**Spring Boot - Versioning a REST API**

Last Updated : 04 Jan, 2025

API Versioning is a defined process of making or managing changes to an API. These changes can be made transparently without interrupting the clients. When users have permission to decide whether they can conveniently upgrade to the latest version of API and clearly state the changes made is known as a Good API Versioning Strategy.

**When to Version a REST API?**

When there is a condition or event of breaking change, then it is considered to be best practice to version your API because API versioning is a costly process for both the API users and API developers. A Breaking Change can be defined as a change that may be required to your software applications to avoid abnormal situations of application.

Breaking changes may include:

* When we change a request/response format
* Changing the data type of the resource
* Adding a mandatory/required field on the client HTTP request
* Removing one or more resources on the response

**REST API Versioning Strategies - Spring Boot**

There are several types of strategies to versioning REST API using Spring Boot.

1. URI Path Versioning
2. Query Parameters Versioning
3. Custom Header Versioning
4. Content Negotiation Versioning

**1. URI Path Versioning**

The process of adding the different version numbers like v1,v2, etc. to the API's URL is known as URI Versioning. It is easy to understand and straightforward. Sometimes it may lead to long URL Paths.

**Example** :

* http://www.your-website.com/v1/data or http://localhost:8080/v1/data
* http://www.your-website.com/v2/data or http://localhost:8080/v2/data

*//URI Versioning*

@RestController

**public** **class** **PersonControllerV1** {

*// Define endpoints for version 1*

@Autowired

**private** PersonService users;

*//http://localhost:8080/api/v1/person*

@Getmapping("/api/v1/person")

**public** List<PersonV1> URIVersioningV1(){

**return** users.getAllV1();

}

*//http://localhost:8080/api/v2/person*

@Getmapping("/api/v2/person")

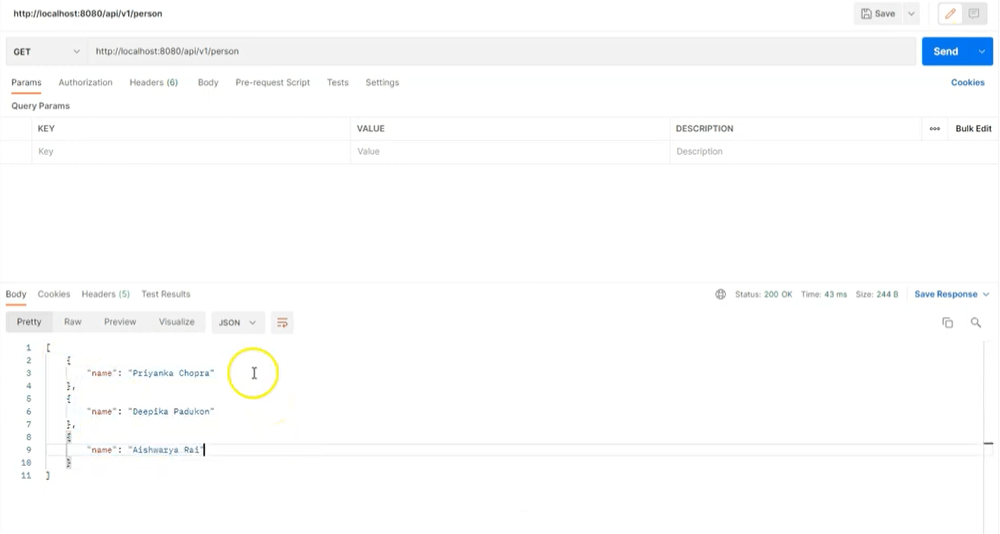
**public** List<PersonV2> URIVersioningV2(){

