API Challenge link - https://apichallenges.herokuapp.com/gui/challenges
API documentation link - https://apichallenges.herokuapp.com/docs
The solved challenge has a green color.

Challenge solutions:

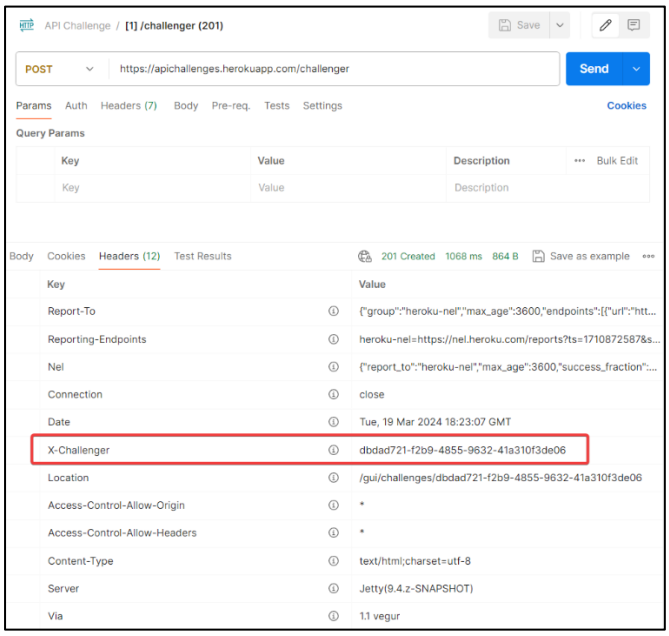
1. Create a new challenger session

Getting Started

If you want to track your challenge progress, in multi-user mode then you need to solve the challenges in this section to generate a unique ID that we can associate your progress with.

ID	Challenge	Done	Description
01	POST / challenger (201)	false	Issue a POST request on the `/challenger` end point, with no body, to create a new challenger session. Use the generated X-CHALLENGER header in future requests to track challenge completion. ▸ Hints ▸ Solution

Solution:



01	POST /challenger (201)	true	Issue a POST request on the `/challenger` end point, with no body, to create a new challenger session. Use the generated X-CHALLENGER header in future requests to track challenge completion. ▸ Hints ▸ Solution

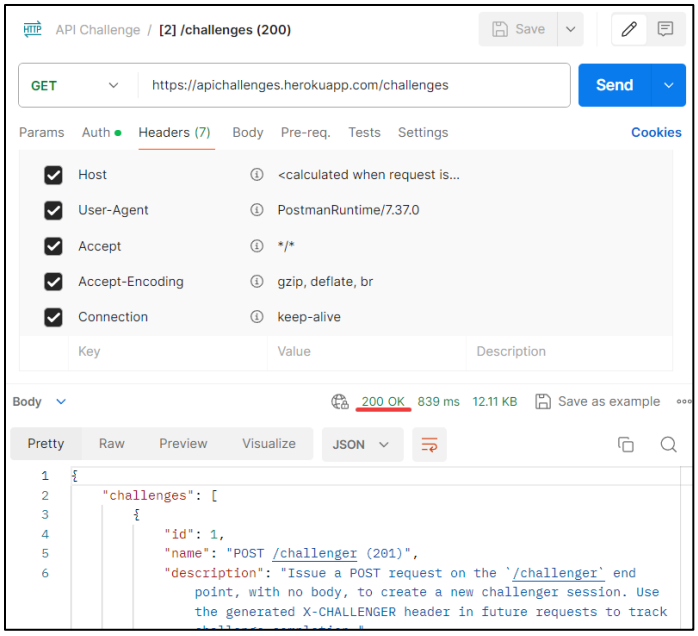
2. Retrieve a list of challenges

First Real Challenge

For your first challenge, get the list of challenges. You'll be able to use this to see your progress in your API Client, as well as using the GUI.

ID	Challenge	Done	Description
02	GET /challenges (200)	false	Issue a GET request on the `/challenges` end point ▸ Solution

Solution:



02	GET /challenges (200)	true	Issue a GET request on the `/challenges` end point ▶ Solution

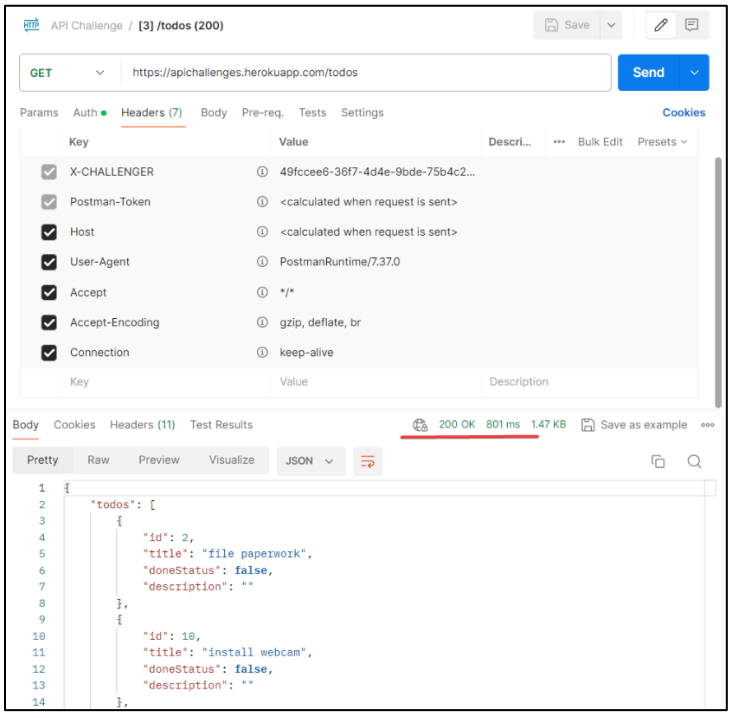
3. Retrieve a list of 'todos'

GET Challenges

To retrieve, or read information from an API we issue GET requests. This section has a bunch of GET request challenges to try out.

ID	Challenge	Done	Description
03	GET /todos (200)	false	Issue a GET request on the `/todos` end point ▶ Solution

Solution:

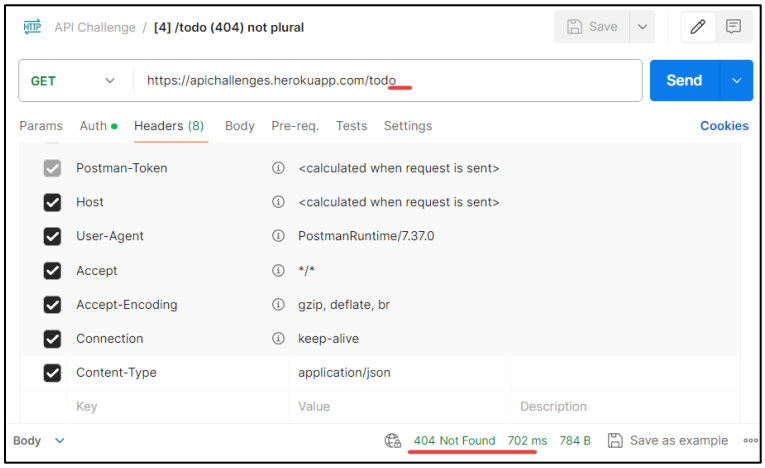


03	GET /todos (200)	true	Issue a GET request on the `/todos` end point ▶ Solution

4. Retrieve the 404 status-code

04	GET /todo (404) not plural	false	Issue a GET request on the `/todo` end point should 404 because nouns should be plural ▸ Solution

Solution:

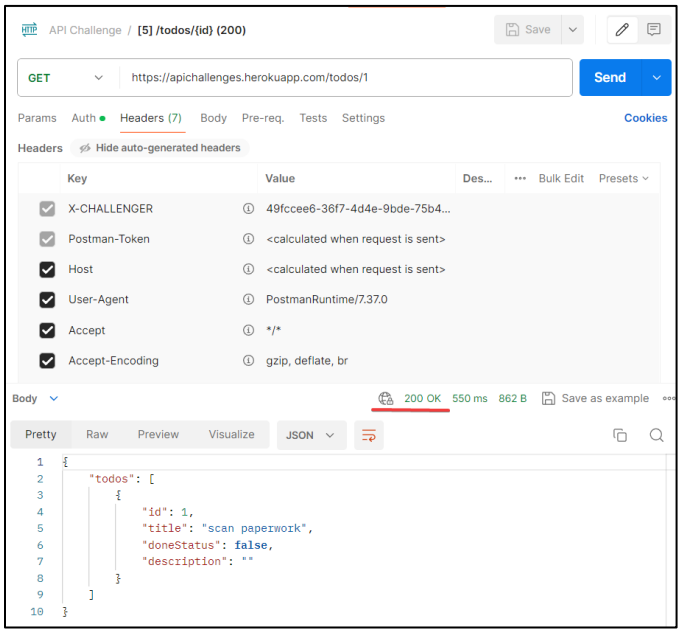


04	GET /todo (404) not plural	true	Issue a GET request on the `/todo` end point should 404 because nouns should be plural ▸ Solution

5. Retrieve specific 'todo' with status code 200

05	GET /todos/{id} (200)	false	Issue a GET request on the `/todos/{id}` end point to return a specific todo ▸ Hints ▸ Solution

Solution:

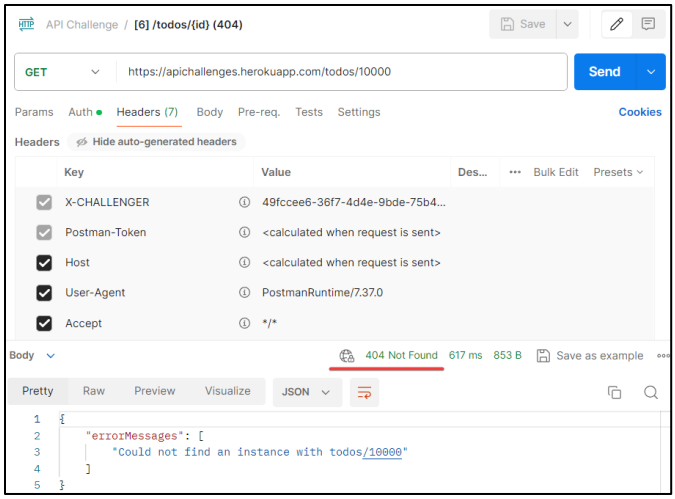


05	GET /todos/{id} (200)	true	Issue a GET request on the `/todos/{id}` end point to return a specific todo ▸ Hints ▸ Solution

6. Retrieve 404 status code when {todo id} doesn't exist

06	GET /todos/{id} (404)	false	Issue a GET request on the `/todos/{id}` end point for a todo that does not exist ▸ Hints ▸ Solution

Solution:

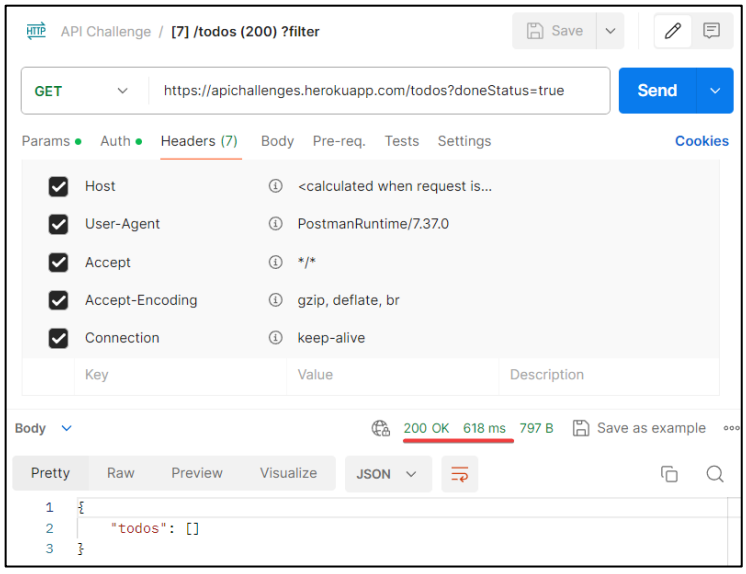


06	GET /todos/{id} (404)	true	Issue a GET request on the `/todos/{id}` end point for a todo that does not exist ▶ Hints ▶ Solution

7. Get a filtered list of todos

07	GET /todos (200) ?filter	false	Issue a GET request on the `/todos` end point with a query filter to get only todos which are 'done'. There must exist both 'done' and 'not done' todos, to pass this challenge. ▶ Hints ▶ Solution

Solution:



07	GET /todos (200) ?filter	true	Issue a GET request on the `/todos` end point with a query filter to get only todos which are 'done'. There must exist both 'done' and 'not done' todos, to pass this challenge. ▶ Hints ▶ Solution

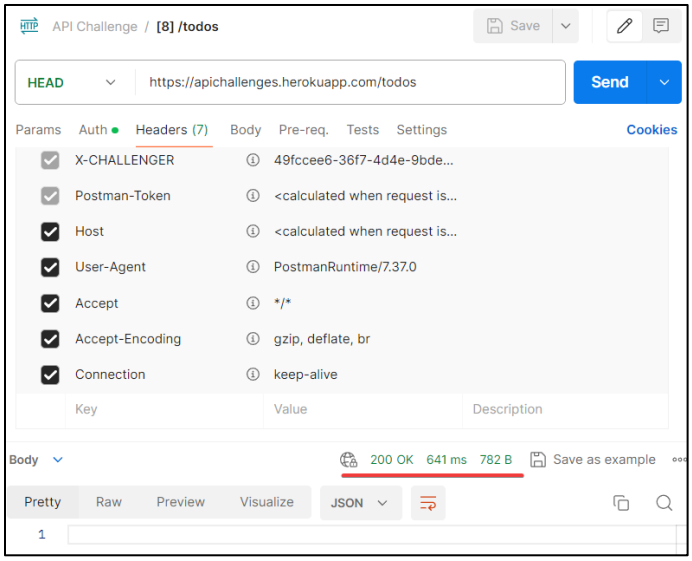
8. Create a HEAD request for todos

HEAD Challenges

A HEAD request, is like a GET request, but only returns the headers and status code.

