

Service Configuration

Form rendering services, isolated Twig and tagged contributions

Anonymous · 30/10/2026

Service Configuration

Import the production configurations and provide your application's translator. The translator receives the catalogues contributed by the imported packages:

```
imports:
  - { resource: '../vendor/derafu/data-processor/resources/config/data-processor-services.yaml' }
  - { resource: '../vendor/derafu/form/resources/config/form-services.yaml' }

services:
  Symfony\Contracts\Translation\TranslatorInterface:
    class: Symfony\Component\Translation\Translator
    factory: ['Derafu\Translation\TranslatorFactory', 'create']
    arguments:
      $defaultLocale: en
      $fallbackLocales: [en]
      $resourceProviders: !tagged_iterator derafu_translation.resource_provider
```

Inject `FormFactoryInterface`, `FormRendererInterface` and `FormDataProcessorInterface` into your application services. The factory creates forms from definitions; the renderer and processor share the configured types, rule evaluator and optional CSRF/captcha integrations.

Isolated Rendering

Form rendering owns qualified services such as `Derafu\Twig\Contract\TwigServiceInterface` `$form` and `Derafu\Renderer\Contract\RendererInterface` `$form`. Importing Form does not replace the application's Twig or Renderer services. The internal `FormTwigExtension` is loaded only in the form environment. Symfony bundles can register a separate prefixed extension in the native page environment.

The package's template directory is contributed automatically. Scalar Twig options belong to `derafu.form.twig.options`, for example `{ extra: false }`.

`derafu.form.twig.component_registrar.cache_key` identifies the form-specific component configuration when a PSR cache pool is provided.

Contributions

Each domain supplies only its own providers. The following tags belong to the form environment:

Tag	Provider contract	Contribution
<code>derafu_form.element_renderer_provider</code>	<code>ElementRendererProviderInterface</code>	Element type to renderer map
<code>derafu_form.widget_renderer_provider</code>	<code>WidgetRendererProviderInterface</code>	Widget type to renderer map
<code>derafu_form.twig.template_path_provider</code>	<code>Derafu\Twig\Contract\TemplatePathProviderInterface</code>	Template directories
<code>derafu_form.twig.extension_provider</code>	<code>Derafu\Twig\Contract\ExtensionProviderInterface</code>	Twig extensions
<code>derafu_form.twig.component_provider</code>	<code>Derafu\Twig\Contract\ComponentProviderInterface</code>	Twig component classes
<code>derafu_form.formatter_handler_provider</code>	<code>Derafu\Renderer\Contract\FormatterHandlerProviderInterface</code>	Named formatter handlers
<code>derafu_form.renderer.engine</code>	<code>Derafu\Renderer\Contract\EngineInterface</code>	Rendering engines

Individual component classes can also use `derafu_form.twig.component`. Providers are processed by descending tag priority. The first contribution for an element/widget type wins. Default element/widget providers and the package's template-path provider have priority -1000, so ordinary consumer providers can override their own entries without copying the defaults. Twig retains its first extension of each class; template directories are searched in priority order. Formatter providers use the first handler of each name.

```
services:
    App\Form\TemplatePathProvider:
        tags: ['derafu_form.twig.template_path_provider']

    App\Form\WidgetRendererProvider:
        tags:
            - { name: derafu_form.widget_renderer_provider, priority: 100 }
```

Registries remain injectable through `ElementRendererRegistryInterface` and `WidgetRendererRegistryInterface`. Rendering extensions are resolved when Twig creates its environment, allowing the internal form functions to call the same form renderer without eager construction cycles.

Last updated on 30/10/2026