Retrofit

http://square.github.io/retrofit/

Retrofit2 + OkHttp3 + Jackson + RxAndroid2/RxJava2

Tips, Restful, web

Library

https://mvnrepository.com/artifact/com.squareup.okhttp3/okhttp
https://mvnrepository.com/artifact/com.squareup.okhttp3/okhttp-urlconnection

```
compile 'com.squareup.okhttp3:okhttp:3.8.0'
compile 'com.squareup.okhttp3:okhttp-urlconnection:3.8.0'
compile 'com.squareup.okhttp3:logging-interceptor:3.8.0'

compile 'com.squareup.retrofit2:retrofit:2.3.0'
compile 'com.squareup.retrofit2:converter-jackson:2.3.0'
compile 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'

compile 'com.fasterxml.jackson.core:jackson-core:2.8.8'
compile 'com.fasterxml.jackson.core:jackson-databind:2.8.8.1'
compile 'com.fasterxml.jackson.core:jackson-annotations:2.8.8'

//Android
compile 'io.reactivex.rxjava2:rxandroid:2.0.1'
//Java
compile 'io.reactivex.rxjava2:rxjava:2.1.0'
```

Source

```
public class RestfulAdapter {

    public static final int CONNECT_TIMEOUT = 15;
    public static final int WRITE_TIMEOUT = 15;
    public static final int READ_TIMEOUT = 15;

    private static volatile ConcurrentHashMap<String, Object> services = new
ConcurrentHashMap<>();

    public static synchronized <T> T getInstance(Class<T> restfulInterface)
throws Exception {
        if(!restfulInterface.isInterface())
            throw new IllegalArgumentException("argument must be
```

```
INTERFACE");

        T service = (T) services.get(restfulInterface.getName());

        if(service == null) {
            if(!restfulInterface.isAnnotationPresent(URL.class))
                throw new Exception("Must be annotated URL");
            URL urlAnnotation =
restfulInterface.getDeclaredAnnotation(URL.class);
            String url = urlAnnotation.value();
            if(url.length() <= 0 || !url.substring(0, 4).equals("http"))
                throw new Exception("Unavailable URL value");
            boolean isUnwrapRootValue = urlAnnotation.isUnwrappedRoot();

            HttpLoggingInterceptor httpLoggingInterceptor = new
HttpLoggingInterceptor();
httpLoggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY);

            CookieManager cookieManager = new CookieManager();
            cookieManager.setCookiePolicy(CookiePolicy.ACCEPT_ALL);

            OkHttpClient client = configureClient() // 인증서 무시
                    .connectTimeout(CONNECT_TIMEOUT, TimeUnit.SECONDS)
                    .writeTimeout(WRITE_TIMEOUT, TimeUnit.SECONDS)
                    .readTimeout(READ_TIMEOUT, TimeUnit.SECONDS)
                    .cookieJar(new JavaNetCookieJar(cookieManager)) //쿠키 설
정
                    .addInterceptor(httpLoggingInterceptor) //http 로그 확인
                    .build();

            service = new Retrofit.Builder()
                    .baseUrl(url)
                    .client(client)
.addCallAdapterFactory(RxJava2CallAdapterFactory.create())
.addConverterFactory(getConverterFactory(isUnwrapRootValue))
                    .build().create(restfulInterface);

            services.put(restfulInterface.getName(), service);
        }

        return service;
    }

    private static JacksonConverterFactory getConverterFactory(boolean
isUnwrapRootValue) {
        ObjectMapper objectMapper = new ObjectMapper();
        objectMapper.registerModule(new JodaModule());
objectMapper.enable(MapperFeature.ACCEPT_CASE_INSENSITIVE_PROPERTIES);
objectMapper.disable(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES);
        if(isUnwrapRootValue)
            objectMapper.enable(DeserializationFeature.UNWRAP_ROOT_VALUE);
```

```
// JsonRootName과 연계
        return JacksonConverterFactory.create(objectMapper);
    }
    /**
  * 인증서 무시
     * @return
     */
    private static OkHttpClient.Builder configureClient() throws Exception {
        TrustManagerFactory trustManagerFactory =
TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        trustManagerFactory.init((KeyStore) null);
        TrustManager[] trustManagers =
trustManagerFactory.getTrustManagers();

        if (trustManagers.length != 1 || !(trustManagers[0] instanceof
X509TrustManager))
            throw new IllegalStateException("Unexpected default trust
managers:" + Arrays.toString(trustManagers));

        X509TrustManager trustManager = (X509TrustManager) trustManagers[0];

        SSLContext sslContext = SSLContext.getInstance("SSL");
        sslContext.init(null, new TrustManager[] { trustManager }, new
SecureRandom());

        OkHttpClient.Builder builder = new OkHttpClient.Builder();
        builder.sslSocketFactory(sslContext.getSocketFactory(),
trustManager).hostnameVerifier((hostname, session) -> true);

        return builder;
    }
}
```

```
@URL(value = "http://xxx.com", isUnwrappedRoot = true)
public interface DemoService {
    @GET("{page}")
    Observable<ContentsVO> read(@Path("page") long page);

    @PUT("{no}")
    Observable<ResponseBody> modify(@Path("no") long no, @Body ServiceVO
serviceVO);

    @Multipart
    @POST(URL)
    Observable<ResponseBody> write(@PartMap Map<String, RequestBody>
params);

    @DELETE("{no}")
```

```
    Observable<ResponseBody> delete(@Path("no") long no);
}
```

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface URL {
    String value() default "";
    boolean isUnwrappedRoot() default false;
}
```