Freemarker那点事儿

吐槽：

讲真，来公司之前一直没有接触过Freemarker。甚至没听说过。。。恩，一定不是我太挫。。。
果然，用起来之后就感觉不太爽。
首先是调试困难（无法运行时调试），错误处理不友好（直接在页面抛出），语法不明快（竟然木有三元运算，运算符也感觉怪怪的），没有Null值校验直接报错（直接置空不挺好的吗？）。。。
最大的痛点是，处理Map和List对象的时候竟然只允许String类型的value，不然就死给你看，就像这样：

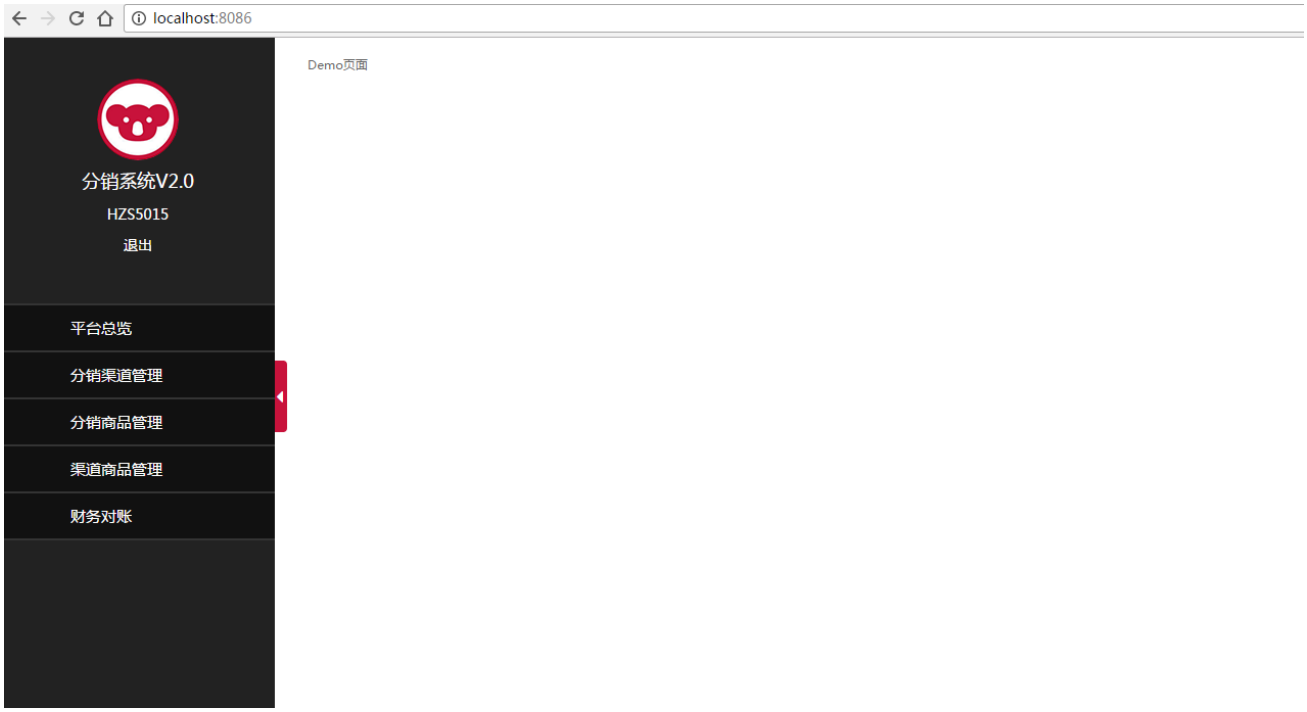
所以在模板上写逻辑真是心累，一点也不free。所以我的口号就是能异步的，咱就别marker。

正题：

话说这两天在整理“分销系统”的一些通用组件，OK，直接搞个Demo页面，再写个controller，就像这样：

```
@RequestMapping("/")
public ModelAndView demo() {
    ModelAndView mv = new ModelAndView();
    mv.addObject("currentUrl", "/demo/index");
    mv.setViewName("/pages/demo");
    return mv;
}
```