ID	Challenge	Done	Description
08	HEAD /todos (200)	false	Issue a HEAD request on the `/todos` end point ▶ Solution

Solution:



08	HEAD /todos (200)	true	Issue a HEAD request on the `/todos` end point ▶ Solution

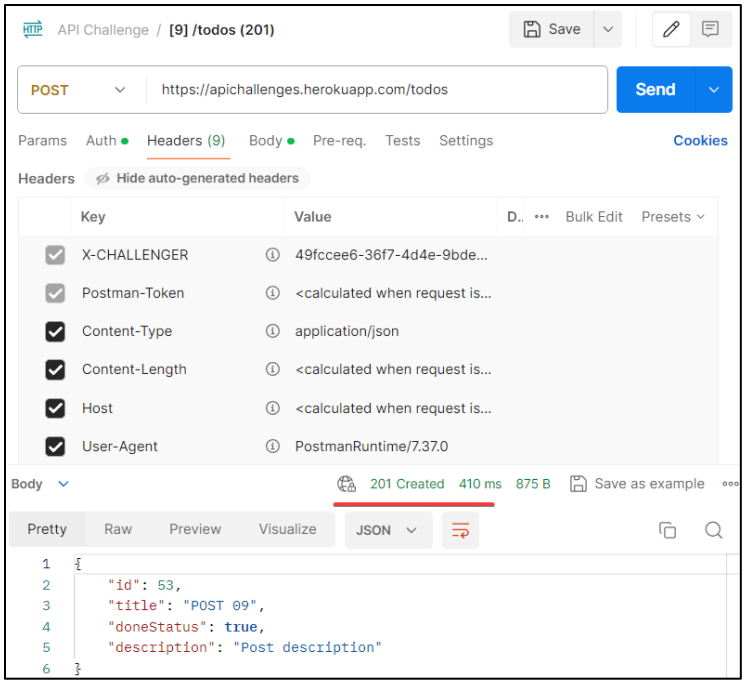
9. Create a new todo record

Creation Challenges with POST

A POST request can be used to create and update data, these challenges are to 'create' data. As a Hint, if you are not sure what the message body should be, try copying in the response from the associated GET request, and amending it.

ID	Challenge	Done	Description
09	POST /todos (201)	false	Issue a POST request to successfully create a todo ▶ Solution

Solution:

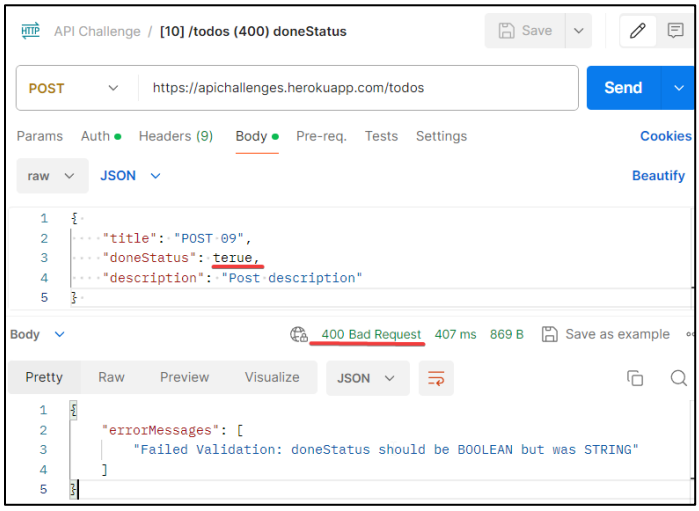


09	POST /todos (201)	true	Issue a POST request to successfully create a todo ▶ Solution

10. Fail validation on the 'doneStatus' field

10	POST /todos (400) doneStatus	false	Issue a POST request to create a todo but fail validation on the `doneStatus` field ▶ Solution

Solution:

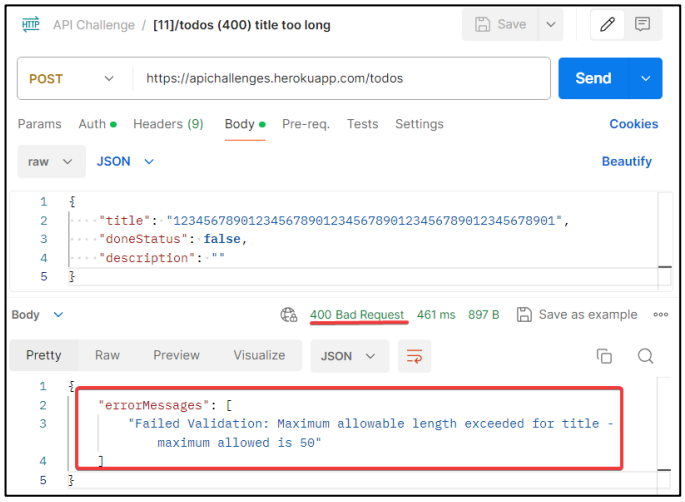


10	POST /todos (400) doneStatus	true	Issue a POST request to create a todo but fail validation on the `doneStatus` field ▶ Solution

11. Create a todo record with fail validation on the 'title' field:

11	POST /todos (400) title too long	false	Issue a POST request to create a todo but fail length validation on the `title` field because your title exceeds maximum allowable characters.

Solution:

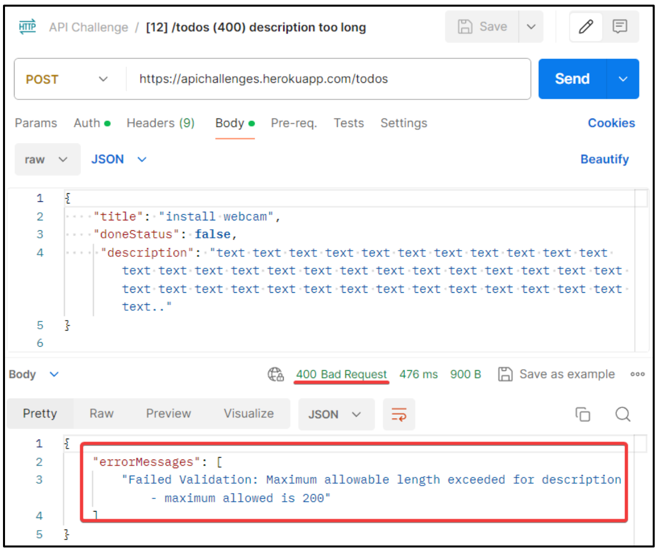


11	POST /todos (400) title too long	true	Issue a POST request to create a todo but fail length validation on the `title` field because your title exceeds maximum allowable characters.

12. Create a todo record with fail validation on the 'title' field:

12	POST /todos (400) description too long	false	Issue a POST request to create a todo but fail length validation on the `description` because your description exceeds maximum allowable characters.

Solution:

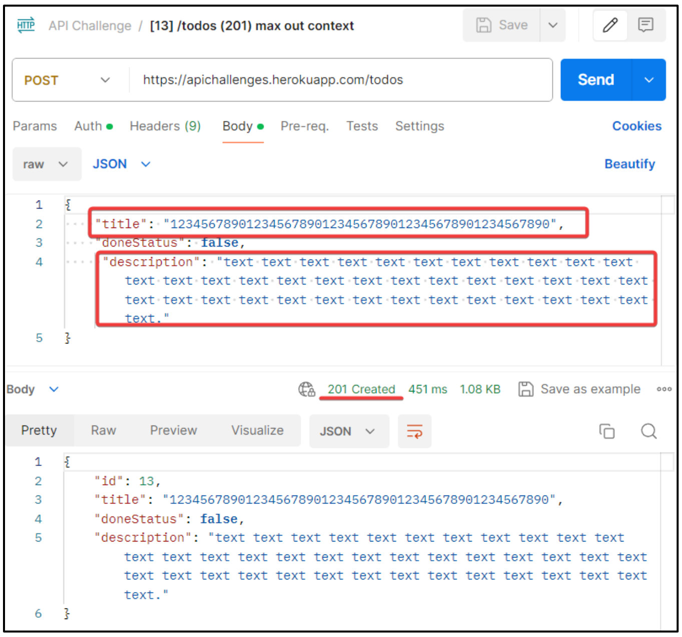


12	POST /todos (400) description too long	true	Issue a POST request to create a todo but fail length validation on the `description` because your description exceeds maximum allowable characters.

13. Create a todo record with max length of title and description fields

13	POST /todos (201) max out content	false	Issue a POST request to create a todo with maximum length title and description fields. ▶ Hints

Solution:

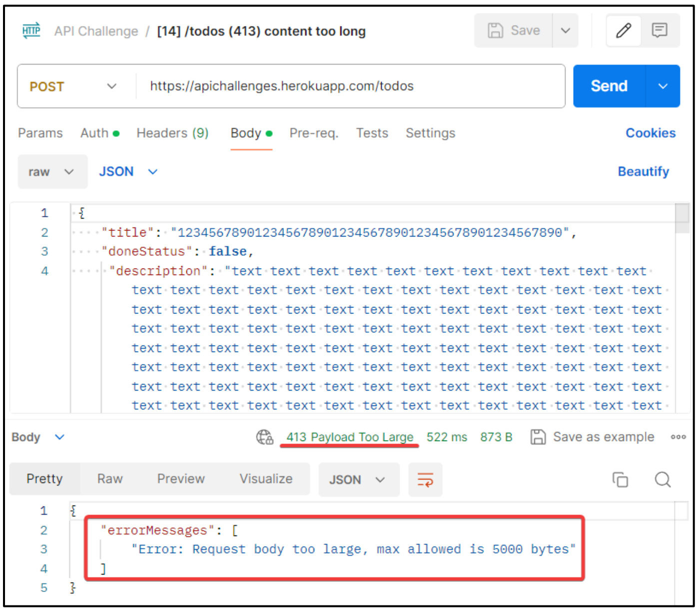


13	POST /todos (201) max out content	true	Issue a POST request to create a todo with maximum length title and description fields. ▶ Hints

14. Create todo with long content

14	POST /todos (413) content too long	false	Issue a POST request to create a todo but fail payload length validation on the `description` because your whole payload exceeds maximum allowable 5000 characters. ▸ Hints

Solution:

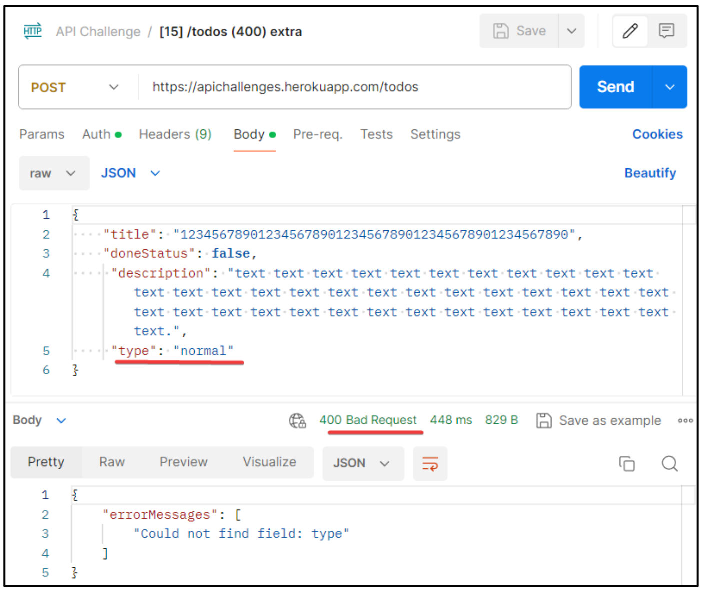


14	POST /todos (413) content too long	true	Issue a POST request to create a todo but fail payload length validation on the `description` because your whole payload exceeds maximum allowable 5000 characters. ▸ Hints

