UploadAction(DCUG)

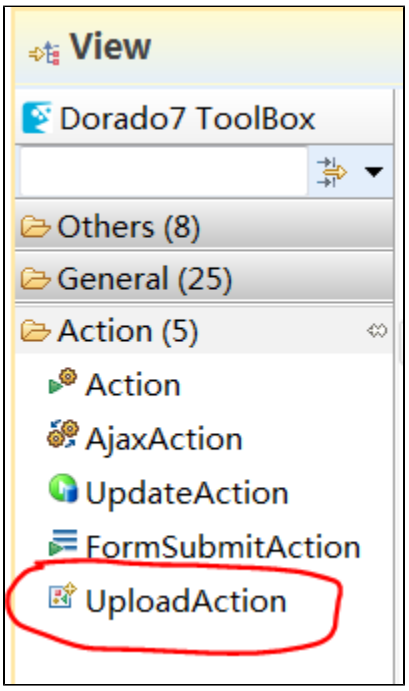
简介

项目主页

示例项目

UploadAction用于帮助开发人员做文件上传的一个控件，常见的操作界面如下：

IDE中控件的位置：



安装

UploadAction不是dorado-core提供的，是由Uploader项目提供的，因此你首先需要下载控件对应的jar，如Maven配置代码:

```
<dependency>
  <groupId>com.bstek.dorado</groupId>
  <artifactId>dorado-uploader</artifactId>
  <version>0.3.0-SNAPSHOT</version>
 </dependency>
```

下载好对应的jar，之后更新项目规则文件就可以在IDE的控件列表中看到这个UploadAction.

依赖包说明

主要依赖的第三方的jar:

- commons-fileupload-1.2.2.jar
- commons-io-2.0.jar
- jackson-databind-2.3.0.jar
- jackson-annotations-2.3.0.jar
- jackson-core-2.3.0.jar

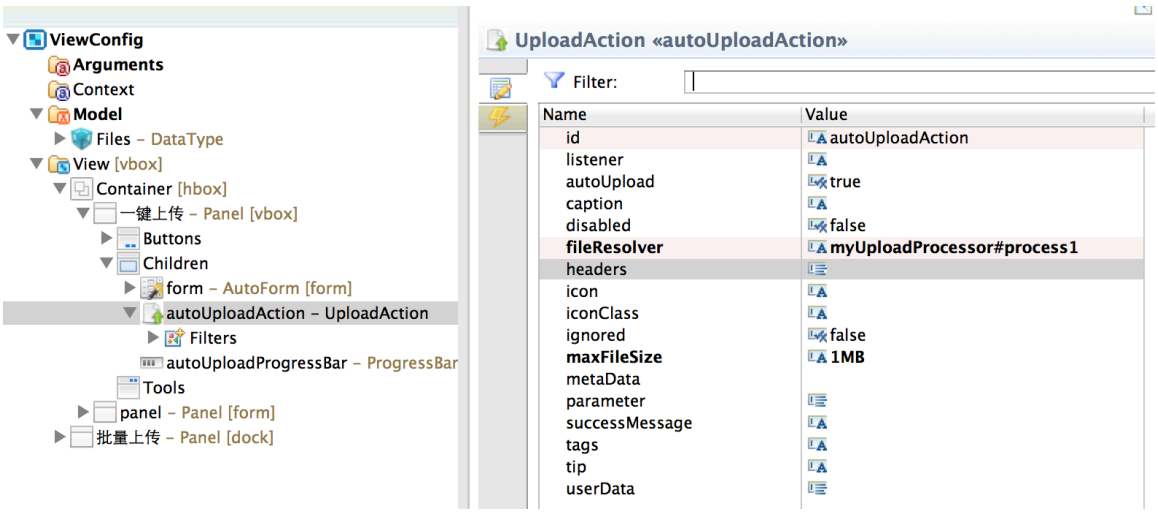
基本使用说明

UploadAction内部采用Spring的MVC机制实现文件上传，不再费笔墨详细说明了，如有兴趣请直接通过Spring的文档了解详情。我们看下UploadAction控件的基本使用技巧了。

注意
如果你用的UploadAction是1.x.x版本，则相关文档请查看：UploadAction 1.x.x

首先请了解UpdateAction或AjaxAction的基本使用技巧。UploadAction与它们基本类似。

可以直接在UploadAction中设置fileResolver，指定服务器端处理文件上传的服务类:



FileResolver的处理代码就与我们定义DataResolver方法很类似:

```
@Component
public class MyUploadProcessor {

 @FileResolver
 public String process1(UploadFile file, Map<String, Object> parameter) {
  try {
   file.transferTo(""./target/upload/"+file.getFileName());
  } catch (IllegalStateException e) {
   e.printStackTrace();
  } catch (IOException e) {
   e.printStackTrace();
  }

  return file.getFileName();
 }
}
```

其中参数是固定的:

- file参数是客户端上传到服务器端的文件对象;
- parameter参数是客户端上传文件同时附带的其他参数;

方法可以为void类型，也可以直接返回数据对象到前台，只用return一个对象就可以，最终这个对象将会被转换为JSON传到前台。

浏览器端获取FileResolver返回数据的示例代码1:

```
/** @Bind #autoUploadAction.onFileUploaded */
!function(arg) {
 var returnValue = arg.returnValue;//获取FileResolver方法返回的信息
 dorado.MessageBox.alert(returnValue);
};
```

示例代码2(JSON数据):

```
/** @Bind #autoUploadAction.onFileUploaded */
!function(arg) {
 //获取FileResolver方法返回的信息
 var info = arg.returnValue;
 dorado.MessageBox.alert("fileName : " + info.fileName + "\n" +
   "absolutePath : " + info.absolutePath);
};
```

上传参数设置

你可以通过UploadAction的beforeFileUpload事件指定文件上传附带的参数，示例代码:

```
/** @Bind #autoUploadAction.beforeFileUpload */
!function( self, form){
 //动态设置参数
 self.set("parameter", {
  param1: "value1",
  param2: "value2"
 });
};
```

FileResolver对应的Java方法中，可以通过parameter获取参数，示例代码：

```
@FileResolver
 public Map<String, String> process1(UploadFile file, Map<String, Object> parameter) throws Exception {
  String param1 = (String)parameter.get("param1");
  String param2 = (String)parameter.get("param2");
        ...
    }
```

重要属性说明

autoOpen

默认值为true，就是你一旦选择一个文件后，UploadAction会自动触发文件上传动作。如果你设置为false，则在文件选择后，你需要通过UploadAction的start()方法启动上传处理动作。示例代码：

```
action.start();
```

selectionMode

可选值为："singleFile"和"multiFiles"，改属性用于控制上传文件选择对话框中是否允许同时选择多个文件，默认值为"singleFile"

事件说明

execute()

由于浏览器的安全限定，我们服务通过脚本直接触发文件上传机制，因此UploadAction与一般的Action有些不一样，它不支持execute()方法，因此与之对应的beforeExecute()与onExecute()事件就不被UplaodAction支持。

onSuccess()和onFailure()

UploadAction支持批量文件上传，对于每一个文件来说可能都有成功和失败，因此原有的Action的onSuccess()和onFailure()事件无法描述单个文件上传的成功还是失败，在UploadAction中采用如下的几个事件来代替：

- onFileUploaded():单个文件上传成功时的事件;
- onError():单个文件上传失败时的事件;

最后当所有文件都处理完成后，会自动触发onUploadComplete()事件，包括有失败的现象