



*Ministerul Educației, Culturii și Cercetării a Republicii Moldova*

*Universitatea de Stat a Moldovei*

*Facultatea Matematică și Informatică*

# Raport

**Lucrare de laborator Nr.4**

**la disciplina Framework-uri pentru dezvoltarea de aplicații web**

**Tema:”Formulare”**

**A efectuat : Pistol Adrian**

**A verificat : Bodrug Svetlana**

**Chișinău 2020**

## Cuprins

<b>Instalare .....</b>	<b>3</b>
<b>Utilizare .....</b>	<b>3</b>
<b>Crearea de formulare în controlere .....</b>	<b>4</b>
<b>Crearea claselor de formular .....</b>	<b>4</b>
<b>Formulare de redare.....</b>	<b>6</b>
<b>Rezultat:.....</b>	<b>7</b>
<b>Formulare de procesare .....</b>	<b>8</b>
<b>Formulare de validare .....</b>	<b>9</b>
<b>Alte caracteristici comune ale formularului.....</b>	<b>10</b>
<b>Trecerea opțiunilor la formulare .....</b>	<b>10</b>
<b>Rezultate:.....</b>	<b>10</b>
<b>Opțiuni tip formular .....</b>	<b>11</b>
<b>Eticheta Opțiune .....</b>	<b>11</b>
<b>Rezultate:.....</b>	<b>12</b>
<b>Rezultate:.....</b>	<b>13</b>

## Instalare

În aplicațiile care utilizează Symfony Flex, rulez această comandă pentru a instala caracteristica formularului înainte de a o utiliza:

```
C:\Users\User\Desktop\FDAW\Symfony\my_project_name> composer require symfony/form
./composer.json has been updated
Running composer update symfony/form
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
```

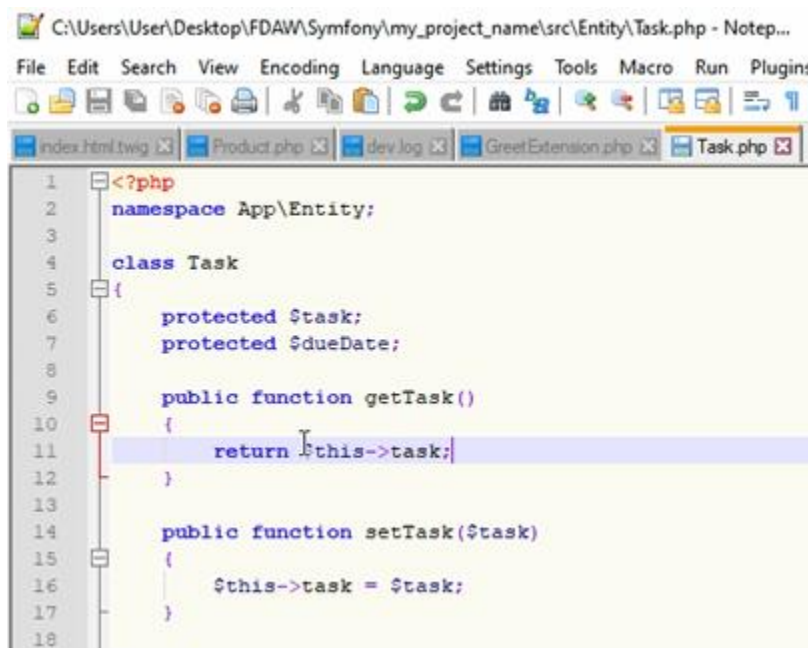
## Utilizare

Fluxul de lucru recomandat atunci când lucrez cu formularele Symfony este următorul:

- Construesc formularul într-un controler Symfony sau utilizând o clasă de formular dedicată;
- Redau formularul într-un șablon, astfel încât să îl pot edita și trimite;
- Procesez formularul pentru a valida datele trimise, transformându-le în date PHP și fac ceva cu acesta (de exemplu, le persist într-o bază de date).

Fiecare dintre acești pași este explicat în detaliu în secțiunile următoare. Pentru a face exemplele mai ușor de urmărit, toți presupun că construim o mică aplicație de listă Todo care afișează „sarcini”.