15. Create a todo record using an unrecognized field in the payload

15	POST /todos (400) extra	false	Issue a POST request to create a todo but fail validation because your payload contains an unrecognised field. ▸ Hints

Solution:



15	POST /todos (400) extra	true	Issue a POST request to create a todo but fail validation because your payload contains an unrecognised field. ▸ Hints

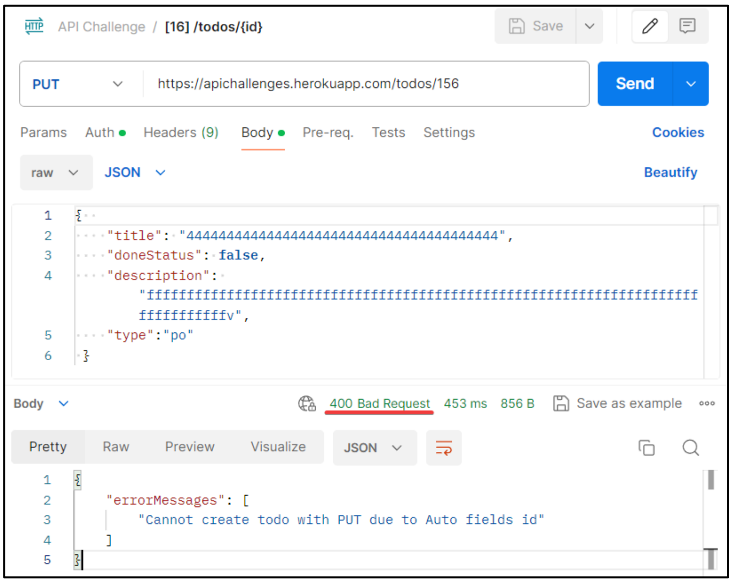
16. Use a PUT request for unsuccessfully creating a todo record

Creation Challenges with PUT

A PUT request can often used to create and update data. The todo application we are using has automatically generated ids, so you cannot use PUT to create. As a Hint, if you are not sure what the message body should be, try copying in the response from the associated GET request, and amending it.

ID	Challenge	Done	Description
16	PUT /todos/{id} (400)	false	Issue a PUT request to unsuccessfully create a todo

Solution:



ID	Challenge	Done	Description
16	PUT /todos/{id} (400)	true	Issue a PUT request to unsuccessfully create a todo

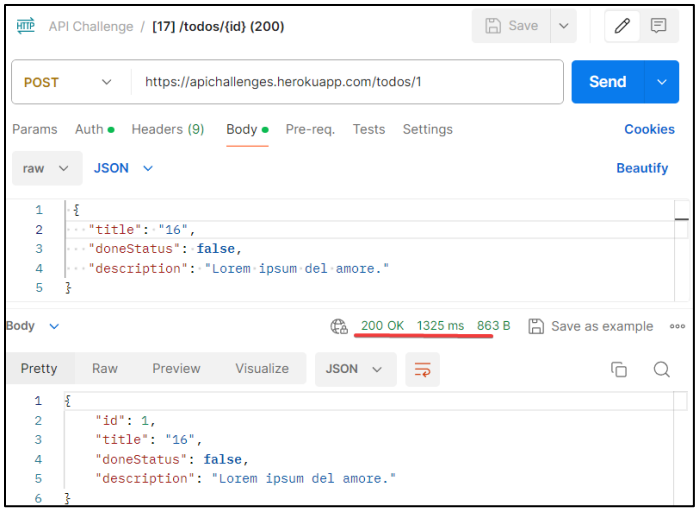
17. Update todo record

Update Challenges with POST

Use a POST request to amend something that already exists. These are 'partial' content updates so you usually don't need to have all details of the entity in the request, e.g. you could just update a title, or a description, or a status

ID	Challenge	Done	Description
17	POST /todos/{id} (200)	false	Issue a POST request to successfully update a todo ▶ Hints ▶ Solution

Solution:

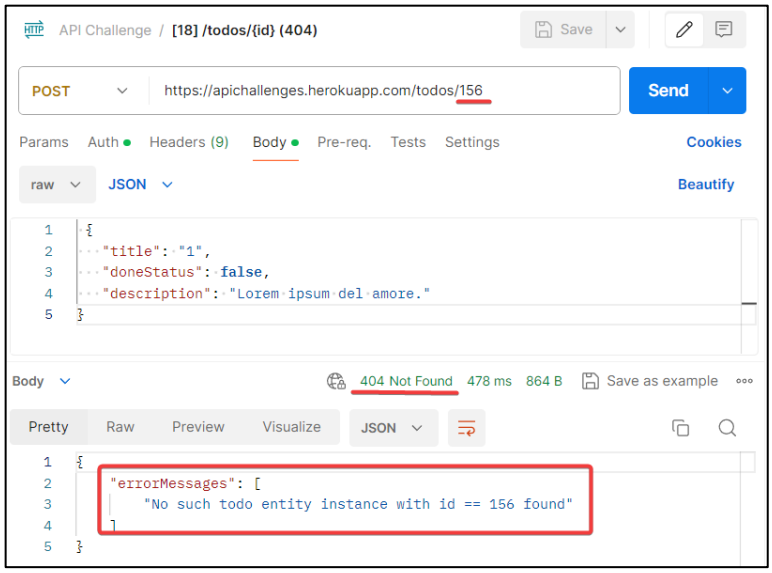


17	POST /todos/{id} (200)	true	Issue a POST request to successfully update a todo ▶ Hints ▶ Solution

18. Update a todo record which doesn't exist

18	POST /todos/{id} (404)	false	Issue a POST request for a todo which does not exist. Expect to receive a 404 response. ▶ Hints

Solution:



18	POST /todos/{id} (404)	true	Issue a POST request for a todo which does not exist. Expect to receive a 404 response. ▶ Hints

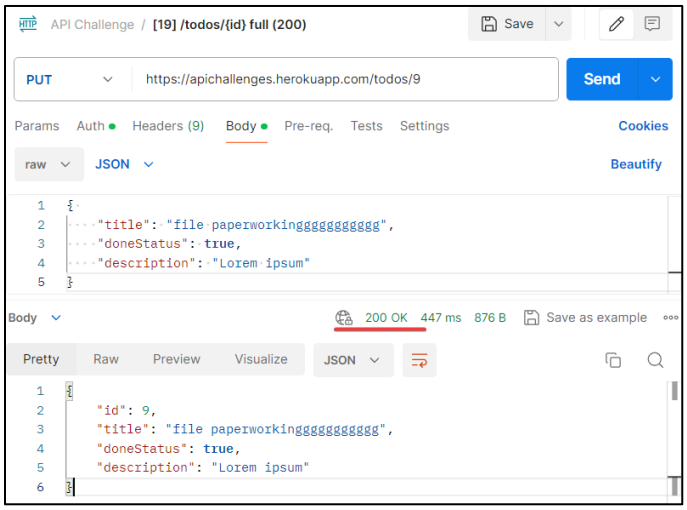
19. Update an existing todo record with a complete payload

Update Challenges with PUT

A PUT request can be used to amend data. REST Put requests are idempotent, they provide the same result each time.

ID	Challenge	Done	Description
19	PUT /todos/{id} full (200)	false	Issue a PUT request to update an existing todo with a complete payload i.e. title, description and donestatus.

Solution:

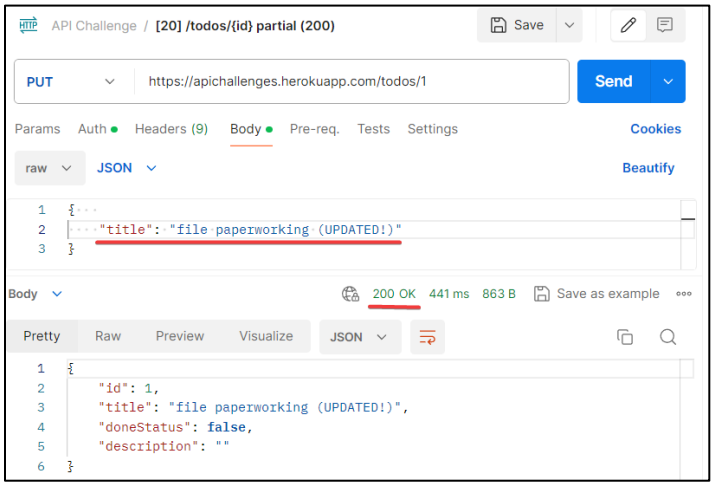


19	PUT /todos/{id} full (200)	true	Issue a PUT request to update an existing todo with a complete payload i.e. title, description and donestatus.

20. Update mandatory items for an existing todo record

20	PUT /todos/{id} partial (200)	false	Issue a PUT request to update an existing todo with just mandatory items in payload i.e. title.

Solution:

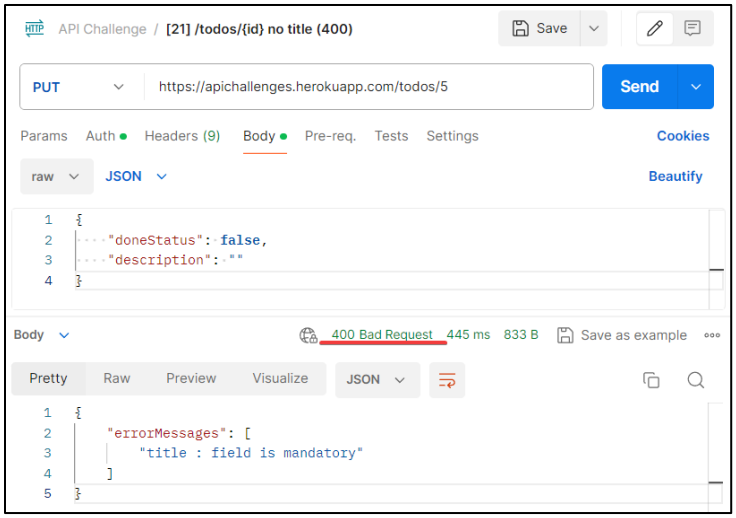


20	PUT /todos/{id} partial (200)	true	Issue a PUT request to update an existing todo with just mandatory items in payload i.e. title.

21. Fail to update an existing todo record

21	PUT /todos/{id} no title (400)	false	Issue a PUT request to fail to update an existing todo because title is missing in payload. ▶ Hints

Solution:

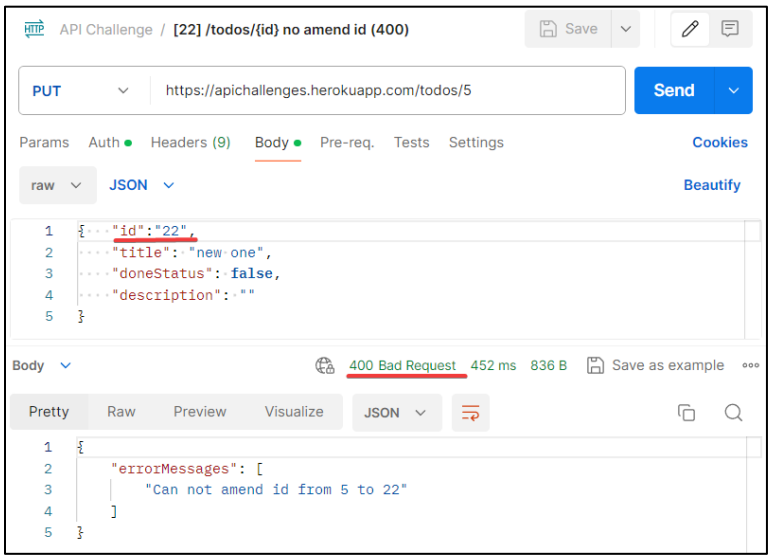


21	PUT /todos/{id} no title (400)	true	Issue a PUT request to fail to update an existing todo because title is missing in payload. ▸ Hints

22. Fail to update an existing todo

22	PUT /todos/{id} no amend id (400)	false	Issue a PUT request to fail to update an existing todo because id different in payload. ▸ Hints

Solution:



22	PUT /todos/{id} no amend id (400)	true	Issue a PUT request to fail to update an existing todo because id different in payload. ▸ Hints

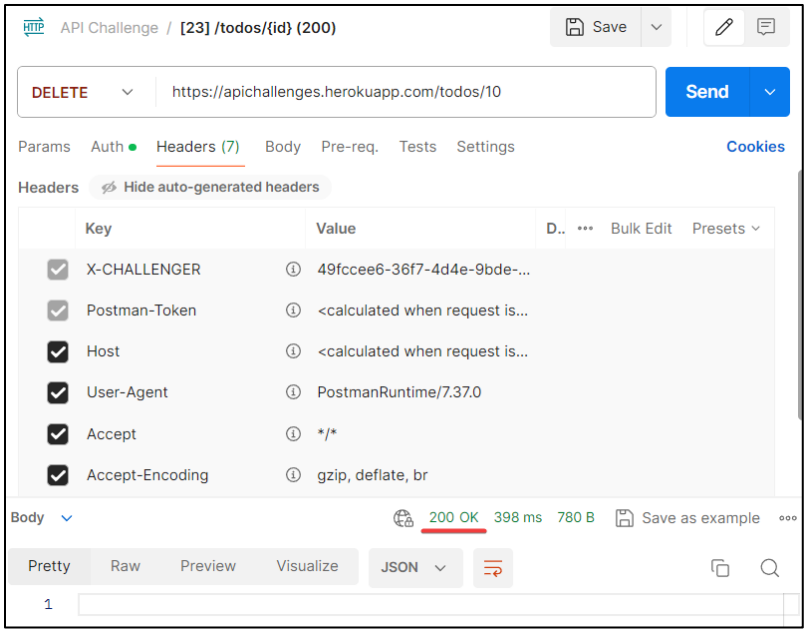
23. Delete the todo record

DELETE Challenges

Use a DELETE request to delete an entity. Since this is an extreme request, normally you have to be logged in or authenticated, but we wanted to make life easier for you so we cover authentication later. Anyone can delete To Do items without authentication in this system.

