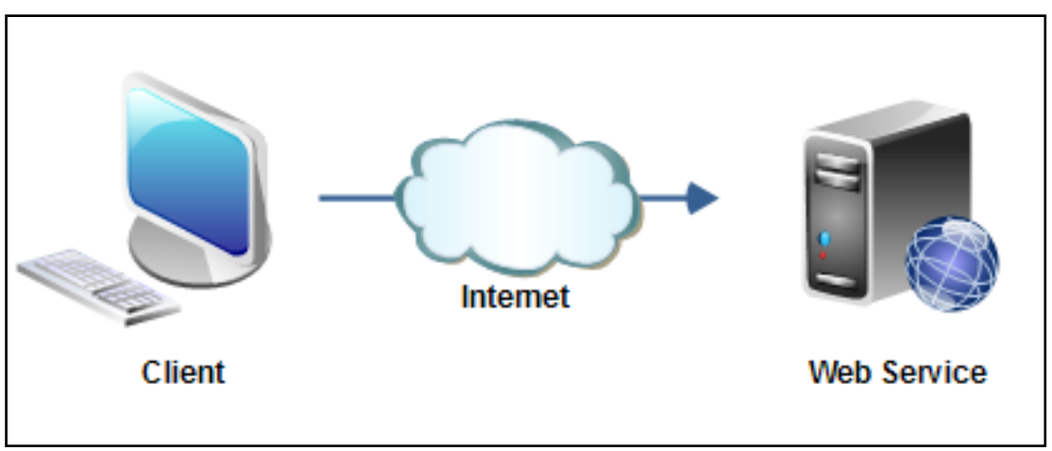
    3. ricevute come parametro alla riga di comando: valori possono essere cambiati senza compilare, però l’usabilità per clienti non-informatici è ridotta, anche quando si devono usare i parametri di *default*.

    4. definite in un file di configurazione: valori possono essere cambiati senza compilare, avvio dell’applicazione con i parametri di *default* molto facile.

Servizi Web

- “A Web service is a software system designed to support interoperable machine-to-machine interaction over a network.” (W3C, Web Services Glossary)
- Applicazione di rete caratterizzata di un’alta interoperabilità (sistema hardware, sistema operativo, linguaggio di programmazione)
- Mette a disposizione dei servizi (operazioni, risorse) - si possono combinare per ottenere servizi più complessi
- La comunicazione usa WWW (il protocollo HTTP) e messaggi XML/HTML/JSON
- 2 tipi
  - servizi web generali (“big”, “arbitrary”)
  - REST-compliant - i servizi e le operazioni

