SSGPlugin

August 17, 2026

SSGPlugin

Kind: Interface

Source: `src/helper/ssg/ssg.ts`

Part of: Helper

`SSGPlugin` defines lifecycle hooks for static site generation requests, responses, and generated output. It groups hooks that run before a request, after a response, and after generation completes.

Properties

Property	Type
`beforeRequestHook`	`BeforeRequestHook
`afterResponseHook`	`AfterResponseHook
`afterGenerateHook`	`AfterGenerateHook

Diagram

```
graph LR
  Plugin[SSGPlugin]
  Before[beforeRequestHook]
  AfterResponse[afterResponseHook]
  AfterGenerate[afterGenerateHook]

  Plugin --> Before
  Plugin --> AfterResponse
  Plugin --> AfterGenerate

  Before --> Request[Request processing]
  Request --> Response[Response processing]
  Response --> Generated[Static output generation]
```

```
  AfterResponse --> Response
  AfterGenerate --> Generated
```

Usage

```ts
import type { SSGPlugin } from "./helper/ssg/ssg";
import { requestLogger } from "./hooks/request-logger";
import { responseCache } from "./hooks/response-cache";
import { outputReporter } from "./hooks/output-reporter";

const plugin: SSGPlugin = {
  beforeRequestHook: [requestLogger],
  afterResponseHook: responseCache,
  afterGenerateHook: [outputReporter],
};

export default plugin;
```

AI Coding Instructions

- Assign a single hook or an array of hooks to each `SSGPlugin` field.
- Keep `beforeRequestHook` focused on work that must happen before request processing begins.
- Use `afterResponseHook` for response inspection, mutation, caching, or logging after a response is available.
- Use `afterGenerateHook` for work that depends on generated static output.
- Preserve hook ordering when arrays are used, since hooks run as part of the generation lifecycle.

How it works

`SSGPlugin` is a TypeScript interface for attaching lifecycle hooks to `toSSG` static-site generation. Each hook property is optional and may be one hook or an ordered array of hooks: `beforeRequestHook`, `afterResponseHook`, and `afterGenerateHook`. [src/helper/ssg/ssg.ts:175-179]

- `beforeRequestHook` receives a `Request` and may return a replacement `Request`, `false`, or a promise of either. [src/helper/ssg/ssg.ts:110] During generation, it runs before the request used to collect SSG route parameters; returning `false` skips that route's parameter-collection work. [src/helper/ssg/ssg.ts:234-243]

- `afterResponseHook` receives a generated `Response` and may return a replacement `Response`, `false`, or a promise of either. [src/helper/ssg/ssg.ts:111] It runs after the route-content request; returning `false` omits that response from output. [src/helper/ssg/ssg.ts:264-281]
- `afterGenerateHook` receives the `ToSSGResult`, the file-system module, and optionally the generation options; it may be synchronous or asynchronous. [src/helper/ssg/ssg.ts:112-116] It runs after generation has produced either a success or failure result. [src/helper/ssg/ssg.ts:460-469]

When a hook property is an array, its hooks run sequentially and are awaited. For request and response hooks, each hook receives the current request or response; a returned replacement becomes the input to the next hook, and `false` stops the chain. [src/helper/ssg/ssg.ts:124-136] [src/helper/ssg/ssg.ts:145-157] After-generation hook arrays also run sequentially and are awaited. [src/helper/ssg/ssg.ts:168-172]

`toSSG` collects the deprecated hook options first, then appends hooks from `options.plugins` in plugin-array order. [src/helper/ssg/ssg.ts:376-419] If `options.plugins` is absent, it uses `[defaultPlugin()]`; that default plugin rejects every response whose status is not `200`, so those responses are not written. [src/helper/ssg/ssg.ts:372] [src/helper/ssg/plugins.ts:11-19] Passing an empty `plugins` array does not select the default plugin because the empty array is used as-is. [src/helper/ssg/ssg.ts:372]

A response accepted by the hook chain is read as text when its `Content-Type` contains `text` or `json`; otherwise it is read as an `ArrayBuffer`. [src/helper/ssg/ssg.ts:74-87] The resulting content is then passed to file writing, which derives an output path from the route path and MIME type, creates its parent directory when first encountered, and writes strings or `ArrayBuffer` content. [src/helper/ssg/ssg.ts:310-334]

There is no explicit runtime validation that plugin hook values are callable; `toSSG` only checks whether each hook property is present and whether it is an array before adding it to a hook list. [src/helper/ssg/ssg.ts:397-418] Errors in route collection or file writing are caught and returned as `{ success: false, files: [], error }`. [src/helper/ssg/ssg.ts:420-464] After-generation hooks run outside that `try` / `catch`, so an exception from one rejects the `toSSG` call rather than being added to `ToSSGResult.error`. [src/helper/ssg/ssg.ts:420-469]