**REST API versioning**

There are **multiple options** that you can use to do versioning of your REST API.

1. You can do **versioning based on URIs,**
2. you can do **versioning based on request parameters**.
3. You can do **versioning based on request headers,** or
4. you can use the accept header to do versioning (**Media type Versioning**).

**Which of these is the recommended approach**? Let's start with discussing the different **factors to consider** when you are deciding how to version your REST API.

* **URI pollution**: When you look at URI versioning and request parameter versioning, what we're doing is we're creating new URLs to represent the new version. So there is a lot of URI pollution when it comes to URI versioning and request parameter versioning. However, in the case of header's versioning and media type versioning, we're using the same URL so they have less amount of URI pollution.
* **Misuse of HTTP headers**: HTTP headers were never meant to be used for versioning. So headers versioning and media type versioning misuse the HTTP headers.
* **Caching:** Typically, caching is done based on the URL. And when it comes to header versioning and media type versioning, we're using different versions. However, both those versions can have the same URL. So when it comes to header versioning and media type versioning, you cannot cash just based on the URL. You also need to look at the headers before you do caching.
* **Can we execute the request on the browser?:** When it comes to URL versioning and request parameter versioning you can easily execute them on the browser because the differentiation is in the URL. However, when it comes to headers versioning and media versioning, the differentiation is in the headers. Typically, you need to have a command line utility or you need to make use of a REST API client to be able to differentiate based on the headers.
* **API documentation**: Typically, generating API documentation for URI versioning and request parameter versioning is easy because the URLs are different for both the versions. Typical API documentation generation tools might not support generating documentation differentiating based on the headers. So generating documentation for headers versioning and media type versioning might be a little difficult.

There is no perfect solution when it comes to versioning. You can see that each of these versioning approaches are being used by different large enterprises.

**URI versioning** 🡪***Twitter***, **request parameter versioning** 🡪 ***Amazon***, **Header versioning** 🡪 ***Microsoft*** and **media type versioning** 🡪 ***GitHub***.

So different enterprises are making use of different ways to version their REST API. When it comes to versioning , think about versioning even before you need it. When you're starting to build your REST API, that's when you'd think about versioning. And number two is to have a consistent versioning approach across your enterprise.

**The best practice always is one enterprise, one versioning approach across your multiple different projects, multiple applications.** You'd want one way to version your REST API. It does not matter which approach you choose, but it's very, very important that the approach used is consistent across all projects and all applications in an enterprise.