  rispettano un set di regole

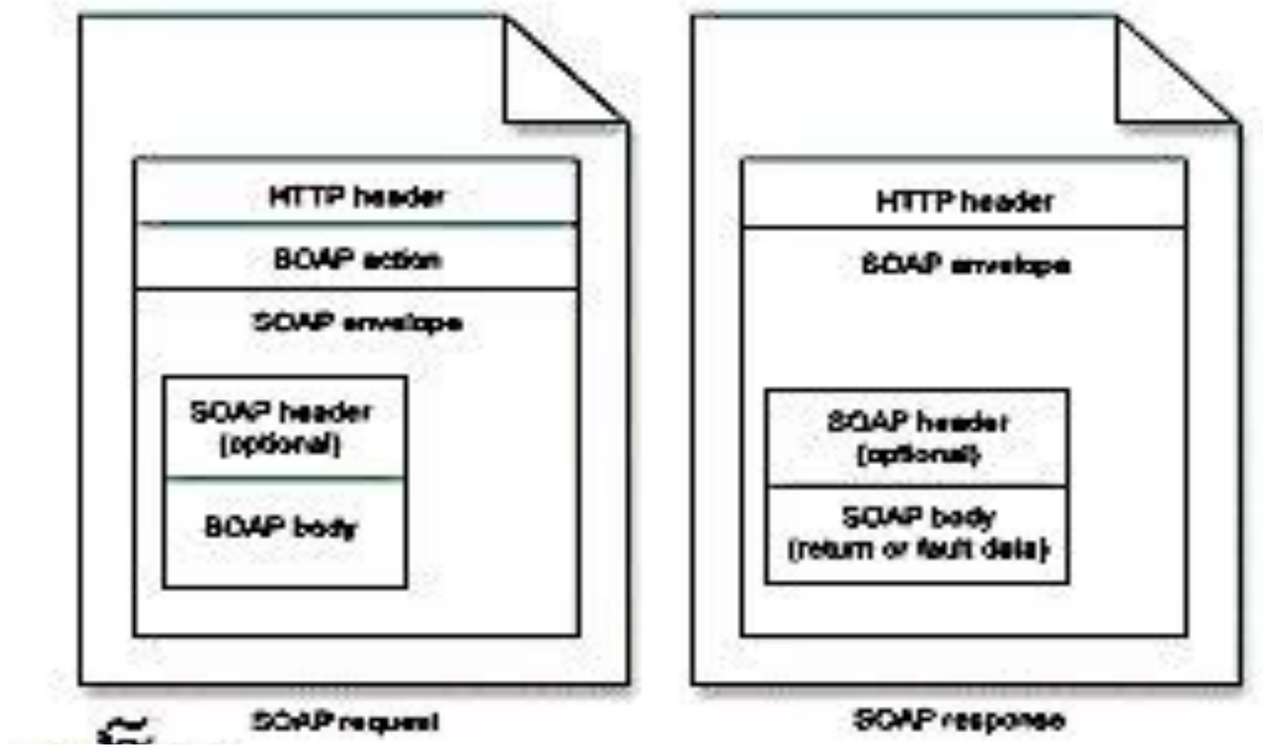


A cosa servono?

- Mettere a disposizione dei servizi agli altri sulla rete (anche a pagamento), in un formato uniforme
- Dividere applicazioni molto grandi (enterprise, monolitiche) in componenti autonomi, che sono più facili da mantenere
- SOA (Service Oriented Architecture)
  - vari componenti autonomi e riusabili
  - ogni componente e un ‘black box’ per le altre
  - ogni componente può avere sotto-componenti
  - ogni componente rappresenta un’unità logica dell’applicazione

Servizi web generali

- Usano il protocollo SOAP (Simple Object Access Protocol) per costruire messaggi XML da interscambiare.
- I messaggi rappresentano richieste (invocazioni) dal cliente o risposte dal server, sempre in formatto XML, con delle regole precise definite da SOAP
- I messaggi vengono spediti usando HTTP



Servizi web generali

- Idea simile a RMI (usando RPC) pero senza serializzazione Java (si passano messaggi XML su HTTP): ad ogni richiesta si invoca un metodo o procedura sul server e si inviano i risultati al cliente
  => **interoperabilità**

- I servizi sono descritti usando XML e il formatto definito da WSDL (Web Services Description Language) - le operazioni offerte (risorse web)

- Usati di solito nel contesto Enterprise (azienda con vari dipartimenti, ogni uno deve comunicare con gli altri, si mettono a disposizione dei servizi web)

Risorse web

- All'inizio del web: documenti e file statici, identificabili usando indirizzo e nome.

- Adesso : qualsiasi entità **identificabile**, **accessibile** sul web o in qualsiasi rete, che possa essere letta, modificata, cancellata, o **usata** in qualche modo.

- I servizi web mettono a disposizione risorse, usate dai clienti

Servizi web REST-ful

- Representation State Transfer (REST)
- Servizi web **semplici**, con operazioni **stateless** (il server non ricorda lo stato del cliente, tutte le informazioni vengono inviate nella richiesta)
- Non usano SOAP e WSDL, si integrano direttamente con HTTP
- Usano meno bandwidth dei servizi SOAP (manca SOAP envelope), quindi più validi per clienti mobili
- Performance, scalabilità, portabilità.
- Usati di solito per servizi web pubblici:
  - Google Search, Google Translate,
  - Google Directions, Amazon API, etc.

Servizi web REST-ful

- Risorse vengono identificate come URI (Uniform Resource Identifier)

- Interfaccia uniforma - ogni servizio web mette a disposizione le stesse operazioni predefinite molto semplici, PUT, GET, DELETE, POST su una risorsa

- Messaggi auto-descrittivi: risorse e la loro rappresentazione sono diverse - clienti usano la rappresentazione (HTML, XML, testo, JSON, PDF, etc)

- Interazioni *stateful* basate su hyperlinks - anche se interazioni con una risorsa è sempre stateless, e possibile fare il trasferimento esplicito dello stato (inviato nella richiesta/risposta)

Clienti REST in Java

- Usano richieste HTTP per ottenere risorse dal server

Richieste HTTP

- HTTP - *HyperText Transfer Protocol*
- Usato per accedere a pagine *web* (incluso immagini, video, audio) e servizi *web*
- GET - richiediamo i contenuti di una risorsa - non ci sono effetti collaterali - richiesta può essere ripetuta - tutte le informazioni per rispondere alla richiesta sono nell'URL
- PUT - inviamo una risorsa ad un server ad un indirizzo fisso- può essere ripetuta con lo stesso effetto finale- ci sono informazioni incluse nel contenuto della richiesta
- DELETE - cancelliamo una risorsa.
- POST - inviamo una risorsa, però non è definito se o dove la risorsa viene memorizzata, o l'effetto dell'operazione. Di solito usata per operazioni non ripetibili. Ci sono informazioni incluse nel contenuto della richiesta.

