PlaygroundProxyService

September 4, 2026

PlaygroundProxyService

Kind: Service

Source: `atloria-monorepo/apps/api/src/technical-docs/playground-proxy.service.ts`

Server-side try-it proxy (Tier 3.4), extracted from the technical-docs MCP controller so the
PUBLIC docs site can share the exact same execution path. Lets the playground call APIs that
block CORS. Two callers:

- owner playground (technical-docs/:projectId/playground/proxy): open target, SSRF-guarded.
- public playground (public/p/:slugWithId/playground/proxy): additionally PINNED via `allowedHosts` to the hosts the project itself declares (spec servers + Live-API base),
  so an anonymous visitor can't use the proxy as a generic fetch service.

`PlaygroundProxyService` executes server-side API requests for the technical documentation playground, allowing browser-based users to call APIs that would otherwise be blocked by CORS. It is shared by owner and public playground routes, applying SSRF protections for all requests and additional `allowedHosts` pinning for anonymous public requests.

Methods

Method	Signature	Returns
`proxyRequest`	`proxyRequest(target: PlaygroundProxyTarget, opts: { allowedHosts?: string[] })`	`Promise<PlaygroundProxyResult>`

Where it refuses work

- `PlaygroundProxyService` stops the work with an early return when `bytes > MAX_PROXY_UPLOAD_BYTES`, in 2 places.
- `PlaygroundProxyService` stops the work with an early return when `!allowed.has(host)`.
- `PlaygroundProxyService` stops the work with an early return when `privateHost`.
- `PlaygroundProxyService` stops the work with an early return when `resolved.some((a) => isPrivateIp(a.address))`.
- `PlaygroundProxyService` stops the work with an early return when `'error' in built`.
- `PlaygroundProxyService` stops the work with an early return when `bytes.byteLength > MAX_PROXY_UPLOAD_BYTES`.

When something fails

- `PlaygroundProxyService` handles failure in 4 places: it turns it into a return value in 3, and discards it silently in 1. A failure discarded silently leaves no trace for whoever debugs this later.

Diagram

```
sequenceDiagram
    participant Browser as Playground Browser
    participant Controller as Playground Controller
    participant Proxy as PlaygroundProxyService
    participant Guard as SSRF / Host Validation
    participant API as Target API

    Browser->>Controller: Submit playground request
    Controller->>Proxy: proxyRequest(request, options)
    Proxy->>Guard: Validate target URL

    alt Owner playground
        Guard-->>Proxy: SSRF-safe target accepted
    else Public playground
        Guard->>Guard: Verify host is in allowedHosts
        Guard-->>Proxy: SSRF-safe pinned host accepted
    end

    Proxy->>API: Execute server-side HTTP request
    API-->>Proxy: HTTP response
    Proxy-->>Controller: PlaygroundProxyResult
    Controller-->>Browser: Return response for display
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PlaygroundProxyService } from './playground-proxy.service';

@Injectable()
export class PlaygroundExecutionService {
  constructor(
    private readonly playgroundProxyService: PlaygroundProxyService,
  ) {}

  async executePublicRequest() {
    return this.playgroundProxyService.proxyRequest({
      url: 'https://api.example.com/v1/users',
      method: 'GET',
      headers: {
        accept: 'application/json',
      },
      allowedHosts: ['api.example.com'],
    });
  }
}
```

AI Coding Instructions

- Route both owner and public playground execution through `proxyRequest()` so they share identical request handling and response formatting.
- Always preserve SSRF validation when adding request options, redirects, URL parsing, or new HTTP methods.
- For public playground callers, derive `allowedHosts` only from project-controlled sources such as OpenAPI spec servers and the configured Live API base URL.
- Do not accept arbitrary client-provided host allowlists; doing so would turn the public endpoint into a generic anonymous proxy.
- Keep controller code focused on authentication, project lookup, and caller context; place outbound request and validation behavior in this service.

Referenced By

- `PublicProjectController` (DEPENDS_ON)
- `TechnicalDocsMcpController` (DEPENDS_ON)
- `TechnicalDocsModule` (MODULE_PROVIDES)
- `TechnicalDocsModule` (MODULE_EXPORTS)