ID	Challenge	Done	Description
23	DELETE /todos/{id} (200)	false	Issue a DELETE request to successfully delete a todo ▸ Hints ▸ Solution

Solution:



23	DELETE /todos/{id} (200)	true	Issue a DELETE request to successfully delete a todo ▸ Hints ▸ Solution

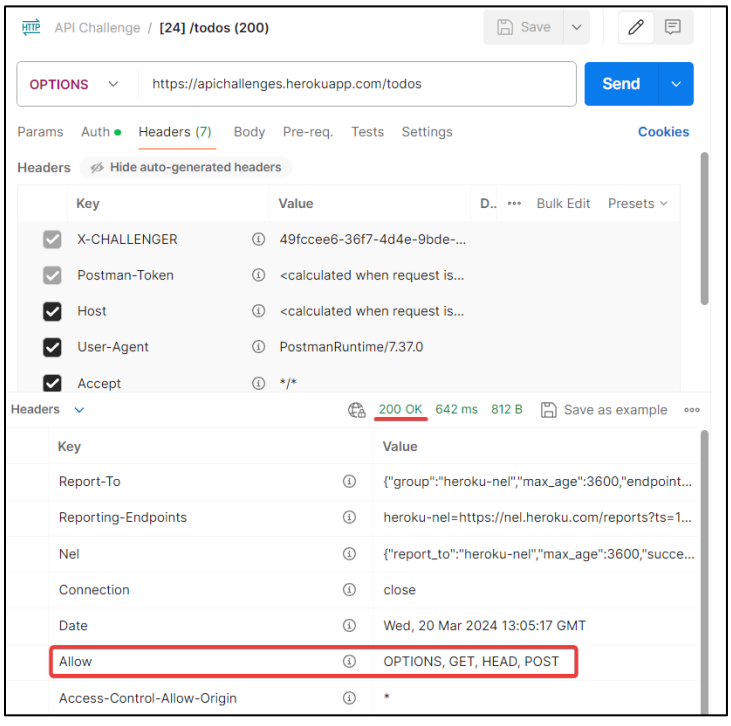
24. Get an 'Allow' header in the OPTION request

OPTIONS Challenges

Use an OPTIONS verb and check the `Allow` header, this will show you what verbs are allowed to be used on an endpoint. When you test APIs it is worth checking to see if all the verbs listed are allowed or not.

ID	Challenge	Done	Description
24	OPTIONS / todos (200)	false	Issue an OPTIONS request on the `/todos` end point. You might want to manually check the 'Allow' header in the response is as expected. ▸ Solution

Solution:



24	OPTIONS /todos (200)	true	Issue an OPTIONS request on the `/todos` end point. You might want to manually check the 'Allow' header in the response is as expected. ▸ Solution

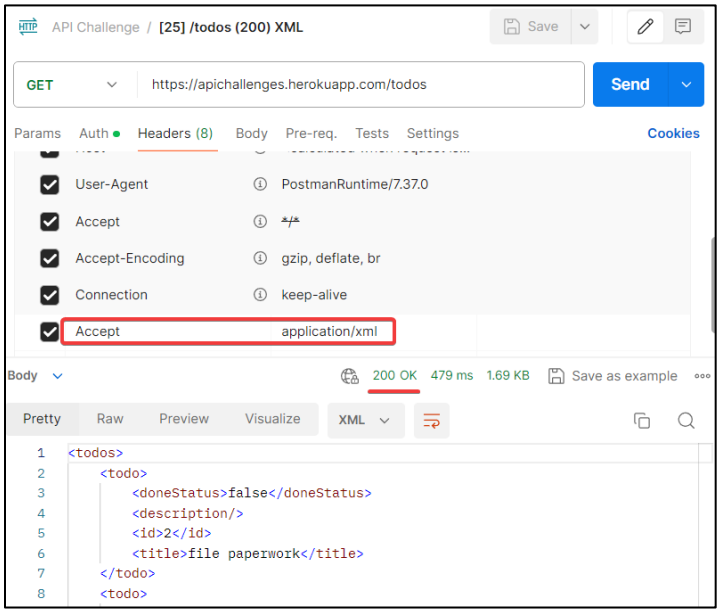
25. Receive a result in XML format

Accept Challenges

The `Accept` header, tells the server what format you want the response to be in. By changing the `Accept` header you can specify JSON or XML.

ID	Challenge	Done	Description
25	GET /todos (200) XML	false	Issue a GET request on the `/todos` end point with an `Accept` header of `application/xml` to receive results in XML format ▶ Solution

Solution:

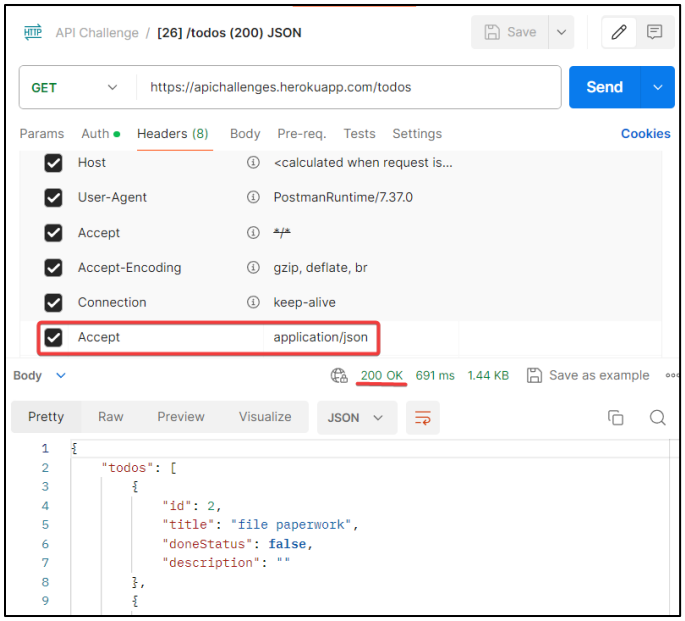


25	GET /todos (200) XML	true	Issue a GET request on the `/todos` end point with an `Accept` header of `application/xml` to receive results in XML format ▶ Solution

26. Receive a result in JSON format

26	GET /todos (200) JSON	false	Issue a GET request on the `/todos` end point with an `Accept` header of `application/json` to receive results in JSON format ▶ Solution

Solution:

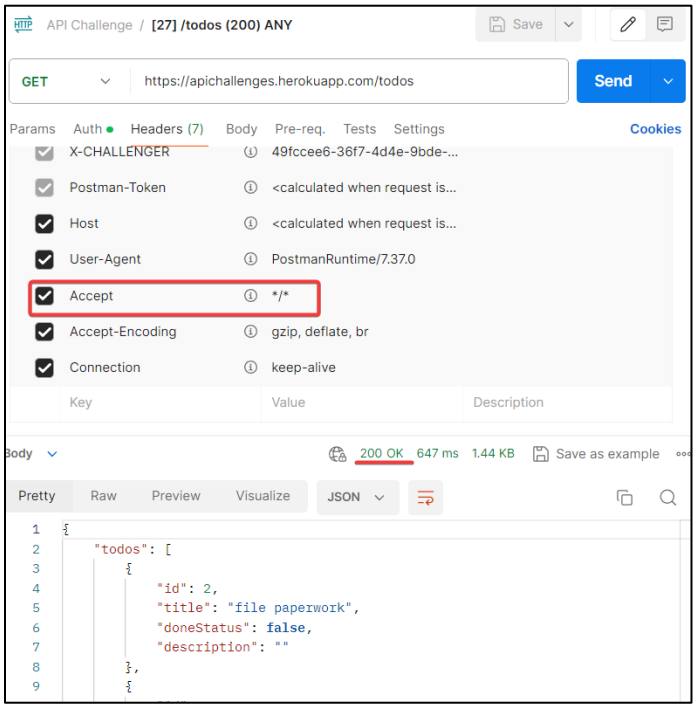


26	GET /todos (200) JSON	true	Issue a GET request on the `/todos` end point with an `Accept` header of `application/json` to receive results in JSON format ▶ Solution

27. Receive a result in default JSON format

27	GET /todos (200) ANY	false	Issue a GET request on the `/todos` end point with an `Accept` header of `*/*` to receive results in default JSON format ▶ Solution

Solution:

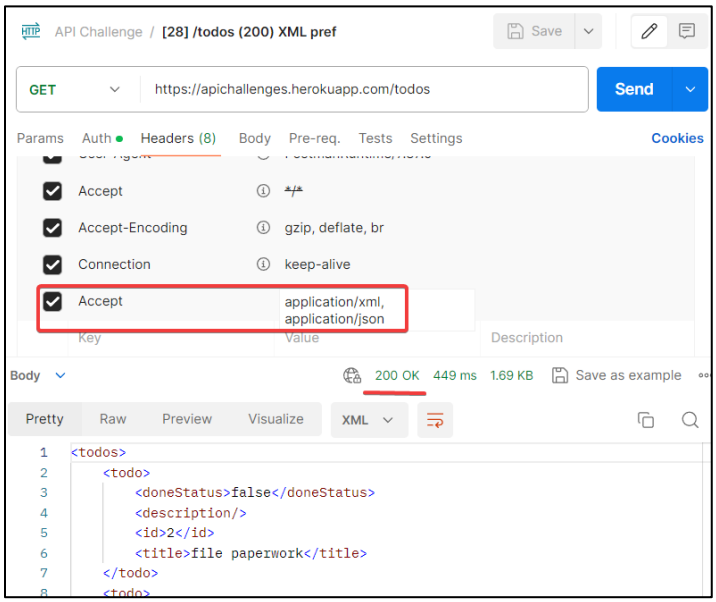


27	GET /todos (200) ANY	true	Issue a GET request on the `/todos` end point with an `Accept` header of `*/*` to receive results in default JSON format ▶ Solution

28. Receive a result in the XML format

28	GET /todos (200) XML pref	false	Issue a GET request on the `/todos` end point with an `Accept` header of `application/xml, application/json` to receive results in the preferred XML format ▸ Solution

Solution:

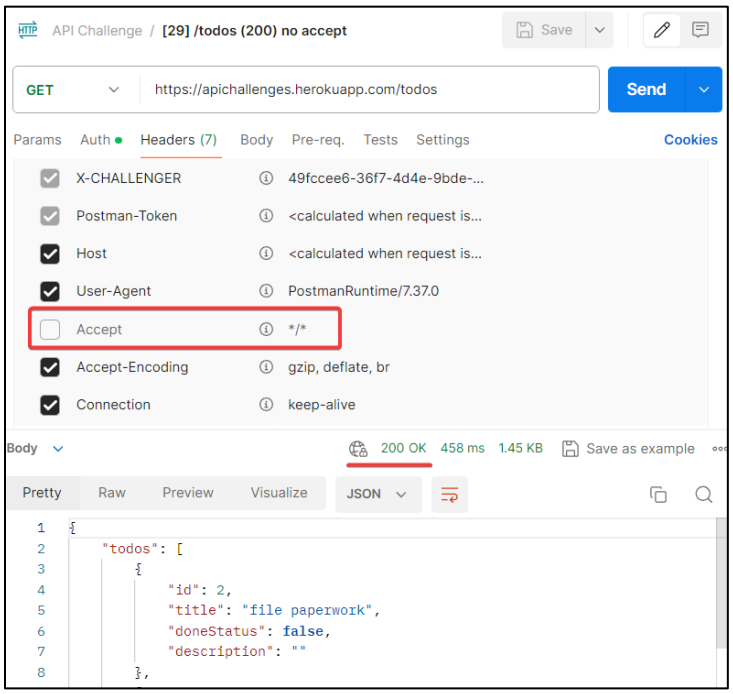


28	GET /todos (200) XML pref	true	Issue a GET request on the `/todos` end point with an `Accept` header of `application/xml, application/json` to receive results in the preferred XML format ▸ Solution