Uniform Resource Locator (URL)

- Riferimento ad una risorsa su Internet
- Pagina *web* o altro tipo di risorsa (immagini, file, servizi web)
- Identifica
  - Protocollo
  - Localizzazione della risorsa - può includere *hostname*, *path*, *port*
  - Parametri della richiesta (*query string*)

https://www.unipi.it/

file:///Users/Alina/Dropbox/unipi/courses/nets/2015-2016/JavaNetworkProgramming.pdf

https://www.google.it/search?q=italy

Java URL

- Creare URL assoluto

```
URL url= new URL(“https://docs.oracle.com/javase/tutorial/
networking/urls/creatingUrls.html”)

URL url=new URL(“https”,”docs.oracle.com”,”/javase/tutorial/
networking/urls/creatingUrls.html”)
```

- Creare URL relativo ad un altro URL

```
URL base= new URL(“https://docs.oracle.com/javase/tutorial/
networking/urls/“)

URL info=new URL(base,”urlInfo.html”)

URL creating= new URL(base,”creatingUrls.html”)
```

- Lanciano `MalformedURLException`

URL

- Formatto limitato - una lista di caratteri permessi
- per cercare su Google il testo “pisa university”, l’URL è http://www.google.com/search?q=pisa+university
- Classe URLEncoder

```
URL url= new URL(“http://www.google.com/search?
q=“+URLEncoder.encode(“pisa university”,"UTF-8"));
```

- Classe URLDecoder

```
String input = "https://www.google.it/webhp?sourceid=chrome-
instant&ion=1&espv=2&ie=UTF-8#q=pisa%20university";
String output = URLDecoder.decode(input, "UTF-8");
System.out.println(output);
```

```
https://www.google.it/webhp?sourceid=chrome-
instant&ion=1&espv=2&ie=UTF-8#q=pisa university
```

Leggere la risorsa

- Usando metodo `openStream`()

```
URL url= new URL(“https://
docs.oracle.com/javase/tutorial/
networking/urls/creatingUrls.html”)

InputStream stream= url.openStream();
```

- Invia un richiesta HTTP GET, la risposta e disponibile nello *stream*

```
public class URLTest {

   public static void main(String[] args) {
      try {
         URL url= new URL("https://docs.oracle.com/"
               + "javase/tutorial/networking/urls/creatingUrls.html");
         try(BufferedReader in = new BufferedReader(
                        new InputStreamReader(url.openStream()));){
            String line;
            while ((line=in.readLine())!=null){
               System.out.println(line);
            }
         }
      } catch (MalformedURLException e) {
         System.err.println("Wrong URL format");
      } catch (IOException e) {
         System.err.println("Error reading resource");
      }
   }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">


<html lang="en-US" xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Creating a URL (The Java&trade; Tutorials &gt;
            Custom Networking &gt; Working with URLs)
</title>
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="description" content="This networking Java tutorial describes networking
capabilities of the Java platform, working with URLs, sockets, datagrams, and cookies" />
    <meta name="keywords" content="java programming, learn java, java sample code, java
networking, java url, java socket, datagram, java cookie" />

<style type="text/css">


. . .
```

URLConnection

- Interfaccia per classi specializzati in controllare in dettaglio la connessione ad un URL
- In base al protocollo usato ci sono varie implementazioni
- Si ottiene richiamando metodo `openConnection()` della classe `URL`
- Permette di scegliere valori per vari parametri della connessione
- Dopo aver scelto le opzioni, la connessione è avviata usando metodo `connect()`
- Permette di leggere però anche di scrivere

URLConnection

- Definisce configurazione della richiesta
- Legge il *header* della risposta
- legge una risorsa = GET (simile a URL)
- scrive una risorsa = POST

Leggere e scrivere

- Leggere una risorsa: usare l'input stream

```
public InputStream getInputStream()
```

