



**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 : Batîr Ana**

**A verificat : Bodrug Svetlana**

**Chișinău 2020**

## Instalare

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

```
F:\symfony\Lab4\quick_tour>composer require symfony/form
./composer.json has been updated
Running composer update symfony/form
Loading composer repositories with package information
Updating dependencies
Restricting packages listed in "symfony/symfony" to "5.1.*"
Lock file operations: 4 installs, 0 updates, 0 removals
  - Locking symfony/form (v5.1.8)
  - Locking symfony/intl (v5.1.8)
  - Locking symfony/options-resolver (v5.1.8)
  - Locking symfony/polyfill-intl-icu (v1.20.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 4 installs, 0 updates, 0 removals
  - Installing symfony/options-resolver (v5.1.8): Extracting archive
  - Installing symfony/intl (v5.1.8): Extracting archive
```

## 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 construiți 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:

```
r\src\Entity\Task.php (quick_tour) - Sublime Text (...)
```

View Goto Tools Project Preferences Help

```
Task.php
1 <?php
2 namespace App\Entity;
3
4
5 class Task
6 {
7
8     protected $task;
9     protected $dueDate;
10
11     public function getTask()
12     {
13         return $this->task;
14     }
15
16     public function setTask($task)
17     {
18         $this->task = $task;
19     }
20
21     public function getDueDate()
22     {
23         return $this->dueDate;
24     }
25
26     public function setDueDate(\DateTime $
27         dueDate = null)
28     {
29         $this->dueDate = $dueDate;
30     }
31 }
32 ?>
```

## Crearea de formulare în controlere

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

```
r\src\Controller\TaskController.php (quick_tour) - Sublime Text (UNREGISTERED)
```

ew Goto Tools Project Preferences Help

```
TaskController.php
1 <?php
2 // src/Controller/TaskController.php
3 namespace App\Controller;
4
5 use App\Entity\Task;
6 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
7 use Symfony\Component\Form\Extension\Core\Type\DateTimeType;
8 use Symfony\Component\Form\Extension\Core\Type\SubmitType;
9 use Symfony\Component\Form\Extension\Core\Type\TextType;
10 use Symfony\Component\HttpFoundation\Request;
11
12 class TaskController extends AbstractController
13 {
14     public function new(Request $request)
15     {
16         // creates a task object and initializes some data for this example
17         $task = new Task();
18         $task->setTask('Write a blog post');
19         $task->setDueDate(new \DateTime('tomorrow'));
20
21         $form = $this->createFormBuilder($task)
22             ->add('task', TextType::class)
23             ->add('dueDate', DateTimeType::class)
24             ->add('save', SubmitType::class, ['label' => 'Create Task'])
25             ->getForm();
26
27         // ...
28     }
29 }
```

## 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:

```
\src\Form\Type\TaskType.php (quick_tour) - Sublime Text (UNREGISTERED)
View Goto Tools Project Preferences Help

TaskController.php x TaskType.php x Task.php x

1  <?php
2  namespace App\Form\Type;
3
4  use Symfony\Component\Form\AbstractType;
5  use Symfony\Component\Form\Extension\Core\Type\DateType;
6  use Symfony\Component\Form\Extension\Core\Type\SubmitType;
7  use Symfony\Component\Form\Extension\Core\Type\TextType;
8  use Symfony\Component\Form\FormBuilderInterface;
9
10 class TaskType extends AbstractType
11 {
12     public function buildForm(FormBuilderInterface $builder, array $options)
13     {
14         $builder
15             ->add('task', TextType::class)
16             ->add('dueDate', DateType::class)
17             ->add('save', SubmitType::class)
18         ;
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`):

```
\src\Controller\TaskController.php (quick_tour) - Sublime Text (UNREGISTERED)
View Goto Tools Project Preferences Help

