“angular-in-memory-web-api” versions

> This in-memory-web-api exists primarily to support the Angular documentation. It is not supposed to emulate every possible real world web API and is not intended for production use.
>
> Most importantly, it is ***always experimental***.

We will make breaking changes and we won’t feel bad about it because this is a development tool, not a production product. We do try to tell you about such changes in this `CHANGELOG.md` and we fix bugs as fast as we can.

0.11.0 (2020-05-13)

- update to support Angular v10.
- no functional changes.

0.9.0 (2019-06-20)

- update to support Angular version 8.x and forward
- no functional changes

0.8.0 (2018-12-06)

- remove `@angular/http` support
- no functional changes

BREAKING CHANGE This version no longer supports any functionality for `@angular/http`. Please use `@angular/common/http` instead.

0.7.0 (2018-10-31)

- update to support Angular v7.
- no functional changes

0.6.1 (2018-05-04)

- update to Angular and RxJS v6 releases

0.6.0 (2018-03-22)

Migrate to Angular v6 and RxJS v6 (rc and beta)

Note that this release is pinned to Angular “^6.0.0-rc.0” and RxJS “^6.0.0-beta.1”. Will likely update again when they are official.

BREAKING CHANGE This version depends on RxJS v6 and is not backward compatible with earlier RxJS versions.

0.5.4 (2018-03-09)

Simulated HTTP error responses were not delaying the prescribed time when using RxJS `delay()` because it was short-circuited by the ErrorResponse. New `delayResponse` function does it right. Should not break you unless you incorrectly expected no delay for errors.

Also, this library no longer calls RxJS `delay()` which may make testing with it easier (Angular TestBed does not handle RxJS `delay()` well because that operator uses `interval()`).

Also fixes type error (issue #180).

0.5.3 (2018-01-06) Can make use of `HttpParams` which yields a `request.urlWithParams`. Added supporting `HeroService.searchHeroes(term: string)` and test.

0.5.2 (2017-12-10) No longer modify the request data coming from client. Fixes #164

0.5.1 (2017-10-21) Support Angular v5.

0.5.0 (2017-10-05) **BREAKING CHANGE**: HTTP response data no longer wrapped in object w/ `data` property by default.

In this release, the `dataEncapsulation` configuration default changed from `false` to `true`. The HTTP response body holds the data values directly rather than an object that encapsulates those values, `{data: ...}`. This is a **breaking change that affects almost all existing apps!**

Changing the default to `false` is a **breaking change**. Pre-existing apps that did not set this property explicitly will be broken because they expect encapsulation and are probably mapping the HTTP response results from the `data` property like this:

```
.map(data => data.data as Hero[])
```

To migrate, simply remove that line everywhere.

If you would rather keep the web api's encapsulation, `{data: ...}`, set `dataEncapsulation` to `true` during configuration as in the following example:

```
HttpClientInMemoryWebApiModule.forRoot(HeroInMemDataService, { dataEncapsulation: true })
```

We made this change because

1. Almost everyone seems to hate the encapsulation

2. Encapsulation requires mapping to get the desired data out. With old `Http` that isn't *too* bad because you needed to map to get data anyway (`res => res.json()`). But it is really ugly for `HttpClient` because you can't use the type HTTP method type parameter (e.g., `get<entity-type>`) and you have to map out of the data property (`.map(data => data.data as Hero[]`). That extra step requires explanations that distract from learning `HttpClient` itself. Now you just write `http.get<Hero[]>()` and you've got data (please add error handling).

3. While you could have turned off encapsulation with configuration as of v.0.4, to do so took yet another step that you'd have to discover and explain. A big reason for the in-mem web api is to make it easy to introduce and

demonstrate HTTP operations in Angular. The *out-of-box* experience is more important than avoiding a breaking change.

4. The security flaw that prompted encapsulation seems to have been mitigated by all (almost all?) the browsers that can run an Angular (v2+) app. We don't think it's needed anymore.

5. A most real world APIs today will not encapsulate; they'll return the data in the body without extra ceremony.

0.4.6 (2017-09-13) - improves README - updates v0.4.0 entry in the CHANGELOG to describe essential additions to SystemJS configuration. - no important functional changes.

0.4.5 (2017-09-11) Feature - offer separate `HttpClientInMemoryWebApiModule` and `HttpInMemoryWebApiModule`.

closes #140

0.4.4 (2017-09-11) closes #136

A **breaking change** if you expected `genId` to generate ids for a collection with non-numeric `item.id`.

0.4.3 (2017-09-11) Refactoring for clarity and to correctly reflect intent. A **breaking change** only if your customizations depend directly and explicitly on `RequestInfo` or the `get`, `delete`, `post`, or `put` methods.

- replace all `switchMap` with `concatMap` because, in all previous uses of `switchMap`, I really meant to wait for the source observable to complete *before* beginning the inner observable whereas `switchMap` starts the inner observable right away.