GET: *query string* deve essere incluso nell'URL

- Scrivere una risorsa: prima abilitare *l'output*, poi usare *l'output stream*

```
void setDoOutput(Boolean )
```

```
public OutputStream getOutputStream()
```

POST: *query string* viene inviato usando *l'output stream*

Chiudere connessione

- Chiudere gli *stream* - non chiude la connessione
- Chiudere connessione - chiude anche gli stream

```
void disconnect()
```

Configurazione della richiesta

- Il metodo di accesso implicito e GET, input è automaticamente abilitato. Può essere disabilitato:

```
public void setDoInput(boolean doInput)
public boolean getDoInput()
```

- Per abilitare l'output (il metodo di accesso al server diventa POST:

```
public void setDoOutput(boolean doOutput)
public boolean getDoOutput()
```

- Per usare caching:

```
public void setUseCaches(boolean useCaches)
public boolean getUseCaches()
```

Configurazione della richiesta

- Per richiedere risorse solo se modificate dalla ultima lettura:

```
public void setIfModifiedSince(long ifModifiedSince)
public long getIfModifiedSince()
```

Server invia la risorsa solo se modificata dopo il *timestamp*. Altrimenti risponde con 304 *Not Modified*

- Per impostare un *timeout*:

```
public void setConnectTimeout(int timeout)
public int getConnectTimeout()

public void setReadTimeout(int timeout)
public int getReadTimeout()
```

Configurazione della richiesta

- Configurare *header* (richiesta)

Accept:text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Charset:ISO-8859-1,utf-8;q=0.7,*;q=0.3
Accept-Encoding:gzip,deflate,sdch
Accept-Language:en-US,en;q=0.8
Cache-Control:max-age=0
Connection:keep-alive
User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_3) AppleWebKit/537.31
(KHTML, like Gecko) Chrome/26.0.1410.65 Safari/537.31

```
void setRequestProperty(String name, String value)

void addRequestProperty(String name, String value)
```

Leggere *header* (risposta)

- Per sapere interpretare la sequenza di *byte* che ci arriva dallo *stream*:

```
String getContentType()
```

e.g.: “text/html; charset=ISO-8859-1”

Possibile risposte: text/plain, image/gif, application/xml, image/jpeg

```
static String guessContentTypeFromName(String name)
static String guessContentTypeFromStream(InputStream in)
```

- Per sapere la lunghezza (numero di *byte*) dello stream:

```
int  getContentLength()
```

- Altre informazioni:

`public String getContentEncoding()` (se non `null` può essere *gzip*, etc)

```
public long getDate()
public long getExpiration()
public long getLastModified()
```

HttpURLConnection

- Classe che implementa l'interfaccia `URLConnection`, usata con il protocollo HTML
- Metodi addizionali, tra cui:

```
public int getResponseCode() throws IOException

public String getResponseMessage() throws
IOException
```

```
public class Search {
    public static void main(String[] args) {
        try {
            String google_service= "http://www.google.com/search?q=";
            String[] terms={"pisa university"};
            for (String term: terms){
                URL google= new URL(google_service+URLEncoder.encode(term,"UTF-8"));
                HttpURLConnection connection = (HttpURLConnection) google.openConnection();
                connection.addRequestProperty("User-Agent","Chrome" );//pretend to be a browser
                connection.connect();
                System.out.println(connection.getResponseCode());
                System.out.println(connection.getResponseMessage());
                String[] contentType=connection.getContentType().split(";");
                if (contentType[0].startsWith("text")){
                String encoding="UTF-8";
                if (contentType.length>1)
                    encoding=contentType[1].substring(9);
                try(BufferedReader in = new BufferedReader(
                        new InputStreamReader(connection.getInputStream(),encoding));
                        BufferedWriter out= new BufferedWriter(
                                new FileWriter(term+"Result.html")));{
                    String resultLine;
                    while ((resultLine = in.readLine()) != null) {
                        out.write(resultLine+"\n");
                    }
                }
            } else{
                System.out.println("Server sent binary response.");
            }
```

UTF-8 è encoding predefinito per HTML5
<head><meta content="text/html; charset=UTF-8" http-equiv="Content-Type">

```
    }
} catch (MalformedURLException e) {
    System.err.println("Wrong URL format:"+e.getMessage());
} catch (IOException e) {
    System.err.println("Error with connection:"+e.getMessage());
}
    }
}
```

```
200
OK
```

