UNIVERSITÀ DI PISA

Programmazione di reti
Corso B

22 Marzo 2016
Lezione 5

Contenuti

- Serializzazione Java
- Inviare oggetti
- Richieste HTTP
- Formatto JSON

Serializzazione

- Trasformare un oggetto in una rappresentazione binaria (sequenza di *byte*) che può essere scritta in un stream (e.g. per memorizzarla in un *file*).
- L'oggetto può essere ricreato dalla sequenza di byte: deserializzazione.
- Il processo è indipendente dalla piattaforma però dipendente dal linguaggio di programmazione: deve essere Java per serializzazione e deserializzazione
- 2 Passi:
  - Creare una classe serializzabile
  - Usare `ObjectInputStream/ObjectOutputStream` per lavorare con gli oggetti.

`Serializable`

- Un oggetto è serializzabile se la classe implementa `Serializable`
- Tutti i tipi di dati primitivi sono serializzabili.
- Per gli oggetti non-primitivi, se implementano `Serializable`, sono serializzabili (controllare documentazione)
- Per essere serializzabile, un tipo di dati custom deve contenere solo attributi serializzabili o `transient/static`
- Gli attributi `transient/static` non vengono serializzati
- Serializzazione implicita: gli attributi vengono serializzati in automatico
- Serializzazione *custom*: la classe sovrascrive la serializzazione implicita

Serializable

- La sequenza di byte ottenuta contiene informazioni che identificano il tipo di oggetto (la gerarchia di classi) e il valore degli attributi

- Se l'oggetto contiene un riferimento ad un secondo oggetto, anche esso viene serializzato (un albero di oggetti)

- Al momento della deserializzazione, il JVM deve trovare la definizione della classe (il file .class), altrimenti viene lanciata una `ClassNotFoundException`

- Questo diventa più problematico quando si lavora in rete.

- I metodi dell'oggetto non vengono serializzati, sono estratti dalla definizione della classe (*file* `.class`)

Serial Version

- Per garantire che i processi di serializzazione e deserializzazione usano versioni compatibili di una classe, Java definisce un UID (*unique ID*) per la definizione di una classe

- è indicato includere un `serialVersionUID` nella definizione della classe. Altrimenti la JVM genera uno per noi (in base alla definizione della classe), però questo può causare problemi se si lavora su piattaforme diverse.

- pe dichiarare la versione:

```
private static final long serialVersionUID = 15L;
```

Serial Version

- Se la classe evolve però è sempre *backwards-compatible*, allora si deve usare lo stesso `serialVersionUID` anche nelle versioni nuove
  - es: aggiungere attributi, aggiungere/rimuovere classi interne, trasformare attributi *transient* in *non-transient*
- Se la nuova versione non e compatibile con quella precedente, il `serialVersionUID` deve essere cambiato
  - es: rimuovere attributi, trasformare attributi *non-transient* in *transient*

```java
public class Student implements Serializable{

    private static final long serialVersionUID = 1L;

    private String fname, lname;
    private String streetAddress;
    private int addressNo;

    public Student(String fname, String lname,String street, int no){
        this.fname=fname;
        this.lname=lname;
        this.streetAddress= street;
        this.addressNo=no;
    }
    public String toString(){
        StringBuilder sb= new StringBuilder();
        sb.append(this.fname);
        sb.append(" ");
        sb.append(this.lname);
        sb.append(" living at ");
        sb.append(this.addressNo);
        sb.append(" ");
        sb.append(this.streetAddress);
        sb.append(" street.");
        return sb.toString();
    }
}
```

Tutti gli attributi sono serializzabili quindi la classe è serializzabile

ObjectOutputStream

- Un *filter*, da usare con *stream raw*, dedicato a serializzare tipi di dati primitivi e oggetti serializzabili

```
void writeBoolean(boolean val)

void writeByte(int val)

void writeBytes(String str)

void writeChar(int val)

void writeChars(String str)

void writeDouble(double val)
```

ObjectOutputStream

```
void writeFloat(float val).

void writeInt(int val)

void writeLong(long val)

void writeShort(int val)

void writeUTF(String str)
```

ObjectOutputStream

```
void  writeObject(Object o)
```

Scrive l'informazione sulla classe e i valori degli attributi da serializzare (non *transient* e non statici). Se un oggetto viene scritto due volte, la prima volta un riferimento è generato e la seconda volta solo il riferimento viene scritto.