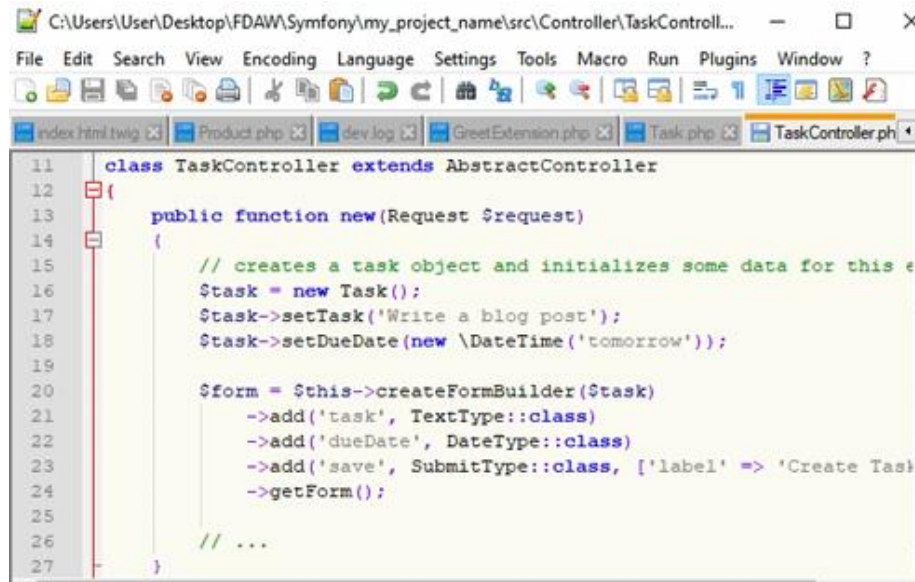
Creez și editez sarcini folosind formularele Symfony. Fiecare activitate este o instanță a următoarei clase de activități:



```
C:\Users\User\Desktop\FDAW\Symfony\my_project_name\src\Entity\Task.php - Notep...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins
index.html.twig x Product.php x dev.log x GreetExtension.php x Task.php x
1 <?php
2 namespace App\Entity;
3
4 class Task
5 {
6     protected $task;
7     protected $dueDate;
8
9     public function getTask()
10    {
11        return $this->task;
12    }
13
14    public function setTask($task)
15    {
16        $this->task = $task;
17    }
18 }
```

## Crearea de formulare în controlere

Dacă controlerul se extinde de la `AbstractController`, utilizez ajutorul de tip `createFormBuilder()`:

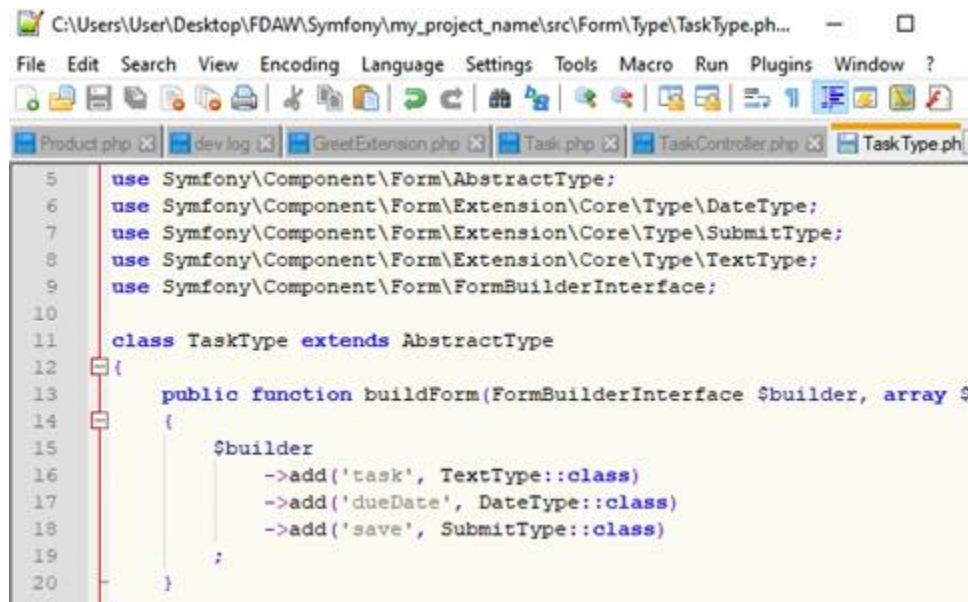


```
11 class TaskController extends AbstractController
12 {
13     public function new(Request $request)
14     {
15         // creates a task object and initializes some data for this e
16         $task = new Task();
17         $task->setTask('Write a blog post');
18         $task->setDueDate(new \DateTime('tomorrow'));
19
20         $form = $this->createFormBuilder($task)
21             ->add('task', TextType::class)
22             ->add('dueDate', DateType::class)
23             ->add('save', SubmitType::class, ['label' => 'Create Task'])
24             ->getForm();
25
26         // ...
27     }
```

