

## СЕРІАЛІЗАЦІЯ ТА ДЕСЕРІАЛІЗАЦІЯ JSON

Розглянемо приклад. Нехай у проекті існує клас (файл Foo.java):

```
public class Foo {  
    private int width;  
    private int height;  
    private int depth;  
    public Foo(int width, int height, int depth){  
        this.width = width;  
        this.height = height;  
        this.depth = depth;  
    }  
}
```

Нехай існує екземпляр класу Foo:

```
Foo fooInstance = new Foo(1, 2, 3);
```

Завдання: представити вміст полів екземпляра класу Foo у форматі JSON. Забезпечити зворотне перетворення: створити екземпляр класу Foo, маючи рядок у форматі JSON, котрий задає значення полів екземпляра класу Foo.

Для роботи необхідно завантажити і встановити бібліотеку серіалізації Java — GSON: [проект на GitHub](#). [Gson 2.6.1 Download](#) на Maven Central. Версія проекту на момент написання цього файлу: 2.6.1.

GSON:

- надає простих toJson() та fromJson() методів для перетворення Java-об'єктів у формат JSON і навпаки;
- дозволяє перетворювати у- та з формату JSON вже існуючі незмінювані об'єкти;
- підтримує Java Generics;
- дозволяє налаштовувати представлення об'єктів;
- підтримує об'єкти довільної складності (з глибокою ієрархією та широким використанням generic-типів).

Створіть проект. Виберіть у IntelliJ IDEA пункт меню File -> Project Structure:

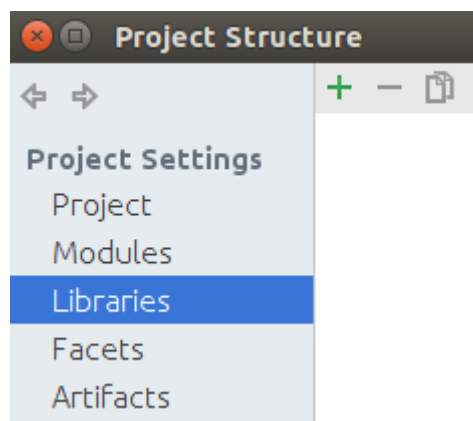


Рисунок 1 — Project Structure

Виберіть з меню Project Settings пункт Libraries. Натисніть “+”. З випадаючого списку New Project Library виберіть Java і вкажіть шлях до попередньо збереженого файлу [gson-2.6.1.jar](#) (чи новішого).

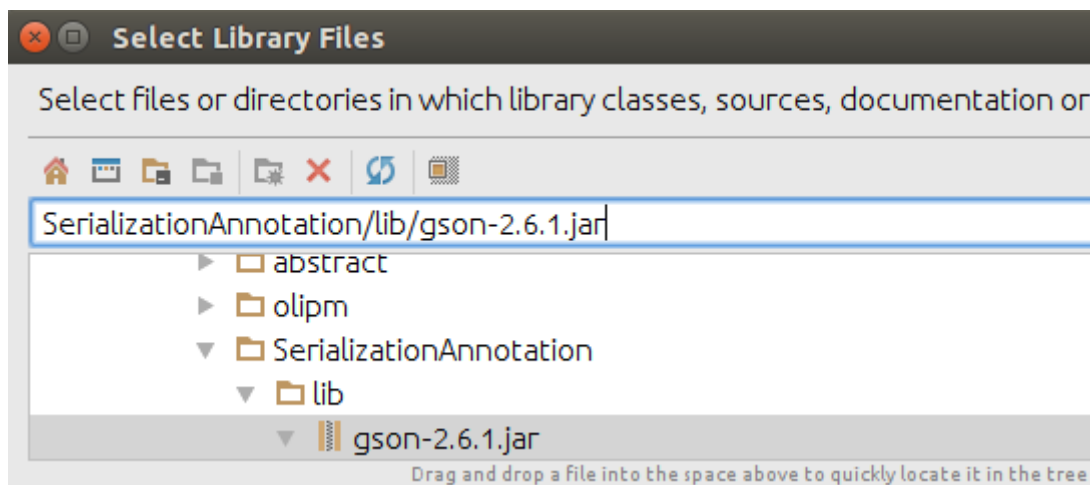


Рисунок 2 — Додавання бібліотеки до проекту уручну

Бібліотеку можна додати без попереднього завантаження, за допомогою Maven. Для цього виберіть з меню Project Settings пункт Libraries. Натисніть “+”. З випадаючого списку New Project Library виберіть пункт From Maven.

