**表单重复提交开发指南**

## 1. 增加过滤器和拦截器配置

### 过滤器配置

web.xml 需要增加过滤器配置，过滤器主要用于ajax形式提交的请求，其功能是将request对象替换为自定义的包装后的request对象。

这里需要增加的是“注1”以下的部分

<filter>

<filter-name>RepeatedlyReadFilter</filter-name>

<filter-class>com.weimob.common.web.filter.RepeatedlyReadFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>RepeatedlyReadFilter</filter-name>

<!-- 注1：这里需要增加需要拦截的url，即需要验证表单重复提交的controller中的request-mapping 路径->

<url-pattern>*/api/moduleName/functionName*</url-pattern> <!--这里需要修改成实际路径:所有ajax形式提交需要验证重复提交的路径-->

</filter-mapping>

### 拦截器的配置：

在dispatcher-servlet.xml中需要增加拦截器的配置。

举个栗子：

<mvc:interceptor>

<!--注2：这里需要增加需要拦截的url，即需要验证表单重复提交的controller中的request-mapping 路径-->  
 <mvc:mapping path="/goods/group/\*\*"/>  <!-- 请尽量使用影响小范围的通配符 -->  
 <mvc:mapping path="/mobile/\*/order/payview/\*/\*"/>  
 <mvc:mapping path="/mobile/\*/order/\*/\*/\*/pay/submit"/>  
 <bean class="com.weimob.common.web.token.RedisTokenInterceptor"/>  
</mvc:interceptor>

## 2. 增加注解

在需要生成token的方法前加下注解，如： @Token(generate = n) ,n为需要生成token的数量，如:1,2,3.....

举个栗子：

@Token(generate=1)  
@RequestMapping("/{merchantId}/{storeId}/addOrEdit/{groupId}")  
public ModelAndView editGoodsGroup(....HttpServletRequest request){  
    merchantId = super.getMerchantId(request);  
    ModelAndView mav = new ModelAndView("storeManage/dishes\_group\_ed");

    ........

}

## 3, 前端增加token变量（ajax）或隐藏域(form方式)

token key的命名规则：前缀"token\_"加上token的顺序号，如： token\_1,token\_2,......

值为request.getAttribute("token\_n")，n为需要生成token的数量，如:1,2,3.....

举个栗子：

<form ...............>

<input type="hidden" name="token" value = '$request.getAttribute("token\_1")'/>

</form>

以上方式需要在Controller中生成token，在vm中使用。

如果直接获取token，可从以下url：/api/token/acquire获取

ajax方式提交示例：

var search = {}

search["username"] = $("#username").val();

search["email"] = $("#email").val();

search['token']= token;

$.ajax({

type: "POST",

contentType: "application/json",

url: "$/api/search",

data: JSON.stringify(search),

dataType: 'json',

timeout: 100000,

success: function (data) {

console.log("SUCCESS: ", data);

},

error: function (e) {

console.log("ERROR: ", e);

}

}

## 4, 在需要删除token的地方加以下注解：@Token(remove = true)

如果是重复提交,在具体的controller中需要在方法(method)的入口处从request的get该属性并进行处理. 同时，如果业务处理成功，需要在return之前增加： TokenUtil.addRemoveTokenFlag(request);

**举个栗子：**

 @Token(remove=true)  
 @RequestMapping("{merchantId}/{storeId}/save/{groupId}")  
 @ResponseBody  
 public CommonResponse saveGoodsGroup(@PathVariable("merchantId") Long merchantId,....... HttpServletRequest request){

    CommonResponse responseJson = ......;  
    if(TokenUtil.isRepeatedSubmission(request)){  
      responseJson.setMessage("请不要重复提交");  
      return responseJson ;  
    }  
    .......

    try{

         ..................

TokenUtil.addRemoveTokenFlag(request);

     }catch(Throwable t){

          .................

    }

 }

## 示例代码：

ajax方式提交示例（spring版本不同，供参考）：

https://github.com/bjo2008cnx/spring4-mvc-ajax-example/