## Crearea claselor de formular

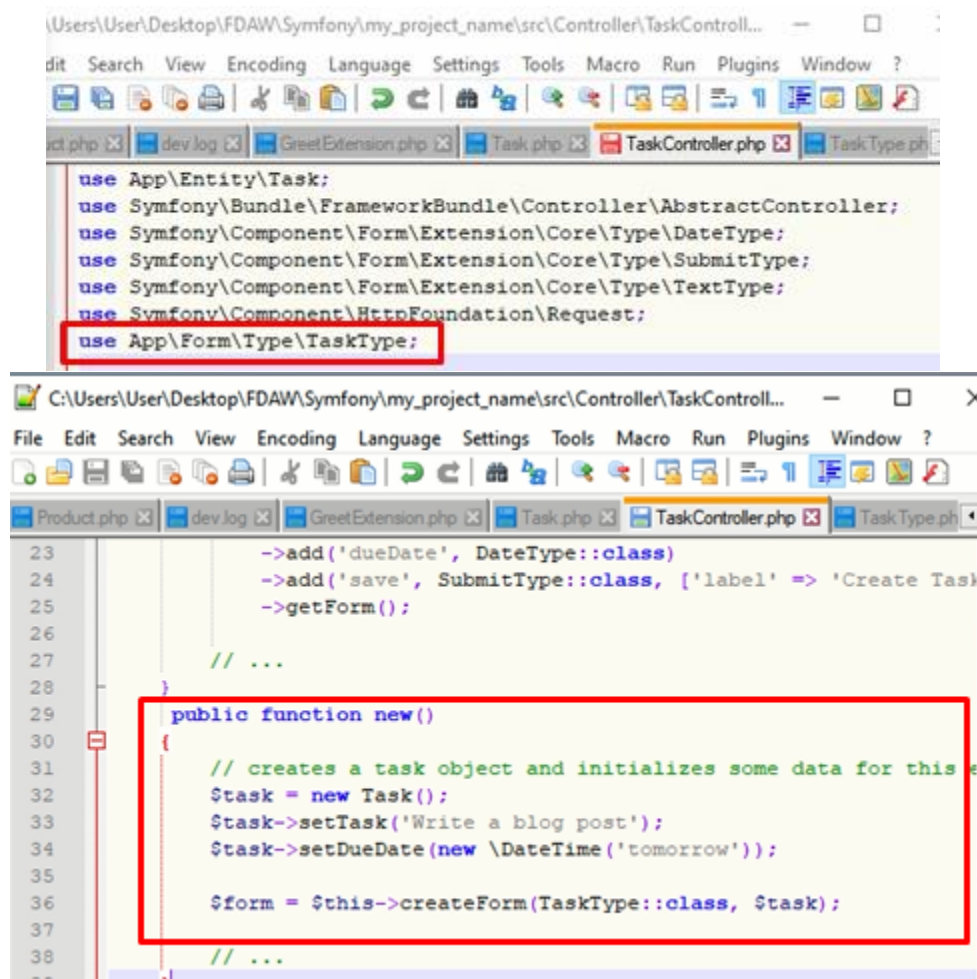
Symfony recomandă să pun cât mai puțină logică în controlere. De aceea, este mai bine să mut forme complexe în clase dedicate, în loc să le definesc în acțiunile controlerului. În plus, formularele definite în clase pot fi refolosite în mai multe acțiuni și servicii.

Clasele de formulare sunt tipuri de formulare care implementează `Symfony \ Component \ Form \ FormTypeInterface`. Cu toate acestea, este mai bine să extind din `Symfony \ Component \ Form \ AbstractType`, care implementează deja interfața și oferă câteva utilitare:



```
5 use Symfony\Component\Form\AbstractType;
6 use Symfony\Component\Form\Extension\Core\Type\DateType;
7 use Symfony\Component\Form\Extension\Core\Type\SubmitType;
8 use Symfony\Component\Form\Extension\Core\Type\TextType;
9 use Symfony\Component\Form\FormBuilderInterface;
10
11 class TaskType extends AbstractType
12 {
13     public function buildForm(FormBuilderInterface $builder, array $
14     {
15         $builder
16             ->add('task', TextType::class)
17             ->add('dueDate', DateType::class)
18             ->add('save', SubmitType::class)
19     }
20 }
```

Clasa formular conține toate direcțiile necesare pentru a crea formularul sarcinii. În controlerele care se extind de la AbstractController, utilizez ajutorul helpFreeForm () (în caz contrar, utilizez metoda create () a serviciului form.factory):

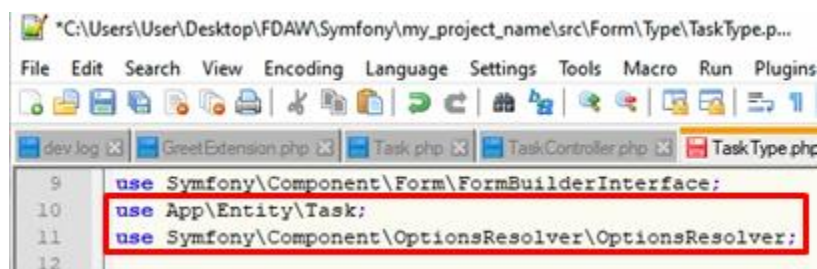


```
use App\Entity\Task;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Form\Extension\Core\Type\DateType;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\HttpFoundation\Request;
use App\Form\Type\TaskType;
```

