@RequestMapping("/index")

@Controller

**public** **class** LoginController {

确认了是spring框架

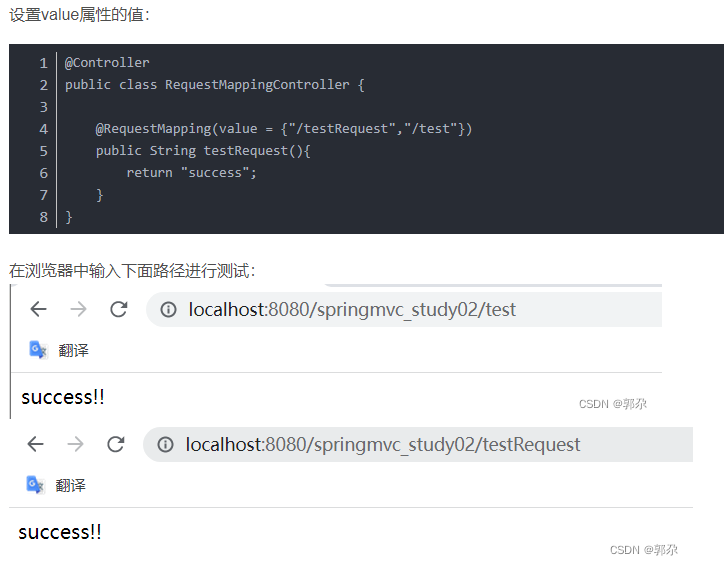
@RequestMapping("/index")

通过 RequestMapping 注解来处理这些映射请求，也就是通过它来指定控制器可以处理哪些URL请求。

用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

@RequestMapping的value属性

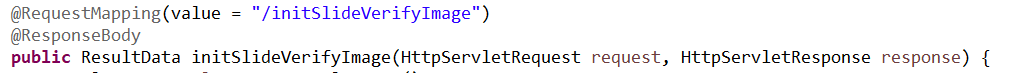
通过当前请求的请求地址来匹配请求



——【参考1】

即LoginController类中相应请求的方法以/index为父路径。

如下方的initSlideVerifyImage方法路径为/index/initSlideVerifyImage

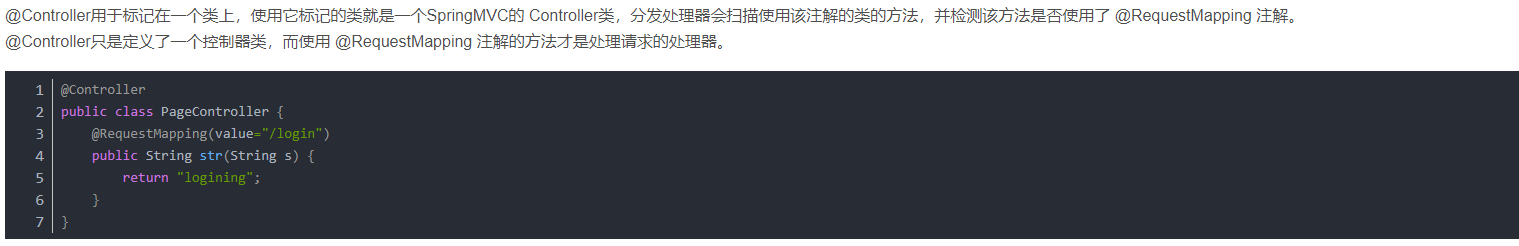


@Controller

用@Controller定义一个控制器类，

用@RequestMapping给出外界访问方法的路径，或者说触发路径 ，触发条件。

用@ResponseBody标记Controller类中的方法。把return的结果变成JSON对象返回。（如果没有这个注解，这个方法只能返回要跳转的路径即跳转的html/JSP页面。有这个注解，可以不跳转页面，只返回JSON数据）



——【参考2】

# initSlideVerifyImage()方法

**public** ResultData initSlideVerifyImage(HttpServletRequest request, HttpServletResponse response)

File templateFile = getSlideFile("templates","-w.png");

File targetFile = getSlideFile("targets",".jpg");

**private** File getSlideFile(String fileType,String suffix)

从静态资源中选取图片。

fileType决定文件路径filePath和选取范围picNo

filePath+picNo+suffix即为完整路径

eg. “static/targets/ ” + “1” + ”.jpg”

slideVerifyInfo = toolkit.pictureTemplatesCut(templateFile,targetFile,"png", "jpg");

将背景图切出一块作为滑块，被抠图区域透明化处理，返回两张图片和相关信息。

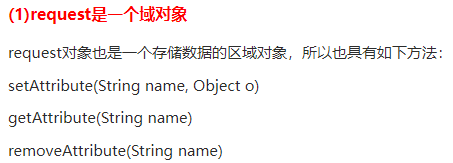
详见SlideVerifyToolkitjava相关.docx

String pointX = slideVerifyInfo.get("pointX");

**if**(request!=**null**) {

request.getSession().setAttribute(com.google.code.kaptcha.Constants.***KAPTCHA\_SESSION\_KEY***,pointX);

}



——【参考3】

这里可以将request理解为一个key-value结构，将缺口的x坐标对应KAPTCHA\_SESSION\_KEY

if(request!=null)是判断该request是否是正常请求。

如果是正常请求，择request非空。

如果是验证失败后自动刷新，即initSlideVerifyImage(null,null);则无request信息。

详见下文checkValidateCode()方法

result.setResult(slideVerifyInfo);

result.setSuccess(**true**);

**return** result;

slideVerifyInfo 中保存着滑块图片，详见SlideVerifyToolkitjava相关.docx

将其存入result中，标记为success并返回，详见ResultData.java相关.docx

# checkValidateCode()方法

**public** ResultData checkValidateCode(HttpServletRequest request)

**if** (!isAllowCheck(ip)) {

result.setNotice("受限IP，拒绝服务！");

**return** result;

}

private boolean isAllowCheck(String ip)

isAllowCheck()方法用于校验ip是否为恶意攻击，逻辑如下：

如果ip首次发起请求，通过；

如果ip距离上次请求超过5分钟，通过；

如果ip距离上次请求不超过5分钟，但5分钟内请求不超过20次，通过。

如果ip5分钟内请求超过20次，不予通过。

**double** maxRange = imagePointX + errLimit;

**double** minRange = imagePointX - errLimit;

**if**(realPointX>minRange && realPointX<maxRange){

*logger*.debug("滑块在范围内，验证OK");

result.setSuccess(**true**);

result.setResult(checkOk);

}**else** {

// 设置结果

result.setNotice("验证码错误！");

}

}

imagePointX为缺口横坐标，realPointX为滑块横坐标，errLimit为误差范围。

当滑块在误差范围内时，验证成功。

**if**(!result.isSuccess()){

putCheckErrorLog(ip);

*logger*.info("防止暴力破解重新生成验证码");

initSlideVerifyImage(**null**,**null**);

}

如果验证失败将自动刷新验证码并统计ip失败次数。

详见上文isAllowCheck()方法。

login()方法和logout()方法

**public** ResultData login(HttpServletRequest request)

**public** ResultData logout(HttpServletRequest request)

login方法会依次校验验证码，用户名，密码。

都校验成功后会清空验证码并存储会话信息。

logout方法用于结束会话后清理信息。参考1：[@RequestMapping详解](https://blog.csdn.net/m0_67401055/article/details/125057030)

参考2：[@Controller注解](https://blog.csdn.net/qq_46046423/article/details/109336282)

参考3：[HttpServletRequest](https://www.cnblogs.com/Kingram/p/9205482.html)