springmvc-offcn

1.动态路径请求问题

```
<servlet-mapping>
    <!--理论上 [/] 这玩意不会拦截jsp资源 但会拦截静态资源以及, 也不会拦截其他的动态映射请求-->
    <!--理论上 [/*] 这玩意不会拦截动态映射请求，但会拦截静态资源以及jsp请求-->
    <servlet-name>dispatchServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

```
06-May-2022 15:52:42.839 警告 [http-nio-6060-exec-7] org.springframework.web.servlet.DispatcherServlet.noHandlerFound
 No mapping for GET /test.jsp
06-May-2022 15:53:07.818 警告 [http-nio-6060-exec-10] org.springframework.web.servlet.DispatcherServlet.noHandlerFound
  No mapping for GET /test.html
06-May-2022 15:53:27.603 警告 [http-nio-6060-exec-2] org.springframework.web.servlet.DispatcherServlet.noHandlerFound
 No mapping for GET /main.jsp
null
null
06-May-2022 15:53:38.827 警告 [http-nio-6060-exec-4] org.springframework.web.servlet.DispatcherServlet.noHandlerFound
 No mapping for GET /main.jsp
```

1 /*访问到了动态路径,但项目路径下的资源均不可访问

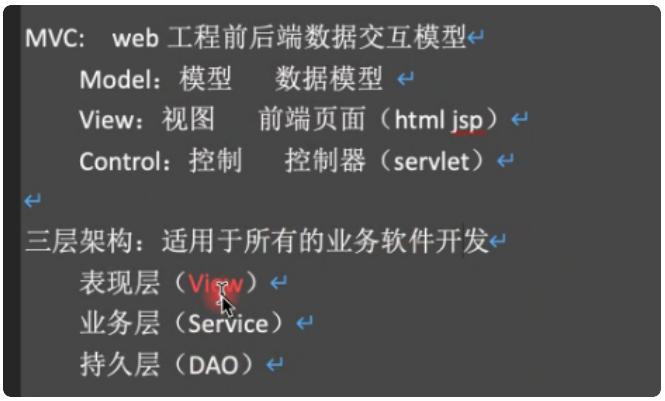
2 转发也不行

注意我们一般使用 /这玩意

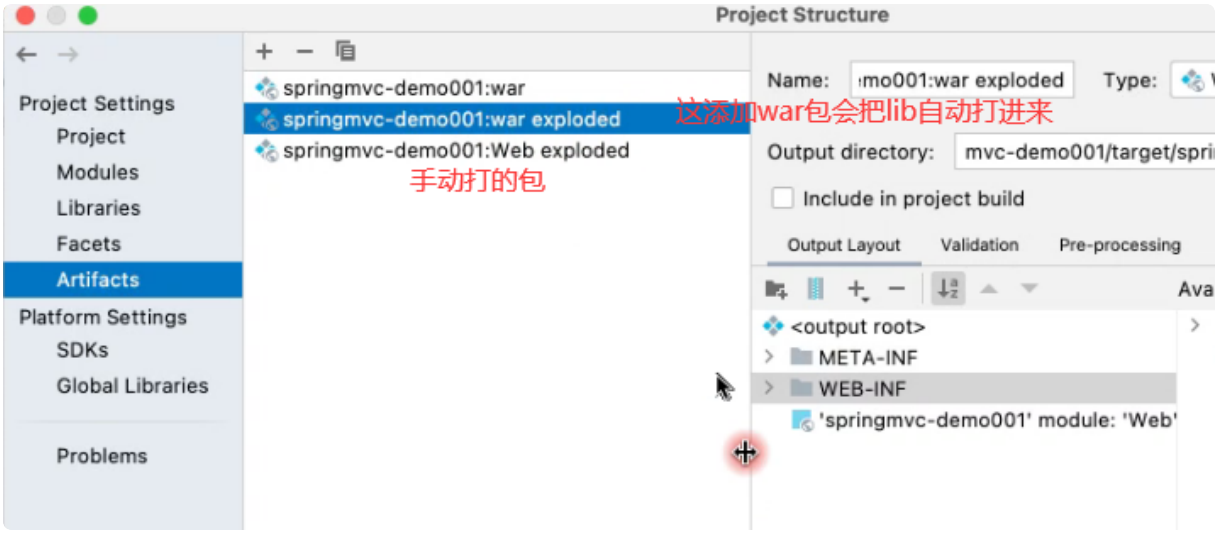
2.springmvc存值机制迷惑点

- modelandview也可以存值

3.三层架构和mvc模式的区别



4.打包动作



我个人不建议这么配置，还是用idea工具打的包

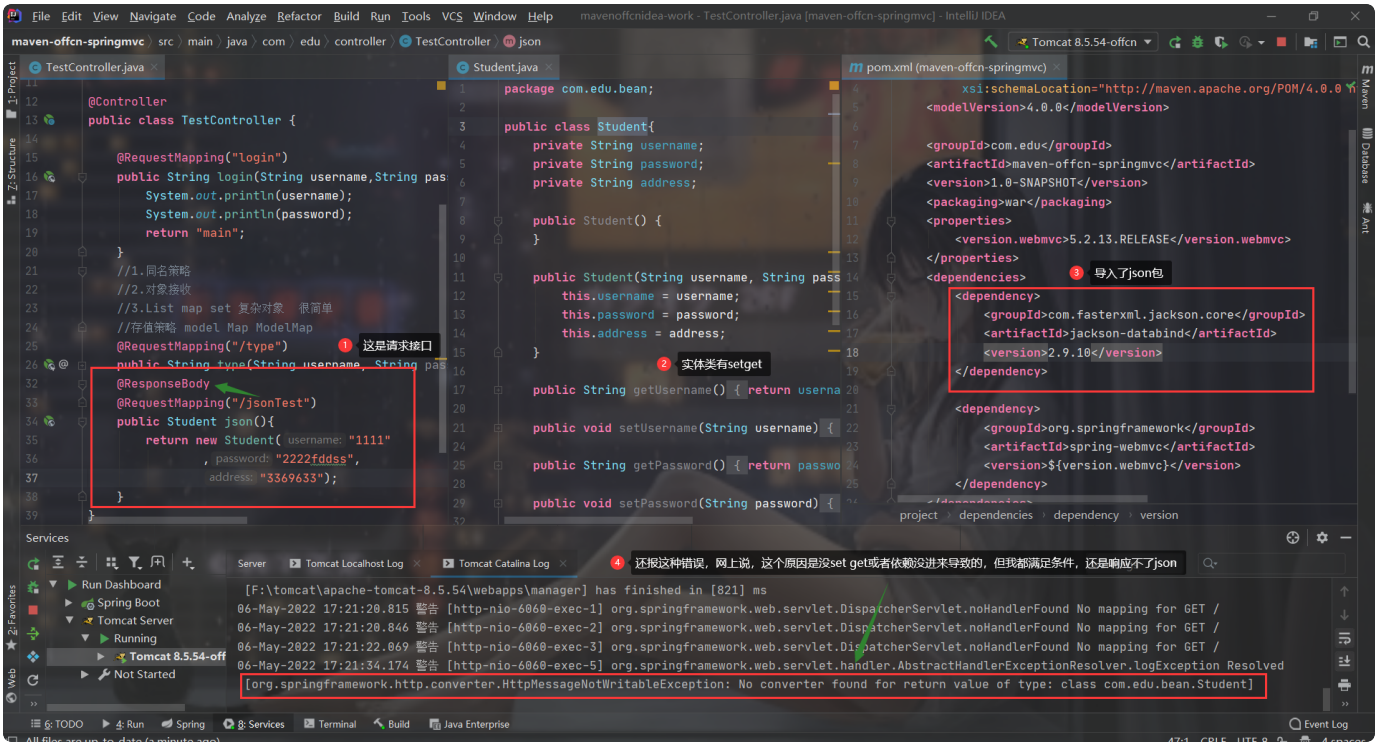
5.springmvc的一个请求流程

- 注意只用做到心中有数，配置变的格外简单

client发送一个请求--->dispatchServlet ----->HandlerMapper处理请求-------->dispatchServlet一个拦截器链----->HandlerAdpter适配器，去匹配handler---->handler----->HandlerAdpter------>dispatchServlet---------->找一个modelandview 视图解析器---->返回给dispatchServlet------>model -------->响应

- 上面看出配置必须有这几个才能处理请求
  dispatchServlet,HandlerMapper,HandlerAdpter,modelandview

6.错误迷惑点响应json串



错误原因是没有加注解驱动；注解驱动可以替代两个handler

🐂超说：没加注解驱动这三个地方会受到影响 Java 复制代码

