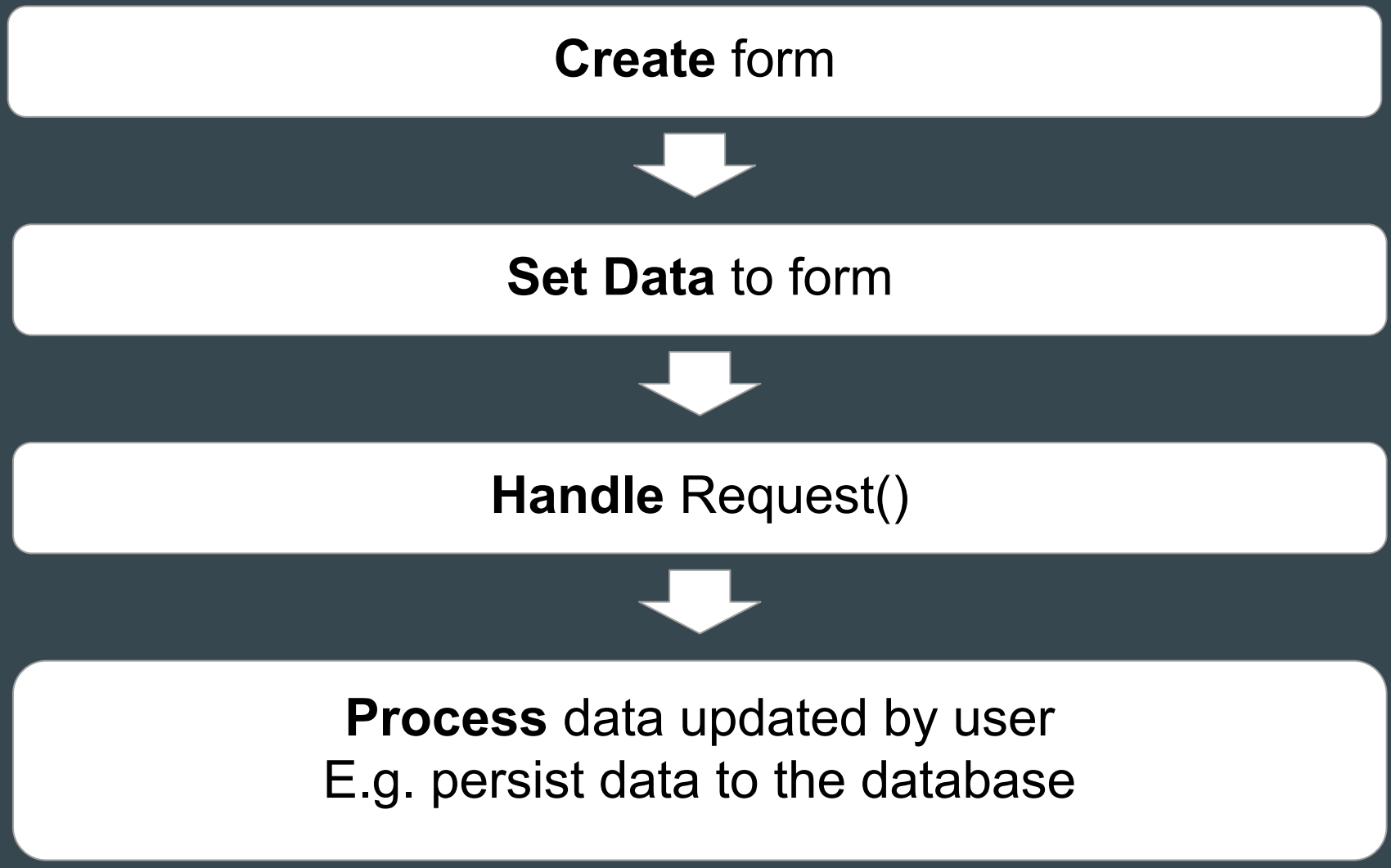
Forms & Validation

Form Flow


Create form
Set Data to form
Handle Request()
Process data updated by user
E.g. persist data to the database

How to Create a form?

Using **FormFactory** | Using **ControllerTrait**

With **FormType**

With **FormBuilder**

How to set form data

1. During Form creation - ***preferred option***
2. After form creation with $form->setData()
    - *not recommended*
    - Form events will be triggered twice

Form options (configuration)

Form options is a form **configuration.** And it's a **data** too. Data that needed for building and submitting the form, but it's not the Form Data. Usually don't need to be updated from user input.

