

Containers

Containers

Anonymous · 20/10/2026

Containers

All the containers implement `Derafu\Container\Contract\ContainerInterface`, which has the methods to read and write by key (with dot notation), plus the [typed getters](#):

```
interface ContainerInterface extends ArrayAccess, JsonSerializable
{
    public function all(): array;
    public function set(string $key, mixed $value): static;
    public function get(string $key, mixed $default = null): mixed;
    public function has(string $key): bool;
    public function clear(?string $key = null): void;
    public function collection(): ArrayCollection;
    public function matching(Criteria $criteria): ArrayCollection;

    // And the typed getters: getString(), requireString(), getInt()...
}
```

The `ArrayAccess` offsets use the same keys: `$container['user.name']` is the same as `$container->get('user.name')`.

Bag

A flexible container without restrictions. It is the one to use when you only need a place to keep data.

```
use Derafu\Container\Bag;

$bag = new Bag();
$bag->set('user.name', 'John');
$bag->set('user.email', 'john@example.com');

echo $bag->get('user.name'); // "John"
```

It adds three methods:

```
$bag->merge(['user' => ['age' => 30]]); // Recursive: keeps user.name and user.email.
```

```
$bag->replace(['other' => 1]);           // Replaces everything.
$bag->remove('other');                   // Removes a key.
```

Vault

A container with a simple schema. Each key of the schema says its types, whether it is required, its accepted values, its default and how to normalize it. The validation is made by `symfony/options-resolver`.

```
use Derafu\Container\Vault;

$vault = new Vault(
    ['user' => ['name' => 'John']],
    [
        'user' => [
            'types' => 'array',
            'required' => true,
            'schema' => [
                'name' => ['types' => 'string', 'required' => true],
                'age' => ['types' => 'int', 'default' => 18],
            ],
        ],
    ],
);

$vault->get('user.age'); // 18, the default.
```

The constructor takes the data, the schema and whether the keys that the schema does not define are allowed (`false` by default). See [Validation](#) for every key of the schema and for what happens on each change.

Store

A container validated with [JSON Schema](#) (draft 2020-12) through `opis/json-schema`. Use it when the schema comes from outside, or when you need what JSON Schema has and the simple schema of `Vault` does not (formats, `items`, `oneOf`, patterns...).

```
use Derafu\Container\Store;

$store = new Store(
    ['user' => ['name' => 'John', 'email' => 'john@example.com']],
    [
        'properties' => [
            'user' => [
                'type' => 'object',
            ],
        ],
    ],
);
```

```

        'required' => ['name', 'email'],
        'properties' => [
            'name' => ['type' => 'string'],
            'email' => ['type' => 'string', 'format' => 'email'],
        ],
    ],
],
];
);

```

`$schema` and `type: object` are added to the schema when you do not give them, so `getSchema()` returns them. The `default` of a property is applied when the property has no value.

Journal

A sequential container: you add items at the end and read them newest first.

```

use Derafu\Container\Journal;

$journal = new Journal();
$journal->add(['level' => 'info', 'message' => 'System started']);
$journal->add(['level' => 'error', 'message' => 'Connection failed']);

$errors = array_filter(
    $journal->reverse(), // Newest first.
    fn (array $entry) => $entry['level'] === 'error'
);

```

Reading the Underlying Collection

`collection()` returns the `ArrayCollection` that holds the data, and `matching()` filters it with a Doctrine `Criteria`:

```

use Derafu\Container\Bag;
use Doctrine\Common\Collections\Criteria;

$users = new Bag([['name' => 'Ana', 'age' => 20], ['name' => 'Bo', 'age' => 10]]);

$adults = $users->matching(
    Criteria::create()->where(Criteria::expr()->gte('age', 18))
);

```

`collection()` is the live collection, not a copy. Changing it directly does not go through the validation of `Vault` and `Store`; see [Validation](#).

Last updated on 20/10/2026