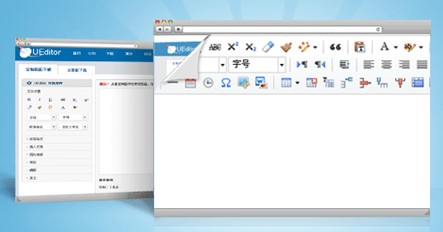
**[springmvc学习笔记--ueditor和springmvc的集成](http://www.cnblogs.com/mumuxinfei/p/5363909.html)**

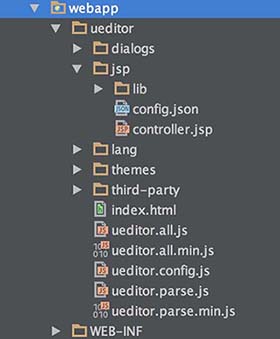
***前言:***  
　　在web开发中, 富文本的编辑器真心很重要. 有电商店铺的打理, 新闻稿/博客文章/论坛帖子的编辑等等, 这种所见即所的编辑方式, 大大方便了非技术人员从事互利网相关的工作.  
　　因为手头有个小项目, 正好涉及到这块, 所以决心好好研究一番. 中间也在ckeditor和ueditor之间徘徊了很久, 后来听闻大名鼎鼎的微信公众号也使用了ueditor, 因此最后倒向了ueditor.  
　　本文将讲解如何集成springmvc+ueditor的一些要点, 并做下简单的展望.

***ueditor简介:***  
 UEditor是由百度web前端研发部开发所见即所得富文本web编辑器，具有轻量，可定制，注重用户体验等特点，开源基于MIT协议，允许自由使用和修改代码...  
 ueditor的[官网地址](http://ueditor.baidu.com/website/index.html" \t "_blank)如下: http://ueditor.baidu.com/website/index.html.



由于涉及到图片上传的部分, ueditor并非纯javascript版本, 其出了php, java, asp等语言的支持版本.  
 源码的[下载链接](http://ueditor.baidu.com/website/download.html" \t "_blank): http://ueditor.baidu.com/website/download.html.

***集成:***  
 ueditor的java版提供的是jsp版本的, 还附带了多个java依赖包. 以下是它的目录结构.



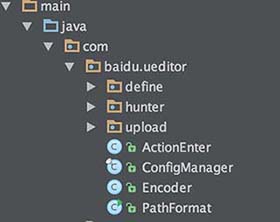
配置文件config.json, 定义了支持的上传文件/图片的接口, 以及限制, 这个很重要.  
 而当前的web开发, 往往都是基于maven来组织构建web工程的. 同时由于springmvc版本的框架限制. 还是得有一方进行妥协, 或者说是修改.  
 在这个的前提下, 我们来进一步的地细化集成的工作.  
 • **依赖包采用maven的方式来组织构建**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <!-- 上传文件的支持 -->  <dependency>      <groupId>commons-fileupload</groupId>      <artifactId>commons-fileupload</artifactId>      <version>1.3.1</version>  </dependency>    <dependency>      <groupId>commons-io</groupId>      <artifactId>commons-io</artifactId>      <version>2.4</version>  </dependency>    <dependency>      <groupId>commons-codec</groupId>      <artifactId>commons-codec</artifactId>      <version>1.10</version>  </dependency>    <!-- org.json -->  <!--JSON is a light-weight, language independent, data interchange format. See http://www.JSON.org/-->  <dependency>      <groupId>org.json</groupId>      <artifactId>json</artifactId>      <version>20160212</version>  </dependency> |

这样就引入了文件上传, 以及json序列化/反序列化处理的依赖包, ueditor-x.x.x.jar后面介绍.  
 • **资源访问配置**  
 单独把ueditor放入到webapp目录下, 因此在springmvc映射处理中需要排除.

|  |  |
| --- | --- |
| 1 | <mvc:resources mapping="/ueditor/\*\*" location="/ueditor/" /> |

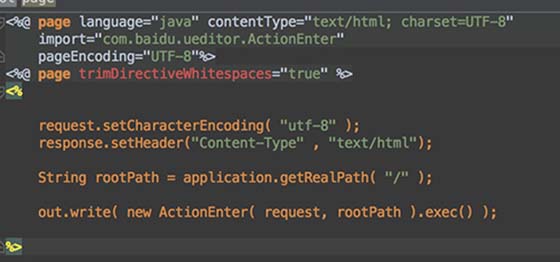
• **引入ueditor-x.x.x的源码**  
 其实作为依赖包引入也可以, 只是引入源码, 方便功能的修改和增强. 和前一种方法相比, 我更加推荐后者, 即引入源码.