Form options (configuration)

- Form Data is an option too. You can just set $option[‘data’], and form factory internally uses it too.
- $options[‘data’] has higher priority than $data (second param)

How to create a form, that depends on external data?

- Don’t use dependency injections (inject data to formType __constructor)
- Because it will change FormType, not the FomConfig, so all forms on page will be configured the same.

How to create a form, that depends on external data?

- Inject to container services (e.g. Doctrine), not concrete Entities

When data not available from service

- Use custom form options,
- But first you must configure them in FormType::configureOptions() method.

Where to place dynamic logic, if I can’t place it in FormType?

- In Event Listeners

Form Events

- PRE_SET_DATA
- POST_SET_DATA

- PRE_SUBMIT
- SUBMIT
- POST_SUBMIT

How to subscribe to form events?

- Create
  - EventListener - any callable
  - EventSubscriber - separate class implemented EventSubscriberInterface
- Add them to FormBuilder

You can even change the form in EventListeners

- But in Symfony you can’t change or replace form fields.

In OroPlatform we have FormUtils that solve this problem

But you can’t register event listener like in Symfony or Doctrine EventDispatcher, from another bundle

How to solve this?

FormTypeExtension

- Like FormType but have getExtendedType() method
- Can be applied to multiple FormTypes

How form works with data internally?

- Controller - timestamp
- Form - DateTime
- Rendered HTML - formatted string

How form works with data internally?

- Model Data - timestamp
- Normalized Data - DateTime
- View Data - formatted string

Data Transformers

- Model Transformer
- View Transformer

2 methods:

- transform($data)
- reverseTransform($data)

Data Transformers

Should only transform data from one format to another, but never changed the data.

If you want to change data - use EventListeners

Form Theming

```
{% form_theme form _self %}
{% form_theme form 'form/fields.html.twig' %}

{% form_theme form with ['common.html.twig', 'form/fields.html.twig'] %}
{% form_theme form with ['common.html.twig', 'form/fields.html.twig'] only %}

{% form_theme form.a_child_form 'form/fields.html.twig' %}


{% use 'form_div_layout.html.twig' with integer_widget as base_integer_widget %}
```

```
# app/config/config.yml
twig:
    form_themes:
        - 'form/fields.html.twig'
    # ...
```

Form Theming

```
{% block form_row %}
    <div class="form_row">
        {{ form_label(form) }}
        {{ form_errors(form) }}
        {{ form_widget(form) }}
    </div>
{% endblock form_row %}

{% block form_errors %}
        {% if errors|length > 0 %}
        <ul>
            {% for error in errors %}
                <li>{{ error.message }}</li>
            {% endfor %}
        </ul>
        {% endif %}
{% endblock form_errors %}
```

```
{% block form_label %}
    {{ block('base_form_label') }}

    {% if label is not same as(false) and required %}
        <span class="required">*</span>
    {% endif %}
{% endblock %}

{% block integer_widget %}
    {% set type = type|default('number') %}
    {{ block('form_widget_simple') }}
{% endblock integer_widget %}

{% block form_widget_simple %}
    {% set type = type|default('text') %}
    <input type="{{ type }}" {{ block('widget_attributes') }}
{% if value is not empty %}value="{{ value }}" {% endif %}/>
{% endblock form_widget_simple %}
```

How to validate the form

- Do not validate the form
- Validate data
  - Yaml is preferred over annotation because yml are overridable at ORO

What to do when form shouldn't update the data?

  - Like "Apply terms checkbox"
  - Make field mapped: false and validate form

Always use objects instead of Array

- When you don't have an object
- or object doesn't match the form?

- Create DTO

Example:

- ProductRow DTO
- ProductRowType FormType
- validation.yml with DTO validation

Validation Constraint

- Built-in constraints
- Contrib bundles with popular constraints
- Create a custom constraint
- Callback validator
- Validation groups
- Validation sequences

Validation Formats

- Annotations
- Yml (preferred for ORO apps)

Debugging

- Symfony Toolbar Forms Tab
- Twig Inspector
- Symfony Toolbar Validation Tab