**return** users.getAllV2();

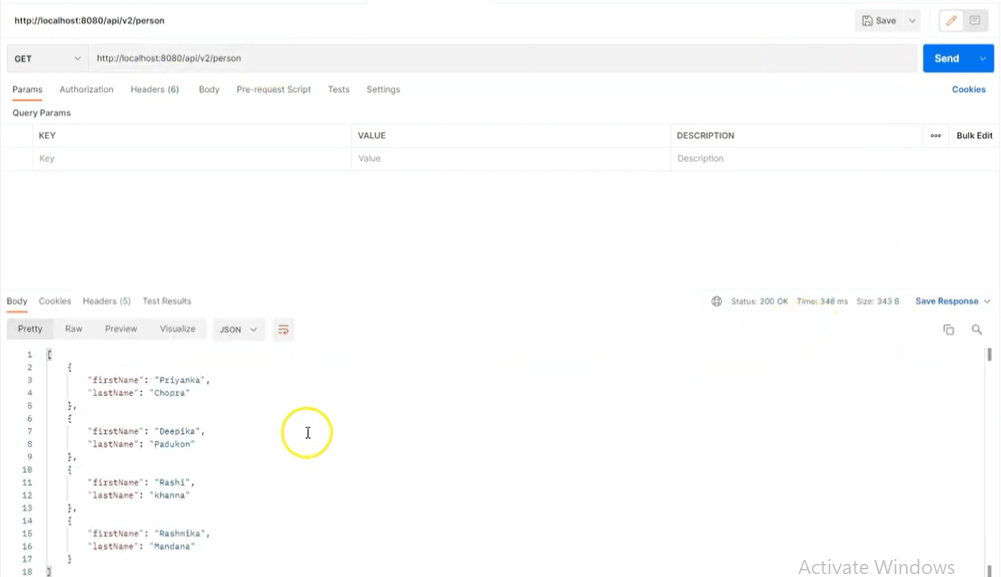
}

}

**Output of Version1:**



**Output of Version2:**



**2. Query Parameter Versioning (Request Parameter Versioning)**

The process in which version number is passed as a query or request parameter. It is less impertinent approach than URI Path Versioning. It is not as much helpful for consumers.

**Example**:

* http://www.your-website.com/param?version=1 or http://localhost:8080/api/param?version=1
* http://www.your-website.com/param?version=2 or http://localhost:8080/api/param?version=2



1

// REQUEST PARAMETER VERSIONING

2

@RestController

3

public class PersonControllerV1 {

4

5

@Autowired

6

private PersonService users;

7

8

//http://localhost:8080/api/person?version=1

9

@Getmapping(path = "/api/person", params = "version=1")

10

public List<PersonV1> requestParamsV1(){

11

return users.getAllV1();

12

}

13

14

// //http://localhost:8080/api/person?version=2

15

@Getmapping(path = "/api/person", params = "version=2")

16

public List<PersonV2> requestParamsV2(){

17

return users.getAllV2();