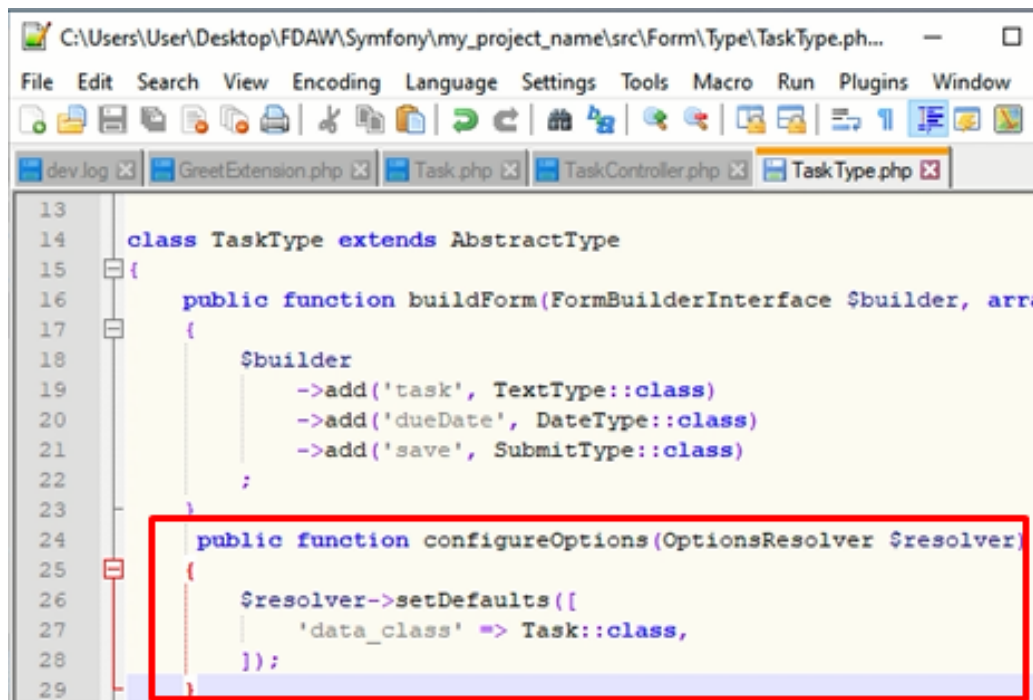
```
23         ->add('dueDate', DateType::class)
24         ->add('save', SubmitType::class, ['label' => 'Create Task'])
25         ->getForm();
26
27         // ...
28     }
29
30     public function new()
31     {
32         // creates a task object and initializes some data for this
33         $task = new Task();
34         $task->setTask('Write a blog post');
35         $task->setDueDate(new \DateTime('tomorrow'));
36
37         $form = $this->createForm(TaskType::class, $task);
38
39         // ...
40     }
```

Fiecare formular trebuie să știe numele clasei care deține datele subiacente (de exemplu, App \ Entity \ Task). De obicei, acest lucru este doar ghicit pe baza obiectului trecut la al doilea argument pentru createForm () (adică \$ task). Mai târziu, când încep să încorporez formulare, acest lucru nu va mai fi suficient.

Deci, deși nu este întotdeauna necesar, este, în general, o idee bună să specific în mod explicit opțiunea data\_class adăugând următoarele la clasa de tip formular:



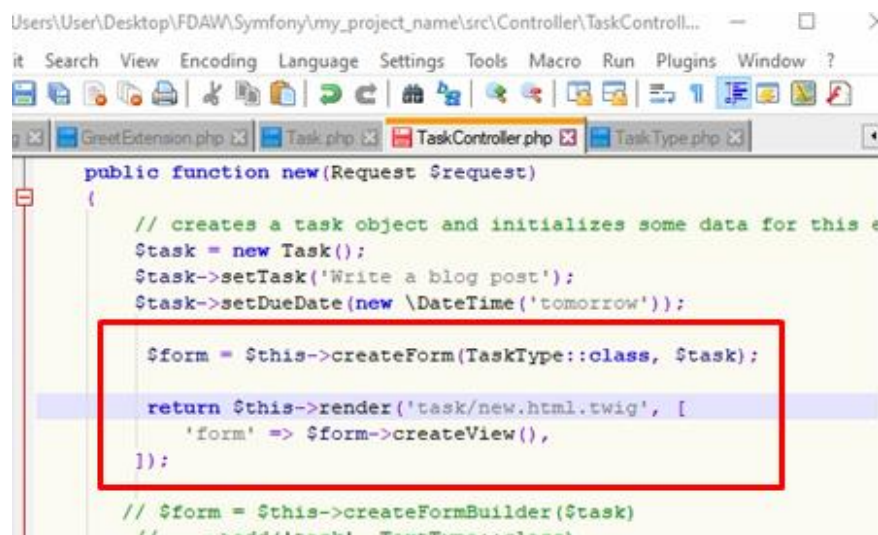
```
use Symfony\Component\Form\FormBuilderInterface;
use App\Entity\Task;
use Symfony\Component\OptionsResolver\OptionsResolver;
```



```
13
14 class TaskType extends AbstractType
15 {
16     public function buildForm(FormBuilderInterface $builder, array $options)
17     {
18         $builder
19             ->add('task', TextType::class)
20             ->add('dueDate', DateType::class)
21             ->add('save', SubmitType::class)
22     }
23
24     public function configureOptions(OptionsResolver $resolver)
25     {
26         $resolver->setDefaults([
27             'data_class' => Task::class,
28         ]);
29     }
}
```