```
1.是配置view-controller
2.是配置访问默认的静态资源，
3.是处理java对象转化为json对象的时候
```

7.参数请求接收

apipost: https://wiki.apipost.cn/document/00091641-1e36-490d-9caf-3e47cd38bcde/0d171147-ce26-4b26-bdfa-7388bf91f0d8

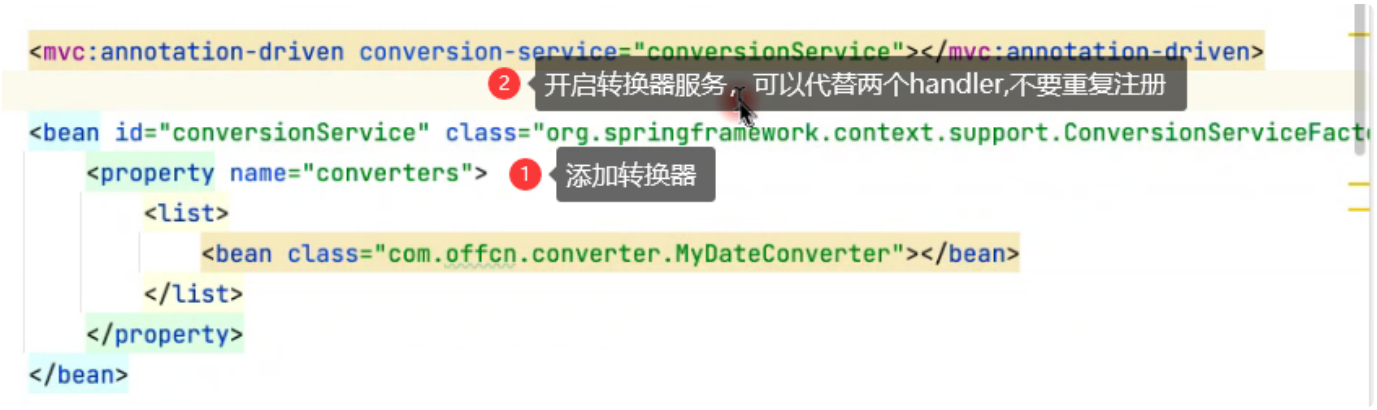
代码块 Java 复制代码

```java
@RequestMapping("/paramsEquals")
public String paramsEquals(String username,String password);
@RequestMapping("/paramsObject")
public String paramsObject(Student student);
@RequestMapping("/paramsObjects")
public String paramsObjects(Student student, Car car);
//car为student属性，前端传值:car.name
@RequestMapping("/paramsStuAndCar")
public String paramsStuAndCar(Student student);
//carList为student属性，前端传值：carList[0].name
@RequestMapping("/paramsStuAndCarList")
public String paramsStuAndCarList(Student student);
//carSet为student属性，前端传值：carSet[0].name-----前提是后端先初始化好set并存放好对象
@RequestMapping("/paramsStuAndCarSet")
public String paramsStuAndCarSet(Student student);
//map为student属性，前端传值 map["aa"].name
@RequestMapping("/paramsStuAndMap")
public String paramsStuAndMap(Student student);
//注解式
@RequestMapping(path = "/params");
public String params(
     @RequestParam(name = "username") String name
    ,@RequestParam(name = "password") String pwd
    );
//map接收所有
@RequestMapping(path = "/paramsMap")
public String paramsMap(@RequestParam Map<String,String> map);
//路径请求
@GetMapping("/pathParams/{id}/{name}");
public String pathParams(
        @PathVariable(name = "id",required = true) String id,
        @PathVariable("name") String name
        );
//路径map
@GetMapping("/pathParamsMap/{id}/{name}")
    public String pathParamsMap(@PathVariable Map<String,String> map);
//json格式
@PostMapping("/requestBodyJson")
    public String requestBodyJson(@RequestBody Student student);
//二进制格式，但需要配置上传解析器
@PostMapping("/upload")
    public String requestBodyJson(MultipartFile source);

```

8.自定义数据格式转换器

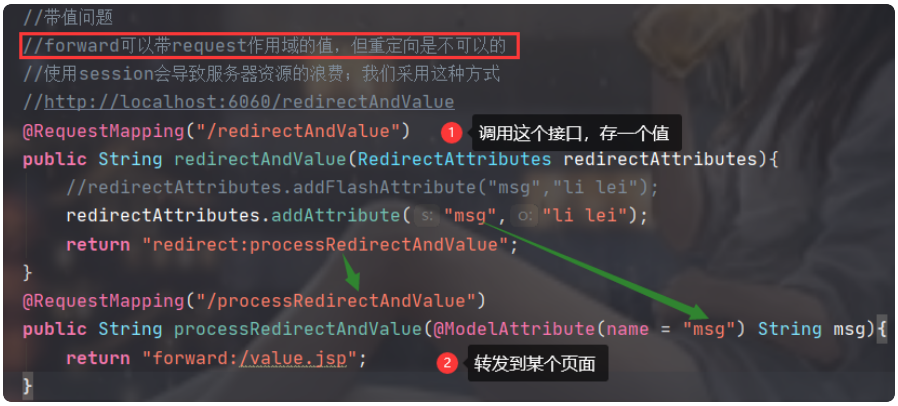
前端传过来都是字符串类型，我想转成日期类型，需要定义格式转换器，当然要在配置中注册格式转换器服务，并开启服务



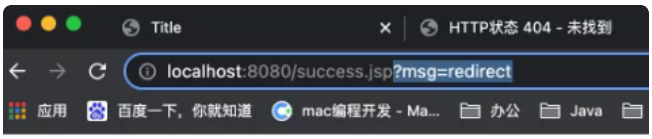
异常解析器handler handlermapper handlerAdpate

