EBook Gratis

APRENDIZAJE gson

Free unaffiliated eBook created from
Stack Overflow contributors.

#gson

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: gson

It is an unofficial and free gson ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official gson.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con gson

Observaciones

Gson es una biblioteca Java de código abierto que se puede usar para convertir objetos Java en su representación JSON. También se puede utilizar para convertir una cadena JSON en un objeto Java equivalente.

Goles para gson

- Proporcionar mecanismos fáciles de usar como toString () y constructor (método de fábrica) para convertir Java a JSON y viceversa
- Permitir que los objetos no modificables preexistentes se conviertan ay desde JSON
- Permitir representaciones personalizadas para objetos
- Soporta objetos arbitrariamente complejos
- Generar salida JSON compacta y legible

El código fuente de Gson está disponible en Github .

Guía del usuario

Examples

Instalación

Para utilizar Gson debes incluirlo en tu proyecto. Puede hacer esto agregando la siguiente dependencia de la versión de Gson disponible en Maven Central:

Maven

Añadir a pom.xml

```
<dependencies>
    <dependency>
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.8.0</version>
      <scope>compile</scope>
    </dependency>
</dependencies>
```

Gradle :

Añadir a build.gradle

```
compile 'com.google.code.gson:gson:2.8.0'
```

Serialización y deserialización.

```
Gson gson = new Gson(); //Create a Gson object
MyType target = new MyType(); //This is the object you want to convert to JSON
String json = gson.toJson(target); // serializes target to Json
MyType target2 = gson.fromJson(json, MyType.class); // deserializes json into target2
```

Arrays

JSON:

```
[
  {
    "id": 8484,
    "name": "David",
    "height": 173.2,
    "weight": 75.42
  },
  {
    "id": 8485,
    "name": "Ronald",
    "height": 183.73,
    "weight": 83.1
  }
]
```

Persona.java

```
public class Person {
    public int id;
    public String name;
    public double height;
    public double weight;

    @Override
    public String toString() {
        return "[ id: " + String.valueOf(id) + ", name: " + name + ", height: " +
String.valueOf(height) + ", weight: " + String.valueOf(weight) + " ]";
    }
}
```

Uso:

```
Gson gson = new Gson();
Person[] persons = gson.fromJson(json, Person[].class);
for(Person person : persons)
    System.out.println(person.toString());
```

Salida:

```
[ id: 8484, name: David, height: 173.2, weight: 75.42 ]
```

```
[ id: 8485, name: Ronald, height: 183.73, weight: 83.1 ]
```

Ejemplo simple

La biblioteca Gson proporciona `Gson.class` que maneja todas las conversiones entre objetos Java y JSON. Se puede crear una instancia de esta clase invocando al constructor predeterminado. Por lo general, le gustaría tener una instancia de Gson para la mayor parte de las operaciones en su programa.

```
Gson gson = new Gson();
```

Primero, necesitamos crear una clase de nuestro objeto con el que trabajaremos

```
class Person {
   public String name;
   public int age;

   public Person(String name, int age){
       this.name = name;
       this.age = age;
   }
}
```

La clase Gson proporciona los métodos `toJson` y `fromJson` que son los principales puntos de entrada para los objetos JSON y java.

Intentemos convertir el objeto java a JSON y volver al objeto java

```
Person person = new Person("Jason", 29);
//using gson object which we created earlier
String json = gson.toJson(person);
System.out.println(json);
//Outputs: {"name": "Jason", "age": 29}
```

Y ahora de nuevo

```
String json = "{\"name\": \"Jason\", \"age\": 29}";
Person person = gson.fromJson(json, Person.class);
System.out.println(person.age + "yo " + person.name + " walks into a bar");
//Outputs "29 yo Jason walks into a bar"
```

Convierte String a JsonObject sin POJO

```
String jsonStr = "{\"name\" : \"Abcd\", \"greeting\": \"Hello\", }"; //Sample Json String

Gson gson = new Gson(); // Creates new instance of Gson
JsonElement element = gson.fromJson (jsonStr, JsonElement.class); //Converts the json string
to JsonElement without POJO
JsonObject jsonObj = element.getAsJsonObject(); //Converting JsonElement to JsonObject

String name = jsonObj.get("name").getAsString(); //To fetch the values from json object
```

```
String greeting = jsonObj.get("greeting").getAsString();
```

Usando GSON con herencia

GSON no admite herencia nuestra de la caja. Digamos que tenemos la siguiente jerarquía de clases:

```
public class BaseClass {
    int a;

    public int getInt() {
        return a;
   }
}

public class DerivedClass1 extends BaseClass {
     int b;

     @Override
     public int getInt() {
         return b;
     }
 }

public class DerivedClass2 extends BaseClass {
    int c;

    @Override
    public int getInt() {
        return c;
    }
}
```

Y ahora queremos serializar una instancia de `DerivedClass1` a una cadena json

```
DerivedClass1 derivedClass1 = new DerivedClass1();
derivedClass1.b = 5;
derivedClass1.a = 10;

Gson gson = new Gson();
String derivedClass1Json = gson.toJson(derivedClass1);
```

Ahora, en otro lugar, recibimos esta cadena json y queremos deserializarla, pero en tiempo de compilación solo sabemos que se supone que es una instancia de `BaseClass` :

```
BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
System.out.println(maybeDerivedClass1.getInt());
```

Pero GSON no sabe que `derivedClass1Json` fue originalmente una instancia de `DerivedClass1` , por lo que esto imprimirá 10.

