很有可能你的Bundle将来要进行持续集成，或者领导要求必须要写测试代码，这时你就不能为了测试Bundle安装一个Symfony项目。 我们可以可以Bundle里准备一个Symfony的内核，让内核启动准备容器，再从容器里获取服务类对象进行测试了。

安装phpunit

```
composer req phpunit --dev
```

第一次执行`php vendor/bin/simple-phpunit`会下载phpunit库

在bundle目录下测试一个tests目录，第一个单元测试类，

```
// tests/UnitTest/SimpleUnitTest.php
class SimpleUnitTest extends TestCase
{
    public function testHello()
    {
        $this->assertTrue(true);
    }
}
```

执行`php vendor/bin/simple-phpunit tests` 测试通过了。

修改namespace，tests目录对应的命名空间改为`Teebb\UploadBundle\Tests`，然后修改`composer.json`

```
{
    "name": "teebbstudios/upload-bundle",
    "description": "Symfony file upload form type bundle.",
    "type": "symfony-bundle",
    "license": "MIT",
    "autoload": {
        "psr-4": {
            "Teebb\\UploadBundle\\": "src/"
        }
    },
    "autoload-dev": {  //添加这个配置
        "psr-4": {
            "Teebb\\UploadBundle\\Tests\\": "tests/"
        }
    },
    "authors": [
        {
            "name": "Quan weiwei",
            "email": "qww.zone@gmail.com"
        }
    ],
```

```
    "require": {
        "php": "^7.2.5|^8.0",
        "symfony/form": "^5.3",
        "symfony/config": "^5.3",
        "symfony/orm-pack": "^2.2"
    },
    "config": {
        "allow-plugins": {
            "composer/package-versions-deprecated": true,
            "symfony/flex": false
        }
    },
    "require-dev": {
        "symfony/test-pack": "^1.0"
    }
}
```

从teebblog项目里复制php.xml.dist，添加phpunit配置文件：

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- https://phpunit.readthedocs.io/en/latest/configuration.html -->
<phpunit xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="vendor/phpunit/phpunit/phpunit.xsd"
         backupGlobals="false"
         colors="true"
         bootstrap="./vendor/autoload.php"
         convertDeprecationsToExceptions="false"
>
    <php>
        <ini name="error_reporting" value="-1" />
        <server name="APP_ENV" value="test" force="true" />
        <server name="SHELL_VERBOSITY" value="-1" />
        <server name="SYMFONY_PHPUNIT_REMOVE" value="" />
        <server name="SYMFONY_PHPUNIT_VERSION" value="9.5.14" />
    </php>

    <testsuites>
        <testsuite name="Project Test Suite">
            <directory>tests</directory>
        </testsuite>
    </testsuites>

    <coverage processUncoveredFiles="true">
        <include>
            <directory suffix=".php">src</directory>
        </include>
    </coverage>

    <listeners>
        <listener class="Symfony\Bridge\PhpUnit\SymfonyTestsListener" />
```

```
    </listeners>

    <!-- Run `composer require symfony/panther` before enabling this
extension -->
    <!-- 注释这里 -->
<!--    <extensions>-->
<!--        <extension class="Symfony\Component\Panther\ServerExtension"
/>-->
<!--    </extensions>-->

</phpunit>
```

修改bootstrap配置，直接执行php vendor/bin/simple-phpunit，可自动对tests目录进行测试。

现在我们要准备一个Symfony内核，然后进行集成测试

首先安装symfony的内核组件，查看Teebblog项目的Kernel所在的组件，只安装`symfony/http-kernel`组件就可以了。

我们安装`"symfony/http-kernel": "^5.0"` 5.x版本的的内核来进行测试。

在`tests`目录下创建Kernel类

```
class TeebbUploadBundleKernel extends Kernel
{
    public function registerBundles()
    {
       // 把我们的bundle添加到容器
        return [
            new TeebbUploadBundle(),
            //出现提示，所以我们要在容器里添加下面Bundle
            new DoctrineBundle(),
            new FrameworkBundle()
        ];
    }

    public function registerContainerConfiguration(LoaderInterface
$loader)
    {
        // TODO: Implement registerContainerConfiguration() method.
        //添加doctrine的bundle
        $loader->load(__DIR__.'/config/doctrine.yaml');
    }
}
```

再编写集成测试代码：

```
class IntegrationTest extends TestCase
{
    public function testIntegration()
    {
        $kernel = new TeebbUploadBundleKernel('test', true);
        $kernel->boot();

        $container = $kernel->getContainer();

        /**@var PhpNamer $namer * */
        $namer = $container->get('teebb.upload.namer.php_namer');

        //在集成测试目录里添加一个文件，并创建UploadedFile对象，进行测试
        $newName = $namer->rename(new UploadedFile(__DIR__ . '/hello.txt',
'hello'));
        $this->assertStringContainsString('hello-', $newName);
    }
}
```

最后测试就通过了。 最后我们可以把测试过的项目把布到https://packagist.org/。现在我们的Symfony Bundle开发视频教程部分就结束了。 下节课，我们来快速阅读一个开源的Symfony项目PrestaShop，来学习一下代码阅读的技巧。