Jason Palmeri
6/6/22
CSD420
Module 11

Google's GSON

While searching around for a JSON API for Java I came across a Stackoverflow post asking which API should be used? The replies gave a plethora of information and the one API library that stood out to me was Google's GSON library. **"GSON is a Java library that can be used to convert Java Objects into their JSON representation. It can also be used to convert a JSON string to an equivalent Java Object"** (Google). While there are many other open-source projects that allow you to convert Java objects to JSON, GSONs goals were focused on allowing the use of Generics. From the official github GSON states it's goals as:

- **Provide simple toJson() and fromJson() methods to convert Java objects to JSON and vice-versa**
- **Allow pre existing unmodifiable objects to be converted to and from JSON**
- **Extensive support of Java Generics**
- **Allow custom representations for objects**
- **Support arbitrarily complex objects**

GSON has a few minimum requirements to be used. For GSON 2.9.0 and newer it requires Java 7, and for GSON 2.8.9 and older it requires Java 6. GSON provides three ways to process JSON; First is their **Streaming API**, which reads and writes JSON content as discrete events using the **JsonReader** and **JsonWriter** methods. The **Streaming API** is considered the **"most powerful approach among the three approaches to process JSON. It has the lowest overhead and it is quite fast in read/write operations"** (*GSON - Quick Guide*). Next up is the **Tree Model**, which prepares an in-memory tree representation of the JSON document. Using **JsonObject** nodes it builds a tree, the **Tree Model** is a flexible approach. Finally there is the **Data Binding** approach, which converts JSON to and from POJO (Plain Old Java Object) using the property accessor. GSON reads/writes JSON using data type adapters in this approach.

To get started parsing JSON we can create a simple User class:

```
Public class User() {
        String name;
        String email;
        Int age;
}
```

```
User myUser = new User("Jason", "jason@example.com", 22);
```

Now that we have our user we can create an instance of the GSON class:

Gson g = new Gson();

Now we can convert our object **myUser** to JSON using GSON's **toJson** method:

String myUserJson = g.toJson(myUser);

Printing out **myUserJson** would output something like this:

```
{
	“Age”: 22,
	“Email”: “jason@example.com”,
	“Name”: “Jason”
}
```

GSON will automatically sort the data alphabetically, and keep the types correctly, age is an int and email/name is a string. GSON also allows us to convert JSON to a Java object:

String x = “{‘name’: ‘jason’, ‘age’: ‘22’, ‘email’: ‘jason@example.com’}”;

First we create the JSON in string format, then we can just call the fromJson method giving it out string, and the class we are converting it to:

User newUserFromJson = g.fromJson(x, User.class);

This library is free to use and seems to be extremely useful for easy JSON manipulation in Java, If I had to pick a Java JSON API this would be at the top of my list for how easy it is to use, and for how well document it is, also it is open-source and managed by Google, making sure that it is kept up to date, and well maintained.

Citations

Google, G. (n.d.). *Google/gson: A java serialization/deserialization library to convert java objects into JSON and back*. GitHub. Retrieved June 7, 2022, from https://github.com/google/gson

GSON - Quick Guide. Tutorials Point. (2022). Retrieved June 7, 2022, from https://www.tutorialspoint.com/gson/gson_quick_guide.htm

Muhammad Maqsoodur RehmanMuhammad Maqsoodur Rehman 32.7k3434 gold badges8181 silver badges124124 bronze badges, S. (2010, April 1). *How to parse JSON in Java*. Stack Overflow. Retrieved June 7, 2022, from https://stackoverflow.com/questions/2591098/how-to-parse-json-in-java

Norman Peitek on May 19 2016. (2019, May 19). *GSON - getting started with Java-JSON Serialization & Deserialization*. Future Studio. Retrieved June 7, 2022, from https://futurestud.io/tutorials/gson-getting-started-with-java-json-serialization-deserialization