- restored `collection` to the `RequestInfo` interface and set it in `handleRequest_`

- `get`, `delete`, `post`, and `put` methods get the `collection` from `requestInfo`; simplifies their signatures to one parameter.

0.4.2 (2017-09-08) - Postpones the in-memory database initialization (via `resetDb`) until the first HTTP request.

- Your `createDb` method *can* be asynchronous. You may return the database object (synchronous), an observable of it, or a promise of it. Issue #113.

- fixed some rare race conditions.

0.4.1 (2017-09-08) **Support PassThru.**

The passthru feature was broken by 0.4.0 - add passthru to both `Http` and `HttpClient` - test passThru feature with jasmine-ajax mock-ajax plugin to intercept Angular's attempt to call browser's XHR - update devDependency packages - update karma.conf with jasmine-ajax plugin

0.4.0 (2017-09-07) **Theme: Support `HttpClient` and add tests**. See PR #130.

The 0.4.0 release was a major overhaul of this library.

You don't have to change your existing application *code* if your app uses this library without customizations.

But this release's **breaking changes** affect developers who used the customization features or loaded application files with SystemJS.

BREAKING CHANGES: Massive refactoring. Many low-level and customization options have changed. Apps that stuck with defaults should be (mostly) OK.

If you're loading application files with **SystemJS** (as you would in a plunker), see the instructions below.

- added support for `HttpClient` -> renaming of backend service classes
- added tests
- refactor existing code to support tests
- correct bugs and clarify choices as result of test
- add some configuration options
  - dataEncapsulation (issue #112, pr#123)
  - post409
  - put404b
- `POST commands/resetDb` passes the request to your `resetDb` method so you can optionally reset the database dynamically to arbitrary initial states (issue #128)
- when HTTP method interceptor returns null/undefined, continue with service's default processing (pr #120)
- can substitute your own id generator, `geniD`
- parseUrl -> parseRequestUrl
- utility methods exposed in `RequestInfo.utils`
- reorganize files into src/app and src/in-mem
- adjust gulp tasks accordingly

Plunkers and SystemJS

If you're loading application files with **SystemJS** (as you would in a plunker), you'll have to configure it to load Angular's `umd.js` for `HttpModule` and the `tslib` package.

To see how, look in the `map` section of the `src/systemjs.config.js` for this project for the following two *additional* lines :

```
'@angular/common/http': 'npm:@angular/common/bundles/common-http.umd.js',
...
'tslib': 'npm:tslib/tslib.js',
```

You've already made these changes if you are using `HttpClient` today.

If you're sticking with the original Angular `Http` module, you *must make this change anyway!* Your app will break as soon as you run `npm install` and it installs >=v0.4.0.

If you're using webpack (as CLI devs do), you don't have to worry about this stuff because webpack bundles the dependencies for you.

0.3.2 (2017-05-02) * Bug fixes PRs #91, 95, 106

0.3.1 (2017-03-08) * Now runs in node so can use in "universal" demos. See PR #102.

0.3.0 (2017-02-27) * Support Angular version 4

0.2.4 (2017-01-02) * Remove reflect-matadata and zone.js as peerDependencies

0.2.3 (2016-12-28) * Unpin RxJs

0.2.2 (2016-12-20) * Update to Angular 2.4.0

0.2.1 (2016-12-14) * Fixed regression in handling commands, introduced in 0.2.0 * Improved README

0.2.0 (2016-12-11)

- BREAKING CHANGE: The observables returned by the `handleCollections` methods that process requests against the supplied in-mem-db collections are now "cold". That means that requests aren't processed until something subscribes to the observable … just like real-world `Http` calls.

  Previously, these request were "hot" meaning that the operation was performed immediately (e.g., an in-memory collection was updated) and *then* we returned an `Observable<Response>`. That was a mistake! Fixing that mistake *might* break your app which is why bumped the *minor* version number from 1 to 2.

  We hope *very few apps are broken by this change.* Most will have subscribed anyway. But any app that called an `http` method with fire-and-forget … and didn't subscribe … expecting the database to be updated (for example) will discover that the operation did ***not*** happen.

- BREAKING CHANGE: `createErrorResponse` now requires the `Request` object as its first parameter so it can prepare a proper error message. For example, a 404 `errorResponse.toString()` now shows the request URL.

- Commands remain "hot" — processed immediately — as they should be.

- The `HTTP GET` interceptor in example `hero-data.service` shows how to create your own "cold" observable.

- While you can still specify the `inMemDbService['responseInterceptor']` to morph the response options, the previously exported `responseInterceptor` function no longer exists as it served no useful purpose. Added the `ResponseInterceptor` *type* to remind you of the signature to implement.
- Allows objects with `id===0` (issue #56)
- The default `parseUrl` method is more flexible, thanks in part to the new `config.apiBase` property. See the ReadMe to learn more.
- Added `config.post204` and `config.put204` to control whether PUT and POST return the saved entity. It is `true` by default which means they do not return the entity (`status=204`) — the same behavior as before. (issue #74)
- `response.url` is set to `request.url` when this service itself creates the response.
- A few new methods (e.g., `emitResponse`) to assist in HTTP method interceptors.