Attenzione: se l'oggetto viene modificato dopo essere stato scritto la prima volta, la modifica non è salvata la seconda volta.

```
Student s=  new Student(“Robert”,”Brown","Dawson",12);
out.writeObject(s);
s.setFname(“Robbie”);
out.writeObject(s);
```

ObjectOutputStream

```
void writeUnshared(Object o)
```

Uguale a **`writeObject`** però oggetti ripetuti sono scritti come nuovi senza fare riferimento agli precedenti.

ObjectInputStream

- *Filter* specializzato in deserializzazione
- Metodi lanciano EOFException alla fine del *stream*

```
boolean readBoolean()

byte readByte()

char readChar()

double  readDouble()

float readFloat()
```

ObjectInputStream

```
int readInt()

long readLong()

short readShort()

int readUnsignedByte()

int readUnsignedShort()

String readUTF()
```

ObjectInputStream

`Object readObject()`

Legge l'informazione sulla classe e i valori degli attributi da deserializzare (non *transient* e non statici). Se si incontra un riferimento ad un oggetto già letto, si restituisce quel oggetto direttamente.

ObjectInputStream

`Object readUnshared()`

Da usare per leggere oggetti scritti con `writeUnshared`

```
public class WriteStudentMain {

   public static void main(String[] args) {
      ArrayList<Student> students= new ArrayList<Student>();
      students.add(new Student("Robert","Brown","Dawson",12));
      students.add(new Student("Michael","Reds","Pearse",40));
      students.add(new Student("Joanna","Moore","Collins",62));
      students.add(new Student("Ann","Brown","Buffallo",132));

      try (ObjectOutputStream out= new ObjectOutputStream(
            new FileOutputStream("students.ser"));) {
         out.writeObject(students);
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

students.ser — Edited

```
¨Ìsrjava.util.ArrayListxÅ“ô«aùIsizexpwsrStudentI
addressNoLfnametLjava/lang/String;Llnameq~L
streetAddressq~xp
tRoberttBrowntDawsonsq~(tMichaeltRedstPearsesq~>tJoa
nnatMooretCollinssq~ÑtAnnq~tBuffallox
```

291 byte

```
public class ReadStudentMain {

   public static void main(String[] args) {
      try(ObjectInputStream in= new ObjectInputStream(
            new FileInputStream("students.ser"));){
         ArrayList<Student> stds= (ArrayList<Student>) in.readObject();
         for (Student s : stds){
            System.out.println(s);
         }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
   }
}
```

```
Robert Brown living at 12 Dawson street.
Michael Reds living at 40 Pearse street.
Joanna Moore living at 62 Collins street.
Ann Brown living at 132 Buffallo street.
```

Se cambio `serialVersionUID` della classe `Student` (dopo aver generato il file), il reader mi lancia `IOException`.

```
java.io.InvalidClassException: Student; local class incompatible: stream classdesc
serialVersionUID = 1, local class serialVersionUID = 2
    at java.io.ObjectStreamClass.initNonProxy(ObjectStreamClass.java:616)
    at java.io.ObjectInputStream.readNonProxyDesc(ObjectInputStream.java:1623)
    at java.io.ObjectInputStream.readClassDesc(ObjectInputStream.java:1518)
    at java.io.ObjectInputStream.readOrdinaryObject(ObjectInputStream.java:1774)
    at java.io.ObjectInputStream.readObject0(ObjectInputStream.java:1351)
    at java.io.ObjectInputStream.readUnshared(ObjectInputStream.java:461)
    at ReadStudentMain.main(ReadStudentMain.java:19)
```

```
public class WriteStudentMain {

   public static void main(String[] args) {
      ArrayList<Student> students= new ArrayList<Student>();
      students.add(new Student("Robert","Brown","Dawson",12));
      students.add(new Student("Michael","Reds","Pearse",40));
      students.add(new Student("Joanna","Moore","Collins",62));
      students.add(new Student("Ann","Brown","Buffallo",132));

      try (ObjectOutputStream out= new ObjectOutputStream(
            new FileOutputStream("students.ser"));) {
         out.writeObject(students);
         for (Student s : students){
            out.writeObject(s);
         }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
}
```

Riscriviamo gli oggetti Student uno per uno

students.ser — Edited

