我们使用Phpstorm打开我们的项目目录，展开项目目录文件夹。 Symfony项目其实也是composer项目，如果你新拿到一个Symfony项目， 你可以在控制台中使用`composer install`命令来安装这个项目所需要的依赖包。

现在我们看一下项目的目录结构。

bin目录下是console文件。如果你没有安装Symfony命令行工具，我们可以使用`php bin/console`来查看所有可用的命令。 在开发过程中，我们会大量的使用命令行工具来辅助开发。如果你安装了Symfony命令行工具，你可以使用`Symfony console`命令，来使用命令行工具。 它的底层也是调用了项目的console文件。

config目录是我们的配置文件目录，一些第三方的包包括一些项目的配置文件都在config目录中。通常使用`yaml`文件来进行配置。在Symfony中，yaml配置文件使用四个空格来进行排版。

public是我们Symfony项目的入口目录，index.php文件就在这里。如果我们项目有一些CSS或者JS资源，也存放在public目录中。它是可以公开访问的目录。

src目录是我们的源码目录，我们编写的所有代码都在src目录中。

var目录是缓存文件和日志文件目录。

vendor目录，当我们安装第三方的包或者Symfony组件时，所有的组件都会安装在vendor目录中。vendor目录中的文件不能修改。如果组件版本更新的话，修改的文件将会丢失。

我们来使用PHP内置的服务器来启动项目，我们可以使用Symfony命令行工具的`serve`命令来启动项目。在`http://127.0.0.1:8000`端口，我们就可以访问我们的项目了。

Symfony默认给了我们一个页面。我们没有编写任何一行代码，但是仍然可以访问这个页面。这是Symfony为我们提供的占位符页面。虽然它已经显示了，但这个页面仍然是404错误的页面。 我们刷新看一下错误代码，仍然是404。

下面我们将创建我们第一个页面。我们打开composer.json文件，Symfony为我们安装了`symfony/flex`这个组件。flex组件是Symfony的composer插件。使用flex我们很方便的就可以把Symfony的组件或者第三方的包，安装到Symfony中，并且进行自动化的配置。

Symfony中为了自动化的安装组件或者包并进行自动化的配置，提出了一个`recipe`的概念。它的中文翻译就是食谱。我们打开`symfony.sh`这个网站。在这里我们可以查看所有Symfony的组件和第三方的包。

当我们安装这个页面上的任意的包时，`recipe`就是这个包对应的配置文件。它就像食谱一样，指引这个包如何和我们的Symfony项目集成。 Symfony的flex组件将会自动下载对应的`recipe`配置文件，到我们的项目中，来完成这个包和我项目的集成。

我们想快速的开发我们第一个页面，我们搜索组件`maker`，`maker`组件下方一个`official官方`这个标签。代表这是官方提供的包，官方提供的包都可以使用别名来进行安装。 如果是社区贡献的包，它没有别名的概念。

回到项目，我们使用`composer req maker --dev`来安装这个包。包安装完成后，我们在composer.json文件中，我们看到`require-dev`下，maker-bundle已经安装成功了。

`require-dev`下的包仅用于开发环境，当我们在生产环境下部署项目时，我们可以使用`composer install --no-dev`来忽略掉dev下所有的bundle。

我们再次查看`symfony console`命令，这时它就给我们生成了`make`命令，所有的make命令都是由maker-bundle提供的。我们使用make命令来创建我们的第一个页面。

Symfony是MVC框架，MVC的C就是controller。当有请求访问的时候，controller负责读取请求的参数，然后根据请求参数来执行不同的数据操作，最后返回一个响应对象，然后响应对象发送到浏览器。

我们在控制台中输入`symfony console make:controller`命令，来创建controller。这里报错了，它提示我们缺少了一个包。我们复制这个命令安装一下。

再次使用`make:controller`命令，我们需要输入controller类名，这里输入`TestController`。命令行会在`src/Controller`目录中创建对应的controller类。

这一行是PHP8.0提供的原生注解。`Route`是个路由，当我们访问`/test`路径时，Symfony会调用controller的`index`方法，最终会返回json方法提供的数据。

我们按command键点击json（`windows系统按ctrl键`），最终响应的是JsonResponse响应对象。我们回到首页输入`/test`，它返回了一个json字串。这不是我们想要的结果。 我们需要它显示一个页面，回到项目手动创建一个`Response`对象，这里输入HTML代码。

```
    #[Route("/test", name: "test")]
    public function index(): Response
    {
        return new Response(<<<EOF
<html>
<head>
<title>这是我们的第一个页面</title>
</head>
<body>
<h1>Hello world</h1>
</body>
</html>
EOF
);
    }
```

回到页面刷新，我们第一个页面它显示了。我们看到我们的源代码，就是刚才自己写的代码。

在下一节我们将讲解一下HTTP协议和Symfony是如何处理这个`/test`请求的。