Use the Descriptions of the challenges below to explore the API and solve the challenges. Remember to use the API documentation to see the format of POST requests.

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)	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

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)	true	Issue a GET request on the `/challenges` end point ▸ Solution

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

HEAD Challenges

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

ID	Challenge	Done	Description
07	HEAD /todos (200)	true	Issue a HEAD request on the `/todos` end point ▸ Solution

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 assocated GET request, and amending it.

ID	Challenge	Done	Description
08	POST /todos (201)	true	Issue a POST request to successfully create a todo ▸ Solution
09	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
10	POST /todos (400) doneStatus	true	Issue a POST request to create a todo but fail validation on the `doneStatus` field ▸ Solution

Update Challenges with POST

Use a POST request to amend something that already exists. These are 'partial' content updates so you useually 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
11	POST /todos/{id} (200)	true	Issue a POST request to successfully update a todo ▸ Hints ▸ Solution

DELETE Challenges

Use a DELETE request to delete an entity. Since this is an extreme request, normall 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
12	DELETE /todos/{id} (200)	true	Issue a DELETE request to successfully delete a todo ► Hints ► Solution

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
13	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

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
14	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

ID	Challenge	Done	Description
			▸ Solution
15	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
16	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
17	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
18	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
19	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

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

ID	Challenge	Done	Description
20	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
21	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
22	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

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
23	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
24	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

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
25	DELETE /heartbeat (405)	true	Issue a DELETE request on the `/heartbeat` end point and receive 405 (Method Not Allowed) ▸ Solution
26	PATCH /heartbeat (500)	true	Issue a PATCH request on the `/heartbeat` end point and receive 500 (internal server error) ▸ Solution
27	TRACE /heartbeat (501)	true	Issue a TRACE request on the `/heartbeat` end point and receive 501 (Not Implemented) ▸ Solution
28	GET /heartbeat (204)	true	Issue a GET request on the `/heartbeat` end point and receive 204 when server is running ▸ Solution

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
29	POST /secret /token (401)	true	Issue a POST request on the `/secret/token` end point and receive 401 when Basic auth username/password

ID	Challenge	Done	Description
			is not admin/password ▸ Hints ▸ Solution
30	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

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
31	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
32	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
33	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

ID	Challenge	Done	Description
			▸ Solution
34	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
35	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
36	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
37	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

ID	Challenge	Done	Description
38	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

Miscellaneous Challenges

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

ID	Challenge	Done	Description
39	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