地址栏里输入localhost:port就进入了Demo， 完美（金星手）：



这次整理组件的目的就是准备在新的 “微商系统1.0”上直接复用，那就先从菜单组件入手。

之前的几个系统我都没有做过用户信息（菜单信息包含在用户信息中）这一块。只知道这些数据是从后端同步上来的。

这些信息的具体数据结构是怎样的呢？看ftl代码：

```
<#if _const_cas_user_?? && _const_cas_user_.menu ?? && _const_cas_user_.menu.childsList??>
    <#assign userInfo=_const_cas_user_>
    <#assign currentUrl=currentUrl>
</#if>
```

恩，答案已经hing清楚了。用户信息（包含菜单信息）是存在一个叫"_const_cas_user_"的对象里。

那么问题来了，刚才的controller明明只传了一个 "currentUrl"上来啊？ “_const_cas_user_”是哪里蹦出来的？

```
@RequestMapping("/")
public ModelAndView demo() {
    ModelAndView mv = new ModelAndView();
    mv.addObject("currentUrl", "/demo/index");
    mv.setViewName("/pages/demo");
    return mv;
}
```

就像刚才展示的那样，页面里的用户信息和菜单信息全部都取到了，说明值一定是取到了哇。可真的不是我传的哇，这里面一定有内鬼！

好，要搞懂这是怎么一回事，就得从后端看起。

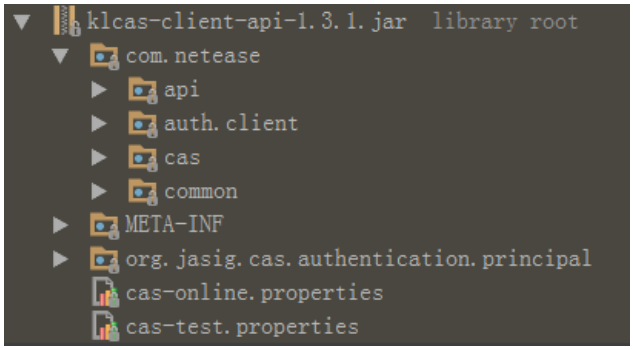
分析一个工程，第一件事就是看ta的**配置文件**，然后一点一点往上摸（邪恶脸）。。。先来看web.xml：

```
<!-- ================================CAS配置开始==================================== -->
<!-- CAS公共配置项开始 -->
<!--认证Filter，用户如果没在CAS登录的话，会将请求redirect到CAS的login接口 -->
<filter>
    <filter-name>CAS Authentication Filter</filter-name>
    <filter-class>com.netease.cas.client.filter.AuthenticationFilter</filter-class>
</filter>
```

哈哈，果然有鬼啊，原来配了一个**过滤器**，所有的请求在进入controller之前会被**一遍！不信你看：

```
<filter-mapping>
    <filter-name>CAS Authentication Filter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

CAS Authentication Filter已经被后端gg打成了jar包，所有登录鉴权和用户信息的操作都在这个包里：



好，既然知道鬼在哪，就好办了，在jar包里打开Authentication的字节码文件，然后直接搜关键字，看看搞的什么鬼。

人赃并获啊，原来“_const_cas_user_”**被塞进了request里面**。从getUser()方法来看，“_const_cas_user_”是被赋了一个**user对象，**。猜也猜到了，这个user对象里存的肯定是用户的登录信息啊。

```
else if(assertion != null) {
    if(null != assertion) {
        request.setAttribute("_const_cas_user_", assertion.getUser());
        UserData.setCurrentUser(assertion.getUser());
    }
```

可是问题还是没有得到解释，“_const_cas_user_”虽然被塞进了request，但是我并没有把request塞进Model对象啊！

回顾一遍controller，我是清白的：

```
@RequestMapping("/")
public ModelAndView demo() {
    ModelAndView mv = new ModelAndView();
    mv.addObject("currentUrl", "/demo/index");
    mv.setViewName("/pages/demo");
    return mv;
}
```

难道freemarker可以直接取request对象？要明白这个问题，那就要看看**freemarker模板上的数据模型到底是怎么定义的**：