• **添加controller处理类**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | @Controller  @RequestMapping("/ued")  public class UEditorController {        @RequestMapping(value="/config")      public void config(HttpServletRequest request, HttpServletResponse response) {            response.setContentType("application/json");          String rootPath = request.getSession()                  .getServletContext().getRealPath("/");            try {              String exec = new ActionEnter(request, rootPath).exec();              PrintWriter writer = response.getWriter();              writer.write(exec);              writer.flush();              writer.close();          } catch (IOException e) {              e.printStackTrace();          }        }  } |

该controller类照搬了ueditor中原本的controller.jsp的处理代码, 并改成springmvc的那种模式. 注意这边的处理ueditor操作的url path为: /ued/config. 它完成了编辑器初始化, 以及文件/图片的上传处理.



• **配置config.json文件**  
 config.json文件, 是定义文件/图片上传的接口, 以及各种限制(文件大小上限, MimeType类型框定).  
 具体配置的细节并不重要, 这边最重要的是: **放在那里, 以及如何被读取到?**  
 放的位置问题, 可以自由点, 让我们霸道一回, 就把config.json文件放置到webapp/conf目录下吧, ^\_^.  
 然后如何被读取到呢? 原先ueditor的代码, 默认指定了请求的Context Path. 原因还是在jsp版本中, 默认controller.jsp和config.json是同目录.

|  |  |
| --- | --- |
| 1  2  3 | private String getConfigPath () {      return this.parentPath + File.separator + ConfigManager.configFileName;  } |

然而现状已经发生变化了, **若不修改源码会导致找不到配置文件config.json的错误**. 因为寻找的路径和和我们设定的webapp/conf有出路.  
 在类ConfigManager修改后为:

|  |  |
| --- | --- |
| 1  2  3  4  5 | private String getConfigPath () {      return this.rootPath              + File.separator + "conf"              + File.separator + ConfigManager.configFileName;  } |

这样就可以了, 这也是为什么, 我推荐使用源码引入的原因, **这也是唯一一处必须需要修改的地方**.  
 • **修改ueditor.config.js**  
 这也是最后一个步骤了, 我们需要制定编辑器访问服务器的初始地址.

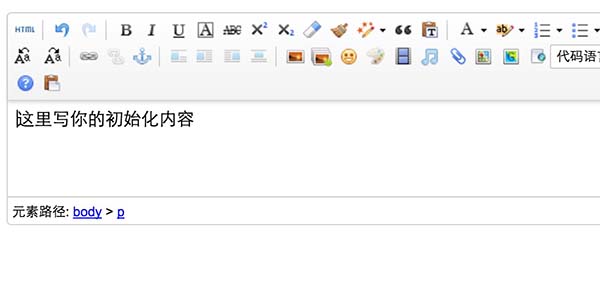
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | window.UEDITOR\_CONFIG = {        //为编辑器实例添加一个路径，这个不能被注释      UEDITOR\_HOME\_URL: URL        // 服务器统一请求接口路径      //      原先默认的      //, serverUrl: URL + "jsp/controller.jsp"      //      修改后的      , serverUrl: "/ued/config" |

注: **修改serverUrl, 使得其指向自己的服务器地址**.  
 这样整个ueditor和springmvc的集成配置工作就做完了, ^\_^. 感觉还是有点赶鸭子上架, 却没有具体讲述原理.

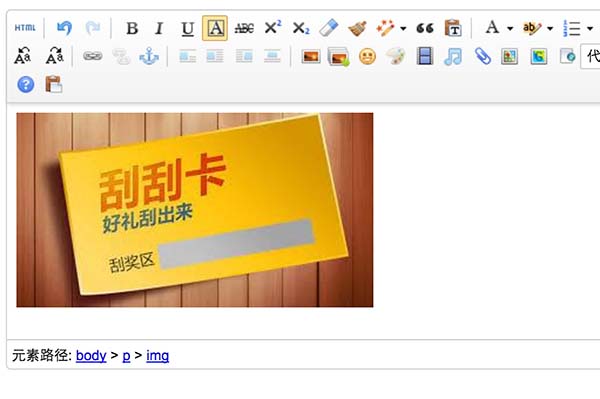
***验证:***  
 验证过程, 相对比较简单一些.  
 引入编辑框代码:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <script src="/ueditor/ueditor.config.js"></script>  <script src="/ueditor/ueditor.all.min.js"></script>  <script src="/ueditor/lang/zh-cn/zh-cn.js"></script>    <script id="container" name="content" type="text/plain">这里写你的初始化内容</script>  <script type="text/javascript">     var editor = UE.getEditor('container');  </script> |

初始化后, 编辑框如下:



上传图片操作后, 编辑框如下:



OK, 非常的成功.  
  
