

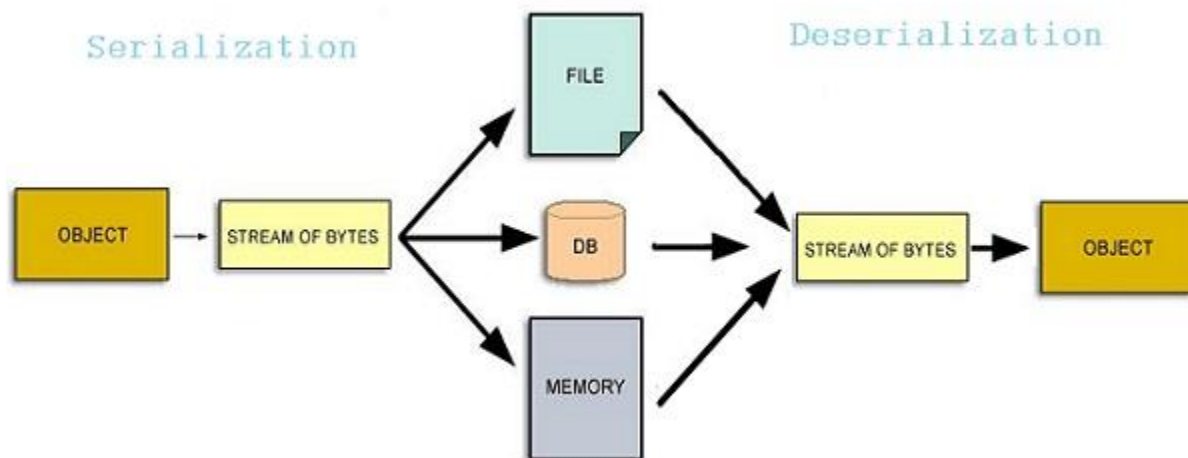
Programmazione avanzata

Lezione 5

Serializzazione oggetti in XML e JSON utilizzando XStream e GSON.

Serializzazione dei dati

La serializzazione dei dati è un processo attraverso il quale delle strutture dati (esempio oggetti o istanze di classi) vengono trasformate in un flusso di bytes il cui formato è facilmente memorizzabile (ad esempio su file) oppure trasmissibile attraverso un canale di comunicazione (e.g. un socket) oppure anche memorizzati su un database. Il processo inverso, quello della de-serializzazione, permette di ricostruire l'oggetto partendo dallo stream di dati. Tale processo è cruciale nel caso in cui si realizzi un'applicazione distribuita in cui le funzionalità offerte agli utenti sono realizzate come la composizione di processi in esecuzione su sistemi differenti che si scambiano dati.



Durante la serializzazione i dati solitamente vengono trasformati secondo un linguaggio di codifica (data encoding language). Linguaggi di codifica standard sono stati definiti in modo da garantire l'interoperabilità tra processi diversi, garantendo uno scambio di informazioni interpretabile in maniera esatta anche nei casi in cui i vari sistemi sono realizzati da sviluppatori differenti. Un esempio di linguaggio di codifica è l'HTML, un linguaggio appositamente definito per codificare contenuti multimediali per pagine web.


Al momento sono due i principali linguaggi di codifica più usati nelle implementazioni: XML e JSON.

Il linguaggio XML – Extensible Markup Language.

XML è un metalinguaggio derivato da una famiglia di linguaggi di markup nati per codificare i documenti web in plain text e trasportarli su http. Lo standard XML definito dal W3C permette tali documenti di essere processati da applicazioni di qualsiasi natura, definendone la struttura mediante XML Schema, e realizzando quindi applicazioni device. La possibilità di definire nuovi elementi (estendibilità) di XML,

rispetto ad HTML, lo rende un metalinguaggio. Attraverso XML, ogni organizzazione può definirsi i propri documenti per le proprie applicazioni e casi d'uso.

```
1. <?xml version="1.0"?>
2. <business-card>
3.   <name>
4.     <title>Ing.</title>
5.     <firstname>Carlo</firstname>
6.     <lastname>Vallati</lastname>
7.   </name>
8.   <organization>
9.     Dip. Ingegneria dell'Informazione
10.  </organization>
11.  <address>
12.    <street>
13.      Via Diotisalvi 2
14.    </street>
15.    <city cap="56122">Pisa</city>
16.    <country>ITALY</country>
17.  </address>
18.  <email>c.vallati@iet.unipi.it</email>
19. </business-card>
20.
```



I dati all'interno di un documento XML possono essere espressi sotto forma di elementi oppure sotto forma di attributi. Nel primo caso i dati vengono memorizzati all'interno di un **TAG** il cui nome caratterizza il campo/dato. Gli **ATTRIBUTI** invece sono dei dati associati con un dato elemento che ad esempio ne specificano una proprietà.

