SpringMVC中的@Controller和@RequestMapping到底什么鬼？

1.1 @Controller是什么
首先看个例子：

```
@Controller
@RequestMapping("/blog")
public class BlogController {

    @RequestMapping("/index")
    public ModelAndView index(HttpServletRequest request){
        ModelAndView mav = new ModelAndView("/index");
        String ctx = request.getContextPath();
        System.out.println(ctx);
        mav.addObject("ctx", ctx);
        return mav;
    }

}
```

@Controller表示在tomcat启动的时候，把这个类作为一个控制器加载到Spring的Bean工厂，如果不加，就是一个普通的类，和Spring没有半毛钱关系。

以下是两个常见的配置：

```
<!-- 开启注解模式驱动 -->

<mvc:annotation-driven></mvc:annotation-driven>

<!-- 扫包 -->

<context:component-scan base-package=*"com.blogMgr.*"*></context:component-scan>
```

其中，base-package表示会扫描com.blogMgr目录下所有的包，一旦发现有个类上面加了类似于@Controller的注解，在容器启动的时候系统就会把它加载到Spring的Bean工厂，并且对其实例化。

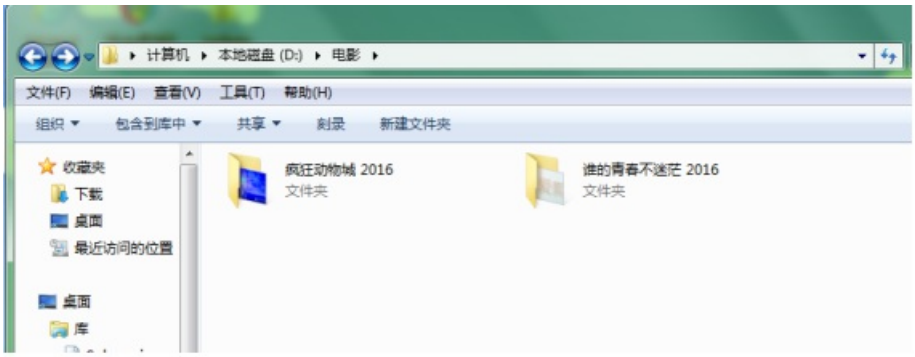
这也是为什么，我们只是写了Controller，但是从来没有在一个地方new这个Controller的原因，因为在Web容器启动的时候，这个Controller已经被Spring加载到自己的Bean工厂里面去了。

这也就是所谓的Spring扫包机制。@Controller就是一个注解，当tomcat启动，我们会看到一些JAVA类挥舞着印有@Controller的旗子大喊："Hey，SpringMVC，I'm here,please take me to your bean factory!"

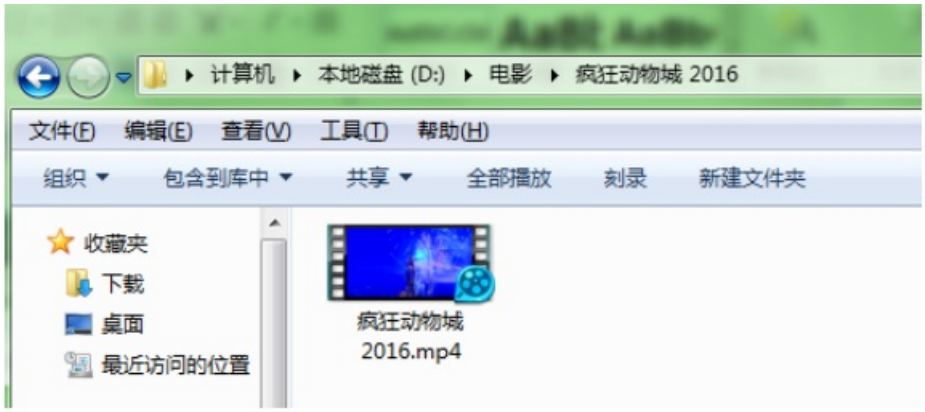
1.2 @RequestMapping是什么

在Controller中，总是会看到RequestMapping这个注解，看起来像是路径的跳转，以下列举了一个方便我们记忆的比喻。

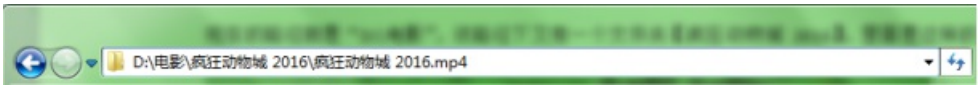
比如，有一天，我发现一部电影挺好看的，就在D盘建了一个文件夹，叫“电影”。里面放了两部电影，各自用一个文件夹来存放。



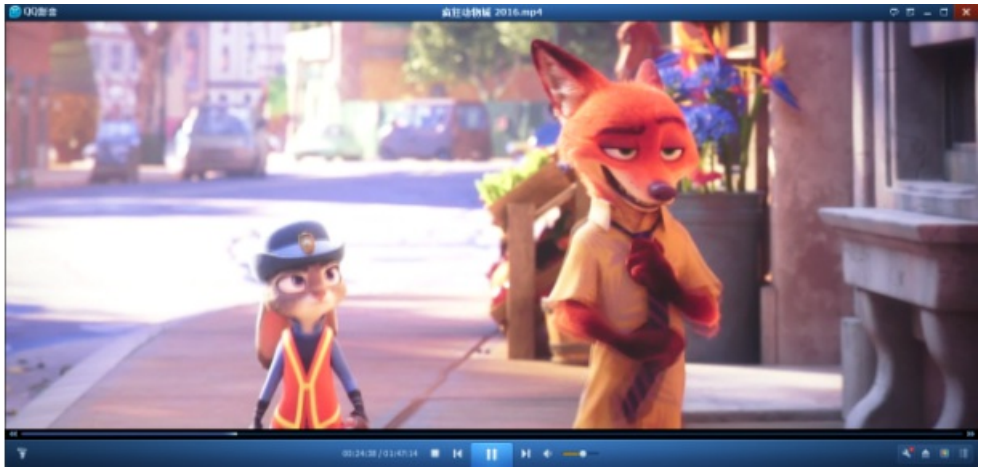
在上图中，我们可以看它的路径是“D:\电影”，该路径下又有一个文件夹【疯狂动物城 2016】，里面是这样的