Senza User-Agent:

```
403
Error with connection:Server
returned HTTP response code: 403
for URL: http://www.google.com/
search?q=pisa+university
Forbidden
```

pisa university

Circa 524.000 risultati

english - Università di Pisa
https://www.unipi.it/index.php/english
A bridge between the Department of Chemistry and KAUST, one of Saudi Arabia's leading universities · Lecturer of the **University of Pisa** is part of the ...
Study Programmes - Students - University - Research

Università di Pisa
https://www.unipi.it/
Le iniziative a Pisa per la manifestazione promossa dalla CRUI - 21 marzo ... Centri e sistemi di ateneo · Amministrazione Centrale · **Pisa University** Press ...
Corsi di laurea - uniMap - Università di Pisa

Università di Pisa - Wikipedia
https://it.wikipedia.org/wiki/Università_di_Pisa
L'Università di **Pisa** è l'unica università italiana a far parte della **Universities** Research Association, consorzio di università fra le più prestigiose al mondo, ...

University of Pisa - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/**University_of_Pisa**
The **University of Pisa** is an Italian public research university located in Pisa, Italy. It was founded in 1343 by an edict of Pope Clement VI. It is the 19th oldest ...
History - Organization and administration - Rankings - Notable people

Pisa University Press
www.**pisauniversity**press.it/
Aula Magna del Polo didattico Carmignani, (Piazza dei Cavalieri, **Pisa**) Presentazione del volume "Libertà di espressione e libertà religiosa in tempi di crisi ...

Università di Pisa

Università a Pisa, Italia

Indicazioni stradali | Sito web

L'Università di Pisa è una delle più antiche e prestigiose università italiane e d'Europa. Fondata nel 1343, tuttora mantiene una dimensione importante, soprattutto in rapporto al numero di abitanti ... Wikipedia

Lungarno Antonio Pacinotti, 43, 56126 Pisa PI

Iscrizione: 45.014 (2014)
Fondazione: 3 settembre 1343
Provincia: Provincia di Pisa
Telefono: 050 221 2111

```
<a href="/url?q=https://www.unipi.it/index.php/english&amp;sa=U&amp;ved=0ahUKEwjPxo7FyMrLAhVD6RQKHSMHDVcQFggUMAA&amp;usg=AFQjCNExCDfLmE9GomJ-65VR79YQ3S8tuA">english - Università di <b>Pisa</b></a></h3><div class="s"><div class="kv" style="margin-bottom:2px"><cite>https://www.unipi.it/index.php/english</cite>
```

Ricerca Immagini Maps Play YouTube News Gmail Drive Altro »

pisa university

Tutti Immagini Video Notizie Shopping Maps Libri

Circa 755.000 risultati

Qualsiasi Paese
Paese: Italia

Qualsiasi lingua
Pagine in italiano

Qualsiasi data
Ultima ora
Ultimo 24 ore
Ultima settimana
Ultimo mese
Ultimo anno

Tutti i risultati
Verbatim

english - Universit&#65533; di Pisa
https://www.unipi.it/index.php/english
A bridge between the Department of Chemistry and KAUST, one of Saudi Arabia's leading universities · Lecturer of the **University of Pisa** is part of the ...
Study Programmes - Students - University - Research

Universit&#65533; di Pisa
https://www.unipi.it/
... Centri e sistemi di ateneo · Amministrazione Centrale · **Pisa University** Press ... Universit&#65533; di Pisa; Lungarno Pacinotti 43; 56126 Pisa; P.I. 00286820501; C.F. ...
Corsi di laurea - uniMap - Universit&#65533; di Pisa

Universit&#65533; di Pisa - Wikipedia
https://it.wikipedia.org/wiki/Universit&#65533;_di_Pisa
L'Universit&#65533; di Pisa &#65533; l'unica universit&#65533; italiana a far parte della **Universities** Research Association, consorzio di universit&#65533; fra le pi&#65533; prestigiose al mondo, ...

University of Pisa - Wikipedia, the free encyclopedia
https://en.wikipedia.org/wiki/University_of_Pisa
The **University of Pisa** (Italian: Universit&#65533; di Pisa, UniPi) is an Italian public research university located in Pisa, Italy. It was founded in 1343 by an edict of Pope ...
History - Organization and administration - Rankings - Notable people

Pisa University Press
www.pisauniversitypress.it/
Aula Magna del Polo didattico Carmignani, (Piazza dei Cavalieri, **Pisa**) Presentazione del volume "Libert&#65533; di espressione e libert&#65533; religiosa in tempi di crisi ...

