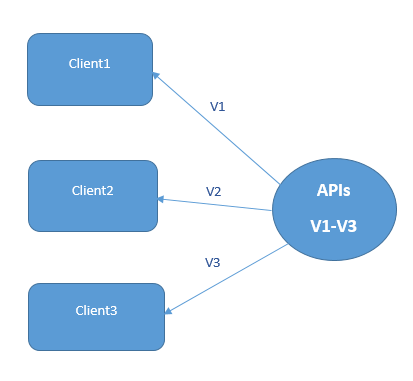
**Rest API Versioning**

**Need of versioning for Apis**

Versioning of Apisis basically useful when you have different external clients who are consuming it. For example, you may have APIs that are used by different clients. Although, the approach has both pros and cons. The obvious advantage of using versioning is you need not to worry about making changes to your APIs. But the disadvantage of the approach is boiler plate code, as lot of the code will be duplicate.

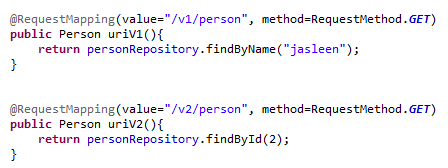


**Different Approaches for Api Versioning**

1. **URI Path** based versioning
2. **Query Parameters** based versioning
3. **Custom Http Headers** based versioning
4. **Content Negotiation/Mime/Media type** based Versioning**.**

**1) URI Path based versioning**

According to this approach, you need to add version number as part of your API endpoint. For example, /Api/V1/person. This approach is followed by Twitter.

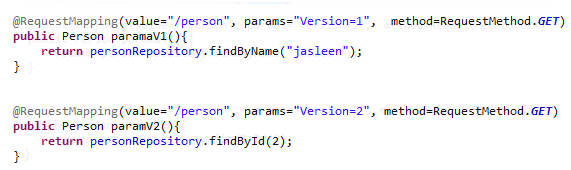


URI path based Versioning

In above figure, shows two different implementations of API, which provides Person details. First endpoint fetches the person details based on name, whereas second endpoint fetches the person details based on id. In this case may be first endpoint is required by client 1 and second by client2, so both can be used by using different endpoints.

**2) Query parameter based versioning**

According to this approach, you pass the version number as a query parameter with specified name For example, /Api/person? Version=1. This approach is followed by Amazon.

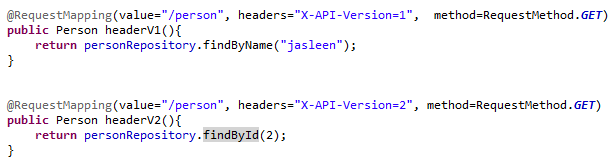


Query Parameter based Versioning

Above figure shows, versioning based on query parameters. So, URL /person?Version=1 return person details by name, whereas URL /person?Version=2 return person details by id.

**3) Custom Header based versioning**

According to this approach, you need to define a new header which sends API version number. For example, X-API-Version=1. This approach is followed by Microsoft.

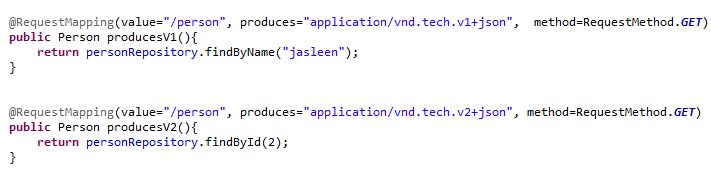


Custom Header based Versioning

Above figure shows, API versioning based on custom header which sends version number as header along with request. So, URL /person with header X-API-Version:1 return person details by name, whereas URL /person with header X-API-Version:2 return person details by id.

**4) Content Negotiation/Mime/Media type based Versioning.**

According to this approach, the version number is included to the “Accept” header together with accepted content type. For Example, **Accept: application/vnd.tech.v1+json** . This approach is followed by Github.



Content negotiation based Versioning

Above figure shows, Content negotiation based versioning. So, URL /person with header Accept: **application/vnd.tech.v1+json** return person details by name, whereas URL /person with header X- Accept: **application/vnd.tech.v2+json** return person details by id.

The decision of which approach to use, is as per requirements of your application.