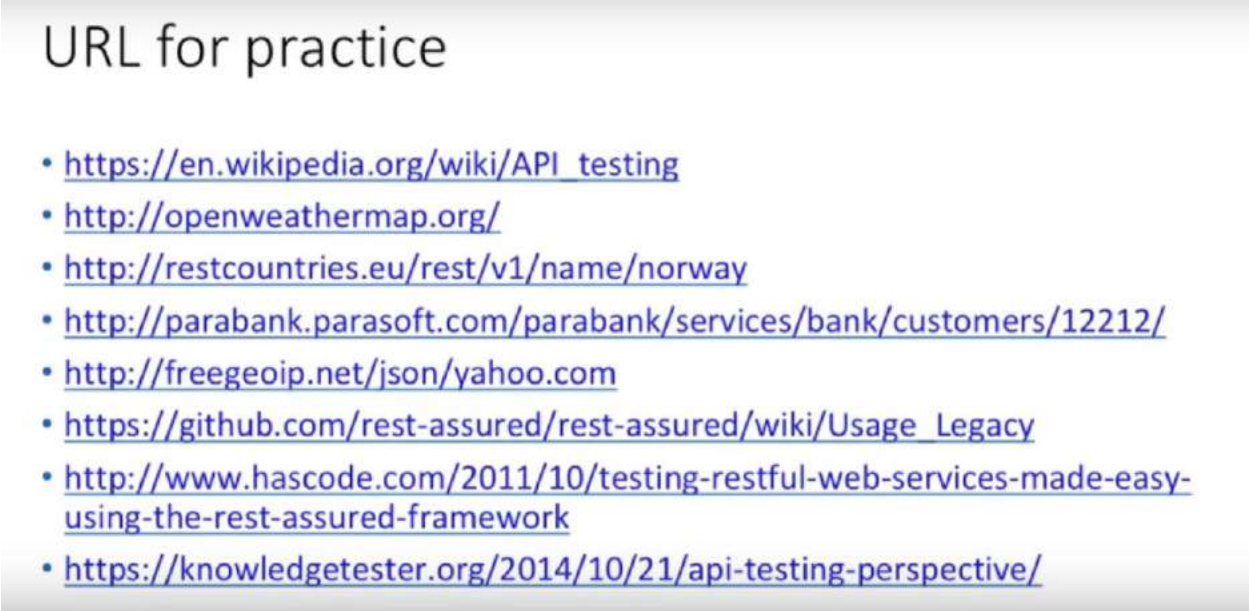
Rest Assured is easiest to learn & use API testing method/library.

Its so easy that our grandmother can also learn it.

Rest Assured supports multiple authentication schemes.

For example Oauth,digest,certificate,form and preemptive basic authentication



Use JSON objects or POJOs in back end service?

Jackson is probably the most popular open-source library for JSON serialization/deserialization in Java.

DTO: In the situation where some amount of transformation is needed between the data received from your customer database and what you actually return to the caller, I usually create a separate DTO (data-transfer object) class that contains only the fields to be serialized.

On the other hand, if you always need to remove the same fields from the customer data before transmittal and don't mind putting serialization-specific annotations on your POJOs, you can use Jackson's @JsonIgnore (or similar) on those fields and skip the DTO.

DTO PROS - Straight forward and verbose code POJO's. - Suitable for more complex transformations. - Easily testable.

DTO CONS - DTO's are models held in code, they cannot changed without manipulating code. - More code as DTO and there population code are directly related to the number of mobile client variants.

JSONObject PROS
- Straight forward code. - Easily testable, it's a function on external factors. - Less code as one shoe fits all, not directly related to the number of mobile client variants
- There is no model held in code, the list of JSON fields for each mobile client can be held in configuration externally.

JSONObject CONS
- Not suitable for more complex transformations. - JSONObject requires navigation JSON model, not so nice to ready or write.

In future, as more complex transformations are required we can add then in and even consider moving to a DTO's based model if it makes the task easier.

Syntax to convert json to Java object

Use below sample syntax to read JSON and populate java objects.

```
ObjectMapper mapper = new ObjectMapper();
Object value = mapper.readValue(jsonSource , javaObject);
```

- `jsonSource` – The input source which will fetch the json string.
- `javaObject` – The target Java object which needs to be populated.