InstancePerContext

August 17, 2026

InstancePerContext

Kind: Interface

Source: `packages/core/injector/instance-wrapper.ts`

Part of: Core

`InstancePerContext` tracks the lifecycle state of a provider instance within a specific dependency-injection context. It stores the resolved instance alongside flags and promises used to coordinate asynchronous construction, prevent duplicate instantiation, and determine whether constructor logic has already run.

Properties

Property	Type
`instance`	`T`
`isResolved`	`boolean`
`isPending`	`boolean`
`donePromise`	`Promise<unknown>`
`isConstructorCalled`	`boolean`

Diagram

```
graph LR
  Context[Injection Context] --> Record[InstancePerContext]
  Record --> Instance[instance: T]
  Record --> Resolved[isResolved]
  Record --> Pending[isPending]
  Record --> Promise[donePromise]
  Record --> Constructed[isConstructorCalled]

  Pending -->|await completion| Promise
```

```
  Promise -->|resolution complete| Resolved
  Constructed -->|prevents duplicate constructor calls| Instance
```

Usage

```
interface InstancePerContext<T> {
  instance: T;
  isResolved: boolean;
  isPending: boolean;
  donePromise: Promise<unknown>;
  isConstructorCalled: boolean;
}

class RequestService {
  constructor(public readonly requestId: string) {}
}

const contextInstance: InstancePerContext<RequestService> = {
  instance: new RequestService('request-123'),
  isResolved: true,
  isPending: false,
  donePromise: Promise.resolve(),
  isConstructorCalled: true,
};

// A resolver can reuse the instance when it has already been created.
if (contextInstance.isResolved) {
  console.log(contextInstance.instance.requestId);
}
```

AI Coding Instructions

- Treat `instance` as scoped to a single injection context; do not reuse it across unrelated contexts.
- Set `isPending` before starting asynchronous resolution, and expose the completion work through `donePromise`.
- Check `isResolved` and await `donePromise` when necessary to avoid creating duplicate instances during concurrent resolution.
- Use `isConstructorCalled` to prevent constructor or initialization logic from running more than once for the same context record.
- Keep lifecycle flags synchronized with promise completion so consumers never observe a resolved state before the instance is ready.