Contatti - Pisa University Press

Mappa per Universit&#65533; di P

Università di Pisa

Universit&#65533; di Pisa

Universit&#65533; a Pisa, Italia

Indicazioni stradali | Sito web

L'Universit&#65533; di Pisa &#65533; una delle pi&#65533; antiche e prestigiose universit&#65533; italiane e d'Europa. Fondata nel 1343, tuttora mantiene una dimensione importante, soprattutto in rapporto al numero di abitanti ... Wikipedia

Lungarno Antonio Pacinotti, 43, 56126 Pisa PI

Iscrizione: 45.014 (2014)
Fondazione: 3 settembre 1343
Provincia: Provincia di Pisa
Telefono: 050 221 2111

Alunni celebri

Se non leggo l'encoding

```
public class Search {
    public static void main(String[] args) {
        try {
            String google_service= "http://www.google.com/search?q=";
            String[] terms={"pisa university"};
            for (String term: terms){
                URL google= new URL(google_service+URLEncoder.encode(term,"UTF-8"));
                HttpURLConnection connection = (HttpURLConnection) google.openConnection();
                connection.addRequestProperty("User-Agent","Chrome" );//pretend to be a browser
                connection.connect();
                System.out.println(connection.getResponseCode());
                System.out.println(connection.getResponseMessage());
                String[] contentType=connection.getContentType().split(";");
                if (contentType[0].startsWith("text")){
                String encoding="UTF-8";
                if (contentType.length>1)
                    encoding=contentType[1].substring(9);
                try(BufferedReader in = new BufferedReader(
                        new InputStreamReader(connection.getInputStream(),encoding));
                        BufferedWriter out= new BufferedWriter(
                                new FileWriter(term+”Result.html")));{
                    String resultLine;
                    while ((resultLine = in.readLine()) != null) {
                        out.write(resultLine+"\n");
                    }
                }
            } else{
                System.out.println("Server sent binary response.");
            }
```

risposta HTTP
charset="ISO-8859-1"

pagina web
<meta charset="UTF-8"/>

protocollo
HTTP

browser

Usa ISO-8859-1 e UTF-8

Caratteri interpretati
correttamente

Usa ISO-8859-1

La nostra classe
Search vecchia

Salva HTML in un file locale usando charset default
Su UNIX : UTF-8
Su Windows: Windows-1252

pagina web
<meta charset="UTF-8"/>

protocollo
file

browser

Usa UTF-8

Caratteri interpretati
correttamente su UNIX
sbagliati su Windows

```
public class Search {
    public static void main(String[] args) {
        try {
            String google_service= "http://www.google.com/search?q=";
            String[] terms={"pisa university"};
            for (String term: terms){
                URL google= new URL(google_service+URLEncoder.encode(term,"UTF-8"));
                HttpURLConnection connection = (HttpURLConnection) google.openConnection();
                connection.addRequestProperty("User-Agent","Chrome" );//pretend to be a browser
                connection.connect();
                System.out.println(connection.getResponseCode());
                System.out.println(connection.getResponseMessage());
                String[] contentType=connection.getContentType().split(";");
                if (contentType[0].startsWith("text")){
    String encoding="UTF-8";
    if (contentType.length>1)
        encoding=contentType[1].substring(9);
    try(BufferedReader in = new BufferedReader(
            new InputStreamReader(connection.getInputStream(),encoding));
            BufferedWriter out= new BufferedWriter(new OutputStreamWriter(
                    new FileOutputStream(term+"Result.html"),encoding));{
        String resultLine;
        while ((resultLine = in.readLine()) != null) {
            out.write(resultLine+"\n");
        }
    }
} else{
    System.out.println("Server sent binary response.");
}
```

risposta HTTP
charset="ISO-8859-1"