TaskController.php x TaskType.php x Task.php x

1  <?php
2  // src/Controller/TaskController.php
3  namespace App\Controller;
4
5  use App\Form\Type\TaskType;
6  use App\Entity\Task;
7  use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
8  use Symfony\Component\Form\Extension\Core\Type\DateType;
9  use Symfony\Component\Form\Extension\Core\Type\SubmitType;
10 use Symfony\Component\Form\Extension\Core\Type\TextType;
11 use Symfony\Component\HttpFoundation\Request;
12
13 class TaskController extends AbstractController
14 {
15     public function new()
16     {
17         // creates a task object and initializes some data for this example
18         $task = new Task();
19         $task->setTask('Write a blog post');
20         $task->setDueDate(new \DateTime('tomorrow'));
21
22         $form = $this->createForm(TaskType::class, $task);
23
24         // ...
25     }
26 }
```

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:

\src\Form\Type\TaskType.php (quick\_tour) - Sublime Text (UNREGISTERED)

ew Goto Tools Project Preferences Help

```
1  <?php
2  namespace App\Form\Type;
3
4  use App\Entity\Task;
5  use Symfony\Component\OptionsResolver\OptionsResolver;
6  use Symfony\Component\Form\AbstractType;
7  use Symfony\Component\Form\Extension\Core\Type\DateType;
8  use Symfony\Component\Form\Extension\Core\Type\SubmitType;
9  use Symfony\Component\Form\Extension\Core\Type\TextType;
10 use Symfony\Component\Form\FormBuilderInterface;
11
12 class TaskType extends AbstractType
13 {
14     public function buildForm(FormBuilderInterface $builder, array $options)
15     {
16         $builder
17             ->add('task', TextType::class)
18             ->add('dueDate', DateType::class)
19             ->add('save', SubmitType::class)
20         ;
21     }
22
23     public function configureOptions(OptionsResolver $resolver)
24     {
25         $resolver->setDefaults([
26             'data_class' => Task::class,
27         ]);
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:

```
(src\Controller\TaskController.php (quick_tour) - Sublime Text (UNREGISTERED))  
ew Goto Tools Project Preferences Help  
TaskController.php x TaskType.php x Task.php x  
1 <?php  
2 // src/Controller/TaskController.php  
3 namespace App\Controller;  
4  
5 use App\Form\Type\TaskType;  
6 use App\Entity\Task;  
7 use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;  
8 use Symfony\Component\Form\Extension\Core\Type\DateType;  
9 use Symfony\Component\Form\Extension\Core\Type\SubmitType;  
10 use Symfony\Component\Form\Extension\Core\Type\TextType;  
11 use Symfony\Component\HttpFoundation\Request;  
12  
13 class TaskController extends AbstractController  
14 {  
15     public function new(Request $request)  
16     {  
17         $task = new Task();  
18  
19         $task = new Task();  
20         $task->setTask('Write a blog post');  
21         $task->setDueDate(new \DateTime('tomorrow'));  
22  
23         $form = $this->createForm(TaskType::class, $task);  
24  
25         return $this->render('task/new.html.twig', [  
26             'form' => $form->createView(),  
27         ]);  
28     }  
29 }
```

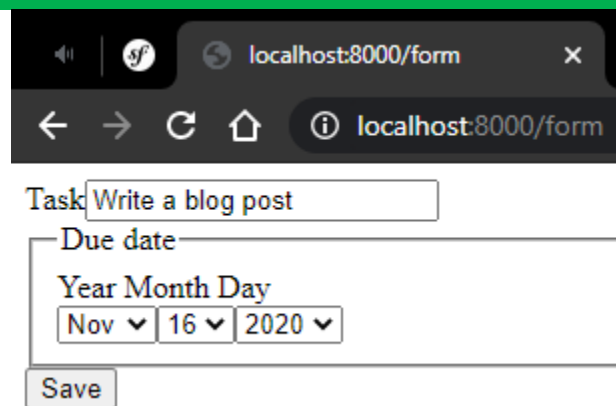
Apoi, utilizez câteva funcții de ajutor de formular pentru a reda conținutul formularului:

```
(templates\task\new.html.twig (quick_tour) ·  
iew Goto Tools Project Preferences  
new.html.twig x Ta  
1 {{ form(form) }}
```