```
protected SimpleHash buildTemplateModel(Map<String, Object> model, HttpServletRequest request, HttpServletResponse response) {
    AllHttpScopesHashModel fmModel = new AllHttpScopesHashModel(this.getObjectWrapper(), this.getServletContext(), request);
    fmModel.put("JspTaglibs", this.taglibFactory);
    fmModel.put("Application", this.servletContextHashModel);
    fmModel.put("Session", this.buildSessionModel(request, response));
    fmModel.put("Request", new HttpRequestHashModel(request, response, this.getObjectWrapper()));
    fmModel.put("RequestParameters", new HttpRequestParametersHashModel(request));
    fmModel.putAll(model);
    return fmModel;
}
```

啧啧，原来freemarker在背后上下其手，偷偷塞了这么多的数据，其中就包括了request（被包装进了一个HttpRequestHashModel对象）。

当然，**官方文档里也已经招了**：

In both templates, when you refer to `user` and `latestProduct`, it will first try to find a variable with that name that was created in the template (like `prod`; if you master JSP: a page scope attribute). If that fails, it will try to look up an attribute with that name in the `HttpServletRequest`, and if it is not there then in the `HttpSession`, and if it still doesn't find it then in the `ServletContext`. In the c

所以，你可以这么理解：freemarker在解析变量的时候，会先在模板的数据模型中查找是否定义了该变量，如果没有，就会在 HttpServletRequest、HttpSession 和 ServletContext 中继续查找。

原来如此，现在我们已经十分的接近问题的真相了： 用户每次发送请求，过滤器都会率先拦截，校验用户权限，然后把校验结果和用户信息塞进request。freemarker直接解析request（HttpRequestHashModel）对象获取信息。

可是，我为什么要说“十分的接近”呢？难道这还不是真相的全部？

之前我们发现“_const_cas_user_”指向的是一个**user**对象，根据ftl上的用法，user里面会有一个**menu**属性来存储用户的菜单信息：

```
<#if _const_cas_user_?? && _const_cas_user_.menu ?? && _const_cas_user_.menu.childsList??>
```

然而，这个bean里面并**没有定义menu属性**啊，这个menu又是哪里来的鬼啊？！：

```
public class User implements Principal, org.jasig.cas.authentication.principal.Principal {
    private static final long serialVersionUID = 1L;
    private Long userId;
    private String username;
    private Long createTime;
    private Long updateTime;
    private Long lastLoginTime;
    private Long lastLoginSystem;
    private String lastLoginSystemName;
    private String password;
    private String nickname;
    private int authFlag = 0;
    private transient CasCredentials credentials;
    private Map<String, Service> userServices = new HashMap();
    private String serverName;
    private String serviceKey;
    private String telephone;
    private String email;
    private String note;
    private String yxOpenId;
    private List<Role> roles;
    private int status;
    private String departmentName;
```

按freemarker的尿性， _const_cas_user_.menu指向null不应该直接报错了吗？但是他没有，莫非freemarker改邪归正了？

咱继续往下看，虽然这个bean里面没有定义menu属性，但是他有一个**menu的getter方法**：

```
public Menu getMenu() {
    if(this.serviceKey != null && null != this.serviceKey) {
        Service us = (Service)this.userServices.get(this.serviceKey);
        return us != null?us.getMenu():null;
    } else {
        return null;
    }
}
```

显然_const_cas_user_.menu 并没有指向menu属性，而是调用了menu 的getter方法。

为什么会这样？还是看官方文档：

TemplateHashModel functionality

Every object will be wrapped into a `TemplateHashModel` that will expose JavaBeans properties and methods of the object. This way, you can use `model.foo` in the template to invoke `obj.getFoo()` or `obj.isFoo()` methods. (Note that public fields are not visible directly; you must write a getter method for them.) Public methods are also retrievable through the hash model as template method models, therefore you can use the `model.doBar()` to invoke `object.doBar()`. More on this on discussion of method model functionality.

原来如此啊，所有的疑团都解开了。原来**user对象被封装成了TemplateHashModel对象。**

对于templateHashModel对象来说，可以通过 model.foo 来调用 model.getFoo(); 所以，menu信息并没有保存在user对象中，而是通过调用getMenu方法去获取的。

总结：freemarker真是个心机boy。鉴定完毕。

参考文档：

http://freemarker.org/docs/pgui_misc_servlet.html#pgui_misc_servlet_include

http://freemarker.org/docs/pgui_misc_beanwrapper.html