29. Receive results in default JSON format

29	GET /todos (200) no accept	false	Issue a GET request on the `/todos` end point with no `Accept` header present in the message to receive results in default JSON format ▸ Solution

Solution:

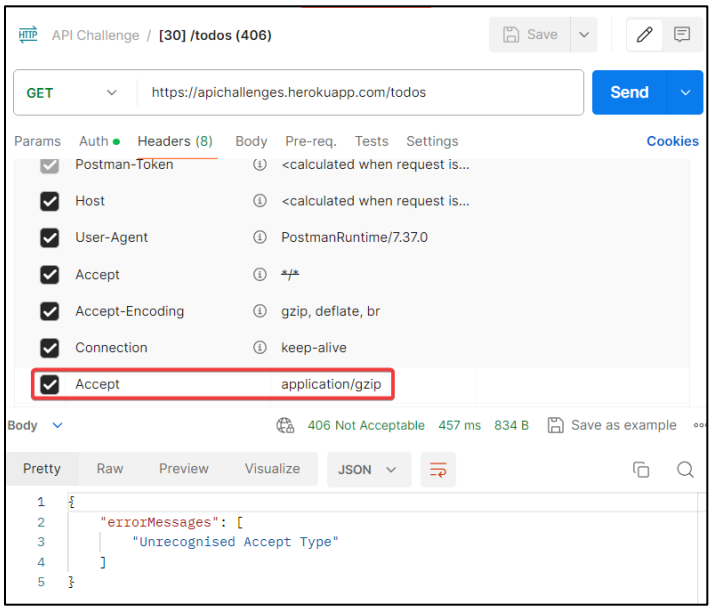


29	GET /todos (200) no accept	true	Issue a GET request on the `/todos` end point with no `Accept` header present in the message to receive results in default JSON format ▸ Solution

30. Create a GET request with an ‘Accept’ header ‘application/gzip’

30	GET /todos (406)	false	Issue a GET request on the `/todos` end point with an `Accept` header `application/gzip` to receive 406 'NOT ACCEPTABLE' status code ▸ Solution

Solution:



30	GET /todos (406)	true	Issue a GET request on the `/todos` end point with an `Accept` header `application/gzip` to receive 406 'NOT ACCEPTABLE' status code ▸ Solution

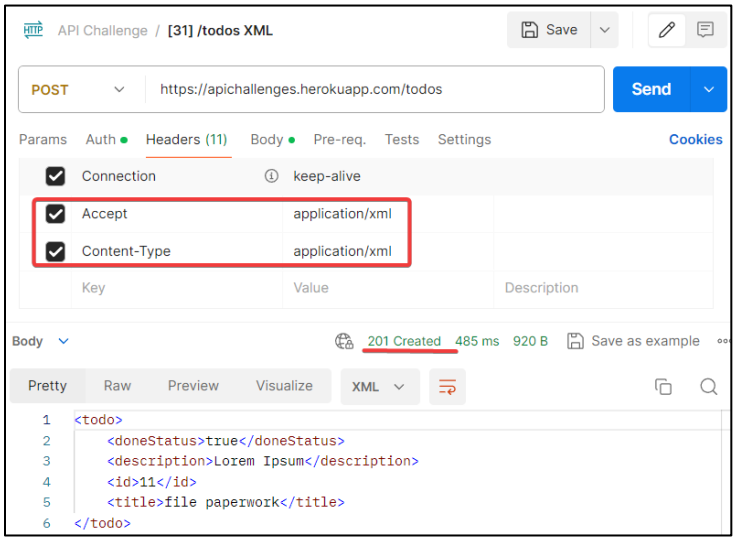
31. Create a todo record using Content-Type ‘application/xml’

Content-Type Challenges

The `Content-Type` header, tells the server what format type your 'body' content is, e.g. are you sending XML or JSON.

ID	Challenge	Done	Description
31	POST /todos XML	false	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/xml`, and Accepting only XML ie. Accept header of `application/xml` ▸ Solution

Solution:

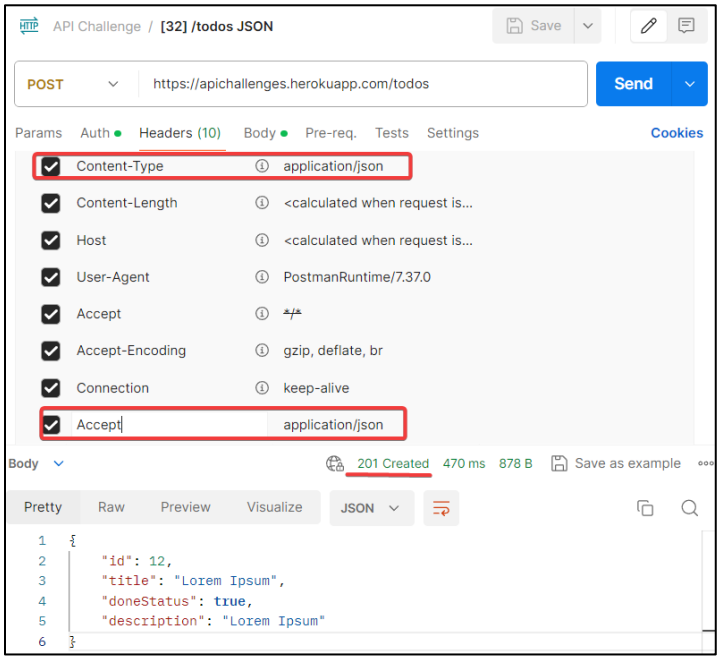


31	POST /todos XML	true	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/xml`, and Accepting only XML ie. Accept header of `application/xml` ▸ Solution

32. Create POST request using Content-Type ‘application/json

32	POST /todos JSON	false	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/json`, and Accepting only JSON ie. Accept header of `application/json` ▸ Solution

Solution:

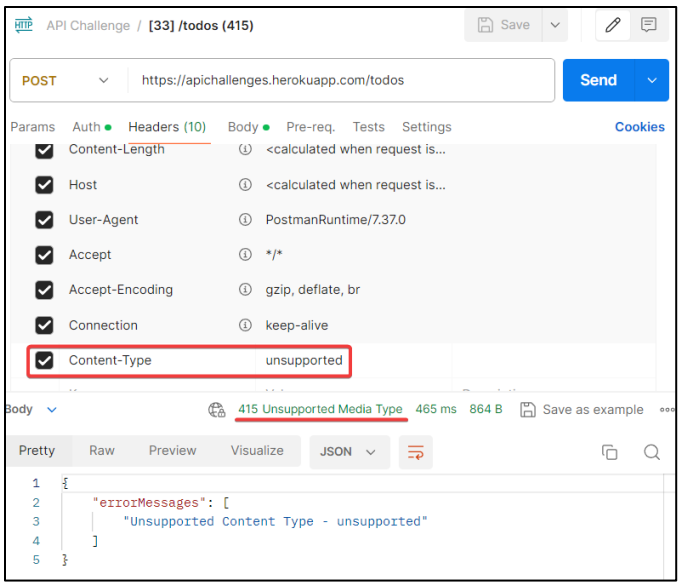


32	POST /todos JSON	true	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/json`, and Accepting only JSON ie. Accept header of `application/json` ▸ Solution

33. Create a request with an unsupported content type

33	POST /todos (415)	false	Issue a POST request on the `/todos` end point with an unsupported content type to generate a 415 status code ▸ Solution

Solution:



33	POST /todos (415)	true	Issue a POST request on the `/todos` end point with an unsupported content type to generate a 415 status code ▸ Solution

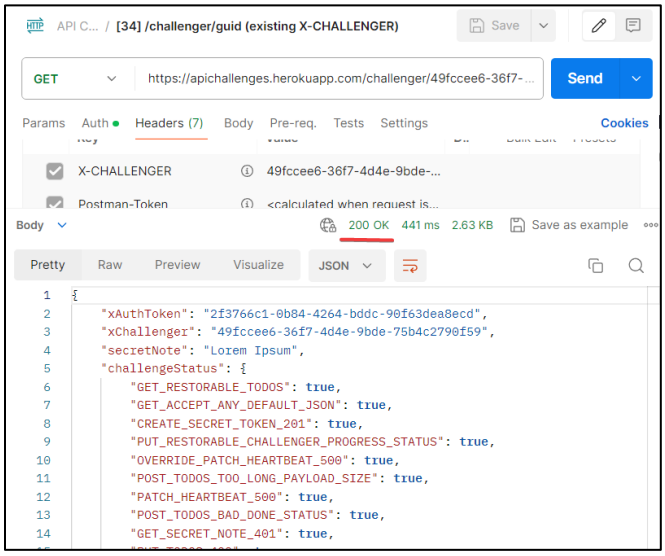
34. Return the progress data payload for current challenger GUID

Fancy a Break? Restore your session

Your challenge progress can be saved, and as long as you remember you challenger ID you can restore it. Leaving a challenger idle in the system for more than 10 minutes will remove hte challenger from memory. Challenger status and the todos database can be saved to, and restored from, the browser localStorage.

ID	Challenge	Done	Description
34	GET /challenger/guid (existing X-CHALLENGER)	false	Issue a GET request on the `/challenger/{guid}` end point, with an existing challenger GUID. This will return the progress data payload that can be used to later restore your progress to this status. ▸ Hints

Solution:

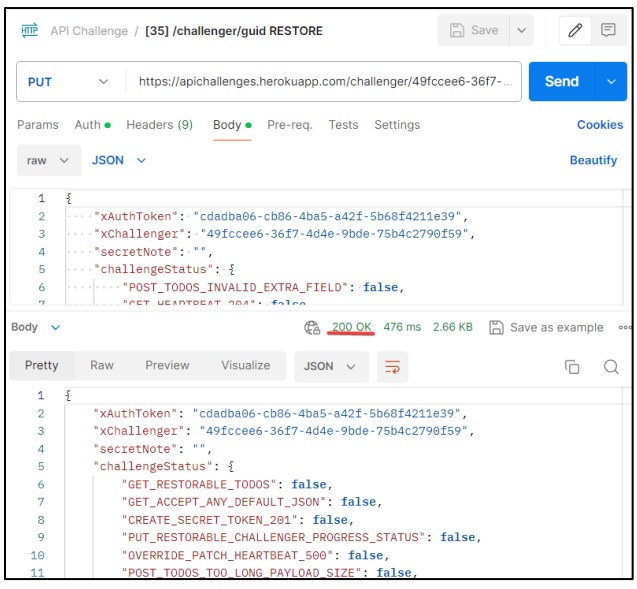


34	GET /challenger/guid (existing X-CHALLENGER)	true	Issue a GET request on the `/challenger/{guid}` end point, with an existing challenger GUID. This will return the progress data payload that can be used to later restore your progress to this status. ▸ Hints

35. Restore the challenger's progress

35	PUT /challenger/guid RESTORE	false	Issue a PUT request on the `/challenger/{guid}` end point, with an existing challenger GUID to restore that challenger's progress into memory. ▸ Hints ▸ Solution

Solution:

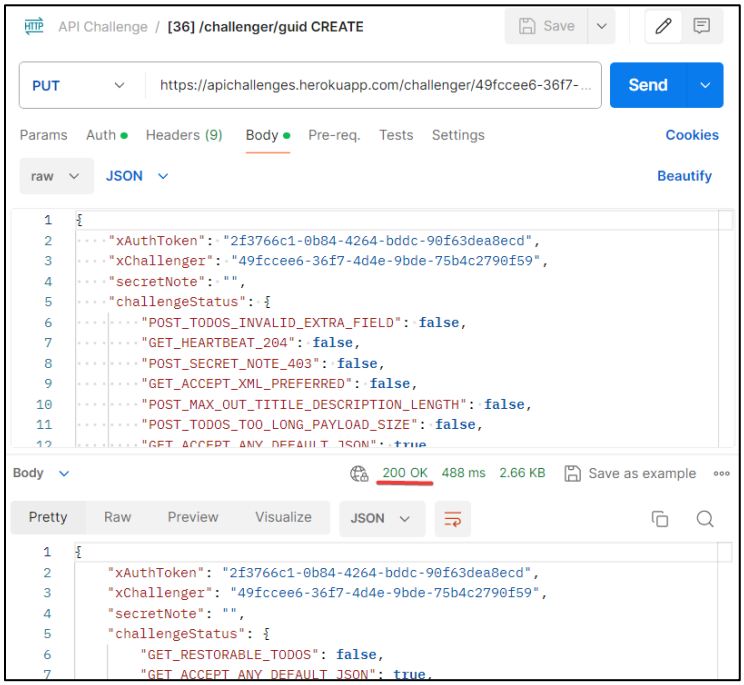


35	PUT /challenger/guid RESTORE	true	Issue a PUT request on the `/challenger/{guid}` end point, with an existing challenger GUID to restore that challenger's progress into memory. ▸ Hints ▸ Solution