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:

```
\config\packages\twig.yaml (quick_tour) - Sublime Text (UNREGISTERED)
iew Goto Tools Project Preferences Help
twig.yaml x new.html.twig x TaskController.php
1 # config/packages/twig.yaml
2 twig:
3   form_themes: ['bootstrap_4_layout.html.twig']
```

## REZULTAT:



localhost:8000/form

Task Write a blog post

Due date

Year Month Day

Nov 16 2020

Save

### 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:

\src\Controller\TaskController.php (quick\_tour) - Sublime Text (UNREGISTERED)

iew Goto Tools Project Preferences Help

```
17 * @Route("/form")
18 */
19 public function new(Request $request)
20 {
21     // creates a task object and initializes some data for this
    example
22     $task = new Task();
23     $task->setTask('Write a blog post');
24     $task->setDueDate(new \DateTime('tomorrow'));
25
26     $form = $this->createForm(TaskType::class, $task);
27
28     $form->handleRequest($request);
29     if ($form->isSubmitted() && $form->isValid()) {
30         // $form->getData() holds the submitted values
31         // but, the original `$task` variable has also been updated
32         $task = $form->getData();
33
34         // ... perform some action, such as saving the task to the
        database
35         // for example, if Task is a Doctrine entity, save it!
36         // $entityManager = $this->getDoctrine()->getManager();
37         // $entityManager->persist($task);
38         // $entityManager->flush();
39
40         return $this->redirectToRoute('task_success');
41     }
```

## 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:

```
F:\symfony\Lab4\quick_tour>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
Generating optimized autoload files
```



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.

```
src\Entity\Task.php (quick_tour) - Sublime Text (UNREGISTERED)
ew Goto Tools Project Preferences Help

TaskController.php x DefaultController.php x TaskType.php x Task.php x
1 <?php
2 namespace App\Entity;
3
4 use Symfony\Component\Validator\Constraints\NotBlank;
5 use Symfony\Component\Validator\Constraints\Type;
6 use Symfony\Component\Validator\Mapping\ClassMetadata;
7 use Symfony\Component\Validator\Constraints as Assert;
8
9 class Task
10 {
11     /**
12      * @Assert\NotBlank
13      */
14     public $task;
15     /**
16      * @Assert\NotBlank
17      * @Assert\Type("\DateTime")
18      */
19     protected $dueDate;
```

## Alte caracteristici comune ale formularului

### Trecerea opțiunilor la formulare

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 ():

```
src\Form\Type\TaskType.php (quick_tour) - Sublime Text (UNREGISTERED)
ew Goto Tools Project Preferences Help

twig.yaml x new.html.twig x TaskType.php x new.html.twig x
9 use Symfony\Component\Form\Extension\Core\Type\TextType;
10 use Symfony\Component\Form\FormBuilderInterface;
11
12 class TaskType extends AbstractType
13 {
14     public function buildForm(FormBuilderInterface $builder, array
15         $options)
16     {
17         $builder
18             ->add('task', TextType::class)
19             ->add('dueDate', DateType::class)
20             ->add('save', SubmitType::class)
21     }
22
23     public function configureOptions(OptionsResolver $resolver)
24     {
25         $resolver->setDefaults([
26             'data_class' => Task::class,
27             'require_due_date' => false,
28         ]);
29
30         // you can also define the allowed types, allowed values
31         // and
32         // any other feature supported by the OptionsResolver
33         $resolver->setAllowedTypes('require_due_date', 'bool');
```

