
Abstractions


Gson

- Gson Provides a layer of abstraction over JSON and Java Objects
- Convert JSON to a Java object
- Convert a Java object to JSON

Gson Goals

- 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

Code

Json Model

```
{
  "login": "octocat",
  "id": 1,
  "avatar_url": "https://github.com/images/error/octocat_happy.gif",
}
```

Gson Model Object

```
//GithubUser.java

public class Track {
    @SerializedName("login")
    private String mLogin;

    @SerializedName("id")
    private String mId;

    //Without SerializedName
    private String avatar_url;

    public GithubUser(String login, String id, String avatarUrl){
        mLogin = login;
        mId = id;
        mAvatarUrl = avatarUrl;
    }
}
```

Gson code

```
Gson gson = new GsonBuilder().setPrettyPrinting().create();
GithubUser user = gson.fromJson(jsonString, GithubUser.class); //to object

String githubUserString = gson.toJson(mTrack); //to JSON string
```

Callback Interfaces

- Callbacks rely upon interfaces to guarantee that methods are implemented
- Used to communicate back to a concerned object (often a controller)

Steps

1. Create an interface with a callback method. ex. `void onItemClicked(int index)`
2. Implement the interface in the concerned class. ex. in Activity `onCreate` implement annonymous inner class
3. Configure Object ex. List to callback to the onItemClicked Method when item has been clicked

Retrofit

Uses Callbacks, Interfaces, and an abstraction over REST API methods to simplify network code

Interface

Create an interface to define each of the clients actions

RestAdapter

Use Rest adapter to define the Java Class, and kick off the API Call