API Challenges Progress

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.

Progress, and the TODOs database content can be saved to, and restored from, LocalStorage in the browser - or managed via the API.

Progress For Challenger ID 8e838301-6013-4190-b965-1fa01755404e

Refresh Status ☑ Auto Save Todos and Progress Locally on refresh

59 Challenges: 59 complete, 0 remain.
saved progress Restore Locally Saved Progress

20 todos in database. View Todos
saved todos Restore Locally Saved Todos Data

▸ Previously Used Challenger GUIDs

Input a Challenger GUID to use

Challenge Sections

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

Back to Section List

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

Back to Section List

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
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

Back to Section List

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)	true	Issue a HEAD request on the `/todos` end point ► Solution

Back to Section List

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)	true	Issue a POST request to successfully create a todo ► Solution
10	POST /todos (400) doneStatus	true	Issue a POST request to create a todo but fail validation on the `doneStatus` field ► 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	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	POST /todos (201) max out content	true	Issue a POST request to create a todo with maximum length title and description fields. ► Hints
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	POST /todos (400) extra	true	Issue a POST request to create a todo but fail validation because your payload contains an unrecognised field. ► Hints

Back to Section List

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)	true	Issue a PUT request to unsuccessfully create a todo

Back to Section List

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)	true	Issue a POST request to successfully update a todo

ID	Challenge	Done	Description
			► 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

Back to Section List

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)	true	Issue a PUT request to update an existing todo with a complete payload i.e. title, description and donestatus.
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	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	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

Back to Section List

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)	true	Issue a DELETE request to successfully delete a todo ► Hints ► Solution

Back to Section List

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)	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.

ID	Challenge	Done	Description
			► Solution

Back to Section List

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	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	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	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	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	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	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

Back to Section List

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	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	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`

ID	Challenge	Done	Description
			► 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

Back to Section List

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)	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	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	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	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	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

Back to Section List

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

ID	Challenge	Done	Description
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

Back to Section List

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)	true	Issue a DELETE request on the `/heartbeat` end point and receive 405 (Method Not Allowed) ► Solution
42	PATCH /heartbeat (500)	true	Issue a PATCH request on the `/heartbeat` end point and receive 500 (internal server error) ► Solution
43	TRACE /heartbeat (501)	true	Issue a TRACE request on the `/heartbeat` end point and receive 501 (Not Implemented) ► Solution
44	GET /heartbeat (204)	true	Issue a GET request on the `/heartbeat` end point and receive 204 when server is running ► Solution

Back to Section List

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)	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	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

ID	Challenge	Done	Description
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

Back to Section List

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)	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	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

Back to Section List

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)	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	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	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	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.

ID	Challenge	Done	Description
			► Hints ► 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	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	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	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

Back to Section List

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	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	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.

Back to Section List