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

```
src/Form/Type/TaskType.php (quick_tour) - Sublime Text (UNREGISTERED)
File Edit View Goto Tools Project Preferences Help

twig.yaml x new.html.twig x TaskType.php x new.html.twig x

5 use Symfony\Component\OptionsResolver\OptionsResolver;
6 use Symfony\Component\Form\AbstractType;
7 use Symfony\Component\Form\Extension\Core\Type\DateType;
8 use Symfony\Component\Form\Extension\Core\Type\SubmitType;
9 use Symfony\Component\Form\Extension\Core\Type\TextType;
10 use Symfony\Component\Form\FormBuilderInterface;
11
12 class TaskType extends AbstractType
13 {
14     public function buildForm(FormBuilderInterface $builder, array
        $options)
15     {
16         $builder
17             ->add('task', TextType::class)
18             ->add('dueDate', DateType::class, [
19                 'required' => $options['require_due_date'],
20             ])
21             ->add('save', SubmitType::class)
22     }
23 }
```

## REZULTAT:

Forms (Symfony) x localhost:8000/form

localhost:8000/form

Task Write a blog post

Due date

Year Month Day

Nov 16 2020

Save

localhost:8000/form

Task Write a blog post

Due date Error This value should not be blank.

Year Month Day

Save

Task Write a blog post

Due date Error This value is not valid.

Year Month Day

Jan 1

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.

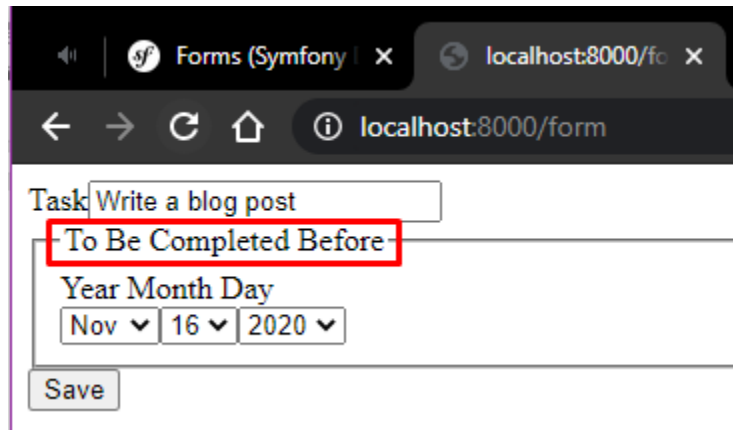
## Eticheta Opțiune

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:

```
src/Form\Type\TaskType.php (quick_tour) - Sublime Text (UNREGISTERED)
ew Goto Tools Project Preferences Help

twig.yaml x new.html.twig x TaskType.php x new.html.twig x
13 {
14     public function buildForm(FormBuilderInterface $builder, array
        $options)
15     {
16         $builder
17             ->add('task', TextType::class)
18             ->add('dueDate', DateType::class, [
19                 'required' => $options['require due date'],
20                 'label' => 'To Be Completed Before'
21             ])
22             ->add('save', SubmitType::class)
23         ;
24     }
25 }
```

## REZULTAT:



Task Write a blog post

To Be Completed Before

Year Month Day

Nov 16 2020

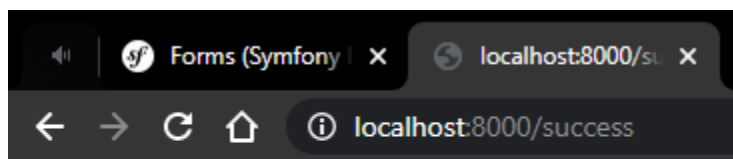
Save

(src)\Controller\TaskController.php (quick\_tour) - Sublime Text (UNREGISTERED)

ew Goto Tools Project Preferences Help

```
new.html.twig TaskController.php DefaultController.php
53 /**
54  * @Route("/success", name="task_success")
55  */
56 public function success(): Response
57 {
58     return new Response(
59         '<html><body>Task is DONE!!!</body></html>'
60     );
61 }
62 }
```

## REZULTAT:



localhost:8000/success

Task is DONE!!!