SearchController

September 4, 2026

SearchController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/documentation/search.controller.ts`

Search Controller

Provides search endpoints for documentation using Azure AI Search.
Includes:

- Full-text search with filtering
- Autocomplete suggestions
- Faceted search results
- Organization + project isolation

Graceful Degradation: When AZURE_SEARCH_ENABLED=false, returns disabled status.

`SearchController` exposes HTTP endpoints for searching the documentation corpus via Azure AI Search. It handles full-text queries with filters, autocomplete suggestions, and faceted results, while enforcing organization/project isolation boundaries. When `AZURE_SEARCH_ENABLED=false`, it gracefully degrades by returning a disabled status instead of attempting search operations.

Diagram

```
graph LR
  Client[API Client] -->|HTTP| SC[SearchController]
  SC -->|checks| Flag[AZURE_SEARCH_ENABLED]
  Flag -->|false| Disabled[Return: search disabled status]
  Flag -->|true| Azure[Azure AI Search]
  SC -->|Full-text search + filters| Azure
  SC -->|Autocomplete suggestions| Azure
  SC -->|Facets| Azure
  SC -->|enforce isolation| Scope[Org + Project Scope]
```

```
  Scope --> SC
  Azure --> Results[Search / Suggest / Facet Results]
  Results --> Client
```

Usage

```
// Example client-side usage calling the SearchController endpoints.
// (Endpoint paths may vary based on your NestJS routing configuration.)

type SearchResponse = {
  enabled: boolean;
  results?: Array<{ id: string; title: string; snippet?: string; url?: string }>;
  facets?: Record<string, Array<{ value: string; count: number }>>;
  message?: string;
};

type SuggestResponse = {
  enabled: boolean;
  suggestions?: string[];
  message?: string;
};

const API_BASE = process.env.API_BASE ?? "http://localhost:3000";

async function searchDocs(params: {
  orgId: string;
  projectId: string;
  q: string;
  filters?: Record<string, string | string[]>;
  facets?: string[];
  top?: number;
}): Promise<SearchResponse> {
  const url = new URL(`${API_BASE}/documentation/search`);
  url.searchParams.set("orgId", params.orgId);
  url.searchParams.set("projectId", params.projectId);
  url.searchParams.set("q", params.q);
  if (params.top) url.searchParams.set("top", String(params.top));
  if (params.facets?.length) url.searchParams.set("facets", params.facets.join(","));
  if (params.filters) url.searchParams.set("filters", JSON.stringify(params.filters));

  const res = await fetch(url.toString(), { method: "GET" });
  return res.json();
}

async function suggestDocs(params: {
  orgId: string;
  projectId: string;
  prefix: string;
  top?: number;
}): Promise<SuggestResponse> {
```

```
  const url = new URL(`${API_BASE}/documentation/search/suggest`);
  url.searchParams.set("orgId", params.orgId);
  url.searchParams.set("projectId", params.projectId);
  url.searchParams.set("prefix", params.prefix);
  if (params.top) url.searchParams.set("top", String(params.top));

  const res = await fetch(url.toString(), { method: "GET" });
  return res.json();
}

(async () => {
  const search = await searchDocs({
    orgId: "org_123",
    projectId: "proj_abc",
    q: "authentication",
    filters: { type: ["guide", "api"], visibility: "public" },
    facets: ["type", "tags"],
    top: 10,
  });

  if (!search.enabled) {
    console.warn("Search is disabled:", search.message);
    return;
  }

  console.log("Results:", search.results);
  console.log("Facets:", search.facets);

  const suggest = await suggestDocs({
    orgId: "org_123",
    projectId: "proj_abc",
    prefix: "auth",
    top: 5,
  });

  console.log("Suggestions:", suggest.suggestions);
})();
```

AI Coding Instructions

- Preserve **graceful degradation**: always gate Azure calls behind `AZURE_SEARCH_ENABLED`, and return a consistent “disabled” payload when off.
- Enforce **organization + project isolation** on every endpoint (never allow cross-scope querying, even if the client passes broader filters).
- Keep request parameters **validated and normalized** (e.g., parsing JSON `filters`, limiting `top`, sanitizing facet fields) to prevent malformed queries and expensive searches.

- Maintain a stable **response shape** across search/suggest/facet endpoints so clients can handle “enabled vs disabled” without special-casing.
- When adding new filters/facets, update both the controller contract and the Azure index/query mapping together to avoid silent mismatches.

Relationships

- MODULE_DECLARES → `search`
- MODULE_DECLARES → `getSuggestions`
- MODULE_DECLARES → `getHealth`
- DEPENDS_ON → `AzureSearchService`
- DEPENDS_ON → `configservice`

Referenced By

- `DocumentationModule` (MODULE_DECLARES)