```
¨Ìsrjava.util.ArrayListxÅ“ô«aùIsizexpwsrStudentI
addressNoLfnametLjava/lang/String;Llnameq~L
streetAddressq~xp
tRoberttBrowntDawsonsq~(tMichaeltRedstPearsesq~>tJoa
nnatMooretCollinssq~ÑtAnnq~tBuffalloxq~q~q~
q~
```

312 byte

```
public class ReadStudentMain {

   public static void main(String[] args) {
      try(ObjectInputStream in= new ObjectInputStream(
            new FileInputStream("students.ser"));){
         ArrayList<Student> stds= (ArrayList<Student>) in.readObject();
         for (Student s : stds){
            System.out.println(s);
         }
         Student s;
         while (true){
            try{
            s = (Student) in.readObject();
            System.out.println(s);
            } catch (EOFException e){
               System.out.println("Stream ended");
               break;
            }        }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
   }
}
```

```
Robert Brown living at 12 Dawson street.
Michael Reds living at 40 Pearse street.
Joanna Moore living at 62 Collins street.
Ann Brown living at 132 Buffallo street.
Robert Brown living at 12 Dawson street.
Michael Reds living at 40 Pearse street.
Joanna Moore living at 62 Collins street.
Ann Brown living at 132 Buffallo street.
Stream ended
```

```
for (Student s : students){
                out.writeUnshared(s);
            }
```

students.ser — Edited

```
¨Ìsrjava.util.ArrayListxÅ“ô«aùIsizexpwsrStudentI
addressNoLfnametLjava/lang/String;Llnameq~L
streetAddressq~xp
tRoberttBrowntDawsonsq~(tMichaeltRedstPearsesq~>tJoa
nnatMooretCollinssq~ÑtAnnq~tBuffalloxsq~
q~q~q~sq~(q~  q~
q~
sq~>q~
q~q~sq~Ñq~q~q~
```

392 byte

```
s = (Student) in.readUnshared();
```

Mescolando *read/write object/unshared* otteniamo errori se ci sono riferimenti agli oggetti. Se tutti gli oggetti sono comunque unici, non otteniamo errori. Meglio usare **`readUnshared`** con **`writeUnshared`** e **`readObject`** con **`writeObject`**

Serializable

- Serializzazione *custom* - implementare metodi per scrivere/leggere dal `ObjectOutput/InputStream`

```
private void writeObject(ObjectOutputStream os)

private void readObject(ObjectInputStream is)
```

```java
public class Address {
   int no;
   String street;

   public Address(int n, String street){
      this.no=n;
      this.street=street;
   }

   public String toString()
   {
      StringBuilder sb = new StringBuilder();
      sb.append(this.no);
      sb.append(" ");
      sb.append(this.street);
      sb.append(" street.");
      return sb.toString();
   }
}
```

Classe non serializzabile (e.g. non può essere cambiata perché ereditata da versione precedente del programma)

```java
public class Student implements Serializable{

   private static final long serialVersionUID = 1L;

   private String fname, lname;
   private transient Address address;

   public Student(String fname, String lname,String street, int no){
      this.fname=fname;
      this.lname=lname;
      this.address= new Address(no, street);
   }

   public String toString(){
      StringBuilder sb= new StringBuilder();
      sb.append(this.fname);
      sb.append(" ");
      sb.append(this.lname);
      sb.append(" living at ");
      sb.append(this.address);
      return sb.toString();
   }
```

Deve usare **Address** qui, marcandolo con **transient** perché non serializzabile

