一种角色权限管理方案的算法设计①

刘鹏远 (湖北经济学院 计算机学院 湖北 武汉 430205)

摘　要： 基于角色的权限管理在信息管理系统的开发中得到普及应用，应用角色权限管理技术可实现不同身份登录验证后具有不同的操作界面，提高了系统的易用性和健壮性。基于作者开发的一个项目，分权限菜单处理、修改角色数据预准备、角色展现页面逻辑、以及客户端复杂逻辑控制多个方面详尽给出一系列核心算法实现。应用表明这些核心算法有着较好的处理性能。

关键词： 角色;权限;功能点;菜单;算法

Algorithm Design of a Role-Based Privilege Management Schema

LIU Peng-Yuan (School of Computer, Hubei University of Economy, Wuhan 430205, China)

Abstract: Nowadays, Role-based Privilege Management has been generally applied in the Management Information System. Using MIS can give different operation interfaces with different identification user logins which thus makes the system easy to use and improves its robustness. On the basis of one provincial project the author helds, several key algorithms around 4 parts are designed, which include privilege-menu processing , data preparation for updating role, web page presentation logic, and client-user complicated login control. Result of practical usage shows that they perform well.

Keywords: role; privilege; function point; menu; algorithm

1　引言

角色权限管理给 MIS 系统带来更好的交互性和健壮性，丰富了业务逻辑控制细粒度，提高了应对新业务需求变更的灵活性。但一些较常使用的如 struts menu[1]，acegi 等框架具有无法重用、接口复杂、变更复杂的缺点。本文将给出一不依赖框架的角色权限管理方案的主要算法设计实现。图 1 给出了一个角色权限管理框架的数据库表设计。

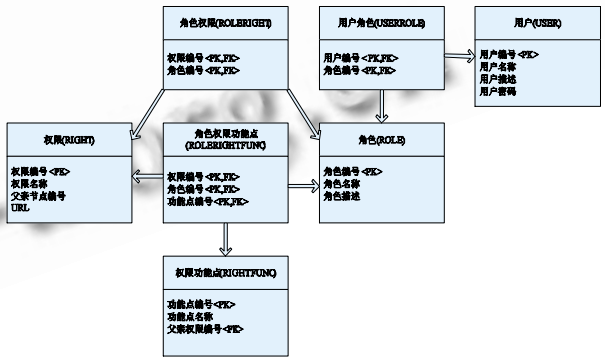

图 1　角色权限管理数据库表设计[2]

2　权限菜单处理

许多 JS 组件应用于在客户端浏览器中绘制菜单，常用的有 DTREE 和 COOLMENU[1]。不论何种 JS 组件，都要求必须传入一个深度优先序的菜单字符串交给 JS 组件，浏览器才能正确展现具有层次序的菜单。

getDFSMenuNo 算法[2]将数据库表中读得的权限集合调整为深度优先序保存到 vOutRight 数组。为保证该递归算法能正常结束，需首先虚构一个根权限节点预加到权限数组 vRight，然后经过多表的连接运算将某用户的权限集合结果集装载入 vRight。详尽处理流程参见文献[2]。

调整权限数组为深度优先序后，vOutRight 做好

① 基金项目:湖北省教育厅重点项目(B200619001)
收稿时间:2010-01-26;收到修改稿时间:2010-03-19

了展现现前的数据准备，但还涉及以下算法需实现：在新建/修改/查看角色时需要按深度优先序展现出系统定义的所有权限和功能点，并将该角色具有的权限和功能点对应的 checkbox 框置初始状态；页面展现后，处理客户端对各 checkbox 的选中和去除选中动作；点击提交时，对选中 checkbox 的对应权限和功能点相应保存到数据库表，对没有选中的更新数据库表。限于篇幅，本文主要对展现数据预准备和页面展现逻辑进行阐述。

3 修改角色 – 页面展现预准备

本节以修改角色为例给出算法实现流程。用户登录系统时首先适配展现其具有权限的对应菜单，具有角色管理权限时才可能可见该菜单项点击从而进入角色管理页面。而在角色管理页面，当具有修改角色功能点时才可见到该修改角色按钮。假设上述条件都具备，则当点击修改角色按钮后，处理流程如下：

(1) 读出权限表和功能点表的所有记录分别入 vRight 和 vRightFunc 保存，读出该角色具有的功能点数组入 vRealRightFunc 保存。

(2) 对 vRight 和 vRightFunc 继续处理，做好修改或增加角色前的数据准备，完成深度递归调整权限、给权限加上祖先权限前缀路径，以及给功能点加上祖先权限前缀路径的任务。处理的结果分别保存在 vOutRightFlag 和 hvOutRightFuncFlag 中。处理流程如下：