36. Restore challenger's progress, when challenger GUID is not currently in memory

36	PUT /challenger/guid CREATE	false	Issue a PUT request on the `/challenger/{guid}` end point, with a challenger GUID not currently in memory to restore that challenger's progress into memory. ▸ Hints ▸ Solution

Solution:

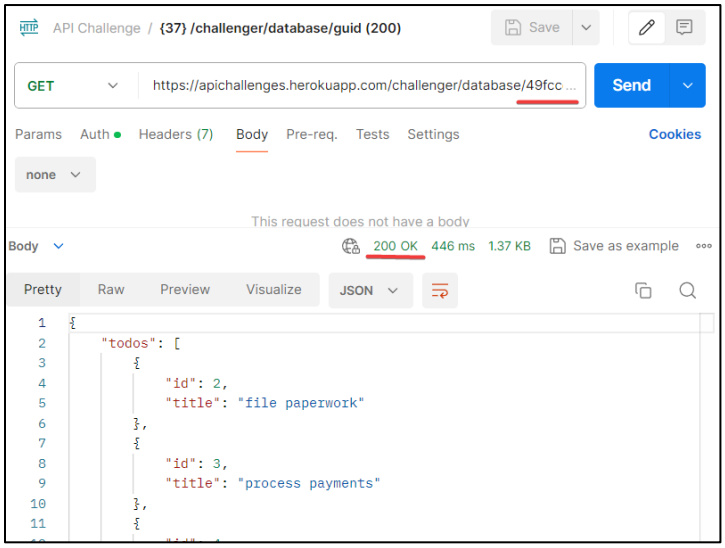


36	PUT /challenger/guid CREATE	true	Issue a PUT request on the `/challenger/{guid}` end point, with a challenger GUID not currently in memory to restore that challenger's progress into memory. ▸ Hints ▸ Solution

37. Retrieve the current todos database for the user.

37	GET /challenger/ database/guid (200)	false	Issue a GET request on the `/challenger/ database/{guid}` end point, to retrieve the current todos database for the user. You can use this to restore state later. ▸ Hints

Solution:

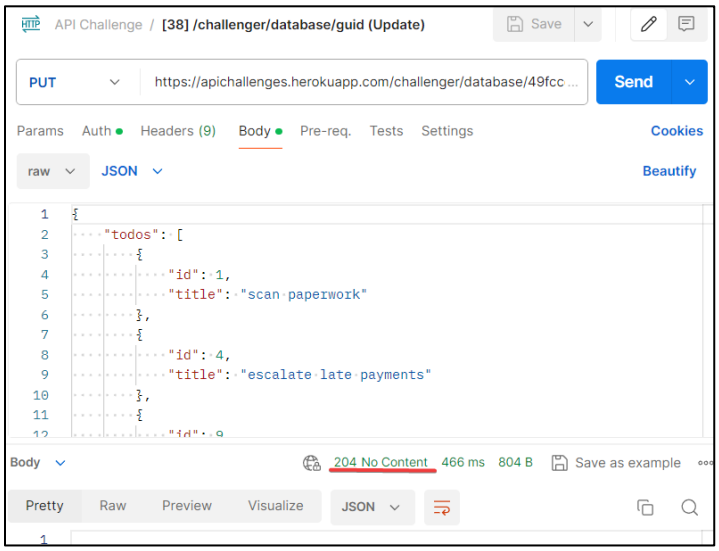


37	GET /challenger/database/guid (200)	true	Issue a GET request on the `/challenger/database/{guid}` end point, to retrieve the current todos database for the user. You can use this to restore state later. ▸ Hints

38. Restore the todo database in memory

38	PUT /challenger/ database/guid (Update)	false	Issue a PUT request on the `/challenger/ database/{guid}` end point, with a payload to restore the Todos database in memory. ▸ Hints ▸ Solution

Solution:



38	PUT /challenger/database/guid (Update)	true	Issue a PUT request on the `/challenger/database/{guid}` end point, with a payload to restore the Todos database in memory. ▸ Hints ▸ Solution

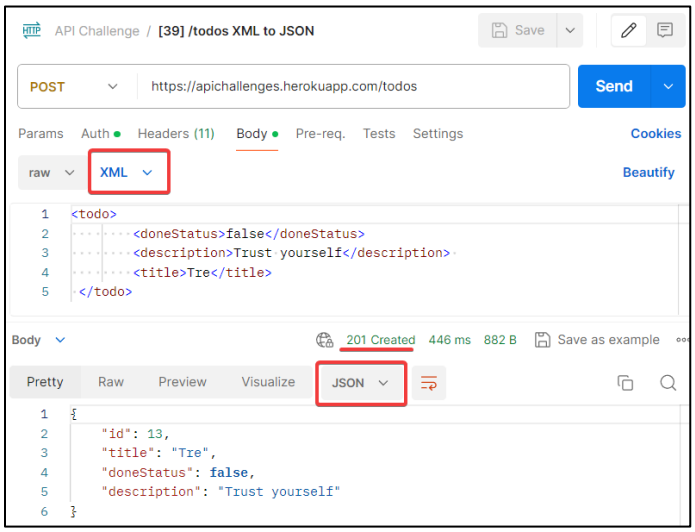
39. Create a todo using Content-Type ‘application/xml’, but Accept ‘application/json’

Mix Accept and Content-Type Challenges

We can mix the `Accept` and `Content-Type` headers so that we can send JSON but receive XML. These challenges encourage you to explore some combinations.

ID	Challenge	Done	Description
39	POST /todos XML to JSON	false	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/xml` but Accept `application/json` ▸ Solution

Solution:



39	POST /todos XML to JSON	true	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/xml` but Accept `application/json` ▸ Solution

40. Create a todo using Content-Type ‘application/json’, but Accept ‘application/xml’

40	POST /todos JSON to XML	false	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/json` but Accept `application/xml` ▸ Solution

Solution:

40	POST /todos JSON to XML	true	Issue a POST request on the `/todos` end point to create a todo using Content-Type `application/json` but Accept `application/xml` ▸ Solution

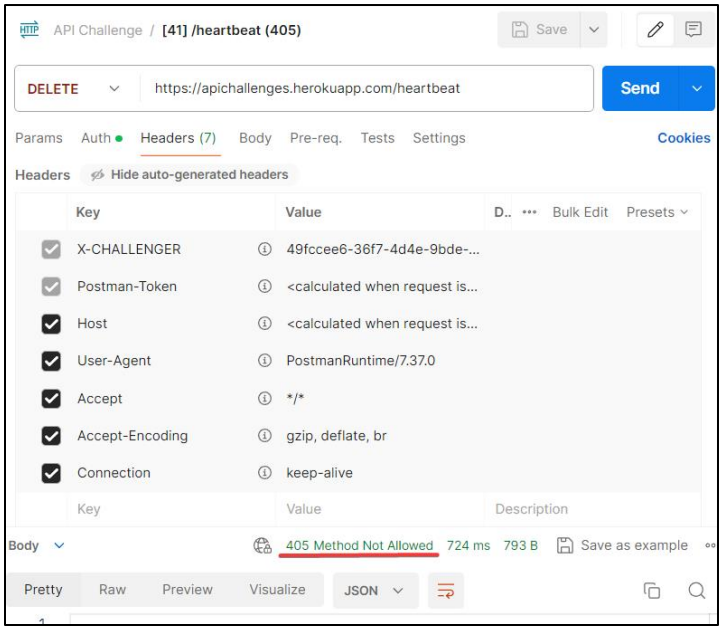
41. Receive 405 status code in DELETE request

Status Code Challenges

Status-codes are essential to understand, so we created some challenges that help you trigger more status codes. Remember to review httpstatuses.com to learn what the status codes mean.

ID	Challenge	Done	Description
41	DELETE / heartbeat (405)	false	Issue a DELETE request on the `/heartbeat` end point and receive 405 (Method Not Allowed) ▸ Solution

Solution:

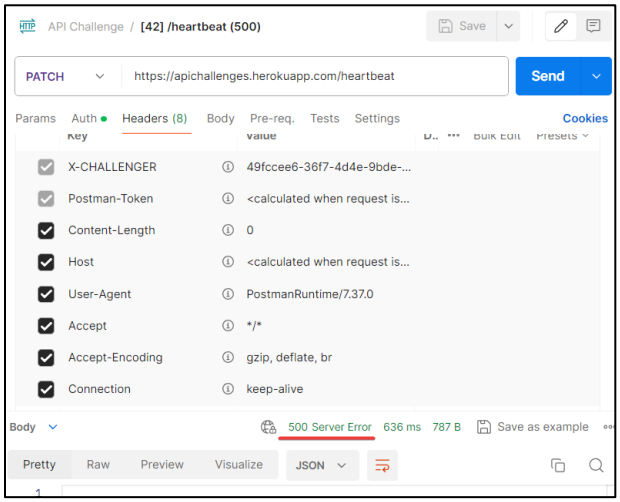


41	DELETE /heartbeat (405)	true	Issue a DELETE request on the `/heartbeat` end point and receive 405 (Method Not Allowed) ▸ Solution

42. Receive 500 status code in the PATCH request

42	PATCH /heartbeat (500)	false	Issue a PATCH request on the `/heartbeat` end point and receive 500 (internal server error) ▸ Solution

Solution:

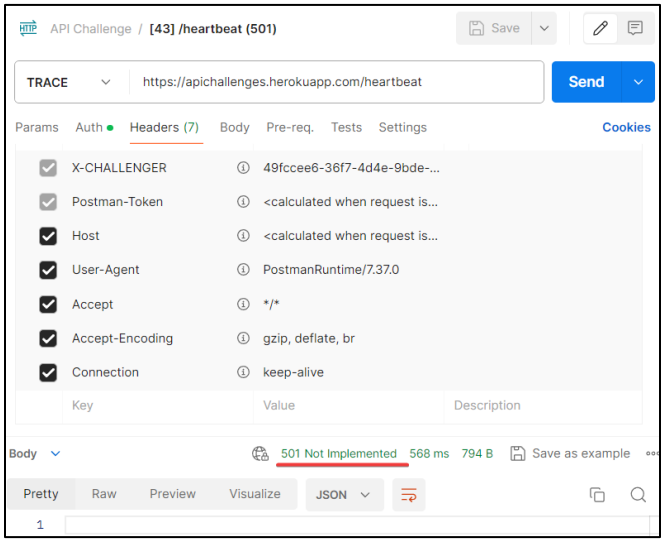


42	PATCH /heartbeat (500)	true	Issue a PATCH request on the `/heartbeat` end point and receive 500 (internal server error) ▸ Solution

43. Receive 501 status code in TRACE request

43	TRACE /heartbeat (501)	false	Issue a TRACE request on the `/heartbeat` end point and receive 501 (Not Implemented) ▸ Solution

Solution:

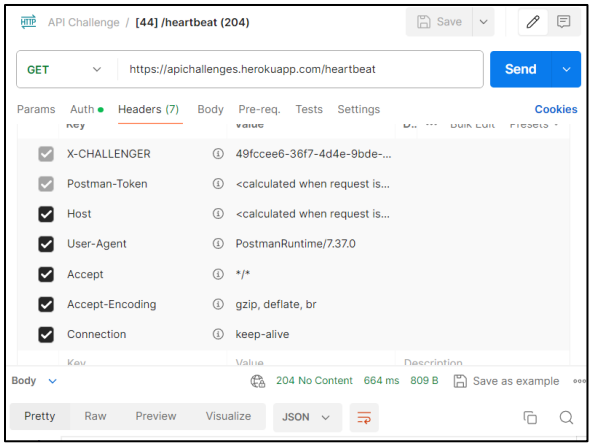


43	TRACE /heartbeat (501)	true	Issue a TRACE request on the `/heartbeat` end point and receive 501 (Not Implemented) ▸ Solution

44. Receive 204 status code when the server is running

44	GET /heartbeat (204)	false	Issue a GET request on the `/heartbeat` end point and receive 204 when server is running ▸ Solution

Solution:



44	GET /heartbeat (204)	true	Issue a GET request on the `/heartbeat` end point and receive 204 when server is running ▶ Solution