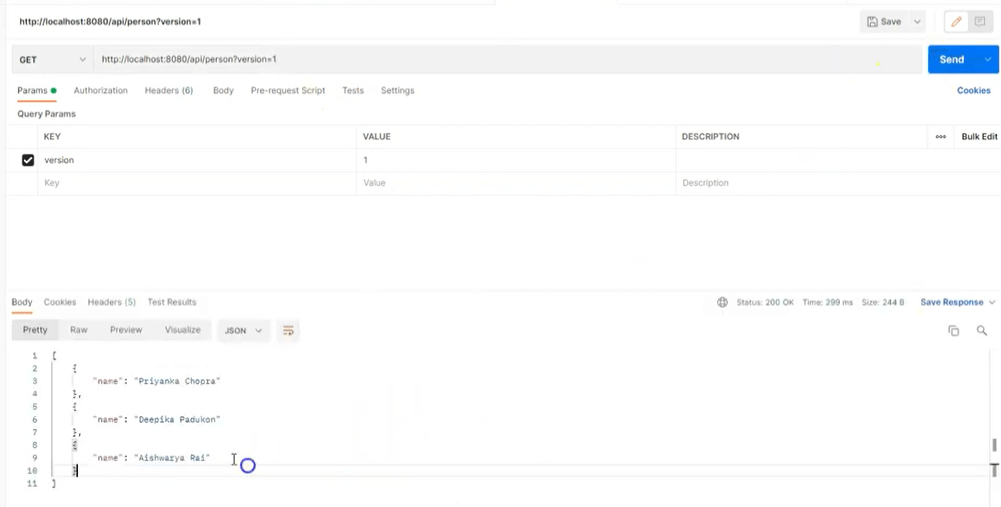
18

}

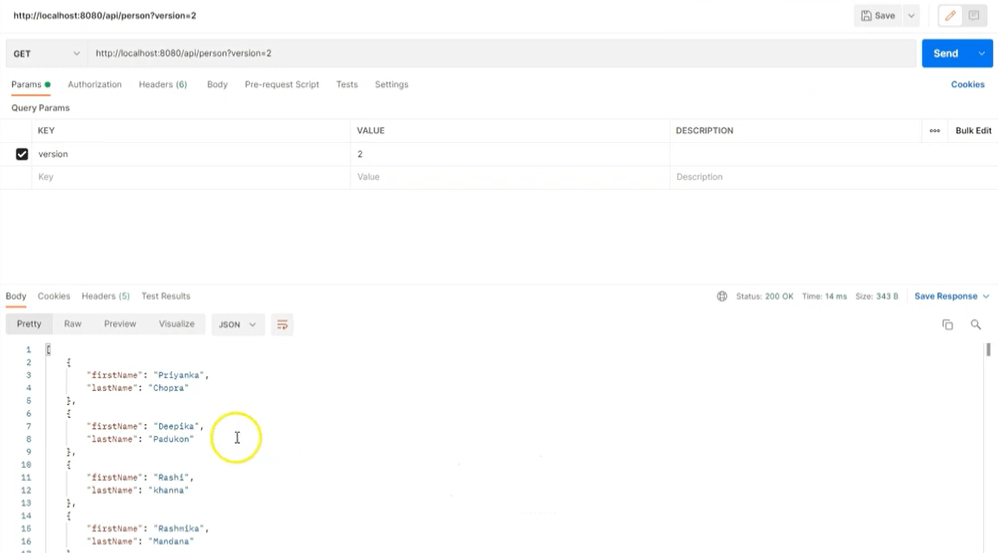
19

}

**Output of Version1 parameter :**



**Output of Version2 parameter:**



**3. Custom Header Versioning**

The procedure of adding a custom header to the client HTTP request with version number is known as Custom Header Versioning. This approach keeps the URI Path clear, but custom header must be included.

**Example**: http://localhost:8080/api/person/header

* headers=[X-API-VERSION=1]
* headers=[X-API-VERSION=2]

*// Custom Request Header VERSIONING*

@RestController

**public** **class** **PersonControllerV1** {

@Autowired

**private** PersonService users;

*//http://localhost:8080/api/person*

@Getmapping(path = "/api/person", headers = "X-API-VERSION=1")

**public** List<PersonV1> requestHeaderV1(){

**return** users.getAllV1();

}

*// //http://localhost:8080/api/person*

@Getmapping(path = "/api/person", headers = "X-API-VERSION=2")

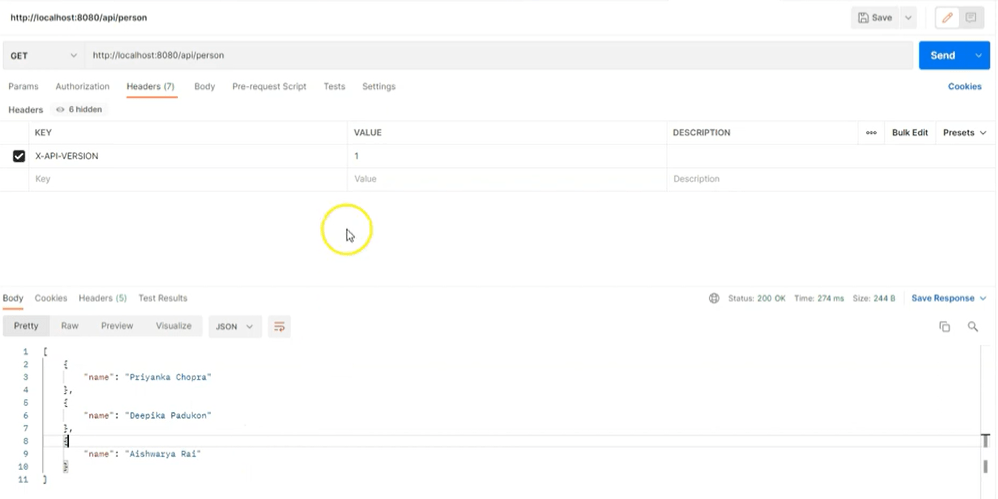
**public** List<PersonV2> requestHeaderV2(){

