**Checklist for Hypermedia API.**

“***REST is easy for consumer but complex to implement for providers”.***

When we are planning to expose an API to the User always think that who are going to use this API. Client of Hypermedia API is another set of developers so we Hypermedia API developers must have to provide following features to client

1. Simple and Understandable URL.
2. Proper Response after client invoking a URL.
3. Easy to debug.

Below are the Do’s and Don’ts in checklist form for below sections

1. **Resource URI design and Resource representation.**
2. **HTTP methods.**
3. **Response Code.**
4. **HATEOAS**
5. **Resource URI design and Resource representation:**

**Do’s**

1. Always use a noun for resources say Articles, Refcardzs, Blogs etc.

1. Use Plural for resource name to represent Collections. Like /refCardz,/articles.
2. Use Identifier after Resource Name to specify an instance. /refCardz/{id}, /articles/{id} etc.
3. Use camelCase for Resource name like /refCardz.
4. To optimize searches, the result always uses Paginated views of collection and provide Hypermedia links for the page number. Like /refCardz should return 10 or 20 RefCardz per request rather than return all Refcardz.
5. Always provide efficient search mechanism by providing filter using “?” and “&” query parameter. Like /articles?category=java&year=2016
6. Enable Soring by query parameter. Like /articles?category=java&sortBy=publishDate.
7. Choose suitable content type header and charset parameter(define media type) for response like set “application/json;chardet=utf-8” for JSON response.

9. Specify content-language to record the language use.Specify Always use versioning in your API url’s like /v1/articles, /v1/refCardz.

**Don’ts**

1. Use Verb for resources like /getRefcardz. /getArtcles.

2. use many url based on filter criteria actually confuse clients. Like /getArtclesByAuthorName ,/getArticlesByCategory ,/getArticlesByValuableAutor etc.

3. use trailing forward slash like /refCardz/, /artcles/{id}/ etc.

4. Use filename extension for content type rather use suitable content type header. Like /refcardz.json , /articles.xml etc.

5. Sent excessive data in response so that it will create network blockage or heavy response.

1. **HTTP Methods:**

**Do’s:**

|  |  |  |
| --- | --- | --- |
| **Operation** | **HTTP Methods** | **Resource URL** |
| Create an Article | POST | /articles |
| Delete an Article | DELETE | /articles/{id} |
| Get a specific Article | GET | /article/{id} |
| Search for all articles | GET | /articles |
| Update a Specific Article | PUT | /articles/{id} |

1. For GET, always use cache control header to explicitly specify if the response is cached or not.

2. Always use POST, to add a new element to collection resources like/article/{id}.

3. Always use Location header in the response to tell the client about the newly created resource, so they have a clear idea in which location the new resource is created.

**Don’ts**

1. use PUT operation for creating resources.

2. Use query String with POST,PUT as input parameters like POST /articles?category=”java”

3. Use POST for an update.

4. Use PUT to create new resources.

5. Allow delete on collections.

**3. Response Code:**

**Do’s**

1. Use appropriate response code for the operation. Go to the link http://www.restapitutorial.com/httpstatuscodes.html

2. In the response body always provide proper error description.

**Don’ts**

1. Use 500 internal error code incorrect client request like missing mandatory header, malformed JSON etc.

1. **HATEOAS:**

**Do’s**

1. Always Provide suggestive links to navigate related resources. Example from Article you can navigate to Author.

{

"Article": "HyperMedia API Checklist",

"links": [ {

"rel": "Author",

"href": "http://dzone.com/author/1"

} ]

}

2. Make sure your API is explorable enough by providing suggestive link.

**Don’ts**

1. Not using HATEOAS fails to reach Richardson matrix mode 3 level.