在《Symfony5全面开发》视频教程里，我们当我们使用make命令创建一些实现了某些接口的服务类时，Symfony会自动的为服务类添加tags标签。 比如：订阅器类。 这里可以使用`symfony console debug:container 订阅器类名`，查看具体的tags数据。

```
symfony console debug:container SimpleFileSubscriber

---------------- ------------------------------------------
  Option           Value
 ---------------- ------------------------------------------
  Service ID       App\EventSubscriber\SimpleFileSubscriber
  Class            App\EventSubscriber\SimpleFileSubscriber
  Tags             kernel.event_subscriber                    这是Symfony框
架自动添加的。
  Public           no
  Synthetic        no
  Lazy             no
  Shared           yes
  Abstract         no
  Autowired        yes
  Autoconfigured   yes
 ---------------- ------------------------------------------
```

我们想实现这个功能，现在我们bundle里有一些Namer命名器类，将来你可能定义自己的Namer类。只要实现`NamerInterface`接口，就自动加个`teebb.upload_namer`这个标签。

修改`Extension`类：

```
class TeebbUploadExtension extends Extension
{
    public function load(array $configs, ContainerBuilder $container)
    {
        ...
        $container->registerForAutoconfiguration(NamerInterface::class)-
>addTag('teebb.upload_namer');
    }
}
```

现在当容器进很编译时，会把容器中所有实现了`NamerInterface`接口的服务类添加tag。回到Teebblog项目，随意新增一个Namer类：

```
class RandomNamer implements NamerInterface
{
    public function rename(UploadedFile $file): string
    {
        return random_bytes(32) . '.' . $file-
>getClientOriginalExtension();
```

```
    }
}
```

在xml文件里定义的namer类需要手动添加tag标签：

```
<?xml version="1.0" encoding="utf-8" ?>
<container xmlns="http://symfony.com/schema/dic/services"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://symfony.com/schema/dic/services
http://symfony.com/schema/dic/services/services-1.0.xsd">
    <services>
        <service id="Teebb\UploadBundle\Namer\PhpNamer" public="true">
            <tag name="teebb.upload_namer"/>
        </service>
        <service id="teebb.upload.namer.php_namer"
alias="Teebb\UploadBundle\Namer\PhpNamer" public="true" />


        <service id="Teebb\UploadBundle\Namer\HashNamer" public="true">
            <tag name="teebb.upload_namer"/>
        </service>
        <service id="teebb.upload.namer.hash_namer"
alias="Teebb\UploadBundle\Namer\HashNamer" public="true"/>
    </services>
</container>
```

然后可以创建一个CompilerPass，获取整个Symfony项目里具有这个标签的服务类，进行处理。

```
class TeebbUploadNamerCompilerPass implements CompilerPassInterface
{
    public function process(ContainerBuilder $container)
    {
        dd($container->findTaggedServiceIds('teebb.upload_namer'));
    }
}
```

然后在Bundle类中添加CompilerPass

```
class TeebbUploadBundle extends Bundle
{
    public function build(ContainerBuilder $container)
    {
        parent::build($container);

        $mappings = [
            realpath(__DIR__.'/Resources/config/doctrine-mapping') =>
'Teebb\UploadBundle\Entity',
```

```
        ];

//        $namespaces = [
//            'Teebb\UploadBundle\Entity'
//        ];
//
//        $directories = [
//            realpath(__DIR__ . '/Entity')
//        ];

        if (class_exists(DoctrineOrmMappingsPass::class)) {
            $container->addCompilerPass(DoctrineOrmMappingsPass::createXmlMappingDriver($mappings));
//            $container->addCompilerPass(DoctrineOrmMappingsPass::createAnnotationMappingDriver($namespaces, $directories));
        }

        $container->addCompilerPass(new TwigFormThemeCompilerPass());

        $container->addCompilerPass(new TeebbUploadNamerCompilerPass());
    }
}
```

在项目中，`symfony console cache:clear`，查看数组，键是服务id,值是tag标签的属性，我们没有设置标签所以都是空数组。可以根据需要设置需要的数据，比如优先级、等等。

```
class TeebbUploadNamerCompilerPass implements CompilerPassInterface
{

    public function process(ContainerBuilder $container)
    {
        foreach ($container->findTaggedServiceIds('teebb.upload_namer') as $serviceId => $tagAttrs) {
            //伪代码，可以根据需要做处理。
            //这里只关心serviceId，可以获取服务类的Definition，然后执行addMethodCall。
//            $definition = $container->getDefinition($serviceId);
//            $definition->addMethodCall('xxxx', []);
        }
    }
}
```

这节课的内容与我们的Bundle项目无关，只是为了讲解一下自动配置。如果你要做一个插件系统，可以使用这种方式。

下节课，调整一下整个表单提交和文件移动的流程。