```
private void writeObject(ObjectOutputStream out){
   try {
      out.defaultWriteObject();
      out.write(this.address.getNo());
      out.writeUTF(this.address.getStreet());
   } catch (IOException e) {
      System.out.println("Could not write object");
   }
}

private void readObject(ObjectInputStream in){
      try {
         in.defaultReadObject();
         int no =in.readInt();
         String street = in.readUTF();
         this.address= new Address(no, street);
      } catch (IOException e) {
         System.out.println("Could not write object");
      } catch (ClassNotFoundException e) {
         System.out.println("Class not found or wrong version");
      }
}
```

Scrivo gli attributi serializzabili in automatico

Scrivo l'attributo non serializzabile a mano

Inviare oggetti

- Si usano `ObjectInputStream` e `ObjectOutputStream` sopra gli *stream* dei socket.

- Sia il *server* che il cliente devono avere una copia della definizione della classe dell'oggetto

- Metodo più facile: includere nella distribuzione del codice cliente una copia della classe.

Possibilità di *deadlock*

- Aprendo un `ObjectOutputStream` sopra un *stream* di un *socket* (in rete), si invia un *header* con delle informazioni.
- Aprendo un `ObjectInputStream` sopra un stream di un *socket* (in rete), si aspetta il header dall'altra parte.
- Quando si usa `ObjectInputStream` insieme a `ObjectOutputStream`, da entrambe le applicazioni (read e write sul *socket*), è possibile generare *deadlock:*
  - cliente e *server* aprono gli `ObjectInputStream` è si bloccano tutti e due aspettando il header
- Una delle parti deve aprire prima l'*output stream*

Esempio

- Server mantiene un registro studenti
- Cliente chiede al server la lista di studenti con lo stesso cognome
- Server invia la lista

```
public class StudentServer {

    public static void main(String[] args) {
        ArrayList<Student> students= new ArrayList<Student>();
        students.add(new Student("Robert","Brown","Dawson",12));
        students.add(new Student("Michael","Reds","Pearse",40));
        students.add(new Student("Joanna","Moore","Collins",62));
        students.add(new Student("Ann","Brown","Buffallo",132));

        ExecutorService es= Executors.newFixedThreadPool(20);

        try(ServerSocket server= new ServerSocket(1500)){
            while(true){
                System.out.println("Waiting for clients...");
                Socket client=server.accept();
                System.out.println("Client arrived.");
                StudentClientHandler handler= new StudentClientHandler(client,students);
                es.submit(handler);
            }
        } catch (IOException e) {
            System.err.println("Error: "+ e.getMessage());
        } finally {
            es.shutdown();
        }
    }
}
```

```
public class StudentClientHandler implements Runnable {
    Socket client;
    ArrayList<Student> students;
    public StudentClientHandler(Socket client, ArrayList<Student> students) {
        this.students=students;
        this.client=client;
    }
    public void run() {
        System.out.println("Handling client");
        try(ObjectInputStream in= new ObjectInputStream(client.getInputStream());
                ObjectOutputStream out= new ObjectOutputStream(client.getOutputStream());){
            while(true){
                System.out.println("Ready for a new name...");
                String lname=(String) in.readObject();
                System.out.println("Received request for "+lname);
                ArrayList<Student> response= new ArrayList<>();
                for (Student student : this.students){
                    if (student.getLname().trim().toLowerCase().equals(
                            lname.trim().toLowerCase())){
                        response.add(student);
                    }
                }
                out.writeObject(response);
                out.flush();
            }
        } catch (IOException e) {System.err.println("Error: "+ e.getMessage());
        } catch (ClassNotFoundException e) {System.err.println("Class not found: "+
e.getMessage());
        } finally{try {client.close();} catch (IOException e) {}}}}
```

```
public class StudentClient {
    public static void main(String[] args) {
        System.out.println("Connecting to server...");
        try(Socket socket= new Socket(InetAddress.getLocalHost(),1500);
                ObjectInputStream in= new ObjectInputStream(socket.getInputStream());
                ObjectOutputStream out= new ObjectOutputStream(socket.getOutputStream());
                BufferedReader localIn= new BufferedReader(
                                        new InputStreamReader(System.in));)
        {
            System.out.println("Connected.");
            String option=null;
            ArrayList<Student> result=null;
            System.out.println("Insert last name to query server, 'exit' to quit.");
            while( !(option=localIn.readLine()).equals("exit")){
                System.out.println("Asking for students with last name "+option);
                out.writeObject(option);
                out.flush();
                result=(ArrayList<Student>) in.readObject();
                System.out.println("Received "+result.size()+" students with last name "+
option);
                for (Student student: result){
                    System.out.println(student);
                }
                System.out.println("Insert last name to query server, 'exit' to quit.");
            }
        } catch (UnknownHostException e) {System.err.println("Unknown host");
        } catch (IOException e) {System.err.println("Error: "+ e.getMessage());
        } catch (ClassNotFoundException e) {System.err.println("Class not found: "+
e.getMessage());}}}
```