## Formulare de redare

Acum că formularul a fost creat, următorul pas este redarea acestuia. În loc să trec întregul obiect formular la șablon, utilizez metoda `createView()` pentru a construi un alt obiect cu reprezentarea vizuală a formularului:



```
Users\User\Desktop\FDAW\Symfony\my_project_name\src\Controller\TaskControll...
it Search View Encoding Language Settings Tools Macro Run Plugins Window ?

GreetExtension.php Task.php TaskController.php TaskType.php

public function new(Request $request)
{
    // creates a task object and initializes some data for this e
    $task = new Task();
    $task->setTask('Write a blog post');
    $task->setDueDate(new \DateTime('tomorrow'));

    $form = $this->createForm(TaskType::class, $task);

    return $this->render('task/new.html.twig', [
        'form' => $form->createView(),
    ]);

    // $form = $this->createFormBuilder($task)
    //     ->add('task', TextType::class)
}
```



```

3
4 use App\Entity\Task;
5 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
6 use Symfony\Component\Form\Extension\Core\Type\DateType;
7 use Symfony\Component\Form\Extension\Core\Type\SubmitType;
8 use Symfony\Component\Form\Extension\Core\Type\TextType;
9 use Symfony\Component\HttpFoundation\Request;
10 use App\Form\Type\TaskType;
11 use Symfony\Component\Routing\Annotation\Route;
12
13 class TaskController extends AbstractController
14 {
15     /**
16      * @Route("/form")
17      */
18     public function new(Request $request)
19     {
20     }
21 }

```

Apoi, utilizez câteva funcții de ajutor de formular pentru a reda conținutul formularului:

```

1 {{ form(form) }}

```

**Rezultat:**

Task Write a blog post

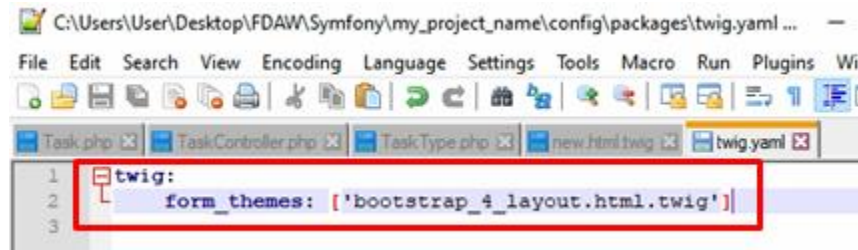
Due date

Year Month Day

Nov 11 2020

Save

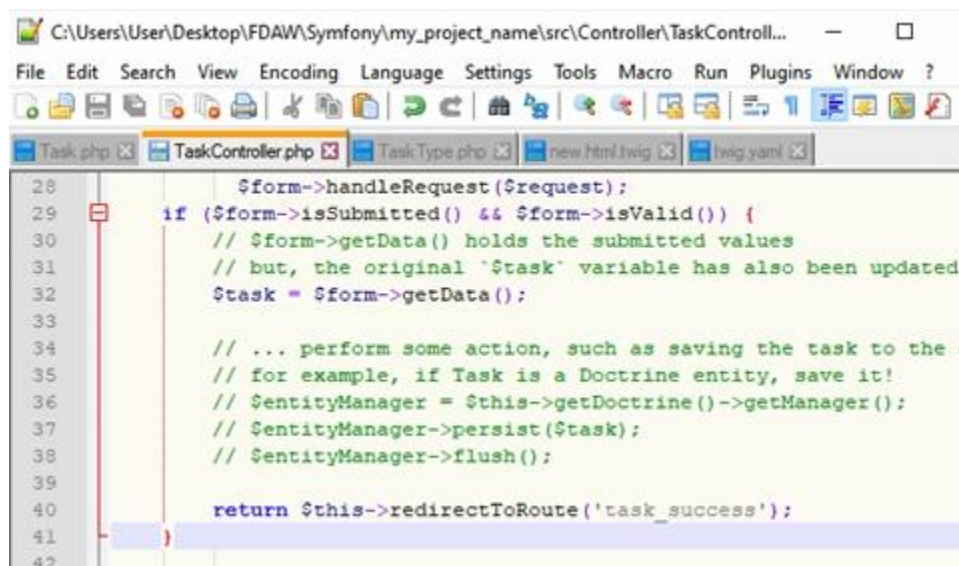
Pe cât de scurtă este această redare, nu este foarte flexibilă. De obicei, voi avea nevoie de mai mult control asupra aspectului întregului formular sau a câtorva dintre câmpurile sale. De exemplu, datorită integrării Bootstrap 4 cu formulare Symfony pot seta această opțiune pentru a genera formulare compatibile cu cadrul Bootstrap 4 CSS:



## Formulare de procesare

Modul recomandat de procesare a formularelor este acela de a utiliza o singură acțiune atât pentru redarea formularului, cât și pentru gestionarea formularului de trimitere. Pot utiliza acțiuni separate, dar folosind o acțiune simplifică totul, păstrând în același timp codul concis și mentenabil.

