**布局：**

创建只需一个链接或者一个按钮。

显示需要一个页面，类似文档。页面上方为举报正文，下面为此举报的处理过程（回复）

功能：

对文档或者发言进行举报。

**Db:**

使用5个coll，

**Impeach**：存储举报信息，

**impeachImages**：储举报信息中的图片，

**impeachAttachments**：存储举报信息中的图片，

**impeachComments**：一个用来存储举报的对话

**impeachResult**：存储举报的处理结果

**PS：impeachImages/ impeachAttachments同时为impeach/ impeachComments服务，所以不提供冗余的外键，指明imgae/attachment对应哪个impeach/comment了。**

**Impeach**:{

Content;string required; 1~1999 举报内容

impeachImages:[ objectId] optional 举报截图

impeachAttachments: [objectId] optional 举报附件

impeachComments: [objectId] optional 举报后的对话

type: string required; enum 举报类型（文档还是评论：article/comment）

impeachedArticleId: article\_objectId optional 如果举报文档，填入文档Id

impeachedCommentId: article\_objectId optional 如果举报评论，填入评论Id

impeachedUserId: user\_objectId 被举报的文档/评论的发布者（以便页面显示给他/她看）

authorId:objectId required; 举报人

dealer: objectId optional 处理人

status：string required;enum 新建/提交/接受/分配/处理中/处理完

cDate: date required;

}

**impeachImages**:{

name:string required;1~255 举报插入的图片

hahsName:string required; 40+4 图片的hash名

authorId: objectId required; 图片插入人（冗余字段，由server决定：发起举报的人）

size: int required

pathId: objectId required 图片存储的位置

cDate: date required; 创建日期

~~uDate/dDate~~

}

**ImpeachAttachments**:{

Name: string required; 1~255

HashName: string required; 40~44

Author: objectId required;

size: int required

pathId: objectId required 图片存储的位置

cDate: date required; 创建日期

~~uDate/dDate~~

}

**ImpeachComments**:{

Content:string, required;1～999 对话内容

AuthorId:objectId required 对话发起者（举报人/处理人）

ImpeachId: objectId required 对话所属的举报

impeachImages:[ objectId] optional 举报截图

impeachAttachments: [objectId] optional 举报附件

cDate required

~~uDate/dDate~~

}

**1. 初始化文档举报(**server自动生成数据。 如果要插入图片/附件，是通过专门的rest API完成**)**

**url:** <http://domain/jb/create> POST

**无需传入参数：**

{Content:””,tpe:enum.article, Status:enum.new}

**步骤：**

~~1. 检查数据结构和值~~

2. 根据sessionId读取authorId，加入记录{Content:””,type:enum.article, Status:enum.new}，最后插入db

**2. 用户完成文档举报**（对于用户，是新建操作，添加内容/图片/附件，更改状态）

**url:** <http://domain/jb/update> POST

**传入参数：**(类似创建CR，先创建记录)

Values:{

“recInfo”:{

Content:{value:”asdf” },

Type:{value:enum.commit},

}

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 检查数据结构和值

2. 对字段content，使用DOMPurify进行XSS检测

3. 根据sessionId，读取当前用户id，和impeachId指定的记录中的authorId比较

4. 如果status存在，根据sessionId读取用户Id，判断status是否为新建/提交

**3. 客服更新举报状态：**（对于举报，客服只能修改dealer和status字段）

**url:** <http://127.0.0.1/jb/update> POST

**传入参数：**

Values:{

“recInfo”:{

dealer:{value:objectId },

status:{value:enum.xxxx},

}

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 如果dealer存在

2.1 dealer是否可用（可能使用了另外一个coll，例如admin而不是普通的user）

2.2 dealer是否有权更新此文档（是否有权处理此举报）

3. status必须存在，判断是否为有效的操作（接受/分配/处理中/处理完）

**4. 用户举报插入图片**（impeachImages）

**url:** <http://domian/jb/upload> Image POST

**传入参数：**

Values:{

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从impeach根据recId获得记录，判断用户id是否等于记录的authorId（判断用户是否有权对当前举报插入图片）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachImages。

**5. 用户举报插入附件**（impeachAttachments）

**url:** <http://domian/jb/upload> Attachment POST

**传入参数：**

Values:{

~~“curColl”:”impeach”,~~

“recId”: impeach\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从impeach根据recId获得记录，判断用户id是否等于记录的authorId（判断用户是否有权对当前举报插入附件）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachAttachments。

**6. 用户（客服） 初始化举报对话（**impeachComments，**）**

**url:** <http://domian>(127.0.0.1)/jbdh /create POST

**传入参数：**

Values:{

“recInfo”:{

impeachId:{value:objectId },

}

~~“curColl”:”impeachComments”,~~

}

**步骤：**

1. 判断传入数据格式和值

2. 根据sessionId读取用户id，然后根据impeachId读取对应记录的authorId（dealerId），判断此举报是否由当前用户创建（用户只能对自己创建的举报进行处理）

3. 将空内容和step 2中读取的用户id填入记录，并存入db。 { content:””, authorId: userId}

**7. 用户（客服）完成举报对话内容（**impeachComments**）**

**url:** <http://domian>(127.0.0.1)/jbdh /update POST

**传入参数：**

Values:{

“recInfo”:{

content:{value:objectId },

}

~~“curColl”:”impeachComments”,~~

“recId”: impeachComments\_objectId,

}

**步骤：**

1. 判断传入数据格式和值

2. 根据sessionId读取用户id，然后根据impeachCommentsId读取impeachComments中对应记录的authorId，判断此举报是否由当前用户（客服）创建（用户（客服）只能对自己创建的举报进行处理）

3. 对上传的content字段，用DOMPurify进行XSS检测

4. 上传的内容更新到coll中

**8. 用户（客服）为impeachComment添加图片：（**impeachImages**）**

**url:** <http://domain/jbdhtp/uploadImage> POST

**传入参数：**

Values:{

~~“curColl”:”impeachImages”,~~

“recId”: impeachComments\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从**impeachComments**根据recId获得记录，判断用户id是否等于记录的authorId（dealerId）（判断用户（客服）是否有权对当前**举报对话**插入附件）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachImages。

**9. 用户（客服）为impeachComment添加附件：（**impeachAttachments**）**

**url:** http://domain/jbdhtp/uploadIAttachments POST

**传入参数：**

Values:{

~~“curColl”:”impeachImages”,~~

“recId”: impeachComments\_objectId

}

**步骤：**

1. 传入参数的结构和数值检查

2. 根据sessionId获得用户id，从**impeachComments**根据recId获得记录，判断用户id是否等于记录的authorId(dealerId)（判断用户是否有权对当前**举报对话**插入附件）

3. 判断上传文化是否符合（名称/类型/大小，其他限制）

4. 获得上传文件的信息，server选取合适的path，生成{name:”aaaa”, hashName:”bbbb”, pathId:”asdf”, size:1234, authorId:”adsfas”}，写入coll: impeachAttachments。