# E.1 – JSON

Il **JSON** (JavaScript Object Notation) è un *formato dati* tra i piu’ diffusi, molto popolare perché **language-agnostic** e **leggibile ad occhio umano.**

A screenshot of a computer

Description automatically generated with medium confidence

Un esempio di json generato randomicamente all’indirizzo <https://random-data-api.com/api/v2/users>

La sorgente dati che utilizzeremo è un **file di testo in formato JSON.**

Apri il file *res/raw/movies.txt:*

Graphical user interface, text, application

Description automatically generated

Puoi copiare il testo in un editor JSON per leggerlo meglio, come <https://jsoneditoronline.org>

I campi del json possono contenere:

* Valori:  
  
* Array:  
  A screen with numbers and letters on it

  Description automatically generated with low confidence
* Oggetti:  
  Text

  Description automatically generated with medium confidence

## Serializzazione\Deserializzazione

I Json possono essere convertiti in Istanze di Oggetti con un processo chiamato Serializzazione/Deserializzazione

Serialization è il processo che converte l’istanza di un oggetto in testo formato json  
Deserialization è il processo inverso

A picture containing graphics, logo, circle, graphic design

Description automatically generated

Per convertire una stringa Json a un oggetto strutturato e popolato con tutti i campi e valori presenti nel JSON è necessario creare una data class che funga da modello, che verrà poi utilizzata da una classe di lavoro che effettuerà il pairing tra i dati del JSON e i campi del modello.

## JsonToKotlin

È possibile creare a mano la data class con tutti i relativi membri. Noi useremo un plugin di Android Studio chiamato **JsonToKotlinClass** che ci permetterà di automatizzare il processo evitando errori umani.

File > Settings > Plugins:

Graphical user interface, application

Description automatically generated

Installa e premi *Apply.*

Una volta installato il plugin, apri *data/Entities* il file che contiene le data classes del nostro progetto, e commenta *Movie*: lo ricreeremo grazie a questo plugin.

Graphical user interface, text

Description automatically generated

Ora clicca con il tasto destro > Generate (o ctrl-insert, o alt-k) e seleziona *Kotlin from Json*

A screenshot of a computer

Description automatically generated with medium confidence

Clicca su Advanced e seleziona Default value > Init with non-null

Ora che hai impostato il plugin, procediamo con la creazione della classe modello.

* Copia il contenuto di movies.txt del campo di testo
* In Class name, inserisci MoviesResponse

Spingendo *Generate* avremo questo risultato

Text

Description automatically generated

Il plugin non ha solo generato tutti i membri della classe, ma anche altre classi in base agli oggetti presenti nella struttura Json.

I nomi dei membri della classe sono uguali ai nomi dei campi nel json.

## Gson

GSON è una libreria che automatizza il processo di serializzazione\deserializzazione

Gson utilizzerà ogni membro della classe MoviesResponse per trovare nel Json un campo con lo stesso nome, effettuando la Deserializzazione.

* La libreria è importata nel progetto nel build.gradle:app   
  

### Annotations

Per il nostro caso dobbiamo fare una modifica: la classe *Result* rappresenta l’oggetto *Movie,* quindi rinominiamola, facendo attenzione a rinominare anche il nome del membro *results* in *movies*.

Text

Description automatically generated

Ora il membro *movies* non ha un corrispettivo nel testo del json, che rimane *results*.

Modifichiamo il comportamento della deserializzazione per venire incontro alla nostra modifica, aggiungendo a *movies* una annotazione:

@SerializedName("results") val movies: List<Movie> = *listOf*(),

La Annotation *SerializedName* annuncia al deserializzatore di GSON che dovrà cercare nel json un campo con il nome “results” e inserire il suo valore nel membro *movies*.

*SerializedName* è una delle annotation fornite da GSON.

### Builder

Ora vediamo come convertire il Json in Oggetto.

Apri *MoviesDataSource* dove è stato iniettato il json come stringa

Text

Description automatically generated

Crea una nuova funzione chiamata *requestMovies* con il seguente contenuto:

fun requestMovies(){  
 val movies: MoviesResponse =

Gson().fromJson(moviesJson, MoviesResponse::class.*java*)

return movies  
}

* Gson() crea un’istanza del builder della libreria
* fromJson Deserializza il json *moviesJson* in una istanza della classe *MoviesResponse*.   
  Richiede come parametri:
  + una stringa con il testo in Json
  + una classe che prenderà come modello, indicata tramite [reflection](https://en.wikipedia.org/wiki/Reflective_programming).

*requestMovies* produrrà in output un MoviesResponse correttamente popolato con i dati del nostro file movies.txt

Il processo di serializzazione è simile, e si utilizza la funzione *toJson(objectInstance)*.   
Qui non è necessario fornire un modello di classe – GSON capisce da solo quale è la data class che stiamo usando:

val jsonString: String = Gson().toJson(movies)