Il linguaggio XML ha una particolarità, include uno strumento di verifica formale del contenuto, i cosiddetti XML schema. Gli XML schema sono dei meta-documenti che descrivono la struttura di un documento affinché questo rappresenti una determinata struttura dati in maniera corretta e completa. Questi schemi permettono di verificare in maniera automatica l'aderenza di un certo documento XML ad uno schema predefinito garantendone non solo la correttezza sintattica ma anche un significato valido.

Il linguaggio JSON – JavaScript Object Notation

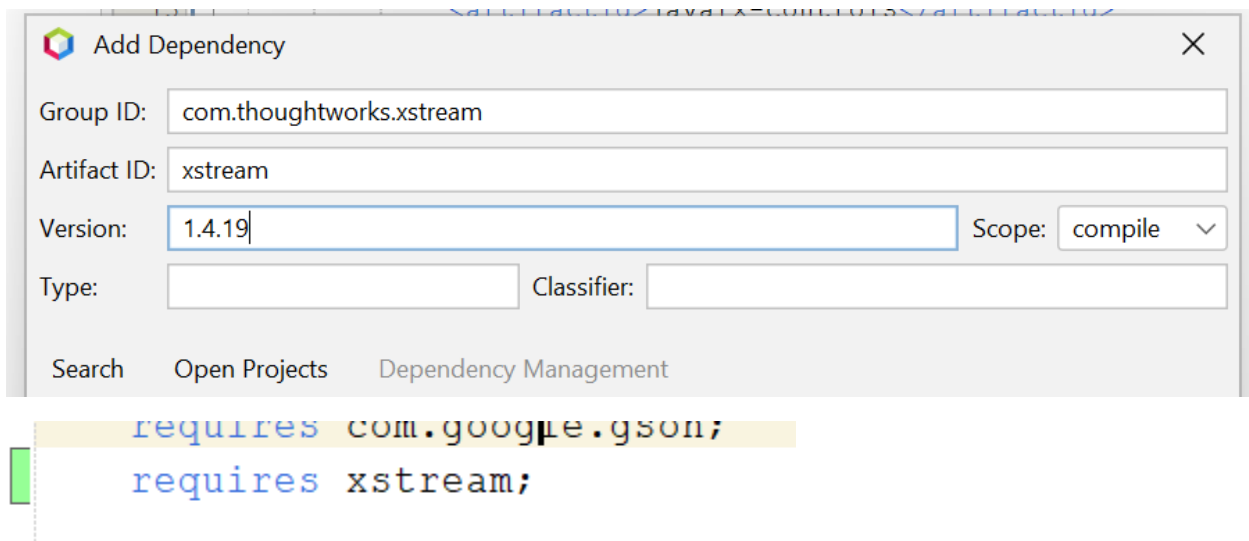
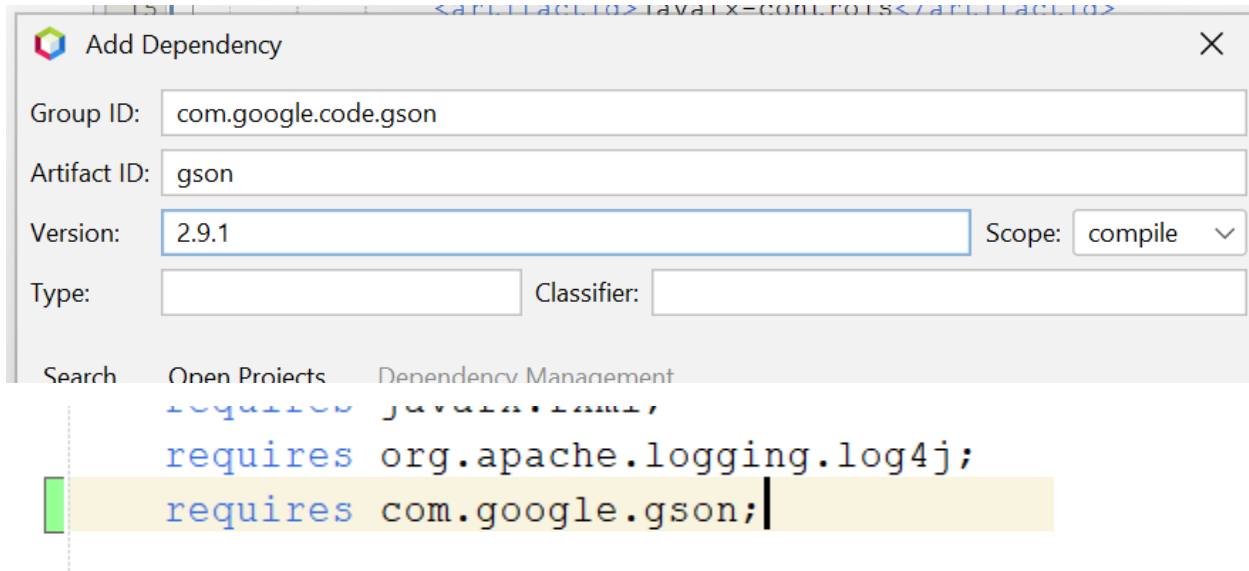
JSON è un formato (alternativo ad XML) per codificare oggetti, nella forma di coppie attributo–valore. Rispetto a XML è più compatto, ed ha regole più semplici. JSON è molto usato nello scambio dati, e nell'archiviazione.

```
1. {
2.   "business-card": {
3.     "name": {
4.       "title": "Ing.",
5.       "firstname": "Carlo",
6.       "lastname": "Vallati"
7.     },
8.     "organization": "Dip. Ingegneria dell'Informazione.",
9.     "address": {
10.      "street": "Via Diotisalvi",
11.      "city": "Pisa",
12.      "country": "ITALY"
13.    },
14.     "email": "c.vallati@iet.unipi.it"
15.   }
16. }
```