Prelucrarea unui formular înseamnă traducerea datelor trimise de utilizator înapoi la proprietățile unui obiect. Pentru a face acest lucru, datele trimise de la utilizator trebuie să fie scrise în obiectul formular:





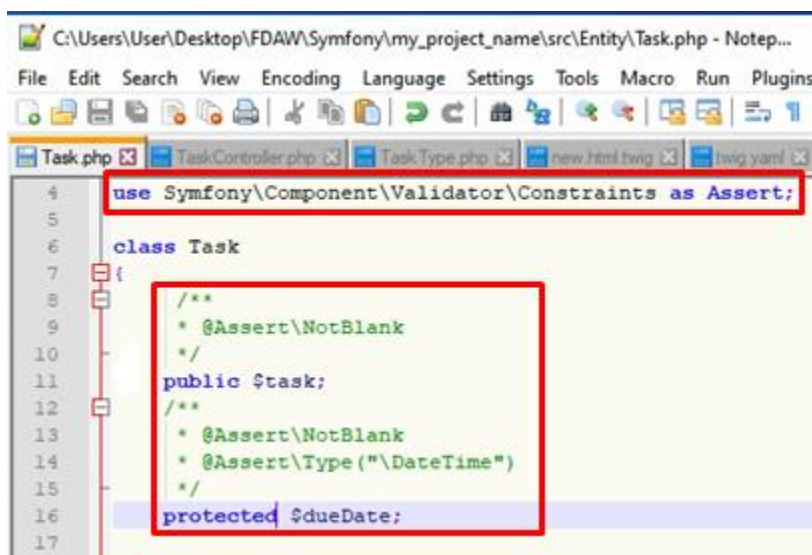
## Formulare de validare

În secțiunea anterioară, am aflat cum poate fi trimis un formular cu date valide sau nevalide. În Symfony, întrebarea nu este dacă „formularul” este valid, ci dacă obiectul subiacent (\$ task în acest exemplu) este valid sau nu după ce formularul i-a aplicat datele trimise. Apelarea \$ form->isValid () este o comandă rapidă care întreabă obiectul \$ task dacă are sau nu date valide.

Înainte de a utiliza validarea, adaug asistență pentru aceasta în aplicație:

```
C:\Users\User\Desktop\FDAW\Symfony\my_project_name> composer require symfony/validator
./composer.json has been updated
Running composer update symfony/validator
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
```

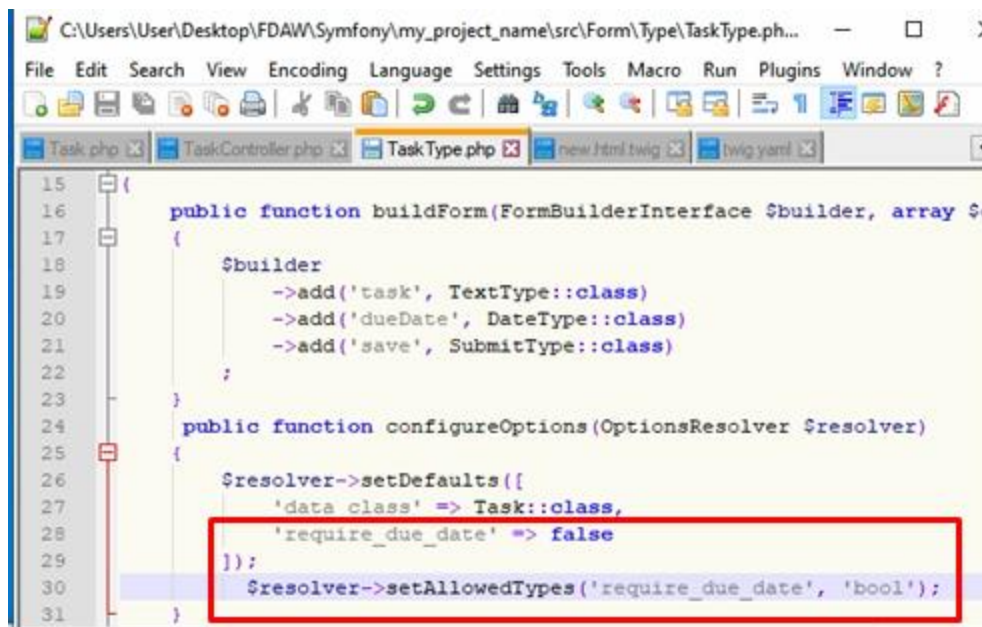
Pentru a vedea prima abordare - adăugarea de constrângeri la entitate - în acțiune, adăugați constrângerile de validare, astfel încât câmpul de activitate să nu poată fi gol, iar câmpul dueDate să nu poată fi gol și trebuie să fie un obiect DateTime valid.