***后记:***  
 在查阅网上资料的时候, 也见过其他的集成方式的. 比如添加额外的servlet(path只匹配ueditor), 单独处理ueditor的jsp. 这样也能很好的做到集成, 也省心省力. 不过采用后者, 可以修改图片上传的模式, 比如放置到专门的图片服务器. 这也是服务做大的一个必经之路.  
 后续有机会将讲述下使用ueditor+图片服务器的修改思路, 以及ueditor的定制工作. 希望自己能努力, good good study, day day up.

/\*\*

\* spring 默认MultipartFile格式的文件，将ServletFileUpload进行强制转换

\* **@author** jiahh 2016年8月15日

\* 这个是使用baiduUEditor融合springMVC会遇到的一个问题

\*/

**public** **class** BinaryUploader {

**public** **static** **final** State save(HttpServletRequest request,

Map<String, Object> conf) {

// jiahh spring 默认MultipartFile格式的文件，将ServletFileUpload进行强制转换

StandardMultipartHttpServletRequest re = (StandardMultipartHttpServletRequest) request;

MultipartFile file = re.getFile("upfile");

FileItemStream fileStream = **null**;

**boolean** isAjaxUpload = request.getHeader( "X\_Requested\_With" ) != **null**;

**if** (!ServletFileUpload.*isMultipartContent*(request)) {

**return** **new** BaseState(**false**, AppInfo.***NOT\_MULTIPART\_CONTENT***);

}

// ServletFileUpload upload = new ServletFileUpload(

// new DiskFileItemFactory());

//

// if ( isAjaxUpload ) {

// upload.setHeaderEncoding( "UTF-8" );

// }

**try** {

// jiahh spring 默认MultipartFile格式的文件，将ServletFileUpload进行强制转换

// FileItemIterator iterator = upload.getItemIterator(request);

//

// while (iterator.hasNext()) {

// fileStream = iterator.next();

//

// if (!fileStream.isFormField())

// break;

// fileStream = null;

// }

// if (fileStream == null) {

// return new BaseState(false, AppInfo.NOTFOUND\_UPLOAD\_DATA);

// }

**if** (file.isEmpty()) {

**return** **new** BaseState(**false**, AppInfo.***NOTFOUND\_UPLOAD\_DATA***);

}

String savePath = (String) conf.get("savePath");

//String originFileName = fileStream.getName();

// jiahh spring 默认MultipartFile格式的文件，将ServletFileUpload进行强制转换

String originFileName = file.getOriginalFilename();

String suffix = FileType.*getSuffixByFilename*(originFileName);

originFileName = originFileName.substring(0,

originFileName.length() - suffix.length());

savePath = savePath + suffix;

**long** maxSize = ((Long) conf.get("maxSize")).longValue();

**if** (!*validType*(suffix, (String[]) conf.get("allowFiles"))) {

**return** **new** BaseState(**false**, AppInfo.***NOT\_ALLOW\_FILE\_TYPE***);

}

savePath = PathFormat.*parse*(savePath, originFileName);

String physicalPath = (String) conf.get("rootPath") + savePath;

//InputStream is = fileStream.openStream();

// jiahh spring 默认MultipartFile格式的文件，将ServletFileUpload进行强制转换

InputStream is = file.getInputStream();

State storageState = StorageManager.*saveFileByInputStream*(is,

physicalPath, maxSize);

is.close();

**if** (storageState.isSuccess()) {

storageState.putInfo("url", PathFormat.*format*(savePath));

storageState.putInfo("type", suffix);

storageState.putInfo("original", originFileName + suffix);

}

**return** storageState;

} **catch** (IOException e) {

**return** **new** BaseState(**false**, AppInfo.***IO\_ERROR***);

}

}

**private** **static** **boolean** validType(String type, String[] allowTypes) {

List<String> list = Arrays.*asList*(allowTypes);

**return** list.contains(type);

}

}