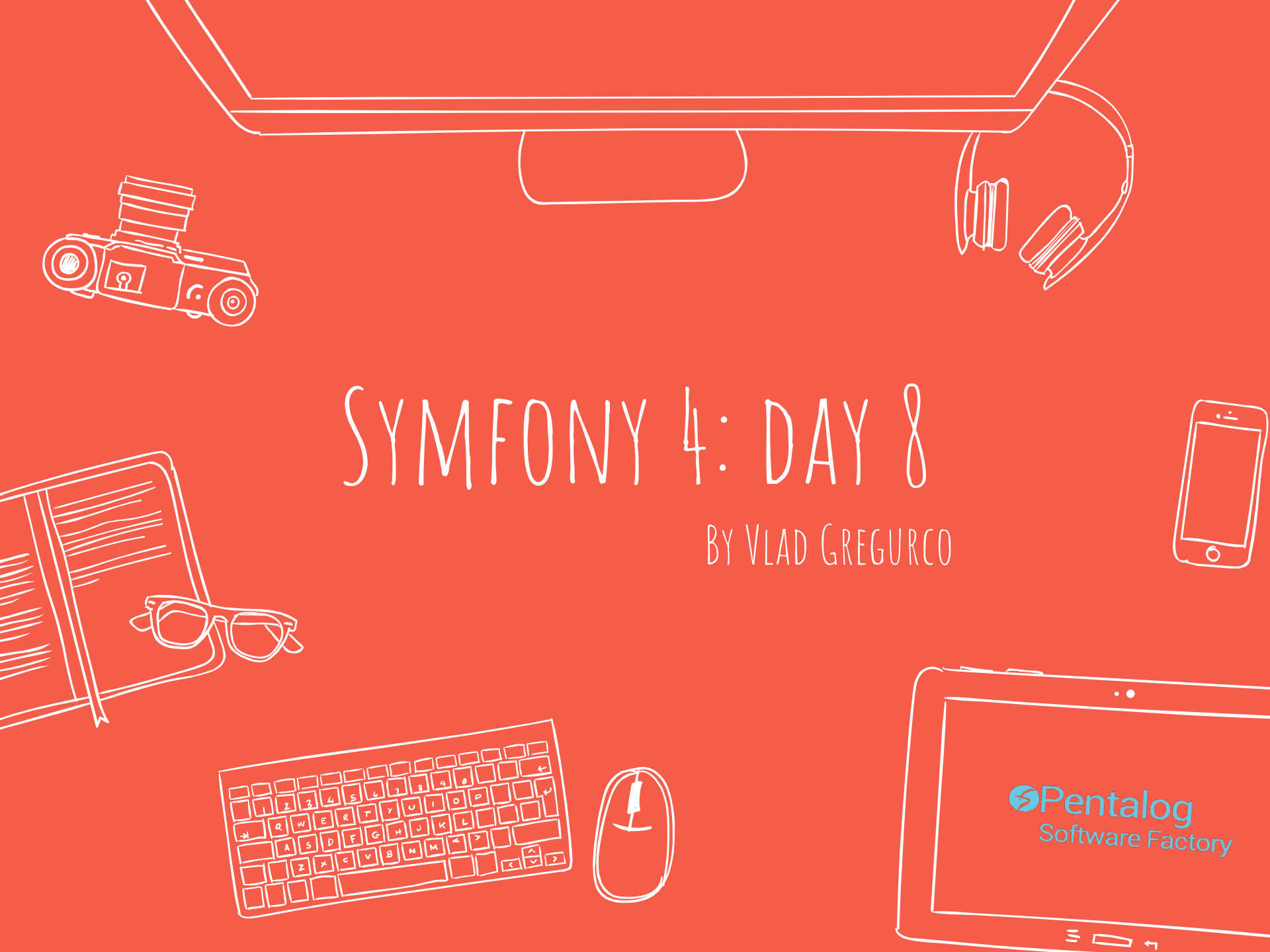


# SYMPHONY 4: DAY 8

BY VLAD GREGURCO





QUESTIONS?