**return** users.getAllV2();

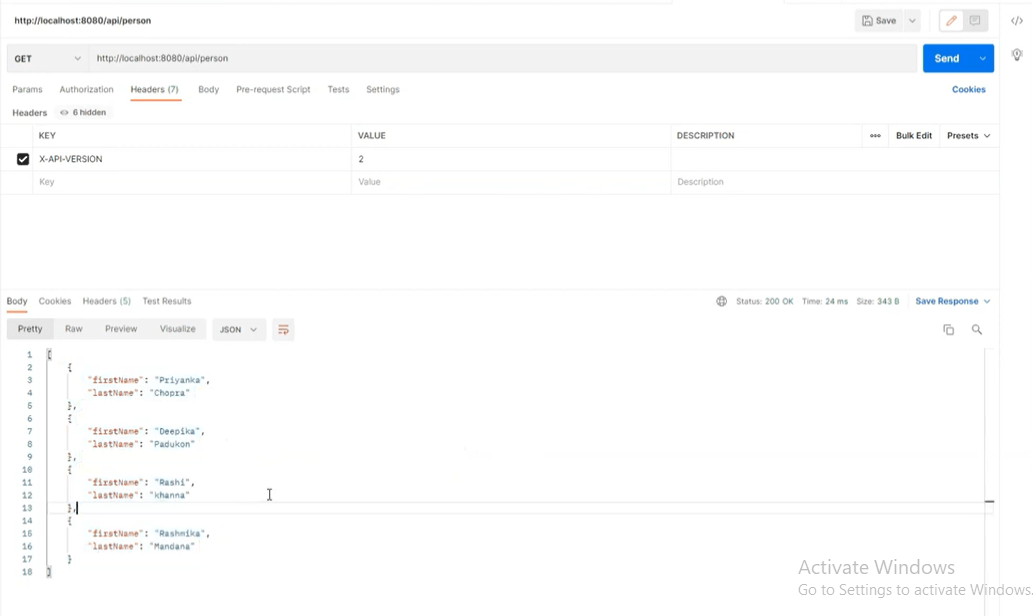
}

}

**Output of Request Head Version1:**



**Output of Request Head Version2:**



**4. Content Negotiation Versioning**

Content Negotiation Versioning uses the HTTP Accept header to define the API version. In this method, version is a part of content Negotiation.

**Example**: http://localhost:8080/api/person

* headers[Accept=application/vnd.company.app-v1+json]
* headers[Accept=application/vnd.company.app-v2+json]

*// Request Accept Header*

@RestController

**public** **class** **PersonControllerV1** {

@Autowired

**private** PersonService users;

*//http://localhost:8080/api/person/*

@Getmapping(path = "/api/person", produces = "application/vnd.company.app-v1+json")

**public** List<PersonV1> requestAcceptHeaderV1(){

**return** users.getAllV1();

}

*//http://localhost:8080/api/person/*

@Getmapping(path = "/api/person", produces = "application/vnd.company.app-v2+json")

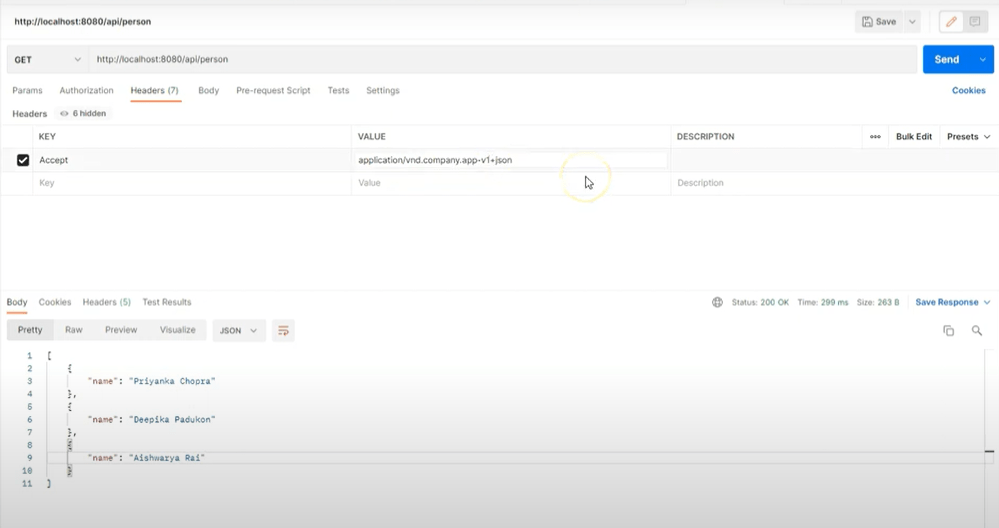
**public** List<PersonV2> requestAcceptHeaderV2(){

**return** users.getAllV2();

}

}

**Output of Negotiation Version1:**



**Output of Negotiation Version2:**