Librerie per la serializzazione XML e JSON

Per la serializzazione/de-serializzazione di documenti XML e JSON da/a strutture dati sono disponibili delle librerie che possono essere aggiunte al programma. Le più popolari sono XSTREAM per XML e GSON per JSON.

Al fine di utilizzare tali librerie nel programma c'è bisogno di configurare la dipendenza nel file pom.xml e, nel caso fosse presente, aggiungere il costrutto 'requires' nel file module-info.java.



Libreria GSON

La libreria è in grado di creare un documento JSON a partire da una struttura dati Serializzabile e di ricostruire una struttura dati a partire da un documento JSON. Il documento JSON può essere un file oppure uno stream di dati ricevuto/inviato attraverso un socket.

In alternativa la libreria può essere utilizzata anche per leggere e interpretare un documento JSON leggendo i campi senza avere una struttura dati di riferimento (questo lo vedremo nelle prossime lezioni).

Come prima cosa si deve creare una struttura dati serializzabile. Una struttura dati serializzabile è una struttura che implementa l'interfaccia 'java.io.Serializable' che internamente contiene le funzioni per la serializzazione/deserializzazione di un oggetto. Nel nostro caso creiamo una struttura dati che contenga le credenziali di login dell'utente.

```
1. import java.io.Serializable;
2.
3. public class Credenziali implements Serializable {
4.     public String nome;
5.     public String password;
6.
7.     public Credenziali(String n, String p){
8.         nome = n;
9.         password = p;
10.    }
11. }
12.
```

Una volta definita la struttura dati la sua serializzazione può essere fatta nel seguente modo:

```
1. Credenziali c;
2. c = new Credenziali("carlo", "segreto");
3. Gson gson = new Gson();
4. String serializzato = gson.toJson(c);
5.
```

In particolare, si deve creare un nuovo oggetto Gson (riga 3) e poi si deve invocare la funzione toJson (riga 4).

La deserializzazione avviene invece nel seguente modo:

```
1. Gson gson = new Gson();
2. Credenziali c = gson.fromJson(line, Credenziali.class);
3.
```

Alla riga 2 si richiama la funzione fromJson che interpreta il testo nella variabile string 'line' come JSON e ricostruisce un'istanza della classe Credenziali.

Esercizio

Modificare la app e il server per scambiare le credenziali in formato JSON e non in formato testo.

Libreria XSTREAM

La libreria XSTREAM funziona in maniera simile per la serializzazione:

```
1. XStream xstream = new XStream();
2.
3. xstream.addPermission(AnyTypePermission.ANY);
4.
```

```
5. xstream.alias("credenziali", Credenziali.class);
6.
7. String xml= xstream.toXML(c);
```

In maniera simile per la deserializzazione:

```
8. XStream xstream = new XStream();
9.
10. xstream.addPermission(AnyTypePermission.ANY);
11.
12. xstream.alias("credenziali", Credenziali.class);
13.
14. Credenziali c = (Credenziali) xstream.fromXML(line);
15.
```

Leggermente diversa è la definizione di una classe serializzabile:

```
1. @XStreamAlias("credenziali")
2. public class Credenziali implements Serializable {
3.     public String nome;
4.     public String password;
5.
6.     public Credenziali(String n, String p){
7.         nome = n;
8.         password = p;
9.     }
10. }
```

Esercizio

Modificare la app in maniera tale da realizzare una localizzazione, cioè cambiare il testo dei bottoni in base alla lingua. Il testo da mostrare per ogni bottone deve essere letto da un file XML. Almeno due file XML devono essere creati per l'inglese e l'italiano.

I passi sono i seguenti:

1. Definire una classe 'Linguaggio' che contiene una variabile String per ogni bottone dell'applicazione
2. Modificare una delle funzioni 'initialize' (esempio quella del secondo controller) per caricare un'istanza della classe 'Linguaggio' da file
3. Modificare il testo dei bottoni utilizzando i valori riportati nella classe caricata

Per caricare un file XML da file si può usare il seguente costrutto:

```
1. lang = (Linguaggio) xstream.fromXML(getClass().getResource("languages/linguaggio_en.xml"));
2.
```

la cartella languages può essere creata all'interno di "resources/it/unipi/nomeapp/languages"