pagina web
<meta charset="UTF-8"/>

protocollo
HTTP

browser

Usa ISO-8859-1 e UTF-8

Caratteri interpretati
correttamente

Usa ISO-8859-1

La nostra classe
Search usando
variabile `encoding`

Salva HTML in un file locale
usando charset ISO-8859-1

pagina web
<meta charset="UTF-8"/>

protocollo
file

browser

Usa UTF-8

**Caratteri sbagliati su
tutte le piattaforme**

```
public class Search {
    public static void main(String[] args) {
        try {
            String google_service= "http://www.google.com/search?q=";
            String[] terms={"pisa university"};
            for (String term: terms){
                URL google= new URL(google_service+URLEncoder.encode(term,"UTF-8"));
                HttpURLConnection connection = (HttpURLConnection) google.openConnection();
                connection.addRequestProperty("User-Agent","Chrome" );//pretend to be a browser
                connection.connect();
                System.out.println(connection.getResponseCode());
                System.out.println(connection.getResponseMessage());
                String[] contentType=connection.getContentType().split(";");
                if (contentType[0].startsWith("text")){
                String encoding="UTF-8";
                if (contentType.length>1)
                    encoding=contentType[1].substring(9);
                try(BufferedReader in = new BufferedReader(
                        new InputStreamReader(connection.getInputStream(),encoding));
                        BufferedWriter out= new BufferedWriter(new OutputStreamWriter(
                                new FileOutputStream(term+"Result.html"),"UTF-8"));{
                    String resultLine;
                    while ((resultLine = in.readLine()) != null) {
                        out.write(resultLine+"\n");
                    }
                }
            } else{
                System.out.println("Server sent binary response.");
            }
```

risposta HTTP
charset="ISO-8859-1"

pagina web
<meta charset="UTF-8"/>

protocollo
HTTP

browser

Usa ISO-8859-1 e UTF-8

Caratteri interpretati
correttamente

Usa ISO-8859-1

La nostra classe
Search usando
"UTF-8"

Salva HTML in un file locale
usando charset UTF-8

pagina web
<meta charset="UTF-8"/>

protocollo
file

browser

Usa UTF-8

Caratteri interpretati
correttamente

- Idealmente non dovremmo usare “UTF-8” hardcoded, anche se probabilmente funziona meglio per tutte le piattaforme.
- Dovremmo leggere il HTML, prendere il charset dal tag <meta>, e usare quello per salvare il file su disco.

Dati

- Le richieste alle **pagine *web*** restituiscono il codice HTML - utile per visualizzare nel *browser*

- Per estrarre informazioni dobbiamo percorrere il HTML

- **Servizi web** REST usano il protocollo HTTP, però possono inviare dati in altri format - evitano di inviare tutti i *tag* HTML che non sono importanti

- XML, JSON

Esempio

- Google *search* con JSON
- Google offre un *web service REST,* **Custom Search Engine** per fare *search* e ricevere risultato in JSON invece di HTML - parte della piattaforma Cloud di Google
    - https://www.googleapis.com/customsearch/v1?parameters
- I parametri sono:
    - q=[search term] - term da ricercare
    - key=[API Key] - chiave da ottenere attivando la API di Google nel google cloud
    - cx=[custom search engine id] - chiave ottenuta dopo aver creato un nuovo search engine al sito https://cse.google.com
    - key e cx identificano il cliente

https://www.googleapis.com/customsearch/v1?key=**AIzaSyAiQN8BMsh5CVe9W443IlwHFbIBc-Upncc**&cx=**006020613947421378551:1rofj878xse**&q=**pisa%20university**

Le mie chiavi - saranno disponibili fino al 19 Dicembre

Per ottenere le chiavi:

API Key :

1. http://console.google.com
2. Creare un nuovo progetto
3. Attivare il billing
4. Attivare l'API di Custom Search Engine
5. Generare le credentials

Custom Search Engine key:

1. https://cse.google.com/cse/create/new

```
{
 "kind": "customsearch#search",
 "url": {
  "type": "application/json",
  "template": "https://www.googleapis.com/customsearch/v1?q={searchTerms}&num={count?}&start={startIndex?}
&lr={language?}&safe={safe?}&cx={cx?}&cref={cref?}&sort={sort?}&filter={filter?}&gl={gl?}&cr={cr?}
&googlehost={googleHost?}&c2coff={disableCnTwTranslation?}&hq={hq?}&hl={hl?}&siteSearch={siteSearch?}
&siteSearchFilter={siteSearchFilter?}&exactTerms={exactTerms?}&excludeTerms={excludeTerms?}
&linkSite={linkSite?}&orTerms={orTerms?}&relatedSite={relatedSite?}&dateRestrict={dateRestrict?}
&lowRange={lowRange?}&highRange={highRange?}&searchType={searchType}&fileType={fileType?}
&rights={rights?}&imgSize={imgSize?}&imgType={imgType?}&imgColorType={imgColorType?}
&imgDominantColor={imgDominantColor?}&alt=json"
 },
"queries": {
 "request": [
  {
   "title": "Google Custom Search - pisa university",
   "totalResults": "824000",
   "searchTerms": "pisa university",
   "count": 10,
   "startIndex": 1,
   "inputEncoding": "utf8",
   "outputEncoding": "utf8",
   "safe": "off",
   "cx": "006020613947421378551:1rofj878xse"
  }
 ],
```