```java
public class MyConverter implements Converter<String, Date> {
    public Date convert(String s) {
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd");
        Date date = null;
        try {
            date = simpleDateFormat.parse(s);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return date;
    }
}
```

9.返回值类型注意点



```java
@RequestMapping("redirect")
public String redirect(RedirectAttributes redirectAttributes){
    redirectAttributes.addAttribute( s: "msg", o: "");
    // 重定向地址不再经过视图解析器
    return "redirect:/success.jsp";
}
```



success

10.mvc上传和下载于原生的区别

CommonsMultipartResolver

下载 Java 复制代码

```java
    //下载功能
    @RequestMapping("/download")
    public ResponseEntity<byte[]> download(String filename) throws Exception {
        String path = "E:\\photo\\" + filename;
        File file = new File(path);
        //设置响应头，而原生是直接response设置
        HttpHeaders httpHeaders = new HttpHeaders();
        httpHeaders.setContentDispositionFormData("attachment",
                new String(filename.getBytes("utf-8"),"ISO-8859-1"));
        //application_octet_stream
        httpHeaders.setContentType(MediaType.APPLICATION_OCTET_STREAM);

        return new ResponseEntity<>(FileUtils.readFileToByteArray(file), httpHeaders, HttpStatus.OK);
    }
```

注意如何设置响应头，如何响应前端responseEntity

```java
//下载功能二
@RequestMapping("/download2")
//http://localhost:6060/download2
public ResponseEntity<byte[]> download2(String filename, HttpSession session) throws Exception {
    ServletContext servletContext = session.getServletContext();
    String realPath = servletContext.getRealPath( s: "/image/"+filename);

    String mimeType = servletContext.getMimeType(realPath);
    File file = new File(realPath);
    //设置响应头
    HttpHeaders httpHeaders = new HttpHeaders();
    httpHeaders.set("Content-Type", mimeType);
    httpHeaders.setContentDispositionFormData( name: "attachment", URLEncoder.encode( s: "图片.jpg",  enc: "utf-8"));
    return new ResponseEntity<>(FileUtils.readFileToByteArray(file), httpHeaders, HttpStatus.OK);
}
```

11.异常解析器

注解驱动已经注入过了异常解析器的handler,无需我们自己在此注入handler

```java
@ControllerAdvice
public class MyThrowableException {
    @ExceptionHandler(value = {NullPointerException.class})
    public String nullException(){
        System.out.println("nullException");
        return "jpg";
    }
    @ExceptionHandler(value = {Throwable.class})
    public String defaultException(){
        System.out.println("defaultException");
        return "jpg";
    }
}
```

1 定义全局异常

2 与注解驱动注入的handler有关

3 先走精准异常，在走默认全局异常

12.拦截器(把控执行时机)

注意拦截器的执行顺序：可以配置多个拦截器，在配置文件中，谁在上，谁先执行。

思考：拦截器为什么不拦截jsp?

我们的tomcat开源项目中配置的加载规则，是用户项目优于，tomcat项目，用户配置了，加载用户的，用户没配置加载tomcat中web.xml的配置，所以*.jsp的请求走的是defalut的servlet,而对于拦截器而言，只拦截----------dispatchServlet接收的所有请求。