那么，该文件的具体路径就是“D:\电影\疯狂动物城 2016”，现在我要访问这个资源，除了双击之外，是不是只需要在地址栏里面输入：**“D:\电影\疯狂动物城 2016\疯狂动物城 2016.mp4”**也可以呢？



是的，当然可以。



成功了，我们通过url的方式得到了我们想要的资源文件！

现在我们把这个文件复制一份，拷贝到相同路径下

如果我尝试将第一个MP4文件的名字也改为“疯狂动物城 2016.mp4”,则会弹出提示如下

可见，在同一个路径下，不能有两个重名的文件。

同理，如果我在同一个Controller里面设置两个相同的RequestMapping

```
@Controller
@RequestMapping("/blog")
public class BlogController {

    @RequestMapping("/index")
    public ModelAndView index(HttpServletRequest request){
        ModelAndView mav = new ModelAndView("/index");
        String ctx = request.getContextPath();
        System.out.println(ctx);
        mav.addObject("ctx", ctx);
        return mav;
    }
    @RequestMapping("/index")
    public ModelAndView index2(HttpServletRequest request){
        ModelAndView mav = new ModelAndView("/index");
        String ctx = request.getContextPath();
        System.out.println(ctx);
        mav.addObject("ctx", ctx);
        return mav;
    }
}
```

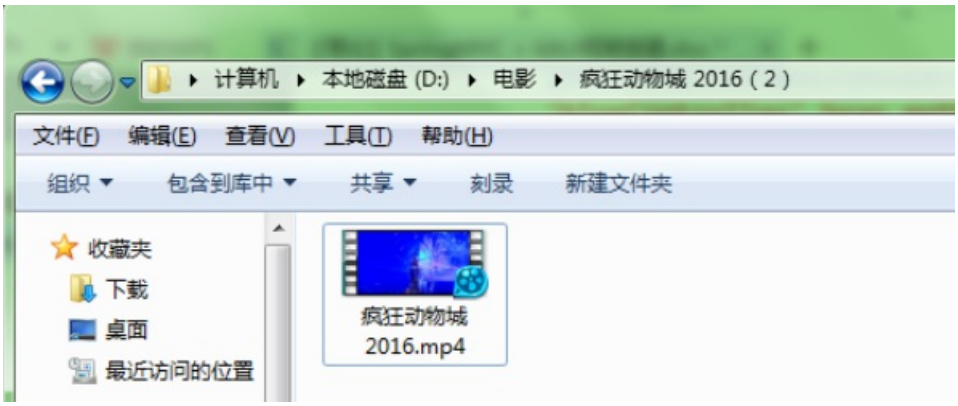
编译是可以通过的，但是当我启动tomcat后，

我通过url “http://localhost:8088/BlogMgr/blog/index”来访问，就会报错：

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping#0': Invocation of init method failed; nested exception is java.lang.IllegalStateException: Ambiguous mapping found. Cannot map 'blogController' bean method

Ambiguous mapping found表示有一个路径模棱两可，也就是路径重名了，系统无法进行映射，所以报错了。这个和无法在同一个文件夹下面创建两个同名文件，是一个道理。

那么，如果我一定要再创建一个“疯狂动物城 2016.mp4”的文件该怎么办呢



然后，我把这个文件放在另一个文件夹里面就可以了。

同理，我也可以另外建一个Controller，Controller类上面的RequestMapping叫“/blog2”,再在里面写一个一模一样的RequestMapping就肯定没有问题了。

小结

一个web项目本身就是一个系统，和操作系统一样，可以把项目当做一个系统，一个应用程序。人为什么要使用电脑，因为电脑可以

1.给我们想要的资源（比如.avi）

2.帮我们做事。

在一个系统中，如果没有图形界面，我们要访问一个资源，必然是通过一个黑窗口来访问的，就是通过路径来访问的。**一个B/S架构的web项目，就是一个类似于命令行一样的应用程序，我们唯有通过url，也就是路径去获得我们想要的资源和服务。**

再来看RequestMapping，最终对应的必然是一个**方法**，方法的功能无非就是进行一些业务的操作，或者返回一个什么东西。

比如

```java
@RequestMapping("/index")
public ModelAndView index(HttpServletRequest request){
    ModelAndView mav = new ModelAndView("/index");
    String ctx = request.getContextPath();
    System.out.println(ctx);
    mav.addObject("ctx", ctx);
    return mav;
}
```

我们就是通过这个方法获得了想要的jsp页面，RequestMapping的作用就是提供了一个句柄，让我们可以访问到对应的**方法**，最终获得我们想要的东西。综上所述，RequestMapping就是一个映射路径。

1.3 @ResponseBody的作用

在Controller里面，我们经常可以看到@ResponseBody这个注解，它的意思很简单，就是说明这个方法返回的东西会通过IO流的方式写入到浏览器。

比如我们写一个方法：

```
@RequestMapping("/testResponseBody")
@ResponseBody
public String testResponseBody(HttpServletRequest request){
    return "<h1 style='color:lightGreen'>Hello Web!</h1>";
}
```

最终在浏览器获得的效果是这样的：