¿Cómo resolver esto?

Necesita crear su propio `JsonDeserializer` , que maneja tales casos. La solución no está

perfectamente limpia, pero no pude encontrar una mejor.

Primero, agregue el siguiente campo a su clase base

```
@SerializedName("type")
private String typeName;
```

E inicialízalo en el constructor de la clase base.

```
public BaseClass() {
    typeName = getClass().getName();
}
```

Ahora agregue la siguiente clase:

```
public class JsonDeserializerWithInheritance<T> implements JsonDeserializer<T> {

 @Override
 public T deserialize(
     JsonElement json, Type typeOfT, JsonDeserializationContext context)
     throws JsonParseException {
     JsonObject jsonObject = json.getAsJsonObject();
     JsonPrimitive classNamePrimitive = (JsonPrimitive) jsonObject.get("type");

     String className = classNamePrimitive.getAsString();

     Class<?> clazz;
     try {
     clazz = Class.forName(className);
     } catch (ClassNotFoundException e) {
     throw new JsonParseException(e.getMessage());
     }
     return context.deserialize(jsonObject, clazz);
 }
}
```

Todo lo que queda por hacer es conectar todo

```
GsonBuilder builder = new GsonBuilder();
 builder
 .registerTypeAdapter(BaseClass.class, new JsonDeserializerWithInheritance<BaseClass>());
 Gson gson = builder.create();
```

Y ahora, ejecutando el siguiente código-

```
 DerivedClass1 derivedClass1 = new DerivedClass1();
 derivedClass1.b = 5;
 derivedClass1.a = 10;
 String derivedClass1Json = gson.toJson(derivedClass1);

 BaseClass maybeDerivedClass1 = gson.fromJson(derivedClass1Json, BaseClass.class);
 System.out.println(maybeDerivedClass1.getInt());
```

Se imprimirá 5.

Lea Empezando con gson en línea: https://riptutorial.com/es/gson/topic/4804/empezando-con-gson

Capítulo 2: Usando Gson con JAX-RS (servicios web RESTful)

Examples

Proveedor de JAX-RS para utilizar Gson

Este es un JAX-RS `@Provider` para usar Gson como el analizador JSON. El ejemplo también muestra cómo utilizar convertidores de fecha / hora personalizados de Java 8.

```
@Provider
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)
public class JerseyServerGson
        implements MessageBodyWriter<Object>, MessageBodyReader<Object>
{
  @Override
  public boolean isReadable(Class<?> type,
                            Type genericType,
                            Annotation[] annotations,
                            MediaType mediaType)
  {
    return true;
  }

  @Override
  public Object readFrom(Class<Object> type,
                         Type genericType,
                         Annotation[] annotations,
                         MediaType mediaType,
                         MultivaluedMap<String, String> httpHeaders,
                         InputStream entityStream)
        throws IOException, WebApplicationException
  {
    try ( InputStreamReader input =
                new InputStreamReader(entityStream, "UTF-8") ) {
      Gson gson = getGson();
      return gson.fromJson(input, genericType);
    }
  }

  @NotNull
  private Gson getGson() {
    return new GsonBuilder()
            .registerTypeAdapter(LocalDateTime.class,
                                 new AdapterLocalDateTime().nullSafe())
            .registerTypeAdapter(LocalDate.class,
                                 new AdapterLocalDate().nullSafe())
            .setPrettyPrinting()
            .serializeNulls()
            .create();
  }

  @Override
```

```
  public boolean isWriteable(Class<?> type,
                             Type genericType,
                             Annotation[] annotations,
                             MediaType mediaType)
  {
    return true;
  }

  @Override
  public long getSize(Object o,
                      Class<?> type,
                      Type genericType,
                      Annotation[] annotations,
                      MediaType mediaType)
  {
    // Deprecated and ignored in Jersey 2
    return -1;
  }

  @Override
  public void writeTo(Object o,
                      Class<?> type,
                      Type genericType,
                      Annotation[] annotations,
                      MediaType mediaType,
                      MultivaluedMap<String, Object> httpHeaders,
                      OutputStream entityStream)
          throws IOException, WebApplicationException
  {
    try ( OutputStreamWriter writer =
                  new OutputStreamWriter(entityStream, "UTF-8") ) {
      getGson().toJson(o, genericType, writer);
    }
  }
}
```

Lea Usando Gson con JAX-RS (servicios web RESTful) en línea: https://riptutorial.com/es/gson/topic/4893/usando-gson-con-jax-rs--servicios-web-restful-

Creditos

S. No	Capítulos	Contributors
1	Empezando con gson	Community, Daniil Dubrovsky, Derlin, Egor Neliuba, Ginandi, James, Maverick, pr0gramist, Uttam
2	Usando Gson con JAX-RS (servicios web RESTful)	Derlin, sargue