Abstract

August 17, 2026

Abstract

Kind: Interface

Source: `packages/common/interfaces/abstract.interface.ts`

Part of: Common

`Abstract<T>` is a generic interface that describes a constructor-like type with a `prototype` property of type `T`. It is useful when APIs need to accept or reference a class definition while preserving the instance type represented by that class.

Properties

Property	Type
`prototype`	`T`

Diagram

```
graph LR
  A[Abstract<T>] --> B[prototype: T]
  C[Class Definition] --> A
  B --> D[Instance Shape T]
```

Usage

```
interface Abstract<T> {
  prototype: T;
}

class UserService {
  findUser(id: string): string {
    return `User ${id}`;
  }
}
```

```ts
function getPrototype<T>(type: Abstract<T>): T {
  return type.prototype;
}

const userServicePrototype = getPrototype(UserService);

userServicePrototype.findUser("123");
```

AI Coding Instructions

- Use `Abstract<T>` when a function or API needs to receive a class-like value and access its instance prototype.
- Keep the generic `T` aligned with the instance type represented by the class or constructor.
- Do not assume `Abstract<T>` guarantees that the value is constructable; it only defines the `prototype` property.
- Prefer constructor interfaces with `new (...args) => T` when callers must instantiate the provided type.
- Ensure classes passed to APIs using `Abstract<T>` expose a prototype compatible with the expected generic type.

Used by

3 references from 3 files. Each is a place in this repository where the symbol is actually used — go read one rather than trusting an example.

Imported by (3)

- `InvalidClassScopeException` — `packages/core/errors/exceptions/invalid-class-scope.exception.ts`:6
- `AbstractInstanceResolver` — `packages/core/injector/abstract-instance-resolver.ts`:12
- `NestApplicationContext` — `packages/core/nest-application-context.ts`:40