1) 调用第 2 节的 getDFSMenuNo 调整传入的 vRight 权限数组为深度优先序，并加上祖先权限前缀保存入 vOutRightFlag；

2) 遍历访问 vOutRightFlag(已调整为深度优先序并加上全祖先权限路径)，对每条权限记录 tempObj1 做如下处理：

a 置初始 changed=false；

b 取 tempObj1 的权限编号 str1(已带祖先权限前缀)的最后一段”-”后子串，定义为 tempStr1(即最后一级权限编号)。

c 遍历访问功能点表 vRightFunc，对每条功能点记录 tempObj2 的 menuNo 字段定义为 str2。比较 str2 与 tempStr1，若相等做如下处理：

ⅰ置布尔变量 changed 为真(后边处理权限数组时用)；

ⅱ拼接 str1 和 tempObj2 的 Functionno 字段串得到该功能点的带祖先权限前缀的全路径，保存修改后的功能点记录为 tempObj3；

ⅲ若 vRealRightFunc 不为 null(表明不是新建角色而是查看/修改已有角色，需要对该角色已具有的权限和功能点置初始选中状态从而方便后继展现)。遍历访问 vRealRightFunc，在该角色实际具有的功能点记录数组中查找是否具有该功能点，将 vRealRightFunc 记录的功能点编号与 tempObj2 的功能点编号比较，若相同则说明该角色具有该功能点，继续如下处理：

① 置功能点记录 tempObj3 的状态位为选中；

② 不重复拼接 str1 串到 rightInStr(记录下所有需要置选中的祖先权限路径，以”;”间隔，方便后边一次性对祖先权限路径上的所有权限置初始选中用)；

③ break 退出本层循环(表示一条功能点记录对于该角色的初始状态处理完毕)。

ⅳ将这样处理过后得到的 tempObj3 加入 vTempRightFuncFlag 数组保存；

ⅴ重复(1)-(4)处理直至 vRightFunc 遍历完毕。

d 若 changed 不变仍为 false，表示该权限为非叶子权限(下一级仍为权限，是中间节点)，给 vTempRightFuncFlag 加上一个空行对象 nullRightFuncFlagObj(这类权限仅放展现页面左侧做缩进用，不在右侧功能点排列中出现)，空行对象的功能点名称字段属性被定义为值”empty”；

e 将 vTempRightFuncFlag 装入 hvOutRightFuncFlag 数组；

f 重复 a)—d)过程处理直至 vOutRightFlag 遍历完毕。

3) 继续处理前面已用 rightInStr 记录的所有需要置选中的祖先权限。遍历 vOutRightFlag，对其每条权限记录 rightFlag 取其 menuNo 定义为 tempStr2(含祖先权限前缀的全路径)，若 rightInStr 含有该串，则对应置该 rightFlag 状态位为初始选中。

(3) 将经过上面处理完毕的 vOutRightFlag 和 hvOutRightFuncFlag 两个数组对象写入浏览器 request 对象的 session 中，转向 addRole 或 viewRole 页面展现全部的权限和功能点。

4 角色展现页面

第 3 节处理已在后台准备好了待展现的角色、权

限以及功能点数据，下面以 JSP 中最常使用的 STRUTS1 标签[3]为例描述展现角色的过程。这里仍然有着复杂的 scriptlet 小算法用于控制权限、子权限、对应功能点的问题。角色展现页面的逻辑处理流程如下：

(1) <tr>标签内以<logic:iterate>标签对浏览器 request 对象 session 内 vOutRightFlag 数组进行迭代，对其每个权限记录命名为 rightFlag 并做如下处理：

1) 一个<td>开始：

a. 取 session 内写入的 vOutRightFlag 数组，以 index 递增整数为下标方式取得一个权限记录；

b. 截取权限记录的 menuNo 字段串，分割为以”-”间隔的多个字符串形成字符串数组，数组的长度是权限记录全路径的长度(”-”的个数)，为左对齐做好准备；

c. 连续 b)中数组长度个空格，这样就确保了不同层次的权限记录有着不同的左缩进；

d. 以 <input type="checkbox"> 开始一个 checkbox 的输出，其 name 属性值是 rightFlag 权限记录的 menuNo 字段值(权限编号)；接着以<logic:equal>标签判断该 rightFlag 权限对象状态是否为选中，若选中则置该 checkbox 的 checked 属性为选中；定义 onclick 事件相应指向 remain 函数处理，传入当前点击的 checkbox 标签。如此完成一个权限编号的显示；

e. 显示该 rightFlag 权限记录的 menuName 字段值(权限名称)；

f. 以 <input type="hidden" name="checkRightFunc" value="<bean:write name='rightFlag' property='menuno'/>"> 对上面的一个 checkbox 定义以一一对应的 hidden 标签，注意该 hidden 标签的 value 属性必须与 rightFlag 权限记录的 menuNo 字段串相同；而 name 属性必须全部相同，这里定义为”checkRightFunc”。

