PmeService

September 5, 2026

PmeService

Kind: Service

Source: `atloria-monorepo/apps/api/src/pme/pme.service.ts`

`PmeService` manages the PME workflow for project-scoped jobs, including job creation and retrieval, variant discovery and selection, strategy execution, and page finalization. It serves as the backend orchestration layer used by controllers or other application services to move a PME job through its lifecycle.

Methods

Method	Signature	Returns
`createJob`	`createJob(dto: CreatePmeJobDto, userId: string)`	`unknown`
`getJob`	`getJob(jobId: string)`	`unknown`
`getJobsByProject`	`getJobsByProject(projectId: string)`	`unknown`
`getVariants`	`getVariants(jobId: string)`	`unknown`
`selectVariant`	`selectVariant(variantId: string)`	`unknown`
`startStrategy`	`startStrategy(dto: CreatePmeJobDto, userId: string)`	`unknown`
`finalizePage`	`finalizePage(dto: FinalizePmePageDto)`	`unknown`

Dependencies

- `PrismaService`
- `Queue`

Where it refuses work

- `PmeService` stops the work with `NotFoundException` when `!job`, in 2 places.

- `PmeService` stops the work with `NotFoundException` when `!variant`.
- `PmeService` stops the work with `NotFoundException` when `!page`.

Diagram

```
sequenceDiagram
  participant Client
  participant Controller as PME Controller
  participant Service as PmeService
  participant Store as Persistence Layer
  participant Strategy as Strategy Engine

  Client->>Controller: Create PME job
  Controller->>Service: createJob(...)
  Service->>Store: Persist job
  Store-->>Service: Job
  Service-->>Controller: Job

  Client->>Controller: Get/select variants
  Controller->>Service: getVariants(jobId)
  Service->>Store: Load variants
  Store-->>Service: Variants
  Service-->>Controller: Variants

  Client->>Controller: Select variant and start strategy
  Controller->>Service: selectVariant(...)
  Service->>Store: Save selection
  Controller->>Service: startStrategy(...)
  Service->>Strategy: Execute selected strategy
  Strategy-->>Service: Generated page/result

  Client->>Controller: Finalize page
  Controller->>Service: finalizePage(...)
  Service->>Store: Persist finalized state
  Service-->>Controller: Finalized job/page
```

Usage

```
import { Injectable } from '@nestjs/common';
import { PmeService } from './pme.service';

@Injectable()
export class PmeWorkflowService {
  constructor(private readonly pmeService: PmeService) {}

  async runWorkflow(projectId: string) {
    // Use the DTOs and argument shapes defined by the PME module.
    const job = await this.pmeService.createJob(/* createJob input */);
```

```
    const variants = await this.pmeService.getVariants(/* job identifier */);

    await this.pmeService.selectVariant(
      /* job identifier */,
      /* selected variant identifier */,
    );

    await this.pmeService.startStrategy(/* job identifier */);

    return this.pmeService.finalizePage(/* job identifier */);
  }

  async listProjectJobs(projectId: string) {
    return this.pmeService.getJobsByProject(projectId);
  }
}
```

AI Coding Instructions

- Keep PME lifecycle transitions inside `PmeService`; controllers should validate and delegate rather than reproduce workflow logic.
- Use `getJobsByProject()` for project-scoped access and ensure callers enforce the appropriate project authorization boundary.
- Retrieve available variants with `getVariants()` before calling `selectVariant()`; do not assume a variant identifier is valid for every job.
- Start strategy execution only after a job and its selected variant are in a valid state, then finalize the resulting page through `finalizePage()`.
- Preserve existing method argument and return contracts when extending the service, especially where controllers, persistence, or strategy integrations depend on them.

Relationships

- DEPENDS_ON → `PrismaService`
- DEPENDS_ON → `queue`

Referenced By

- `PmeController` (DEPENDS_ON)
- `PmeModule` (MODULE_PROVIDES)
- `PmeModule` (MODULE_EXPORTS)