Custom Parser

Creating Custom Rule Parser

Anonymous · 09/09/2026

Creating Custom Rule Parser

You can create your own rule parser to customize how rules are parsed from strings or arrays:

```
use Derafu\DataProcessor\Contract\RuleParserInterface;

final class CustomRuleParser implements RuleParserInterface
{
    public function parse(string|array $rules): array
    {
        if (is_array($rules)) {
            return $this->parseArray($rules);
        }
        return $this->parseString($rules);
    }

    private function parseArray(array $rules): array
    {
        // Your array parsing logic here.
        $parsed = [];
        foreach ($rules as $type => $typeRules) {
            // Handle different formats for rules.
            $parsed[$type] = $this->parseTypeRules($typeRules);
        }
        return $parsed;
    }

    private function parseString(string $rules): array
    {
        // Your string parsing logic here.
        // Example: parse rules like "t(rule1|rule2) s(rule3) required|email"
        return [
            'transform' => ['rule1', 'rule2'],
            'sanitize' => ['rule3'],
            'validate' => ['required', 'email'],
        ];
    }

    private function parseTypeRules(string|array $rules): array|string
    {
```

```
        // Handle parsing of individual rule types.
        if (is_string($rules)) {
            return explode('|', $rules);
        }
        return $rules;
    }
}

// Usage with processor factory.
$processor = ProcessorFactory::create(
    parser: new CustomRuleParser()
);

// Or manual instantiation.
$registry = new RuleRegistry(); // Then register the rules you need.
$resolver = new RuleResolver($registry);
$parser = new CustomRuleParser();
$processor = new Processor($resolver, $parser);

// Using your custom parser.
$result = $processor->process('test@example.com', 't(lowercase) s(trim)
required|email');

// Or.
$result = $processor->process('test@example.com', [
    'transform' => 'lowercase',
    'sanitize' => 'trim',
    'validate' => 'required|email',
]);
```

This allows you to:

- Create custom rule parsing formats.
- Support different string formats for rules.
- Add custom shorthand notations.
- Implement domain-specific rule syntax.
- Handle complex rule configurations.

Last updated on 09/09/2026