# CREATE JOB ACTION

```
57  /**
58   * Creates a new job entity.
59   *
60   * @Route("/job/create", name="job.create")
61   * @Method({"GET", "POST"})
62   *
63   * @param Request $request
64   * @param EntityManagerInterface $em
65   *
66   * @return RedirectResponse|Response
67   */
68  public function createAction(Request $request, EntityManagerInterface $em) : Response
69  {
70      $job = new Job();
71      $form = $this->createForm(JobType::class, $job);
72      $form->handleRequest($request);
73
74      if ($form->isSubmitted() && $form->isValid()) {
75          $em->persist($job);
76          $em->flush();
77
78          return $this->redirectToRoute('job.list');
79      }
80
81      return $this->render('job/create.html.twig', [
82          'form' => $form->createView(),
83      ]);
84  }
```

# CREATE JOB TEMPLATE

```
1 {% extends 'base.html.twig' %}
2
3 {% block body %}
4     <h1>Job creation</h1>
5
6     {{ form_start(form, {'attr': {'novalidate': 'novalidate'}}) }}
7     {{ form_widget(form) }}
8
9     <div class="form-group">
10         <div class="col-sm-offset-2 col-sm-10">
11             <button type="submit" class="btn btn-default">Create</button>
12         </div>
13     </div>
14     {{ form_end(form) }}
15 {% endblock %}
```

# BUILD FORM (PARTIAL)

```
24 class JobType extends AbstractType
25 {
26     /**
27      * {@inheritdoc}
28      */
29     public function buildForm(FormBuilderInterface $builder, array $options)
30     {
31         $builder
32             ->add('company', TextType::class, [
33                 'constraints' => [
34                     new NotBlank(),
35                     new Length(['max' => 255]),
36                 ]
37             )
38             ->add('logo', FileType::class, [
39                 'required' => false,
40                 'constraints' => [
41                     new Image(),
42                 ]
43             )
44             ->add('url', UriType::class, [
45                 'required' => false,
46                 'constraints' => [
47                     new Length(['max' => 255]),
48                 ]
49             )
50             ->add('category', EntityType::class, [
51                 'class' => Category::class,
52                 'choice_label' => 'name',
53                 'constraints' => [
54                     new NotBlank(),
55                 ]
56             )
57     }
```

# BUILD FORM (PARTIAL)

```
24 class JobType extends AbstractType
25 {
26     /**
27      * {@inheritdoc}
28      */
29     public function buildForm(FormBuilderInterface $builder, array $options)
30     {
31         $builder
32             ->add('company', TextType::class, [
33                 'constraints' => [
34                     new NotBlank(),
35                     new Length(['max' => 255]),
36                 ]
37             )
38             ->add('logo', FileType::class, [
39                 'required' => false,
40                 'constraints' => [
41                     new Image(),
42                 ]
43             )
44             ->add('url', UriType::class, [
45                 'required' => false,
46                 'constraints' => [
47                     new Length(['max' => 255]),
48                 ]
49             )
50             ->add('category', EntityType::class, [
51                 'class' => Category::class,
52                 'choice_label' => 'name',
53                 'constraints' => [
54                     new NotBlank(),
55                 ]
56             )
57     }
```



# CONFIGURE FORM

```
24 class JobType extends AbstractType
25 {
26     /** {@inheritdoc} ... */
29     public function buildForm(FormBuilderInterface $builder, array $options){...}
120
121     /**
122      * {@inheritdoc}
123      */
124     public function configureOptions(OptionsResolver $resolver)
125     {
126         $resolver->setDefaults([
127             'data_class' => Job::class
128         ]);
129     }
130 }
```