"**items**": [
 {
  "kind": "customsearch#result",
  "title": "ENGLISH",
  "htmlTitle": "ENGLISH",
  "**link**": "https://www.unipi.it/english/",
  "displayLink": "www.unipi.it",
  "snippet": "Cherubino, logo Università di Pisa ... When a star 'eats' a planet · A study on '\nalien' puffer fish in our seas · Activation of the University of Pisa's “Foundation \nYear” ...",
  "htmlSnippet": "Cherubino, logo Università di \u003cb\u003ePisa\u003c/b\u003e ... When a star 'eats' a planet · A study on '\u003cbr\u003e\nalien' puffer fish in our seas · Activation of the \u003cb\u003eUniversity\u003c/b\u003e of Pisa's “Foundation \u003cbr\u003e\nYear” ...",
  "cacheId": "f63CR8EdhF8J",
  "formattedUrl": "https://www.unipi.it/english/",
  "htmlFormattedUrl": "https://www.unipi.it/english/",
  "pagemap": {
   "cse_thumbnail": [
    { "width": "308", "height": "164", "src": "https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcSk9f4y9eeCEOpBQd7o12CUiOA2bIgCKHC7DfOMpB-s2GmvLTem6ItMRdk"}
   ],
   "metatags": [
    {
     "viewport": "user-scalable=yes, width=device-width, initial-scale=1.0","og:url": "https://www.unipi.it/index.php/english", "og:title": "ENGLISH","og:type": "website"
    }
   ],
   "cse_image": [
    {
     "src": "http://pages.di.unipi.it/throughthefog/wp-content/uploads/sites/13/2016/05/marchio_unipi_black.png"
    }
   ]
  }
 }, [etc…..]

```
public class SearchJson {
      public static void main(String[] args) {
         try {
            JSONParser parser= new JSONParser();

            String google_service= "https://www.googleapis.com/customsearch/v1?"
                  + "key=AIzaSyAiQN8BMsh5CVe9W443llwHFbIBc-Upncc"
                  + "&cx=006020613947421378551:1rofj878xse&q=";
            String term="pisa university”;

            URL google= new URL(google_service+URLEncoder.encode(term,"UTF-8"));
            HttpURLConnection connection =
                        (HttpURLConnection) google.openConnection();

            connection.connect();

            System.out.println(connection.getResponseCode());
            System.out.println(connection.getResponseMessage());
```

```
String[] contentType=connection.getContentType().split(“;");

if (contentType[0].contains("json")){
      String encoding="UTF-8";
      if (contentType.length>1)
         encoding=contentType[1].substring(9);
      try(BufferedReader in = new BufferedReader(
                  new InputStreamReader(
                  connection.getInputStream(),encoding));){
              JSONObject response= (JSONObject) parser.parse(in);
              JSONArray results=(JSONArray)(response.get("items"));
              for (Object result: results){
                  System.out.println(((JSONObject)result).get("link"));
              }
          }
   } else{System.out.println("Server sent binary response.");
   }
} catch (MalformedURLException e) {
   System.err.println("Wrong URL format:"+e.getMessage());
} catch (IOException e) {
   System.err.println("Error with connection:"+e.getMessage());
} catch (ParseException e) {
   System.err.println("Error parsing:"+e.getMessage());
}}}
```

```
200
OK
https://www.unipi.it/english/
https://en.wikipedia.org/wiki/University_of_Pisa
http://www.aboutpisa.info/university-of-pisa.html
http://www.mastersportal.eu/universities/501/university-of-pisa.html
http://www.topuniversities.com/universities/university-pisa/undergrad
http://mba.master.unipi.it/
http://www.topuniversities.com/universities/university-pisa/postgrad
https://www.timeshighereducation.com/world-university-rankings/university-of-pisa
http://masterriskmanagement.ec.unipi.it/
https://www.youtube.com/watch?v=Rx6_GhpxRG8
```

Esempio Flickr

- Cliente Java che utilizza le API Flickr - ricerca e scarica le prime 5 immagini.
  - Parametri:
    - tagName
- REST API:
  - https://api.flickr.com/services/feeds/photos_public.gne
  - Parametri:
    - format=json
    - nojsoncallback=1
    - tag=<tagName>