Richardson Maturity Model

Richardson Maturity Model grades API by their RESTful maturity. It is proposed by **Leonard Richardson**. The Richardson maturity model is a way to grade your API according to the constraints of REST. It breaks down the principal element of the REST approach into **four** levels (0 to 3).

There are four levels:

- Level 0: The Swamp of POX
- Level 1: Resources
- Level 2: HTTP Verbs
- Level 3: Hypermedia Control

Level 0: The Swamp of POX

Level 0 is also known as POX (Plain Old XML). At level 0, HTTP is used only as a transport protocol that is used as a remote interaction. It does not take the advantages of HTTP like different HTTP methods, and HTTP cache. To get and post the data, we send a request to the same URI, and only the POST method may be used. These APIs use only one URI and one HTTP method called POST. In short, it exposes SOAP web services in the REST style.

For example, there can be many customers for a particular company. For all the different customers, we have only one endpoint. To do any of the operations like get, delete, update, we use the same POST method.

To get the data: http://localhost:8080/customer

To post the data: http://localhost:8080/customer

In the above two URIs, we have used the same URI and method to get and post the customers.

Level 1: Resources

When an API can distinguish between different resources, it might be at level 1. It uses multiple URIs. Where every URI is the entry point to a specific resource. It exposes resources with proper URI. Level 1 tackles complexity by breaking down huge service endpoints into multiple **different endpoints**. It also uses only one HTTP method POST for retrieving and creating data.

For example, if we want a list of specific products, we go through the URI http://localhost:8080/products. If we want a specific product, we go through the URI http://localhost:8080/products/mobile.

Remember the following points while building a URI:

- Use domain and subdomain to logically group or partition resources.
- Use **/** to indicate a hierarchical relationship.
- Use **,** and **;** to indicate non-hierarchical relationships.
- Use **-** and _ to improve the readability.
- Use **&** to separate parameters.
- Avoid including **file extensions**.

Level 2: HTTP Verbs

Level 2 indicates that an API must use the protocol properties to deal with scalability and failures. At level 2, correct **HTTP verbs** are used with each request. It suggests that in order to be truly RESTful, HTTP verbs must be used in API. For each of those requests, the correct HTTP response code is provided.

We don't use a single POST method for all requests. We use the **GET** method when we request a resource, and use the **DELETE** method when we want to delete a resource. Also, use the response codes of the application protocol.

For example, to get the customers, we send a request with the URI http://localhost:8080/customers, and the server sends proper response **200 OK**.

The following table shows the HTTP verbs and their usage:

Verbs	Safety & Idempotency	Usage

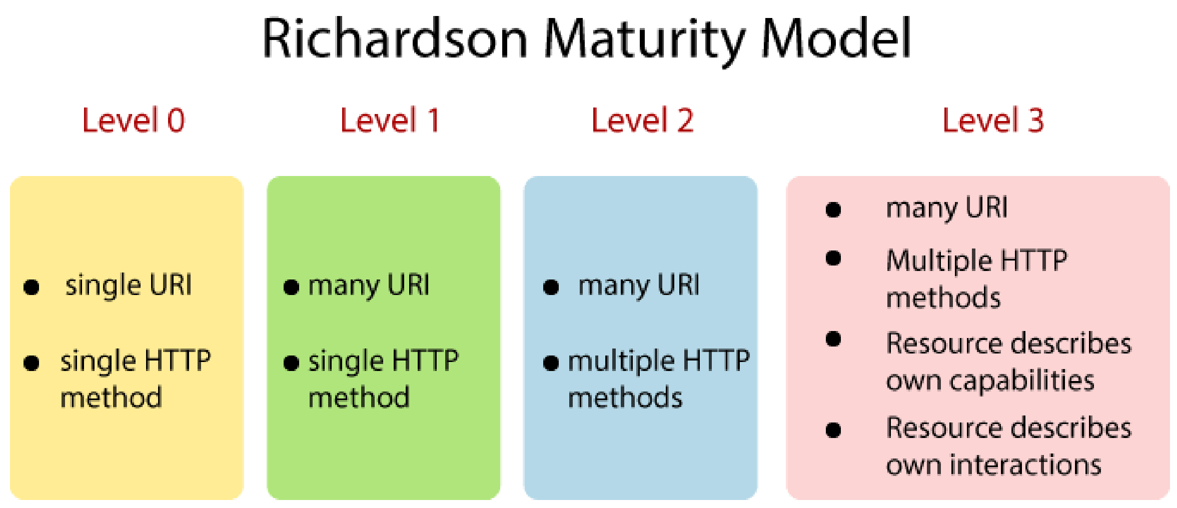
GET	Y/Y	It retrieves the information.
POST	N/N	It is used to perform a variety of actions on the server, such as create a update an existing resource, or making a mixture of changes to one or mo
DELETE	N/Y	It is used to delete a resource.
PUT	N/Y	It is used to update or replace an existing resource or to create a new specified by the client.
HEAD	Y/Y	It is used to retrieve the same headers as that of GET response but with response.
OPTIONS	Y/Y	It is used to find the list of HTTP methods supported by any resource or to
TRACE	Y/Y	It is used for debugging, which echo's back headers that it has received.

Level 3: Hypermedia Controls

Level 3 is the highest level. It is the combination of level 2 and HATEOAS. It also provides support for HATEOAS. It is helpful in self-documentation.

For example, if we send a GET request for customers, we will get a response for customers in JSON format with self-documenting Hypermedia.

The following figure shows the overview of the model:


Richardson Maturity Model
Level 0
single URI
single HTTP method
Level 1
many URI
single HTTP method
Level 2
many URI
multiple HTTP methods
Level 3
many URI
Multiple HTTP methods
Resource describes own capabilities
Resource describes own interactions