```
Waiting for clients...
Client arrived.
Waiting for clients...
Handling client
```

```
Connecting to server...
```

DEADLOCK

```
public class StudentClient {
    public static void main(String[] args) {
        System.out.println("Connecting to server...");
        try(Socket socket= new Socket(InetAddress.getLocalHost(),1500);
            ObjectOutputStream out= new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream in= new ObjectInputStream(socket.getInputStream());
            BufferedReader localIn= new BufferedReader(
                                new InputStreamReader(System.in));)
        {
            System.out.println("Connected.");
            String option=null;
            ArrayList<Student> result=null;
            System.out.println("Insert last name to query server, 'exit' to quit.");
            while( !(option=localIn.readLine()).equals("exit")){
                System.out.println("Asking for students with last name "+option);
                out.writeObject(option);
                out.flush();
                result=(ArrayList<Student>) in.readObject();
                System.out.println("Received "+result.size()+" students with last name "+
option);
                for (Student student: result){
                    System.out.println(student);
                }
                System.out.println("Insert last name to query server, 'exit' to quit.");
            }
        } catch (UnknownHostException e) {System.err.println("Unknown host");
        } catch (IOException e) {System.err.println("Error: "+ e.getMessage());
        } catch (ClassNotFoundException e) {System.err.println("Class not found: "+
e.getMessage());}}}
```

Swap ->

```
Waiting for clients...
Client arrived.
Waiting for clients...
Handling client
Ready for a new name...
Received request for sirbu
Ready for a new name...
Received request for brown
Ready for a new name...
Received request for Moore
Ready for a new name...
Error: null
```

```
Connecting to server...
Connected.
Insert last name to query server, 'exit' to quit.
sirbu
Asking for students with last name sirbu
Received 0 students with last name sirbu
Insert last name to query server, 'exit' to quit.
brown
Asking for students with last name brown
Received 2 students with last name brown
Robert Brown living at 12 Dawson street.
Ann Brown living at 132 Buffallo street.
Insert last name to query server, 'exit' to quit.
Moore
Asking for students with last name Moore
Received 1 students with last name Moore
Joanna Moore living at 62 Collins street.
Insert last name to query server, 'exit' to quit.
exit
```

Operazioni HTTP

- HTTP - *HyperText Transfer Protocol*
- Usato per offrire pagine *web* (incluso immagini, video, audio) e servizi *web* (e.g. architettura REST - *Representational State Transfer* , RESTful API)
- 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

http://duckweed.isti.cnr.it:5000/top/20160115

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 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.");
            }
```

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

Tutti Immagini Video Notizie Shopping Maps Libri

Circa 524.000 risultati

Qualsiasi Paese
Paese: Italia

Qualsiasi lingua
Pagine in italiano

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

Tutti i risultati
Verbatim

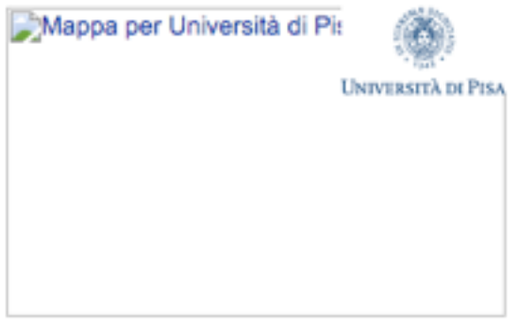
english - Università di Pisa
https://www.unipi.it/index.php/english
A bridge between the Department of Chemistry and KAUST, one of Saudi Arabia's leading universities · Lecturer of the **University of Pisa** is part of the ...
Study Programmes - Students - University - Research

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
Ultime 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.**pisauniversity**press.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

Dati

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

- Per estrarre informazioni dobbiamo percorrere il HTML

- Molti servizi web usano il protocollo HTTP, pero inviano dati in altri format - evitano di inviare tutti i *tag* HTML che non sono importanti

- XML, JSON (*Javascript Object Notation*)

