eFID系統關於導航欄的管理和控制(包含權限管控)

類AbstractDomainObject是dom4j來對xml進行讀寫的。系統的所有的菜單都是以XML的形式保存在menu的文件夾下，他們的格式大多如下：

<?xml version="1.0" encoding="UTF-8"?>

<section sid="section.am">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am</labelKey>

<action actionId="action.am">

<mode>RELOAD</mode>

<href>/C110.do?action=getList&amp;showAll=false</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected="section.am.user">

<section sid="section.am.user">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.user.management</labelKey>

<action actionId="action.am.listUser">

<mode>RELOAD</mode>

<href>/C110.do?action=getList&amp;showAll=false</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

<section sid="section.am.guest">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.guest.management</labelKey>

<action actionId="action.am.listGuest">

<mode>RELOAD</mode>

<href>/C110.do?action=getList&amp;showAll=false&amp;type=guest</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

<section sid="section.am.role">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.role.management</labelKey>

<action actionId="action.am.listRole">

<mode>RELOAD</mode>

<href>/C120.do?action=getList</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

<section sid="section.am.rate">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.exchange.management</labelKey>

<action actionId="action.am.listRate">

<mode>RELOAD</mode>

<href>/C220.do?action=getExchangeRateList</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

<section sid="section.am.logger">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.log</labelKey>

<action actionId="action.am.logger">

<mode>RELOAD</mode>

<href>/C310.do?action=getLogger</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

<section sid="section.am.kpi">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.kpi</labelKey>

<action actionId="action.am.kpi">

<mode>RELOAD</mode>

<href>/C410.do?action=query</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

<section sid="section.am.web">

<role>fid.director,fid.am.manager,fid.pm.manager</role>

<labelKey>module.am.web</labelKey>

<action actionId="action.am.web">

<mode>RELOAD</mode>

<href>/C410.do?action=queryWebSite</href>

<dwrScript/>

<jsTemplate/>

</action>

<section-list selected=""/>

</section>

</section-list>

</section>

首先，每一個菜單節點都是一個section，其中sid是這個節點的ID，role標誌了可以訪問該菜單的角色，labelKey指定了該菜單的標籤名稱，actionID標誌了要執行的對做的ID，mode指定來菜單加載的模式，href標誌來點擊菜單的時候，指向的鏈接，dwrScript應該是指定了點擊菜單要執行的dwrScript代碼，jsTemplate制定來點擊菜單後要套用的JSP模板。 section-list指定了該菜單下的子菜單，elected="section.am.user"標識當點到父菜單的時候，默認的要選中的子菜單。

生成菜單的三個類包的說明

iPLM.bean.menu.domain

iPLM.bean.menu.manager

iPLM.bean.menu.web

**iPLM.bean.menu.domain**

含有四個類，分別是MenuAction,MenuModel,SectionItem和SectionLevels。由上面xml的菜單組織形式可以看到，一個菜單類應該包含 角色、標籤等基本信息部分，action行為部分，以及sectionList子菜單部分。其中MenuAction就是action行為部分的映射類，包含href,mode,dwrscript,jstemplate,ref等信息，MenuModel是菜單視圖，是對每個菜單的映射類，其中sections標識其中含有的所有元素，也就是子菜單 ，selected用來描述被選中的菜單，SectionItem是整個一個菜單項的描述，role屬性用來描述使用的角色，labelkey是標籤，action是行為，submenus是子菜單，講一個主菜單完整的映射出來.Sectionlevels是用來對菜單的階層進行設置的。該包定義了菜單的基本元素。

**iPLM.bean.menu.manager**

用來對菜單進行管理。首先定義ICustomMenuService接口，其中衹有一個方法getCategoryMenu()用來取得菜單

AdminCustomMenuService，ContactProjectListCustomMenuService，CustomerListMenuService，KMAnalysisMenuService, NewsCategoryMenuService, PhoneBrandCustomMenuService, ProjectAgentCustomMenuService, ProjectAnnualListCustomMenuService, ProjectDetailCustomMenuService, TodayNewsCustomMenuService, XMLCustomMenuService 都是它的實現類，用來取得各種菜單。MenuDWRService 定義了一些dwr的一些函數，用來管理菜單，比如取得下一層菜單,判斷用戶是否在角色內，計算菜單，自定義菜單以及設置菜單層次等。MenuManager 是用來管理整個菜單的，其中menuResources是用來存放所有定義菜單的xml文件(其他的一些屬性以後介紹)

**iPLM.bean.menu.web**

衹有一個類，MenuController，用來和頁面進行交互，menuView定義了裝載菜單的頁面，是logo.jsp,menuService是MenuDWRService的實類，用來對菜單 進行管理。menuManager是MenuManager 的實類。