## Alte caracteristici comune ale formularului

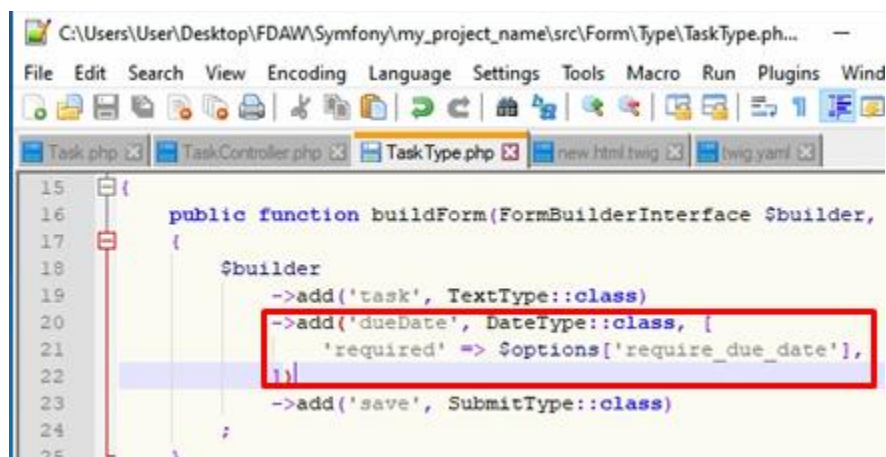
### Trecerea opțiunilor la formular

Dacă încerc să utilizez formularul acum, voi vedea un mesaj de eroare: opțiunea „require\_due\_date” nu există. Acest lucru se datorează faptului că formularele trebuie să declare toate opțiunile pe care le acceptă folosind metoda `configureOptions()`:



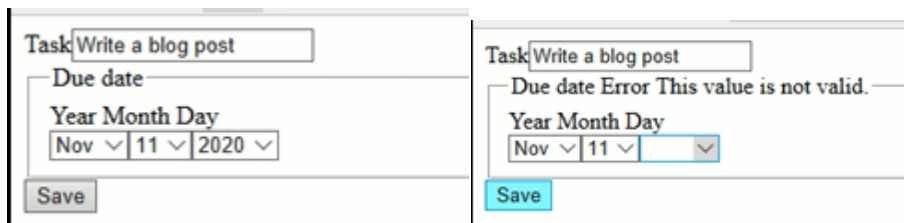
```
15 {  
16     public function buildForm(FormBuilderInterface $builder, array $options)  
17     {  
18         $builder  
19             ->add('task', TextType::class)  
20             ->add('dueDate', DateType::class)  
21             ->add('save', SubmitType::class)  
22     }  
23  
24     public function configureOptions(OptionsResolver $resolver)  
25     {  
26         $resolver->setDefaults([  
27             'data_class' => Task::class,  
28             'require_due_date' => false  
29         ]);  
30         $resolver->setAllowedTypes('require_due_date', 'bool');  
31     }  
}
```

Acum pot utiliza această nouă opțiune de formular în cadrul metodei `buildForm()`:



```
15 {  
16     public function buildForm(FormBuilderInterface $builder,  
17     {  
18         $builder  
19             ->add('task', TextType::class)  
20             ->add('dueDate', DateType::class, [  
21                 'required' => $options['require_due_date'],  
22             ])  
23             ->add('save', SubmitType::class)  
24     }  
25 }
```

### Rezultate:



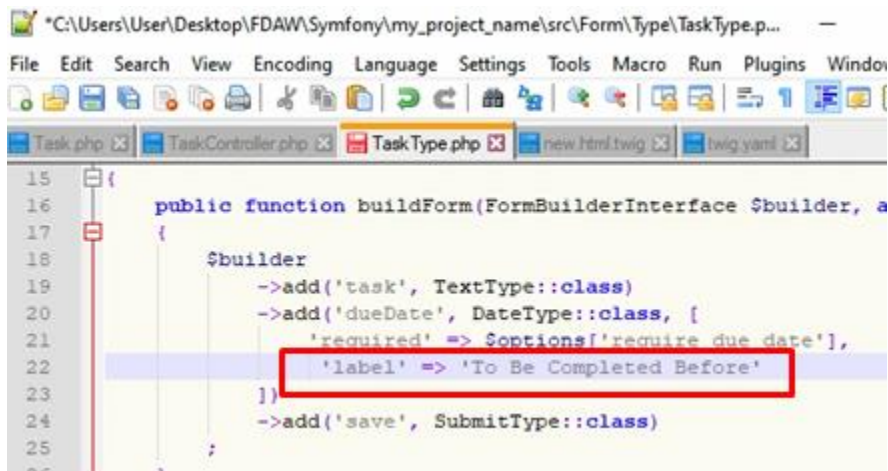
Task Write a blog post	Task Write a blog post
Due date	Due date Error This value is not valid.
Year Month Day	Year Month Day
Nov 11 2020	Nov 11
Save	Save