JSON

- Formato di dati *lightweight* e indipendente dalla piattaforma
- Basato su testo
- Usato per interscambiare dati - alternativa a serializzazione - non dipende dal linguaggio di programmazione
- Leggibile e parsabile facilmente
- 2 strutture di base:
  - coppie **chiave:valore**
  - liste ordinate di valori
- Struttura ad albero

Esempio

{“students":[

{"fname":"Robert", “lname”:"Brown", “street”:”Dawson”, “number”:12},

{"fname":"Michael", “lname”:"Reds", “street”:”Pearse”, “number”:40},

{"fname":"Joanna", “lname”:"Moore", “street”:”Collins”, “number”:62},

{"fname":"Ann", “lname”:"Brown", “street”:”Buffalo”, “number”:132},

]}

Sintassi

- Oggetto - attributi rappresentati da coppie **chiave:valore** separate da “,”, delimitate da “{}”
- Array - lista di valori delimitata da “[]”, separati da “ ”
  ,”
- I valori possono essere: stringhe, numeri, boolean, oggetti, array o null
- Le chiavi sono stringhe

JSON in Java

- *package* “JSON simple “
- scaricare *jar* è collegare al progetto come *referenced library (external)*
- offre classi specializzate in creare oggetti JSON dalla loro rappresentazione testuale (deserializzazione) e trasformare oggetti nel formatto JSON di testo (serializzazione)

Tipi di dati

- Stringa JSON -> `String`
- Numero JSON -> `Double, Integer`
- Boolean JSON -> `Boolean`
- Array JSON - > `ArrayList`
- Oggetto JSON - > `HashMap`

JSONObject

- Rappresenta le coppie **chiave:valore** come elementi di una map

- Implementa interfaccia `Map`: metodi `put(chiave, valore)`, `get(chiave)`, etc.

- Metodo per trasformare la *map* in una stringa in formato JSON

```
String toJSONString()
```

- Metodo per scrivere il testo JSON in un *writer*

```
voidwriteJSONString(java.io.Writer out)
```

JSONArray

- Estende `ArrayList` - metodi `set, get, remove` etc
- Metodi per trasformare in testo JSON

```
java.lang.String toJSONString()

voidwriteJSONString(java.io.Writer out)
```

JSONParser

- Trasforma testo JSON in oggetto o array JSON

```
java.lang.Object parse(java.io.Reader in)

java.lang.Object parse(java.lang.String s)
```

Esempio

- Google *search* con JSON
- Google offre un *web service* per fare *search* e ricevere risultato in JSON invece di HTML (*deprecated* però la usiamo per esemplificare l'uso)

`http://ajax.googleapis.com/ajax/services/search/web`

- Deve essere usata solo per fare *search* iniziato da un utente (non bot)

{"responseData":
{“results":
[{"GsearchResultClass":"GwebSearch","unescapedUrl":"https://www.unipi.it/index.php/english","url":"https://www.unipi.it/index.php/english","visibleUrl":"www.unipi.it","cacheUrl":"http://www.google.com/search?q\u003dcache:xzGV1Vs_jrYJ:www.unipi.it","title":"english - Università di \u003cb\u003ePisa\u003c/b\u003e","titleNoFormatting":"english - Università di Pisa","content":"... European and Central Asian universities are closer thanks to TuCAHEA · \nTaylor University students visit the \u003cb\u003eUniversity of Pisa\u003c/b\u003e · Archeologia 2.0, the \nEuropean …"},
{"GsearchResultClass":"GwebSearch","unescapedUrl":"https://en.wikipedia.org/wiki/University_of_Pisa","url":"https://en.wikipedia.org/wiki/University_of_Pisa","visibleUrl":"en.wikipedia.org","cacheUrl":"http://www.google.com/search?q\u003dcache:5qYtSM8XewkJ:en.wikipedia.org","title":"\u003cb\u003eUniversity of Pisa\u003c/b\u003e - Wikipedia, the free encyclopedia","titleNoFormatting":"University of Pisa - Wikipedia, the free encyclopedia","content":"The \u003cb\u003eUniversity of Pisa\u003c/b\u003e is an Italian public research university located in Pisa, Italy. \nIt was founded in 1343 by an edict of Pope Clement VI. It is the 19th oldest …”} etc….
],
"cursor":{"resultCount":"795,000","pages":[{"start":"0","label":1},{"start":"4","label":2},{"start":"8","label":3},{"start":"12","label":4},{"start":"16","label":5},
{"start":"20","label":6},{"start":"24","label":7},{"start":"28","label":8}],"estimatedResultCount":"795000","currentPageIndex":0,"moreResultsUrl":"http://www.google.com/search?oe\u003dutf8\u0026ie\u003dutf8\u0026source\u003duds\u0026start\u003d0\u0026hl\u003den\u0026q\u003dpisa+university","searchResultTime":"0.31"}},
"responseDetails": null,
"responseStatus": 200}

