Introduction

How It Works

Quickie operates by injecting essential middleware and services into your application pipeline. It provides base classes that you can implement to create APIs and backend services with minimal effort and focus solely on your logic. With Quickie, all you need to do is define your required classes and you're ready to go — no endless boilerplate, no hassle. Quickie is perfect for building tracer bullet APIs, prototypes, or any sorta proof-of-concept projects. Whether you're experimenting with ideas or rapidly iterating, build APIs or backend service real quick.

Here's a high-level overview of its workflow:

1. **Service Registration**: Quickie sets up services for rate limiting, idempotency, and other configurable options through `QuickieConfig()` and complete the setup with `AddQuickie()` which resolve all dependencies.
2. **Idempotency made simple**: Ensure API calls yield consistent results when retried, with support for custom idempotency providers.
3. **Built-In Rate Limiting**: Protect your APIs from overuse with an easy-to-setup rate-limiting mechanism.
4. **Customizable Options**: You can fine-tune configurations, such as permitting limits for rate limiting or choosing a custom idempotency provider (Redis, or any external databases).
5. **Custom Error Messages**: Fine-tune responses with the option to show user-friendly error messages or show exact exception message.

Namespace Quickie.Apis.Crud

Classes

CrudController<TViewModel, TRequestHandler, TIdType>

Base class providing CRUD apis.

CrudForCollectionController<TViewModel, TRequestHandler, TIdType>

Base class providing CRUD apis in collection.