0.1.17 (2016-12-07) * Update to Angular 2.2.0.

0.1.16 (2016-11-20) * Swap `"lib": [ "es2015", "dom" ]` in `tsconfig.json` for @types/core-jsinpackage.json‘ issue #288

0.1.15 (2016-11-14) * Update to Angular 2.2.0.

0.1.14 (2016-10-29) * Add `responseInterceptor` for issue #61

0.1.13 (2016-10-20) * Update README for 0.1.11 breaking change: npm publish as `esm` and a `umd` bundle

Going to `umd` changes your `systemjs.config` and the way you import the library.

In `systemjs.config.js` you should change the mapping to: `'angular-in-memory-web-api': 'npm:angular-in-memory-web-api/bundles/in-memory-web-api.umd.js'` then delete from `packages: 'angular-in-memory-web-api': {      main: './index.js',      defaultExtension: 'js'   }` You must ES import the in-mem module (typically in `AppModule`) like this: `import { InMemoryWebApiModule } from 'angular-in-memory-web-api';` ## 0.1.12 (2016-10-19) * exclude travis.yml and rollup.config.js from npm package

0.1.11 (2016-10-19) * BREAKING CHANGE: npm publish as `esm` and a `umd` bundle. Does not change the API but does change the way you register and import the in-mem module. Documented in later release, v.0.1.13

0.1.10 (2016-10-19) * Catch a `handleRequest` error and return as a failed server response.

0.1.9 (2016-10-18) * Restore delay option, issue #53.

0.1.7 (2016-10-12) * Angular 2.1.x support.

0.1.6 (2016-10-09) * Do not add delay to observable if delay value === 0 (issue #47) * Can override `parseUrl` method in your db service class (issue #46, #35) * README.md explains `parseUrl` override. * Exports functions helpful for custom HTTP Method Interceptors * `createErrorResponse` * `createObservableResponse` * `setStatusText` * Added `examples/hero-data.service.ts` to show overrides (issue #44)

0.1.5 (2016-10-03) * project.json license changed again to match angular.io package.json

0.1.4 (2016-10-03) * project.json license is "MIT"

0.1.3 (2016-09-29) * Fix typos

0.1.2 (2016-09-29) * AoT support from Tor PR #36 * Update npm packages * `parseId` fix from PR #33

0.1.1 (2016-09-26) * Exclude src folder and its TS files from npm package

0.1.0 (2016-09-25) * Renamed package to "angular-in-memory-web-api" * Added "passThruUnknownUrl" options * Simplified `forRoot` and made it acceptable to AoT * Support case sensitive search (PR #16)

"angular2-in-memory-web-api" versions

The last npm package named "angular2-in-memory-web-api" was v.0.0.21

0.0.21 (2016-09-25) * Add source maps (PR #14)

0.0.20 (2016-09-15) * Angular 2.0.0 * Typescript 2.0.2

0.0.19 (2016-09-13) * RC7

0.0.18 (2016-08-31) * RC6 (doesn't work with older versions)

0.0.17 (2016-08-19) * fix `forRoot` type constraint * clarify `forRoot` param

0.0.16 (2016-08-19) * No longer exports `HttpModule` * Can specify configuration options in 2nd param of `forRoot` * jsDocs for `forRoot`

0.0.15 (2016-08-09) * RC5 * Support for NgModules

0.0.14 (2016-06-30) * RC4

0.0.13 (2016-06-21) * RC3

0.0.12 (2016-06-15) * RC2

0.0.11 (2016-05-27) * add RegExp query support * find-by-id is sensitive to string ids that look like numbers

0.0.10 (2016-05-21) * added "main:index.js" to package.json * updated to typings v.1.0.4 (a breaking release) * dependencies -> peerDepen-

dencies|devDependencies * no es6-shim dependency. * use core-js as devDependency.

0.0.9 (2016-05-19) * renamed the barrel core.js -> index.js

0.0.8 (2016-05-19) * systemjs -> commonjs * replace es6-shim typings w/ core-js typings

0.0.7 (2016-05-03) * RC1 * update to 2.0.0-rc.1

0.0.6 (2016-05-03) * RC0 * update to 2.0.0-rc.0

0.0.5 (2016-05-01) * PROVISIONAL - refers to @angular packages * update to 0.0.0-5

0.0.4 (2016-04-30) * PROVISIONAL - refers to @angular packages * update to 0.0.0-3 * rxjs: "5.0.0-beta.6"

0.0.3 (2016-04-29) * PROVISIONAL - refers to @angular packages * update to 0.0.0-2

0.0.2 (2016-04-27) * PROVISIONAL - refers to @angular packages

0.0.1 (2016-04-27) * DO NOT USE. Not adapted to new package system. * Initial cut for Angular 2 repackaged * target forthcoming Angular 2 RC