2) </td>结束，一个新的<td>开始，完成功能点数组的一行排列展现：

a. 以<logic:iterate>标签对浏览器 request 对象 session 内 hvOutRightFuncFlag 数组进行迭代，取 1)a)中 index 整数偏移量为下标的元素命名为 vRightFuncFlag（功能点数组）并做如下处理：

ⅰ连续 1)c)中的空格数个空格(相当于整行平移)；

ⅱ以<logic:iterate>标签对 vRightFuncFlag 功能点数组进行迭代，对其每个元素命名为 rightFuncFlag 功能点对象并做如下处理：

b. 以 <logic:notEqual> 标签判断 rightFuncFlag 功能点对象的功能点名称字段是否为值”empty”，若不则说明不是空行对象，即需要输出对应的一行功能点(见 3 中(2)2)d)部分)：以<input type="checkbox">开始一个 checkbox 的输出，其 name 属性值是 rightFuncFlag 功能点对象的 functionNo 字段值(功能点编号)；接着以<logic:equal>标签判断该 rightFuncFlag 功能点对象状态是否为选中，若选中则置该 checkbox 的 checked 属性为选中；定义 onclick 事件相应指向 remain 函数处理，传入当前点击的 checkbox 标签，如此完成一个功能点编号的显示；接着显示该 rightFuncFlag 权限记录的 functionName 字段值(功能点名称)；

c. 重复 a)直至 vRightFuncFlag 遍历完毕。

(3)重复(1)-(2)处理直至 hvOutRightFuncFlag 遍历完毕。

3) </td>结束，完成功能点数组一行的排列显示(对应左侧的一个叶子权限)；

4) </tr>结束，完成一行权限和功能点的排列显示。

5) 重复 1)-4）直至 vOutRightFlag 遍历完毕。

5 浏览器客户端控制逻辑

上节给出了展现角色权限功能点的页面辑，此时还需定义客户端动作的算法，使用 java script 语言实现。另外，还需跟踪用户选中情况，为提交后存储该角色的权限和功能点做好准备。为记录用户选中了哪些 checkbox，一个技巧是除”全选”控制外，对每个 checkbox 框定义一个 hidden 元素——其 value 属性值就是 checkbox 框的 name 属性值(实际源于第 3 节中写入到浏览器 session 的对象)，所有 hidden 元素 name 属性相同，这里定义为”checkRightFunc”(所有 hidden 元素对应的 checkbox 必须同 name 值)。

对某角色具有的权限或功能点，注意其 checkbox 以及所有祖先的初始选中状态。方法是对权限数组保存的权限对象和功能点数组保存的功能点

对象各增加字段 checkFlag 用于标识是否选中；在展现页面前对 checkFlag 赋未选中初值然后将权限对象数组和功能点对象数组保存到 request；展现页面时取 request，迭代取两个数组的元素并置 checkbox 初始状态。对选中和去除选中的控制需编写脚本将递归与遍历结合，这样要求对 checkbox 的 name 做层次命名安排，可将 name 定义为带上祖先权限编号前缀(多级祖先前缀以”-”间隔)的对应权限编号或功能点编号。

角色权限功能点表只保存选中的叶子级权限及对应的功能点，而角色权限表要保存选中的权限以及它的所有祖先权限，因此对点击确定后的处理，需要区分权限和功能点。为此对 checkbox 的 name 属性值加上约定——功能点编号 checkbox 的 name 值以”/”结尾表示是功能点。对于如何识别选中的 checkbox 这里巧妙地使用了许多同 name 的 hidden 字段来跟踪对应的 checkbox 的选中状态，以方便后继的保存和修改处理。

(1)提交检查 sendChecked

该函数检查处理未选中的 checkbox，将对应 hidden 同步标识为””，再转 saveRole 后台 action 继续处理。

1) 遍历表单的所有页面元素，对当前元素 eachhidden 做如下处理：

a. 若 eachhidden 的 type 是”hidden”，那么以其 value 值做 id 或 name 的元素就是对应 checkbox 页面元素。使用 DOM 的 getElementById 方法抓取该 checkbox 元素(命名为变量 eachcheck)；

b. 若 eachcheck 的 checked 属性值为”false”(未选中)，将 eachhidden 的 value 取””。

c. 继续 a)-b)直至循环处理完毕当前 form 的所有元素。

2) 重定向到保存角色页面的后台一步处理

当定义好某角色的权限和功能点点击提交后，经过上述处理，浏览器 request 对象中 name 为 checkRightFunc 的变量得到了所有选中的权限和功能点(是一个字符串，未选中的已经被 sendchecked 函数去除了)，后台可从 request.getparameters(“checkRightFunc”)获得客户端配置的角色定义，完成后继处理。