# FILE UPLOADER SERVICE

```
3 namespace App\Service;
4
5 use Symfony\Component\HttpFoundation\File\UploadedFile;
6
7 class FileUploader
8 {
9     /** @var string */
10    private $targetDirectory;
11
12    /**
13     * @param string $targetDirectory
14     */
15    public function __construct(string $targetDirectory)
16    {
17        $this->targetDirectory = $targetDirectory;
18    }
19
20    /**
21     * @param UploadedFile $file
22     *
23     * @return string
24     */
25    public function upload(UploadedFile $file) : string
26    {
27        $fileName = \bin2hex(\random_bytes(10)) . '.' . $file->guessExtension();
28
29        $file->move($this->targetDirectory, $fileName);
30
31        return $fileName;
32    }
33 }
```



# FILE UPLOADER LISTENER

```
11 class JobUploadListener
12 {
13     /** @var FileUploader */
14     private $uploader;
15
16     /** @param FileUploader $uploader ...*/
19     public function __construct(FileUploader $uploader)
20     {
21         $this->uploader = $uploader;
22     }
23
24     /**
25      * @param LifecycleEventArgs $args
26      */
27     public function prePersist(LifecycleEventArgs $args)
28     {
29         $entity = $args->getEntity();
30
31         $this->uploadFile($entity);
32     }
33
34     /** @param PreUpdateEventArgs $args ...*/
37     public function preUpdate(PreUpdateEventArgs $args){...}
43
44     /**
45      * @param $entity
46      */
47     private function uploadFile($entity)
48     {
49         // upload only works for Job entities
50         if (!$entity instanceof Job) {
51             return;
52         }
53
54         $logoFile = $entity->getLogo();
55
56         // only upload new files
57         if ($logoFile instanceof UploadedFile) {
58             $fileName = $this->uploader->upload($logoFile);
59
60             $entity->setLogo($fileName);
61         }
62     }
```

# SERVICES CONFIGURATION (SERVICES.YAML)

```
parameters:
  locale: 'en'
  max_jobs_on_homepage: 10
  max_jobs_on_category: 20
  jobs_directory: '%kernel.project_dir%/public/uploads/jobs'

services:
  App\Service\FileUploader:
    arguments:
      $targetDirectory: '%jobs_directory%'

  App\EventListener\JobUploadListener:
    tags:
      - { name: doctrine.event_listener, event: prePersist }
      - { name: doctrine.event_listener, event: preUpdate }

  App\EventListener\JobTokenListener:
    tags:
      - { name: doctrine.event_listener, event: prePersist }
```

# HOMEWORK





# HOMEWORK





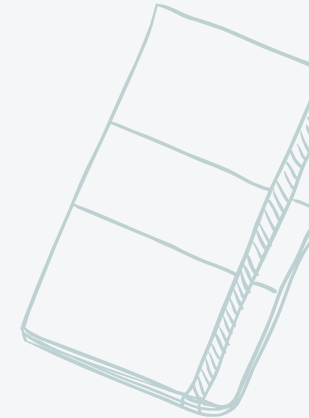
- Create an action for creating job action in controller
- Create JobType class where will be described how to build jobs form
- Create template for job creation page
- Handle job creation

## Optional:

- Handle file upload
- Prepopulate job token automatically



# LINKS

- 
- 
- 
- 
- 
- <https://symfony.com/doc/4.0/forms.html>
  - <https://symfony.com/doc/4.0/reference/forms/types.html>
  - <https://symfony.com/doc/4.0/reference/constraints.html>
  - [https://symfony.com/doc/4.0/reference/forms/twig\\_reference.html](https://symfony.com/doc/4.0/reference/forms/twig_reference.html)
  - [https://symfony.com/doc/4.0/form/form\\_customization.html](https://symfony.com/doc/4.0/form/form_customization.html)
  - [https://symfony.com/doc/4.0/doctrine/event\\_listeners\\_subscribers.html](https://symfony.com/doc/4.0/doctrine/event_listeners_subscribers.html)



QUESTIONS?







thank you!

