PublicCaptureController

September 4, 2026

PublicCaptureController

Kind: Controller

Source: `atloria-monorepo/apps/api/src/capture/public-capture.controller.ts`

PUBLIC capture endpoint — consumed by the tour player embedded on the customer's site. Returns the full recipe (selectors + step titles) so the player can highlight the customer's REAL elements. Secret input values are re-redacted here; nothing fabricated is ever served.

`PublicCaptureController` exposes a **public** capture endpoint consumed by the embedded tour player running on a customer’s site. It returns the full tour “recipe” (selectors plus step titles) so the player can match and highlight the customer’s real DOM elements. Before responding, it **re-redacts secret input values** to ensure no sensitive data is ever served.

Diagram

```
graph LR
  A[Tour Player (embedded on customer site)] -->|HTTP request| B[PublicCaptureController]
  B --> C[Capture/Recipe Service Layer]
  C --> D[(Persistence: capture/recipe data)]
  C --> E[Redaction/Re-redaction]
  E -->|sanitized recipe| B
  B -->|JSON: selectors + step titles (sanitized)| A
```

Usage

```
// Example: the embedded tour player fetching the public capture recipe
// (run in the browser on the customer's site)

type PublicCaptureStep = {
  title: string;
  selector: string;
```

```
};

type PublicCaptureRecipeResponse = {
  captureId: string;
  steps: PublicCaptureStep[];
};

async function fetchPublicCaptureRecipe(captureId: string) {
  const res = await fetch(`https://api.example.com/public/capture/${captureId}`, {
    method: "GET",
    headers: {
      "Accept": "application/json",
    },
    credentials: "omit", // typically public/anonymous
  });

  if (!res.ok) {
    throw new Error(`Failed to load recipe (${res.status})`);
  }

  const recipe = (await res.json()) as PublicCaptureRecipeResponse;

  // Use selectors to find and highlight real elements on the customer page
  for (const step of recipe.steps) {
    const el = document.querySelector(step.selector);
    if (el) {
      el.setAttribute("data-tour-highlight", "true");
      // render step.title in your UI
    }
  }

  return recipe;
}

// Usage
fetchPublicCaptureRecipe("cap_12345").catch(console.error);
```

AI Coding Instructions

- Preserve the contract: the public endpoint must only return what the tour player needs (selectors + step titles + minimal identifiers); avoid adding internal fields.
- Always apply **re-redaction** before responding; never leak original captured input values, even if already stored redacted.
- Keep this controller thin: delegate data fetching/assembly to a service and keep response shaping/sanitization explicit and testable.
- Validate public inputs carefully (e.g., `captureId` format) and return consistent HTTP errors; assume callers are anonymous/untrusted.

- When changing selector/step structures, update the embedded player expectations and version/compatibility behavior together.

Relationships

- MODULE_DECLARES → `getFlow`
- DEPENDS_ON → `CaptureService`

Referenced By

- `CaptureModule` (MODULE_DECLARES)