



# GSON - Parser für JSON Dokumente

- Eine häufige Aufgabe ist die Umwandlung von Dokumenten im JSON Format in entsprechende JAVA Klassen. Eine von Google entwickelte und bereitgestellte Library namens **GSON** erleichtert diesen Prozess: <https://github.com/google/gson>



GSON - Parser für JSON

Dokumente

Verwendung

Wir wollen  
folgende Java  
Klasse als  
Ausgangsbeispiel  
verwenden:

```
public class IdVal {  
    private int id;  
    private String name;  
    public IdVal(int id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
    public String toString() { return name + ", " + id ;}  
    public int getId() { return this.id; }  
    public void setId(int val) { this.id = val; }  
    public String getName() { return this.name; }  
    public void setName(String val) { this.name = val; }  
}
```



# GSON - Parser für JSON Dokumente

- Um GSON verwenden zu können, muss die Abhängigkeit im Gradle build file eingetragen werden:

```
dependencies {  
    implementation 'com.google.code.gson:gson:2.8.5'  
}
```



# GSON - Parser für JSON Dokumente

Verwendet wird die Klasse dann über die Klasse GSON. Benötigt man weitere Konfiguration, so kann ein GSON Object mithilfe der von GsonBuilder erstellt und anschließend konfiguriert werden.

```
GsonBuilder builder = new GsonBuilder();  
Gson gson = builder.create();
```

# GSON - Parser für JSON Dokumente

## Java nach JSON parsen



```
List<IdVal> idVals = Arrays.asList(  
    new IdVal(1, "Max"),  
    new IdVal(2, "Mizi"),  
    new IdVal(3, "Xaver")  
);  
Gson gson = new Gson();  
String sJson = gson.toJson(idVals);  
Toast.makeText(this, sJson, Toast.LENGTH_LONG).show();
```

Aus einem Java-Objekt (einzelnes Objekt oder auch Liste) kann man mit der Methode `toJson()` sehr einfach einen JSON-String erzeugen.

```
[{"id":1,"name":"Max"}, {"id":2,"name":"Mizi"}, {"id":3,"name":"Xaver"}]
```

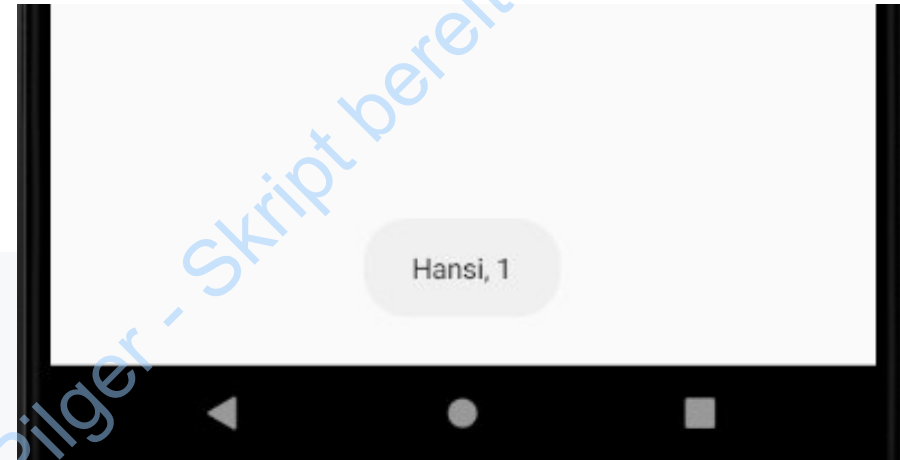
# GSON - Parser für JSON Dokumente

## JSON-Object nach Java



```
String sJson = "{id:1, name:\"Hansi\"}";

Gson gson = new Gson();
IdVal result = gson.fromJson(sJson, idVal.class);
Toast.makeText(this, result.toString(), Toast.LENGTH_LONG).show();
```



- Als Beispiel soll ein einfaches JSON-Object in ein Java-Objekt verwandelt werden. *Voraussetzung ist natürlich, dass die Properties des JSON-Objects den getter/setter-Methoden des Java-Objects entsprechen!*

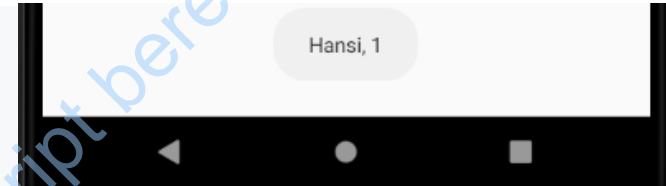
# GSON - Parser für JSON Dokumente

## JSON-Object nach Java



```
String sJson = "{id:1, name:\"Hansi\"}";

Gson gson = new Gson();
IdVal result = gson.fromJson(sJson, idVal.class);
Toast.makeText(this, result.toString(), Toast.LENGTH_LONG).show();
```



Zuerst wird wieder ein Gson-Objekt erzeugt. Dann muss angegeben werden, welchen Java-Typ man sich erwartet. Das erfolgt über den class-Deskriptor des Zieltyps. Mit obigem Ausdruck gibt man eben an, dass in ein Objekt vom Typ `IdVal` transformiert werden soll. Die eigentliche Umwandlung wird dann durch `fromJson()` vollzogen. Etwaige Exceptions müssen natürlich gefangen werden.

# GSON - Parser für JSON Dokumente

JSON-Array nach Java



```
String sJson = "[{id:1, name:\"Hansi\"},  
                {id:2, name:\"Mizi\"},  
                {id:3, name:\"Xaver\"}]";  
  
Gson gson = new Gson();  
TypeToken<List<IdVal>> token = new TypeToken<List<IdVal>>(){};  
List<IdVal> result = gson.fromJson(sJson, token.getType());  
Toast.makeText(this, "Persons count: "+result.size(), Toast.LENGTH_LONG).show();
```

- Die Umwandlung eines JSON-Strings in eine Java-Array erfolgt ähnlich. Ausgangspunkt ist der obige String:
- Den erwarteten Typ der Objekte in der Liste muss man mittels TypeToken angeben.

<https://github.com/google/gson/blob/master/UserGuide.md>