Введіть у поле пошуку рядок `com.google.code.gson:gson:2.6.1` (за потреби — вкажіть потрібну версію) і натисніть кнопку пошуку. Відзначивши Download to, можна вказати шлях для завантаження бібліотеки. Нехай, наприклад, це буде папка lib проекту.

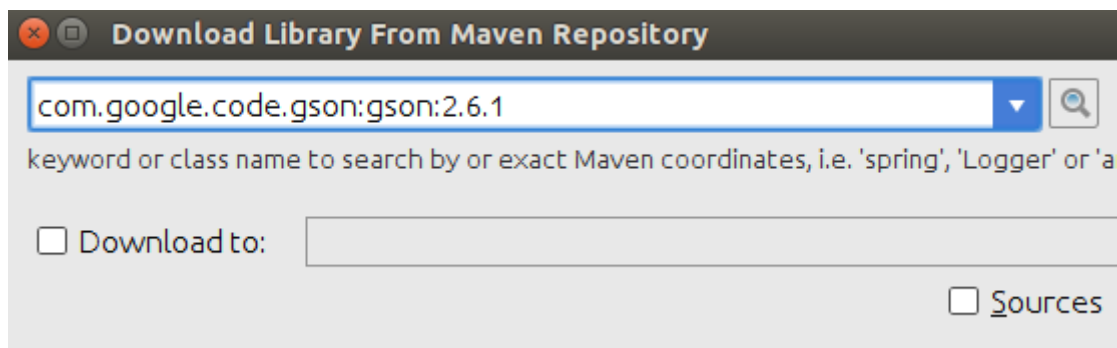


Рисунок 3 — Завантаження бібліотеки з Maven репозиторію

Створіть у проекті клас із довільним іменем, нехай — Bar:

```
import com.google.gson.Gson;
public class Bar {
    public static String serializeToJson(Foo fooInstance) {
        Gson gson = new Gson();
        String j = gson.toJson(fooInstance);
        return j;
    }
    public static Foo deserializeFromJson(String jsonString) {
```

```

    Gson gson = new Gson();
    Foo myBox = gson.fromJson(jsonString, Foo.class);
    return myBox;
}
public static void main(String[] args) {
    Foo fooInstance = new Foo(1, 2, 3);
    System.out.println(serializeToJson(fooInstance));
    fooInstance = deserializeFromJson("{\"width\":4,\"height\":5,\"depth\":6}");
    System.out.println(serializeToJson(fooInstance));
}
}

```

Результат:

```

{"width":1,"height":2,"depth":3}
{"width":4,"height":5,"depth":6}

```

Клас Bar містить два статичних методи: `public static String serializeToJson(Foo fooInstance)` та `public static Foo deserializeFromJson(String jsonString)`.

Перший як параметр приймає посилання на екземпляр класу Foo і повертає рядок у форматі JSON, другий — навпаки: працює із рядком-параметром у форматі JSON і повертає посилання на екземпляр класу Foo.

Зверніть увагу на те, що поля у JSON форматі мають ті ж назви, що і поля класу. За потреби їх можна звільнити на довільні за допомогою анотацій. Змінимо клас Foo:

```

import com.google.gson.annotations.SerializedName;
public class Foo {
    @SerializedName("w")
    private int width;
    @SerializedName("h")
    private int height;
    @SerializedName("d")
    private int depth;
    public Foo(int width, int height, int depth){
        this.width = width;
        this.height = height;
        this.depth = depth;
    }
}

```

Відповідно, у класі Bar змінимо параметр методу `deserializeFromJson`

```
з ("{"width":4,\"height\":5,\"depth\":6}"); на ("{"w":4,\"h\":5,\"d\":6}");
```

Результат:

```
{"w":1,"h":2,"d":3}
```

```
{"w":4,"h":5,"d":6}
```

Розглянемо випадок вибіркової серіалізації полів класу (детальніше: <https://google-gson.googlecode.com/svn/trunk/gson/docs/javadocs/com/google/gson/annotations/Expose.html>).