現在來看一下，菜單的生成過程。

以首頁為例：

當程序運行的時候，通過spring工廠可以生成xml中配置的人所有的bean。

來看spring.common.menu.xml中的bean定義

<bean id="menuDelegate" class="iPLM.bean.menu.web.MenuController" scope="singleton">

<property name="menuView" value="/common/logo.jsp"/>

<property name="menuManager" ref="commonMenuManager"/>

<property name="menuService" ref="commonMenuService"/>

</bean>

上面的bean通過下面的bean來生成

<bean id="commonMenuManager" class="iPLM.bean.menu.manager.MenuManager" scope="singleton">

<property name="menuResources">

<list>

<value>/WEB-INF/conf/menu/menu.xml</value>

<value>/WEB-INF/conf/menu/menu.module.about.xml</value>

<value>/WEB-INF/conf/menu/menu.module.am.xml</value>

<value>/WEB-INF/conf/menu/menu.module.mypage.xml</value>

<value>/WEB-INF/conf/menu/menu.module.pm.xml</value>

<value>/WEB-INF/conf/menu/menu.module.km.xml</value>

</list>

</property>

</bean>

加載所有定義菜單的xml文件，來生成bean

<bean id="commonMenuService" class="iPLM.bean.menu.manager.MenuDWRService" scope="singleton">

<property name="customMenuService">

<map>

<entry key="section.pm.list"><ref bean="pmListCustomMenuService"/></entry>

<entry key="section.pm.project"><ref bean="pmProjectCustomMenuService"/></entry>

<entry key="section.pm.workLoading"><ref bean="pmListCustomMenuService"/></entry>

<entry key="section.profile.addresses"><ref bean="addressCustomerMenuService"/></entry>

<entry key="section.am.user"><ref bean="adminMenuService"/></entry>

<entry key="section.km.todayNews"><ref bean="todayNewsMenuService"/></entry>

<entry key="section.km.newsCategory"><ref bean="newsCategoryMenuService"/></entry>

<entry key="section.km.productDatabase"><ref bean="mobilePhoneMenuService"/></entry>

<entry key="section.km.analysisWizard"><ref bean="kmAnalysisMenuService"/></entry>

<entry key="section.pm.projectagent"><ref bean="pmListCustomMenuService"/></entry>

</map>

</property>

</bean>

定義Map，來配置每一種菜單的生成類

菜單生成的流程

1. <prop key="/index.do">index</prop>

當打開首頁的時候，會自動執行/index.do, 從而去執行index這個bean，這個bean導向index.jsp頁面

<bean id="index"

class="org.springframework.web.servlet.mvc.ParameterizableViewController" scope="singleton">

<property name="viewName" value="/index.jsp"/>

</bean>

所以當打開首頁的時候，會自動的跳轉到index.jsp頁面.

2.跳到index頁面後，首先執行前面的java代碼(其實是servlet代碼)，獲得名為MENUS\_SELECTED的session，並將其轉為SectionLevels類的對象(用來保存菜單的階層數)，因為是第一次進入頁面，所以為空。

所以redirectUrl也為空，這樣就會自動的去包含aboutUs.do頁面，也就是我們看到的，每次登陸，首頁總會顯示到“關於我們”的介面。

3.再來看導航欄的生成

(綠色部分為不太理解部分)從index頁面直接執行logo.do,然後根據spring.common.mapping.xml 中的配置<prop key="/common/logo.do">menuViewController</prop>從而去執行menuViewController bean。

<bean id="menuViewController"

class="org.springframework.web.servlet.mvc.multiaction.MultiActionController" scope="singleton">

<property name="methodNameResolver" ref="parameterMethodResolver"/>

<property name="delegate" ref="menuDelegate"/>

</bean>

這個bean有倆個屬性，一個是methodNameResolver,定義了解決問題的方法，我們來看一下parameterMethodResolver 該bean在spring .common.mapping.xml中

<bean id="parameterMethodResolver"

class="org.springframework.web.servlet.mvc.multiaction.ParameterMethodNameResolver" scope="singleton">

<property name="paramName" value="action"/>

<property name="defaultMethodName" value="index"/>

</bean>

parameterMethodResolver定義了參數名為action,默認的方法名為index

(現在還不明白action到底有什麼用處，為什麼會定義參數名為action)

這樣，系統就會自動的到menuDelegate中的index方法。

4.執行了index方法之後，會返回logo頁面，用來顯示導航欄，同時會返回一個MenuDWRService.*MENUS\_SELECTED，*是一個SectionLevelsde的對象，其中記錄了location,level等信息，還有menus，也就是需要顯示的菜單內容。在執行loadMenus()

的時候，已經講menus中的list sections中的元素從AbstractDomainObject轉為

SectionItem

5.從而轉向logo介面。這樣我們就可以在logo.jsp頁面動態的將導航欄顯示出來。注意，第二層菜單衹有當第一層菜單選中的時候才顯示。