## Opțiuni tip formular

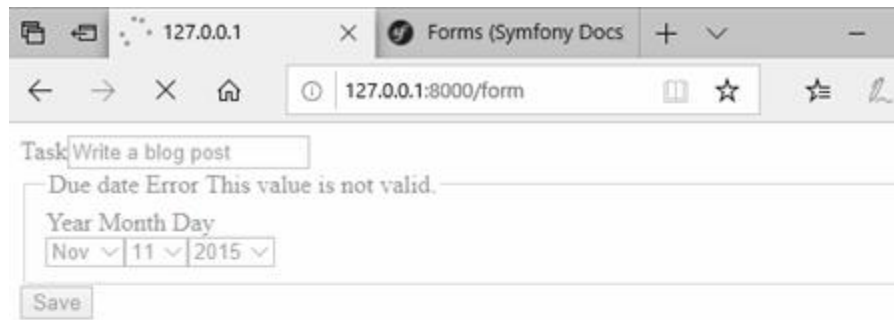
Fiecare tip de formular are un număr de opțiuni pentru a-l configura, așa cum se explică în referința tipurilor de formular Symfony. Sunt necesare două opțiuni utilizate în mod obișnuit și etichetă.

### Eticheta Opțiune

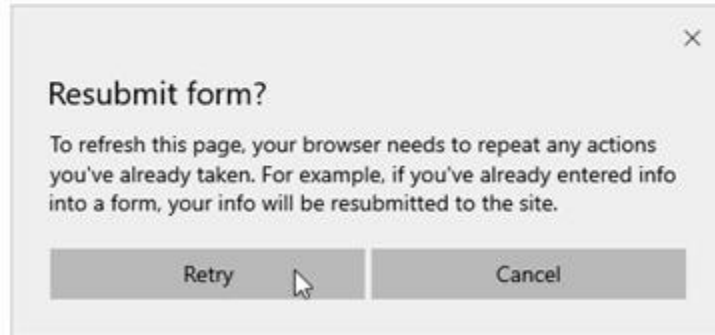
În mod implicit, eticheta câmpurilor formularului este versiunea umanizată a numelui proprietății (utilizator -> Utilizator; Adresă poștală -> Adresă poștală). Setez opțiunea etichetă pe câmpuri pentru a le defini în mod explicit etichetele:



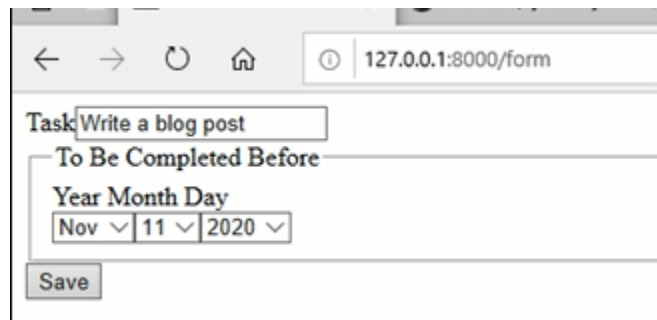
## Rezultate:



A screenshot of a web browser window. The address bar shows '127.0.0.1:8000/form'. The page title is 'Forms (Symfony Docs)'. The form contains a 'Task' field with the value 'Write a blog post'. Below it is a 'Due date' field with the error message 'Error This value is not valid.' and a date picker showing 'Nov 11 2015'. A 'Save' button is at the bottom.



A dialog box titled 'Resubmit form?' with a close button (X) in the top right corner. The text inside reads: 'To refresh this page, your browser needs to repeat any actions you've already taken. For example, if you've already entered info into a form, your info will be resubmitted to the site.' At the bottom are two buttons: 'Retry' and 'Cancel'. A mouse cursor is hovering over the 'Retry' button.



A screenshot of the same web browser window after a page refresh. The 'Task' field still contains 'Write a blog post'. The 'Due date' field now shows 'Nov 11 2020' and no longer has an error message. The 'Save' button remains at the bottom.

```
C:\Users\User\Desktop\FDAW\Symfony\my_project_name\src\Controller\TaskControll...
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Windc
Task.php TaskController.php TaskType.php new.html.twig twig.yaml
49
50
51 // $form = $this->createFormBuilder($task)
52 //   ->add('task', TextType::class)
53 //   ->add('dueDate', DateType::class)
54 //   ->add('save', SubmitType::class, ['label' => 'C
55 //   ->getForm();
56
57 // ...
58
59 /**
60  * @Route("/success", name="task_success")
61  */
62 public function success(): Response
63 {
64     return new Response(
65         '<html><body>Task Success!!!</body></html>'

```

## Rezultate:

← → ↻ 🏠 127.0.0.1:8000/form

Task

To Be Completed Before

Year Month Day

Nov ▾ 11 ▾ 2020 ▾