Скористаємося типом анотації `Expose`. Дана анотація вказує на те, що член класу підлягає серіалізації або десеріалізації:

`@Expose(serialize = true)` або `@Expose()` використовують, якщо член класу повинен бути серіалізований або десеріалізований.

`@Expose(serialize = false)` — член класу не підлягає серіалізації або десеріалізації.

Аналогічно записують для десеріалізації (замість `serialize` — `deserialize`; обох можна поєднувати за допомогою коми). Значення за замовчуванням — `true`.

Така анотація не дасть ефекту, доки екземпляр `Gson` не буде побудовано з `GsonBuilder` та виконано метод `GsonBuilder.excludeFieldsWithoutExposeAnnotation()`.

Таким чином, класи `Foo` та `Bar` виглядатимуть наступним чином.

```
import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;
public class Foo {
    @SerializedName("w")
    @Expose
    private int width;
    @SerializedName("h")
    @Expose(serialize = true, deserialize = true)
    private int height;
    @SerializedName("d")
    @Expose(serialize = false, deserialize = false)
    private int depth;
    public Foo(int width, int height, int depth){
        this.width = width;
        this.height = height;
        this.depth = depth;
    }
}
```

```
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
public class Bar {
    public static String serializeToJson(Foo fooInstance) {
        Gson gson = new GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
        String j = gson.toJson(fooInstance);
        return j;
    }
}
```

```

public static Foo deserializeFromJson(String jsonString) {
    Gson gson = new GsonBuilder().excludeFieldsWithoutExposeAnnotation().create();
    Foo myBox = gson.fromJson(jsonString, Foo.class);
    return myBox;
}
public static void main(String[] args) {
    Foo fooInstance = new Foo(1, 2, 3);
    System.out.println(serializeToJson(fooInstance));
    fooInstance = deserializeFromJson("{\"w\":4,\"h\":5,\"d\":6}");
    System.out.println(serializeToJson(fooInstance));
}
}

```

Результат:

```

{"w":1,"h":2}
{"w":4,"h":5}

```

Такий підхід корисний, якщо потрібно явно вказати опції серіалізації для кожного члена класу окремо. Якщо потрібно тільки виключити певне поле з серіалізації/десеріалізації, достатньо позначити його як `transient`, наприклад:

```
private transient int depth;
```

Модифікуємо класи `Foo` та `Bar`.

```

import com.google.gson.annotations.Expose;
import com.google.gson.annotations.SerializedName;
public class Foo {
    @SerializedName("w")
    @Expose
    private int width;
    @SerializedName("h")
    @Expose(serialize = true, deserialize = true)
    private int height;
    private transient int depth;
    public Foo(int width, int height, int depth){
        this.width = width;
        this.height = height;
        this.depth = depth;
    }
}

```

```
import com.google.gson.Gson;
public class Bar {
    public static String serializeToJson(Foo fooInstance) {
        Gson gson = new Gson();
        String j = gson.toJson(fooInstance);
        return j;
    }
    public static Foo deserializeFromJson(String jsonString) {
        Gson gson = new Gson();
        Foo myBox = gson.fromJson(jsonString, Foo.class);
        return myBox;
    }
    public static void main(String[] args) {
        Foo fooInstance = new Foo(1, 2, 3);
        System.out.println(serializeToJson(fooInstance));
        fooInstance = deserializeFromJson("{\"w\":4,\"h\":5,\"d\":6}");
        System.out.println(serializeToJson(fooInstance));
    }
}
```

Результат:

```
{"w":1,"h":2}
{"w":4,"h":5}
```