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

            String google_service= "http://ajax.googleapis.com/ajax/services/
search/web?v=1.0&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].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));){

            JSONObject response= (JSONObject) parser.parse(in);
            JSONArray results=(JSONArray)(
                  (JSONObject)(response.get("responseData"))).get("results");
            for (Object result: results){
               System.out.println(((JSONObject)result).get("url"));
            }
         }
   } 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/index.php/english
https://en.wikipedia.org/wiki/University_of_Pisa
https://en.wikipedia.org/wiki/Pisa_University_System
http://www.mastersportal.eu/universities/501/university-of-pisa.html
```

Google search

- Una nuova API è Custom Search (potete provare di usarla da soli):
  - più complesso fare il setup, c'è bisogno di una chiave, ed è a pagamento per più di 100 richieste al giorno
  - le richieste si fanno comunque con GET
  - https://www.googleapis.com/customsearch/v1?parameters

Conclusioni serializzazione

- usando meccanismo Java `Serializable`
    - interoperabilità tra piattaforme - si
    - interoperabilità tra linguaggi - no
    - *heavy weight* - si memorizzano informazioni extra sui tipi di dati
- usando meccanismi universali
    - e.g. JSON
    - interoperabilità tra piattaforme - si
    - interoperabilità tra linguaggi - si
    - *light weight* - si memorizzano solo i dati

Esercizio 1

- Duckweed è un servizio web che fornisce il ranking degli hashtag più popolari su Twitter giorno per giorno. Può essere contattato per richiedere il ranking di un giorno qualsiasi al seguente link: duckweed.isti.cnr.it:5000/top/20160115. La data deve essere passata come parametro nel formato YYYYMMDD, nell'esempio corrisponde a 20160115.

- Scrivere un programma che esegue una richiesta HTTP GET al servizio descritto precedentemente. La data deve essere passata come parametro da linea di comando. Il programma deve fare la richiesta, ricevere la risposta dal servizio in formato JSON, ordinare i risultati secondo il ranking fornito dal servizio e scrivere l'output ordinato.

Esercizio 2

- Trovaprezzi: sviluppare un programma che implementa un servizio trova prezzi e i suoi clienti.

- Consideriamo 3 negozi web diversi che vendono telefonini e smartphone. Ogni negozio ha una lista di prodotti da vendere. Ogni prodotto è descritto da: nome del produttore, modello, prezzo, negozio in cui è venduto. Ogni negozio mette a disposizione un server che offre la lista completa dei prodotti, su richiesta. Tutti e tre i negozi offrono la stessa API, e usano socket per ricevere le richieste. (Tip: scrivere una classe che modella i server, e avviare 3 istanze diverse, una per ogni negozio)

- Un server trova prezzi mantiene una lista di prodotti ed il negozio in cui sono venduti. All'avvio del server, contatta i tre negozi per avere tutte le liste. Ogni giorno il server trova prezzi ricontatta i negozi per aggiornare la lista di prodotti di ognuno. Allo stesso tempo, il server trova prezzi offre un API per i clienti che cercano un prodotto. Un cliente invia il nome di un prodotto al server, che risponde con una lista di prodotti con negozio e prezzo, in ordine crescente. Implementare il server ed i clienti, usando socket. Il server deve essere in grado di gestire più clienti alla volta. L'aggiornamento della lista di prodotti si fa ogni 24 secondi (invece di ogni 24 ore).

- Attenzione: l'accesso alla lista di prodotti del server trova prezzi deve essere thread safe. Usare serializzazione per scambiare i prodotti.