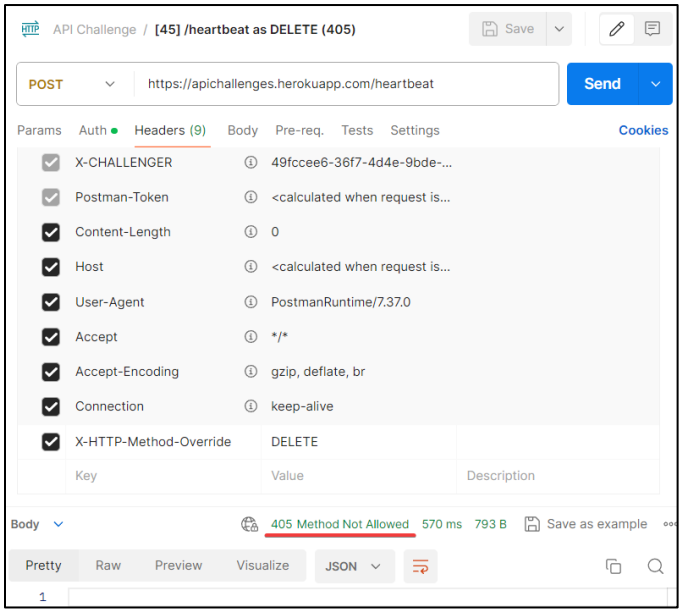
45. Create a POST request and receive 405 status code

HTTP Method Override Challenges

Some HTTP Clients can not send all verbs e.g. PATCH, DELETE, PUT. Use an X-HTTP-Method-Override header to simulate these with a POST request

ID	Challenge	Done	Description
45	POST /heartbeat as DELETE (405)	false	Issue a POST request on the `/heartbeat` end point and receive 405 when you override the Method Verb to a DELETE ▶ Hints ▶ Solution

Solution:

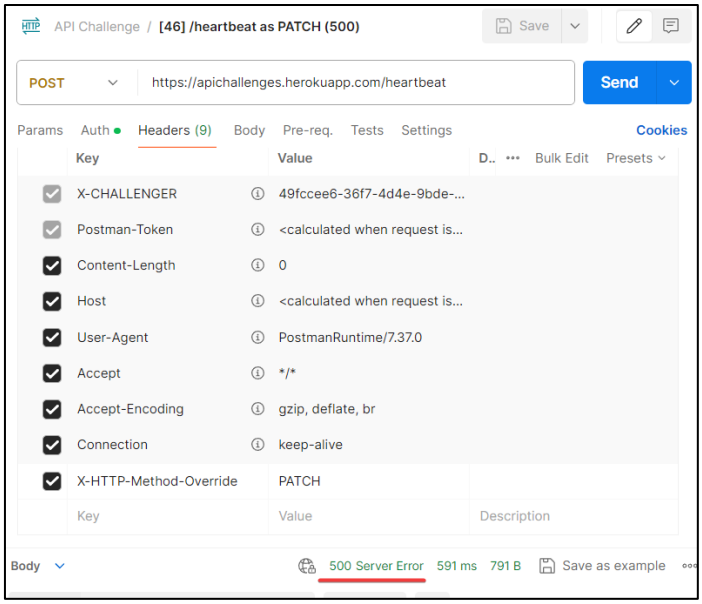


45	POST /heartbeat as DELETE (405)	true	Issue a POST request on the `/heartbeat` end point and receive 405 when you override the Method Verb to a DELETE ▶ Hints ▶ Solution

46. Receive 500 status code when override the Method Verb to a PATCH

46	POST /heartbeat as PATCH (500)	false	Issue a POST request on the `/heartbeat` end point and receive 500 when you override the Method Verb to a PATCH ▶ Hints ▶ Solution

Solution:

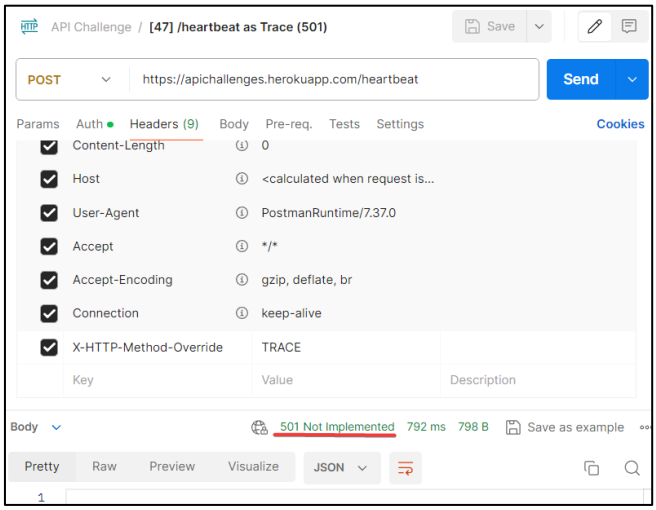


46	POST /heartbeat as PATCH (500)	true	Issue a POST request on the `/heartbeat` end point and receive 500 when you override the Method Verb to a PATCH ▸ Hints ▸ Solution

47. Receive 500 status code when override the Method Verb to a TRACE

47	POST /heartbeat as Trace (501)	false	Issue a POST request on the `/heartbeat` end point and receive 501 (Not Implemented) when you override the Method Verb to a TRACE ▸ Hints ▸ Solution

Solution:



47	POST /heartbeat as Trace (501)	true	Issue a POST request on the `/heartbeat` end point and receive 501 (Not Implemented) when you override the Method Verb to a TRACE ▸ Hints ▸ Solution

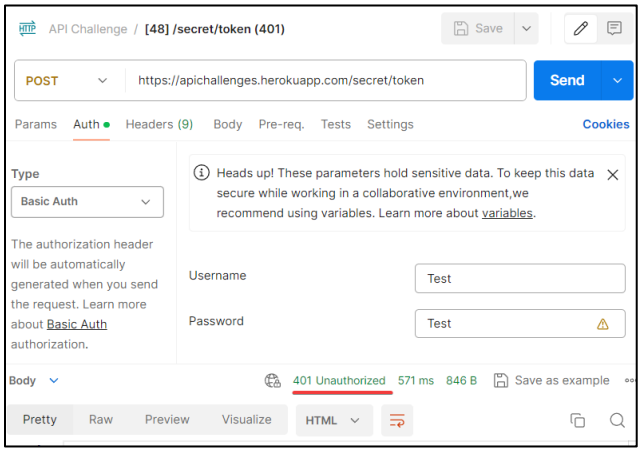
48. Receive 401 status code when Basic auth username/password is not admin/password

Authentication Challenges

Authentication is telling the system who you are. In multi-user mode you are already doing that with the X-CHALLENGER header, but we have added an extra level of security on the /secret section. So first Authenticate with Basic Authentication to find out the token to use for authorisation for later challenges.

ID	Challenge	Done	Description
48	POST /secret/token (401)	false	Issue a POST request on the `/secret/token` end point and receive 401 when Basic auth username/password is not admin/password ▶ Hints ▶ Solution

Solution:

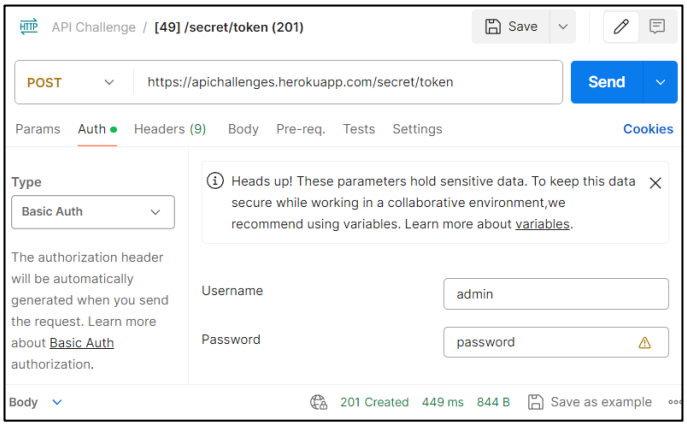


48	POST /secret/token (401)	true	Issue a POST request on the `/secret/token` end point and receive 401 when Basic auth username/password is not admin/password ▶ Hints ▶ Solution

49. Receive 201 status code when Basic auth has admin/password values for username/password fields

49	POST /secret/ token (201)	false	Issue a POST request on the `/secret/token` end point and receive 201 when Basic auth username/password is admin/password ▸ Hints ▸ Solution

Solution:



49	POST /secret/token (201)	true	Issue a POST request on the `/secret/token` end point and receive 201 when Basic auth username/password is admin/password ▶ Hints ▶ Solution

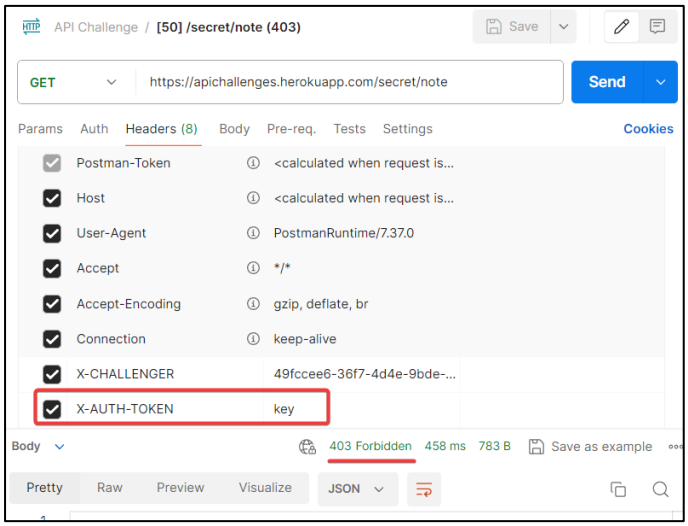
50. Receive 403 status code when X-AUTH-TOKEN doesn't match a valid token

Authorization Challenges

Once the system knows who you are, authorization is if you have the correct level of access. In these challenges the authorization is granted using a custom API header X-AUTH-TOKEN or using a Bearer Authorization header.

ID	Challenge	Done	Description
50	GET /secret/note (403)	false	Issue a GET request on the `/secret/note` end point and receive 403 when X-AUTH-TOKEN does not match a valid token ▶ Hints ▶ Solution

Solution:

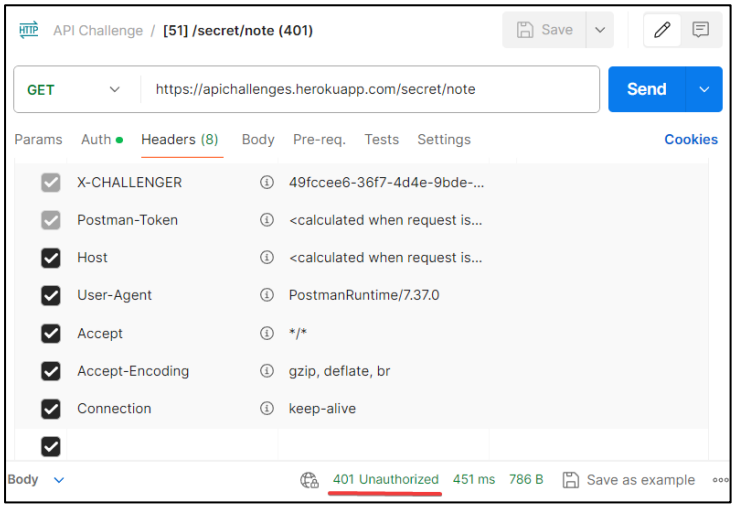


50	GET /secret/note (403)	true	Issue a GET request on the `/secret/note` end point and receive 403 when X-AUTH-TOKEN does not match a valid token ▶ Hints ▶ Solution

51. Receive 401 when no X-AUTH-TOKEN header present

51	GET /secret/ note (401)	false	Issue a GET request on the `/secret/note` end point and receive 401 when no X-AUTH-TOKEN header present ▸ Hints ▸ Solution

Solution:

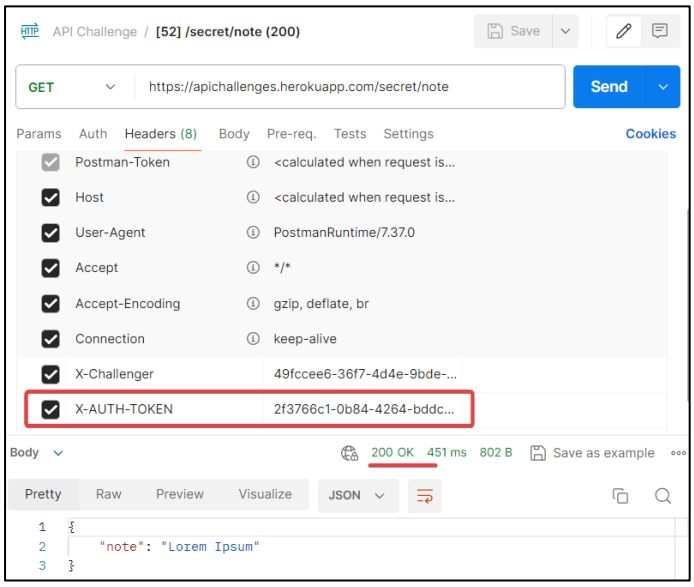


51	GET /secret/note (401)	true	Issue a GET request on the `/secret/note` end point and receive 401 when no X-AUTH-TOKEN header present ▸ Hints ▸ Solution

