在上一节课中，我们通过在请求地址后面添加问号，添加name这样的方式来添加请求的参数。我们还有另外一种添加参数的方法，删除问号和后面的name参数，我们在/test后面添加斜杠，添加一个参数的值。

点击回车访问，出错了。它提示没有找到这个路由，我们修改12行路由注解的路径参数，在/test后面添加/weiwei。再次访问，现在可以了。但是如果我们想访问另外一个呢？比如说hello，这样就又不行了。

我们希望在路径中添加一个变量，这个变量可以用来接受任何的参数，这就是占位符的意义。占位符我们使用大括号中间加上占位符的名称这种写法。

我们再次访问页面，这就可以了，再输入其他的也可以。现在我们查看一下Request对象，我们回到浏览器，再次访问这个路径。

我们展开request属性，它的参数是空的。我们也展开query属性，它的参数是空的。我们看第一个属性attributes属性，展开，再展开，我们看到name参数(口误)它在attributes属性中。

占位符它并不是POST请求参数，也不是query请求参数，他是Symfony路由的参数，我们叫作路由参数。回到代码，我们可以使用$request，attributes属性，get方法来获取到这个name的值。

```
$name = $request->attributes->get('name');
```

再次刷新，这就显示了，比如说我们输入weiwei，他也就显示了，这样写有些麻烦。在action方法中，我们还有另外一种更简单的方法来获取到这个占位符的值。我们可以直接在action方法的参数中，注入占位符这个变量，注释掉第15行代码，再次刷新页面，它仍然显示了，很神奇！

怎么做到的呢？回想上一节，在HttpKernel类的handleRaw方法中，会处理index方法的所有参数。

我们双击shift键，输入HttpKernel，找到handleRaw方法，在149行获取参数这里下断点，打开xdebug监听。

刷新页面，断点断到了这里，我们进入getArguments方法。我们在56行下个断点，针对不同的参数类型，它会使用不同的解析器来处理参数。我们执行代码，第一次$resolver它使用了RequestValueResolver来处理我们action方法的第一个参数Request对象。现在我们不关心第一个参数，我们关心的是第二个参数，再次执行代码，这是第二次断点，我们再看$resolver变量，这次$resolver变量，它是RequestAttributeValueResolver的对象。

我们找到这个类，查看他的代码，在Controller文件夹中找到ArgumentResolver文件夹，打开RequestAttributeValueResolver这个类。它首先会执行supports方法，Symfony根据我们的参数自动生成Metadata数据，我们的name属性它不是一个变量，并且在请求对象的attributes属性中有这个name属性，所以他会继续往下执行。

在resolve方法中，请求对象会获取到我们的name属性，和我们代码写法是一样的。这样name参数就通过这种方式注入到了我们controller的index方法中。我们可以直接使用name参数。

我们整理一下流程，当我们访问带占位符的路径时，Symfony会将占位符的变量存入到请求的attributes属性中，然后在resolver中，获取attributes，所以我们注入的参数名称要和占位符的名称是一致的。如果写错的话就会出错。

另外我们可以对参数传递默认值，回到页面，我们删掉最后的hello占位符。默认值显示了，这就是Symfony占位符的使用，

我们已经学习了如何在action方法中获取到请求参数，在下一节中，我们将安装Symfony开发插件，另外使用docker安装mysql容器，使用mysql作为我们的开发数据库。