(2)全选控制 selectAll

该函数传入事件源 checkbox 变量 e(即”全选”checkbox)，将所有 checkbox 置为与其相同的状态，即”全部选中”或”全部取消”。处理方法是遍历表单的所有 checkbox 元素，将每个元素的 checked 属性赋值为 e.checked 属性值。

(3)向上处理 upSelect

该函数传入事件源 checkbox 变量 e(即点击的那个 checkbox)，不断截取事件源的前缀，从而向上处理事件源 checkbox 选中时的所有祖先(均选中)。如果事件源是取消选中动作，转 upSelect2 交给事件源的父节点处理(见图 2，当取消选中时，祖先是否选中还与当前兄弟的选中状态有关)。

1) 取事件源 name 值的前缀(去掉最后一个”-”子串)，只要不空，使用 DOM 的 getElementById 方法抓取该 checkbox 元素(即直接父亲)t；

2) 若事件源 e 是取消选中动作，传入 t 调用 deterSelfAbove 继续处理，否则转 3)；

3) 置 t 选中并用 t 代换 e 递归调用 upSelect。

(4)根据儿子状态确定自身状态 deterSelfAbove

该函数传入一个 checkbox 元素 e，用于确定 e 的该置选中状态与否。显然，判断选中与否只需读取其所有儿子的状态，若全部儿子都未选中则置未选中并继续激发递归向上动作。

1) 预设控制变量 nonSelected=true；

2) 遍历表单所有 checkbox 元素，对每个 checkbox 元素 each 做如下处理：

a) 取 e 长度对 each.name 求前向的该长度子串；

b) 若该子串与 e.name 相同(说明 each 为传入 e 元素的儿子)继续如下处理：

ⅰ.若 e 为未选中继续 a)-b)处理直至表单遍历完毕；

ⅱ.否则置 nonSelected=false 并退出循环；

3) 若 nonSelected=true，表明所有儿子均未选中，置 e 未选中。并用 e 传入调用 upSelect，即向上处理该元素因改变为未选中状态引发的级联效应。

(5)用户选择控制

该函数传入事件源 checkbox 变量 e(点击发生处的 checkbox)，处理除全选控制外的所有 checkbox 框控制。处理顺序是先遍历处理置自己子孙 checkbox 元素的状态，再调用 Item 函数向上递归处

理置当前节点 e 的所有祖先 checkbox 元素的状态，即先往下，再往上的处理顺序。最后要注意置”全选”控制 checkbox 的状态。注意到各 checkbox 代表的权限和功能点 name 属性值来自于 2.1 处理后的 session 对象值，都带有祖先权限前缀，因此可借用该特征迅速匹配到当前事件源的子孙元素。函数处理过程如下：

1) 求得 e.name.length，遍历当前 form，对当前元素 each 的 name 值字符串取自 0 开始长为 length 的子串，并与 e.name 比较。若相同，则说明该 each 页面元素是事件源元素的子孙元素，对其置 checked 属性为 e.checked 属性相同的值。

2) 传入 e 调用 upSelect，向上处理祖先 checkbox 状态，过程见(3)

3) 预设控制变量 allSelected=true；

4) 遍历表单所有元素，对每个元素 each 做如下处理：

a 若 each 是”全选”checkbox，continue 循环；

b 若 each 是未选中，置 allSelected=false；break 退出循环；

5) 若 allSelected=true，表明都选中，置全选控制 checkbox 为选中。

6 小结

角色权限控制涉及算法较多，算法异常复杂，限于篇幅本文主要对新增角色时页面角色展现数据预准备算法，以及页面展现后客户的复杂动作逻辑算法给出完整实现。其它复杂算法有待后继发表，有兴趣读者可与本文作者联系。

参考文献

1 飞思科技产品研发中心, JSP 应用开发详解. 第 2 版, 北京:电子工业出版社, 2006.255－270.

2 刘鹏远.基于角色的权限管理可重用解决方案.计算机系统应用, 2007,16(6):28－31.

3 孙卫琴.精通 STRUTS.北京:电子工业出版, 2005.95－102.

www.c-s-a.org.cn