52. Receive 200 status code when valid X-AUTH-TOKEN used – response body should contain the note

52	GET /secret/ note (200)	false	Issue a GET request on the `/secret/note` end point receive 200 when valid X-AUTH-TOKEN used - response body should contain the note ▸ Hints ▸ Solution

Solution:

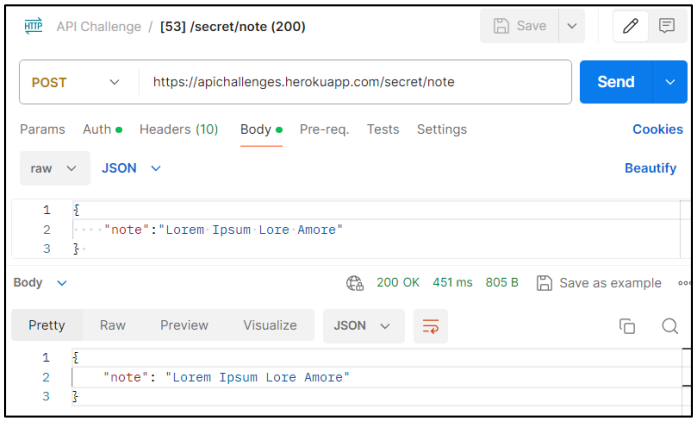


52	GET /secret/note (200)	true	Issue a GET request on the `/secret/note` end point receive 200 when valid X-AUTH-TOKEN used - response body should contain the note ▸ Hints ▸ Solution

53. Receive 200 status code when valid X-AUTH-TOKEN used.

53	POST /secret/ note (200)	false	Issue a POST request on the `/secret/note` end point with a note payload e.g. {"note":"my note"} and receive 200 when valid X-AUTH-TOKEN used. Note is maximum length 100 chars and will be truncated when stored. ▸ Hints ▸ Solution

Solution:

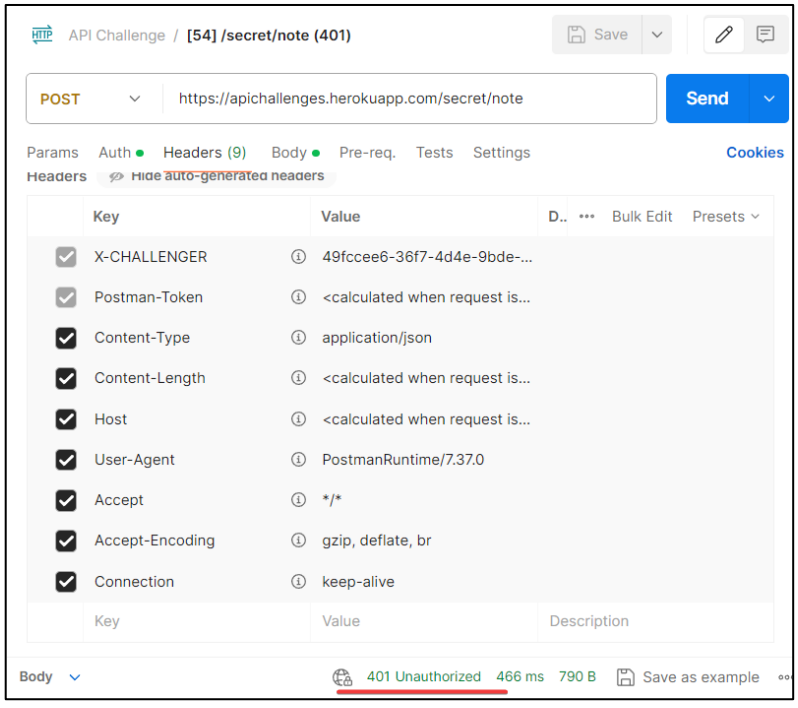


53	POST /secret/note (200)	true	Issue a POST request on the `/secret/note` end point with a note payload e.g. {"note":"my note"} and receive 200 when valid X-AUTH-TOKEN used. Note is maximum length 100 chars and will be truncated when stored. ▸ Hints ▸ Solution

54. Receive 401 when no X-AUTH-TOKEN present

54	POST /secret/ note (401)	false	Issue a POST request on the `/secret/note` end point with a note payload {"note":"my note"} and receive 401 when no X-AUTH-TOKEN present ▸ Hints ▸ Solution

Solution:

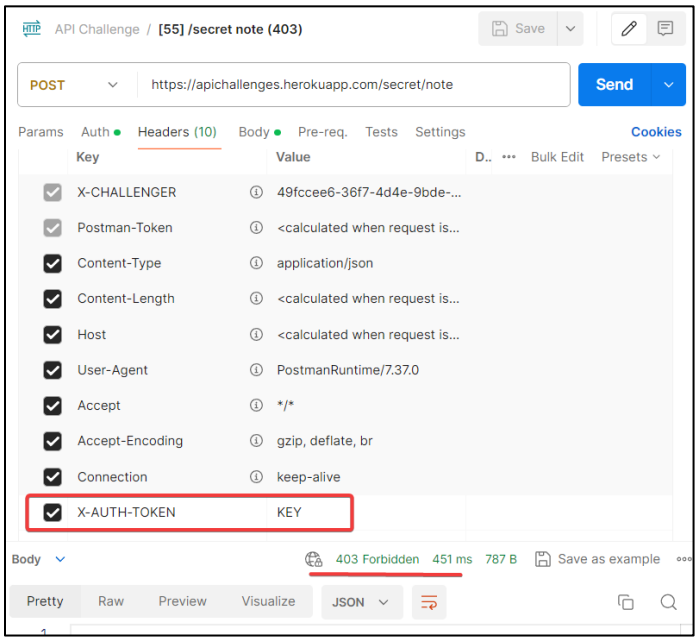


54	POST /secret/note (401)	true	Issue a POST request on the `/secret/note` end point with a note payload {"note":"my note"} and receive 401 when no X-AUTH-TOKEN present ▸ Hints ▸ Solution

55.Receive 403 status code when X-AUTH-TOKEN does not match a valid token

55	POST /secret/note (403)	false	Issue a POST request on the `/secret/note` end point with a note payload {"note":"my note"} and receive 403 when X-AUTH-TOKEN does not match a valid token ▶ Hints ▶ Solution

Solution:

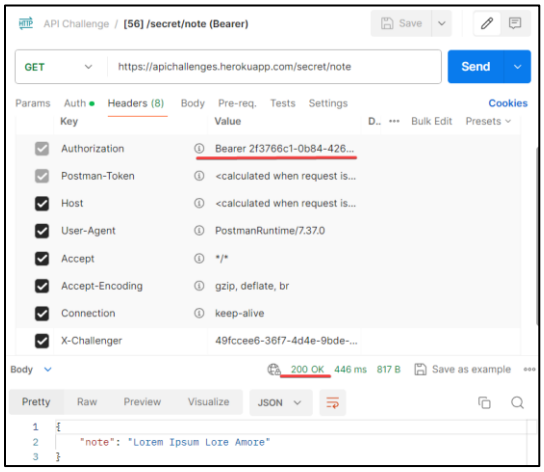


55	POST /secret/note (403)	true	Issue a POST request on the `/secret/note` end point with a note payload {"note":"my note"} and receive 403 when X-AUTH-TOKEN does not match a valid token ▶ Hints ▶ Solution

56. Receive 200 when using the X-AUTH-TOKEN value as an Authorization Bearer token

56	GET /secret/ note (Bearer)	false	Issue a GET request on the `/secret/note` end point receive 200 when using the X-AUTH-TOKEN value as an Authorization Bearer token - response body should contain the note ▶ Hints ▶ Solution

Solution:

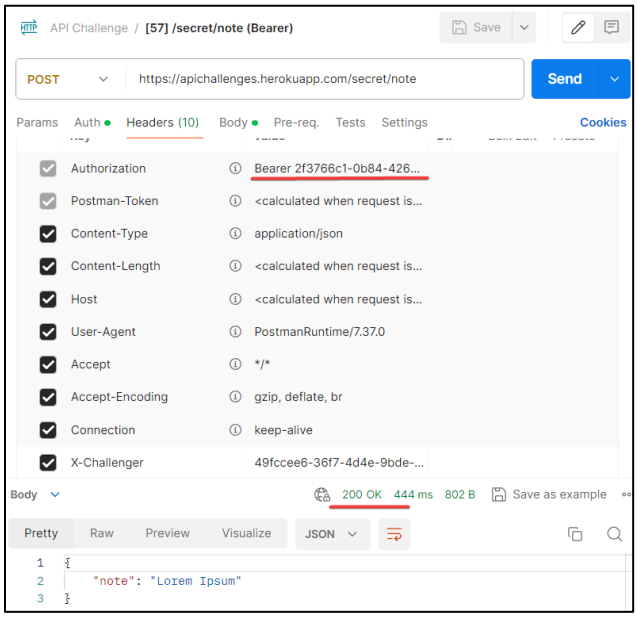


56	GET /secret/note (Bearer)	true	Issue a GET request on the `/secret/note` end point receive 200 when using the X-AUTH-TOKEN value as an Authorization Bearer token - response body should contain the note ▶ Hints ▶ Solution

57. Receive 200 status code when a valid X-AUTH-TOKEN value used as an Authorization Bearer token.

57	POST /secret/ note (Bearer)	false	Issue a POST request on the `/secret/note` end point with a note payload e.g. {"note":"my note"} and receive 200 when valid X-AUTH-TOKEN value used as an Authorization Bearer token. Status code 200 received. Note is maximum length 100 chars and will be truncated when stored. ▸ Hints ▸ Solution

Solution:



57	POST /secret/note (Bearer)	true	Issue a POST request on the `/secret/note` end point with a note payload e.g. {"note":"my note"} and receive 200 when valid X-AUTH-TOKEN value used as an Authorization Bearer token. Status code 200 received. Note is maximum length 100 chars and will be truncated when stored. ▸ Hints ▸ Solution

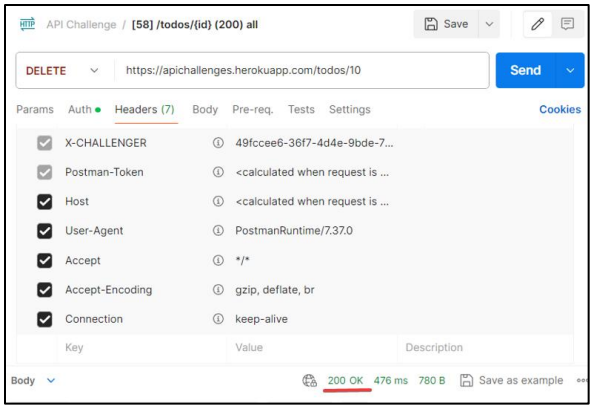
58. Delete the last todo record in the system so that there are no more todos in the system

Miscellaneous Challenges

We left these challenges to the end because they seemed fun, but... different.

ID	Challenge	Done	Description
58	DELETE /todos/{id} (200) all	false	Issue a DELETE request to successfully delete the last todo in system so that there are no more todos in the system ▶ Hints

Solution:

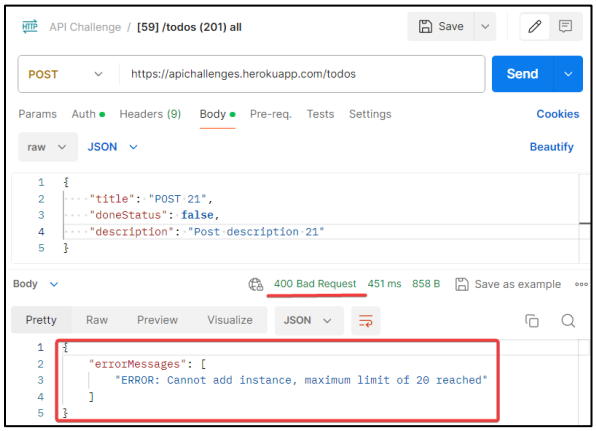


58	DELETE /todos/{id} (200) all	true	Issue a DELETE request to successfully delete the last todo in system so that there are no more todos in the system ▶ Hints

59. Create as many POST requests as possible for todos

59	POST /todos (201) all	false	Issue as many POST requests as it takes to add the maximum number of TODOS allowed for a user. The maximum number should be listed in the documentation.

Solution:



59	POST /todos (201) all	true	Issue as many POST requests as it takes to add the maximum number of TODOS allowed for a user. The maximum number should be listed in the documentation.